ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

DEPARTAMENTO DE ENGENHARIA DE ENERGIA E AUTOMAÇÃO ELÉTRICAS

VICTOR SERPA CARVALHO NOGUEIRA

VICTOR WOLF

DESENVOLVIMENTO DE SIMULAÇÃO BASEADA EM HIL (HARDWARE-IN-THE-LOOP) PARA LABORATÓRIO DIDÁTICO DE AUTOMAÇÃO

SÃO PAULO

2014

VICTOR SERPA CARVALHO NOGUEIRA

VICTOR WOLF

DESENVOLVIMENTO DE SIMULAÇÃO BASEADA EM HIL (HARDWARE-IN-THE-LOOP) PARA LABORATÓRIO DIDÁTICO DE AUTOMAÇÃO

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia.

Orientador: Prof. Dr. Eduardo Cesar Senger

Coordenador: Prof. Dr. Lourenço Matakas Jr.

SÃO PAULO

2014

RESUMO

O tema proposto pelo Prof. Dr. Eduardo Cesar Senger teve como plano desenvolver um novo modelo de simulação para o laboratório didático de automação. Através da simulação de um reator químico em tempo real, é esperado ensinar o aluno num ambiente factível, tornando o aprendizado concreto. Este trabalho mostra o desenvolvimento de uma simulação baseada em *Hardware-In-The-Loop* e procura compreender vantagens e limitações do modelo, dos *hardwares* e *softwares* utilizados. Também tem como objetivo servir de guia e exemplo para futuras aplicações desenvolvidas em arquiteturas semelhantes.

Palavras-chave: Automação. Simulação. *Hardware-In-the-Loop. Real-Time.*

ABSTRACT

The theme proposed by Prof. Ph.D. Eduardo Cesar Senger had as a plan to design a new model of simulation for the automation didactic laboratory. Through the simulation of a chemical reactor in real time, it is expected to teach the student in a feasible environment, making the act of learning a concrete experience. This project displays the development of a Hardware-In-The-Loop based simulation and seeks to understand the advantages and limitations of the chosen model, hardware and software. Also has as the purpose to serve as a guide and be an example to future applications developed in similar architectures.

Keywords: Automation. Simulation. Hardware-In-The-Loop. Real-Time.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A engenharia de automação é uma área de fundamental importância no panorama dos mais diversos tipos de indústrias, fábricas, plantas de geração de energia, aeroportos e diversos outros locais onde se precisa garantir não somente produção com eficiência, dinamismo e precisão, mas também a segurança dos usuários, operadores e funcionários. Os equipamentos necessários para que se consiga realizar uma instalação com confiabilidade e padronização por normas são invariavelmente caros e de tecnologia bastante avançada. Em poucas universidades do mundo alunos possuem um contato satisfatório com tais equipamentos e experimentos que traduzam o dia a dia de um engenheiro de automação de forma próxima da realidade. É muito raro também que as universidades estejam próximas a situações reais de aplicação, tornando necessário que se criem alternativas para capacitar seus alunos no ambiente acadêmico.

O desenvolvimento das tecnologias computacionais nos trouxe muitos avanços, sendo que um dos principais foi a capacidade de simular situações reais em sistemas robustos de computação, podendo prever seus resultados de forma próxima do esperado na realidade a um preço relativamente baixo se comparado à construção da aplicação em si. Um exemplo disso são aplicações que utilizam como base o conceito de *Hardware-In-the-Loop* (HIL) para simular em tempo real situações para prever a viabilidade de projetos complexos e algumas vezes perigosos. Simular nos possibilitou evitar danos e perdas e aumentou nosso poder de criatividade. Encontrou-se em fim a resposta de como aproximar o estudante de situações reais e colocá-lo de forma segura no meio do mundo da engenharia de automação. Com a simulação tem-se a possibilidade de errar e tentar novamente sem causar danos nem ferir a alguém, algo de grande probabilidade de ocorrência quando se é inexperiente tem-se controle de algo de grandes dimensões em mãos.

Atualmente temos, na Escola Politécnica da Universidade de São Paulo, o curso de graduação em Engenharia Elétrica com ênfase em Energia e Automação Elétricas. Dentre as disciplinas ministradas no âmbito de automação, temos aquela chamada "Laboratório de Automação de Sistemas Elétricos", posicionada no 10º semestre do curso. Essa disciplina conta com diversas experiências incluindo a programação de CLPs (Controladores Lógicos Programáveis) em *Ladder, SFP* e

outras linguagens para controlar simples situações que exemplificam a automação em alguns casos reais. Em algumas delas a simulação que se vê necessária é executada no próprio CLP junto ao código que deve ser escrito pelo aluno.

Um laboratório didático possui como meta a habituação do aluno aos equipamentos, aplicações e sistemas utilizados no seu campo de atuação. A aproximação do aluno com o objeto de estudo é dificultada em razão das dimensões, dos custos e da complexidade dos equipamentos. Através do desenvolvimento de simulações em tempo real, esperamos criar um ambiente de estudo mais factível e aproximar o que é estudado no curso de engenharia com a realidade.

2 OBJETIVOS

O objetivo deste trabalho é fornecer ferramentas através de simulação com HIL para que as experiências ministradas neste laboratório sejam mais diversificadas, complexas e concretas, possibilitando um aprendizado mais dinâmico aos alunos daqui para frente.

Além disso, busca-se deixar um guia para o desenvolvimento de projetos em arquiteturas semelhantes, acentuando vantagens e desvantagens dos equipamentos e programas utilizados, pontificando cada etapa e desafio do trabalho. Procuramos fundamentar os trabalhos que virão a seguir, servir como alicerce de uma nova aproximação das experiências dadas no curso de engenharia.

3 METODOLOGIA

A técnica utilizada para a simulação, como apontado nas seções anteriores, é o *Hardware-In-The-Loop*, detalhada a seguir. Outro ponto do trabalho são os *softwares* e *hardwares* utilizados. As peças fundamentais do plano traçado para o funcionamento da simulação são dois *softwares*: LabVIEW da *National Instruments* e MATLAB da *MathWorks* e um *hardware*: CompactRIO, também da *National Instruments*. A escolha de tais ferramentas será fundamentada a seguir, ao tratarmos individualmente de cada uma.

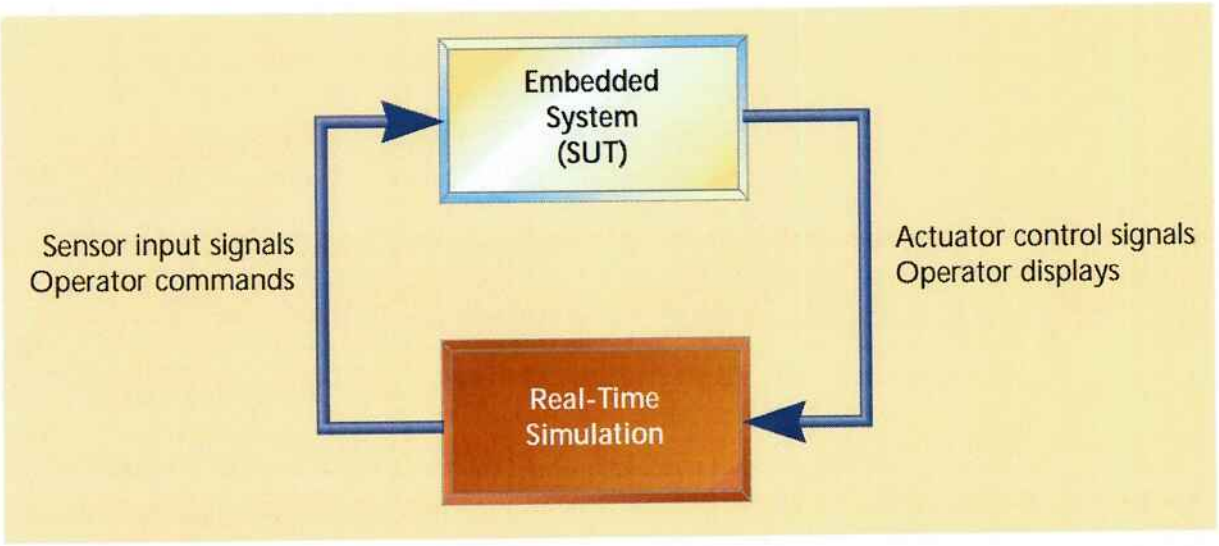
3.1 Hardware-In-the-Loop

O *Hardware-In-the-Loop*, HIL, é uma técnica de simulação em tempo real utilizada para o desenvolvimento e teste de sistemas de controle de sistemas multifacetados; a simulação desenvolvida substitui um sistema complexo ou uma máquina, habilitando o controlador a receber respostas verossímeis. Esse tipo de aproximação possui diversas vantagens que outros métodos de análise não possuem. Ela habilita que um *hardware* seja testado e estudado repetidamente sem nenhum tipo de ônus à máquina. Minimiza custos e remove os riscos do estudo de condições de alto perigo que o *hardware* pode encontrar. Proporciona condições reais de teste muito antes do sistema real estar construído ou funcionando, aumentando a chance de qualquer defeito ou erro no projeto ser encontrado antes de qualquer perda. Portanto, o potencial e o objetivo da simulação HIL é claro: mostrar todos os aspectos e revelar cada particularidade do funcionamento de um sistema.

A importância de um modelo preciso é óbvia; sem uma simulação precisa, o controlador - no caso desse projeto - não será testado verdadeiramente. A complexidade do processo é traduzida através de modelos matemáticos de todos os componentes relacionados ao sistema. A junção de todos esses modelos é chamada de *plant simulation*. De forma simples, a simulação monitora as saídas do sistema a ser testado (SUT – *System Under Test* ou HUT – *Hardware Under Test*)

(LEDIN, 1999) e injeta sinais da mesma forma que o sistema real. Uma visão superficial de uma simulação HIL está demonstrada na figura a seguir (LEDIN,1999).

Figura 1 - Exemplo de uma simulação HIL



Fonte: Ledin (1999)

Sistemas HIL oferecem uma plataforma eficiente para todo tipo de aplicação. A técnica é utilizada para o teste de satélites, veículos, aeronaves, navios, motores, sistemas de potência e muitos outros. A limitação é apenas a representação matemática.

3.2 MATLAB e Simulink

O procedimento para a simulação *HIL* possui três etapas (HALVORSEN, 2011): desenvolvimento de um modelo matemático, a simulação HIL (teste do *hardware* na simulação) e a implementação do *hardware* no ambiente que ele será utilizado.

A etapa de desenvolvimento do modelo matemático pode, dependendo do assunto a ser tratado na simulação, ser extremamente demorada e complicada, fugindo completamente do objetivo deste projeto que tenta simplificar os passos para uma simulação de qualidade nos laboratórios. Podemos, no entanto, obter as

equações e diagramas de blocos que necessitamos com a ferramenta *Simulink* do *software MATLAB*.

O Simulink é um ambiente de diagrama de blocos para projetos desenvolvidos a partir de modelos. Ele habilita o trabalho em diversos níveis, a partir de subsistemas que podem ser desenvolvidos separadamente da aplicação principal. Além disso, o programa possui um editor gráfico para o gerenciamento hierárquico dos blocos.

Por ser um *software* aceito e de amplo uso por vários ramos da engenharia, possui uma vasta biblioteca de blocos, totalmente customizáveis, já preparados para a simulação de diversos tipos de sistemas. Tais bibliotecas multiplicam as opções disponíveis e foram determinantes na escolha deste *software* como ponto de partida de nosso projeto.

3.3 CompactRIO

O controlador CompactRIO é um sistema reconfigurável embarcado de controle e aquisição. O processador e o FPGA reconfigurável habilitam o controlador a diferentes tipos de aplicações. O processador é utilizado para controle, processamento, *data logging* e comunicação em rede. O FPGA programável habilita o usuário a implementar *hardwares* customizados para aplicações de controle em alta velocidade, processamento de dados *in-line* ou funções complexas de temporização.

O controlador possui módulos acopláveis de entrada e saída tanto analógicas como digitais e permite a realização de cálculos e processamento de dados em tempo real, com o sistema operacional LabVIEW Real-Time, se utilizando dos sinais de entrada e fornecendo a partir disso as respectivas saídas. Estes sistemas foram adquiridos recentemente pela Escola Politécnica e ficaram à disposição dos professores para melhorias no laboratório. O CompactRIO fornece uma arquitetura barata para a implementação da técnica de simulação *Hardware-in-the-loop* e, portanto, a aplicação necessária no trabalho.

O modelo utilizado no projeto é o NI cRIO-9024. O controlador de tempo real possui um processador de tempo real *Freescale* de 800 MHz, 512MB de RAM DDR2

e 4GB de armazenamento não volátil para programas e *data logging*. Também são oferecidas duas portas Ethernet – 10/100 e 10/100/1000 – utilizadas para a comunicação com o *software* ou internet. Além disso, utilizamos quatro módulos de entrada e saída digital. São eles: NI9205, para entradas analógicas, NI9421, para entradas digitais, NI9264, para saídas analógicas e NI9472 para saídas analógicas.

A seguir, são mostradas fotos do equipamento utilizado.

Figura 2 - Controlador NI cRIO-9024 e fonte

Fonte: Autores

Figura 3 - Controlador NI cRIO com o módulo de encaixe dos cartões

Fonte: Autores

Figura 4 - Controlador com os cartões de entrada e saída

Fonte: Autores

Por ser um produto da *National Instruments*, o CompactRIO depende do *software* LabVIEW, ferramenta de programação gráfica da empresa. É através do LabVIEW que ocorre a programação do CompactRIO pelo computador. Temos então as duas pontas da metodologia apresentadas, resta o laço entre elas para entender como irá ocorrer a integração.

3.4 LabVIEW

O LabVIEW é o *software* base do sistema de projeto da *National Instruments* (NATIONAL INSTRUMENTS, 2009). O *software* é uma plataforma gráfica que auxilia no teste de sistemas de pequeno e grande porte. Além disso, oferece integração entre diversos *hardwares* da própria empresa, o que aumenta o poder da ferramenta.

Como citado nos tópicos anteriores, o LabVIEW é a ferramenta necessária para programar o CompactRIO. Logo, para nos utilizarmos do *Simulink* para desenhar a simulação, é necessário um meio de conversão deste arquivo salvo no MATLAB para o LabVIEW. A solução surgiu recentemente, com um módulo adicional (*add-on)* para o LabVIEW lançado pela *NI* chamado "*Control Design and Simulation*". Este módulo tem, dentre outras funcionalidades, a ferramenta de conversão de arquivos ".mdl", o formato de arquivos de simulação salvos pelo Simulink, para o espaço de trabalho do LabVIEW.

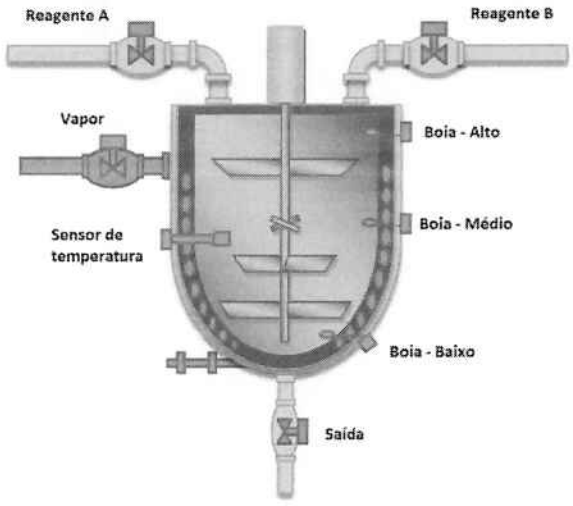
Além de simplesmente servir como ponte entre MATLAB e CompactRIO, o LabVIEW será o responsável por comandar a simulação a partir de um computador, comunicando-se com o CompactRIO através de sua saída Ethernet e apresentar a interface gráfica da planta de simulação na tela, em tempo real.

4 DESENVOLVIMENTO

4.1 Modelo do Reator Químico

O exemplo implementado para o desenvolvimento do projeto consiste num reator químico, baseado num reator perfeitamente agitado (RPA), também conhecido como *Continuous Stirred-Tank Reactor* (CSTR). O reator é formado por um tanque com admissão para dois fluidos reagentes, um agitador que garante a mistura adequada dos reagentes, uma válvula de vapor que aquece o tanque e uma válvula de saída, utilizada para esvaziar o tanque após a produção do produto para armazenagem. Além disso, o modelo possui alguns equipamentos de controle, como boias, para controle do volume, um sensor de temperatura e um alarme, para indicar o transbordamento do tanque. Todo o esquema está demonstrado a seguir, na Figura 1.

Figura 5 - Modelo do Reator Químico



Fonte: Pellini (2013)

Algumas condições devem ser respeitadas pelo aluno, que deve programar o CLP de forma a garantir, dentre outros: correta temperatura do processo, correta

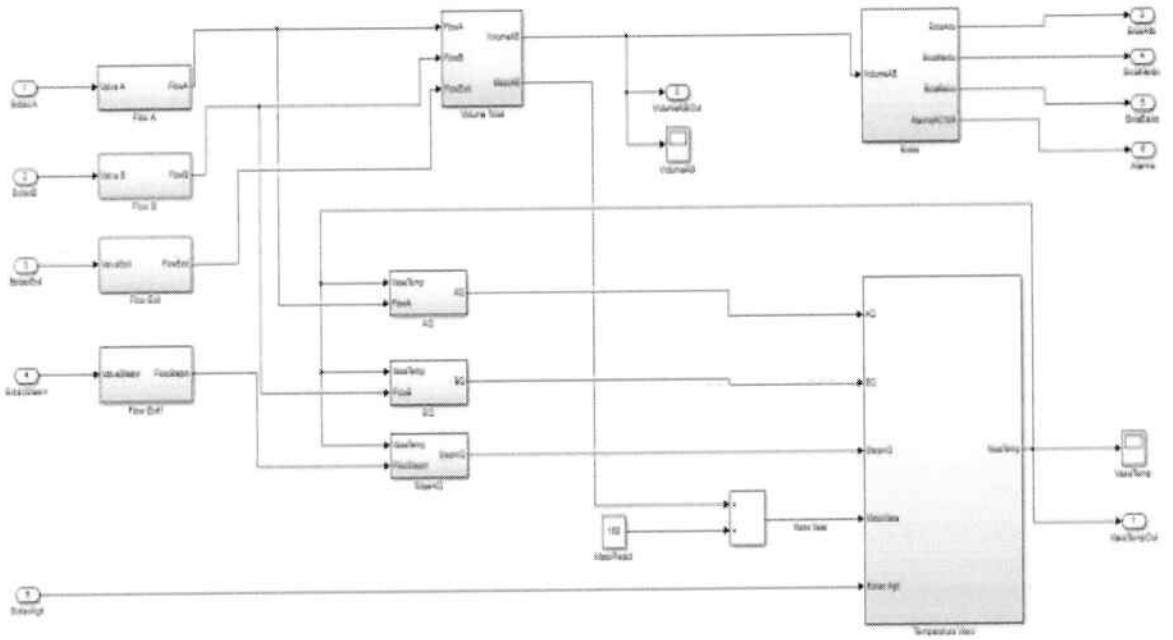
ordem de execução das tarefas, correto volume de entrada dos reagentes, alarmes em caso de erro.

Para isso, em nosso diagrama de blocos, temos entradas que recebem comandos de liga/desliga das válvulas de entrada dos reagentes. Logo em seguida, integradores que executam o controle do volume e nos mostram em gráficos em tempo real. Boias são acionadas dependendo do volume de mistura de reagentes no tanque. A temperatura é controlada e mostrada em outro gráfico em tempo real, ela depende de perdas de calor para o ambiente, que são aceleradas ao ligar-se o agitador, e de aquecimento via vapor, acionado por outra válvula.

4.1.1 Diagrama de blocos e funções de transferência

A seguir, na figura 2, está o diagrama de blocos, desenvolvido no *Simulink*, do modelo do reator implementado.

Figura 6 - Diagrama de blocos do modelo no Simulink

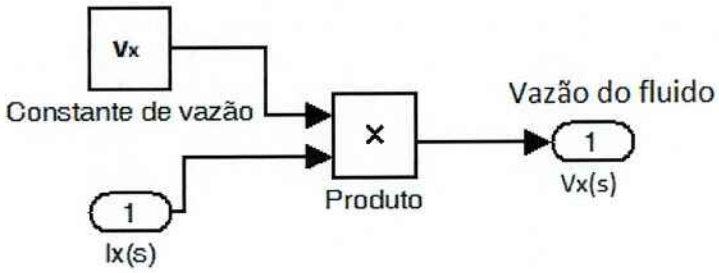


Fonte: Autores

Para uma melhor compreensão, alguns blocos de cálculo serão explicados com maior detalhe, com a demonstração dos diagramas e funções de transferência.

4.1.1.1 Vazão de fluidos

Figura 7 - Diagrama de blocos para vazão de fluidos



Fonte: Autores

A vazão de todos os fluidos do modelo segue apenas uma função de transferência. Há uma constante de vazão, que varia conforme o fluido e um parâmetro booleano - uma entrada digital -, que determina a abertura ou fechamento da válvula. A equação (1) demonstra a função de transferência para este bloco de cálculo.

$$I_x(s) * v_x = V_x(s) \quad (1)$$

Para

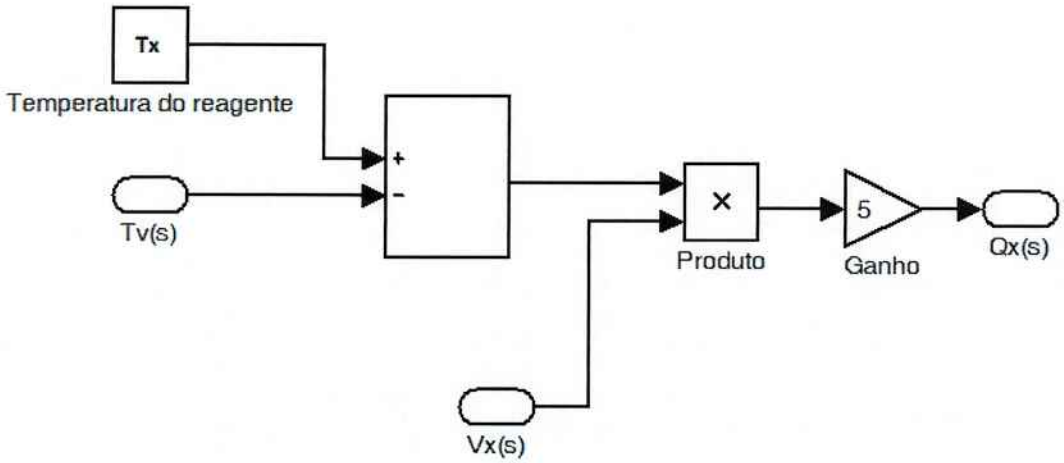
$I_x(s)$ – Entrada digital;

v_x – Constante de vazão;

$V_x(s)$ – Vazão do fluido.

4.1.1.2 Calor dos fluidos

Figura 8 - Diagrama de blocos para calor dos fluidos



Fonte: Autores

O calor de todos os fluidos do modelo, como no caso da vazão dos reagentes, segue apenas uma função de transferência, diferindo entre si apenas pela constante de temperatura. O diagrama de blocos possui duas entradas, a temperatura do vaso e a vazão do reagente. A equação (2) demonstra a função de transferência para este bloco de cálculo.

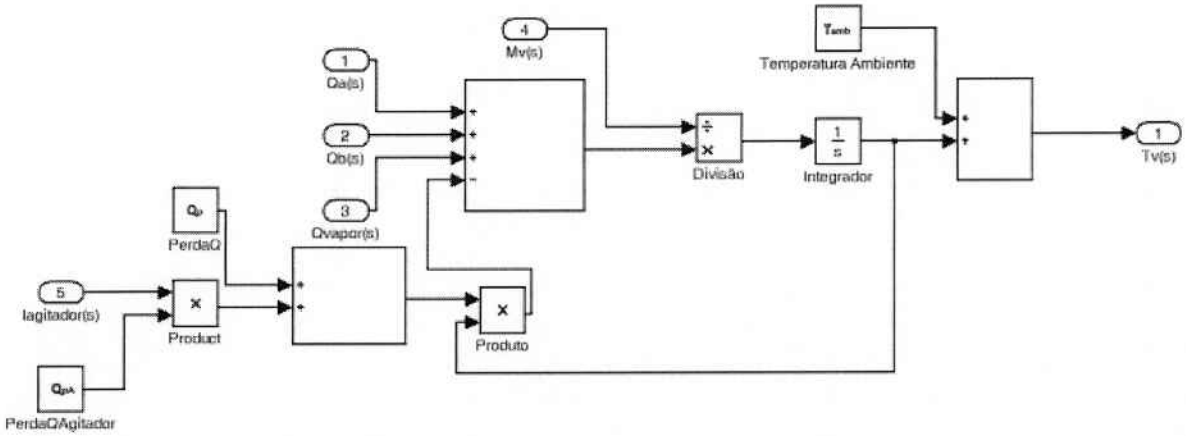
$$(\left(T_x - T_v(s)\right) * V_x(s)) * 5 = \ Q_x \tag{2}$$

Para

t_x – Temperatura do fluido;
$T_v(s)$ – Temperatura do tanque;
$V_x(s)$ – Vazão do fluido;
$Q_x(s)$ – Calor do fluido.

4.1.1.3 Temperatura do tanque

Figura 9 - Diagrama de blocos para a temperatura do tanque



Fonte: Autores

A temperatura do tanque é o cálculo mais complexo do modelo. O subsistema recebe cinco entradas diferentes: o valor do calor dos reagentes A e B, o valor do calor do vapor, a massa do fluido no tanque e a entrada digital que indica se o agitador está ligado ou não. A equação (3) demonstra a função de transferência para este bloco de cálculo.

$$\frac{Q_a(s)+Q_b(s)+Q_{vapor}(s)-(q_p+I_{agit}(s)*q_{pA})*T_1(s)}{s*M_v(s)} + T_{amb} = T_v(s) \qquad (3)$$

Para

$Q_a(s)$ – Calor do reagente A;

$Q_b(s)$ – Calor do reagente B;

$Q_{vapor}(s)$ – Calor do vapor;

q_p – Constante de perda de calor para o meio;

q_{pA} – Constante de perda de calor do agitador;

$I_{agit}(s)$ – Entrada do agitador;

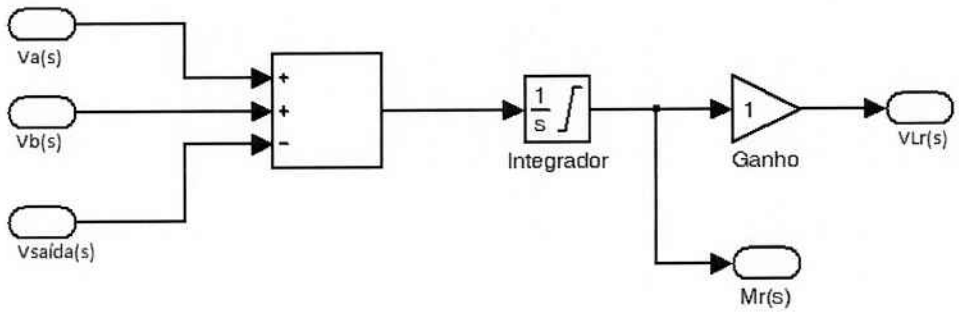
$T_1(s)$ – Temperatura instantânea do tanque;

$T_v(s)$ – Temperatura do tanque;

$M_v(s)$ – Massa total do reator.

4.1.1.4 Volume total

Figura 10 - Diagrama de blocos para o volume e massa dos reagentes



Fonte: Autores

O volume e massa totais dos reagentes são calculados através de um integrador da resultante da soma das vazões dos reagentes, subtraindo a vazão de saída. A equação (4) demonstra a função de transferência para este bloco de cálculo.

$$\frac{(V_a(s)+V_b(s)-V_{saída}(s)}{s} = VL_r(s) = M_r(s) \quad (4)$$

Para

$V_a(s)$ – Vazão do reagente A;

$V_b(s)$ – Vazão do reagente B;

$V_{saída}(s)$ – Vazão da saída;

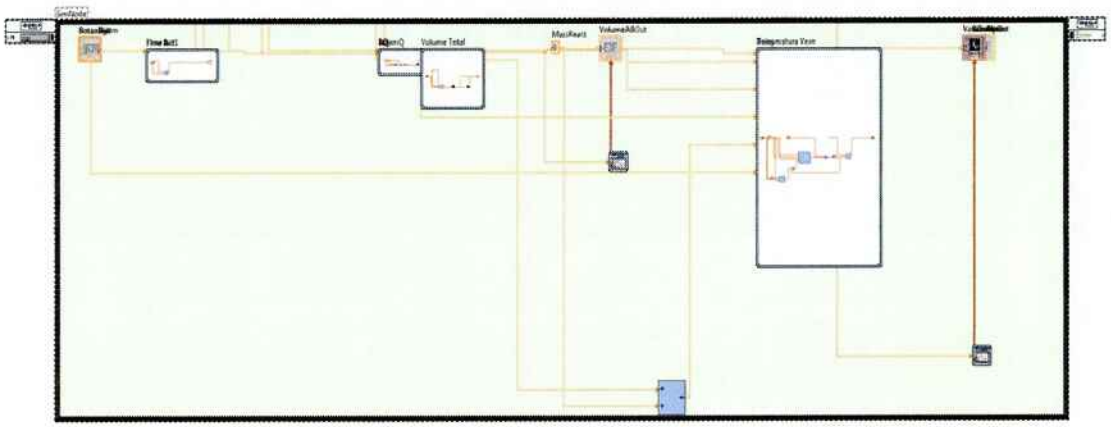
$VL_r(s)$ – Volume total dos reagentes;

$M_r(s)$ – Massa total dos reagentes.

4.2 Modelo convertido para o LabVIEW

Após o desenvolvimento do modelo apresentado na seção anterior, iniciou-se a etapa de conversão do arquivo feito no Simulink para o ambiente LabVIEW. Na próxima seção apresentamos um passo a passo detalhado do funcionamento do conversor e do LabVIEW. Nesta seção, ilustraremos rapidamente os estágios de conversão e apresentaremos o resultado, o modelo final, que será posteriormente transferido para o CompactRIO. Iniciaremos com o resultado da ferramenta de conversão, na figura a seguir.

Figura 11 - Resultado inicial da conversão do modelo

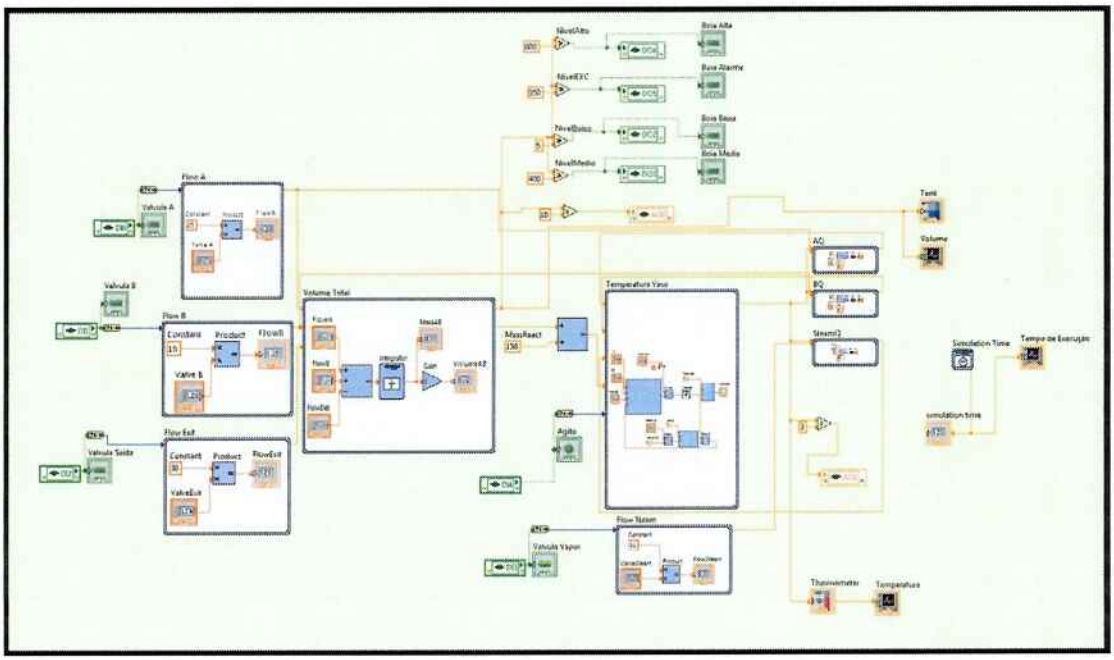


Fonte: Autores

Apesar de simplificar o processo de conversão, a ferramenta não faz todo o trabalho. Todas as entradas, saídas e constantes precisam ser configuradas novamente no modelo convertido, além de ser necessário ordenar o modelo novamente. Outro ponto a ser configurado é o step-time do modelo, que determinará a taxa de atualização que o controlador irá calcular as saídas. Por limitação do *software*, não há qualquer indicação automática da insuficiência de tempo para cálculo. Sempre existirá uma resposta, mesmo que o controlador não preserve o step-time. Este é um ponto de atenção para qualquer aplicação feita para esta arquitetura. O problema será mais bem discutido nas próximas seções.

O modelo final, após a configuração das constantes, entradas e saídas, está a seguir. Para garantir que a aplicação funcione em tempo real, implementamos um relógio no projeto, que compara o tempo simulado com o tempo verdadeiro.

Figura 12 - Modelo final do reator químico

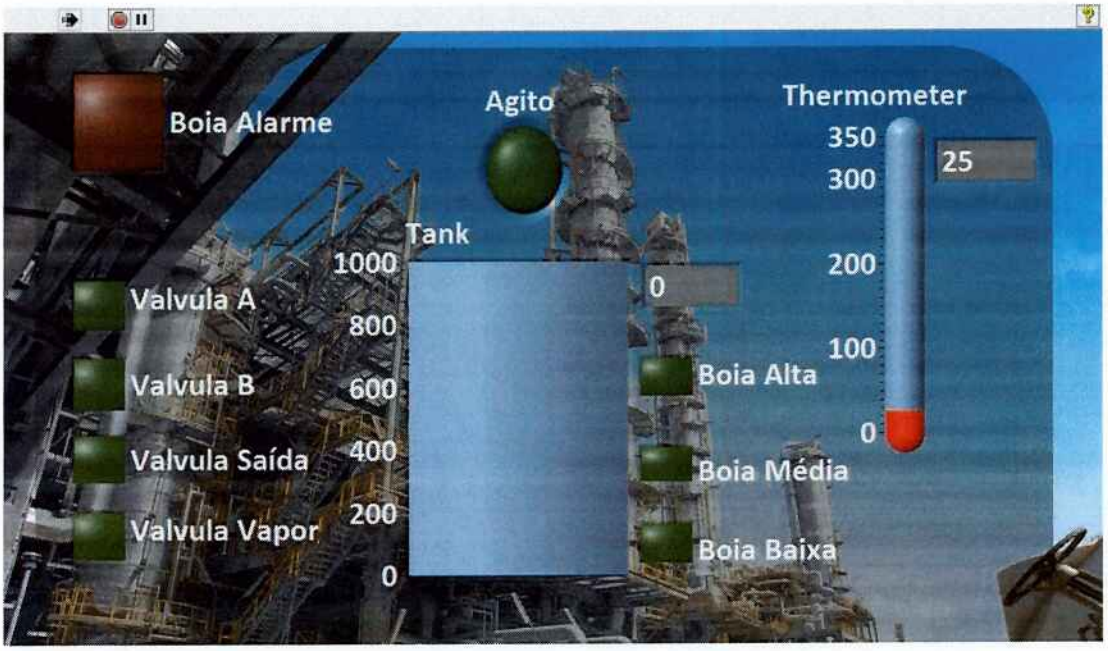


Fonte: Autores

Para uma melhor visualização dos resultados e acompanhamento da temperatura e volume, criamos um painel, que auxiliará o aluno mostrando os valores reais das variáveis e tornando a ação do controlador mais concreta. Este painel está demonstrado a seguir.

O painel (janela *Front Panel)* foi criado com as próprias ferramentas do LabVIEW. Todos os *LEDs* indicadores, o tanque e o termômetro, são de simples colocação e funcionam em conjunto com o Diagrama de Blocos, para facilitar a programação. Uma imagem de fundo e alguns ajustes estéticos para assegurar melhor experiência ao usuário foram empregados. O ideal é que em uma aplicação, o aluno só se comunique com o Painel Frontal, uma vez que o Diagrama de Blocos da simulação já estará completamente definido, logo também é importante que essa parte seja de estética compreensível e didática.

Figura 13 - Painel do LabVIEW



Fonte: Autores

Após a configuração e *upload* do modelo para o CompactRIO, restam os testes do projeto, para assegurar o bom funcionamento da experiência.

A próxima sessão conta com um guia para auxiliar às configurações futuras que se utilizem desta metodologia para empregar uma simulação em tempo real no laboratório. Ela conta também com instruções para ajustes a serem feitos ao diagrama convertido para que o programador possa mensurar e assegurar o bom funcionamento do modelo no CompactRIO.

5 GUIA PARA CONFIGURAÇÃO DE NOVAS APLICAÇÕES

Este passo a passo foi escrito com o objetivo de simplificar a montagem de uma experiência que se utiliza do CompactRIO para simular em tempo real um modelo matemático representando uma situação real que possibilitará o aluno a realizar estudos de automação.

5.1 Passo 1: Softwares e Hardwares necessários

Os *softwares* e *hardwares* aqui listados foram aqueles utilizados pelo grupo:

- Computador com Windows 7 Professional (Service Pack1) 64 bits
- MATLAB R2012a (possui licenciamento para a USP)
- LabVIEW 2013 (possui licenciamento para a USP) contendo:
- FPGA MODULE
- REAL TIME MODULE
- CONTROL DESIGN AND SIMULATION MODULE
    - Todos os módulos citados podem ser baixados diretamente do site da *National Instruments* em versão teste e posteriormente licenciados dentro da USP pelo servidor Tera
- CompactRIO (NI cRIO-9024) contendo:
- Módulo NI 9205 (Entradas Analógicas)
- Módulo NI 9421 (Entradas Digitais)
- Módulo NI 9264 (Saídas Analógicas)
- Módulo NI 9472 (Saídas Digitais)
- Fonte NI PS-15

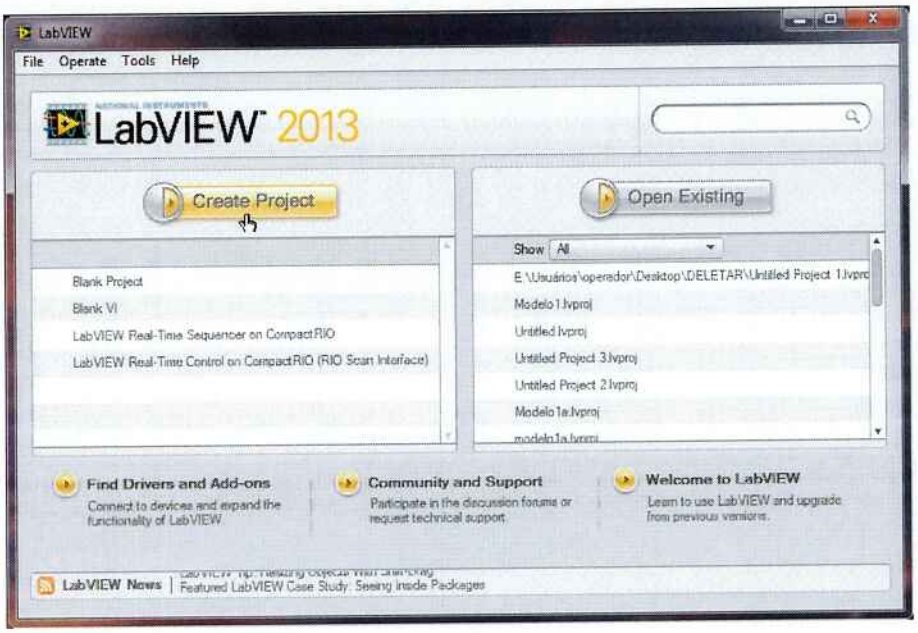
5.2 Passo 2: Conversão MATLAB/LabVIEW

a) Confeccionar o Diagrama de Blocos no Simulink (MATLAB) e salvá-lo no formato ".mdl".

b) Todo este procedimento deve ser executado com ambos programas abertos e ativos no computador, para se evitar erros e falhas.

c) Crie um novo projeto no LabVIEW.

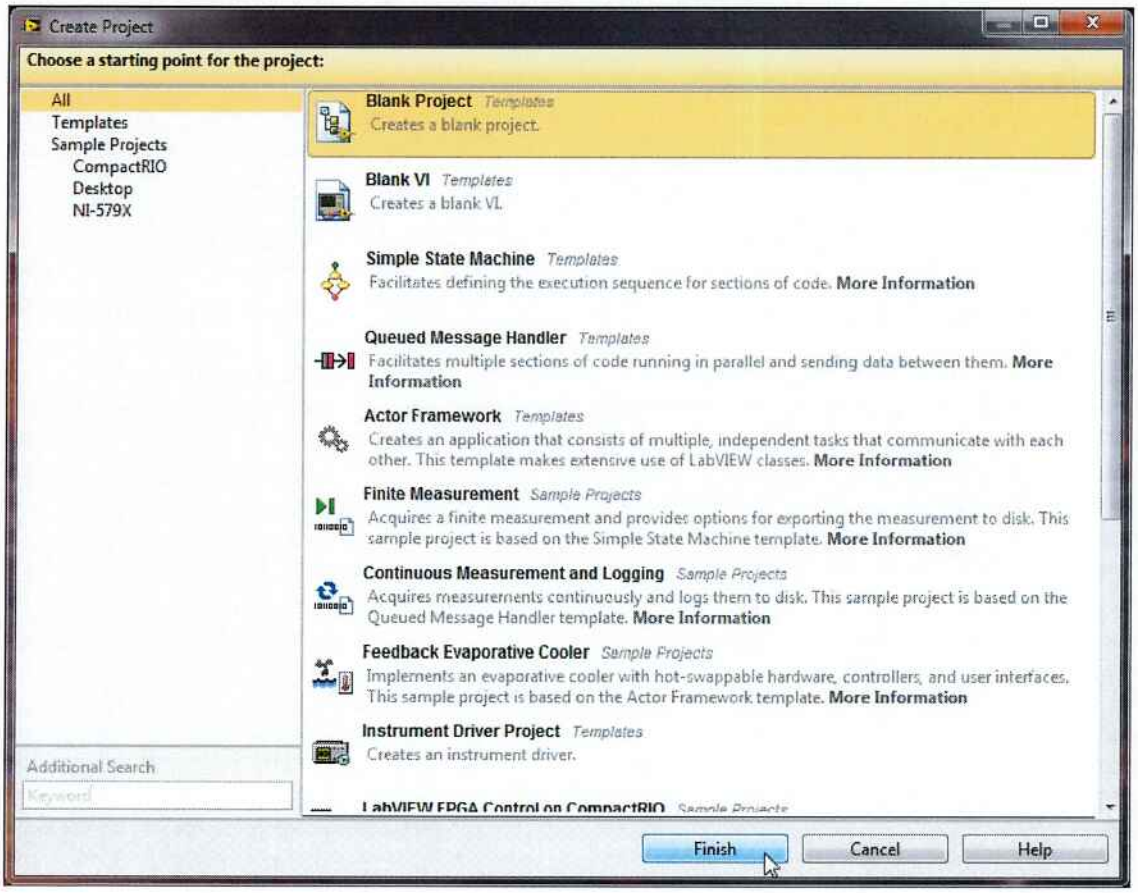
Figura 14 - Janela Inicial do LabVIEW



Fonte: Autores

d) Selecione "Blank Project" e clique em Finish.

Figura 15 - Janela "Create Project"



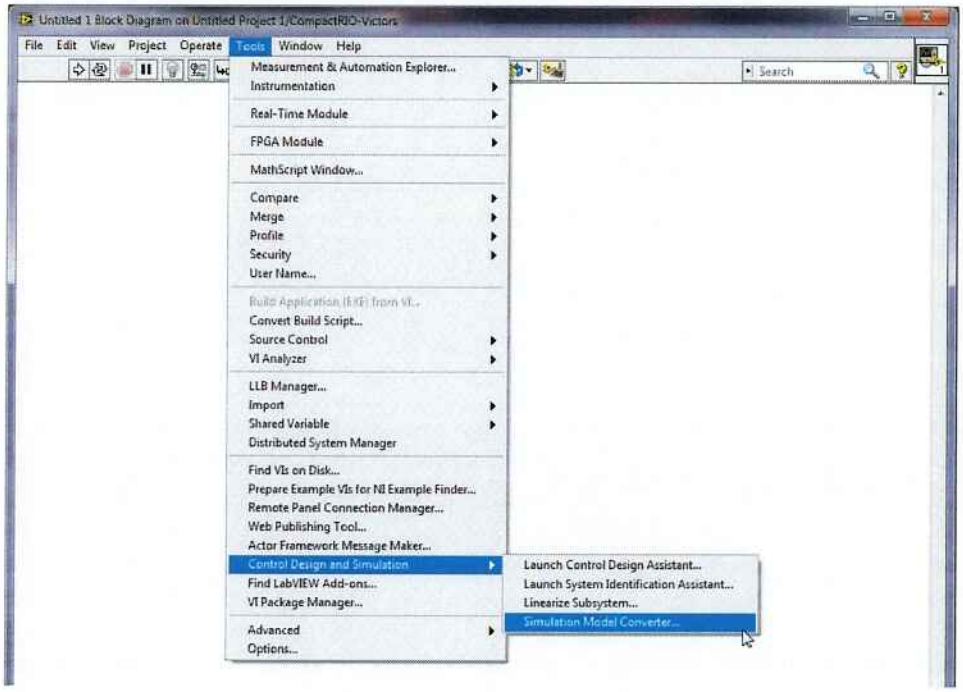
Fonte: Autores

e) Crie um novo VI.

f) Duas janelas irão surgir, "*Block Diagram*" e "*Front Panel*", Ctrl+T mostra ambas ao mesmo tempo.

g) Na janela "*Block Diagram*" selecione *Tools>Control Design and Simulation>Simulation Model Converter...*

Figura 16 - Instruções para acessar o "Simulation Model Converter"



Fonte: Autores

h) Surgirá então uma janela como a que segue:

Figura 17 - Janela do "Simulation Model Converter"

Fonte: Autores

i) Em "*Source File or Directory to Convert*" selecione o arquivo ".mdl" a ser convertido.

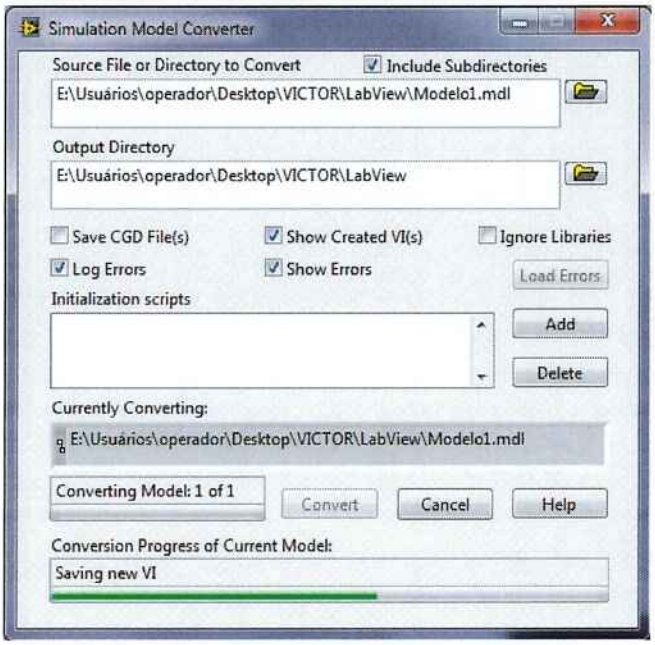
j) Em "*Output Directory*" selecione o local onde será gravado o arquivo do LabVIEW após conversão.

k) Selecione a caixa "*Show Created VI(s)*".

l) Clique em "*Convert*".

m) A janela em progresso é a que segue e o arquivo estará pronto quando a barra verde for completada.

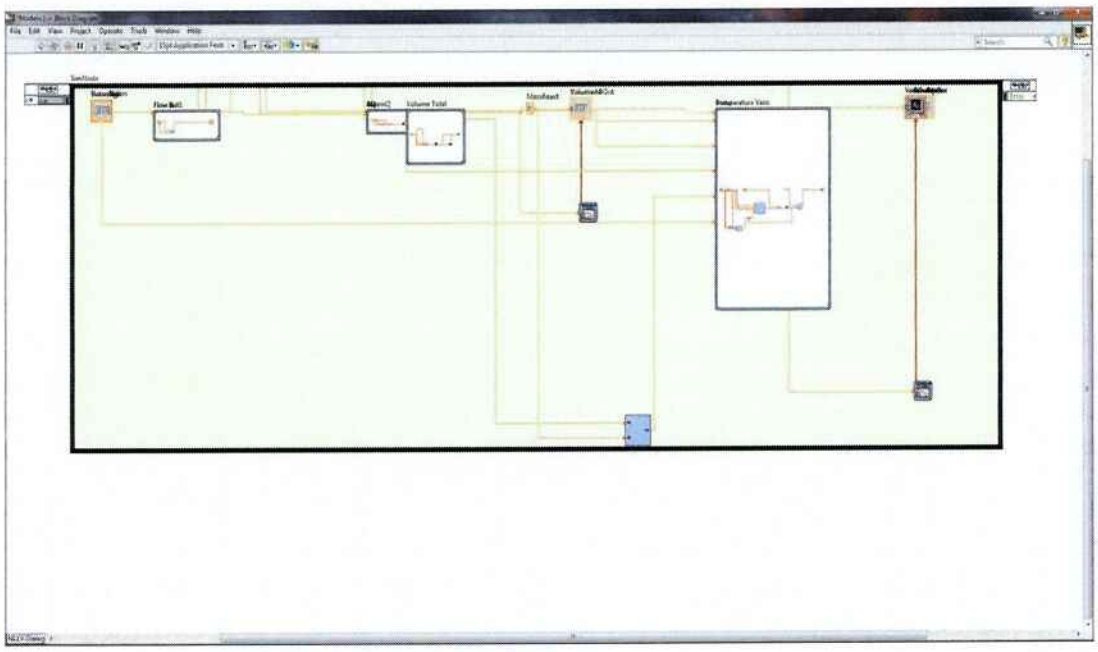
Figura 18 - Conversão do arquivo ".mdl" para o LabVIEW em progresso



Fonte: Autores

n) Após a correta conversão, o modelo está desorganizado e necessita de alguns ajustes.

Figura 19 - Diagrama de blocos logo após a conversão



Fonte: Autores

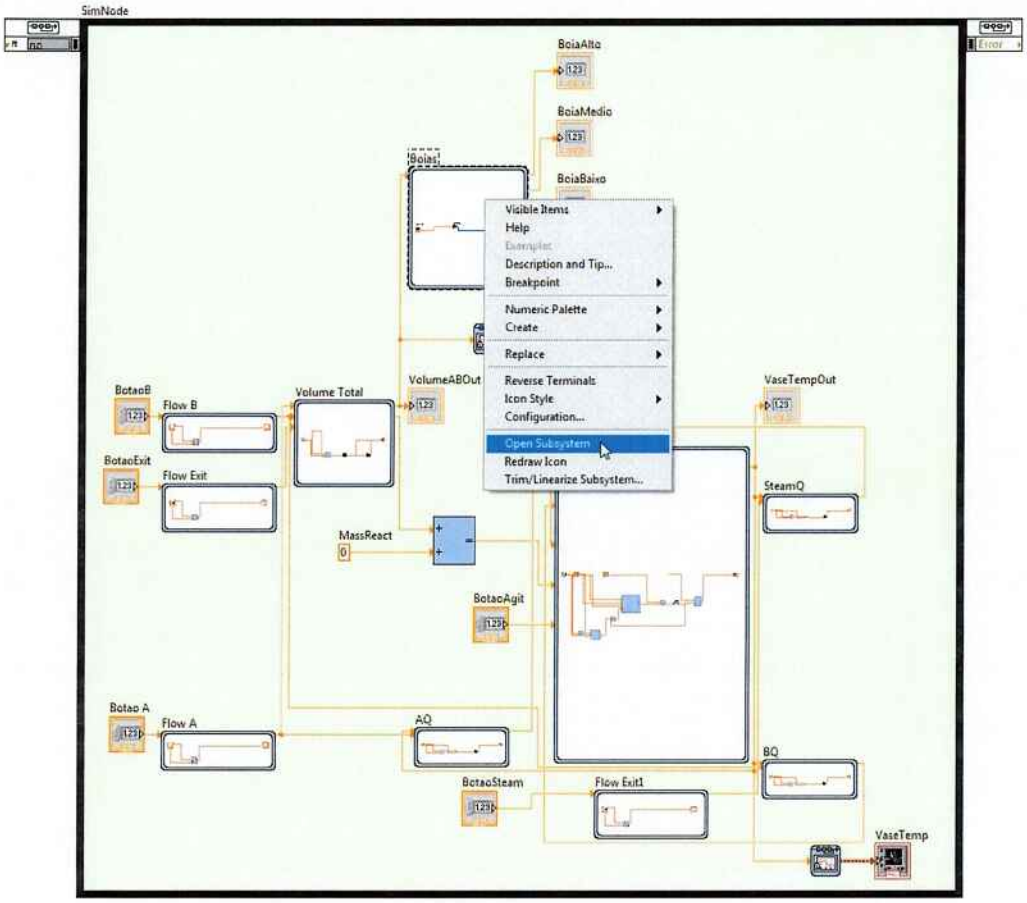
o) Utilizando-se do botão "*Clean Up Diagram*" (ou "*Clean Up Selection*" quando algo estiver selecionado) podemos agilizar este trabalho.

Figura 20 - Indicação do comando "Clean Up Diagram"

Fonte: Autores

p) Para acessar os subsistemas e organizá-los, selecionamos a opção "*Open Subsystem*" após clicar com o botão direito no subsistema. Lembre-se de utilizar o comando Ctrl+T para trazer à vista tanto o Diagrama de Blocos como Painel Frontal

Figura 21 - Instruções para acessar um subsistema no Diagrama de Blocos

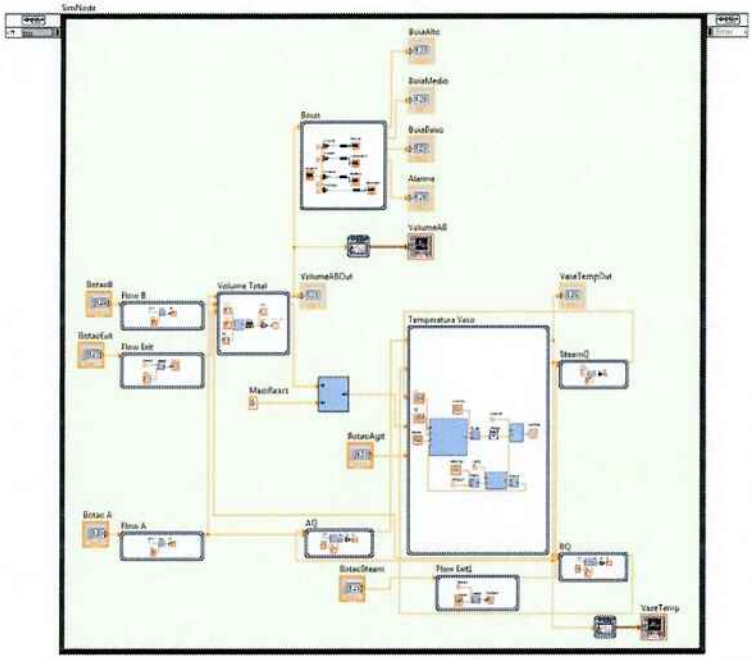


Fonte: Autores

q) A conversão não é perfeita. Resta trocar as entradas e saídas de acordo com o que se pretende fazer e reescrever todos os valores de constantes que se perdem no processo de conversão. Na medida em que se acessam os subsistemas para organizá-los, pode-se aproveitar para reescrever as constantes que dali foram apagadas.

r) Pode-se adicionar quaisquer gráficos e outros tipos de representação numérica no Painel Frontal, instantaneamente surgirá no Diagrama de Blocos o bloco correspondente, que deve ser ligado à informação que se deseja mostrar no gráfico. Antes de poder conectar as entradas e saídas da simulação com as respectivas entradas e saídas digitas do projeto, precisamos adicionar o CompactRIO ao projeto, assunto do próximo tópico.

Figura 22 - Diagrama de blocos organizado

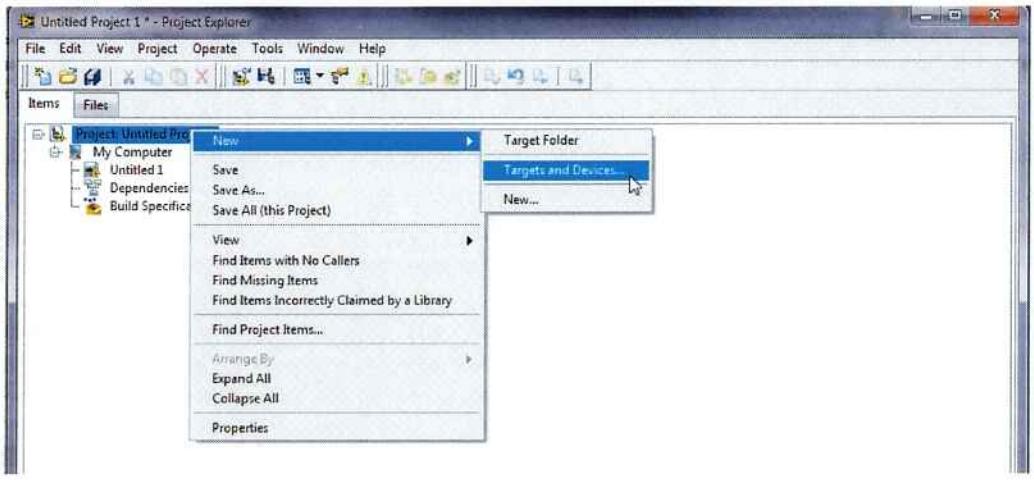


Fonte: Autores

5.3 Passo 3: Conexão do CompactRIO

a) Na nova tela, clique com o botão direito em *Project* em seguida selecione *New>Targets and Devices...*

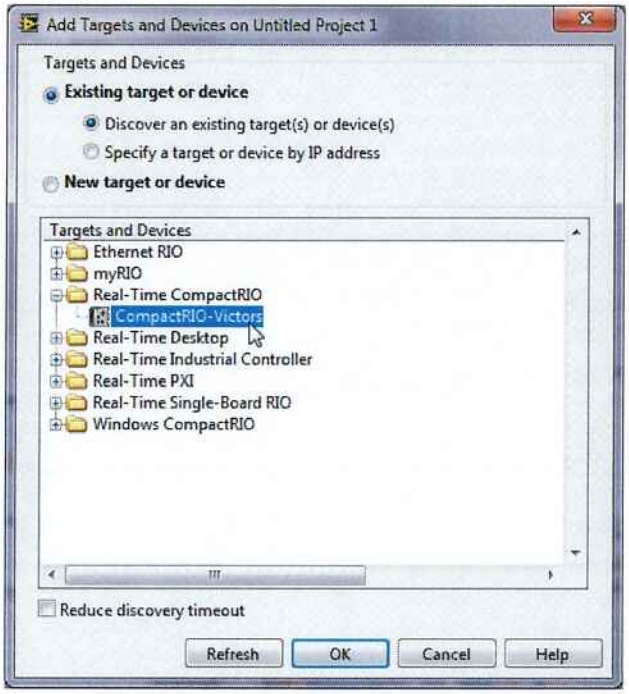
Figura 23 - Instruções para acessar "Targets and Devices..."



Fonte: Autores

b) Na janela "*Add Targets and Devices*, selecione o CompactRIO que será utilizado como na figura a seguir e clique em "OK"

Figura 24 - Adicionando o CompactRIO ao projeto

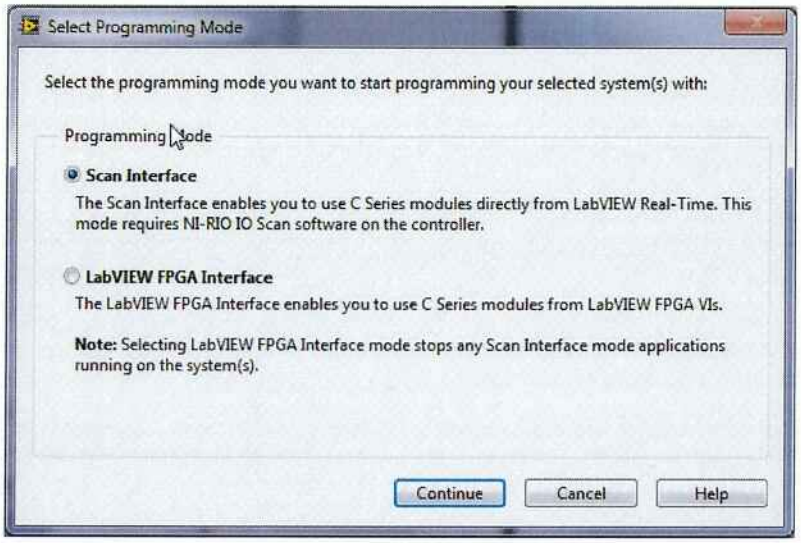


Fonte: Autores

c) Na janela "*Select Programming Mode*" selecione *Scan Interface* e clique em "*Continue*"

d) Quaisquer erros na conexão causarão o não aparecimento do CompactRIO nessa janela. Deve-se primeiro verificar no programa NI MAX, parte do pacote LabVIEW, que a conexão está de acordo. Se for uma rede com muitos computadores e equipamentos, é necessário verificar as configurações de IP e certificar-se de que não há nenhum conflito com aquele designado ao CompactRIO.

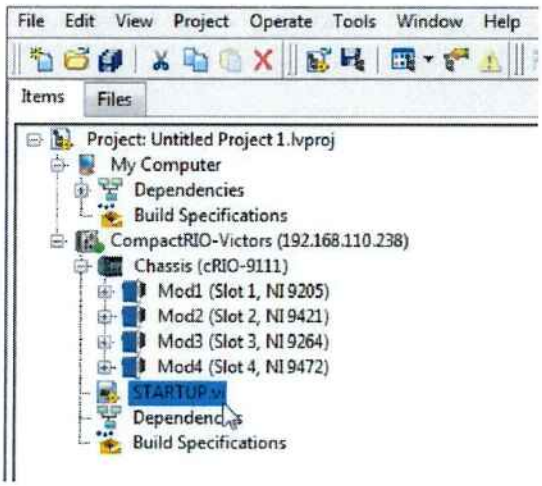
Figura 25 - Seleção do modo de programação



Fonte: Autores

e) Uma vez com o CompactRIO adicionado ao projeto, clique e arraste o VI convertido de seu projeto para baixo do Chassis do CompactRIO, para que o VI não seja executado no seu computador e sim no CompactRIO. O exemplo da figura a seguir conta com um VI de nome STARTUP.

Figura 26 - Adicionando o VI ao CompactRIO

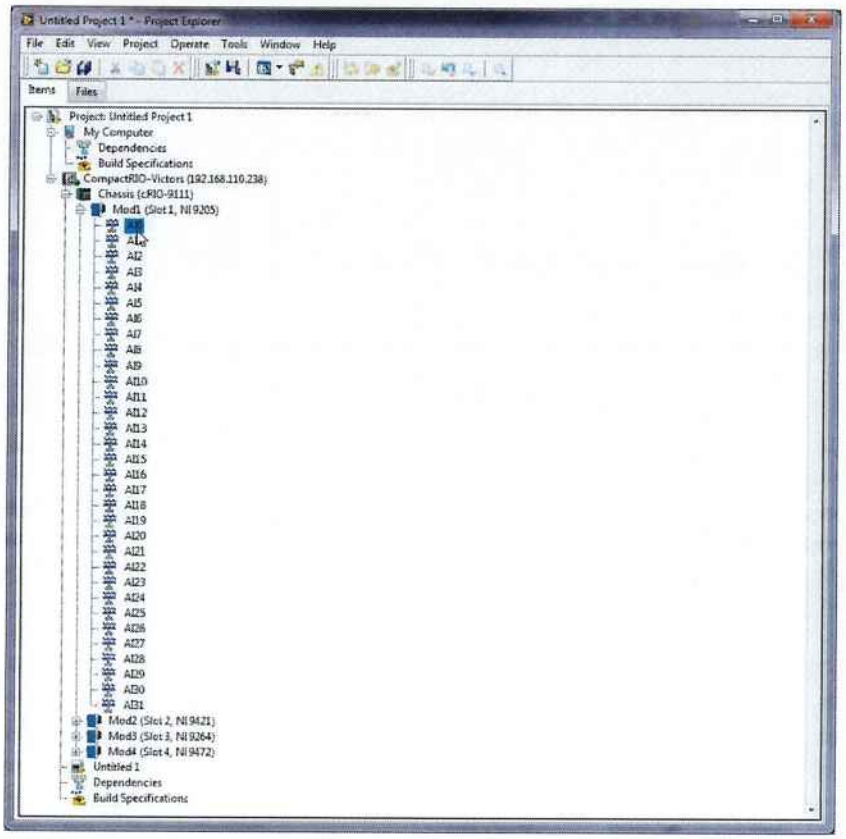


Fonte: Autores

5.4 Passo 4: Adição das entradas e saídas ao Diagrama de Blocos

a) Abaixo do CompactRIO na janela de projeto também se encontram seus módulos I/O, e as respectivas entradas e saídas.

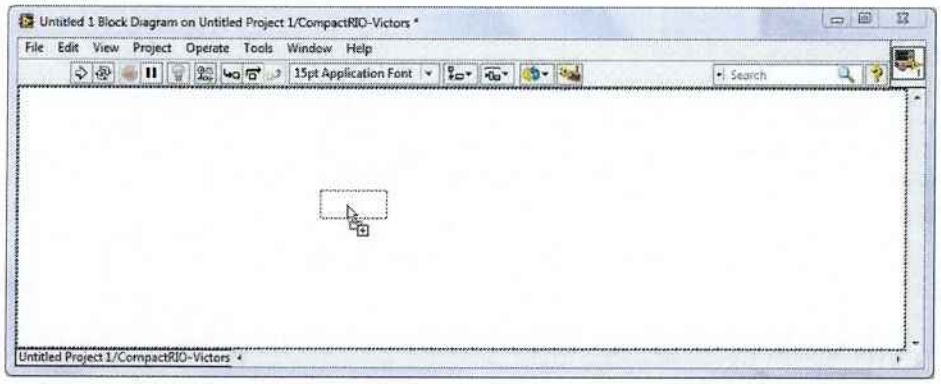
Figura 27 - Selecionando uma entrada analógica



Fonte: Autores

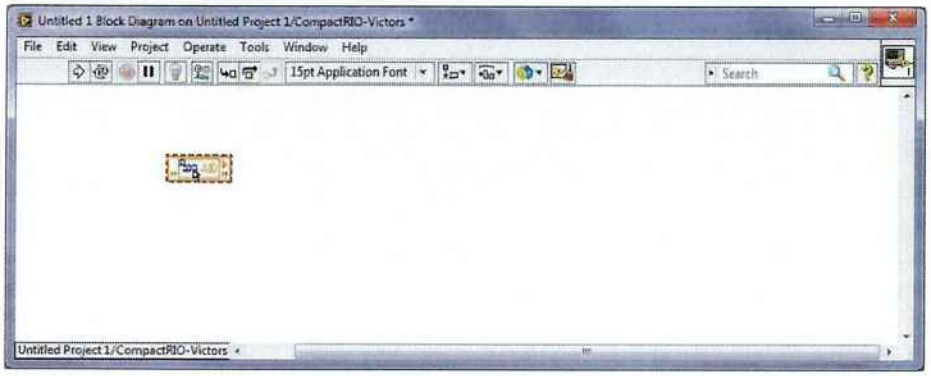
b) Para adicionar uma saída ao projeto, clique sobre ela na janela de projeto e arraste até o Diagrama de Blocos, como nas figuras a seguir.

Figura 28 - Adicionando uma entrada/saída ao diagrama de blocos parte 1



Fonte: Autores

Figura 29 - Adicionando uma entrada/saída ao diagrama de blocos parte 2



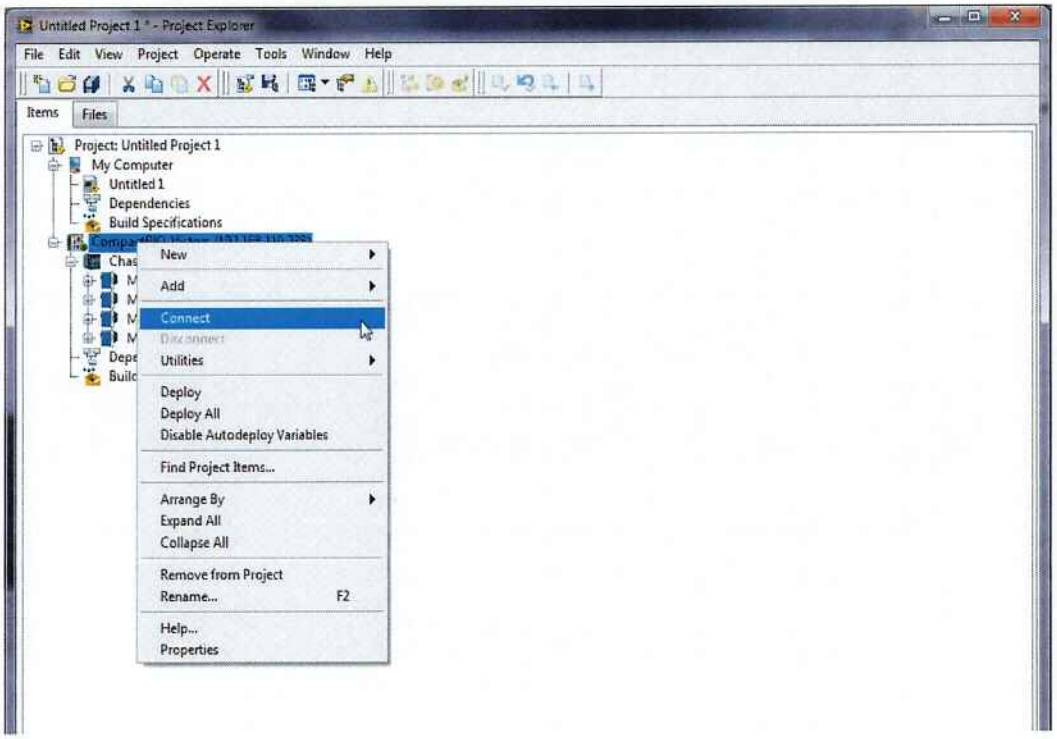
Fonte: Autores

5.5 Passo 5: Como definir VI de inicialização

Apesar de não utilizarmos esse comando na simulação final, é um jeito de se utilizar a simulação que deve ser considerado no futuro, tornando-a menos dependente de um PC

a) Através do *Project Explorer*, conectar o PC ao CompactRIO.

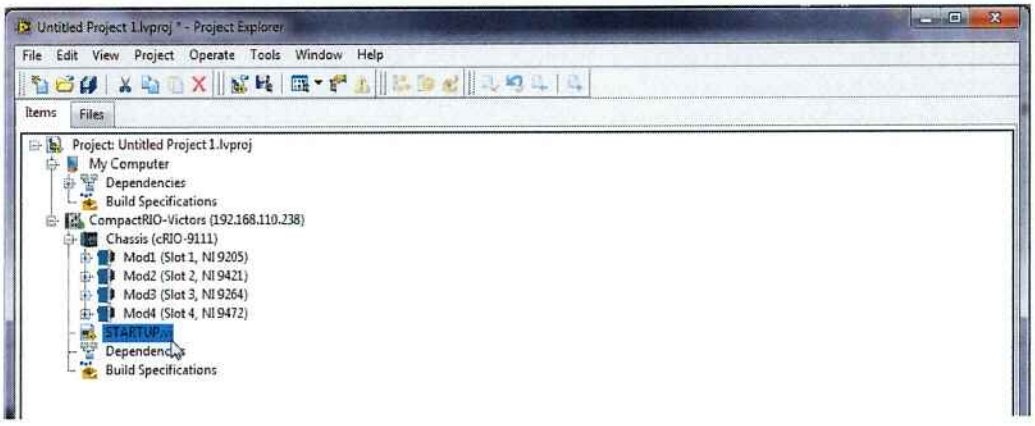
Figura 30 - Conectando o CompactRIO ao PC



Fonte: Autores

b) Através do *Project Explorer*, transferir o VI desejado do computador para o CompactRIO

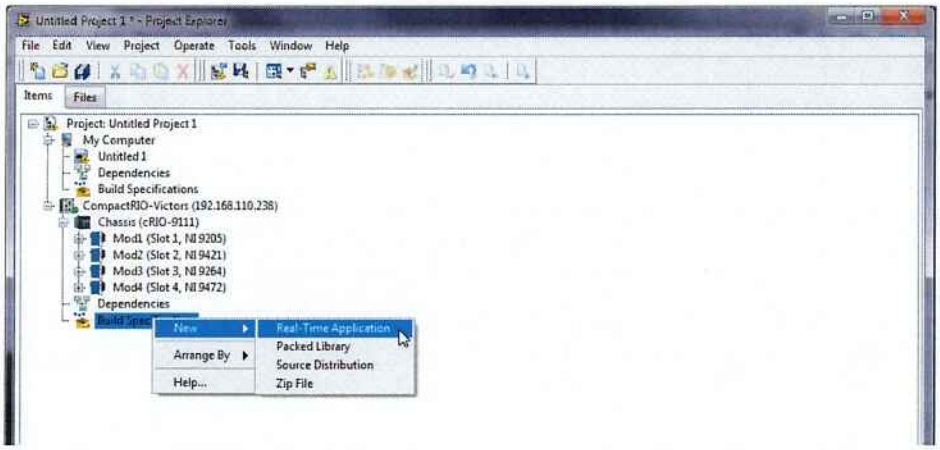
Figura 31 - Adicionando um VI ao CompactRIO



Fonte: Autores

c) Clique com o botão direito em *Build Specifications* (abaixo do dispositivo que se deseja configurar), em seguida selecione *New* > *Real-Time Application*

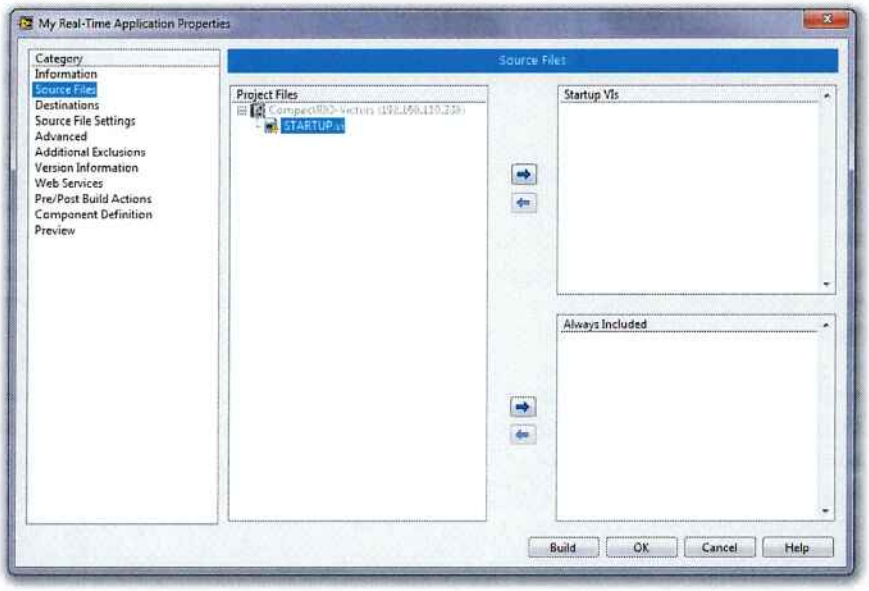
Figura 32 - Construindo uma aplicação Real-Time



Fonte: Autores

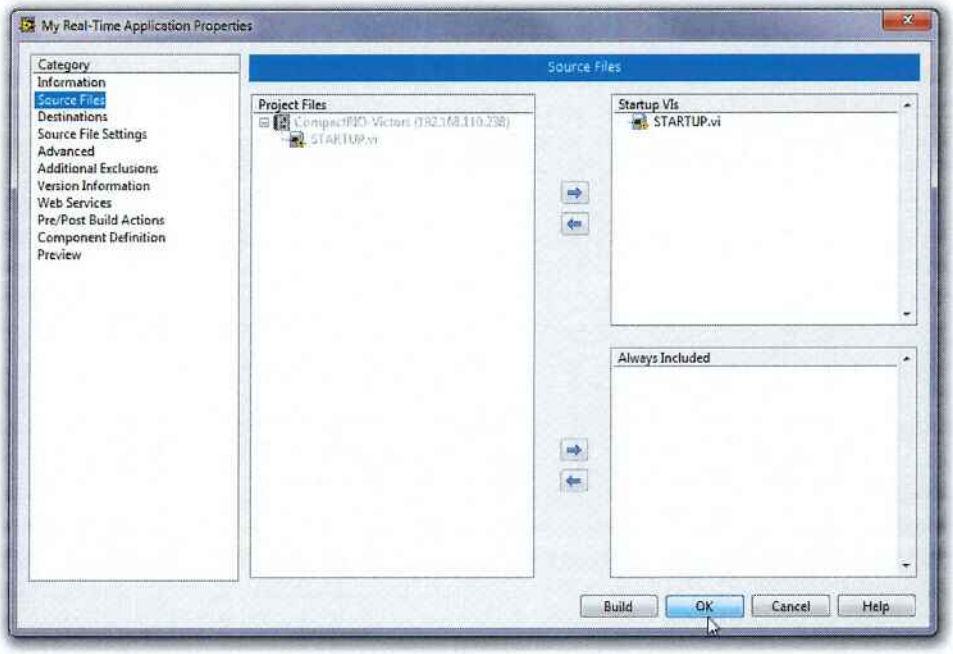
d) Na janela *"My Real-Time Application Properties"*, selecionar a categoria "*Source Files*" e transferir o VI de *start-up* para "*Startup VIs*".

Figura 33 - Janela "My Real-Time Application Properties"



Fonte: Autores

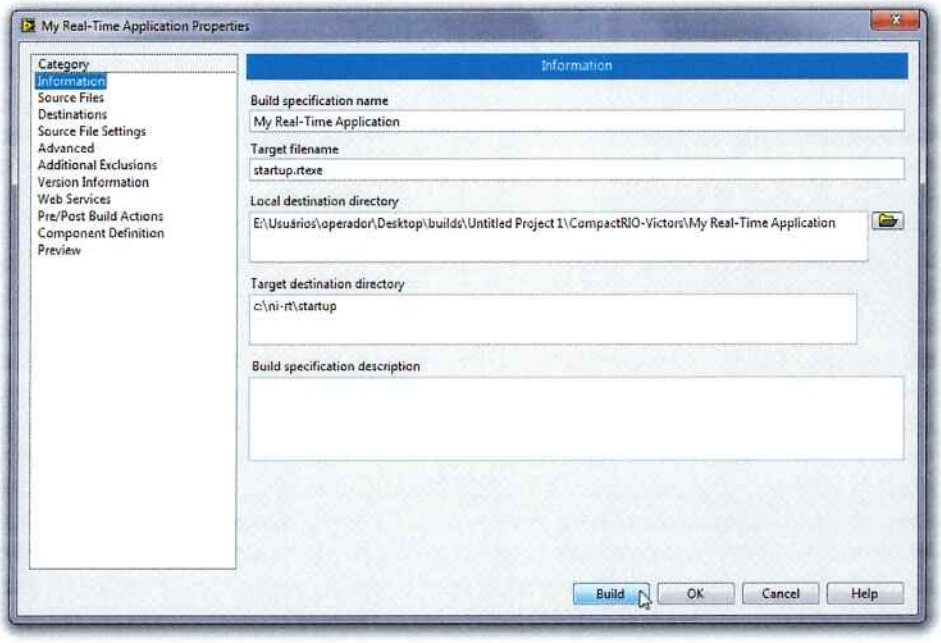
Figura 34 - Transferindo o VI para inicialização



Fonte: Autores

e) Selecionar categoria "*Information*" e clicar em "*Build*".

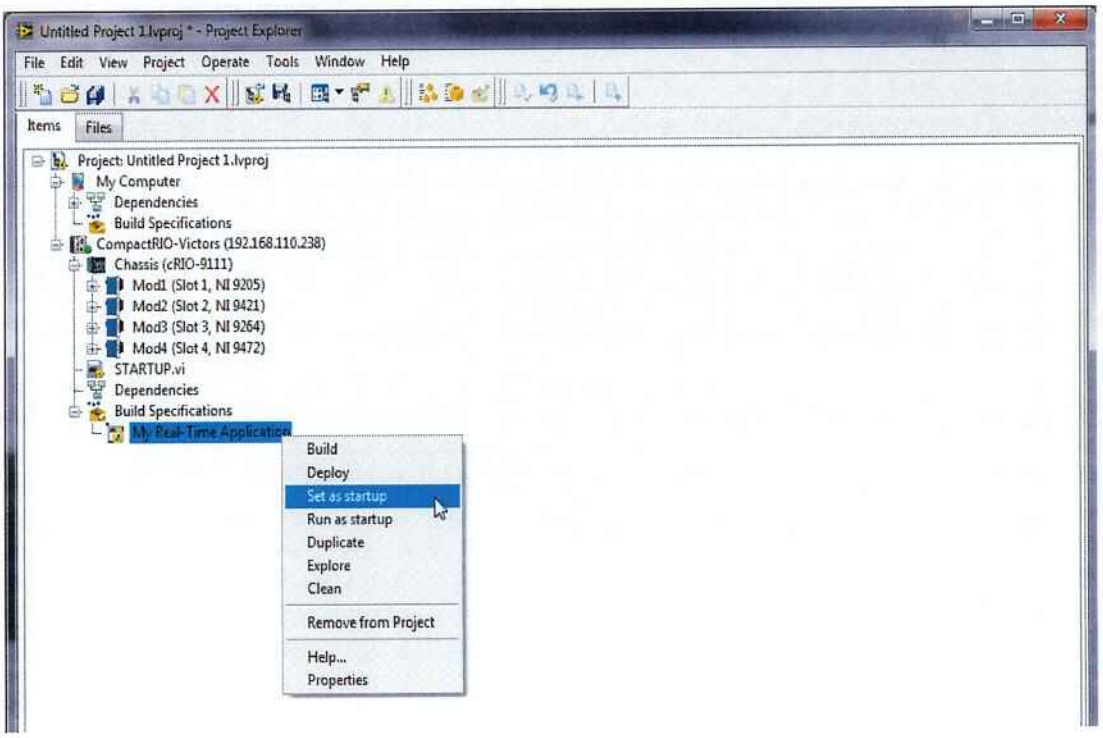
Figura 35 - Construindo a aplicação no CompactRIO



Fonte: Autores

f) Após o processo de construção terminar, clicar com o botão direito em "*My Real-Time Application*" e selecionar "*Set as startup*" para o VI partir automaticamente toda vez que o CompactRIO iniciar.

Figura 36 - Definindo a aplicação como inicialização



Fonte: Autores

5.6 Passo 6: *Shared Variables* e registro em ".txt"

Apesar de não utilizarmos esse método no fim do projeto, ele pode ser interessante para algumas aplicações e registros.

a) Para adicionar uma *Shared Variable* ao projeto, clique com o botão direito em cima do CompactRIO e selecione *NEW>VARIABLE*.

b) Selecione o nome e o *Data Type* na janela que abre.

c) Essa nova variável será salva em uma biblioteca de variáveis que será criada. Ao salvar o programa também surgirá uma janela para salvar esta biblioteca.

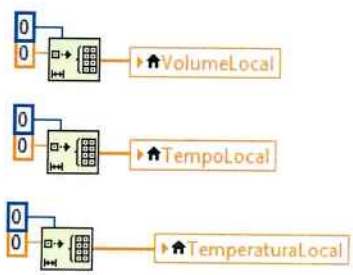
Salve-a no mesmo diretório do resto dos arquivos.

d) Adicione a *Shared Variable* ao projeto da mesma forma que se adiciona uma entrada ou saída do CompactRIO, como mostrado no último passo. Ao ser adicionada, ela aparece em formato de leitura. Para mudar entre formato de leitura e escrita, clique com o botão direito e selecione "*Change to write*", o inverso é análogo. Em seguida ligue-a à informação que se deseja guardar. Lembre-se de que a *Shared Variable* deve ter o mesmo *Data Type* da informação, o não cumprimento desse requisito acarretará erros.

e) No caso do registro de arrays, deve-se criar também *Local Variables*. O método mais fácil é criar um array no Painel Frontal (botão direito>*Array, Matrix...>Array*), em seguida colocando dentro dele o tipo de variável que ele deve conter (arrastando e soltando o clique), booleana colocando um botão direito>*Boolean>Vertical Toggle Switch* ou *double* colocando um botão direito>*Numeric>Numeric Control* são as mais utilizadas. No Diagrama de Blocos irá surgir uma representação. As *Local Variables* podem ser criadas no Diagrama de Blocos com botão direito>*Programming>Structures>Local Variables*.

f) Deve-se inicializar as *Local Variables* como na figura a seguir, fora do SimNode, com o bloco *Inicialize array* e entradas zero em *elemento* e *dimention size* (botão direito>*Create>Constant*).

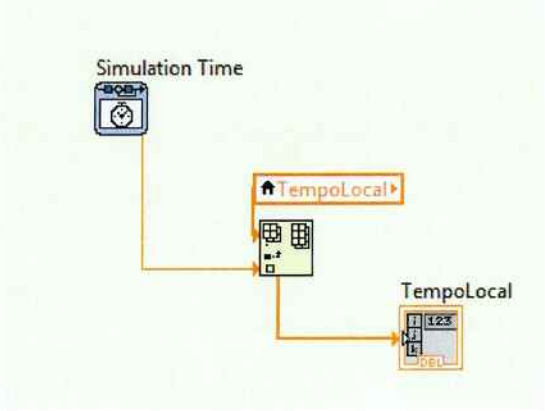
Figura 37 - Inicialização de Local Variables



Fonte: Autores

g) Para registrar a informação em uma Local Variable, faz-se como na figura a seguir. No exemplo estamos registrando o *Simulation Time.* Liga-se a *Local Variable* na entrada *n-sim array*, *Simulation Time* na entrada *n or n-1 dim array* e em *output array* o próprio *array* onde se está escrevendo.

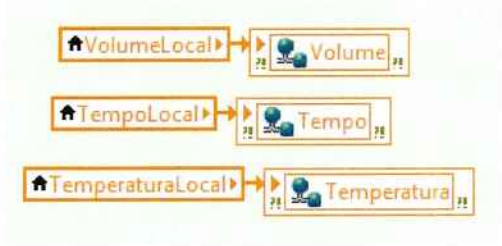
Figura 38 - Registro em uma Local Variable



Fonte: Autores

h) Por fim passa-se direto às *Shared Variables* de mesmo *Data Type* como na figura 39.

Figura 39 - Registro de Shared Variables

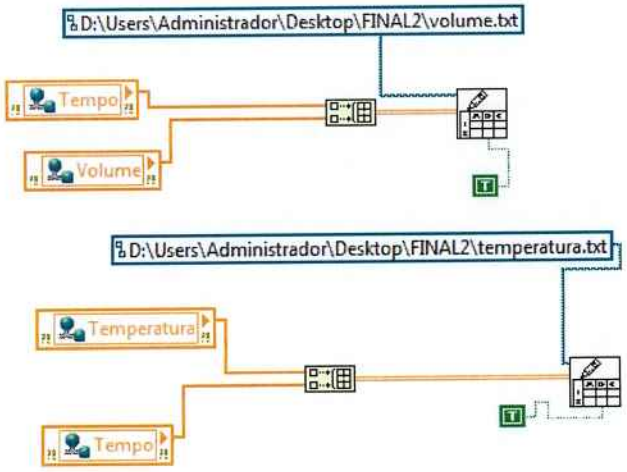


Fonte: Autores

i) O VI de registro criado no PC deve ser como o da figura a seguir, utilizando-se da função *build array*, depois de *write do spreadsheet file,* Colocando em file path o caminho do arquivo em que se deseja registrar os sinais um sinal

de *true* na opção de transposição. E a compilação dos 2 arrays em *2D data*. O arquivo deve ser executado quando se quiser registrar (e não continuamente).

Figura 40 - VI de registro



Fonte: Autores

j) Durante todo esse processo de comunicação, o LabVIEW cria VIs internos de comunicação de dados. É importante que ao pedir para salvar, você salve todos eles. Geralmente eles vêm com nome "*Untitled*" seguido de alguma numeração, dependendo de quantos com esse nome foram salvos antes.

5.7 Passo 7: Ajuste, correções e critérios de tempo

Esta última parte é fundamental ao projeto e são essas instruções que garantem uma simulação de qualidade em tempo real.

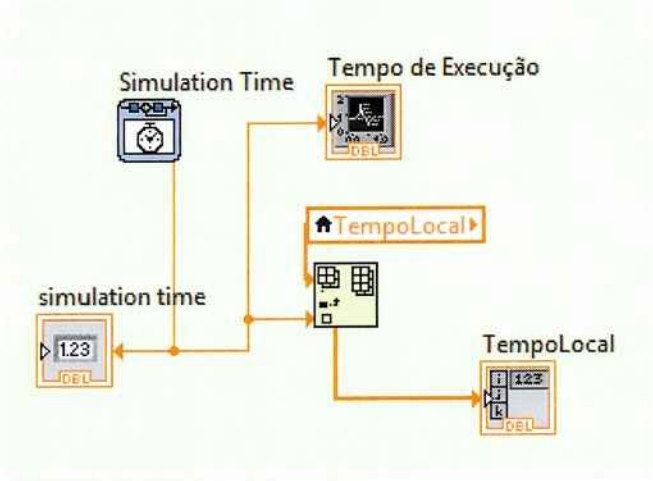
a) Logo que efetuamos a conversão de um projeto seguindo o passo 2, as características de tempo vêm pré estipuladas no SimNode e devemos realizar modificações de acordo com cada projeto, para que a simulação resulte em uma de qualidade e confiabilidade.

b) Primeiramente, vamos criar um mecanismo para comparar o tempo de simulação contado pelo bloco "*Simulation Time*" que funciona dentro do SimNode e compará-lo a um contador externo que representa o tempo em segundos.

c) Deve-se criar um Loop For do lado de fora do SimNode para que rode em paralelo e internamente contenha um bloco da função "*wait*" (botão direito>*Timing*>*Wait*) e em sua entrada clique com o botão direito e selecione *Create*>*Constant* e digite 1000, para que o bloco conte 1000 milissegundos.

d) Em seguida, execute os procedimentos do último tópico do passo a passo para salvar as informações de um bloco "*Simulation Time*" em uma *Local Variable*. Pode-se também estampar essa variável em um gráfico como na figura a seguir, mas isso não é necessário. É interessante, no entanto, registrá-la em uma variável para acompanhar sua evolução, clicando com botão direto na saída de *Simulation Time* e escolhendo *Create*>*Control*.

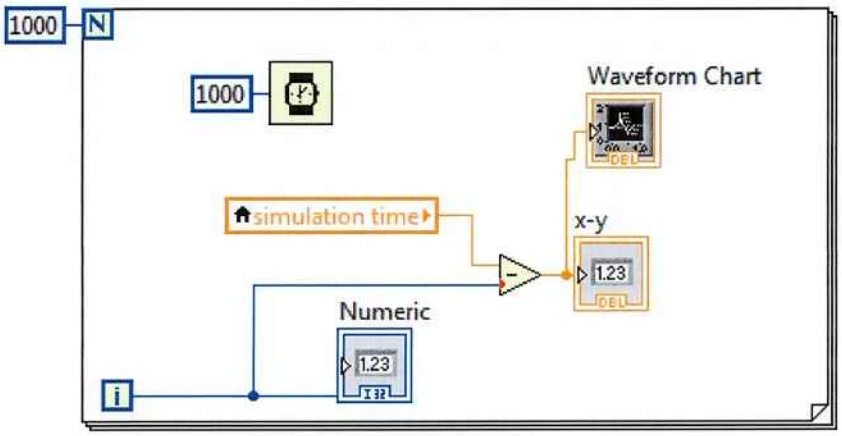
Figura 41 - Registro de Simulation Time em um array



Fonte: Autores

e) Na sequência, coloca-se a *Local Variable* dentro do Loop For e compara-se com o valor da contagem do Loop For, localizado no canto inferior esquerdo (que também pode ser mostrado no *Front Panel* criando-se um *control* com botão direito>*Create*>*Control* na saída do "i"). Para isso necessitamos de um bloco de subtração, localizado no menu *Math*, após clicar com o botão direito. Na saída cria-se um controle, como para o "i". Pode-se também criar um gráfico como esse que está estampado na figura a seguir, porém não há necessidade, o indicador é suficiente e interfere menos no tempo de processamento.

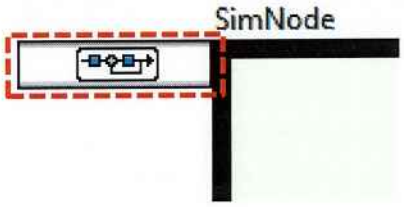
Figura 42 - Loop For



Fonte: Autores

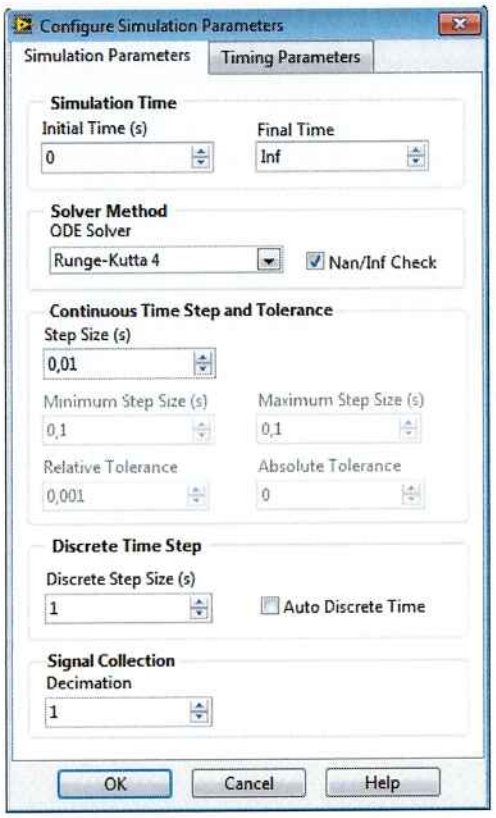
f) Tendo criado um meio de avaliar como o programa reage à mudança de parâmetros de simulação, podemos começar a focar nos parâmetros em si. No canto superior esquerdo do SimNode existe uma caixa na parte de fora, que com um duplo clique em sua área abre as configurações dos parâmetros de simulação do SimNode.

Figura 43 - SimNode com caixa de configurações em destaque



Fonte: Autores

Figura 44 - Janela dos parâmetros de simulação



Fonte: Autores

g) Ao abrir-se a janela deve-se configurar os parâmetros da simulação. *Initial Time* deve ser zero, para que o programa comece a simular assim que o usuário clique em "*Run*" para o programa rodar. O final time deve ser colocado como "*Inf*" escrito dessa forma porém sem as aspas (igual à figura

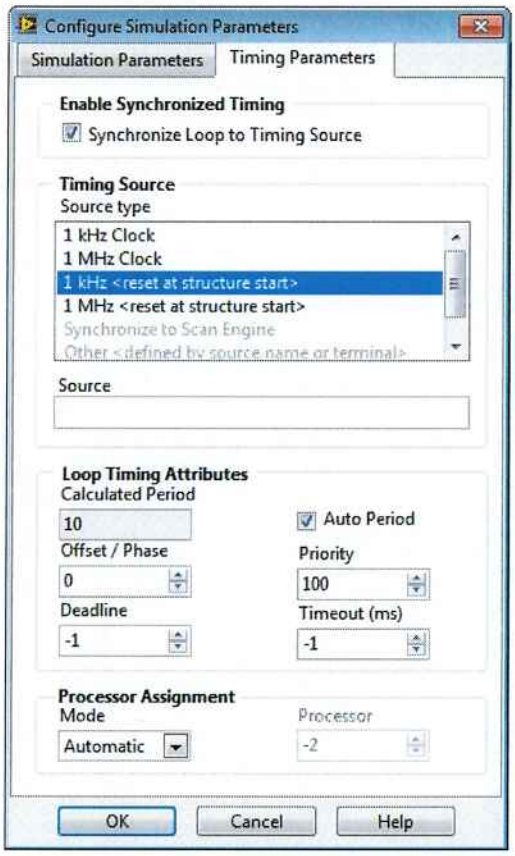
anterior) para que a simulação não tenha tempo final, ou seja, que continue rodando até o usuário parar o processo.

h) Em seguida vem o lugar onde se escolhe o método de resolução das equações diferenciais. Nossos testes mostraram que para o exemplo do Reator Químico, o método Runge-Kutta 4 foi o que forneceu as respostas mais rápidas e apresentou menos atraso para menores tempos de passo. Aconselhamos que seja usado um método de passo fixo para a simulação, mesmo que seja necessário um tempo de passo ligeiramente maior. Mantenha o box *Nan/Inf Check* selecionado.

i) Em seguida devemos escolher o tempo de passo. A simulação vem com o tempo default de 1s. Deve-se reduzir esse tempo de passo e ficar de olho no controle colocado pela diferença entre o *Simulation Time* e o contador do loop for. Deve-se encontrar um tempo de passo suficientemente pequeno, porém que forneça erro zero, ou seja, que o CompactRIO esteja considerando como simulation time exatamente o tempo real. O parâmetro do tempo de passo está vinculado a outra opção que está na outra aba e comentaremos em seguida.

j) A outra aba, de nome "*Timing parameters*" deve ser selecionada, a figura a seguir é sua representação:

Figura 45 - Timing Parameters



Fonte: Autores

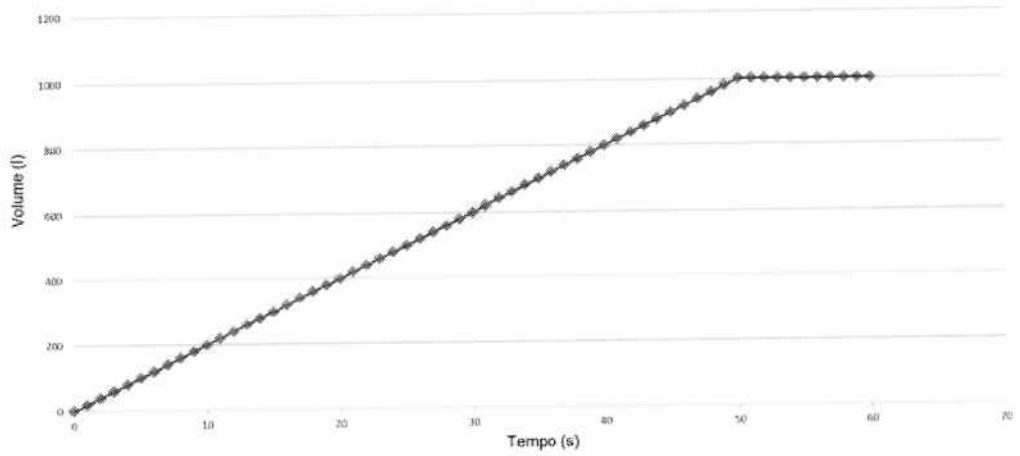
k) Nesta janela deve-se selecionar a opção "*Syncronize Loop To Timing Source*". Em seguida, deve-se selecionar um Timing Source de 1kHz. Todos nossos testes com 1MHz resultaram em erros. Na parte do *"Loop Timing Attributes"* deve-se selecionar "*Auto Period*". Isso é fundamental, pois dessa forma a simulação ajusta o quanto ela calcula ao tempo de passo, resultando no valor correto e em tempo real. As demais configurações devem seguir a figura anterior.

l) Após o termino dessas configurações resta testar a simulação com diferentes tempos de passo. Com a configuração do "*Auto Period*" essa tarefa fica mais fácil, restando verificar no *Front Panel* quais valores que fornecem uma simulação em tempo real. É importante que durante a simulação se executem todas as tarefas que podem ocorrer, contando com as entradas e saídas, para se certificar que para o pior caso de processamento não há atrasos.

6 TESTES E RESULTADOS

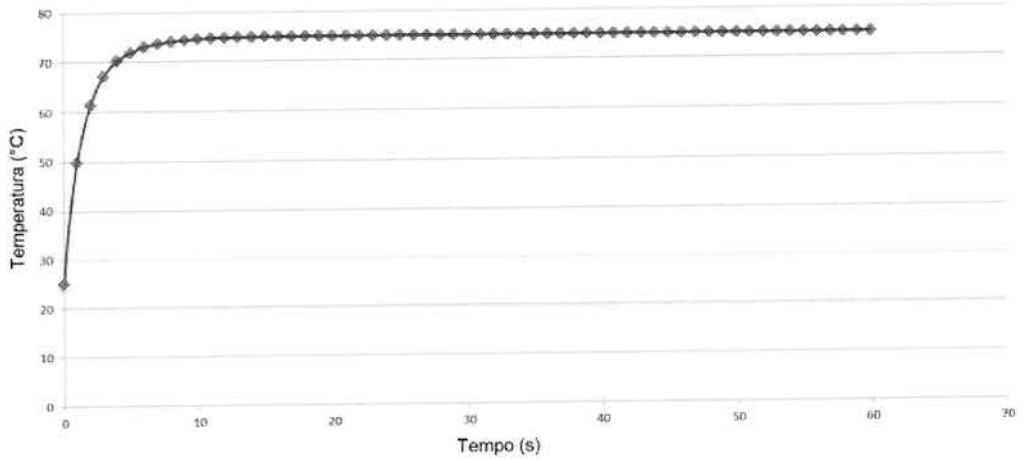
Diversos testes foram realizados para garantir que o CompactRIO fornece a resposta esperada, ou seja, a mesma resposta obtida na simulação feita inicialmente no MATLAB. Neste teste, a válvula do reagente A é aberta no início e nada mais é feito. Nos gráficos representados nas figuras 46 (Volume do tanque) e 47 (Temperatura da mistura), a linha contínua representa a simulação obtida no MATLAB, com um ponto a cada 10 ms, já os pontos que a sobrepõe, representam o resultado obtido com a simulação no CompactRIO, com apresentação de uma amostra a cada cem (espaçadas de 1s), para possibilitar melhor visualização. Repare como o resultado é exato, não há absolutamente nenhuma discrepância entre as curvas.

Figura 46 - Gráfico Volume x Tempo



Fonte: Autores

Figura 47 - Gráfico Temperatura x Tempo

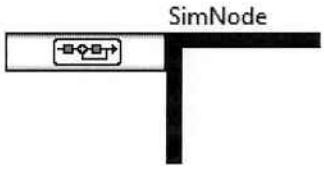


Fonte: Autores

Apesar das respostas estarem todas de acordo com o esperado, ao diminuirmos o tempo de passo da simulação de forma exagerada, começaram a haver discrepâncias entre o tempo de simulação e o tempo real, que não foram reportadas pelo LabVIEW. Tais discrepâncias nos levaram a averiguar mais a fundo o funcionamento dos blocos e o porquê de tais atrasos.

Após a conversão de um arquivo ".mdl" para o LabVIEW pela ferramenta embutida no add-on *Control Design and Simulation*, o Diagrama de Blocos equivalente é representado no LabVIEW dentro de um bloco de nome SimNode. Tal bloco engloba a simulação e a restringe lá, ou seja, não é possível retirar partes do Diagrama de Blocos convertido para o *workspace* comum do LabVIEW. Isso se dá pois existem certos blocos menores como integradores que são específicos para esse tipo de conversão e só funcionam dentro de um envoltório de simulação

Figura 48 - SimNode



Fonte: Autores

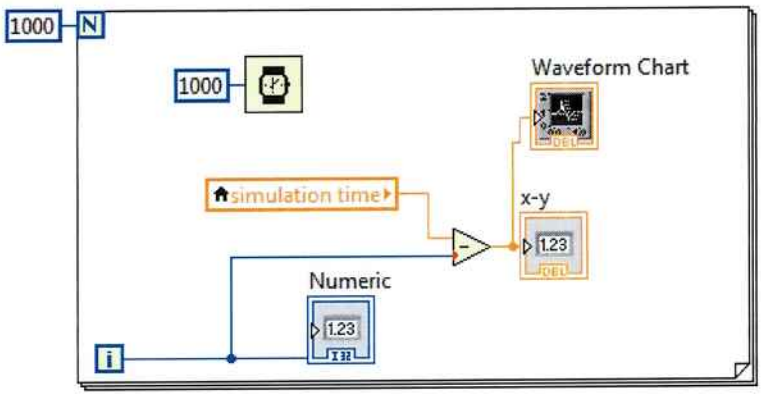
O SimNode conta com algumas opções de simulação como o tempo de início e fim de simulação, o método de solução das equações diferenciais, o tempo de passo e tolerância (em caso de tempo variável), dentre outros. Ao ser convertido, o arquivo nos fornece tempo de passo inicial de um segundo.

Para uma aplicação em tempo real, deve-se ter um tempo de passo pequeno o suficiente para que a simulação mantenha-se fiel à planta que se deseja simular. Começamos a fazer testes então com os diferentes tipos de métodos de solução das equações diferenciais, bem como com o tempo de passo da simulação. Percebemos então que começaram a haver discrepâncias entre o tempo reportado pelo programa e o tempo real, que conta em um relógio. Ao reduzir drasticamente o tempo de

passo, digamos 1 milissegundo (que já sabemos de antemão ser um valor exageradamente pequeno e que uma CPU daquele tamanho nas condições atuais de tecnologia e que não seria capaz de realizar os cálculos a tempo) percebemos que o programa não reporta nenhum tipo de erro avisando o usuário que não foi possível realizar o cálculo em tempo, simplesmente atrasa seu relógio de forma a calcular o valor correto, mesmo que com atraso em relação ao mundo real.

Isso representou um problema para nosso trabalho, uma vez que o intuito é realizar uma simulação em tempo real. Para avaliarmos esse problema de forma mais precisa, decidimos então variar o tempo de passo e compará-lo ao tempo real, estampando na tela da simulação a diferença entre eles, para tentar encontrar um valor tal que a o CompactRIO fosse capaz de realizar os cálculos a tempo. Para isso fomos obrigados a contar os segundos com um "Loop FOR" da maneira que está representado na figura a seguir, uma vez que o LabVIEW não fornece maneiras mais eficientes para a contagem do tempo.

Figura 49 - Contagem de tempo através do Loop FOR



Fonte: Autores

O tempo de simulação, por sua vez, é fornecido por um bloco do *próprio Control Design and Simulation*, que se chama "*Simulation Time*" e funciona dentro do SimNode

Figura 50 - Bloco Simulation Time

Fonte: Autores

Percebemos então duas coisas. O melhor método de cálculo das equações diferenciais para nossa simulação foi o Runge-Kutta 4, com passo fixo. Além disso, percebemos que a diferença entre o tempo de simulação e o tempo real começavam a se aproximar de zero com tempo de passo em torno de 10 milissegundos, o que já é aceitável para nossa proposta.

7 DIFICULDADES

Acreditamos ser bastante importante comentar acerca das dificuldades do percurso, pois se tratando de um trabalho que servirá como uma espécie de manual de ajuda às futuras instalações que se utilizam desse método no laboratório de automação, é importante que se tenha em mente que tipos de contratempo deve-se esperar podendo, com um guia em mãos, evitá-los.

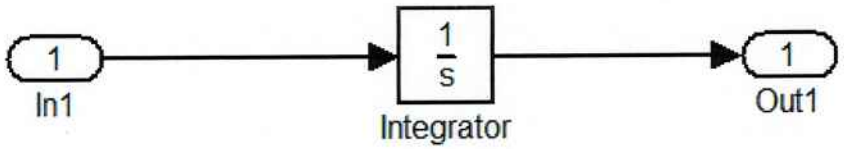
Contamos com diversos problemas de conexão entre o LabVIEW e o CompactRIO. Por diversos dias tentamos sanar os problemas, que só foram resolvidos quando desinstalamos e instalamos o LabVIEW e todos seus componentes (sem poder de fato entender qual era o problema).

É importante comentar que as dificuldades com os *softwares* e *hardwares* foram um ponto que tomou bastante tempo, principalmente pois o *download* de tais módulos é demorado, uma vez que são grandes arquivos de instalação (o arquivo de *drivers* tem mais que 3 *Gigabytes*) e aconteceram vezes de mesmo concluído o *download* com êxito ocorrer falha na instalação, segundo o erro por "arquivo corrompido".

Outro ponto importante de se ressaltar são as conexões físicas que devem ser realizadas nos cartões de saída. Mesmo que o programador execute um código correto, se não se atentar às ligações de terminais comuns e tensões de referência, erros ocorrerão, e sua identificação pode ser demorada. Para facilitar essa parte incluímos, no APÊNDICE A - Alocação de pinos nos cartões de entradas e saídas, instruções para tais ligações.

Com o modelo do MATLAB em mãos e todos os módulos do LabVIEW instalados e funcionando, pudemos tentar a conversão de um para o outro. Nossa idéia inicial foi tentar antes com um modelo mais simples, para entender aos poucos como funciona esse módulo de conversão. Fizemos então um simples integrador no Simulink, como a seguir:

Figura 51 - Integrador no Simulink



Fonte: Autores

Porém, não obtivemos êxito. O LabVIEW simplesmente informava um problema e logo em seguida fechava o programa. Tentamos então com vários outros modelos simples, como somadores de dois *inputs*, também sem sucesso.

Tentamos então com o modelo de simulação do reator químico, um arquivo muito maior que esses e, para nossa surpresa, a conversão funcionou. Não fomos capazes de entender o porquê do erro seguido de fechamento do programa para alguns arquivos, principalmente os menores.

8 CONCLUSÃO

A proposta principal, o oferecimento de uma simulação HIL para o laboratório didático, foi alcançada. Ao longo do trabalho, foram ilustrados os passos necessários para o desenvolvimento e garantia de respostas precisas e qualidade da simulação. O modelo do reator químico obteve um funcionamento robusto quando em conjunto com o CLP, com pouco ou nenhum atraso do tempo simulado após o ajuste dos parâmetros de cálculo do LabVIEW.

O emprego do método apresentado, conversão de um modelo do Simulink para o LabVIEW e a utilização deste modelo no controlador CompactRIO, se mostrou viável para aplicações de baixa ou média complexidade e sem necessidade de uma alta taxa de atualização. O melhor *step-time* para o modelo do Reator Químico foi de 10ms, enquanto simuladores digitais de alta performance, como o *RTDS Simulator* da *RTDS Technologies*, alcançam taxas de 50μs para modelos de sistemas elétricos. A melhor saída para a simulação de modelos complexos no CompactRIO é desenvolvê-los totalmente no LabVIEW, para a utilização de todos os blocos e estruturas que o programa oferece. Entretanto, o ganho de performance requere um tempo muito maior de desenvolvimento do modelo, ponto que o Simulink se mostrou mais eficiente.

É necessário apontar algumas das dificuldades que encontramos. A estrutura de blocos SimNode, apesar de simplificar a construção do modelo no LabVIEW e habilitar a conversão de modelos do Simulink, mostrou-se pouco transparente. A estrutura garante que todos os valores de saída estejam corretos, mas não garante o *step-time* configurado. O LabVIEW não dá alertas quanto a discrepância entre o tempo de simulação e o tempo real. Foi apresentada uma solução para o problema, mas o funcionamento do programa é não é ideal caso seja necessário a garantia do tempo real. Além disso, a estrutura proíbe a utilização de blocos básicos do LabVIEW, o que dificulta o ajuste fino do modelo após a conversão.

Vale ressaltar o caráter prático do trabalho, que se propõe a mostrar o caminho para o desenvolvimento de novas experiências. Espera-se que o trabalho contribua para a melhoria do aprendizado dos alunos em laboratório, pois oferece

uma opção mais visual e concreta dos problemas que um engenheiro pode enfrentar na vida profissional. As limitações do método apresentado podem ser resolvidas, em grande parte, com criatividade no desenvolvimento dos modelos. Dessa maneira, gostaríamos de deixar um legado positivo no fim da nossa graduação.

REFERÊNCIAS

EUTECH SCIENTIFIC ENGINEERING. **Thermolib FAQ,** 2011. Disponível em: <http://www.eutech-scientific.de/fileadmin/inhalte/pdf/products_services/tools/thermolib/Thermolib-FAQ-130301.pdf>. Acesso em: 30 nov. 2014.

FLYNN, D. **Thermal power plant simulation and control,** 2003. Disponível em: <http://www.125books.com/inc/pt4321/pt4322/pt4323/pt4324/pt4325/data_all/books/T/Thermal_Power_Plant.pdf>. Acesso em: 30 nov. 2014.

HALVORSEN, H. P. **Control and Simulation in LabVIEW,** 2011. Disponível em: <http://home.hit.no/~hansha/documents/labview/training/Control and Simulation in LabVIEW/Control and Simulation in LabVIEW.pdf>. Acesso em: 30 nov. 2014.

HALVORSEN, H. P. **Hardware-in-the-Loop-Simulation,** 2011. Disponível em: <http://home.hit.no/~hansha/documents/lab/Lab Work/HIL Simulation/Background/Introduction to HIL Simulation.pdf>. Acesso em: 30 nov. 2014.

HEMALATHA, B.; JULIET, A. V.; NATARAJAN, N. **Boiler level control using LabVIEW,** 2010. Disponível em: <http://www.ijcaonline.org/journal/number17/pxc387540.pdf>. Acesso em: 30 nov. 2014.

LEDIN, J. A. **Hardware-in-the-Loop Simulation**, 1999. Disponível em: <http://www.idsc.ethz.ch/Courses/embedded_control_systems/Exercises/Hardware-in-the-Loop.pdf>. Acesso em: 30 nov. 2014.

MOREIRA, E.; PANTONI, R.; BRANDAO, D. **Equipment based on the Hardware In The Loop (HIL) concept to test automation equipment using plant simulation,** 2011. Disponível em: <http://www.intechopen.com/download/get/type/pdfs/id/17625>. Acesso em: 30 nov. 2014.

NATIONAL INSTRUMENTS. **Ambiente gráfico de desenvolvimento de sistemas LabVIEW**. Disponível em: <http://www.ni.com/labview/pt/>. Acesso em 30 nov. 2014.

NATIONAL INSTRUMENTS. **Controle e simulação utilizando hardware reconfigurável e NI LabVIEW,** 2011. Disponível em: <http://www.ni.com/white-paper/4566/pt/>. Acesso em: 30 nov. 2014.

NATIONAL INSTRUMENTS. **Operating instructions and specifications NI 9205,** 2009. Disponível em: <http://www.ni.com/pdf/manuals/374188d.pdf>. Acesso em: 30 nov. 2014.

NATIONAL INSTRUMENTS. **Operating instructions and specifications NI 9421/9423,** 2009. Disponível em: <http://www.ni.com/pdf/manuals/373504f.pdf>. Acesso em: 30 nov. 2014.

NATIONAL INSTRUMENTS. **Operating instructions and specifications NI 9264,** 2009. Disponível em: <http://www.ni.com/pdf/manuals/374404e.pdf>. Acesso em: 30 nov. 2014.

NATIONAL INSTRUMENTS. **Operating instructions and specifications NI 9472/9474,** 2009. Disponível em: <http://www.ni.com/pdf/manuals/373509e.pdf>. Acesso em: 30 nov. 2014.

PELLINI, E. L. **Controle de reator químico**, v 2.1, p 2, 2013.

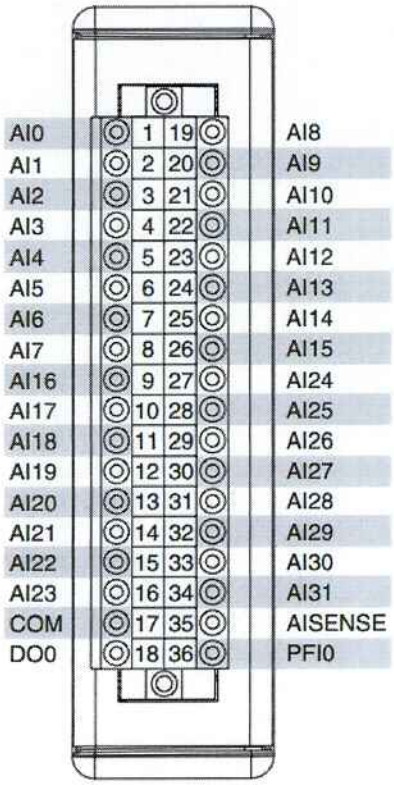
YAZDI, A. F. **Modeling industrial chemical processes with MATLAB and Simulink at HUGO PETERSEN GmbH,** 2012. Disponível em: <http://www.mathworks.com/company/newsletters/articles/modeling-industrial-chemical-processes-with-matlab-and-simulink-at-hugo-petersen-gmbh.html>. Acesso em: 30 nov. 2014.

APÊNDICE A - Alocação de pinos nos cartões de entradas e saídas

- **Entradas Analógicas (NI 9205)**

O terminal COM deve ficar conectado a um terra comum.

Figura 52 - Relação de pinos das entradas analógicas



Fonte: National Instruments (2014)

Pode-se ligar também pares diferenciais de entrada analógica de acordo com a tabela a seguir:

Figura 53 - Tabela de pares diferenciais de entrada analógica

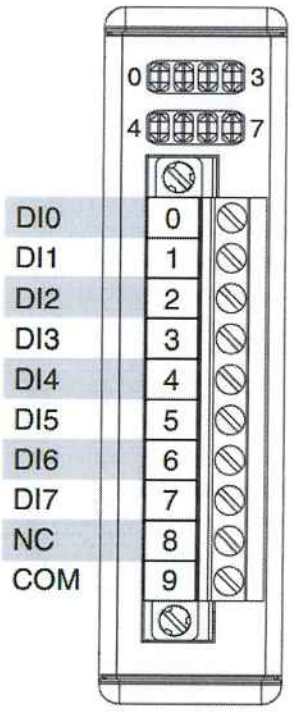
Channel	Signal+	Signal–	Channel	Signal+	Signal–
0	AI0	AI8	16	AI16	AI24
1	AI1	AI9	17	AI17	AI25
2	AI2	AI10	18	AI18	AI26
3	AI3	AI11	19	AI19	AI27
4	AI4	AI12	20	AI20	AI28
5	AI5	AI13	21	AI21	AI29
6	AI6	AI14	22	AI22	AI30
7	AI7	AI15	23	AI23	AI31

Fonte: National Instruments (2014)

- **Entradas Digitais (NI 9421)**

O terminal COM deve ficar ligado a um terra comum.

Figura 54 - Relação de pinos de entradas digitais



Fonte: National Instruments (2014)

O CompactRIO reconhece entradas segundo a tabela a seguir:

Figura 55 - Relação de tensão nas entradas digitais

OFF state

Input voltage ..... ≤5 V

Input current (NI 9421)..... ≤300 µA

Input current (NI 9423)..... ≤150 µA

ON state

Input voltage ..... 11–30 V

Input current..... ≥3 mA

Fonte: National Instruments (2014)

- **Saídas Analógicas (NI 9264)**

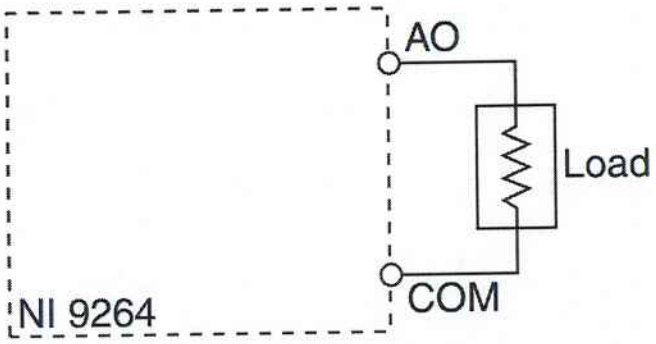
Todos os terminais COM estão internamente conectados a um terra comum. Existem terminais COM adicionais.

Figura 56 - Relação de pinos de saídas Analógicas

Fonte: National Instruments (2014)

Cada saída deve ser ligada a uma carga como na figura a seguir:

Figura 57 - Exemplo de ligação de uma saída analógica

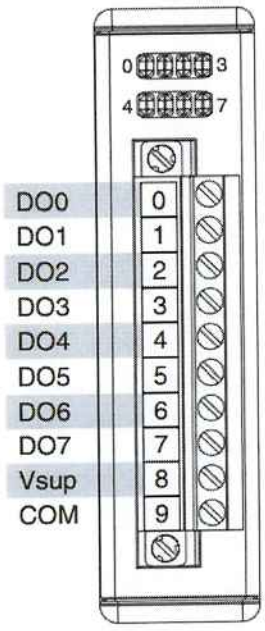


Fonte: National Instruments (2014)

- **Saídas Digitais (NI 9472)**

Este módulo deve ser alimentado com a tensão DC que se deseja fornecer na saída digital entre os terminais Vsup e COM. Como utilizamos 24V DC, alimentamos este módulo em paralelo ao CompactRIO.

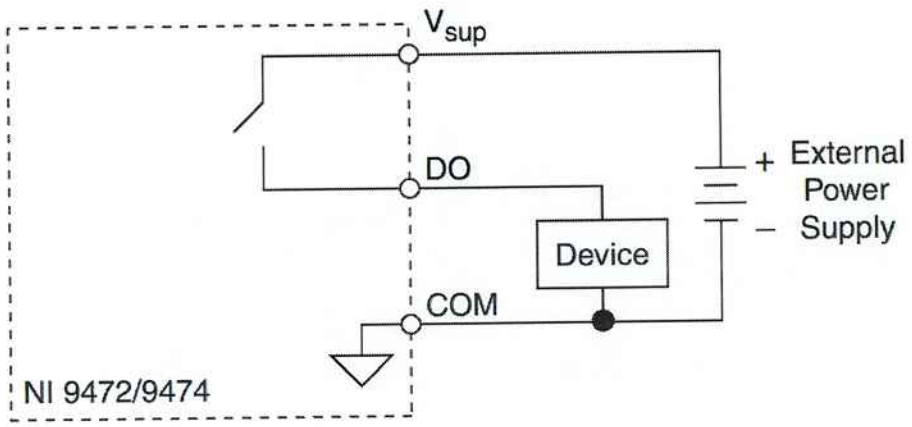
Figura 58 - Relação de pinos de saídas digitais



Fonte: National Instruments (2014)

O módulo funciona segundo a figura que segue:

Figura 59 - Ligação de uma saída digital



Fonte: National Instruments (2014)

- **Informações adicionais**

Todas as informações contidas nesse apêndice foram tiradas de *datasheets* dos componentes. Para informações adicionais e mais específicas e para cuidados que devem ser tomados, consulte nos links que seguem (acesso em 30/11/2014):

- Módulo NI 9205 (Entradas Analógicas)

  http://www.ni.com/pdf/manuals/374188d.pdf

- Módulo NI 9421 (Entradas Digitais)

  http://www.ni.com/pdf/manuals/373504f.pdf

- Módulo NI 9264 (Saídas Analógicas)

  http://www.ni.com/pdf/manuals/374404e.pdf

- Módulo NI 9472 (Saídas Digitais)

  http://www.ni.com/pdf/manuals/373509e.pdf