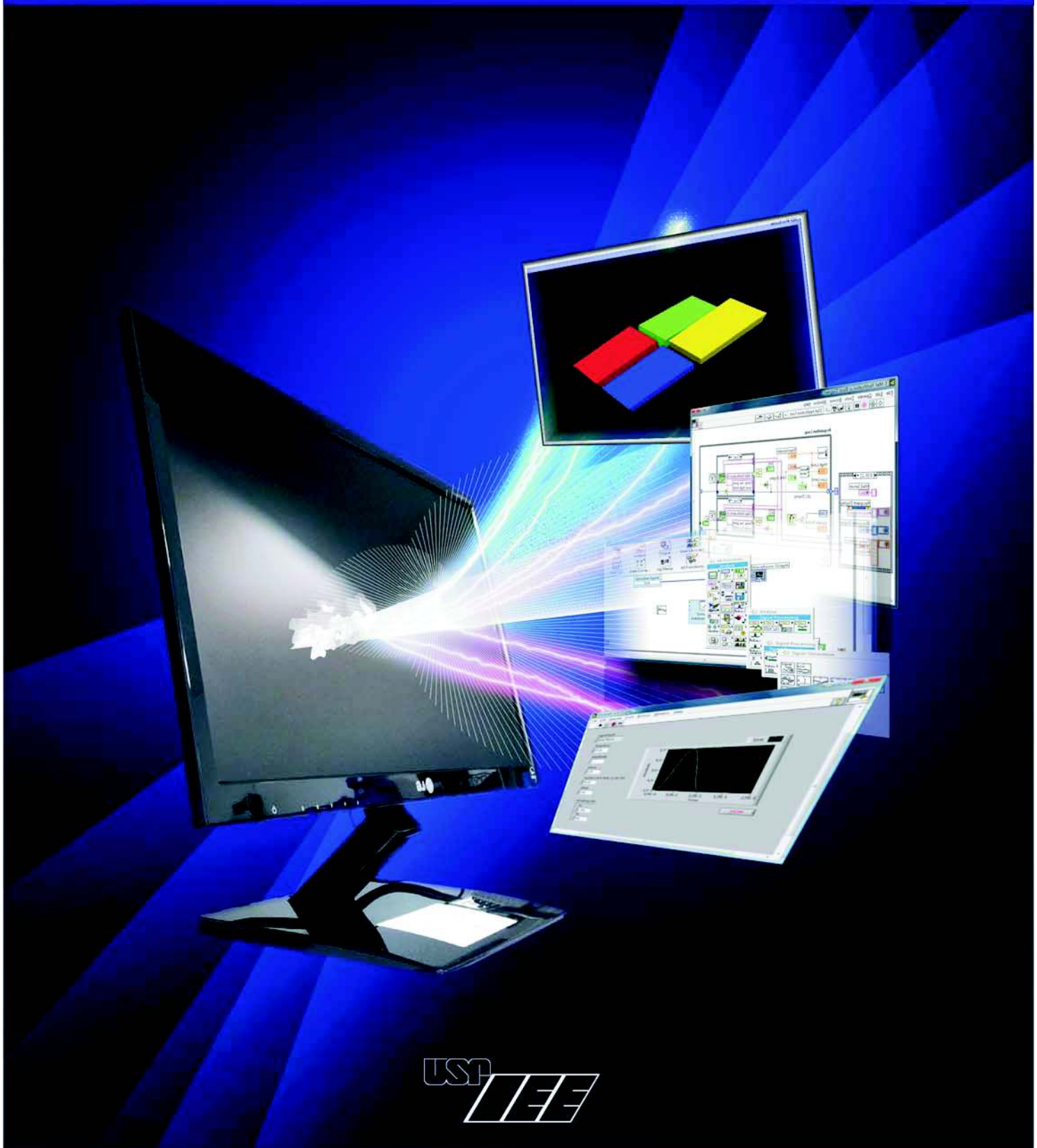
Instrumentação Virtual em Laboratório
Introdução ao LabView



Por Márcio Bottaro

Márcio Bottaro

Instrumentação Virtual em Laboratório

Introdução ao LabVIEW

São Paulo
IEE-USP

ARTE DA CAPA E CONTRACAPA

Regina Célia Barboza
Editoração Eletrônica e Publicações

FICHA CATALOGRÁFICA

B751
Bottaro, Márcio .
Instrumentação virtual em laboratório : introdução ao LabVIEW./ Marcio Bottaro. – São Paulo : IEE-USP, 2012.
125p.

ISBN **978-85-86923-27-2**

1 Instrumentação virtual 2. Software I. Título.

615.47

Edição 01

Márcio Bottaro - Possui graduação em Tecnologia em Saúde pela Faculdade de Tecnologia de Sorocaba (1995), pós-graduação em Engenharia da Qualidade pela Escola Politécnica da Universidade de São Paulo (1999) e Mestrado (2007) e Doutorado (2012) em Tecnologia Nuclear - Aplicações, pelo Instituto de Pesquisas Energéticas Nucleares (IPEN) da Universidade de São Paulo. Tem experiência nas áreas de Instrumentação Eletrônica, Engenharia Biomédica e Física Médica. Concentra suas atividades em pesquisas relacionadas a segurança e desempenho essencial de equipamentos e materiais elétricos, no desenvolvimento de instrumentos, dispositivos eletrônicos, métodos de ensaio e softwares.

Sumário

Exercícios

1 APRESENTAÇÃO

Em 1995, com o advento do primeiro laboratório de ensaios em equipamentos eletromédicos do Brasil, o IEE-USP, a demanda de serviços tecnológicos no setor começara a crescer e a necessidade de automação dos processos de ensaio para atendimento às necessidades nacionais era prioritária. Naquele momento, a automação consistia basicamente no desenvolvimento de circuitos eletrônicos de controle e aquisição de dados digitais, utilizando-se os guias da IBM para desenvolvimento de hardwares acopláveis a PCs da família 286 e 386 por meio de seus barramentos ISA, padrão inexistente na atualidade. Posteriormente, circuitos eletrônicos para conversão de dados analógicos de baixíssima amplitude, realidade dos sinais tratados pelos equipamentos eletromédicos nos aspectos de desempenho e segurança elétrica, foram desenvolvidos para atuação com os hardwares digitais. A complexidade dos endereçamentos estabelecidos manualmente e o grande tempo de montagem e teste destes hardwares já geravam grande entrave em sua aplicação em maior escala, o que limitaram a aplicação destes circuitos na época, a alguns ensaios mais críticos.

A interação do hardware com o usuário era toda desenvolvida em ambiente de programação DOS, utilizando-se linguagens como PASCAL e BASIC, evoluindo posteriormente no sistema Windows para linguagens como Visual Basic e Delphi e mais tarde para o C++ e C#. As DLLs desenvolvidas em Pascal eram o link entre o ambiente de programação de alto nível e o hardware que assumia possibilidades limitadas de programação, mas atendia basicamente as necessidades urgentes. O desenvolvimento destes softwares levavam meses e algumas vezes ultrapassavam um ano e sua validação era ainda bastante sofisticada, uma vez que as simulações eram efetuadas na prática dos ensaios laboratoriais, o que contribuía para elevar ainda mais o tempo de desenvolvimento.

Esta realidade estendeu-se até meados de 2001, quando o bolsista de graduação, físico João dos Santos Justo Pires, integrou a Seção Técnica de Ensaios em Equipamentos Eletromédicos do IEE e trouxe a notícia do LabVIEW, plataforma de programação gráfica da National Instruments, que caminhava naquele momento para sua versão de número 6.1. João Pires começou a explorar o LabVIEW na época mas sentiu a resistência de todos em adotar aquela plataforma, nomeada pelos mais antigos, inclusive eu, como “caixa preta” de programação, ou ainda “LEGO da engenharia”, o que viria a tornar-se realidade mais tarde com o LabVIEW para LEGO MINDSTORMS. O tempo de desenvolvimento, no entanto, era seu grande atrativo e começava a despertar nossa atenção.

Em 2002, em uma visita técnica ao John Perry Laboratory, integrado ao Saint Georges Hospital em Londres, aplicações de calibração de medidores de radiação X já estavam automatizadas e mais uma vez o LabVIEW mostrava as suas vantagens. Ao perguntar ao Físico Responsável pelo local, David Arnold, sobre o tempo de desenvolvimento e integração da tecnologia, os surpreendentes 6 meses de resposta foram intrigantes.

Dando continuidade em 2003 com as linguagens tradicionais e agora com a implementação de Java e necessidade de sistemas microcontrolados para aumentar o grau de complexidade das aplicações, o surgimento do LabVIEW 7.0 express foi determinante para pensarmos em começar a mudar. E foi o que aconteceu. Juntando esforços com João Pires passamos a desenvolver aplicativos para utilização em laboratório e ainda, com a aquisição de hardware para controle e medição, os primeiros projetos passaram a ser desenvolvidos com tempos impressionantemente reduzidos e com resultados extremamente satisfatórios. Aliados aos programas de medição e controle, simulações para validação anteriores a aplicação prática, proporcionadas pela plataforma LabVIEW, davam ainda mais credibilidade ao processo.

Com os primeiros resultados da credibilidade da plataforma em aplicações laboratoriais, inclusive as acreditadas pelo INMETRO, os primeiros treinamentos internos coordenados pelo Físico João Pires, na época referência interna de LabVIEW no IEE, começaram a ser ministrados e logo, com uma parceria com a empresa CNZ Indústria e Comércio Ltda., cursos externos também começaram a ser aplicados, principalmente no IAE (Instituto de Aeronáutica e Espaço) e ITA em São José dos Campos, onde meu papel principal eram as aplicações em laboratório, com a experiência de vários anos, e as integrações com eletrônica, planilhas de ensaio e incerteza de medição em hardwares de aquisição de dados.

Em 2006, com a primeira publicação envolvendo utilização de LabVIEW em ambiente laboratorial para controle e aquisição de dados, o Analisador de Marcapassos cardíacos, verifiquei a necessidade de elaborar um material realmente adequado para abordar os conhecimentos básicos de inserção na plataforma e posteriormente integração com instrumentação, hardwares de controle e aquisição de dados e outras plataformas eletrônicas de processamento de dados e emissão de relatórios. De 2007 a 2011, infelizmente sem o colega João Pires que dava andamento em sua vida acadêmica longe do IEE, vários cursos em parceria com a CNZ foram ministrados, além de participações em congressos e conferências, entre elas o NI Days, publicações de artigos científicos utilizando-se o NI LabVIEW e por fim uma tese de doutorado na plataforma LabVIEW FPGA, consolidaram este material que visa dar os primeiros passos para quem pretende iniciar ou aprimorar a utilização desta surpreendente plataforma de programação gráfica em rotinas de ensaios de desempenho e segurança.

Dr. Márcio Bottaro

2 INTRODUÇÃO

A pesquisa e o desenvolvimento são as ferramentas mais importantes para o crescimento sustentável de qualquer área tecnológica. Nas últimas décadas, várias ferramentas computacionais foram desenvolvidas e muitas linguagens de programação foram ganhando características otimizadas e versatilidade que permitiram grande avanço nos ambientes computacionais. O mesmo pode-se dizer da eletrônica em termos de desenvolvimento de microprocessadores, microcontroladores, DSPs e FPGAs, que atingiram proporções notáveis permitindo que a eletrônica embarcada em equipamentos e instrumentos pudesse contribuir com recursos operacionais extremamente avançados com grande capacidade de processamento e melhor interatividade com o usuário. Os custos destes dispositivos também foram reduzidos e permitiram um maior acesso a muitas dessas tecnologias, mesmo pelas empresas de menor porte.

A escolha da ferramenta computacional depende essencialmente da expertise dos envolvidos nas atividades de pesquisa e desenvolvimento. Não existe forma mais eficiente de contribuição para o desenvolvimento de um processo do que a participação do desenvolvedor ou pesquisador no mesmo. Quando conseguimos aliar conhecimentos específicos do processo a capacidade de desenvolvimento em uma equipe técnica, temos um avanço notável do grupo e uma motivação interna que gera trabalhos de alto nível. Este fato, no entanto, nem sempre é possível devido à complexidade dos sistemas eletrônicos e computacionais, levando à separação do especialista em relação às atividades de desenvolvimento mais complexas. Em poucas palavras podemos dizer que poucos profissionais conseguem atuar de forma direta na rotina de pesquisa em áreas diversas e ter habilidades ou mesmo tempo disponível para o desenvolvimento de sistemas elétricos, mecânicos, eletrônicos e computacionais de controle e aquisição de dados e processos, para unir estas capacidades e transformá-las em resultados reais de operacionalização com ganhos em tempo, custo e qualidade. Na maioria das vezes, na experiência laboratorial, o tempo despendido no trabalho com desenvolvimento de software e hardware é tão grande ou superior ao tempo investido nos objetivos prioritários da pesquisa e desenvolvimento.

Com o advento da programação gráfica, que possui baixo tempo de aprendizado e excelente retorno de resultados, esta aliança de conhecimentos começou a tornar-se possível e ao mesmo tempo estimulante. Os especialistas ou mesmo pesquisadores começaram ter um horizonte para propor soluções reais e com tempo de implementação consideravelmente baixo, acompanhando de forma participativa todas as etapas do processo de desenvolvimento de suas pesquisas.

A linguagem de programação gráfica que mais se destacou na última década foi sem dúvida o LabVIEW ™ da National Instruments. Além de oferecer um ambiente propício ao desenvolvimento de sistemas de automação e controle, amigável a qualquer pessoa da área técnica ou tecnológica, o LabVIEW traz a grande vantagem de um tempo de aprendizado extremamente reduzido, com aplicabilidade facilitada pelos inúmeros hardwares de automação disponibilizados pela própria

National Instruments, que reduzem bruscamente os tempos de desenvolvimento de sistemas eletrônicos. Este fato, no entanto, não restringe a utilização destes sistemas atrelada somente a empresa criadora do LabVIEW, pois mais e mais empresas disponibilizam hoje hardwares e softwares de comunicação para automação e controle já desenvolvidos para compatibilidade com a plataforma LabVIEW.

A possibilidade proporcionada pelo LabVIEW de compor painéis de instrumentos que normalmente estão disponíveis nas bancadas dos laboratórios trouxe o real significado da Instrumentação Virtual. A simulação de dados de entrada e saída que podem ser traduzidos como medição e controle potencializou a Instrumentação Virtual e abriu as portas para as mais diversas aplicações e integrações entre ambiente virtual e real.

2.1 Rotina laboratorial e processo de automação de ensaios

Quando pensamos em laboratórios de calibração e ensaios a primeira coisa que visualizamos são atividades rotineiras conduzidas por procedimentos técnicos e gerenciais, elaborados com base normativa ou mesmo desenvolvidos pelo próprio laboratório, de forma a atender uma demanda prevista de corpos de prova que chegam por meio de solicitações coerentes com o escopo acreditado ou capacitado do laboratório. Quem passou ao menos uma semana em um laboratório sabe que as coisas não ocorrem exatamente desta forma.

A rotina laboratorial é carregada de eventos que podem ser planejados de acordo com a experiência da equipe, mas que não podem ser totalmente previstos. Muitos deles acabam sendo resolvidos pelo talento individual de membros da equipe, especialistas de muitos anos nas áreas de interesse.

Vamos colocar aqui o exemplo de ensaios de desempenho e segurança em equipamentos radiológicos utilizados na área médica, onde a variedade de tamanho, tecnologia, parâmetros e aplicação são imensas. Os procedimentos técnicos para este tipo de ensaio vão desde avaliação documental e inspeções gerais, até ensaios elétricos, mecânicos e radiológicos, e levam em conta aspectos gerais normativos de forma a orientar o trabalho dentro de aspectos de segurança elétrica e radiológica, o uso da instrumentação laboratorial de forma adequada e em linhas gerais, orientação sobre algumas possibilidades já vivenciadas pelo laboratório com determinados tipos de equipamentos.

A primeira conclusão que se chega e que é impossível planejar totalmente um ensaio numa categoria de equipamentos tão diversificada, mas o planejamento de ações comuns a todos pode ser efetuado. Estas devem ser incorporadas aos procedimentos a medida que o laboratório evolui em sua experiência, e a medida do possível devem ser desvinculadas do conhecimento proprietário de alguns ou de um único colaborador da equipe, o que acaba ocorrendo em áreas que requerem grande especialidade.

O exemplo dado anteriormente, envolvendo equipamento de médio e grande porte, não se restringe a esta categoria e pode ser aplicável a um simples componente de um equipamento ou

sistema. O cenário ideal para automação é sempre o de ensaios repetitivos com grande similaridade de corpos de prova, de forma similar aos ensaios de rotina de uma linha de produção, porém esta não é a realidade laboratorial.

Um dos objetivos aqui é mostrar que com alguns ajustes e auxílio das ferramentas adequadas de sistemas e dispositivos, mesmo onde um processo não é exatamente repetitivo também é possível e necessário a implantação de métodos automatizados que tornam a rotina laboratorial mais dinâmica e agradável a seus colaboradores.

A automação de processos de ensaios apresenta ampla variedade de métodos, podendo ser simples como um motor que movimenta um cabo sob um ponto de apoio para avaliar sua durabilidade, ou mesmo extremamente complexa, como um sistema que produz vários sinais elétricos em diferentes faixas e pontos de calibração, e que tem a capacidade de coletar os resultados do corpo de prova efetuando cálculos, propagações de erros e gerando imensos relatórios automáticos, que possuem ainda algoritmos de análise crítica dos resultados e submetem julgamentos sobre o comportamento de determinado instrumento.

As soluções de automação vão desde pequenos hardwares e softwares que auxiliam a tomada de decisões, execução de pequenos cálculos e ações repetitivas que facilitam a rotina laboratorial, que podem ser embarcadas em pequenos sistemas eletroeletrônicos ou eletromecânicos, até sistemas de grande porte que podem se adaptar a condições variadas de operação laboratorial e que necessitam de uma plataforma mais robusta de desenvolvimento e operação.

Neste livro vamos abordar os princípios básicos de programação gráfica da plataforma LabVIEW com exemplos didáticos para inserção no ambiente virtual e com exemplos práticos de rotina laboratorial e sugestões de rotinas e uso de hardware para auxílio nos processos de automação.

3 INTRODUÇÃO AO LABVIEW

3.1 LabVIEW e Virtual Instruments

O LabVIEW, *Laboratory Virtual Instrument Engineering Workbench*, é uma linguagem de programação gráfica, "G", que possibilita a criação de testes (simulação), medição, aquisição de dados (DAQ), controle de instrumentos, armazenamento de dados, análise de dados e geração de relatórios.

A criação em 1986 sob a ideia de um dos fundadores da National Instruments, James Truchard (Mr. T): *"Fazer para a engenharia o que a planilha de cálculo fez para a área financeira"*, e executada por Jeff Kodosh, outro de seus fundadores e que vislumbrou a possibilidade de cria um ambiente de desenvolvimento voltado para engenheiros e cientistas, foi um marco para história da engenharia moderna, e sua inserção nas mais diversas áreas tecnológicas é a prova de seu sucesso.

Os programas em LabVIEW são chamados de "virtual instruments" ou VIs, porque sua aparência e forma de operação imitam instrumentos de bancada como multímetros, osciloscópios, analisadores, etc. A interface com o usuário é construída por meio de um painel frontal com o auxílio de objetos e ferramentas disponíveis.

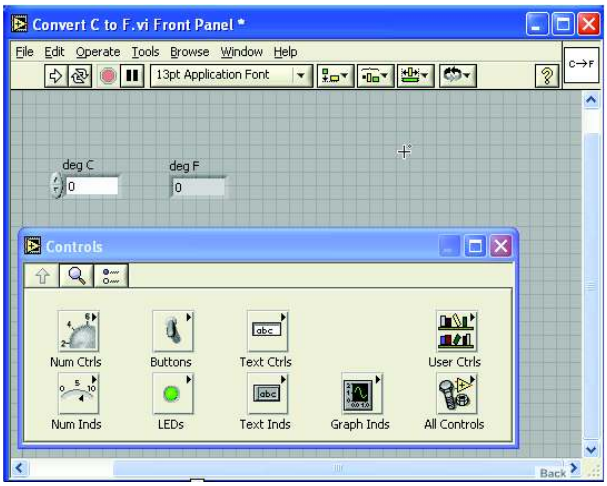


Figura 1 – Exemplo de painel frontal no LabVIEW

A primeira vez que montamos um painel frontal com cerca de uns dois ou três controles e mais uns três ou quatro indicadores, ou mesmo um indicador gráfico, a pergunta é quase sempre a mesma: *"quanto tempo eu levaria para montar isso em uma plataforma de programação convencional?"*.

Associado a cada painel frontal temos um diagrama de blocos para edição do código de execução da VI por meio de funções representadas graficamente.

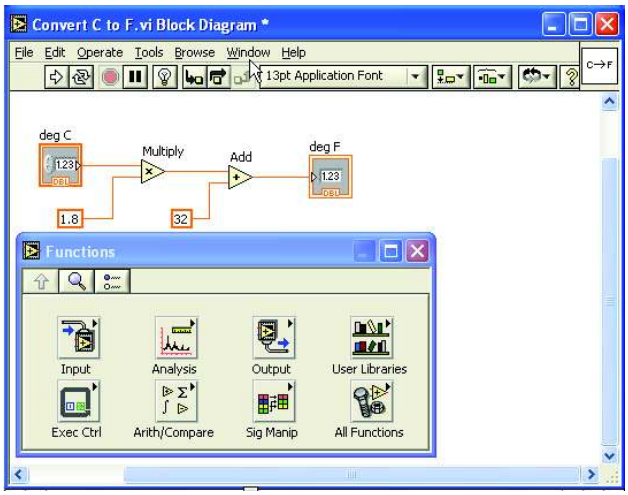


Figura 2 – Exemplo de diagrama de blocos no LabVIEW

As VIs utilizam um sistema hierárquico e uma estrutura modular, ou seja, pode-se criar programas compostos por subprogramas. Uma VI dentro de outra VI é chamada de subVI (sub-rotina).

Com estas características, o LabVIEW se enquadra no conceito de programação modular. Você pode dividir uma aplicação complexa em uma série de sub-tarefas simples, construir uma VI para agrupar cada sub-tarefa e depois as combinar para que uma tarefa mais complexa seja executada. Ao final obtém-se uma VI, contendo várias subVIs, que realiza uma função de alta complexidade. Este conceito é o mesmo utilizado na programação convencional baseada em linhas de comando, onde, em linguagem estruturada, utilizamos sub-rotinas. Utilizando as subVIs fica mais fácil encontrar erros no programa uma vez que cada subVI é separadamente analisada. Estas SubVIs possibilitam ainda a reutilização de códigos em outros projetos, com mais ganhos em tempo de desenvolvimento e validação de resultados.

Uma das características importantes do LabVIEW é ser uma plataforma de programação neutra permitindo migração, traduzindo e recompilando automaticamente os dados entre sistemas operacionais como MAC e Windows.

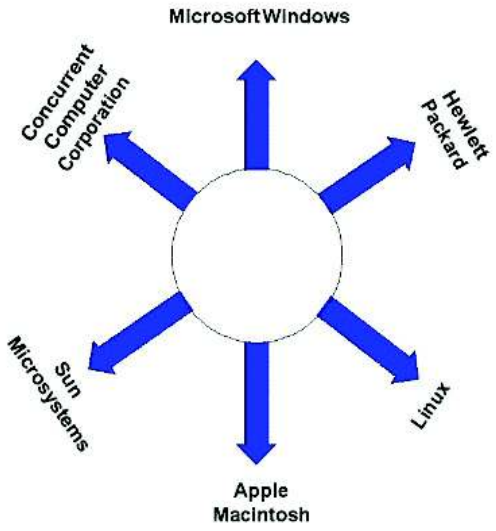


Figura 3 – LabVIEW, uma plataforma neutra em relação ao sistema operacional

O LabVIEW é totalmente integrado para comunicação com hardwares como GPIB, PXI, RS-232, RS-485, USB, ethernet e outros padrões de dispositivos para aquisição de dados (DAQ). Além de ser compatível com todos os hardwares da National Instruments, o LabVIEW possui compatibilidade com vários dispositivos de outras empresas fabricantes de hardwares.

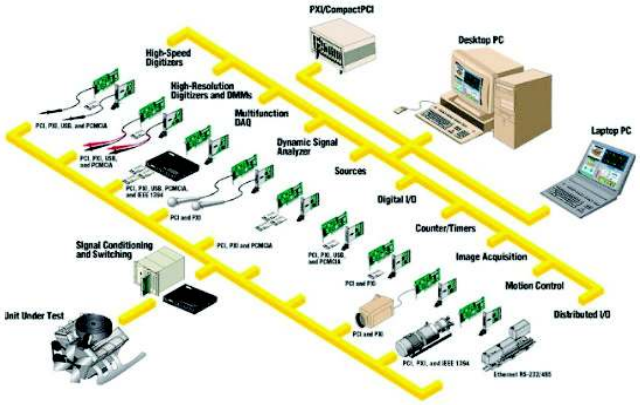


Figura 4 – Conectividade de Hardware do LabVIEW

A conectividade do LabVIEW permite ainda o acesso via WEB com outros dispositivos e sistemas por meio de protocolos TCP/IP e Active X. O acesso a planilhas na plataforma Microsoft Excel por meio do protocolo Active X é um dos grandes atrativos para laboratórios de ensaios que trabalham com documentos eletrônicos e planilhas de dados. Mesmo com o desenvolvimento de softwares como o DIAdem da National Instruments, dedicados a análise de dados e geração de relatórios, e muito mais poderoso em ferramentas e processamento, muitos trabalhos desenvolvidos na plataforma Excel para processamento de dados e formatação de resultados podem ser utilizados

sem a necessidade de retrabalho no desenvolvimento de novos sistemas de geração de relatórios de ensaios.

Um cuidado importante que deve ser observado no uso do Excel como ferramenta de trabalho de laboratório é a validação tanto da inserção de dados, posições das células, quanto da compatibilidade dos mesmos. **Muitas vezes o Excel estabelece truncamento de dados e a informação pode ser perdida ou invalidada**. Exemplos típicos são os processos de cálculos de incerteza de medição e tratamento de grandes volumes de dados, onde, conforme veremos adiante, é mais interessante um processamento prévio por meio das ferramentas disponíveis no LabVIEW para posterior envio de dados às planilhas.

Uma característica interessante da programação na plataforma LabVIEW é que a medida que os conceitos básicos são assimilados, a correlação com problemas da rotina laboratorial, ou de qualquer outra área de trabalho, começa a ser estabelecida e ideias de trabalho de alto nível já podem ser estabelecidas, com a pesquisa dentro do próprio ambiente ou com auxílio de exemplos da web, muitos aplicativos já podem ser desenvolvidos e implementados de imediato. Você poderá compreender melhor esta vantagem no decorrer das seções a seguir.

3.2 O ambiente LabVIEW

O objetivo aqui é apresentar o LabVIEW e introduzir os conceitos iniciais que possibilitam uma primeira interação com a linguagem e criação das primeiras VIs, manipulação de painéis frontais e diagramas de blocos, técnicas de debug, criação de SubVIs, manipulação de strings e entrada/saída de arquivos. Com esta informação inicial, ferramentas de simulação e validação de resultados já podem ser introduzidas e o trabalho de automação propriamente dito já começa a ser desenvolvido. Posteriormente, análise de sinais e processamento de dados associados a necessidade laboratorial serão abordados, bem como o tratamento da incerteza de medição e comunicação com a instrumentação e hardware disponível ou desenvolvido.

A programação começa efetivamente pela caixa de diálogo inicial de navegação do LabVIEW (*NAVIGATION DIALOG BOX*), e é nela que encontramos itens padronizados de acesso, como Arquivo, Saída (*EXIT*), etc., e por onde temos acesso a configurações de ambiente, aquisição de dados, obtenção de ajuda e outros detalhes que serão elucidados no decorrer deste documento.

Figura 5 – Caixa de diálogo inicial de navegação do LabVIEW

Por meio da caixa de diálogo inicial podemos abrir, configurar, criar e salvar os projetos e as VIs que iremos trabalhar daqui para frente.

Quando criamos um novo projeto com LabVIEW, uma caixa de diálogo com informações detalhadas é mostrada. A este projeto podemos vincular VIs e então gerenciar de maneira apropriada uma aplicação futura.

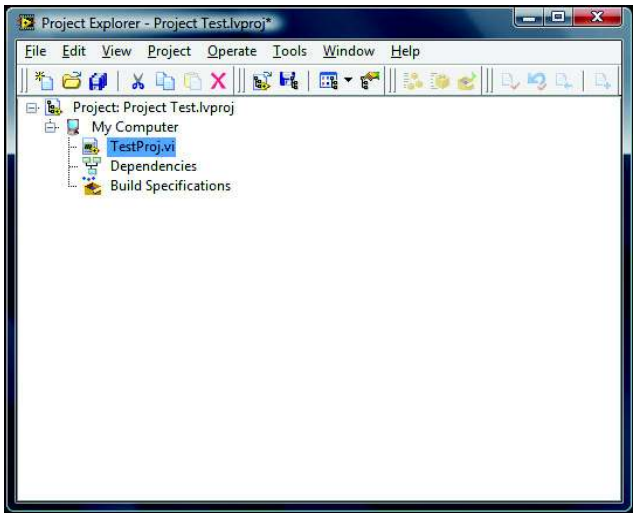


Figura 6 – Caixa de diálogo de projeto – *Project Explorer*

Padrões de arquivos podem ser utilizados e detalhados conforme a destinação da aplicação (*VI from template*), ou uma VI em branco (*Blank* VI) se é desejável iniciar um projeto do "zero". É

sempre interessante vincular nossas VIs ao nosso projeto, o que facilita seu gerenciamento, organização e aplicação. Quando iniciarmos uma escolha de VI proveniente dos modelos do LabVIEW, uma caixa de diálogo com uma visualização prévia de nossa VI será apresentada.

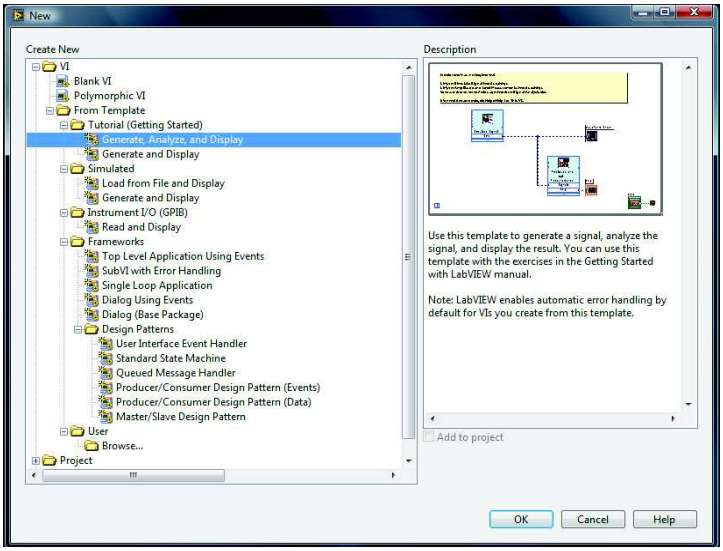


Figura 7 – Inicialização de VI a partir de modelo – VI *from template*

Uma descrição abaixo do *preview* da janela de diagrama de blocos da VI também é apresentada para entendermos basicamente as funções implementadas pela mesma.

Podemos ainda obter VIs provenientes de exemplos do LabVIEW, e estas normalmente estarão disponíveis no conteúdo de ajuda para que possamos explorar sua funcionalidade e utilizá-las em nossas aplicações. Normalmente, a utilização de modelos e exemplos é um bom caminho para iniciarmos nossas aplicações e nos familiarizarmos com o LabVIEW.

Partindo para a utilização do ambiente LabVIEW começaremos com uma *Blank* VI. Nesta VI iremos explorar inicialmente o painel frontal. O painel frontal possui uma barra de ferramentas que é brevemente ilustrada e descrita aqui e que é muito importante no contexto da utilização do LabVIEW:

Onde:


Executar a VI. O LabVIEW compila/recompila e executa a VI.


Indicado quando a VI está em execução.


Execução de subVI.


VI contém erros de edição e não pode ser executada.

 Executa VI indefinidamente.

 Interrompe execução da VI a qualquer momento.

 Pausa na execução da VI.

Opções de formatação de texto.

 Configuração de alinhamento de objetos.

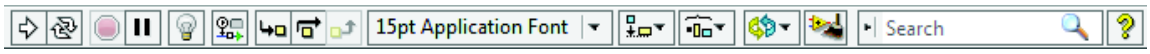
 Configuração de distribuição de objetos.

 Redimensionar objetos.

 Reorganizar objetos sobrepostos.

 Exibe a janela “Context Help”.

Partindo agora para o diagrama de blocos, da mesma forma que no painel frontal, este apresenta uma barra de ferramentas:



Nesta barra de ferramentas temos alguns comandos adicionais que auxiliam no trabalho com o diagrama de blocos:

 Highlight Execution, exibe de forma detalhada e clara o fluxo de dados no diagrama de blocos durante a execução da VI.

 Step into, Step over e Step out, que permitem trabalhar passo-a-passo dentro de loops ou rotinas e sub-rotinas ,sobrepondo e suspendendo as mesmas se necessário.

 Clean Up Diagram, que organiza o diagrama de blocos, reposicionando nodes e wires. É importante utilizarmos esta função com cautela, pois a organização é efetuada com critérios internos do sistema e nem sempre a disposição final é a esperada, ou quase nunca!

3.2.1 Menus do LabVIEW

Os menus do LabVIEW, situados no topo da janela do painel de comando e diagrama de blocos, é comum a outros aplicativos, com funções como Abrir, Salvar, Sair, etc. Outros campos são específicos do LabVIEW e utilizados para o trabalho com os ambientes de edição do programa,

para exibição de ferramentas e para recursos específicos como acesso a VIs, Exemplos, MAX (*Measurement & Automation Explorer*), entre outros. Itens importantes como as paletas de ferramentas, funções e controles são acessíveis por meio dos menus do LabVIEW.

É importante ter sempre em mente que o acesso a propriedades de cada objeto inserido por meio das paletas de controle no painel frontal e diagrama de blocos. Estas propriedades devem sempre ser editadas de forma conveniente a atender especificações de nossa aplicação.

Agora, no painel frontal vamos inserir dois objetos de controle, um numérico e um gráfico, por meio da paleta de controle, e explorar algumas de suas propriedades. A paleta de controle pode ser visualizada por um click (botão direito do mouse) ou por meio do menu VIEW.

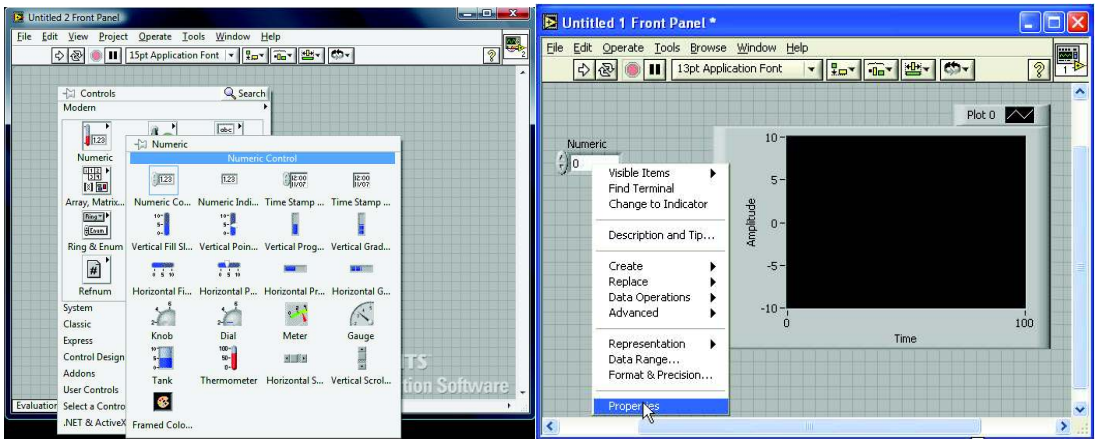


Figura 8 – Objetos do Painel Frontal do LabVIEW

Procure explorar as propriedades do objeto numérico e gráfico para começar a se ambientar com a quantidade de possibilidades oferecidas pelo LabVIEW em seus objetos.

3.2.2 Paleta de ferramentas (tools palette)

A paleta de ferramentas é utilizada para edição do ambiente, incluindo objetos para controle e execução da VI, e a mesma é acessível por meio do menu VIEW.

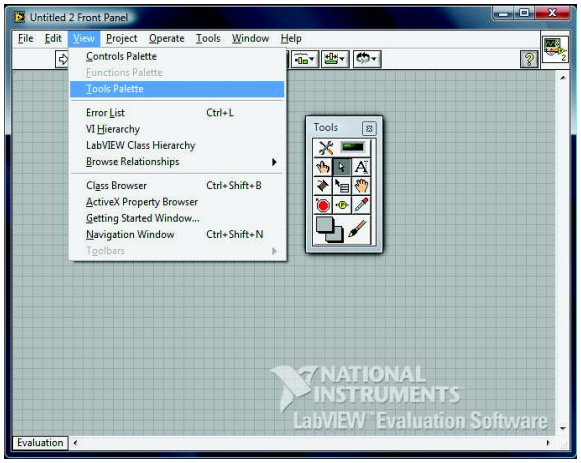


Figura 9 – Paleta de Ferramentas do LabVIEW

 Automatic tool selection: seleciona automaticamente a melhor função para o mouse

 Operating tool: Altera valores e seleciona textos de objetos

 Positioning tool: Seleciona, movimenta e redimensiona objetos

 Labeling tool: Edita textos e cria rótulos livres

 Wiring tool: Conecta objetos dentro do diagrama de blocos

 Object Shortcut Menu tool: acessa o menu de atalho dos objetos

 Scrolling tool: movimenta a imagem sem a utilização da barra de rolagem

 Breakpoint tool: estabelece paradas de execução em funções, nodes, ligações e estruturas

 Probe tool: cria pontas de prova no diagrama de blocos que podem auxiliar o debug de VIs

 Color copy tool: copia rapidamente cores de objetos para colar com o coloring tool

 Coloring tool: utilizado para colorir objetos, com indicações de cor frontal e fundo

3.2.3 Paleta de Controle (Controls Palette)

A paleta de controle consiste de ferramentas como indicadores e controles e ela é utilizada para editar o painel frontal. Esta paleta também é acessível pelo menu *VIEW*. A seguir é mostrada uma

ilustração da paleta de controle. Propriedades de todos os objetos inseridos podem ser acessadas pelo atalho do botão direito do mouse para que detalhes e configurações avançadas, quando disponíveis, possam ser efetuadas.

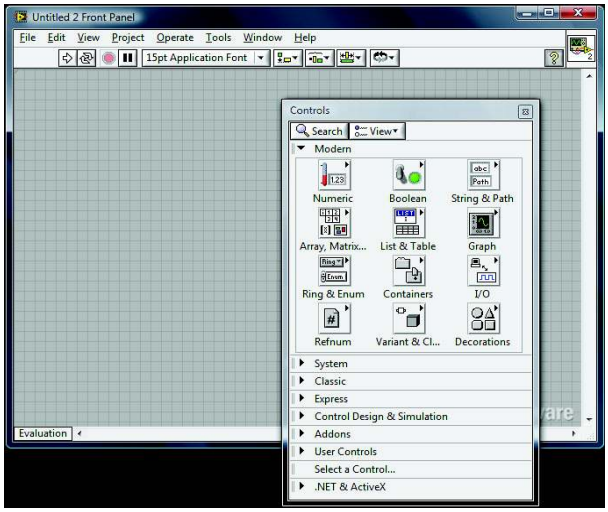


Figura 10 – Paleta de controle do Painel Frontal do LabVIEW

A exploração de cada item constituinte desta palheta de edição do painel frontal será efetuada a medida que você explorar seus projetos e VIs.

Não é necessário decorar a localização de nodes dentro da paleta de controle, a familiarização ocorrerá com o tempo e nos primeiros projetos a busca das funções por meio do botão *Search* pode ser efetuada.

3.2.4 Paleta de Funções

A paleta de funções consiste de ferramentas para programação e é utilizada para editar o diagrama de blocos. A paleta de funções é acessível pelo menu *VIEW* da janela do diagrama de blocos. Da mesma forma que na paleta de controle do painel frontal, propriedades de todos os objetos inseridos podem ser acessadas pelo atalho do botão direito do mouse para que detalhes e configurações avançadas, quando disponíveis, possam ser efetuadas.

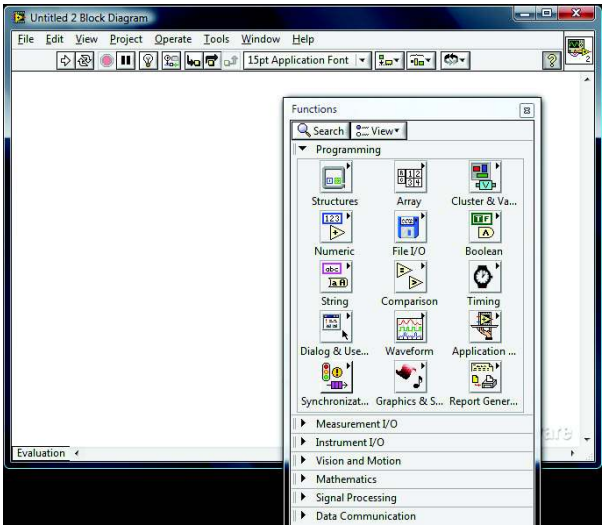


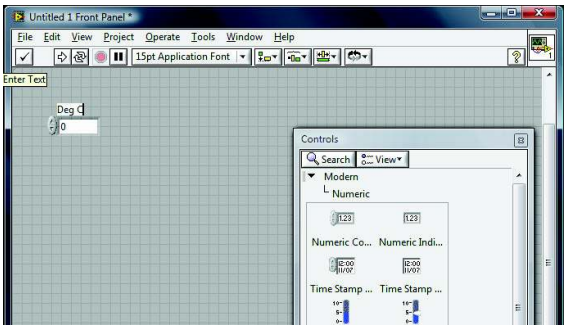
Figura 11 – Paleta de funções do Diagrama de Blocos do LabVIEW

3.2.5 Painel Frontal

Utilizando o menu inicial crie uma *Blank VI*. Serão exibidos seu painel frontal e em segundo plano seu diagrama de blocos. Você pode ainda criar um novo projeto e neste inserir uma nova VI em branco para gerenciar o trabalho com o LabVIEW daqui para frente.

No painel frontal utilize o menu *VIEW* para exibir a paleta de controles. Você também pode exibir a paleta de ferramentas e manter o *Automatic Tool Selection* ligado para facilitar o trabalho com nossa VI.

Na paleta de controles selecione o Controle Numérico e posicione o mesmo no painel de controle. Guiando o mouse até o *label* do controlador, edite seu nome:



Agora adicione um indicador numérico ao painel de controle utilizando a paleta de controles. Nomeie o mesmo como “Deg F” e vamos utilizar um controlador Booleano, o comando STOP. Você pode acessar as propriedades deste comando e remover seu *label*. Insira também um indicador

Booleano (LED) para continuarmos. Perceba que tanto um controle quanto um indicador podem mudar seu estado (Indicador/Controle) quando acessamos suas propriedades.

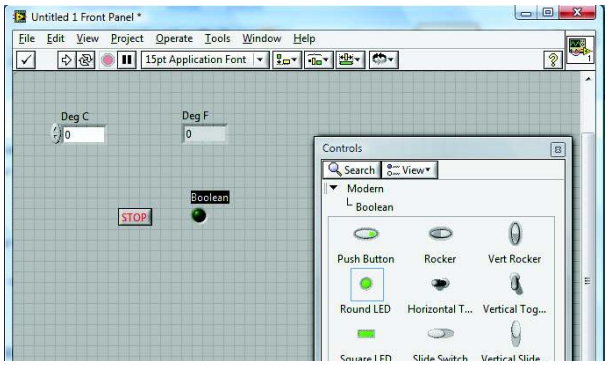


Figura 12 – Inserindo objetos no Painel Frontal do LabVIEW

Agora vamos ao diagrama de blocos por meio do menu *Window*. Verifique os componentes que foram criados no diagrama de blocos. Agora vamos explorar melhor o diagrama de blocos para compreendermos melhor esta relação.

3.2.6 Diagrama de blocos

O diagrama de blocos consiste basicamente de terminais, nodes e ligações (fios). Os terminais são as entradas e saídas, ou seja, a interface entre painel de controle e diagrama de blocos, os nodes podem ser entendidos como operandos, objetos ou sub VIs que efetuam as operações com os dados de entrada e estabelecem uma saída. Por fim, as ligações, ou fios, são os caminhos ou transferências de informação de entrada e saída. Pela figura abaixo podemos examinar alguns detalhes do que abordamos acima:

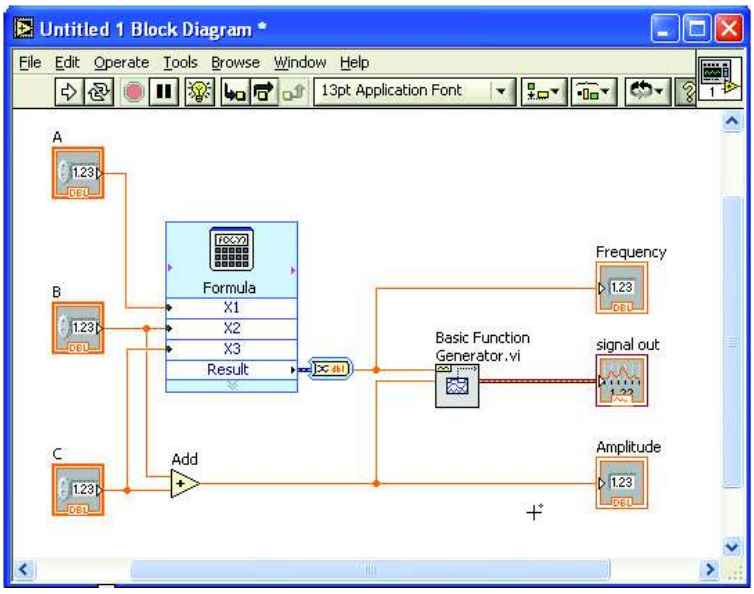


Figura 13 – Objetos no Diagrama de Blocos do LabVIEW

Os elementos A, B e C, quando posicionados no painel frontal, automaticamente são relacionados no diagrama de blocos. Da mesma forma as saídas *Frequency*, *signal out* e *Amplitude* são exibidas no painel frontal. Assim caracterizamos os terminais no diagrama de blocos acima. Observe as bordas externas assim como as setas de conexão destes elementos e verifique as diferenças entre entradas e saídas.

Os nodes são elementos similares a operadores, funções ou sub-rotinas em linguagem de programação convencional. Eles são adicionados por meio da paleta de funções no diagrama de blocos. Na figura, temos os nodes *Add, Formula, Basic function generator e From DDT*.

As VIs e Express VIs podem ser exibidas como ícones ou em seu modo expandido (*expandable* node). Esta aparência pode ser alterada nas propriedades das VIs e então podemos acessar de duas formas diferentes, da forma mais conveniente, suas entradas e saídas:

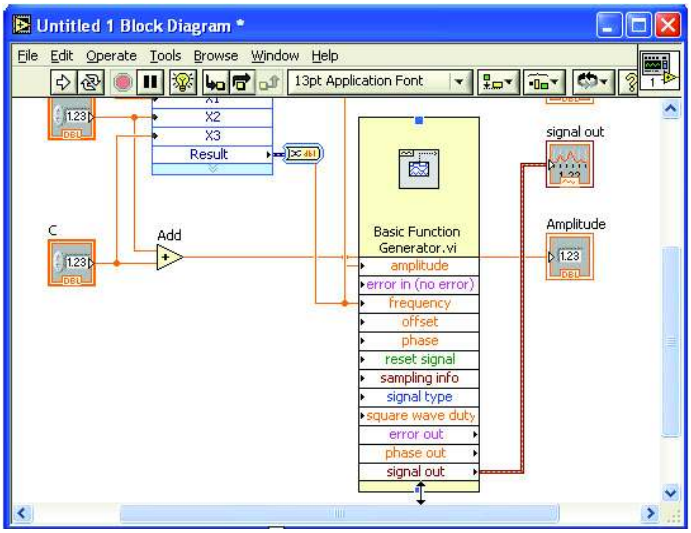


Figura 14 – Expansão de nodes no diagrama de blocos do LabVIEW

Também podemos mudar a visualização de terminais para conveniência de espaço na utilização da VI.

Os fios (*wires*), estabelecem a comunicação de variáveis em nossos programas em LabVIEW e estas variáveis podem ser direcionadas a diversas VIs ou sub VIs simultaneamente, o que confere grande versatilidade aos códigos e facilita em muito sua elaboração. Estas ligações possuem cores e formas que as caracterizam de acordo com seu tipo.

Na variável numérica uma linha laranja indica um Ponto Flutuante e uma linha azul um Inteiro.

Caso uma ligação indevida ocorra, uma linha tracejada interrompida será apresentada:

	Scalar	1-D Array	2-D Array
Numeric			
Boolean			
String			
Dynamic			

No diagrama de blocos, sobre os nodes, é possível verificar as conexões tornando visível as mesmas por meio da janela de propriedades (botão direito do mouse). O LabVIEW pode ainda ligar objetos automaticamente se esta função estiver habilitada nas opções do LabVIEW (menu Tools):

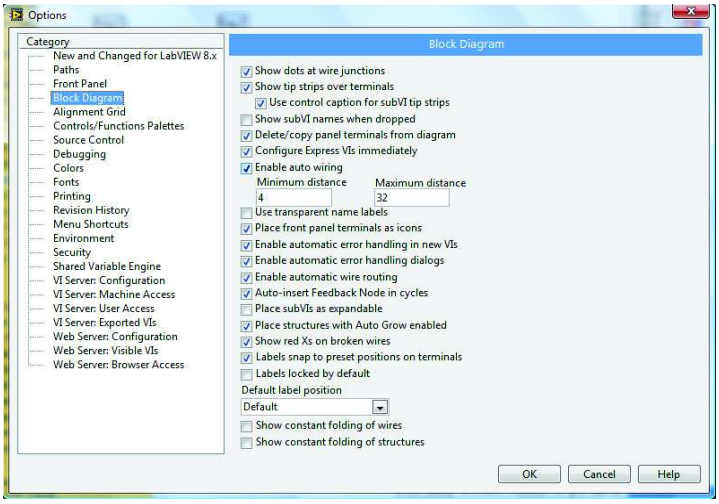


Figura 15 – Menu de opções do Diagrama de Blocos do LabVIEW

Outra propriedade é a de organizar as ligações no LabVIEW clicando-se com o botão direito do mouse sobre os fios e selecionando a função *Clean Up wire*.

Observe sempre com atenção as ligações utilizando os "*tip strips*" que são *labels* de conexão exibidos pelo LabVIEW nas entradas e saídas das VIs para facilitar a orientação das ligações.

Exercício 1: Aquisição de sinais

Agora vamos abrir uma VI proveniente dos modelos do LabVIEW e explorar algumas possibilidades de explorar o LabVIEW.Para isso vamos abrir um exemplo em *NEW>VI from Template>Generate and Display*.

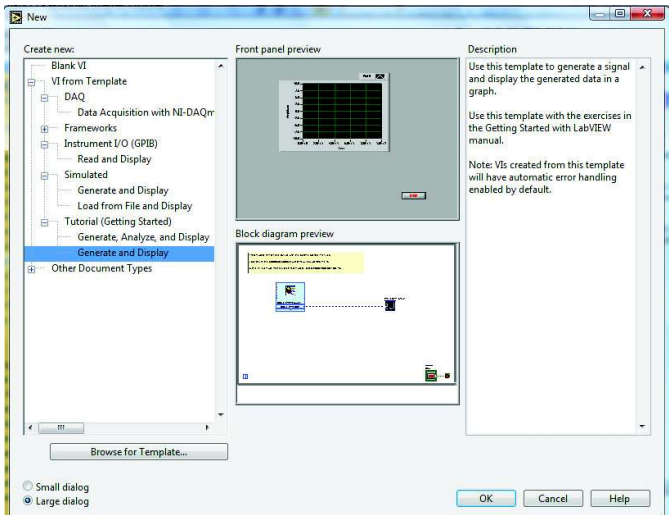


Figura 16 – Abertura de *Template* do LabVIEW

Abrindo este modelo temos a apresentação do painel frontal com um *Waveform Graph* e um botão *STOP*.

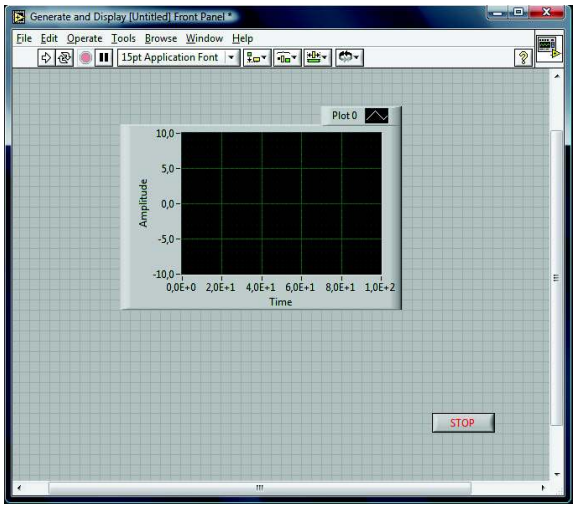


Figura 17 – Painel Frontal de *Template* do LabVIEW

Exibindo o diagrama de blocos por meio do menu *Window*, temos a conexão de uma Express VI *Simulate Signal*. Com o acesso as propriedades desta VI podemos configurar vários de seus atributos. Correndo o mouse externamente à Express VI *Simulate Signal* podemos visualizar suas

entradas e saídas, o que pode ser visualizado também pela expansão vertical gráfica da VI por meio da dupla seta em sua base.

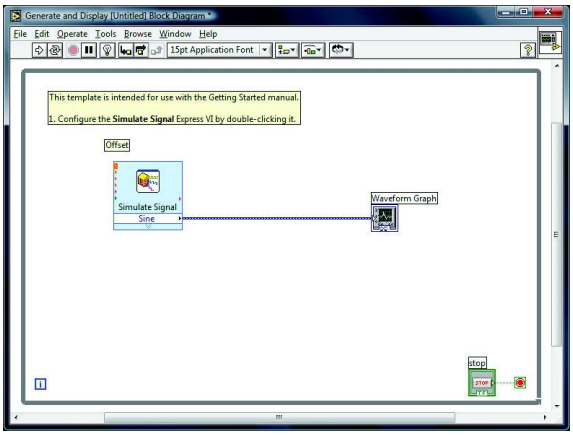


Figura 18 – Diagrama de blocos de *Template* do LabVIEW

Voltando ao painel frontal, podemos prontamente rodar este *Template* e visualizar a forma de onda gerada pela Express VI no *Waveform Graph*. Podemos agora no painel frontal configurar as propriedades do *Waveform Graph* para que o mesmo se adéque a apresentação desejada do projeto.

No diagrama de blocos podemos editar a forma de onda de saída, explorando propriedades como Tipo da forma de onda, amplitude, freqüência, off-set, número de amostras, entre outros. Procure explorar estas informações e conhecer melhor esta Express VI que será amplamente utilizada em suas aplicações.

Podemos agora acrescentar controles de parâmetros de entrada a Express VI como, por exemplo, amplitude. No painel frontal coloque um botão de controle por meio da paleta de controles disponível no menu VIEW.

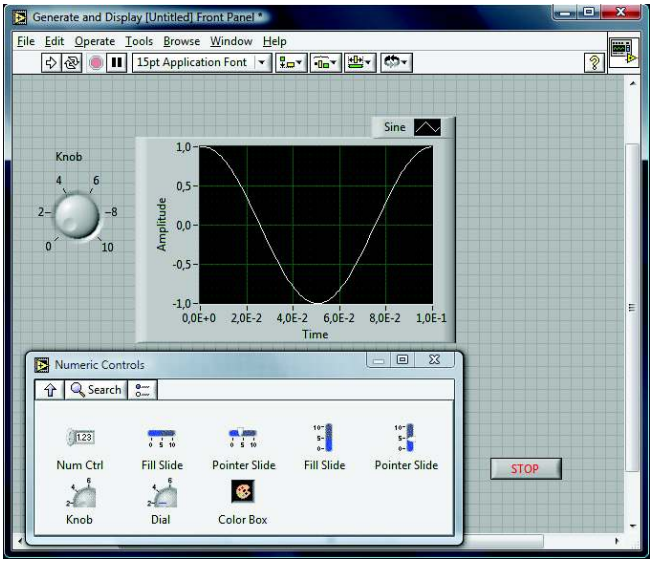


Figura 19 – Manipulação de painel de *Template* do LabVIEW

No diagrama de blocos efetue a ligação do Knob à entrada Amplitude da Express VI. Retorne ao painel de controle e rode a VI para verificar a atuação deste comando. Este simples exemplo começa mostrar um pouco da capacidade de interação para simulação de sinais propiciada pelo LabVIEW.

Agora introduza ao diagrama de blocos, utilizando a paleta de controles, a Express VI *Scaling and Mapping*, configurando a mesma para um coeficiente angular (*Slope*) igual a 10. Introduza uma segunda ligação do simulador de sinais ao indicador gráfico e observe que o LabVIEW automaticamente irá compor os dois sinais para estabelecer a entrada de dados. Rode a VI e verifique o comportamento do indicador gráfico.

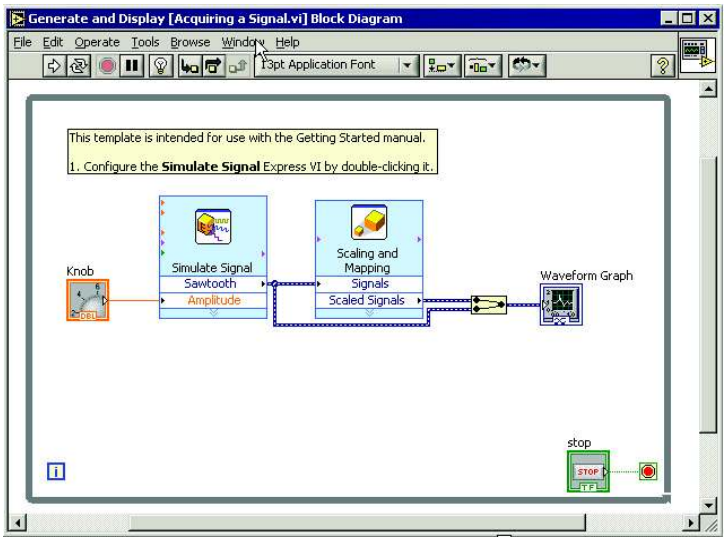


Figura 20 – Manipulação de Diagrama de Blocos de *Template* do LabVIEW

Não se esqueça de explorar todas as propriedades do indicador gráfico e do controle (*Knob*) para ampliar seus conhecimentos no LabVIEW e adequar cada painel frontal às suas necessidades de projeto.

Quanto tempo você levaria para montar todas as ferramentas gráficas ilustradas e colocá-las para funcionar, validar os resultados e então trabalhar realmente em seu projeto? Este questionamento sempre é lembrado quando já tivemos que perder boa parte do tempo de desenvolvimento na programação. A possibilidade de validação dos processos de análise de dados ou mesmo de controle podem ser vislumbrados neste simples exemplo.

3.3 Controle de elaboração e execução de VIs e técnicas de debug

Assim como em qualquer ambiente de programação em qualquer outra linguagem por linha de comandos algumas ferramentas auxiliam bastante o controle e a execução de VIs e consequentemente a identificação de erros (debug):

A execução não é possível: Clicando sobre esta seta podemos identificar, por meio de uma caixa de diálogo, os pontos que implicam em uma impossibilidade de execução da VI. Esta primeira indicação nos direciona pontualmente a todos os erros encontrados na VI.

Outra ferramenta importante no controle da execução de nossas VIs e que fornece informação detalhada do fluxo do programa é o comando *highlight execution*. Com este comando podemos identificar todo o fluxo de dados em um programa e exibição de valores das variáveis nas ligações. Esta poderosa ferramenta auxilia em muito no tratamento de erros e verificação da eficácia de códigos previamente elaborados.

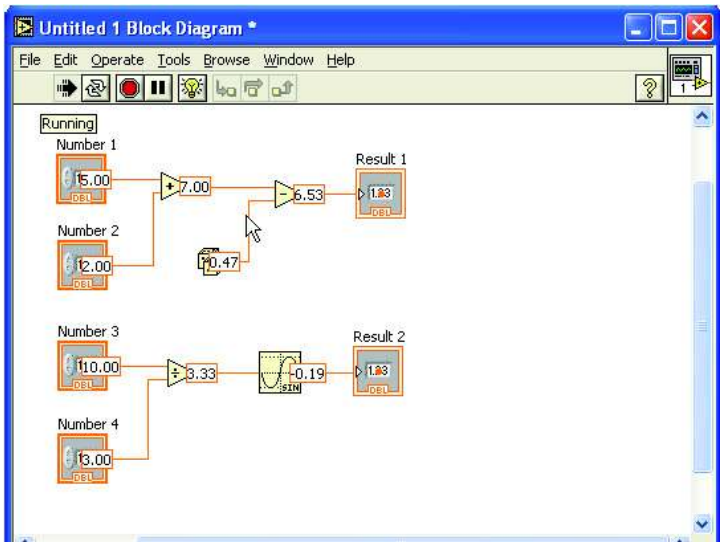


Figura 21 – Ilustração de execução com *Highlight Execution* no LabVIEW

A utilização dos comandos *STEP into, over e out* podem facilitar a compreensão ponto a ponto de nossas VIs e assim podemos identificar problemas previamente ou saltar por funções com erro ou falta de informação para prosseguir com outras etapas funcionais do programa.

Outro identificador importante que auxilia a elaboração de nossas VIs é o *context help* que indica de forma simples e simultânea uma breve descrição dos objetos utilizados na VI, bastando para isso passar o mouse sobre os objetos.

A *probe tool*  também é um recurso importante que auxilia no tratamento de erros e acompanhamento do funcionamento das VIs em elaboração ou já desenvolvidas. Esta ferramenta evita a utilização de muitos indicadores desnecessários quando os mesmos não são fundamentais dentro das funções de nossos projetos.

Por fim, podemos posicionar ainda interrupções (*break point*) em seções do programa, sobre as ligações, para que possamos avaliar pontos específicos de interesse em nossas VIs.

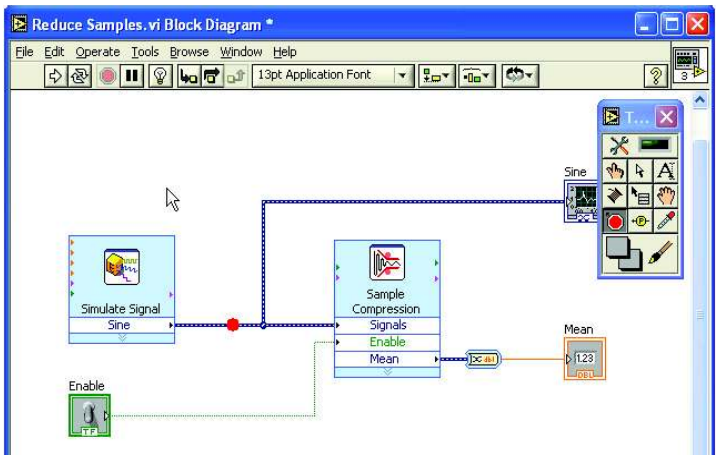


Figura 22 – Utilização do *Break point* na depuração de VI do LabVIEW

Procure utilizar as ferramentas de *debug* como *break point* e *probe* no primeiro exemplo para familiarizar-se e avaliar o potencial das mesmas no processo de depuração de programas.

Uma dica importante: especialmente quando estamos com um processo em andamento, envolvendo hardware para aquisição ou controle mais especificamente, não é aconselhável a utilização de ferramentas como *break point* ou *highlight execution*. Estas ferramentas não operam em tempo real e provocam propositalmente paradas para que o processo possa ser analisado com detalhes. Imagine um ensaio de sobrecarga de um transformador onde uma parada de emergência deve apresentar resposta imediata e a mesma aguarda alguns segundos para ser efetivamente executada! O uso das ferramentas de depuração deve ser efetuado preferencialmente nas fases de projeto e validação.

4 Programação modular

4.1 SubVIs

As SubVIs são análogas às sub-rotinas em linguagens de programação baseadas em texto, ou linhas de comando. As SubVIs, como as sub-rotinas nas outras linguagens de programação, simplificam os diagramas de bloco e ajudam a gerenciar alterações em VIs.

4.1.1 Personalizando ícones das VIs e SubVIs

Todas as VIs são representadas por ícones que são apresentados no canto direito do painel frontal e do diagrama de blocos. O ícone default mostra ainda o número de VIs que você abriu após inicializar o LabVIEW para construção de seu programa.

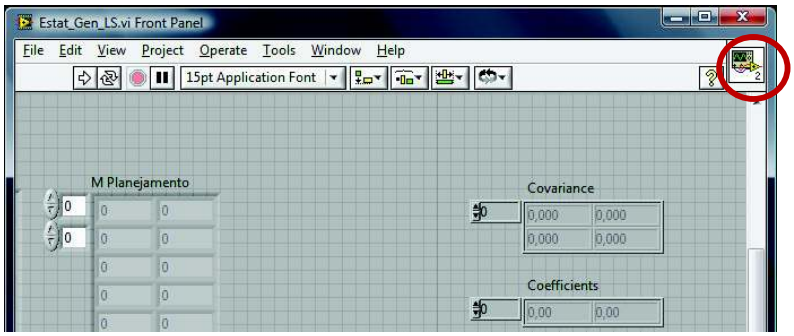


Figura 23 – Indicação do ícone da VI em uso no LabVIEW

Você pode editar seus ícones com o *Icon Editor* , acessível por meio do botão direito do mouse sobre o ícone. As ferramentas gráficas do editor de ícones permite uma limitada, porém suficiente, gama de ferramentas para que você possa personalizar seu ícone. As ferramentas são simples e intuitivas, permitindo fácil manipulação.

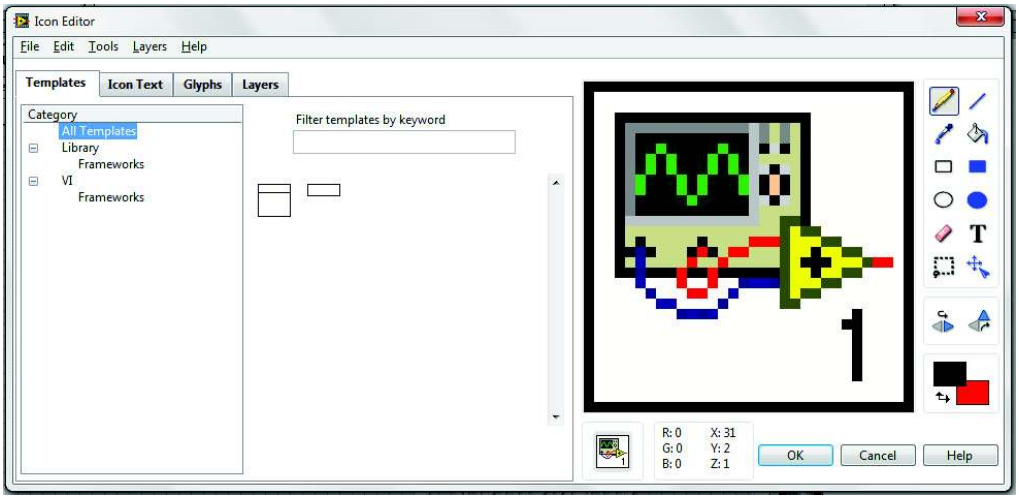


Figura 24 – Editor de ícone da VI em uso no LabVIEW

É sempre importante editar o ícone da VI em desenvolvimento, especialmente se esta for utilizada posteriormente em outros programas como SubVI. Este processo facilita a rápida identificação de sua VI no Diagrama de Blocos de sua aplicação.

4.2 Criando uma SubVI

Para criar uma SubVI deve-se criar um "connector pane", que é um grupo de terminais que definem as entradas e saídas da VI. Você pode atribuir terminais do "*connector pane*" a controles e indicadores no painel frontal.

Utilizando o botão direito do mouse sobre o ícone da VI podemos exibir seus terminais utilizando a opção "*show connector*". O "*connector pane*" irá substituir o ícone da VI, mostrando retângulos que representam os terminais da VI. O número de retângulos que aparecem depende do numero de entradas e saídas, ou seja, do número de controles e indicadores presentes no painel frontal. Como regra, o LabVIEW representa as entradas a direita e saídas a esquerda.

Você pode ainda selecionar padrões para seu "*connector pane*" utilizando a opção "*Patterns*" com o botão direito do mouse sobre o mesmo.

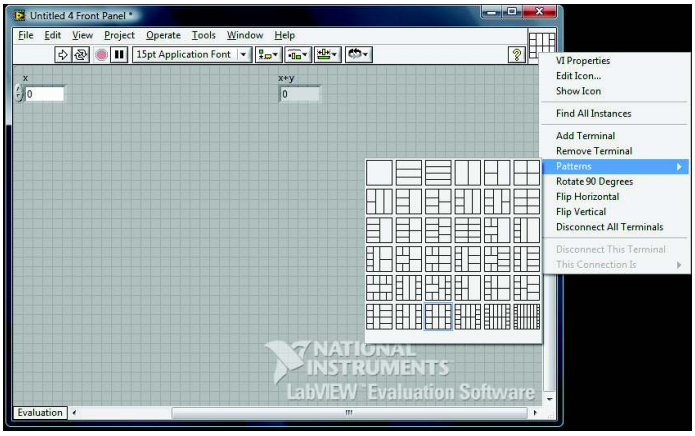


Figura 25 – Edição de modelos de entrada e saída da VI em uso no LabVIEW

Podemos disponibilizar até 28 terminais entre entradas e saídas, mas é recomendável que não trabalhemos com mais de 16 terminais em uma mesma VI, preservando sua função principal de simplificar e organizar nossos códigos.

Quando acessamos os retângulos do *pane* estaremos estabelecendo suas entradas e saídas, para isso o LabVIEW apresenta a ferramenta *Wiring* que não irá estabelecer conexões de fios e sim associações. Cada retângulo selecionado será associado à entrada ou saída posteriormente selecionada. Assim, selecionamos todas entradas e saídas de nosso interesse.

A cada seleção os retângulos alteram suas cores de acordo com o tipo de variável estabelecida como entrada e/ou saída.

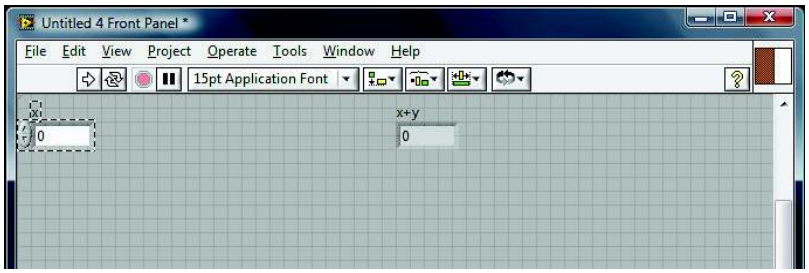


Figura 26 – Associações ao *connector pane* da VI em uso no LabVIEW

Clicando em cada retângulo posteriormente, cada entrada e saída correspondente será identificada. Podemos retornar ao ícone selecionando novamente com o botão direito do mouse nosso "*connector pane*" e escolhendo a opção "*show icon*".

Exercício 2: A famosa SubVI Convert Celsius to Fahrenheit

Este exemplo é muito utilizado para sua primeira VI em LabVIEW, por isso esta SubVI é tão "famosa". Construa uma VI para conversão de temperatura de Celsius para Fahrenheit. Lembrando que esta conversão consiste em multiplicar a temperatura de entrada, em graus Celsius, por 1,8 e somando 32 a este produto. Posteriormente edite o ícone desta VI de forma a ser facilmente identificada. Atribua entrada e saída exibindo seu "*connector pane*" e salve sua VI. Verifique no *Context Help* as entradas e saídas de sua VI.

Com este exercício você já estará explorando operadores numéricos e de fato criando sua primeira VI. Este passo é sua introdução real no LabVIEW e por meio deste pequeno exemplo você já estará se familiarizando com o painel frontal e diagrama de blocos.

4.3 Utilizando uma VI como SubVI

Você pode utilizar uma VI como SubVI em outra VI, para isso, selecione na paleta de funções do diagrama de blocos, *Select VI...,* e indique a VI que se deseja utilizar como SubVI.

Todas as VIs, ou SubVIs podem ser exploradas com um duplo clique, inclusive as VIs disponibilizadas pelo LabVIEW, e assim começamos a entender a estrutura de programação da plataforma. Uma boa organização de um programa requer a utilização de SubVIs. Adiante veremos outras ferramentas para agrupamento de variáveis distintas que também pode ser uma boa forma de organização de subgrupos e consequentemente do diagrama de blocos.

Exercício 3: Utilizando a VI Convert Celsius to Fahrenheit como SubVI

Construa uma VI para gerar sinais aleatórios de temperaturas entre 30 e 50 graus Celsius e indicação de temperatura em Celsius e Fahrenheit com ajuda da SubVI construída no exercício 2.

4.4 Criando SubVIs com seções de VIs

Um recurso de fácil utilização dentro de uma VI, para organizar e auxiliar na estruturação de um programa é selecionar partes da VI e converte-las em SubVIs. Isto pode ser efetuado pela simples seleção de uma parte da VI no diagrama de blocos e no menu “Edit” utilizar a função “*create SubVI*”. Esta SubVI pode ser salva e se não for salva durante a elaboração do código, será solicitada esta ação quando a VI for finalizada.

Aplicam-se os mesmos limites de entradas e saídas a estas SubVIs e sua aplicação deve ser cuidadosa, lembrando-se de se estabelecer as corretas entradas e saídas para sua funcionalidade ou aplicação posterior.

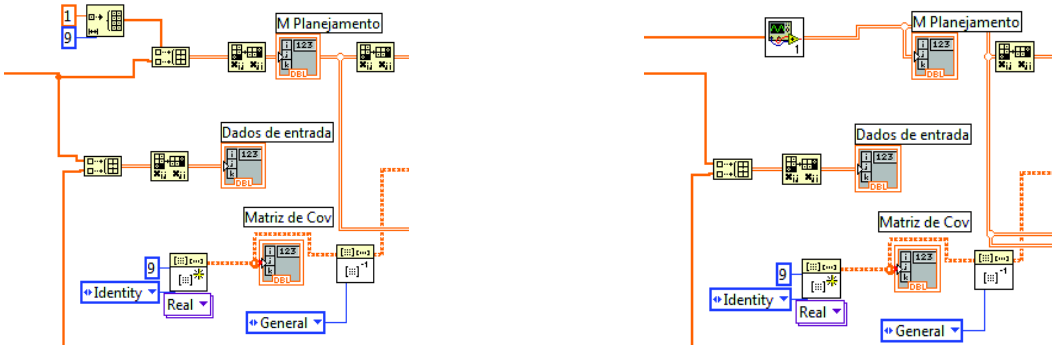


Figura 27 – Conversão de códigos do Diagrama de Blocos em SubVIs por meio da função *create SubVI*

Até aqui o controle de execução fica restrito basicamente as ferramentas do painel de controle e diagrama de blocos, mas o intuito de desenvolver programas para automação de processos laboratoriais é o controle sobre a execução e a distribuição de ferramentas para um grupo de trabalho ou para equipamentos dedicados a ensaios, sem a necessidade de atuação neste tipo de comando, inibindo também erros operacionais durante as rotinas. Esta possibilidade só é alcançada com a introdução de estruturas de repetição e sequenciamento que serão abordadas na próxima seção.

5 Loops e Charts

5.1 While Loops

A estrutura While Loop é similar as estruturas *Do Loop* ou *Repeat_Until Loop* em linguagens de programação baseadas em linhas de comando. Esta estrutura é utilizada no diagrama de blocos no LabVIEW e é amplamente empregada, executando uma operação em subdiagrama até que uma condição seja alcançada.

Na paleta de funções do diagrama de blocos, junto ao grupo *Structures* podemos acessar a estrutura *While Loop*. Você pode criar um subdiagrama e inserir elementos dentro de uma estrutura *While Loop* ou utilizá-la sobre um código ou diagrama pré-estabelecido.

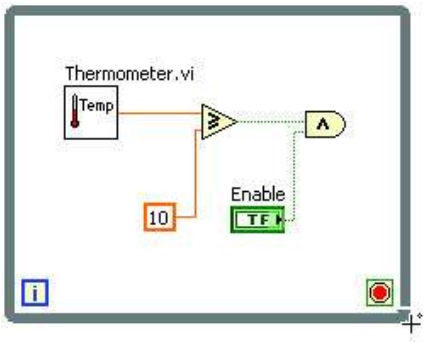


Figura 28 – Ilustração da estrutura *While Loop* no Diagrama de Blocos

Nesta estrutura visualizamos no canto inferior esquerdo o indicador de número de iterações que em seu estado inicial, ou seja, na primeira iteração, o indicador "i" é igual a zero, respeitando a composição vetorial de todas as estruturas e funções do LabVIEW, com elementos componentes com indexação de zero "0" a n-1. No canto inferior direito temos o terminal condicional que aguarda um sinal Booleano "*true*" para finalizar as iterações. Podemos alterar esta condição acessando as propriedades deste comando com o botão direito do mouse e assim invertemos a resposta booleana do controle de execução.

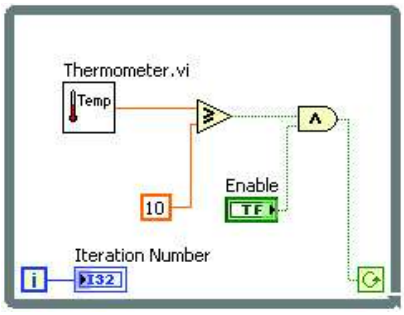


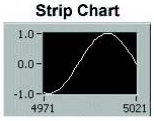
Figura 29 – Ilustração da estrutura *While Loop* no Diagrama de Blocos com finalização invertida

Nesta condição, enquanto verdadeiro ou "*true*", o *While Loop* continuará executando as operações internas, e com um evento falso este será finalizado.

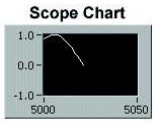
Uma observação importante nas estruturas de loop é que os sinais externos que entram na mesma por meio de tuneis são assumidos como constantes durante a execução, e não podem ser alterados externamente ao loop durante suas execuções. As saídas só serão enviadas quando o loop for interrompido, ou seja, muito cuidado deve ser tomado em condições de controle, utilizando sinais resultantes de estruturas de laço, principalmente aquelas que envolvem paradas críticas e de segurança em um sistema de ensaios, pois a resposta depende da finalização das mesmas. Isto valerá para as demais estruturas exploradas a seguir.

5.2 Waveform Charts

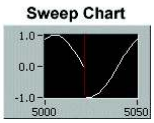
O Waveform Chart é um indicador numérico especial que pode mostrar de forma gráfica uma ou mais entradas. O Waveform Chart é acessível no painel frontal por meio da paleta de controle no grupo Graph. Os charts apresentam três modos de exibição: Strip, Scope e Sweep:



- O modo default é o Strip, onde os dados são apresentados de forma contínua, da esquerda para direita através do chart;



- O Scope apresentam um item de um dado de entrada, como um pulso ou forma de onda, rolando parcialmente da esquerda para direita no chart;



- O Sweep é similar ao Scope, plotando os dados da esquerda para direita, onde são apresentados simultaneamente os resultados antigos e novos, a direita e esquerda respectivamente, por uma linha separadora.

Este modo pode ser alterado no *chart* utilizando-se o botão direito do mouse sobre o mesmo. Acessando suas propriedades podemos alterar sua aparência e outras propriedades, como formato, precisão, escalas, etc. Você pode ainda padronizar a maneira de exibição do *Waveform Chart* no painel frontal, utilizando ferramentas de edição de legendas, exibição de escalas e paletas gráficas. As escalas podem ser editadas diretamente em seus valores máximos e mínimos, facilitando a utilização e padronização de exibição. Procure explorar estes recursos e outras propriedades do *Waveform Chart* para compreender melhor sua abrangência.

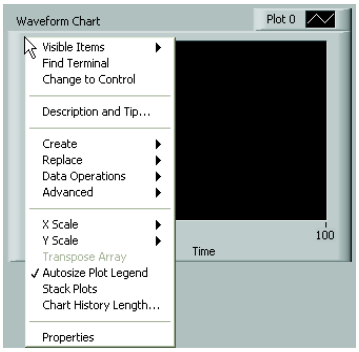


Figura 30 – Ilustração do acesso a propriedades de um *Waveform Chart* no Painel Frontal do LabVIEW

Quando trabalhamos com um chart podemos lembrar de um painel de controle estatístico de processos, onde os dados são plotados um a um. Uma característica muito interessante de utilização de um *waveform chart* é sua utilização com dados escalares, o que nos permite plotagens dentro de uma estrutura de loop, por exemplo, exibindo resultados durante sua execução, o que nos dá uma idéia de "anotações", ou melhor, de acompanhamento dos dados.

A característica de "monitoramento" de um *waveform chart* nos direciona a ensaios com aquisição de dados temporais. Um exemplo interessante que lembra este node é o monitoramento de sinais vitais em ensaios de equipamentos eletromédicos, ou monitoramento de elevação de temperatura, amplamente utilizado em ensaios de segurança de equipamentos e materiais elétricos, onde o comportamento é exibido ao ensaísta a cada evento relevante ou tempo pré-determinado.

5.3 Atuação mecânica dos controles Booleanos

Na maioria dos aplicativos desenvolvidos por equipes de laboratório é necessário utilizarmos controles para manipulação de instruções que representam ações a serem tomadas por nosso código. O LabVIEW traz algumas vantagens na configuração destes botões que podem executar funções, como no painel de controle de um equipamento, possibilitando diferentes configurações que facilitam e normalmente cobrem todas as necessidades dos aplicativos.

O LabVIEW oferece 6 possibilidades de atuação mecânica nos controles Booleanos, facilmente configuráveis:

- Switch when pressed: é a ação default dos comandos do LabVIEW, onde as alterações no estado do botão ocorrem a cada vez que o botão é pressionado;
- Switch when released: as alterações de estado ocorrem quando soltamos o botão, outro estado só ocorre quando outra ação de soltar o botão ocorre;
- Switch until released: as alterações de estado ocorrem enquanto pressionamos o botão, sendo retornadas ao estado original quando o soltamos, de forma similar a um push-button;

- Latch when pressed: as alterações de estado ocorrem até a leitura do sistema da ação de pressionar o botão. Mesmo que você mantenha o botão pressionado o estado não se altera mais e aguardo uma nova atuação;
- Latch when released: as alterações de estado ocorrem até a leitura do sistema da ação de soltar o botão;
- Latch until released: as alterações de estado ocorrem até a leitura do sistema da ação de pressionar o botão, ou se você soltar o botão, o que ocorrer primeiro.

A alteração destes estados pode ser efetuada com o acesso as propriedades do comando com o botão direito do mouse.

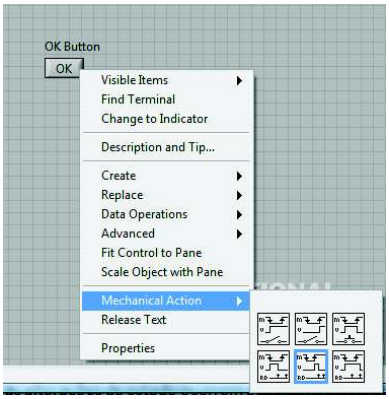


Figura 31 – Ilustração do acesso a propriedades *Mechanical Actions* de um botão no Painel Frontal

Um cuidado na construção do painel frontal é que acionamentos involuntários de botões podem levar a estados padrão de "ligado" ou "desligado" de maneira indesejada. Observe sempre os estados padrão antes da inicialização de sei aplicativo. Se necessário, o valor default pode ser alterado pelo menu *Data Operations.*

Exercício 4: Identificador de números

Construa uma VI para gerar números arbitrários indefinidamente, até os mesmos coincidirem com um número pré-selecionado. Utilize um range de 0 a 1000 com números inteiros para facilitar a busca. Você pode utilizar indicadores booleanos para indicar que um número foi encontrado. Explore a paleta de comparadores para este trabalho.

5.4 For Loops

A estrutura For Loop é semelhante a estruturas *For While* utilizadas em linguagens baseadas em linhas de comando, executando uma ação por um número determinado de vezes. Esta estrutura está localizada na paleta de funções no diagrama de blocos junto ao grupo *Structures*. Da mesma forma que na estrutura *While Loop*, podemos criar um subdiagrama e inserir elementos dentro de uma estrutura *For Loop* ou utilizá-la sobre um código ou diagrama pré-estabelecido.

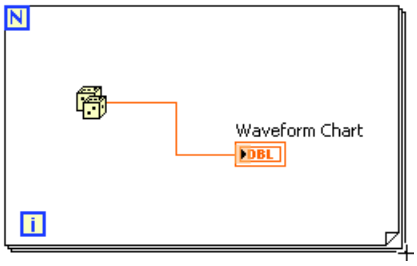


Figura 32 – Ilustração da estrutura *For Loop* no Diagrama de Blocos

Nesta estrutura visualizamos no canto inferior esquerdo o indicador de número de iterações que em seu estado inicial, ou seja, na primeira iteração o indicador "i" é igual a zero. No canto superior esquerdo temos o controle de número de eventos. Utilizando o menu de atalho do controlador N podemos configurar uma constante com o numero de eventos ou um controle para estabelecer o mesmo, por meio do sub-menu "*create*". A figura abaixo mostra uma estrutura *For Loop* que executa 100 vezes uma função.

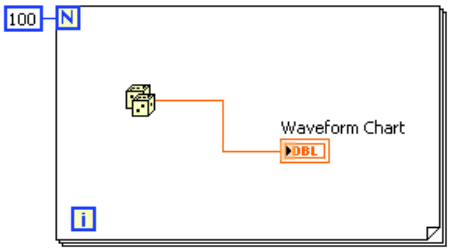


Figura 33 – Ilustração da estrutura *For Loop* para geração de 100 eventos no Diagrama de Blocos

Um detalhe importante do controle de algumas variáveis no LabVIEW é que existe uma redução de resolução automática para determinados controles. Até agora, todos os controles numéricos e indicadores que você utilizou tem precisão de pontos-flutuantes representados com 32 bits. O LabVIEW, entretanto, pode representar números como inteiros (byte, Word ou long) ou pontos-flutuantes (single, double). O padrão da representação numérica é ponto-flutuante.

Se você liga dois terminais que possuem tipos de dados diferentes, o LabVIEW converte a variável um dos terminais para a mesma representação do outro terminal. Lembrando, o LabVIEW coloca um ponto com a cor da variável de destino, chamado ponto de coerção, sobre o terminal onde a conversão é feita.

Por exemplo, considere o terminal de contagem do *For Loop*, onde o terminal representa um inteiro. Se você liga um ponto-flutuante para o terminal de contagem, o LabVIEW converte o número para um inteiro.

Quando a VI converte ponto-flutuante em inteiros, é arredondado mais próximo do inteiro. Se um número está exatamente no meio de dois inteiros, é arredondado para mais próximo do inteiro par. Por exemplo, a VI arredonda 6.5 para 6, mas arredonda 7.5 para 8.

5.5 Shift Registers

Os *Shift Registers* (disponível para *While Loops* e *For Loops*) transferem valores de uma interação do loop para interação seguinte. O *shift register* possui um par de terminais diretamente opostos entre si. O terminal da direita recebe os dados da primeira interação e o registra. Na segunda interação, estes dados são transferidos para o terminal da esquerda enquanto o terminal da direita registra novos dados (a figura abaixo ilustra *shift registers*). Um *shift register* pode trabalhar com qualquer tipo de dados (Booleano, string, array, ect).

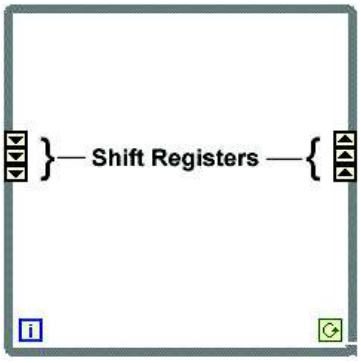


Figura 34 – Ilustração de *Shift registers* em uma estrutura *While Loop* no Diagrama de Blocos

Cria-se um *shift register* clicando, na borda do loop, com o botão direito do mouse e selecionando *Add Shift Register*.

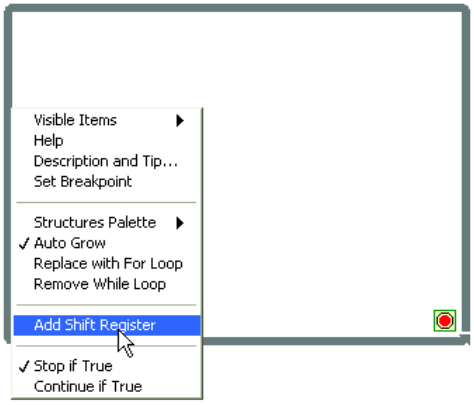


Figura 35 – Inserindo *Shift registers* em uma estrutura *While Loop* no Diagrama de Blocos

Pode-se configurar o *shift register* para "registrar" valores de varias interações. Cria-se um terminal adicional para que se possa acessar os dados das interações anteriores. Esse terminal adicional cria-se, em qualquer um dos terminais, clicando com o botão direito do mouse sobre o *shift register* e selecionando *add element*. Por exemplo, se um *shift register* possui três elementos no terminal da esquerda, pode-se acessar os valores das ultimas três interações.

Podemos ainda inicializar um *shift register* criando constantes ou controles do lado esquerdo do *shift register*, como na figura abaixo. Se nenhum valor inicial for estabelecido para um *shift register* numérico, seu estado inicial é zero.

Figura 36 –*Shift register* com mais de um elemento de registro e inicialização diferente de zero

Exercício 5: Acessando valores prévios com o *shift register*

Construa uma VI para gerar interações com valores prévios provenientes de *shift registers* como ilustrado abaixo. Procure visualizar a operação desta VI utilizando o *highlight execution* para explorar melhor o funcionamento dos *shift registers*.

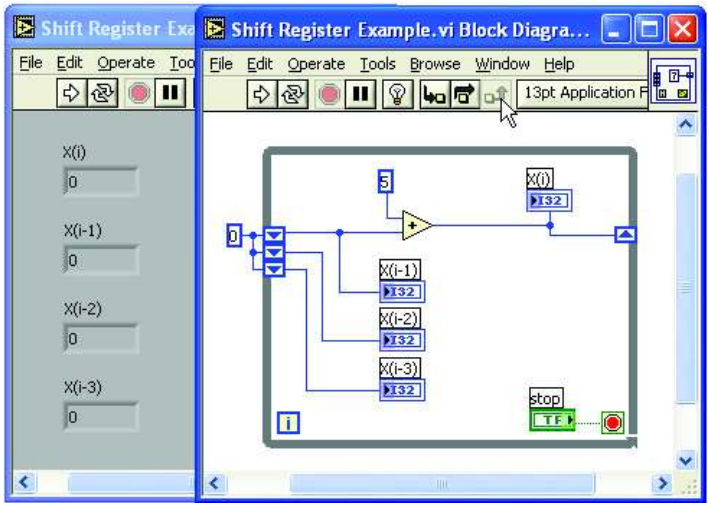


Figura 37 – Acesso aos dados do *Shift register* em uma estrutura *While Loop* no Diagrama de Blocos

Exercício 6: Exibindo dados atuais e registros simultaneamente

Construa uma VI para gerar um gráfico com duas plotagens simultâneas, uma exibindo valores aleatórios e outra exibindo valores médios correntes dos quatro últimos pontos da plotagem aleatória. Para esta VI você deverá utilizar um For Loop com 200 iterações, um *shift register* com três elementos à esquerda, uma VI *Compound Arithmetic*, uma VI *Division*, uma VI *Random number* (0-1), uma VI *Bundle* para agrupar os dados randômicos com a média de dados anteriores a plotagem e uma VI *Wait Until Next* (ms).

Após este exercício você já perceberá sua capacidade em montar pequenos programas e visualizar senários de automação laboratorial. Este exercício com uma pequena modificação da entrada de sinais aleatórios para sinais periódicos com inserção de ruídos por meio da VI *Simulate Signal* traduz-se num filtro de média móvel, e expandindo-se os elementos do *shift register* podemos intensificar os efeitos deste filtro. Outra aplicação é a comparação periódica de dados remanescentes para avaliação da estabilidade de sistemas térmicos.

6 Arrays, Cluster e Gráficos

6.1 Arrays

Um array (vetor) consiste em uma coleção de elementos de dados onde todos possuem o mesmo tipo. Um array possui uma ou mais dimensões e elementos por dimensão até onde a memória permita. O acesso a cada array se faz por um índice (index). O índice é dado em uma faixa de 0 a n-1, onde n é o número de elementos do array. A seguir veja um array de uma dimensão de valores numéricos com 10 elementos. Observe que o primeiro elemento do índice é 0, o segundo elemento é 1, e assim por diante:

Índice	0	1	2	3	4	5	6	7	8	9
Array	1,5	3,0	2,5	2,0	8,0	8,5	7,0	6,5	6,0	5,0

Um array pode possuir várias dimensões como já dito e o limite para estas dimensões é de $2x10^{+31}$–1 elementos. Os elementos de um Array são todos indexados de forma a permitir o acesso individual a qualquer momento.

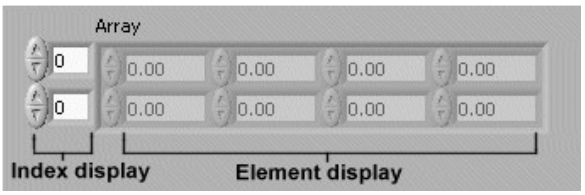


Figura 38 – Representação de um array no LabVIEW

Você pode adicionar um Array por meio da paleta de controle no painel frontal, no sub-menu *Array & Cluster*. Dentro de um Array podemos inserir elementos booleanos, numéricos, strings, paths, refnums, etc. Este reconhecimento do tipo de elemento em um Array é automático, a medida que incluímos uma variável dentro do mesmo.

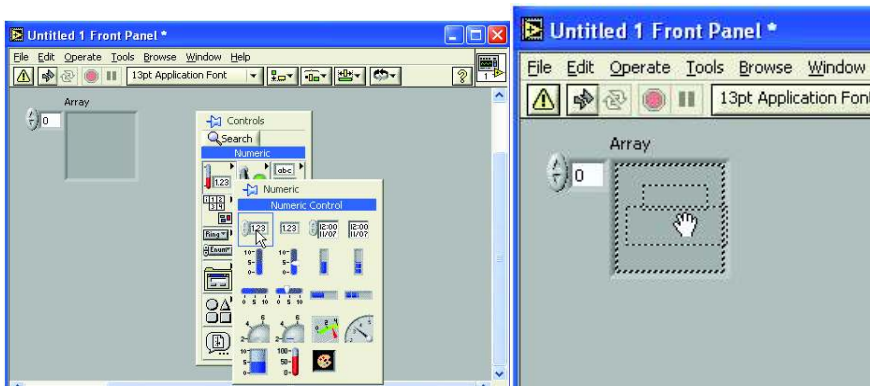


Figura 39 – Inserção de um array no Painel Frontal do LabVIEW

Para adicionar uma dimensão ao Array é só incrementar o indexador de elementos a esquerda do Array utilizando o botão direito do mouse e clicando em "*Add Dimension*".

A ferramenta de posicionamento pode ser utilizada para dimensionar o Array e assim incrementar ou decrementar elementos do mesmo.

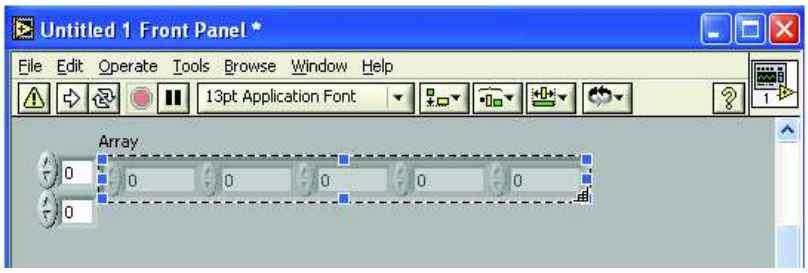


Figura 40 – Alterações de visualização de um array no Painel Frontal do LabVIEW

Para criar um Array de constantes, podemos acessar também o diagrama de blocos e utilizando a paleta de funções, no sub-menu *Array>>Array Constants* inserir este elemento. A caracterização do tipo dos elementos do Array e suas dimensões ocorre da mesma forma que na inserção no painel frontal.

6.1.1 Auto indexação de um Array

Quando conectamos um Array a uma saída de uma estrutura *For Loop* ou *While Loop*, podemos criar um Array auto indexado. O LabVIEW disponibiliza este recurso como padrão quando conectamos qualquer Array a estrutura For Loop, mas você pode alterar esta configuração clicando com o botão direito do mouse sobre o "*tunnel*" de conexão entre a estrutura e o Array.

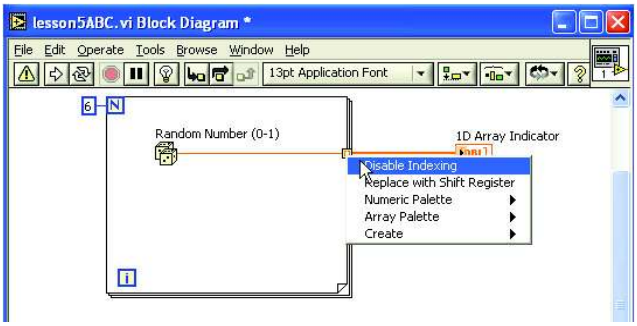


Figura 41 – Configurando a indexação de um array na estrutura *For Loop.*

Na estrutura While Loop, o estado padrão é a Auto indexação desabilitada, já que uma aplicação pode gerar arrays muito grandes e provocar estouro de memória. Mas, da mesma forma que podemos desabilitar a auto indexação na estrutura For Loop, podemos habilitá-la na estrutura While Loop e posteriormente revertê-la.

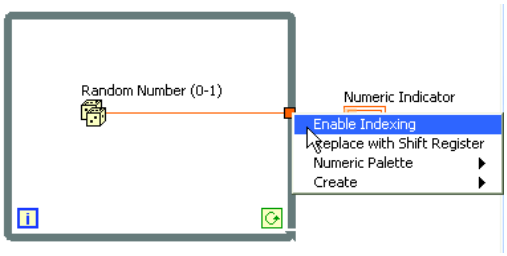


Figura 42 – Configurando a indexação de um array na estrutura *While Loop.*

É importante ter em mente que a auto indexação desabilitada na estrutura *While Loop* é proposital uma vez que podemos ter um estouro de memória se o array gerado não for adequadamente controlado e limitado.

Uma técnica bastante interessante é inserir uma estrutura For Loop em outra e produzir Array bidimensionais, auto indexados.

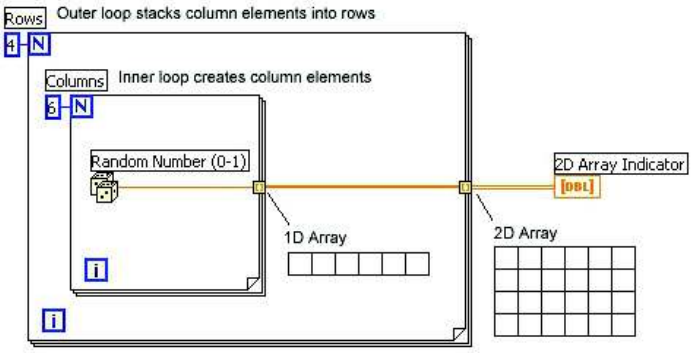


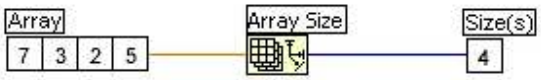
Figura 43 – Utilizando a indexação 2D de um array na estrutura *For Loop.*

Se você insere um Array de n elementos a entrada de uma estrutura *For Loop*, esta estrutura passa a executar seus eventos pelo número de vezes equivalente ao número de elementos, não sendo necessário estabelecer para a mesma o número N de eventos de entrada. Se, no entanto, este estiver sendo utilizado e um Array for aplicado a entrada, o indicador que possuir menor número será respeitado. Assim, para uma estrutura For Loop com N = 20 e com Array de entrada de 5 elementos, segue-se a execução de 5 eventos.

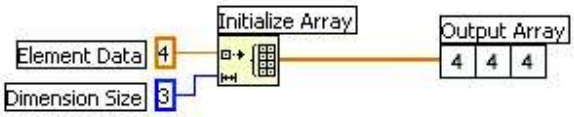
6.1.2 Funções para operações com Arrays

Na sub-paleta de funções Array é possível criar e manipular Arrays. Como estes recursos são utilizados com muita frequência em vários programas, é muito importante conhecermos os recursos básicos de algumas das VIs de criação e manipulação de Arrays.

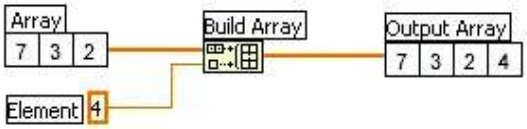
Array size: Esta VI retorna o número de elementos em cada dimensão de um Array:



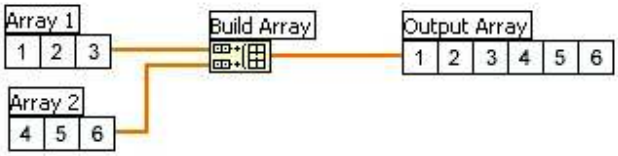
Initialyze Array: Cria um Array com dimensão n e valores específicos pré-estabelecidos:



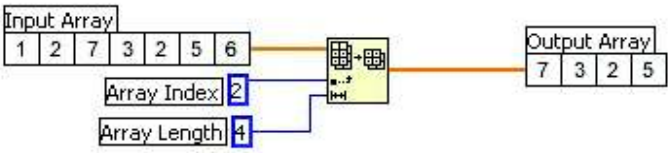
Build Array: Esta VI cria um Array ou adiciona elementos em um Array de dimensão n:



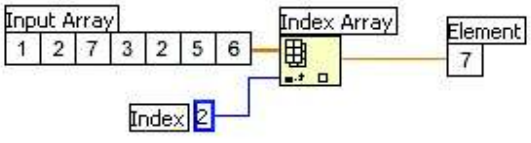
É possível concatenar Arrays com dimensão n na VI *Build Array*, clicando com o botão direito do mouse sobre esta VI:



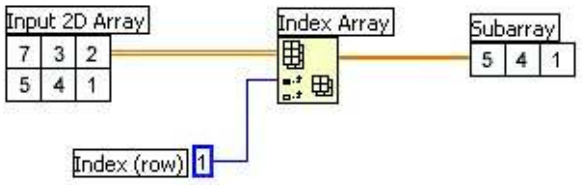
Array Subset: Esta VI retorna uma parte de um Array com início em Index e final estabelecido por *Lenght*. É importante lembrar que a indexação é iniciada em 0:



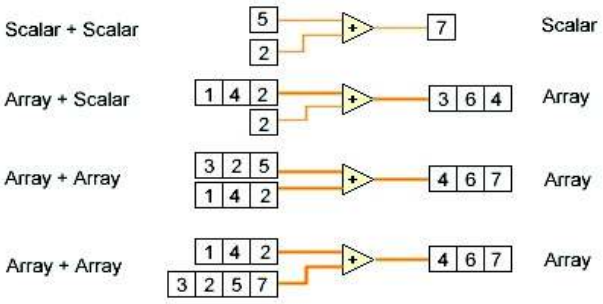
Index Array: Esta VI retorna um elemento, ou extrai um elemento do Array:



Ainda é possível extrair colunas ou linhas de um Array 2D. Quando conectamos este tipo de Array a esta VI o primeiro controle de indexação se refere a linhas e o segundo a colunas, lembrando que a indexação é iniciada em 0:



O Polimorfismo permite somar elementos em diferentes formatos, dentro ou não de Arrays:



Esta interação deve ser observada com detalhes na programação e a validação de dados onde arrays e escalares são processados mutuamente é muito importante.

Exercício 7: Criando e manipulando Arrays

Construa uma VI para gerar um Array de 50 números randômicos, posteriormente criar uma escala para o resultado e extrair um subset do resultado começando do índice n/5 e terminando no índice n/2. Posteriormente varie o número do vetor por meio de um controle mantendo o critério de isolamento de dados.

6.2 Waveform graphs e XY graphs

Os gráficos utilizados no LabVIEW, da mesma forma que em qualquer outra plataforma são utilizados para apresentação de dados e formas de onda. Os dados normalmente são coletados em um Array e então reproduzidos de forma gráfica, o que permite uma primeira análise ou acompanhamento dos resultados.

No LabVIEW temos os gráficos Waveform e XY que estão presentes nas VIs da sub-paleta de controle *Graph*. Os *Waveform Graphs* exibem apenas funções com valores únicos, representados por dados ao longo do eixo X, como qualquer função y=f(X). Os XY Graphs podem exibir, ou

"plotar" qualquer conjunto de valores, como gráficos circulares, conforme ilustrado na Figura 44, ou formas de onda com bases de tempo variáveis.

6.2.1 Waveform Graphs

Um *Waveform Graph* é como um *Waveform chart*, mas utilizado para mostrar múltiplos pontos de uma forma de onda ou vetor de dados no tempo para uma ou mais plotagens. Os *Graphs* são mais utilizados quando queremos exibir os dados depois que estes foram processados ou gerados.

Um *Waveform Chart* mostra os dados ponto a ponto, armazenando dados correntes num *buffer* para exibição posterior. É mais utilizado quando queremos mostrar dados correntes, ou durante a aquisição.

Desta forma o *Waveform Graph* acaba sendo mais eficiente pois sua taxa de atualização é por conjunto de amostras e não por amostras, mas cabe ao programador definir sua utilização previamente, levando em conta os detalhes abordados.

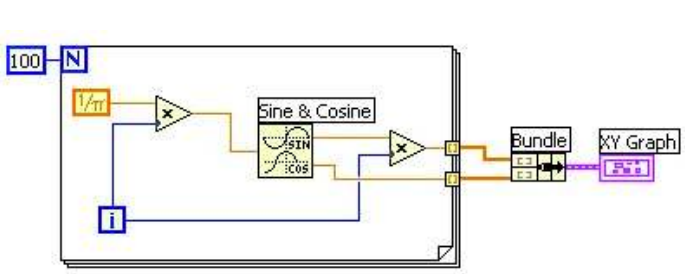
Single-Plot Waveform Graphs: Os *Waveform graphs* com exibição de um único vetor de dados os interpreta como pontos no gráfico, incrementando o eixo X a partir de 0. Estes gráficos também podem aceitar clusters de dados contendo valores iniciais de X, a variação de X (ΔX) e o vetor Y. Os clusters serão abordados posteriormente com mais detalhes.

Multi-Plot Waveform Graphs: Os *Waveform graphs* com exibição de mais de um vetor de dados os interpreta como pontos em gráficos independentes, incrementando o eixo X a partir de 0 para os dois vetores de entrada. Estes gráficos também podem aceitar clusters de dados contendo valores iniciais de X, a variação de X (ΔX) e o vetor Y para cada uma das entradas.

Clicando com o botão direito do mouse sobre os gráficos, podemos transpor os vetores de entrada, exibindo cada coluna de dados na base X, como múltiplos vetores de entrada.

6.2.2 XY Graphs

Da mesma forma que um *Waveform Graph* os gráficos XY podem exibir vetores únicos ou múltiplos e apresentam recursos similares de exibição quanto à formatação, no entanto, as coordenadas de X e Y podem ser aplicadas de forma independente e variável ponto a ponto (não uniformes), conferindo outras propriedades a estes.



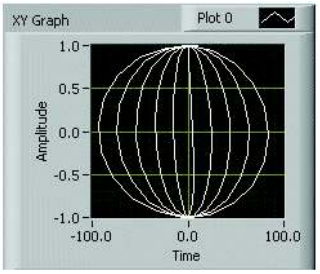


Figura 44 – Exemplo de utilização de gráficos XY.

Exercício 8: Gráfico circular

Construa uma VI para gerar e exibir um Waveform circular. Utilize este exercício para explorar as propriedades dos indicadores gráficos. Posteriormente conecte dois sinais senoidais aos eixos X e Y, defasando os sinais entre si para observar as curvas de Lissajous.

6.3 Intensity plots (gráficos de intensidade 2D)

Os gráficos de intensidade são muito úteis quando queremos expressar comportamentos em duas dimensões ou padrões de formas como terrenos, superfícies, comportamento térmico, etc. No LabVIEW os gráficos de intensidade aceitam blocos de dados e podem trabalhar com no máximo 256 cores.

Os *Intensity plots* estão disponíveis na paleta de controle *Graphs*, na sub-paleta *Graphs*. Estes gráficos utilizam a maioria dos recursos dos *Waveform*, possuindo, entretanto, recursos adicionais não disponíveis em *waveforms* e *charts*, devido a entrada de um terceiro parâmetro de exibição, as cores. Esta terceira entrada é utilizada justamente para configurar o esquema de cores que se deseja estabelecer no indicador, indicado por um terceiro eixo, z, exibido Juno ao gráfico:

Figura 45 – Exemplo de *Intensity Plot* no painel frontal do LabVIEW.

Para alterar a cor do gráfico, ou esquema de cores para mapeamento, utilize o botão direito do mouse sobre o marcador à direita do eixo z, ou seja, sobre o indicador numérico no eixo z que representa a escala de cor do gráfico, selecionando no menu a opção "*Marker Color*".

Você também pode adicionar novos valores de cores ao eixo, clicando sobre este com o botão direito do mouse e selecionando "*Add Marker*", podendo ainda deslocar o marcador dentro do eixo z com o mouse ou editando seu valor, onde o mesmo será automaticamente deslocado.

Estes tipos de gráficos aceitam como entrada um Array 2D, onde os índices de linhas e colunas irão compor os eixos x e y do gráfico, e os valores dos componentes da matriz irão determinar as intensidades ou amplitudes em z.

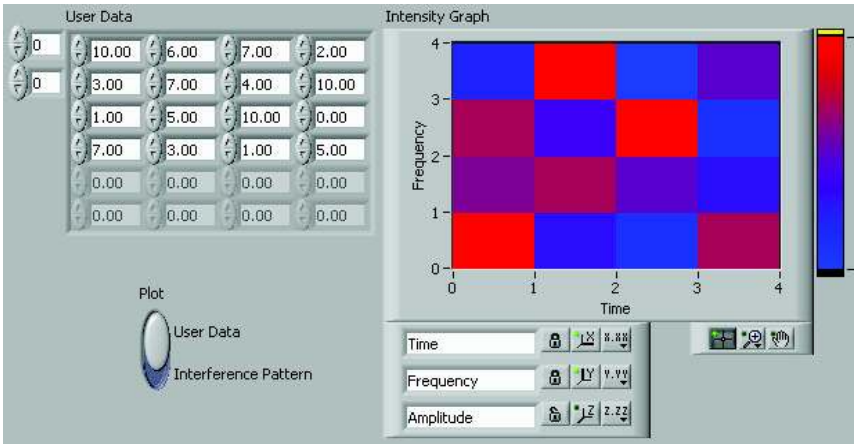


Figura 46 – Exemplo de aplicação de um *Intensity Plot* no painel frontal do LabVIEW.

Devido a sua relativa complexidade, é interessante explorarmos os exemplos da seção "*intensity*" do LabVIEW, onde, no diagrama de blocos, poderemos entender melhor seu funcionamento. A aplicação de "*porperty nodes*" ainda não abordada neste curso, será utilizada nestes exemplos, pois constitui a maneira mais fácil de viabilizar sua aplicação.

6.4 Gráficos 3D

Mais um recurso que merece grande atenção são os gráficos 3D. Normalmente quando trabalhamos com distribuições de informações do mundo real, a exibição em três dimensões é fundamental. Exemplos como exibição de dados em uma superfície, ilustrando comportamentos de temperatura, vibração, rugosidade, entre outros, podem ser alcançados com este recurso

Na paleta de controle do painel frontal, os indicadores gráficos 3D estão junto ao grupo *"Graph"*. Expandindo este grupo temos os seguintes tipos de gráficos 3D:

Scatter – Exibe dados com coordenadas de três eixos e relações entre dois grupos de dados.

Stem — Ilustra uma resposta de impulse organizando os dados por sua distribuição.

Comet — Gera uma plotagem animada com um círculo que segue os dados plotados.

Surface — Gera uma plotagem dos dados conectados em uma superfície.

Contour — Gera uma plotagem de linhas de contorno.

Mesh — Gera uma plotagem de uma malha de superfície com espaços aberto, como uma tela.

Waterfall — Gera uma plotagem da superfície dos dados e no eixo y a área abaixo dos dados plotados.

Quiver — Gera uma plotagem de vetores normais.

Ribbon — Gera uma plotagem de linhas paralelas.

Bar — Gera uma plotagem de barras verticais.

Pie — Gera uma plotagem tipo “pizza” ou “torta”.

3D Surface Graph — Gera uma plotagem de superfície num espaço 3D.

3D Parametric Graph — Gera uma plotagem paramétrica de superfície num espaço 3D.

3D Line Graph — Gera uma plotagem de linhas num espaço 3D.

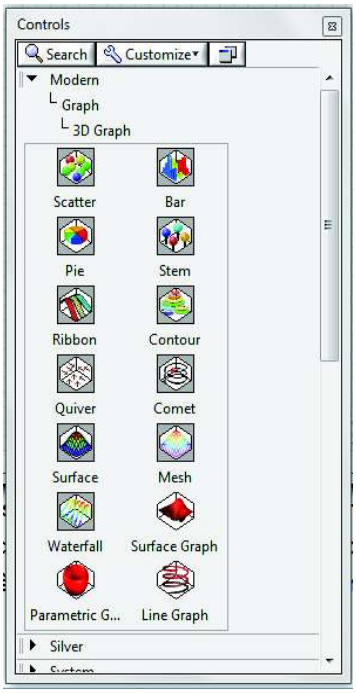


Figura 47 – Exibição do grupo *3D Graph* no painel frontal do LabVIEW.

Um exemplo de gráfico de superfície é ilustrado na figura abaixo.

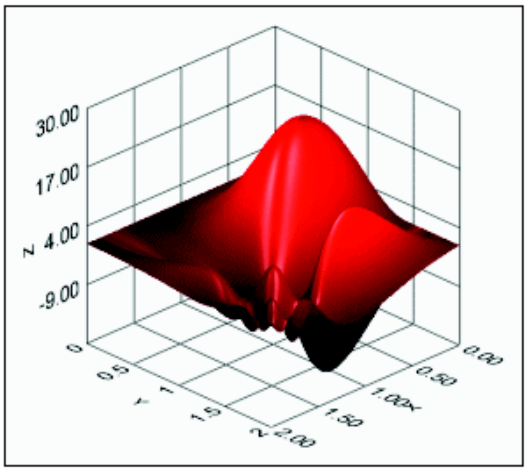


Figura 48 – Exemplo de gráfico de superfície 3D do LabVIEW.

Você pode explorar os exemplos de Vis para gráficos 3D no LabVIEW e utiliza-los como modelos em suas aplicações. Este grupo de VIs e exemplos não são os mais simples para serem utilizados na prática mas sua exploração traz resultados muito bons para aplicativos que requerem riqueza na visualização 3D.

6.5 Figuras 3D

Outro recurso que pode trazer grande riqueza a apresentação dos resultados ou mesmo possibilitar controles e ilustrações de alto nível é a manipulação de figuras 3D. No LabVIEW, existem algumas possibilidades de criação e visualização de sistemas em 3D e vamos explorar algumas delas aqui.

No painel frontal junto a paleta de controle no grupo "*Modern >> Graph*" temos o indicador "*3D Picture*" conforme indicado na figura abaixo. Este indicador pode exibir objetos e figuras em três dimensões de maneira muito eficiente e simplificada.

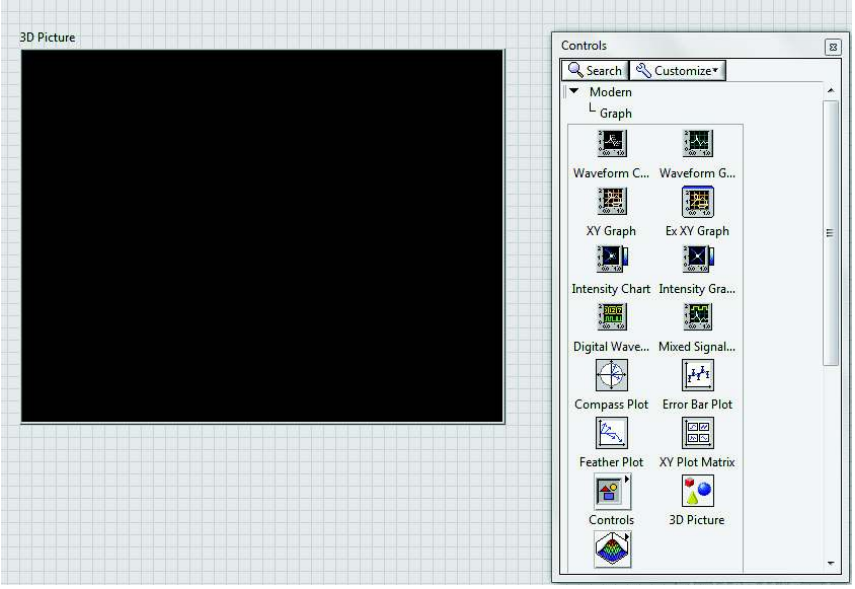


Figura 49 – Indicador "*3D Picture*" no painel frontal do LabVIEW.

No diagrama de blocos podemos visualizar este indicador e podemos utilizar os recursos adicionais para manipulação dos dados que serão exibidos conforme desejado.

Um exemplo bastante simples é a criação de uma figura 3D utilizando o recurso de controle de figuras 3D do diagrama de blocos junto a paleta de funções no grupo "*Programming >> Graphics & Sound >> 3D Picture Control*". Para isso podemos selecionar no subgrupo "*Geometries*" a VI *create cylinder*. Vamos explorar esta VI, conforme indicado na figura abaixo e estabelecer seus parâmetros de entrada: altura 2 e raio 0,5.

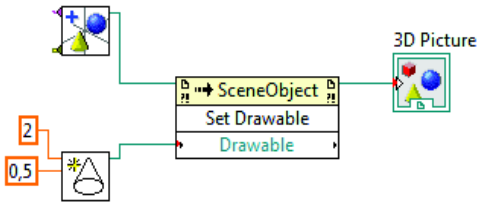


Figura 50 – Exemplo de criação de cilindro no diagrama de blocos do LabVIEW.

No painel frontal, podemos configurar o indicador de figura 3D para permitir o controle de câmara esférico, acessando as propriedades do mesmo. Utilizando a ferramenta *Run Continuously* podemos visualizar a figura do cone em suas diferentes projeções.

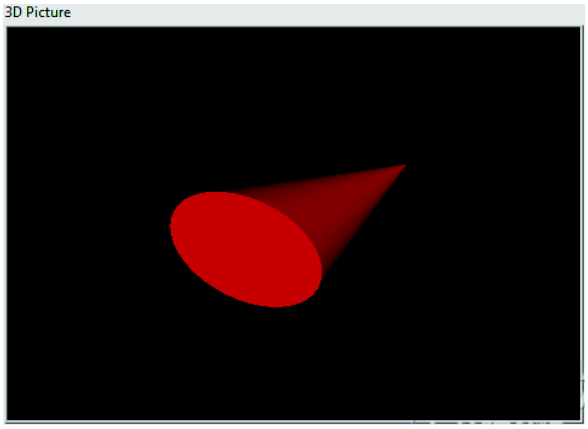


Figura 51 – Exemplo de criação de cilindro no painel frontal do LabVIEW.

Verifique os resultados neste exemplo. Mais uma vez é uma ferramenta simples que nos permite vislumbrar aplicações em laboratório.

Exercício 9: Simulação 3D

Imagine que necessitamos avaliar o movimento relativo de quatro placas de elementos de suporte de um motor num ensaio de vibração. Neste cenário, teríamos estas placas com acelerômetros em suas bases, conectadas a um sistema de aquisição de dados que nos permitiria avaliar, com coordenadas de três eixos, os efeitos de vibração do motor em sua base. Não trabalharemos ainda com o sistema de aquisição, mas vamos montar este exemplo com a finalidade de avaliar ferramentas de simulação que podem ser utilizadas como validação.

Podemos iniciar criando as quatro placas, aqui com as seguintes dimensões fictícias nas coordenadas x, y e z: 0,5, 0,3 e 1,0. Vamos atribuir cores diferentes a estas placas por meio de constantes no diagrama de blocos. Uma maneira simples de se fazer esta atribuição de constante é utilizando o botão direito do mouse sobre a entrada *color* e selecionando a opção *create constant*.

Vamos criar os objetos no diagrama dentro do diagrama de blocos já em uma estrutura While Loop, o que nos permite rodar o programa indefinidamente nos testes. Observe os elementos no diagrama de blocos ilustrado na figura abaixo.

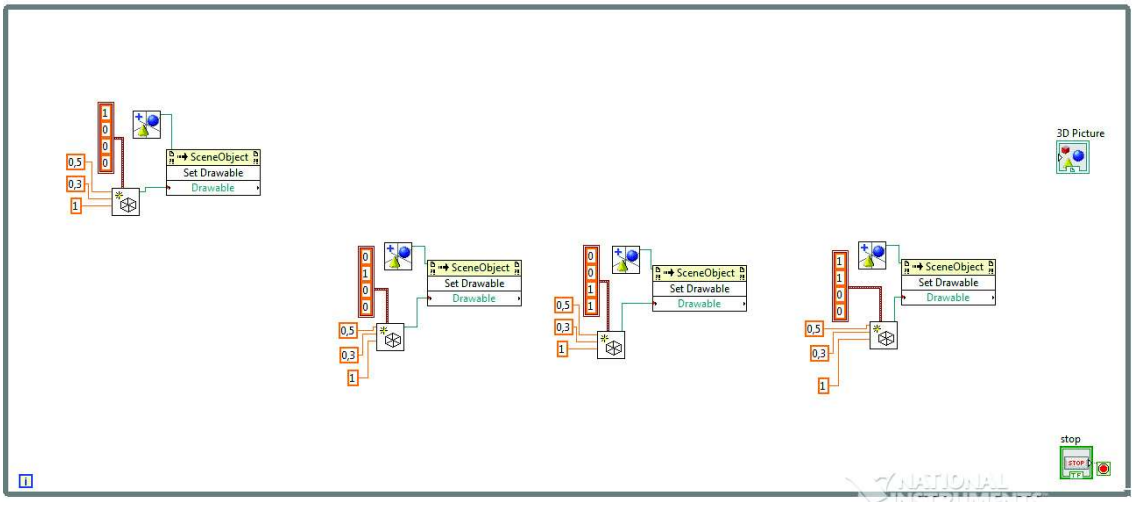


Figura 52 – Criação das 4 placas do exemplo no diagrama de blocos.

Agora vamos distanciar as placas utilizando a VI *Translation*. Com esta VI podemos atribuir as coordenadas de posicionamento relativo dentro de nosso plano 3D, e desta forma teremos a disposição das placas conforme ilustrado na figura abaixo, onde temos o resumo do diagrama de blocos e o resultado.

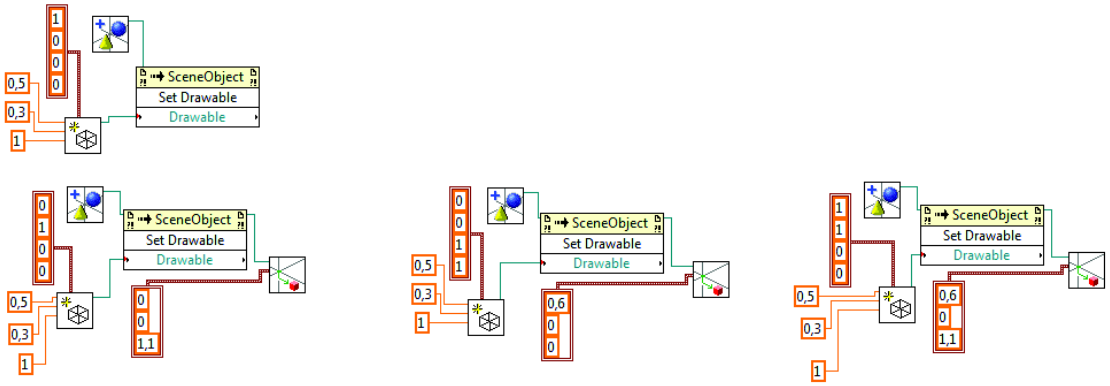


Figura 53 – Translação das 4 placas do exemplo no diagrama de blocos.

Para unir estes cenários de forma organizada e promovendo movimentos independentes, conforme proposto na simulação, já que os sinais virão de acelerômetros virtuais, temos que encadear os objetos utilizando a ferramenta *Invoke Node* da maneira mais adequada possível.

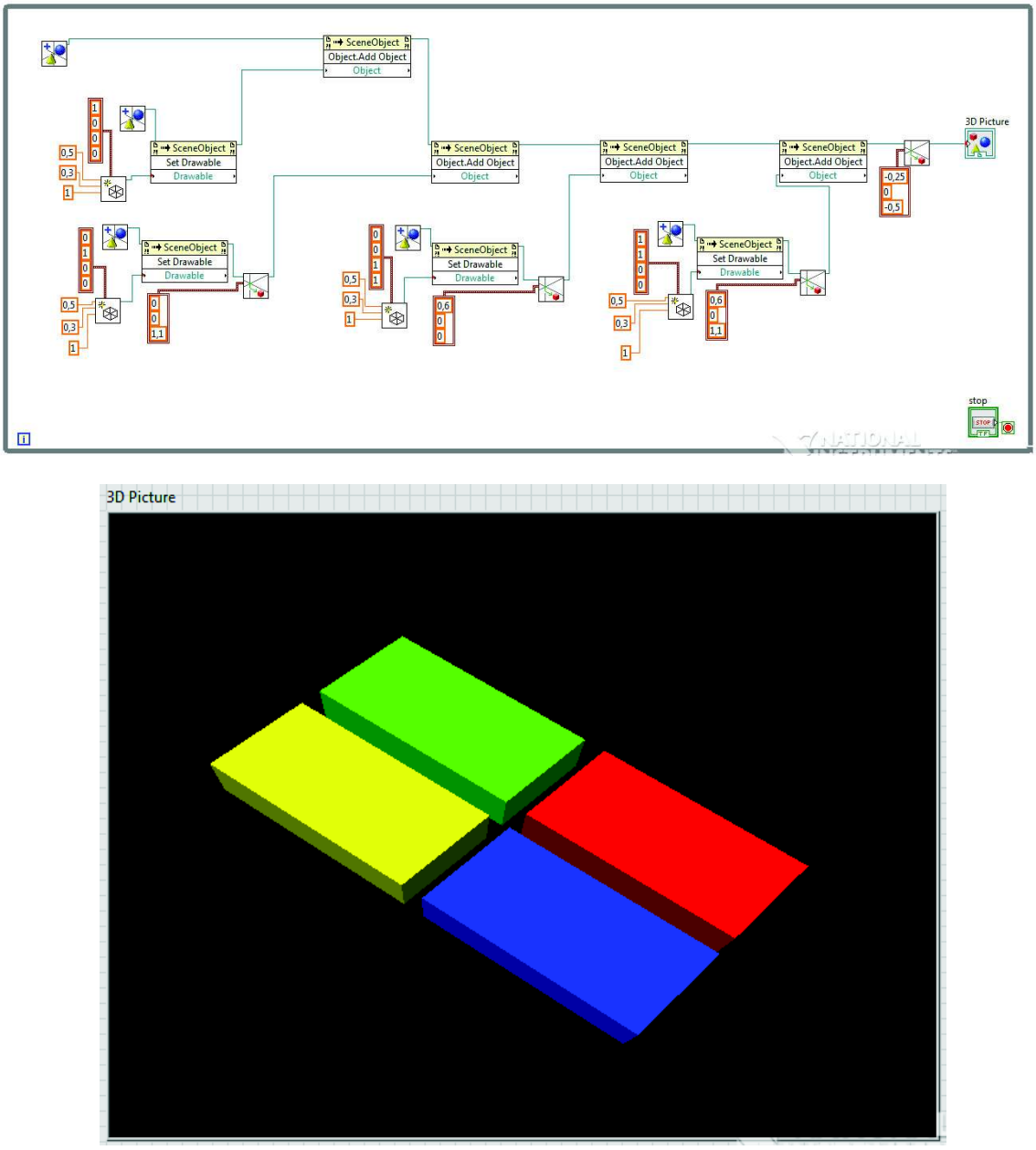


Figura 54 – Criação das 4 placas do exemplo no diagrama de blocos e resultado no painel frontal.

Observe que um deslocamento final aplicado ao cenário total é efetuado e permite uma melhor centralização da imagem 3D.

Agora temos o desafio, simular vibrações nos três eixos das quatro placas de forma independente. Vamos fazer um simples exemplo aplicado a uma das placas, em um dos eixos, x, conforme ilustrado na figura abaixo.

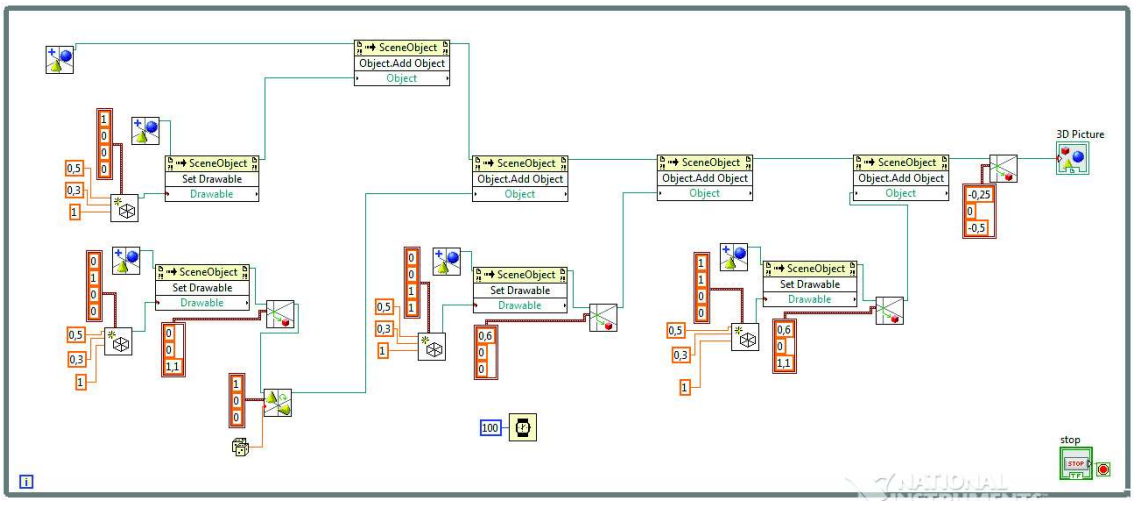


Figura 55 – Exemplo de aplicação de vibração a uma das placas do exemplo no diagrama de blocos.

Aqui utilizamos uma VI *Rotate Object* e limitamos este efeito a um eixo de forma aleatória, usando a VI *Random* para este objetivo. Como este trabalho é ilustrativo e a VI *Rotate Object* possui entrada de ângulo de rotação da imagem em radianos, o efeito é extremamente pronunciado. Outro recurso utilizado foi temporizar o *Loop* com a VI *Wait* (ms) para que possamos ver as vibrações com detalhes. Confira os resultados no painel frontal.

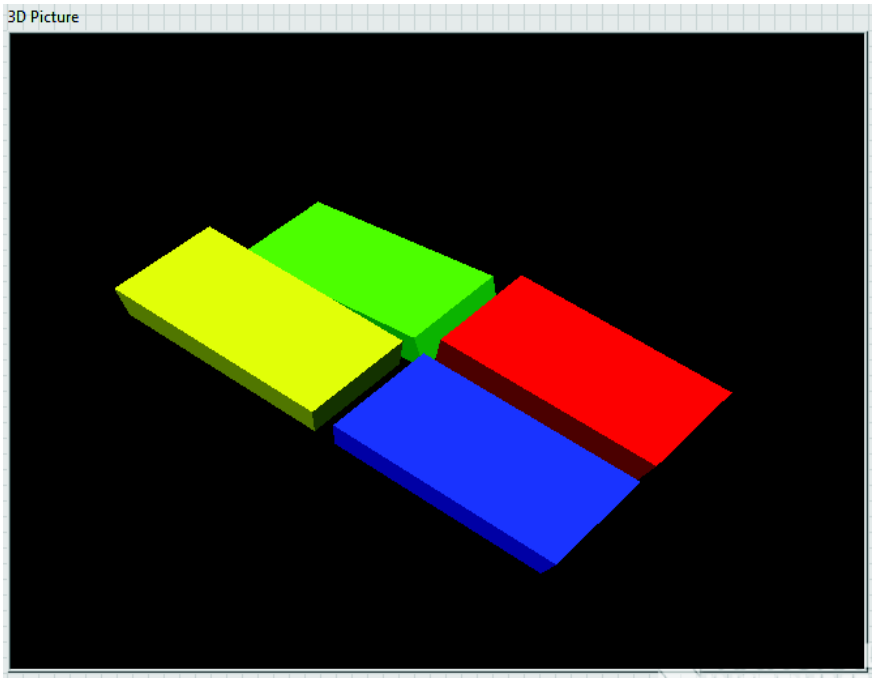


Figura 56 – Vibração no eixo x em uma das placas do exemplo (verde) no painel frontal.

Procure montar este exemplo e testar sua aplicação. Depois de explorar este exemplo, atribua as vibrações as demais placas e analise o resultado. Uma forma bastante eficiente de implementar os módulos de vibração nos três eixos é criar sub-VIs como verificamos no capítulo 4.4.

Podemos considerar esta situação como uma situação real de validação de um sistema de ensaios de segurança e desempenho. A aplicação evidentemente pode ser simples, mas pode abrir muitas possibilidades e trazer o contato com o ambiente real de forma muito integrada, o que nos proporciona a ferramenta de imagens 3D.

6.6 Clusters

Um cluster é um conjunto de dados ou elementos mistos, ou seja, de tipos diferentes. A utilização de clusters tem o objetivo de reduzir o número de ligações e terminais necessários em uma VI ou SubVI. Numa SubVI, o objetivo maior é a redução de conexões de seus conectores de entrada e saída, que possuem limites restritos de variáveis de entrada e saída conforme já vimos anteriormente.

Os clusters também podem agrupar informações para exibição em *Waveforms* e *Charts*. A VI *Bundle* executa exatamente esta ação de agrupar as informações, como já vimos anteriormente.

Como num Array, um Cluster no painel frontal pode ser um controle ou indicador, sem poder ser uma mistura de ambos.

Adicionar elementos em um cluster no painel frontal é muito similar ao procedimento executado na criação de Arrays, com a diferença essencial de que num cluster os elementos podem ser de tipos diferentes. Os cluster podem ser inseridos pela sub-paleta de controles *Array & Cluster*.

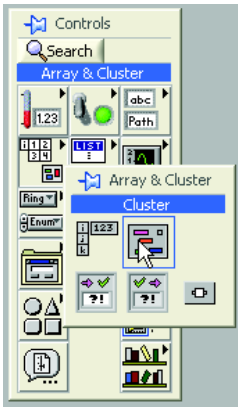


Figura 57 – Exibição do grupo *Array & Cluster* no painel frontal do LabVIEW.

Um Cluster pode ter seu tamanho alterado prontamente com o mouse, de forma simples, para que você possa inserir a quantidade necessária de elementos no mesmo de forma clara e visível para sua aplicação. É sempre importante lembrar que todos os elementos em um cluster devem ser padronizados como controles ou indicadores, podendo ser de diversos tipos: Booleanos, *Strings*, Numéricos, etc. Abaixo temos um cluster de controle.

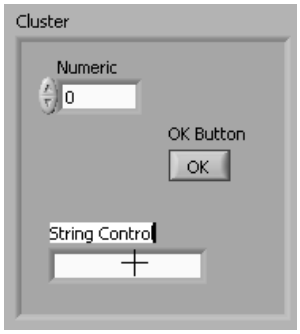


Figura 58 – Ilustração de um Cluster de controle no painel frontal do LabVIEW.

Outro ponto interessante é que VIs polimórficas podem trabalhar com clusters diretamente.

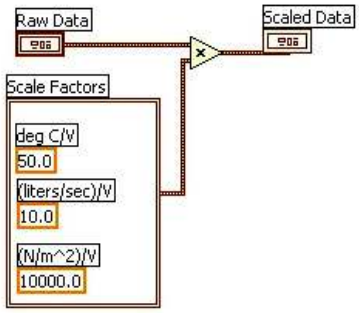


Figura 59 – Polimorfismo do Cluster no Diagrama de Blocos do LabVIEW.

Também podemos criar Clusters no diagrama de blocos no Lab VIEW, da mesma forma como efetuado no painel frontal. Vamos efetuar os mesmos procedimentos do painel frontal no diagrama de blocos para explorar melhor este ponto que é muito importante na assimilação desta ferramenta.

Podemos ainda no diagrama de blocos criar um cluster de constantes baseado num Cluster existente, lembrando que estas constantes não podem ser alteradas durante a execução da VI. Esta ação pode ser executada com o botão direito do mouse sobre o cluster existente.

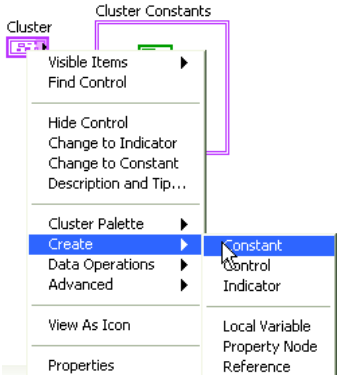


Figura 60 – Criação de Cluster de constantes no Diagrama de Blocos do LabVIEW.

6.6.1 Ordenando elementos num Cluster

Uma configuração muito importante para o trabalho com Clusters é a ordem de seus elementos, já que as entradas e saídas devem respeitar uma sequência para que possamos posteriormente processar diferentes tipos de variáveis.

Para configurar a ordem dos elementos dentro de um Cluster você deve clicar com o botão direito do mouse sobre a borda do mesmo e selecionar a opção "*Reorder Controls in Cluster*". Você perceberá que os elementos apresentaram uma indexação para ordenação estabelecida com a ordem com que você inseriu os elementos no Cluster. Você pode então editar cada elemento para que se ordenem novamente de acordo com as necessidades de seu programa.

Primeiramente preencha o campo com o valor da ordem que deseja atribuir a um elemento e então selecione o elemento a ser colocado na ordem desejada. A nova ordem dos elementos do Cluster será apontada pelas caixas pretas. Para confirmar as alterações utilize o botão OK, que automaticamente retornará ao Cluster no painel frontal ou diagrama de blocos.

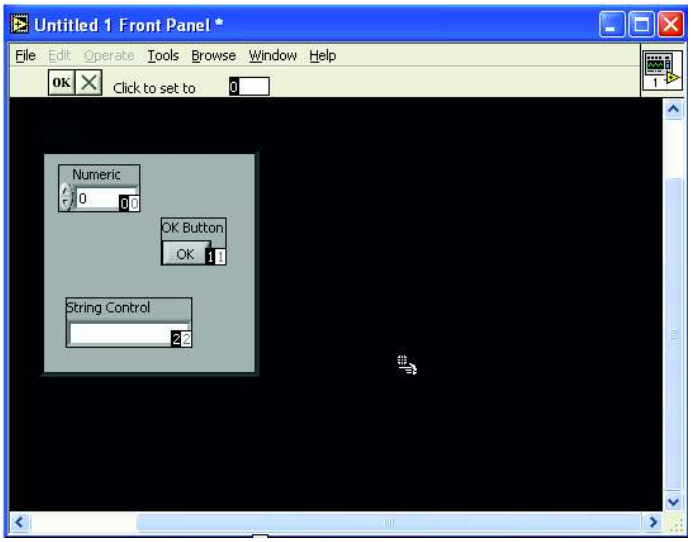


Figura 61 – Alterando a ordem dos elementos do Cluster no Painel de Controle do LabVIEW.

6.6.2 Criando e manipulando Clusters

Na paleta de funções no diagrama de blocos temos a sub-paleta Cluster, onde podemos criar um Cluster. Temos ainda nesta paleta as funções *Bundle* e *Umbundle* (que podem ser "*by name*") para compor e decompor Clusters, respectivamente.

A utilização da composição, *Bundle*, é aplicável quando queremos inserir elementos em um cluster ou quando queremos modificar valores de elementos individuais de um Cluster existente.

Utilizamos este recurso para definir parâmetros de um *Array* como vimos na composição de gráficos.

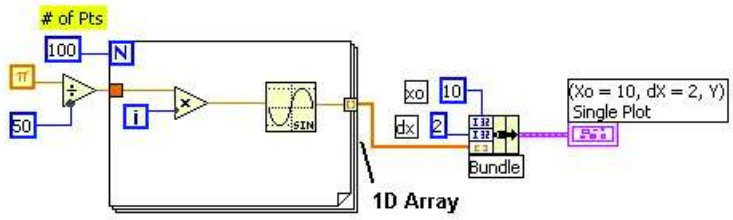


Figura 62 – Definição de parâmetros de um Array compondo um Cluster no Diagrama de Blocos.

A alteração de parâmetros de um cluster com a VI *Bundle* é outra importante função que é exemplificada abaixo, onde *New Command* estabelece o novo valor do primeiro elemento do Cluster, que é justamente a String "*Command*". Este procedimento é muito utilizado nas rotinas feitas em LabVIEW durante a manipulação de Clusters pré-existentes.

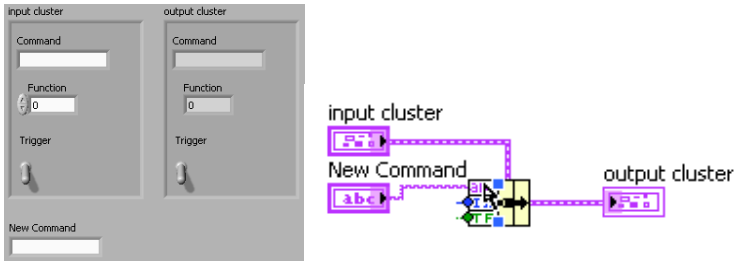


Figura 63 – Alteração de parâmetros de um Cluster.

Outra ferramenta importante para manipulação de Clusters é a VI *Bundle by name*, utilizada sempre que queremos acessar ou alterar elementos de um Cluster. Ela opera da mesma forma que a VI Bundle sem referenciar-se aos elementos por ordem e sim por nome (*lable*).

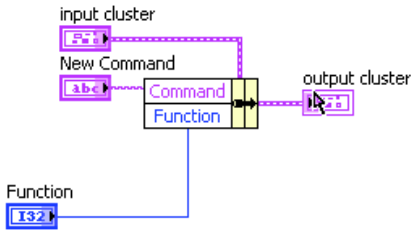


Figura 64 – Alteração de parâmetros de um Cluster com a VI *Bundle by name*.

Temos ainda a aplicação das VIs *Unbundle* e *Unbundle by name* que oferecem recursos similares para obtermos valores e estados de elementos contidos em um Cluster.

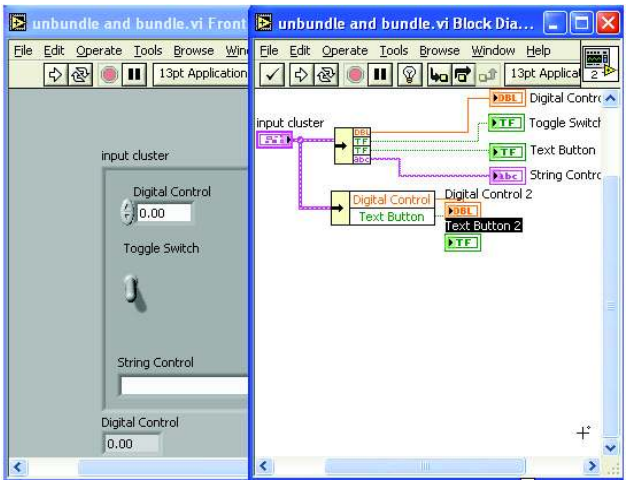


Figura 65 – Obtenção de parâmetros de um Cluster com a VI *Umbundle* e *Unbundle by name.*

É interessante lembrarmos que nas funções *Bundle by name* e *Unbundle by name*, podemos acrescentar elementos e alterar suas entradas e saídas com o botão direito do mouse sobre cada ponto e selecionando os objetos.

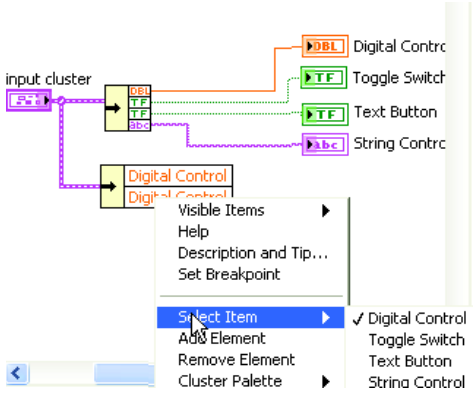


Figura 66 – Escolha de parâmetros de um Cluster com a VI *Unbundle by name.*

Exercício 10: Manipulando Clusters

Construa uma VI com um Cluster controle e um indicador, que será modificado. O cluster deve conter duas variáveis numéricas e duas booleanas. Crie adicionalmente um cluster reduzido, indicador, com uma variável numérica e uma booleana, onde a numérica deve conter o resultado da segunda variável numérica do cluster principal (controle) e da primeira booleana. As variáveis restantes devem ser apresentadas em indicadores não pertencentes a Clusters, ou seja, indicadores individuais. O cluster indicador total deverá manipular os resultados da seguinte forma: Incrementar o valor da primeira variável numérica e negar (inverter) o valor da segunda variável booleana. Monte a estrutura em um while loop para facilitar a visualização dos resultados de forma contínua.

7 Estruturas CASE e SEQUENCE

7.1 Estrutura CASE

A estrutura CASE utilizada pelo LabVIEW é similar a uma estrutura CASE de qualquer linguagem de programação baseada em texto. Certamente quem já programou ou programa em linguagens baseadas em texto sabe a importância de estruturas como *If....then...else*. As estruturas CASE no LabVIEW possuem dois ou mais subdiagramas, onde somente um deles é exibido durante a edição do código e também executado por vez.

A criação de estruturas no diagrama de blocos tem início na paleta de funções *Structures*. Nela podemos encontrar prontamente a estrutura CASE e adicioná-la a nosso programa.

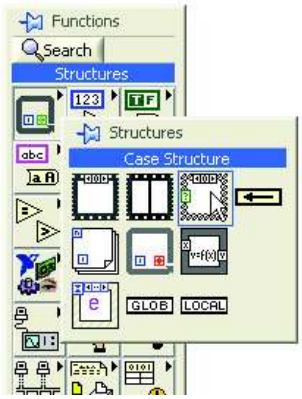


Figura 67 – Acesso a estrutura CASE no Diagrama de blocos.

A estrutura padrão de uma estrutura CASE no LabVIEW é booleana, com as condições TRUE e FALSE.

Como exemplo podemos criar uma VI que calcula o quadrado de um número qualquer de entrada, porém, quando ele é negativo, retorna uma notificação de erro. As duas condições utilizadas nesta VI estão ilustradas abaixo:

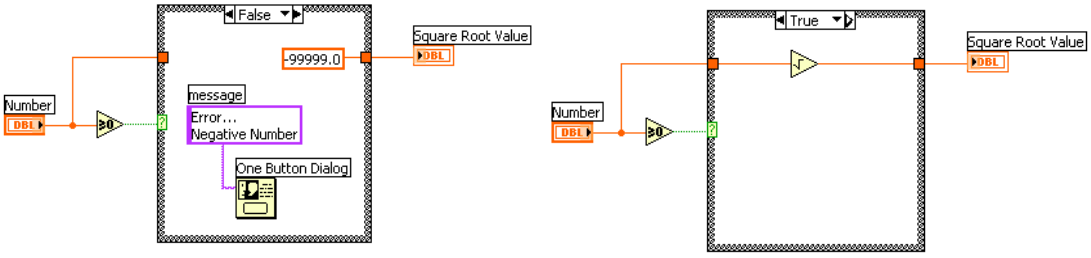


Figura 68 – Exemplo de utilização da estrutura CASE no Diagrama de blocos.

Procure montar esta VI e explorar a operação desta estrutura CASE. Aproveite também para conhecer novas VIs, como as VIs de caixas de diálogo, muito úteis durante as programações com sequências e estruturas.

Você pode criar múltiplas entradas e saídas em uma mesma estrutura CASE, o que facilita a tomada de decisões com controle de múltiplas variáveis.

Vamos montar uma segunda VI que explora esta propriedade de múltiplas entradas e saídas, exibida na figura abaixo. Observe que algumas entradas podem simplesmente ser suprimidas em algumas das condições, no entanto devem ser empregadas em alguma delas. Já as variáveis de saída devem possuir conexões em todas as condições presentes na estrutura CASE.

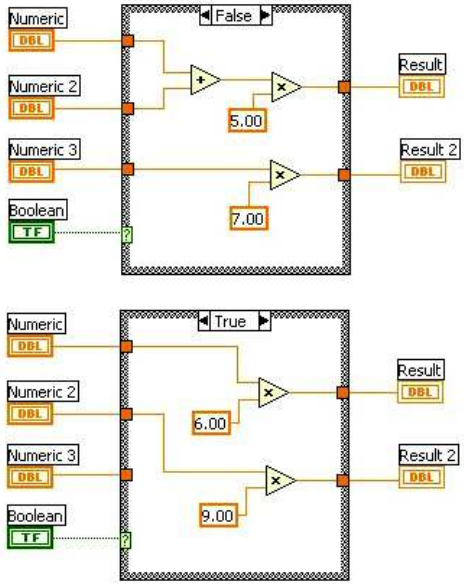


Figura 69 – Exemplo de utilização da estrutura CASE no Diagrama de blocos.

Observe agora que quando alteramos a variável condicional de entrada, podemos alterar o tipo de operação CASE, utilizando ao invés de Booleanos, condição padrão no LabVIEW, variáveis numéricas, strings, etc.

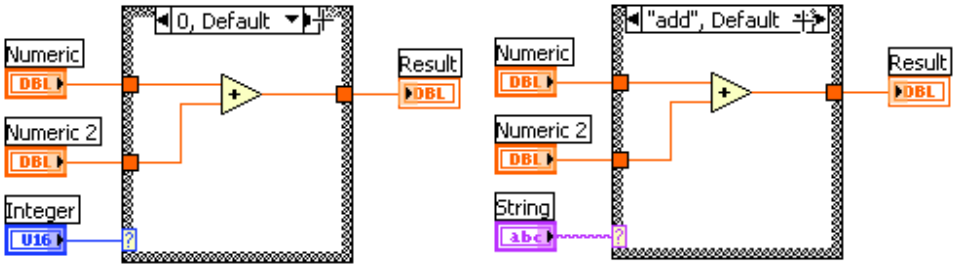


Figura 70 – Utilização de variável não booleana na estrutura CASE.

É interessante também a possibilidade de edição dos estados, selecionando com o mouse os mesmos e modificando seu indicador para um indicador de interesse, como por exemplo, alterar um indicador de estado numérico 1, para 4, o que implica que aquela ação será executada quando sua entrada for igual a 4 e não mais a 1.

Exercício 11: VI para controle de temperatura

Construa uma VI que aciona um BEEP e emite um aviso de erro (Mensagem) quando um determinado número ultrapassa um valor. Como sugestão, utilize a VI de conversão de temperatura, gerando temperaturas aleatórias e detectando as mesmas para avaliação de limites e atuação de controle.

7.2 Estrutura SEQUENCE

Uma estrutura sequencial é baseada em um conjunto de subdiagramas, ou frames, que executam instruções sequenciais, iniciando pelo frame de índice "0" e seguindo pelos frames 1, 2 , etc.

Este tipo de estrutura não completa uma execução ou retorna dados até que o último frame seja executado. Desta forma, esta estrutura deve ser utilizada quando a interdependência de dados não existe durante sua execução. Sua maior aplicação é para a geração de rotinas de inicialização ou preparação anterior a execução da função específica de processamento de dados de um programa.

Outro ponto importante neste tipo de estrutura é que, diferentemente da estrutura CASE, esta só pode ter como entrada um único dado, que ficará disponível para todos os frames sequenciais.

As estruturas sequenciais estão disponíveis na sub-paleta de funções *Structures*.

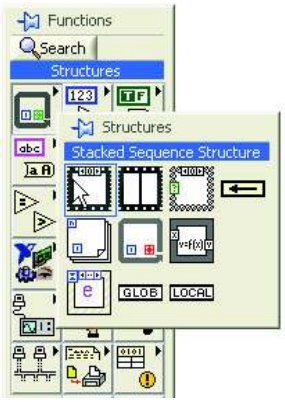


Figura 71 – Acesso a estrutura *Sequence* no Diagrama de blocos.

Depois de colocada uma estrutura sequencial no diagrama de blocos, podemos configurar seus detalhes como adicionar, remover ou duplicar frames, com o botão direito do mouse posicionado em uma das bordas da estrutura. Na borda superior podemos navegar nos frames criados para sua edição individual.

Um bom exemplo de aplicação da estrutura sequencial é a VI *Time to Match*. Esta VI calcula o tempo que um gerador randômico leva para atingir um número específico indicado pelo usuário.

Abaixo vemos o diagrama de blocos. Procure criar uma VI como esta para se familiarizar com seus componentes, e observe que o Frame 0 irá executar uma comparação até que esta seja atendida e então passe para a segunda etapa, Frame 1. Neste Frame final, o tempo corrente é obtido e subtraído o tempo corrente inicial que é executado juntamente com o primeiro Frame (0).

Utilize o *righlight execution* para entender melhor o funcionamento de sua VI.

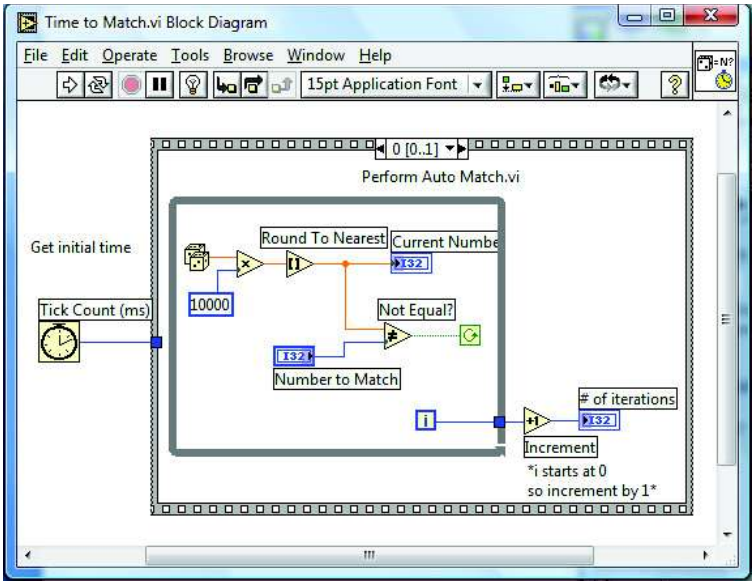


Figura 72 – VI *Time to Match* no Diagrama de blocos, evento "0".

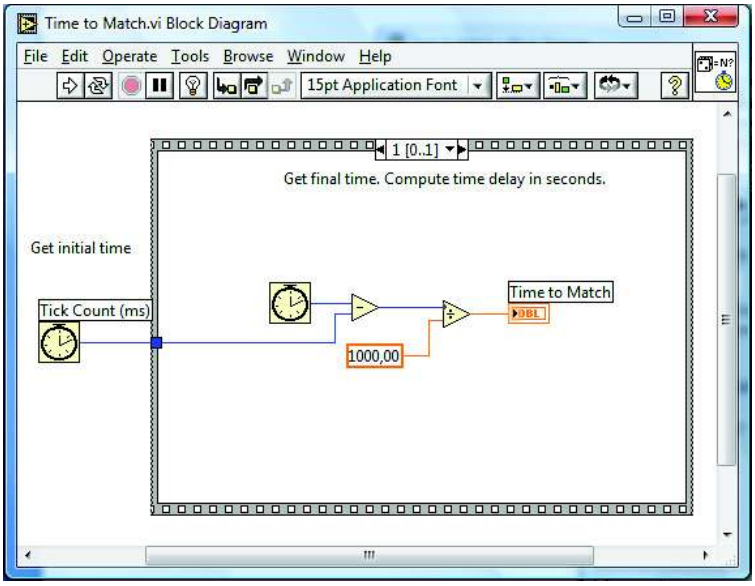


Figura 73 – VI *Time to Match* no Diagrama de blocos, evento "1".

7.2.1 Sequence Locals

Quando precisamos transferir dados entre frames de uma estrutura sequencial precisamos utilizar os *Sequence Locals*, que são portas de saídas e entradas de dados especificamente aplicados nestas estruturas. Ao utilizarmos um *Sequence* Local veremos que um seta de saída é automaticamente disponibilizada para os frames subsequentes, para que os mesmos possam receber os resultados resultantes de uma ação anterior.

É importante saber que não poderemos utilizar o dado de um frame com uma *Sequence Local* origem, em um Frame anterior, se esta estrutura estiver dentro de um loop por exemplo. O dado estará disponível somente a partir de uma origem num frame onde foi estabelecido o *Sequence Local*, para os frames posteriores.

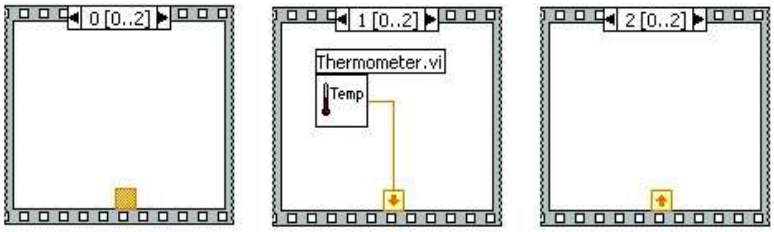


Figura 74 – Exemplo de utilização da estrutura *Sequence* no Diagrama de blocos.

Também devemos lembrar que dados não poderão ser transmitidos para fora de uma estrutura SEQUENCE antes do final de sua execução completa, ou seja, não podemos utilizar os dados imediatamente de um frame no meio de uma sequência.

Para inserir um *Sequence Local*, selecione a borda superior ou inferior da estrutura e com o botão direito do mouse selecione a opção de inserção. Ao conectar qualquer fio a este *Sequence Local*, uma seta de saída é apresentada no frame de origem e setas de saída serão apresentadas nos frames subsequentes.

7.2.2 Substituindo estruturas sequenciais por estruturas CASE

A utilização demasiada da estrutura sequencial não é recomendada em muitos programas elaborados no LabVIEW, isto se deve, principalmente, a impossibilidade de execução de tarefas paralelas a execução destas estruturas. Um exemplo interessante é a utilização de tarefas assíncronas, como PCI, GPIB, SERIAL e DAQS, que podem ser executadas concomitantemente com outras, desde que não estejam inseridas em estruturas sequenciais.

No entanto, sabemos que mesmos dentro destas tarefas, que normalmente envolvem entradas e saídas de dados e controle e aquisição por hardware, é sempre necessário sequenciar tarefas, o que implica na utilização de outros recursos, quando queremos que outras ações ocorram concomitantemente. Uma forma interessante de sequenciar dados e ações é utilizar a ordem por erros de I/O.

A atualização de status também não deve ser utilizada com dependência de estruturas sequenciais, pois somente um frame estabelecerá o status, ao final da execução completa da estrutura. Nestes casos a utilização de uma estrutura CASE em um *While Loop* é mais interessante e como esta estrutura pode trabalhar com maior número de entradas e saídas, pode ser mais versátil nas aplicações pretendidas.

O exemplo abaixo apresenta um caso onde a estrutura CASE é mais recomendada, pois a execução de duas SubVIs distintas que alteram o estado de um mesmo indicador de status é necessária. Observe que a interação com o contador do *While Loop* (“i”) garante uma operação sequencial a VI com uma funcionalidade bem mais interessante.

Outro ponto interessante é que dentro de qualquer frame da estrutura CASE podemos interromper a operação do *While Loop*, com o uso de uma variável booleana, já que não é necessário aguardar o término da sequência.

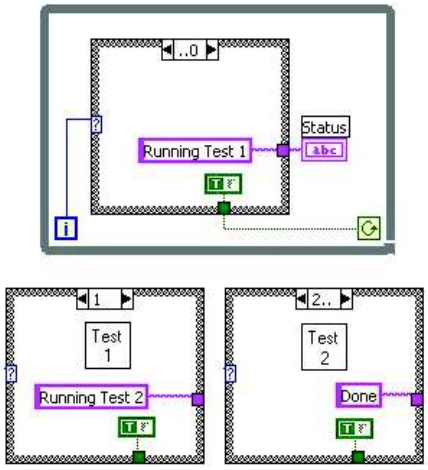


Figura 75 – Exemplo de substituição da estrutura *Sequence* pela estrutura CASE no Diagrama de blocos.

7.3 *Formula Node* e *Expression Node*

Muitas vezes as operações matemáticas que desejamos aplicar a nossos programas requer a concatenação de inúmeras VIs de operações aritméticas entre outras, o que torna o programa pesado e complexo para desenvolvimento e debug.

Nestes casos, o mais recomendado é utilizar *Formula Nodes* e *Expression Nodes*, onde podemos visualizar melhor nossas funções e operações e melhorar até o aspecto visual de nossos diagramas de blocos em nossas VIs.

7.3.1 Formula Nodes

O *Formula Node* é uma VI para expressão de códigos baseados em texto para operações matemáticas. Estes Nodes são muito interessantes quando temos um grande número de variáveis para processar. No exemplo abaixo temos um *Formula Node* que executa a função $y=x^2+x+1$, onde é possível observar a similaridade com expressões utilizadas em linguagem C.

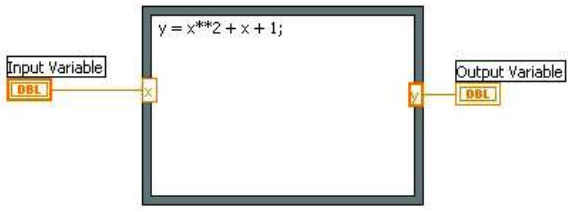


Figura 76 – Exemplo de utilização da estrutura *Formula Node* no Diagrama de blocos.

Dentro do bloco podemos inserir nossas fórmulas baseadas em texto, e nas bordas esquerda e direita podemos estabelecer nossas entradas e saídas, respectivamente, utilizando o botão direito do mouse sobre as mesmas.

O *Formula Node* está disponível na sub-paleta de funções *Structures*, no diagrama de blocos.

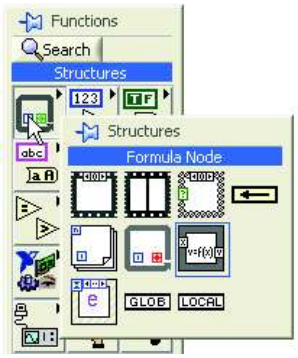


Figura 77 – Acesso a estrutura *Formula Node* no Diagrama de blocos.

O *Formula Node* pode ainda executar outras operações como condicionais, loops internos, etc. Estas funções podem ser exploradas separadamente na ajuda do LabVIEW e requer um pouco de familiaridade do programador com linguagem C.

7.3.2 Expression Nodes

Os *Expression nodes* devem ser utilizados quando possuímos uma única variável em uma equação. É uma opção interessante quando temos certa complexidade em nossa função porém uma única variável.

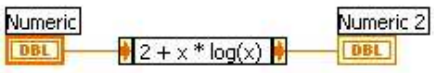


Figura 78 – Exemplo de utilização da estrutura *Expression Node* no Diagrama de blocos.

Dentro do bloco podemos inserir nossas fórmulas baseadas em texto, e nas bordas esquerda e direita podemos estabelecer nossas entradas e saídas, respectivamente, ligando a ela nossos controles e indicadores.

O *Expression Node* está disponível na sub-paleta de funções *Structures*, no diagrama de blocos.

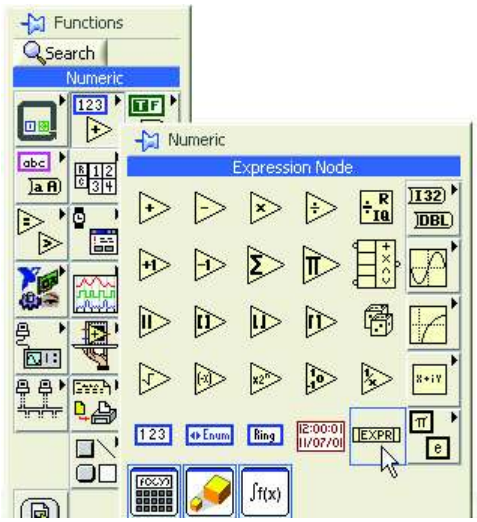


Figura 79 – Acesso a estrutura *Expression Node* no Diagrama de blocos.

Exercício 12: Formula Node com saída gráfica

Construa uma VI que calcula as seguintes equações: B=A*sin(X), Y=B**3+B. Utilize a saída Y para exibir o resultado graficamente, após 100 interações, onde X é a interação i dividida por 20.

8 Análise e processamento de sinais

O LabVIEW traz uma série de recursos relacionados à aquisição, geração e processamento de sinais analógicos e digitais. São disponibilizadas VIs para configuração, aquisição e envio de dados a dispositivos de aquisição de sinais (DAQ). Nesta etapa, poderemos visualizar como o LabVIEW trabalha com estes hardwares e ainda como podemos simular sinais para geração e medição de parâmetros para que a aplicação futura em DAQs e outros dispositivos tenha sucesso.

8.1 Geração de formas de onda

A primeira etapa a ser trabalhada é a geração de formas de onda para pós-processamento, ou seja, a simulação de sinais no ambiente LabVIEW. Para isso vamos abrir um exemplo em *NEW>VI from Template>Generate and Display*. Utilizando o modo *Large dialog* na caixa de diálogo *NEW*, podemos ter uma prévia do ambiente:

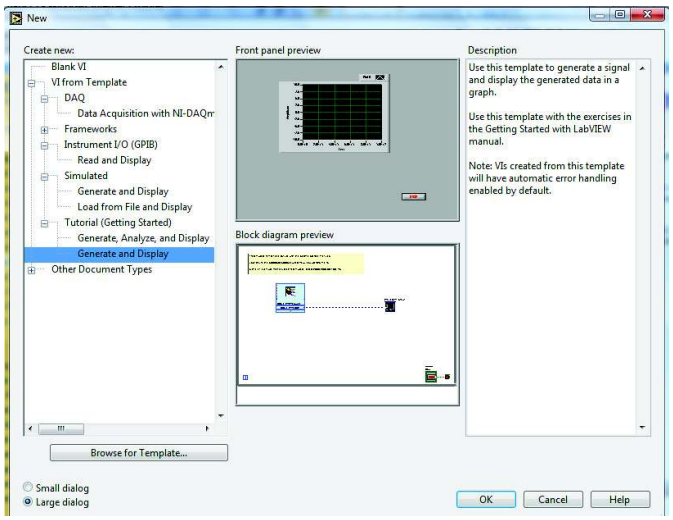


Figura 80 – Exemplo *Generate and Display*.

Abrindo este modelo temos a apresentação do painel frontal com um *Waveform Graph* e um botão STOP.

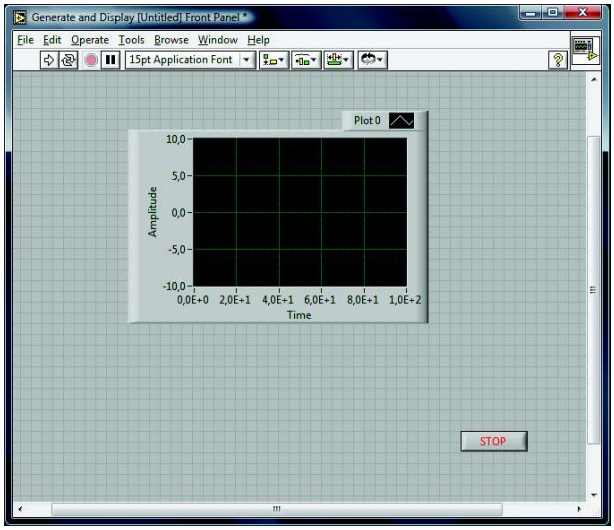


Figura 81 – *Template Generate and Display* no Painel Frontal.

Exibindo o diagrama de blocos por meio do menu Window, temos a conexão de uma Express VI *Simulate Signal*. Com o acesso as propriedades desta VI podemos configurar vários de seus atributos. Correndo o mouse externamente à Express VI *Simulate Signal* podemos visualizar suas entradas e saídas, o que pode ser visualizado também pela expansão vertical gráfica da VI por meio da dupla seta em sua base.

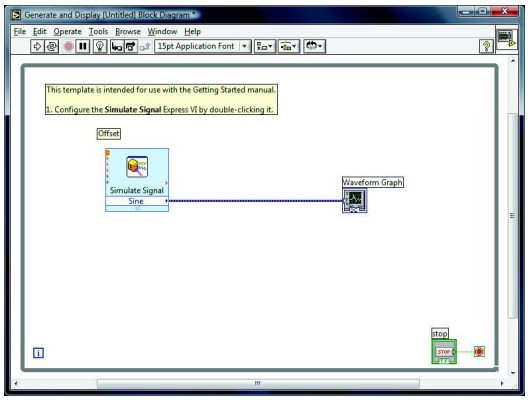


Figura 82 – *Template Generate and Display* no Diagrama de Blocos.

Voltando ao painel frontal, podemos prontamente rodar este *Template* e visualizar a forma de onda gerada pela *Express* VI no *Waveform Graph*. Podemos agora no painel frontal configurar as propriedades do *Waveform Graph* para que o mesmo se adéque a apresentação desejada do projeto.

No diagrama de blocos podemos editar a forma de onda de saída, explorando propriedades como Tipo da forma de onda, amplitude, frequência, off-set, número de amostras, entre outros. Procure explorar estas informações e conhecer melhor esta Express VI que será amplamente utilizada em suas aplicações.

Podemos agora acrescentar controles de parâmetros de entrada à Express VI como por exemplo amplitude. No painel frontal coloque um botão de controle por meio da caixa *Control Pallets* disponível no menu *Window*.

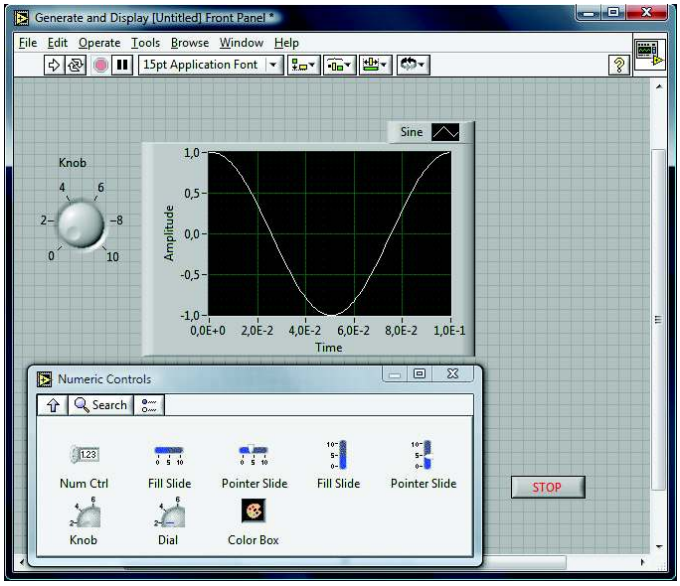


Figura 83 – Manipulando o *Template Generate and Display* no Painel Frontal.

No diagrama de blocos efetue a ligação do Knob à entrada Amplitude da Express VI. Retorne ao painel de controle e rode a VI para verificar a atuação deste comando. Este simples exemplo começa mostrar um pouco da capacidade de interação para simulação de sinais propiciada pelo LabVIEW. Podemos montar um gerador de largura de pulso modulada (PWM), sem componente negativa, variando parâmetros de uma forma de onda quadrada, como *off-set* e *duty cycle*. Este tipo de gerador pode ser muito útil no controle de inversores de geradores e motores.

Exercício 13: Gerador PWM

Agora vamos montar um gerador PWM, sem componente negativa, com a utilização dos recursos de simulação de sinais, porém sem a utilização dos recursos internos da Express VI para seleção de *Duty Cycle*. Primeiramente faremos um gerador PWM com *Duty Cycle* variável lembrando que o princípio básico de funcionamento de um gerador PWM é a comparação de formas de onda Dente de Serra e DC de amplitude variável. Um exemplo do resultado deste exercício (painel frontal) é ilustrado abaixo.

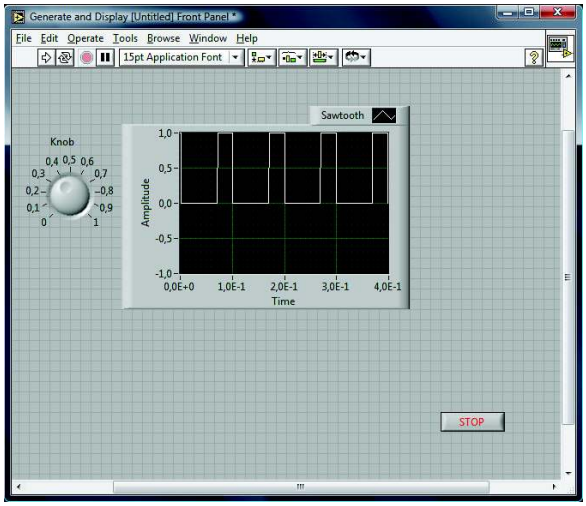


Figura 84 – Manipulando o *Template Generate and Display* para controle de *duty cycle*.

Podemos utilizar outros recursos para criação e posterior edição de atributos de formas de onda utilizando o LabVIEW. Utilizando a mesma VI do modelo (*templates*), podemos modificar a forma de geração de sinais, utilizando agora uma VI diferente de uma Express VI:

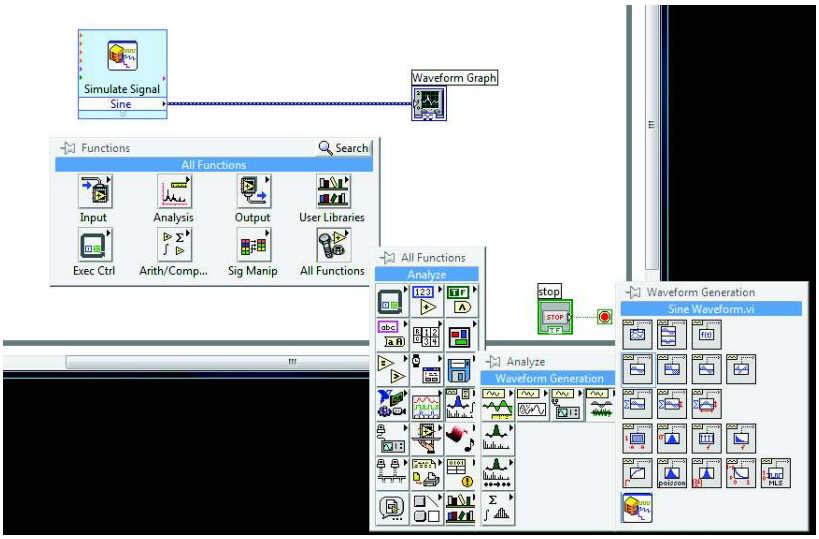


Figura 85 – Manipulando o *Template Generate and Display*.

Utilizando por exemplo a VI *Basic Function Generator*, podemos gerar a forma de onda PWM da mesma forma que nos exemplos e exercícios acima, disponibilizando os controles externamente, da mesma forma que efetuaríamos na Express VI. Neste caso devemos atentar para os valores default da VI que podem ser obtidos por meio do *context help* na opção "*more help*".

Utilizando o botão direito do mouse podemos criar controles rapidamente para todos os parâmetros da VI:

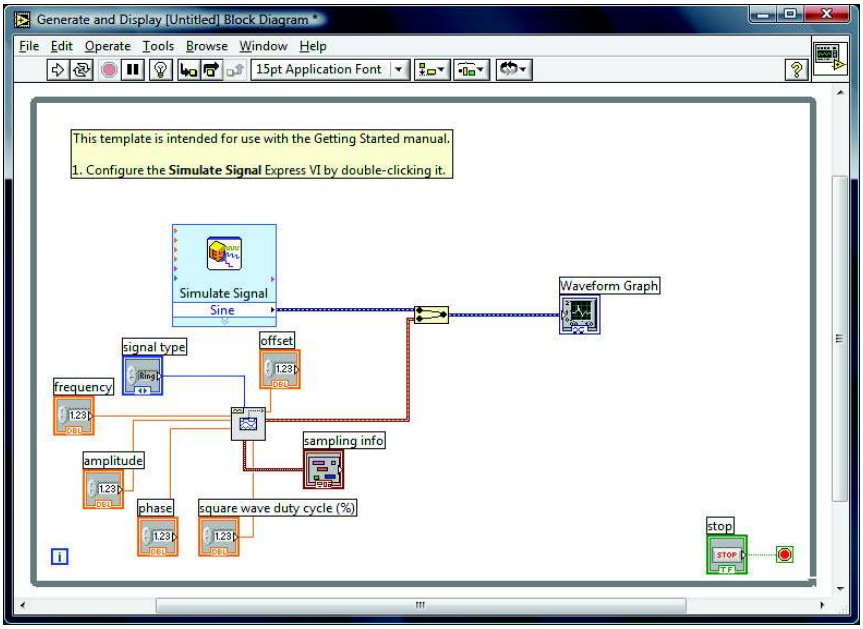


Figura 86 – Manipulando o *Template Generate and Display*.

Retornando ao painel frontal podemos visualizar todos os controles gerados no diagrama de blocos. Estes podem ser organizados e visualizados simultaneamente com o sinal gerado pela Express VI. Obviamente podemos colocar controles na Express VI e manipular os dois sinais livremente.

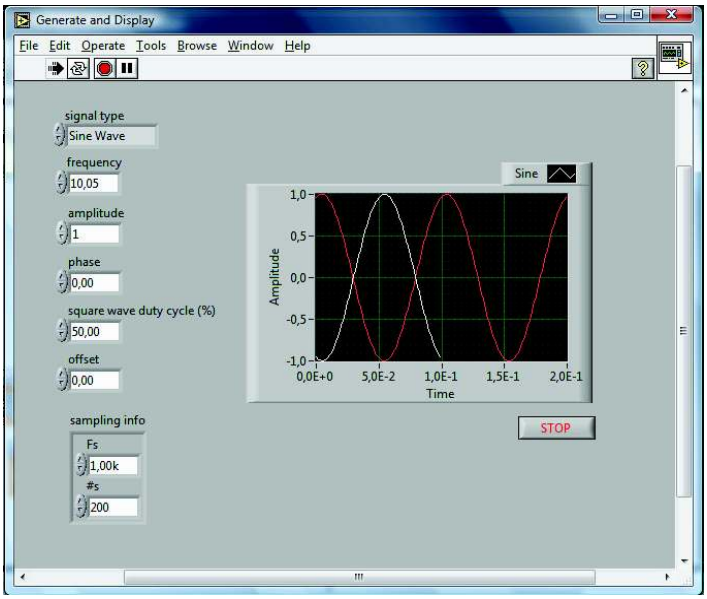


Figura 87 – Manipulando o *Template Generate and Display*.

Com isso podemos utilizar as VIs de geração de formas de onda da forma mais conveniente possível e estabelecer inclusive simulações de ruídos adicionados aos sinais.

Podemos ainda gerar sinais digitais ou converter sinais analógicos para digitais e vice-versa utilizando as VIs disponibilizadas pelo LabVIEW, de acordo com cada aplicação desejada.

Além das VIs de geração de formas de onda da *Function Pallet* "*Waveform*" temos as VIs da *Function Pallet* "*Analyze*" (ou *Signal Processing>>Sig Generation* no LabVIEW após versão 8):

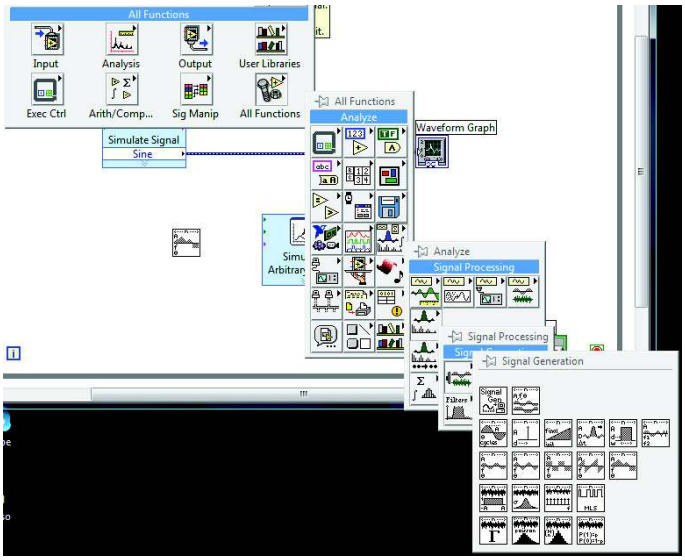


Figura 88 – Manipulando o *Template Generate and Display*.

Vamos utilizar, por exemplo, um padrão senoidal com VI "*Sine Pattern*". As configurações agora não são exatamente como nos exemplos anteriores, e devemos obter ajuda detalhada do LabVIEW para entendermos seu procedimento de utilização.

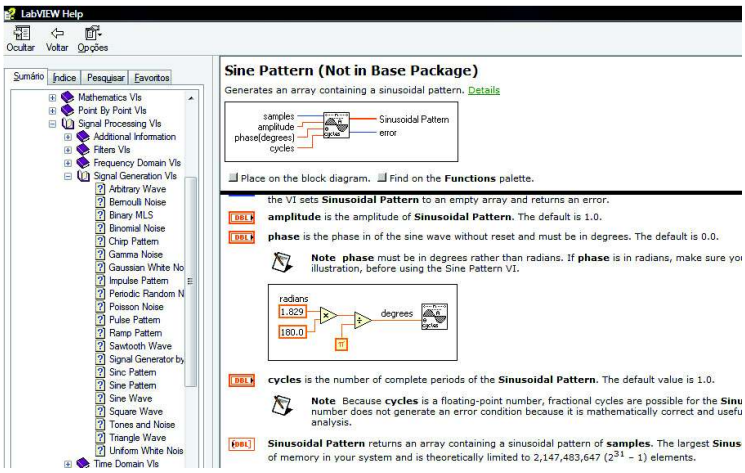


Figura 89 – Detalhamento da VI "*Sine Pattern*".

Podemos observar primeiramente que o ângulo de fase é em graus e não em radianos, o que requer um pré-processamento quando queremos trabalha em radianos. Outro detalhe é

estabelecer um sinal com frequência conhecida, já que a VI trabalha com número de ciclos e amostras, como mostra a equação no detalhamento da VI.

Sine Pattern Details

If the sequence *Y* represents **Sinusoidal Pattern**, the Sine Pattern VI generates the pattern according to the following equations.

$$y_i = \text{asin}(x_i)$$

$$x_i = \frac{2\pi k}{n} + \frac{\pi\phi_0}{180}$$

for $i = 0, 1, 2, \ldots, n - 1$,

where *a* is the **amplitude**, *k* is the number of **cycles** in the pattern, ϕ_0 is the initial **phase** in degrees, and *n* is the number of **samples**

Figura 90 – Detalhamento da VI "*Sine Pattern*".

Se utilizarmos a fase 0 (zero), temos a frequência angular vezes a razão amostra/número de amostras na equação modelo do sinal. A frequência do sinal de saída é obviamente a razão do número de ciclos pelo número de amostras e o sinal será repetido por n ciclos estabelecidos. A frequência 1 não é alcançada, e uma equidade do número de ciclos e amostras resulta em sinal contínuo dependente da fase. Assim, a relação de ciclos e amostras é fundamental na simulação do sinal por meio desta VI.

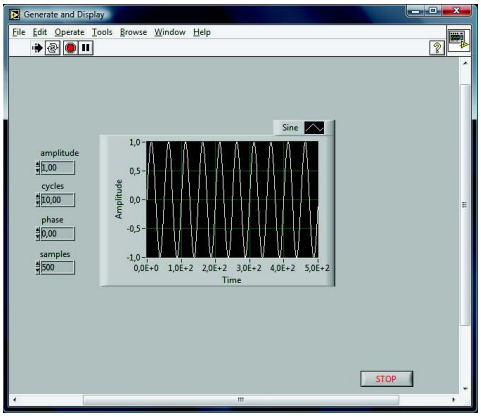


Figura 91 – Utilizando a VI "*Sine Pattern*".

Além destes exemplos de simulação de formas de onda podemos gerar sinais arbitrários e para isso utilizaremos duas VIs ilustradas abaixo. Uma delas proveniente da paleta *Analyze* (ou *Signal Processing>>Sig Generation* a partir do LabVIEW 8):

Outra (Express VI) proveniente da paleta Input e muito importante quando queremos trabalhar sinais arbitrários em aplicações ou simulações e validações de dados é a VI *Simulate Arbitrary Signal*.

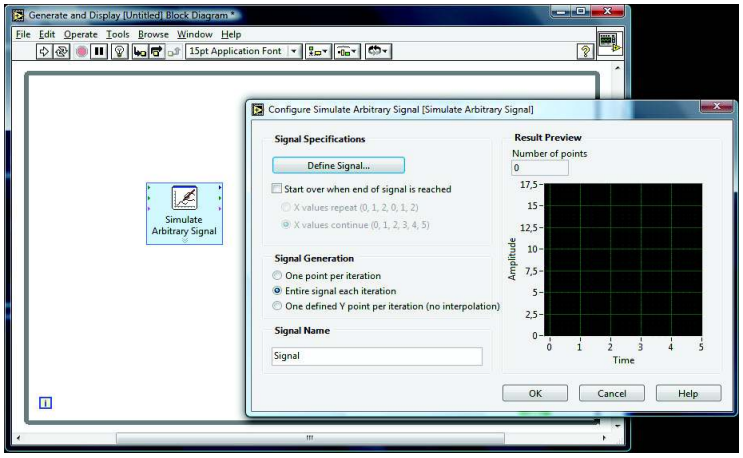


Figura 92 – Utilizando a VI "*Simulate Arbitrary Signal*".

Nesta Express VI é possível configurar ponto a ponto o sinal de saída a ser simulado, levanto em conta a variação de tempo para se estabelecer o período do sinal e sua frequência. Na VI *Simulate Arbitrary Signal* os sinais são computados em forma de tabela.

Exercício 14: Sinal arbitrário

Utilize agora a Express VI e a VI *Simulate Arbitrary Signal* para gerar sinais coincidentes, lembrando que os parâmetros de entrada por meio de tabela podem ser gerados de maneira conveniente por um controle no painel frontal. Como sugestão utilize 6 pontos para simulação em ambos os sistemas, utilizando intervalos de amostragem maiores ou iguais a 1, devido às limitações da VI *Simulate Arbitrary Signal*. Consulte o LabVIEW Help para aprofundar seus conhecimentos na VI e entender a função de saída da mesma e a correlação de parâmetros.

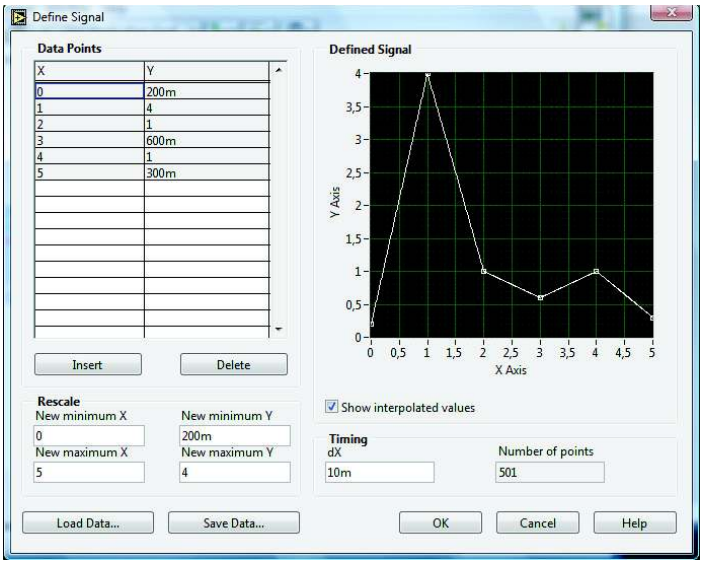


Figura 93 – Utilizando a VI "*Simulate Arbitrary Signal*".

A exploração da geração das formas de onda estende-se muito além dos exemplos citados, mas é possível, com este ponto de partida, compreender melhor a utilização deste recurso e expandir muito sua aplicação.

8.2 Medições em formas de onda

Da mesma forma que na simulação para geração de formas de onda, a medição de formas de onda no LabVIEW traz uma diversidade imensa de possibilidades para obtenção de parâmetros e processamento dos sinais obtidos por meio de sistemas de aquisição de sinais.

O ponto de partida mais comum é utilizar sinais simulados, como já abordados anteriormente, para se extrair informações que se esperam em um sistema de medição real. A simulação da medição é, portanto, o melhor caminho quando estamos desenvolvendo um sistema para processamento de sinais.

Aqui também iremos explorar VIs e Express VIs que possibilitam efetuar medições com sucesso. A utilização destas em processos de medição e controle dar-se-á a medida que formos evoluindo na utilização dos sistemas de geração e medição de sinais, e dos dispositivos que iremos utilizar como hardware em nossas aplicações.

Por meio do menu NEW, podemos abrir um primeiro exemplo modelo (*template*) de medição de um sinal, a VI *Generate, Analyze and Display*. Esta VI utiliza os recursos de duas Express VIs para geração e medição de um valor RMS do sinal simulado. Explorando um pouco mais os recursos da VI *Amplitude and Level Measurements* podemos visualizar um pouco dos parâmetros que podemos obter prontamente em nossas aplicações com LabVIEW. Perceba que mais de um parâmetro pode ser obtido simultaneamente, o que pode ser muito conveniente em uma aplicação real.

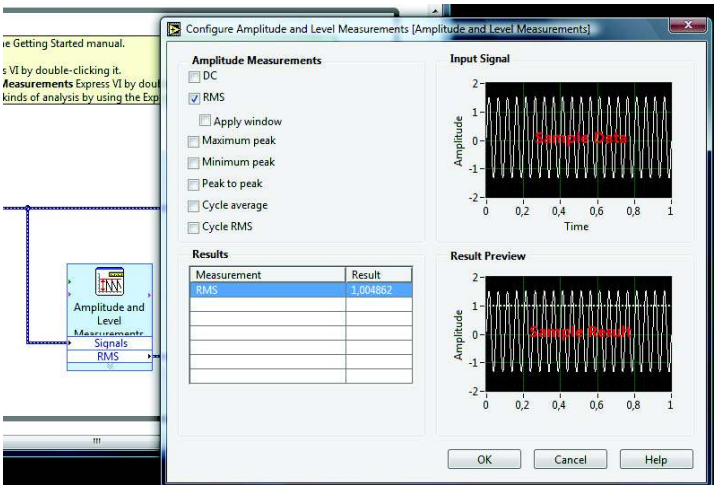


Figura 94 – Utilizando a VI "*Amplitude and Level Measurements*".

Esta VI faz parte do grupo de funções *Signal Analysis*. Neste grupo temos outra diversidade de VIs que efetuam medições e análise de sinais e que podem ser exploradas de forma conveniente.

Exercício 15: Medição em formas de onda

No exemplo do gerador PWM do exercício 1, tínhamos a variação de um sinal DC para compor a saída, mas se quiséssemos certificar o valor do *duty cycle* de saída poderíamos medi-lo. Implemente a medição do *duty cycle* e duração de pulso do exercício 1 utilizando as Express VIs de análise.

Podemos utilizar outras VIs que nos trarão as mesmas informações mas que não são do grupo das Express VIs. Na paleta *Waveform >> Analog Waveform >> Waveform Measurements*, podemos encontrar algumas delas. Se utilizarmos a VI *Pulse Measurements* podemos obter os mesmos resultados que os obtidos com a Express VI utilizada no exercício 3.

Da mesma forma podemos utilizar as VIs deste grupo para efetuar medições de muitos parâmetros de interesse. Em alguns casos, no entanto, teremos que condicionar o sinal para que possamos trabalhar os parâmetros medidos e isto requer a utilização de recursos também disponibilizados pelo LabVIEW. Na maioria das vezes as medições podem ser efetuadas com estes parâmetros.

Vamos agora configurar um sinal simulado senoidal, utilizando a VI *template* do começo desta seção como base. Vamos medir o valor médio desta senoide. Altere o valor do número de amostras e taxa de amostragem para 200 pra maior conveniência. Utilize VIs não Express para obter os mesmos resultados e comparar sua complexidade de implementação. Meça outros parâmetros e compare outras VIs, se necessário, alterando o padrão do sinal simulado.

Outras informações podem ser obtidas de fontes de dados e sinais, como parâmetros matemáticos estatísticos. Estes parâmetros podem ser obtidos pelas VIs existentes na paleta *Mathematics* ou por meio da Express VI *Statistics*. Procure comparar resultados destas VIs também e assim familiarizar-se com estes recursos. Você pode simular sinais padrão ou sinais arbitrários para avaliar o comportamento destas VIs.

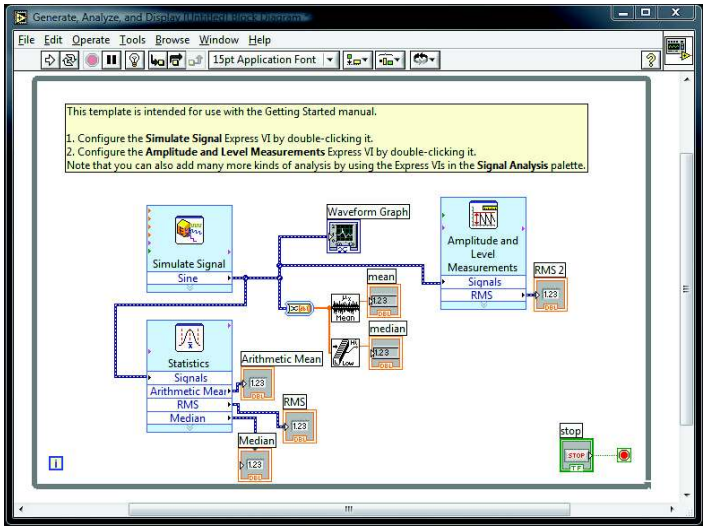


Figura 95 – Manipulando a VI "*Amplitude and Level Measurements*".

Assim como parâmetros matemáticos, uma infinidade de parâmetros podem ser obtidos com o LabVIEW, e muitos sinais podem ser condicionados como por exemplo por meio de filtros, clipping e triggers. A utilização de triggers é uma forma conveniente de trabalharmos níveis de referência de entrada quando desejamos obter parâmetros de porções do sinal medido. A diversidade destes recursos permitem ao LabVIEW contornar alguns problemas de medição encontrados na prática. Nestes casos, o uso de VIs que fornecem parâmetros diretamente de um sinal não é conveniente, mas os recursos disponibilizados pelo LabVIEW possibilitam alcançar os objetivos propostos.

Exercício 16: Medição em sinais não periódicos

Utilizando a Express VI para sinais arbitrários meça o período do sinal simulado, conforme abaixo, utilizando níveis de referência de medição de 25 e 75%. Perceba o comportamento de Express VIs e VIs de medição direta dos parâmetros e tente contornar o problema de forma efetiva utilizando os recursos que você já conhece. Para que a alteração seja sensível, ajuste a variação de tempo dt para 0,5 ms.

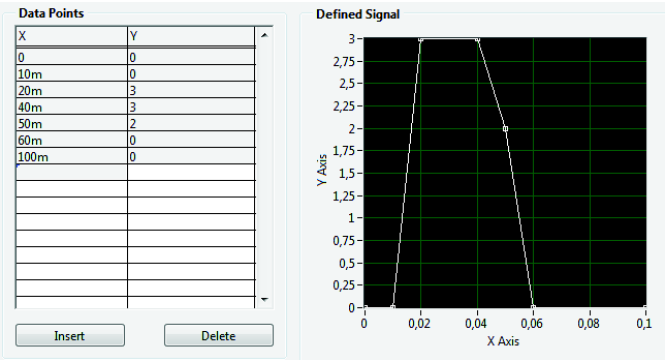


Figura 96 – Exercício "Medição em sinais não periódicos".

Outras manipulações de sinais podem ser efetuadas com o LabVIEW, mas a exploração de todos estes recursos não é viável em um curso. Com as informações fornecidas é possível explorar estes recursos, lembrando-se sempre da utilização do LabVIEW help para maior aprofundamento nas funções. Ao longo deste módulo voltaremos a explorar outros recursos de medição e processamento de sinais.

9 *Strings* e E/S de Arquivos

9.1 *Strings*

Uma *string* é um conjunto de caracteres ASCII que podem ou não ser exibidos. As *strings* apresentam plataformas independentes para informações e dados. Pode-se usar *string* para uma simples mensagem de texto, para transferência de dados para um instrumento para controle e aquisição de dados do mesmo neste formato para posterior conversão, envio de alertas por meio de *prompts*, etc. As *Strings* no painel frontal apresentam-se em forma de tabelas, caixas de entrada de texto e *Labels*.

Strings podem ser utilizados acessando o grupo *String & Path* na paleta de controle do painel frontal. *Strings* de controle e indicação estão disponíveis neste menu. Estes componentes podem ser redimensionados com o mouse para adequação a sua utilização, possuindo quatro tipos de exibição acessíveis com o botão direito do mouse:

- Exibição Normal;
- *'\' Codes Display*, que indica caracteres que não serão impressos num texto;
- *Password Display*, que oculta os caracteres por meio da máscara "*";
- *Hex Display*, que exibe o código ASCII de tosos os caracteres digitados.

Quando trabalhamos com transmissão de dados em ASCII por meio de terminais ou outros recursos equivalentes, o que normalmente fazemos em comunicações com instrumentos e equipamentos de bancada, a exibição de *'\' Codes* ou mesmo caracteres hexadecimais para visualizarmos códigos de envio, finalização de dados e ferramentas de verificação.

Como uma caixa de texto disponível na maioria das plataformas de programação Windows, o LabVIEW permite a exibição de barras de rolagem (*Scrollbar*) para minimizar o tamanho dos campos de texto sem que o usuário perca o acesso a todos os caracteres do campo.

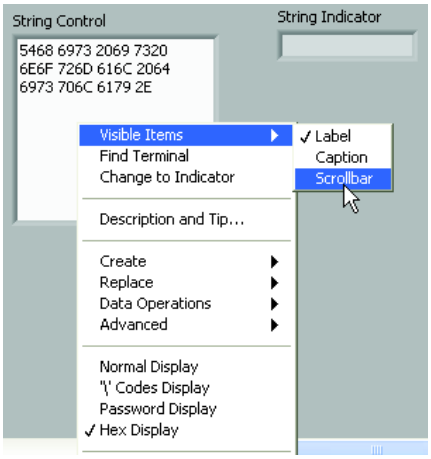


Figura 97 – Acessando os itens que podem ser visualizados em uma caixa de texto.

9.1.1 String Functions

Na paleta de funções, a sub-paleta *String* traz VIs para edição e manipulação de *Strings* que são amplamente utilizadas no trabalho em códigos com *strings* no diagrama de blocos.

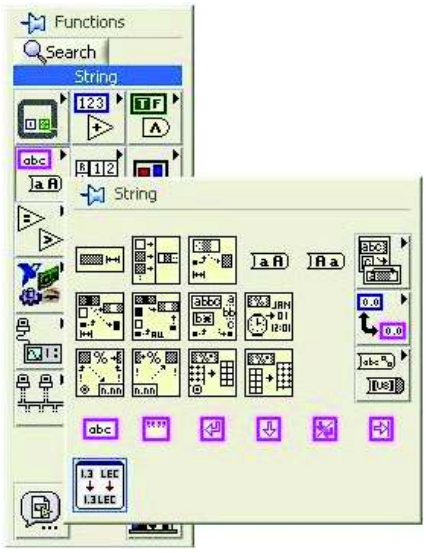
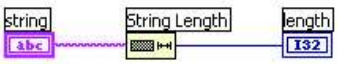


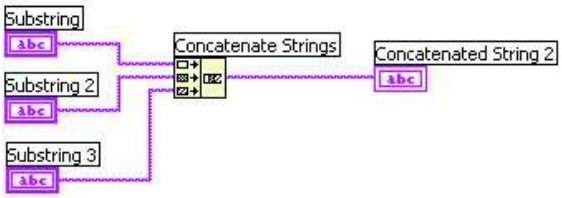
Figura 98 – Acesso a paleta *String* no diagrama de blocos.

Agora vamos aborda as VIs de manipulação de *Strings* mais usuais no LabVIEW:

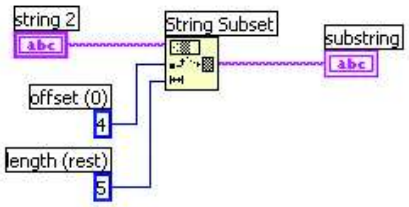
***String Lenght*:** esta função retorna o "comprimento" de uma *string*, ou seja, o número de caracteres em uma *string*, incluindo espaços. A entrada desta VI aceita uma *String* e sua saída é representada por um número inteiro:



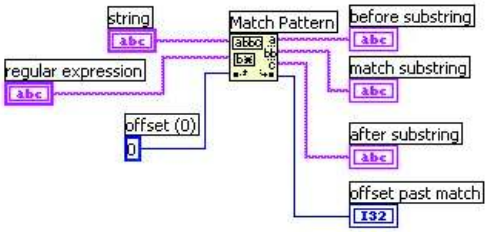
***Concatenate Strings*:** esta função concatena *strings* em um *Array* de *strings* final que irá efetuar uma função específica. É importante utilizar espaços e outros caracteres de separação de forma adequada para formação dos *Arrays*. Esta função é muito utilizada quando queremos compor controles ASCII, concatenando inclusive caracteres CR e LF, disponíveis no LabVIEW.



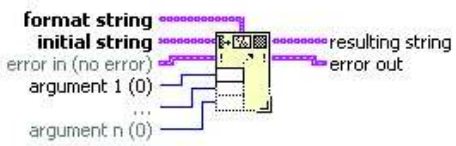
***String Subset*:** esta função retorna uma *substring* que possui um offset, ou início, e um número de caracteres específico que irá determinar o tamanho da *String*, lembrando que a *String* possui índice inicial 0 como qualquer vetor no LabVIEW.



***Match Pattern*:** esta função efetua uma "busca" dentro de uma *String* com início a partir de um determinado offset. Quando a função encontra uma expressão, ou "*match*", dentro de uma *string*, esta retorna uma divisão da *String* em três *substrings*: a que aparece antes do "*match*", o "*match*" procurado, e o que está após o mesmo. Além disso, o número de caracteres após o "*match*" é indicado. Quando nenhuma expressão é encontrada, esta função retorna -1 para o número de caracteres e o resto dos retornos é vazio.



***Format Into String*:** esta função converte formatos numéricos em *String*. Esta função retorna um formato específico para uma *String* de saída, concatenando com uma *String* inicial de entrada e um argumento numérico a ser convertido.



As propriedades desta função podem ser editadas, utilizando-se o botão direito do mouse, e assim configurando de forma fácil a saída desta função.

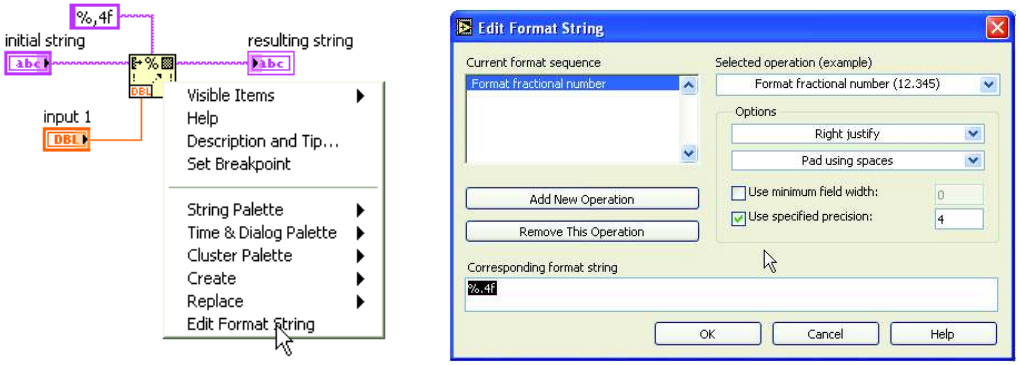
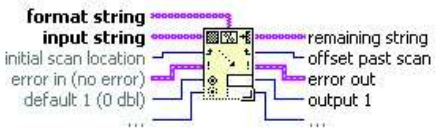


Figura 99 – Edição da VI *Format String* no diagrama de blocos.

***Scan From String*:** esta função converte formatos *String* com caracteres numéricos válidos em números em vários formatos, inclusive Booleanos. O número convertido é retornado na saída Output 1.



As propriedades desta função podem ser editadas, utilizando-se o botão direito do mouse, e assim configurando de forma fácil a saída desta função.

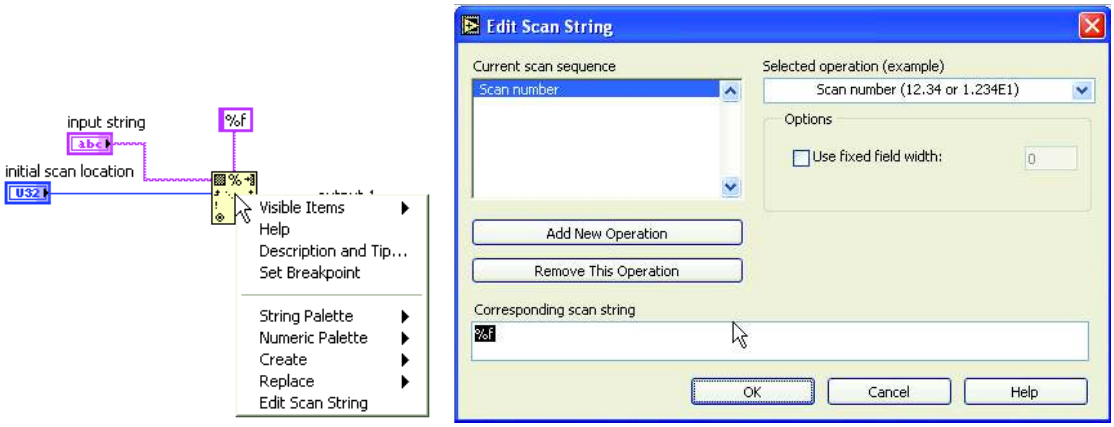


Figura 100 – Acesso a paleta *String* no diagrama de blocos.

É possível ver que nestas duas funções de conversão de dados, podemos editar os formatos manualmente quando já temos experiência de manipulação das mesmas.

Exercício 17: Expressão de resultados para saída String

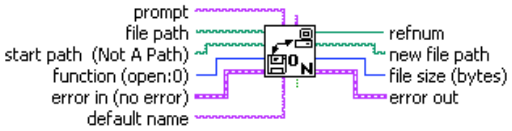
Construa uma VI que indica um valor de medição qualquer como “O resultado é X.XXX Volts”, por exemplo, utilizando a formatação de três casas decimais.

9.2 VIs e Funções para File I/O

Na sub-paleta de funções File I/O temos as VIs para manipulação de arquivos, como abertura, escrita, leitura, fechamento, etc.

A operação com arquivos no LabVIEW, assim como na maioria das linguagens de programação baseadas em linhas de texto, consiste em três etapas fundamentais:

- Abrir, criar ou substituir: esta ação é efetuada pela VI *Open/Create/Replace File*. Caso a entrada de localização do arquivo “File Path” não seja preenchida, uma janela de *prompt* para busca do arquivo será aberta ao usuário da VI. Após a utilização desta VI, a criação de um número de referência é estabelecida para o trabalho com este arquivo.



- Escrita e/ou leitura de arquivos: A segunda etapa de escrita e/ou leitura no arquivo utiliza o número de referência para operação.
- Fechamento: esta operação finaliza o trabalho com o arquivo, permitindo que outras operações com o mesmo possam ser efetuadas posteriormente.

A interconexão do cluster de erros de saída é muito importante durante a manipulação deste tipo de VI. O tratamento de erros e seus resultados são enviados ao final do processamento de cada uma destas VIs. O cluster de erros é então passado de VI em VI com os seguintes indicadores principais:

- *Status*: retorna TRUE na ocorrência de erros no processamento;
- *Code*: um indicador *Integer* de 32 bits com a identificação de erro;
- *Source*: Indicador de onde ocorreu o erro na VI.

O número de referência e o cluster de erros são passados em sequência pelas VIs, e como nenhuma delas executa uma operação sem que a entrada “*refnum*” esteja disponível, ocorre execução sequenciada sem a necessidade de utilização de estruturas sequenciais ou equivalentes.

Explorando melhor as VIs do segundo estágio abordado acima podemos detalhar mais um pouco a VI de escrita *Write File*. Esta VI recebe um *refnum* válido para iniciar sua operação, bem como o cluster de erros sem qualquer indicação TRUE anterior. Ela inicia sua operação escrevendo a

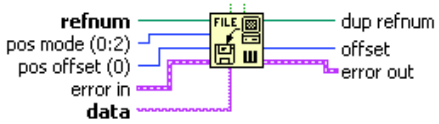
informação, "*data*", numa posição do arquivo estabelecida por "*pos mode*" e "*pos offset*" para alguns formatos de arquivos e sempre ao final do arquivo para *Dataloggers*.

As funções de E/S de arquivos disponibilizadas pelo LabVIEW são muito flexíveis para utilizar as ferramentas no manuseio destes arquivos. Para ler e escrever dados, as funções do LabVIEW movem e renomeiam arquivos e diretórios, criam arquivos legíveis em texto ASCII e dados escritos em formato binário de maneira rápida e compacta.

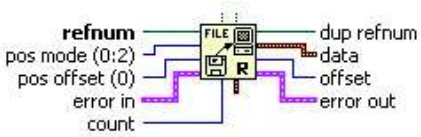
Podem-se armazenar dados de arquivos em três diferentes formatos:

- ASCII *Byte Stream* (*default*): Você deverá armazenar dados no formato ASCII quando você necessitar acessar de um outro programa.
- Arquivos de Log: Estes arquivos estão em formato binário que somente o LabVIEW pode acessar. Arquivos Log são similares a arquivos de base de dados, pois podem armazenar muitos tipos diferentes de dados dentro de um registro de um arquivo.
- *Binary Byte Stream*: Estes arquivos são mais compactos e mais rápidos para se armazenar. Você deve converter os dados para um formato de *string* binário e deverá conhecer exatamente que tipos de dados você está salvando.

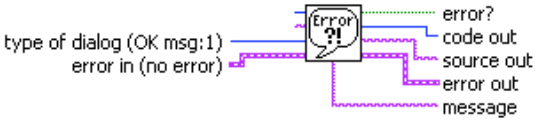
Maiores detalhes desta operação estão disponíveis no LabVIEW Help, mas sua aplicação normalmente utiliza os valores default para operação com texto e *spreadsheet*.



A VI de leitura *Read File* utiliza praticamente as mesmas configurações da Write File e retorna a informação "data" com um controle da quantidade de dados a serem lidos pela entrada "*count*".



Um quarto elemento que pode e deve ser utilizado para auxiliar no tratamento dos cluster de erros é a VI *Simple Error Handler*, que recebe o cluster de erros e processa os resultados, exibindo os detalhes numa caixa de diálogo em caso de detecção de erros.



Exercício 18: Escrevendo resultados de uma *String* para Arquivo

Construa uma VI que indica escreve o valor de medição do Exercício 17 em um arquivo de saída. Utilize um caminho fixo e um caminho por *Prompt* para explorar estas duas formas de acesso ao arquivo e suas vantagens.

Exercício 19: Lendo resultados de um Arquivo

Construa uma VI que lê o valor armazenado no Exercício 18. Utilize um caminho fixo e um caminho por *Prompt* para explorar estas duas formas de acesso ao arquivo e suas vantagens.

9.2.1 Formatando Strings para um arquivo Spreadsheet

Para escrever dados em um arquivo *Spreadsheet* é necessário formatar as *strings* para atenderem a este formato, *spreadsheet string*, que consiste em uma *string* com delimitadores como tabulações ou espaços.

É possível utilizar a VI *Format Into File*, localizada na sub-paleta de funções File I/O, para formatar variáveis numéricas, *strings*, paths e booleanos em texto e então escrevê-lo em um arquivo texto.

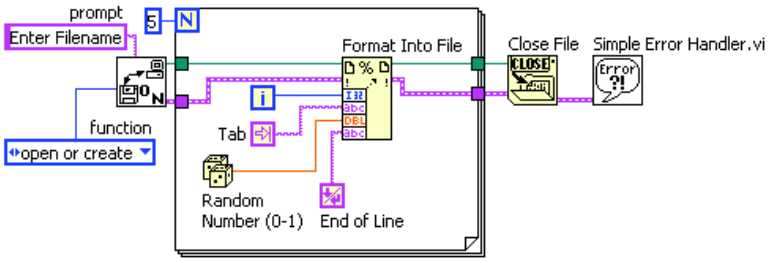


Figura 101 – Exemplo de utilização da VI *Format Into File* no diagrama de blocos.

Neste simples exemplo podemos verificar a utilização de constantes *strings*, como *Tab* e *End of Line*, utilizados para estabelecer os separadores e preparar a variável para um arquivo *Spreadsheet*, separando colunas e linhas respectivamente.

Esta VI é utilizada para determinar a ordem em que os dados aparecem em um spreadsheet. Para inserção de dados em um arquivo, utilizamos a VI Format Into String juntamente com a VI Write File.

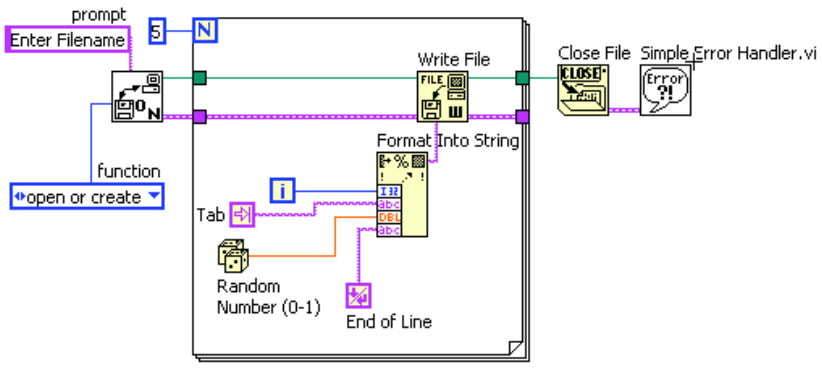


Figura 102 – Exemplo de utilização da VI *Format Into String* no diagrama de blocos.

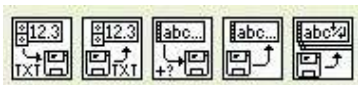
É importante lembrar que o LabVIEW efetua como saída padrão numa estrutura For Loop a auto-indexação de arrays de dados, o que deve ser desabilitado para que a saída possa ser reconhecida pela próxima VI, a de fechamento de arquivo, que não aceitará uma entrada do tipo Array.

Exercício 20: Logger de temperatura

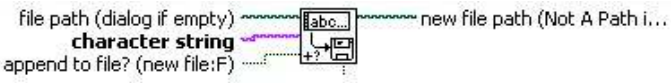
Baseado na VI de conversão de temperatura, crie uma VI que armazena os dados de temperatura que ultrapassam um determinado limite para posterior leitura em um visualizador de texto.

9.2.2 VIs de alto nível para FILE I/O

O LabVIEW possui um grupo de VIs consideradas de alto nível, disponíveis na sub-paleta de funções FILE I/O. Estas VIs podem ser utilizadas para executar operações de I/O como escrita e leitura em arquivos, com vários tipos de dados.

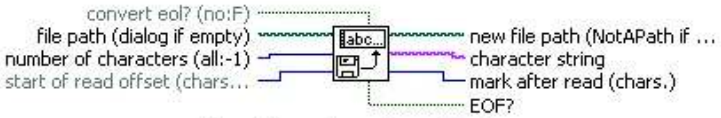


A VI *Write Characters to File* é utilizada para escrever caracteres *string*, ou mesmo adicioná-los a um arquivo. Esta VI abre ou cria o arquivo selecionado, efetua as operações necessárias e posteriormente fecha o mesmo.



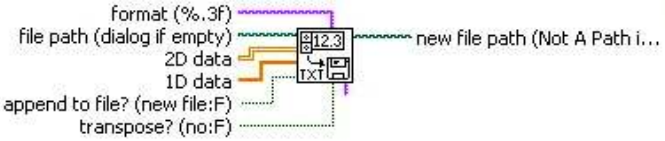
Write Characters To File.vi

A VI *Read Characters from File* efetua a leitura de um número de caracteres, ou todos eles, iniciando a partir de um offset. Esta VI abre o arquivo, efetua a leitura e posteriormente fecha o mesmo.



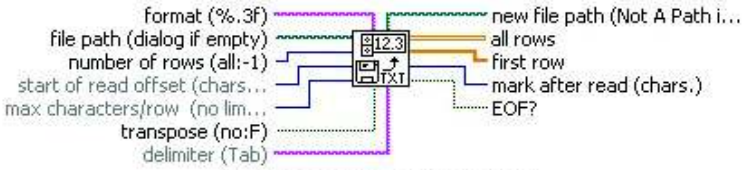
Read Characters From File.vi

Outra VI para escrita em arquivos é a *Write to Spreadsheet File* que converte *arrays* 1D e 2D de números (*single-precision*) para uma *string* de texto e a escreve em um novo arquivo ou num arquivo existente. Esta VI abre ou cria o arquivo selecionado, efetua as operações necessárias e posteriormente fecha o mesmo, sendo disponibilizado para qualquer outro leitor de *Spreadsheet*.



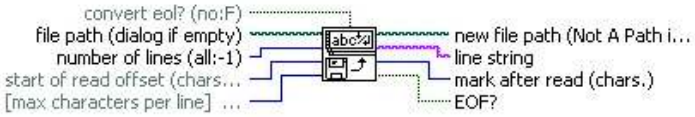
Write To Spreadsheet File.vi

Da mesma forma temos a VI de leitura de arquivos *Spreadsheet, Read to Spreadsheet File*, que retorna um Array 2D de numéricos (*single-precision*). Esta VI pode ser utilizada para ler arquivos *Spreadsheet* em formato de texto criados ou modificados por qualquer editor. Esta VI abre o arquivo, efetua a leitura e posteriormente fecha o mesmo.



Read From Spreadsheet File.vi

Temos ainda a VI *Read Lines from File* para leitura de um número específicos de linhas de um arquivo de texto ou dados binários iniciando do controle de entrada "*start of read offset*". Esta VI abre o arquivo, efetua a leitura e posteriormente fecha o mesmo.



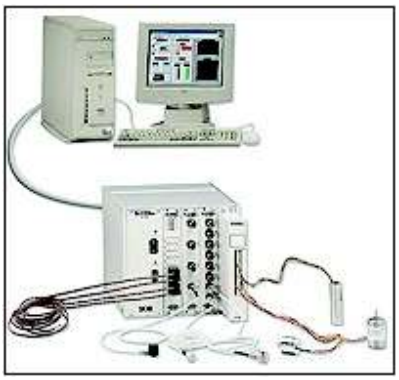
Read Lines From File.vi

É possível utilizar tabelas no painel frontal para exibição de dados num *Array* 2D de *Strings* provenientes da leitura de um arquivo.

O importante é explorarmos os recursos destas VIs para manipulação de dados provenientes de arquivos e aplicarmos em nossos programas em desenvolvimento ou programas a serem desenvolvidos.

10 Utilizando o DAQ Assintant e o Measurement and Automation

O LabVIEW oferece grupo de VIs que possibilitam configurar, enviar e coletar dados através de um sistema de aquisição (DAQ). Esta geração e aquisição de dados por meio destes dispositivos ocorrerá, normalmente, em tempo real, e sua aplicação possibilita grande versatilidade.



Primeiramente, anteriormente a qualquer aplicação, os sistemas de aquisição requerem o uso de transdutores ou sistemas de condicionamento de sinais (por exemplo: aplicação para medição e geração de alta tensão) que fazem o primeiro trabalho de preparo para o sistema de aquisição. Os sistemas de aquisição e geração podem então trabalhar com sinais analógicos e digitais em tempo real, efetivando a aplicação pretendida. É sempre importante verificar requisitos de cabos, aterramentos, pinagem e limites do sistema de condicionamento de sinais e do DAQ.

A utilização do LabVIEW em qualquer aplicação deste nível facilita e reduz drasticamente tempos de desenvolvimento e o tempo de configuração de hardware, que normalmente é muito elevado.

A National Instruments oferece uma linha completa de DAQs mas o LabVIEW não esta limitado a estes. NI-DAQs possuem dois drivers de aquisição de dados, os NI-DAQs tradicionais e os NI-DAQmx, cada um com sua API (*application programming interface*), configuração de hardware e software. Os tradicionais são utilizados em códigos editados em versões anteriores a 6.1 e em dispositivos sem suporte para DAQmx. Estes drivers possuem várias séries de VIs próprias para sua utilização para geração e aquisição de sinais, especificamente.

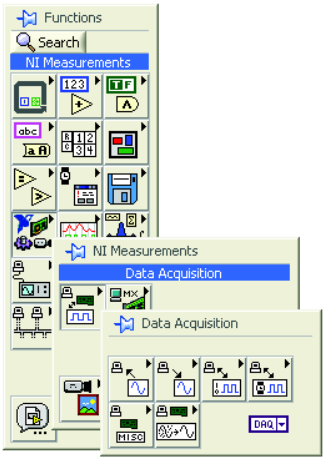


Figura 103 – Acesso a paleta *Data Acquisition* no diagrama de blocos.

Os DAQmx são os drivers mais atuais da National e possuem como maior vantagem o DAQ *Assistant* que possibilita a configuração dos dispositivos de maneira mais conveniente e funcional. Ao contrário dos tradicionais, os drivers DAQmx utilizam um único set de VIs para todas as funções.

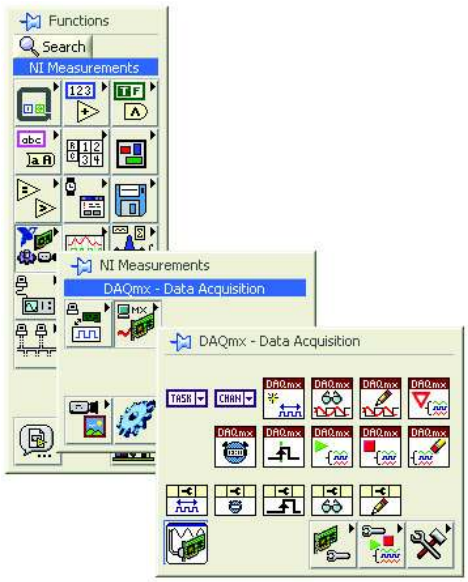


Figura 104 – Acesso a paleta DAQmx *Data Acquisition* no diagrama de blocos.

A conveniência do gerenciamento destes DAQs é outro ponto forte do LabVIEW e a ferramenta *Measurement & Automation Explorer*, utilizada de forma conveniente, pode auxiliar este trabalho. O MAX reconhece automaticamente qualquer dispositivo da NI mas outros também podem ser configurados e reconhecidos por este. Para configuração adequada de outros dispositivos no MAX, uma consulta no LabVIEW *Zone* é importante.

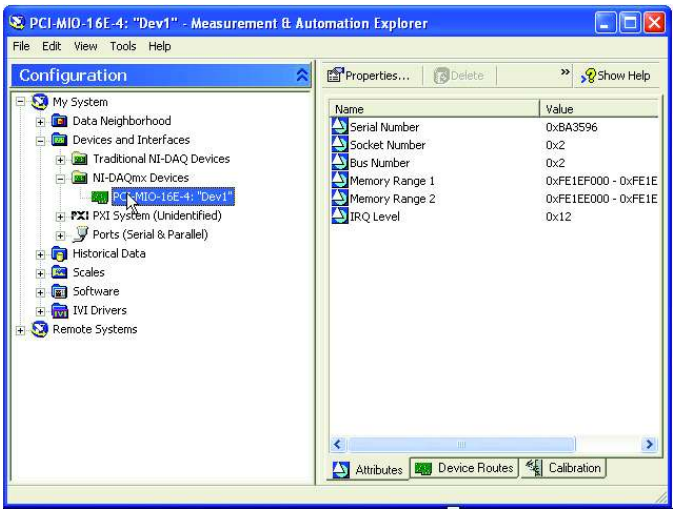


Figura 105 – MAX – *Measurement & Automation Explorer*.

Por meio do MAX você pode detectar e testar entradas e saídas (digitais e analógicas), e contadores do DAQ.

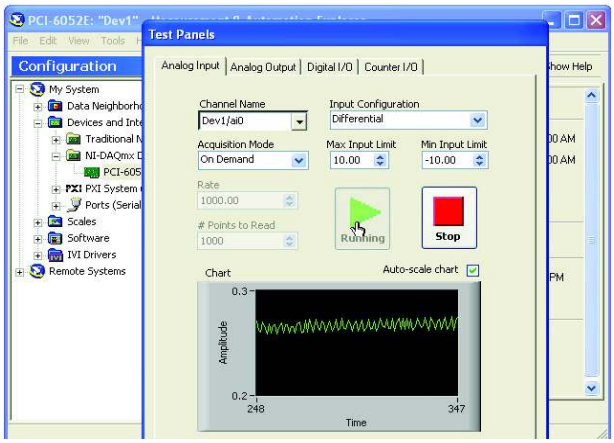


Figura 106 – Exemplo de Painel de Teste no *Measurement & Automation Explorer*.

No ambiente LabVIEW, o auxílio para desenvolvimento de códigos que utilizam DAQs é facilitado pelo DAQ *Assistant*. Esta Express VI facilita a configuração de hardware (NI ou outros drivers reconhecidos).

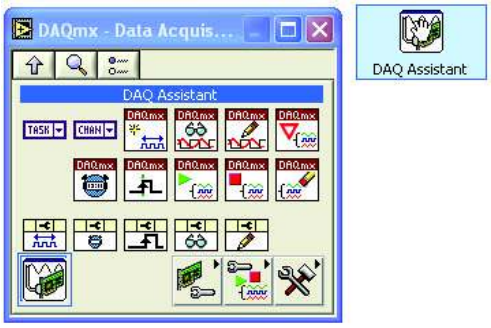


Figura 107 – Acesso ao DAQ *Assistant* no diagram de blocos.

Procure explorar o DAQ *Assistant* e configurar seu hardware de forma adequada a suas aplicações de entrada e saída de sinais e contadores. Cada recurso pode ser configurado e testado simultaneamente quando conveniente.

11 Filtros

O projeto de filtros analógicos é uma das mais importantes áreas em desenvolvimento eletrônico. Apesar de a literatura atual trazer uma série de simplicidades no projeto e desenvolvimento de filtros e ainda exemplificar filtros funcionais e existentes no mercado, o trabalho com esta tecnologia ainda é de propriedade de especialistas devido à complexidade matemática e necessidade do conhecimento dos processos envolvidos nos sistemas e que normalmente afetam o desempenho dos filtros.

Sistemas de aquisição e processamento de sinais por meios digitais trouxeram a possibilidade de se trabalhar com filtros de forma eficaz com a necessária flexibilidade e versatilidade. Estes filtros trazem como vantagem a capacidade de ser programáveis, ser previsíveis em sua aplicação (simulação do comportamento e resposta), não são susceptíveis a variações ambientais e não requerem utilização de componentes especiais para sua implementação alem de apresentar melhor relação custo-benefício.

Outras vantagens surgem durante o desenvolvimento e implementação de filtros digitais, e com o LabVIEW, a implementação destes torna-se dinâmica e previsível. Com o LabVIEW você pode utilizar filtros e controlar parâmetros como ordem, freqüências de corte, ripple, etc.

É importante sempre nos lembrarmos do teorema de amostragem (Nyquist) onde a freqüência de amostragem deve ser pelo menos duas vezes a mais alta freqüência presente no sinal que se deseja medir. Alguns filtros serão apresentados aqui, mas novamente, a gama de filtros disponíveis no LabVIEW pode ser explorada mediante a necessidade de aplicação. Os filtros baseados no princípio do teorema de amostragem utilizados no LabVIEW são:

- *Smoothing wondows* (filtros de média);
- *Infinite Impulse Response* / *Recursive filters* (filtros com resposta em pulso infinita);
- *Finite Impulse Response* / *Nonrecursive filters* (filtros com resposta em pulso finita);
- Não-lineares.

A primeira e mais simples forma de utilizar um filtro no LabVIEW é a *Express* VI *Filter*. Ao inserir esta Express VI teremos alguns detalhes:

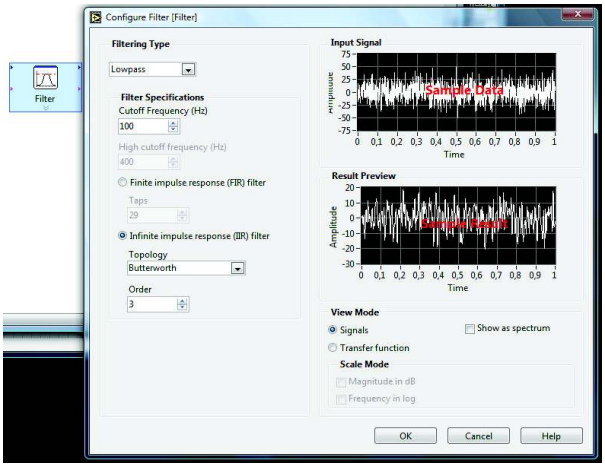


Figura 108 – Configuração da *Express* VI *Filter*.

Esta Express VI traz algumas possibilidades oferecidas pelo LabVIEW que podem ser acessíveis individualmente pela paleta *Analyze>>Signal Processing>>Filters*. Nesta paleta temos funções avançadas que podem ser utilizadas dependendo da aplicação. A seguir vamos explorar de forma simplificada a teoria de alguns filtros e mostrar como encontrar e utilizar VIs do LabVIEW que executam as funções de filtro digital.

11.1.1 Smoothing Windows

Na pratica de aquisição de sinais por meio de sistemas conversores (A/D), somente uma porção finita do sinal é adquirido de uma amostra que normalmente é contínua, e este fato leva a distorções espectrais em relação ao sinal original. Uma forma simples de aperfeiçoar a informação obtida é efetuar uma analise de Fourier ou espectral dos dados obtidos por meio de um filtro do tipo *Smoothing Windows* que pode minimizar transientes de formas de onda truncadas pelo sistema de aquisição e então reduzir fugas, ou desvios espectrais.

Os filtros *Smoothing*, normalmente traduzidos como filtros de média ou de interpolação, atuam como filtros passa-baixa em banda estreita.

As VIs de LabVIEW que efetuam esta ação de filtro *Smoothing Windows* utilizam um código DFT (*Discrete Fourier Transform*) para análise e representação no domínio da frequência. Outro ponto importante é que o código de DFT utilizado pelo LabVIEW nas funções de *Smoothing Windows* são do tipo DFT-*even*, sem componentes imaginárias (senos iguais a zero). A paleta *Analysis>>Windows* traz uma série de filtros que pode ser explorada no LabVIEW. Dois tipos estão na *Express* VI *Filter*, os filtros com resposta Triangular (*Barltlett smoothing window*) e Quadrada.

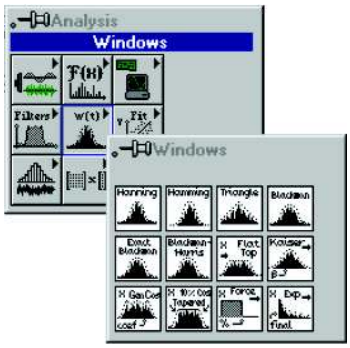


Figura 109 – Acesso aos filtros da paleta *Windows*.

O LabVIEW Help traz detalhes e funções de transferência destes filtros, mas os dados de entrada destas VIs possuem limitações que podem ser visualizadas em cada uma delas individualmente.

11.1.2 Filtros IIR (Infinite Impulse Response)

Estes filtros que são parte da configuração default da *Express* VI *Filter* são amplamente utilizados em projetos de sistemas de controle. São filtros digitais com resposta ao impulso que podem ser teoricamente de duração infinita. A equação diferencial que caracteriza estes filtros é descrita de forma geral como:

$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{N_b-1} b_j x_{i-j} - \sum_{k=1}^{N_a-1} a_k y_{i-k}\right)$$

Onde N_b é o número de coeficientes diretos (*forward* - b_j) e N_a o número de coeficientes reversos (reverse - a_k). Na maioria dos filtros digitais IIR utilizados em softwares de eletrônica e em todos os filtros IIR do LabVIEW, o coeficiente a_0 é 1. A amostra na saída no índice i presente é a soma das entradas escalonadas atuais e anteriores (x_i e x_{i-j} quando ≠ 0) e saídas escalonadas anteriores (y_{i-k}). Por isso os filtros IIR são conhecidos como filtros recursivos ou ARMA (*AutoRegressive Moving Average filter*).

A resposta de um filtro genérico IIR a um impulso é então chamada de resposta recursiva ao impulso. A resposta ao impulso do filtro descrito na equação acima é de fato de comprimento infinito para coeficientes diferentes de zero. Na prática, no entanto, a resposta ao impulso cai próximo de zero num número finito de elementos. Estas informações estão detalhadas no Help do LabVIEW, para os filtros do grupo *Poit-by-point*, onde maior aprofundamento pode ser obtido.

O importante aqui é conhecermos algumas propriedades dos filtros IIR no LabVIEW:

- Índices negativos resultantes da equação apresentada acima são assumidos como zero na primeira vez que a VI é chamada;
- Devido ao estado inicial zero (índices negativos), um transiente proporcional a ordem do filtro ocorre antes que o filtro alcance seu estado estacionário. Resumindo, a duração da

resposta ao transiente, ou retardo (*delay*), para filtros passa-alta ou passa-baixa é proporcional a ordem do filtro;

- Para filtros passa-banda e filtros de corte (*bandstop*), o *delay* é de duas vezes a ordem do filtro.

A vantagem dos filtros digitais IIR sobre os de resposta finita (FIR) é que os IIR normalmente requerem poucos coeficientes para executar as mesmas operações, assim, estes filtros trabalham com maior eficiência e velocidade.

A desvantagem dos filtros IIR é que a resposta em fase não é linear, mas para aplicações onde não se deseja obter informações da fase, ou onde não é necessário resposta linear desta, estes filtros são simples e eficientes.

11.1.3 Filtros FIR (Finite Impulse Response)

Os filtros de resposta em pulso finita são filtros digitais conhecidos como não-recursivos, filtros de convolução ou média-móvel (MA *moving-average*), pois a expressão de saída do filtro é expressa como uma convolução finita:

$$y_i = \sum_{k=0}^{n-1} h_k x_{i-k}$$

Onde *x* representa uma sequência de entrada a ser filtrada, *y* representa a sequência de saída filtrada e *h* os coeficientes. Aqui listamos algumas das características mais importantes dos filtros FIR, que também podem ser encontradas no Help do LabVIEW:

- Eles podem alcançar a linearidade de fase;
- São sempre estáveis;
- A função do filtro pode ser executada utilizando a convolução e seu retardo pode ser obtido por:

$$\text{delay} = \frac{n-1}{2},$$

Onde *n* é o número de coeficientes do filtro.

11.1.4 Filtros não-lineares

Os filtros *smoothing*, IIR e FIR são filtros lineares por satisfazerem condições de proporcionalidade e superposição, o que não é satisfeito por um filtro não-linear. Algumas características como a combinação de comportamento passa-baixa e filtro de alta frequência são vantagens apresentadas por filtros não lineares. Este é o caso do filtro *Median*, apresentado pela VI *Median*

Filter, localizado também na paleta *Analysis>>Filters*. Um exemplo de resposta deste filtro é ilustrado na figura abaixo:

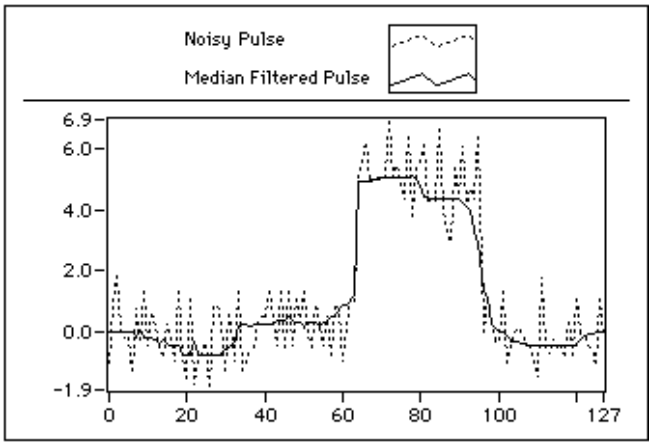


Figura 110 – Exemplo da VI *Median Filter*.

11.1.5 Explorando alguns filtros

Voltando a Express VI podemos explorar seus recursos observando as funções de transferência para cada topologia e ordem dos filtros *Smoothing*, IIR e FIR, sendo interessante observar a magnitude e fase, com frequência em escala log em alguns casos, para avaliarmos o comportamento destes filtros.

Vamos montar um exemplo de pulso gerado (simulado) e observar o comportamento de resposta do filtro (Express VI) para explorarmos melhor algumas destas propriedades.

Utilizando um sinal de forma de onda quadrada vamos primeiramente avaliar o atraso na resposta do filtro em suas diferentes topologias. Observe o *delay* do transiente do filtro para as topologias IIR e FIR, variando as frequências de corte nos modos passa-baixa e corte (*bandstop*) e a ordem.

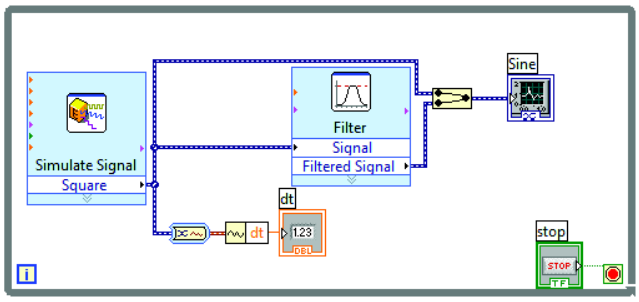


Figura 111 – Exemplo de utilização da *Express* VI *Filter*.

Varie as frequências do sinal de entrada observando a atuação do filtro e seu desempenho e resposta conforme a ordem dos filtros e o número de coeficientes finitos.

No LabVIEW help um detalhamento dos filtros IIR e FIR, assim como dos filtros de média e não-lineares é apresentado para quem deseja aprofundar seus conhecimentos ou necessita de maiores detalhes para sua aplicação. A paleta *Analysis>>Filters* também traz uma série de filtros:

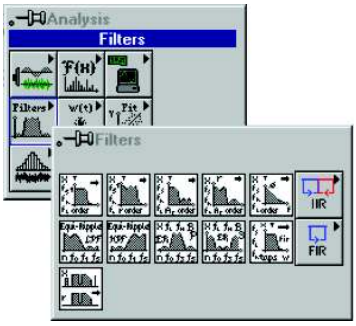


Figura 112 – Acesso a paleta *Filters*.

Acoplando um filtro *Butterworth* ao sistema de avaliação com Express VI *filter* podemos obter maiores variáveis, como por exemplo, a frequência de amostragem, que influencia diretamente a resposta de nosso filtro.

11.2 Funções estatísticas

Como já apresentado anteriormente podemos trabalhar com várias funções estatísticas dentro de um código em LabVIEW e as mesmas são obtidas de forma prática e com aplicação direta, exigindo poucas ligações e configurações para que possamos alcançar nossos resultados. Podemos trabalhar com Express VIs e com VIs independentes que trazem informações da mesma forma quanto a dados e sinais que estamos manipulando.

A Express VI *Statistics* traz uma primeira visão geral de dados que podemos obter com as VIs estatísticas. Podemos além dos cálculos estatísticos mais comuns obter valores de extremidades e características de amostragem dos sinais, de forma simultânea, em uma mesma VI.

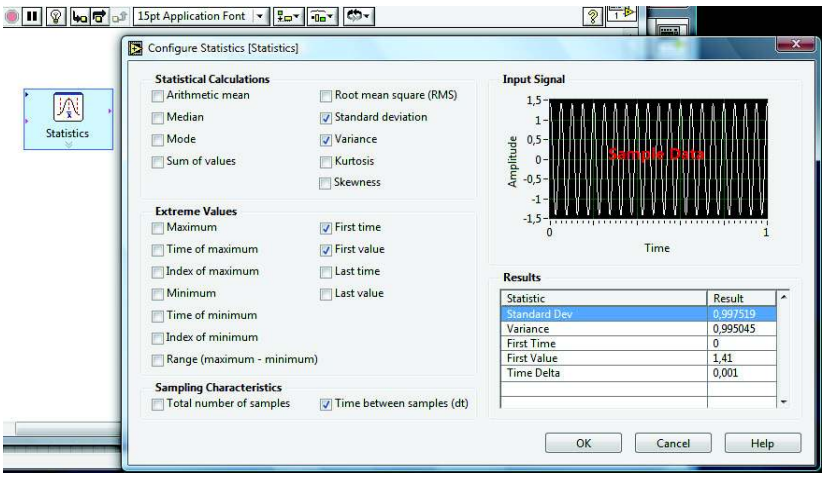


Figura 113 – Configuração da VI *Statistics*.

Podemos também criar histogramas utilizando outra Express VI,a *Create Histogram*.

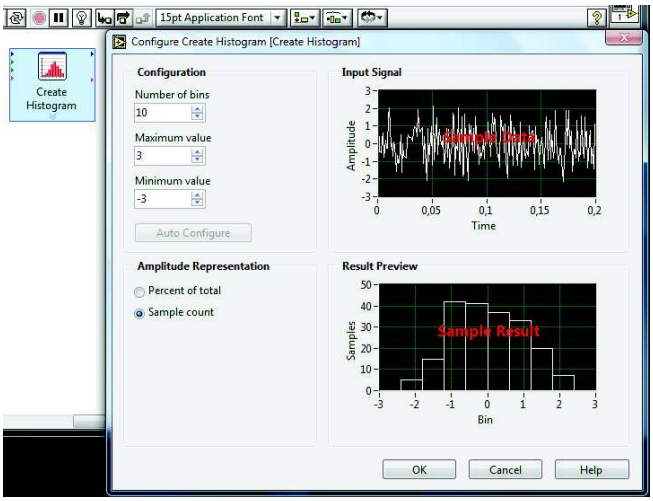


Figura 114 – Configuração da VI *Create Histogram*.

Mesmo com estas facilidades da utilização destas Express VIs, podemos ainda trabalhar outras VIs que podem trazer informações relevantes. Estas VIs encontram-se na paleta de funções *Probability and Statistics*. Dentro deste grupo temos o sub-grupo *Probability*, de onde podemos obter informações para diferentes distribuições de probabilidade.

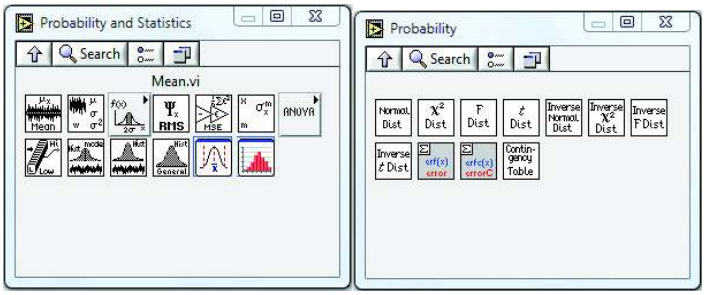


Figura 115 – Acesso a VI *Probability and Statistics*.

Vamos efetuar um exercício para utilização de funções estatísticas de forma simplificada, mas para que possamos ganhar liberdade na utilização destas VIs. Primeiramente vamos supor a situação em que um DAQ transfere dados de uma célula de carga para avaliação de força de compressão de um material X. Estes dados estão expressos no CHART abaixo e estão no painel frontal da VI *Estat*:

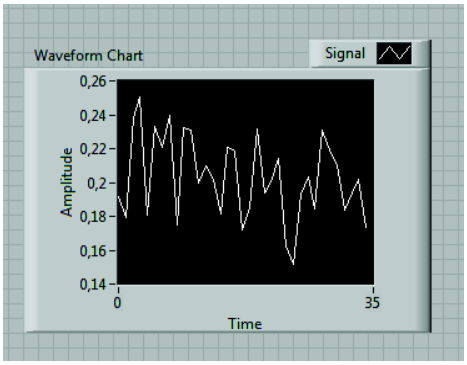


Figura 116 – Exibição de dados na VI *Estat*.

O arquivo na verdade apresenta um sinal arbitrário gerado para finalidade do exercício. Este sinal pode ser elaborado de forma aleatória para finalidade do mesmo, não devendo necessariamente ser reproduzido.

Vamos colocar estes dados em uma tabela no LabVIEW e obter algumas informações sobre os mesmos: Média, desvio padrão e histograma, agrupando os mesmos em 5 classes.

Agora vamos obter algumas probabilidades de ocorrências, considerando estes dados pertencentes a uma distribuição aproximadamente normal.

Exercício 21: Probabilidade (distribuição Normal)

Encontre a probabilidade de ocorrência de força maior que 0,2 N, e que forças estariam num intervalo de confiança de 90%. Lembrando que o fator z (x no LabVIEW) pode ser obtido pela equação (distribuição normal com N>30):

$$z = \frac{x - \mu}{\sigma}$$

Exercício 22: Covariância e correlação

Vamos agora observar uma medição da massa e do volume de algumas amostras e obter destas a correlação massa - volume.

Massa (g)	0,1918	0,1795	0,2382	0,2505	0,1810	0,2333	0,2206
Volume (ml)	0,105	0,095	0,114	0,120	0,098	0,113	0,102

Sendo a covariância:

$$cov(x,y) = \frac{\Sigma(xi - \bar{x}) \cdot (yi - \bar{y})}{N - 1}$$

E a correlação:

$$\rho = \frac{cov(x,y)}{\sigma x \cdot \sigma y}$$

Com estes exemplos pudemos explorar um pouco das funções estatísticas no LabVIEW. Obviamente a aplicação pode ser muito mais ampla e com esta VIs podemos obter dados qu podem delinear tomadas de decisões em controles de processos, já que podemos tornar todas estas aplicações dinâmicas em nossos códigos.

11.3 Ajustes de funções

Com as informações obtidas até o momento é possível avaliar a profundidade da aplicação do LabVIEW como ferramenta de aquisição e controle de processos. Muitas vezes é necessário inferir sobre o comportamento de determinados eventos para que possamos atuar no controle de forma adequada, e para isso ajustes de funções são imprescindíveis.

Um dos métodos mais comuns para isso é o Método dos Mínimos Quadrados (MMQ), que em seu modelo matricial pode facilitar muito a vida de uma rotina computacional ou código em LabVIEW. Obviamente o LabVIEW traz VIs que facilitam o emprego deste tipo de método e VIs que podem por si só efetuar o ajuste de um sinal de entrada para que possamos trabalhá-lo posteriormente. É importante, no entanto, lembrar que estes ajustes são apresentados sem erros dos parâmetros para Express VIs e algumas VIs de uso simplificado.

Podemos obter estes erros, ou variâncias, por meio de VIs mais complexas, com a necessidade de algumas manipulações iniciais. Outro fator importante são as matrizes de covariância, se desejamos propagar o erro destas variáveis para nossos resultados ou para futuras interpolações, que também não são apresentadas por todas VIs de ajuste de funções. Todas estas informações podem ser obtidas com o LabVIEW, mas exigem manipulação de outras VIs que acabam trazendo maior complexidade ao problema, mas apresentam ótimos resultados.

O LabVIEW apresenta métodos de ajuste lineares e não lineares, utilizando algoritmos de convergência que em alguns casos podem ser selecionáveis. Cada VI e seus algoritmos pode ser explorada e exigem estudos particulares que não são exatamente nosso objetivo aqui, mas com a introdução ao problema podemos expandir nossos conhecimentos e explora-los futuramente.

Vamos efetuar agora um exemplo de ajuste linear utilizando uma Express VI e os seguintes parâmetros de entrada.

X	-4	-3	-2	-1	0	1	2	3	4
Y	-1,1	-2,2	-0,4	0,9	3,1	3,5	3,1	4,7	6,8

Utilizando uma Express VI vamos ajustar uma reta (ajuste linear) montando dois vetores com os parâmetros X e Y, e exibindo um gráfico ou chart para ilustrar o resultado. Vamos também apresentar os coeficientes ajustados para esta reta no painel frontal.

O MMQ com formalismo matricial é baseado em três equações fundamentais apresentadas a seguir.

$$\boldsymbol{Y = X \cdot A_0 + e}$$

$$\boldsymbol{\tilde{A} = \left(X^t \cdot V^{-1} \cdot X\right)^{-1} \cdot X^t \cdot V^{-1} \cdot Y}$$

$$\boldsymbol{V_{\tilde{A}} = \left(X^t \cdot V^{-1} \cdot X\right)^{-1}}$$

onde $\boldsymbol{Y}$ é o vetor de dados,$\boldsymbol{A_0}$ o vetor de parâmetros, $\boldsymbol{X}$ a matriz de planejamento, $\boldsymbol{e}$ o vetor de erros e $\boldsymbol{\tilde{A}}$ o vetor de parâmetros ajustados quando igualadas as derivadas parciais de cada parâmetro a zero. Estas equações representam o conjunto de dados experimentais na forma matricial:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & x_{1m} \\ x_{21} & x_{22} & x_{2m} \\ x_{n1} & x_{n2} & x_{nm} \end{pmatrix} \cdot \begin{pmatrix} a_{01} \\ a_{02} \\ a_{0n} \end{pmatrix} + \begin{matrix} e_1 \\ e_2 \\ e_n \end{matrix}$$

Exercício 23: Express VI Curve Fitting e MMQ com formalismo matricial

Utilizando o MMQ vamos obter os mesmos coeficientes sem utilizar a Express VI. Primeiramente vamos utilizar uma VI da paleta Curve Fitting. Posteriormente vamos utilizar o método matricial e então entender o funcionamento da VI General LS Linear Fit da paleta Curve Fitting.

Com estes conceitos podemos entender melhor os procedimentos de ajuste de funções, obtendo resultados que podem inclusive ser utilizados para interpolações futuras e levando em conta variâncias e covariâncias para propagação de erros nos resultados. Podemos ainda aprofundar a utilização com outras funções e modelos de ajuste.

12 Comunicação de dados

12.1 Comunicação em Rede – TCP/IP

TCP/IP é proveniente de um pacote de protocolos de comunicação, originalmente desenvolvido para a DARPA (*Defense Advanced Researche Projects Agency*). Desde seu desenvolvimento e se tornou amplamente aceitável e hoje é disponível em quase a totalidade dos microcomputadores e em muitos dispositivos eletrônicos.

O nome TCP/IP vem de dois protocolos principais do pacote, o de controle de transmissão (*Transmission Control Protocol*) e o de internet (*Internet Protocol*). TCP, IP e UDP (*User Data Protocol*) são as ferramentas básicas para comunicação em rede.

TCP/IP permite comunicações em redes individuais ou conjunto de redes (Internet). É importante lembrar que mesmo as redes individuais podem estar separadas por grandes distâncias geográficas.

O protocolo IP é raramente utilizado de forma direta nas aplicações porque ele não garante que o dado irá chegar em seu destino, ou mesmo que a ordem seja garantida no recipiente. Este fato se deve a capacidade do IP de gerenciar pequenas porções de dados somente. Desta forma o TCP é um protocolo de mais alto nível, que utiliza o IP para transferência de dados. O TCP quebra os dados em componentes que o IP possa gerenciar. Esta ação habilita a detecção de erros e garante que os dados chegarão no destino, em ordem e sem duplicatas. Por esta razão, o TCP é normalmente a melhor opção para aplicativos em rede.

O LabVIEW possui VIs específicas para o trabalho com protocolos TCP/IP e alguns detalhes iremos explorar aqui. As VIs de protocolo TCP/IP encontram-se na paleta Communication>>TCP:

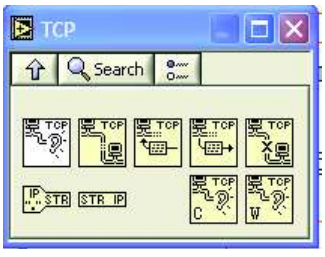


Figura 117 – Acesso a paleta TCP.

Para permitir um roteamento utilizando o protocolo TCP/IP, o LabVIEW utiliza o endereçamento lógico IP, portanto, a primeira parte do problema se caracteriza na determinação do endereço IP da maquina que será utilizada. A forma mais fácil de se encontrar este endereço no sistema Windows é utilizar o "*Prompt* de comando" da maquina que esta sendo utilizada com o comando: *ipconfig.*

Se a maquina não estiver conectada a uma rede o programa indicará apenas uma mensagem indicando a ausência de rede conectada a qualquer de seus dispositivos de acesso (Rede, Rede sem fio, etc.). Caso a maquina esteja conectada a uma rede, o programa indicará o número IP (constituído de 4 bytes e representado na forma decimal – Ex. 255.255.255.255).

A seguir veremos a construção de um modelo Cliente/servidor para comunicação TCP/IP e, neste caso, a VI Cliente irá requisitar serviços para a VI Servidor. Este exemplo ilustra de forma bem simplificada uma primeira conexão TCP/IP para comunicação de dados.

12.1.1 Cliente

Para montar sua VI Cliente, selecione a VI *TCP Open Connection* para abrir a conexão utilizado um endereço IP específico (ou o nome do computador). Selecione a VI *TCP read* e indique o número de bytes da conexão de rede. Em seguida, selecione a VI *TCP Close Connection* para fechar a conexão de rede TCP. Selecione a VI *Type Cast* disponibilizada na paleta de funções em *Advanced>>Data Manipulation*. Esta função converte qualquer tipo de dado para *string*.

A figura abaixo apresenta o diagrama de blocos para a VI Cliente. Repare que, dentro do *While loop*, existem duas funções *"TCP Read"* na sequência. A primeira função serve para leitura da quantidade de bytes que será enviado pelo servidor (definida como 4).

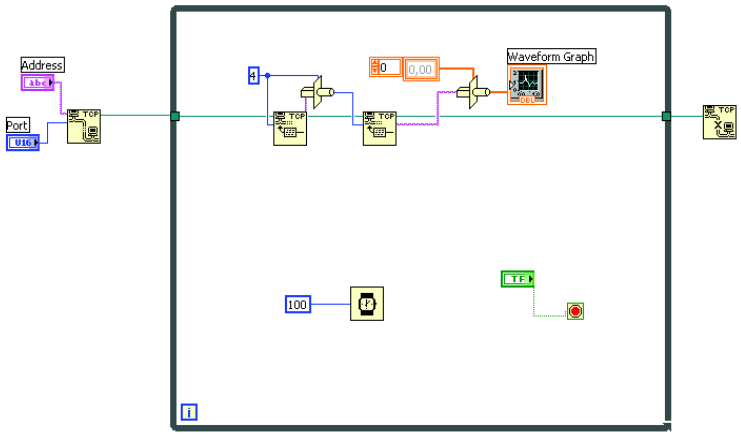


Figura 118 – Exemplo de utilização de VIs do grupo TCP.

12.1.2 Servidor

Para montarmos o Servidor, vamos selecionar a VI *TCP Listener* que tem a função de esperar um tempo determinado para o cliente se conectar. Aqui podemos verificar a sequência de operação Servidor/Cliente, que ficará evidente durante a execução final do exemplo. Rode sempre a VI que contem seu código para Servidor e posteriormente a VI que apresenta o código para Cliente.

Selecione a VI *TCP Write* para escrever os dados pela conexão de rede TCP e então selecione a VI *TCP Close Conection* para fechar a conexão de rede TCP. Selecione a VI *Type Cast* disponibilizada na paleta de funções em *Advanced>>Data Manipulation*. Esta função converte qualquer tipo de dado para string.

A figura abaixo apresenta o diagrama de blocos para a VI Servidor. Repare que, dentro do *While loop*, existem duas funções *"TCP Write"* na sequência. A primeira função serve somente para escrever a quantidade de bytes que será enviado pelo servidor e a segunda os dados.

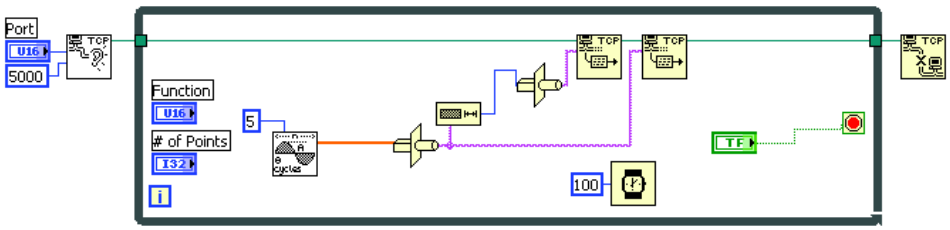


Figura 119 – Exemplo de utilização de VIs do grupo TCP.

Você pode rodar as VIs (cliente e Servidor) no mesmo computador para um teste inicial. Escreva, no endereço da VI "TCP *Open Connection*", o IP da maquina que esta sendo utilizada ou *"localhost"*.

Estas VIs de Cliente e servidor são baseadas nos modelos disponíveis nos exemplos do LabVIEW *Simple Dara Client*.vi e *Simple Data Server*.vi. Abrindo estes modelos você também pode explorar sua funcionalidade e utilizá-los para maior aprofundamento no assunto.

12.2 ACTIVE X

O LabVIEW fornece acesso a outros ambientes do Windows através da tecnologia ActiveX. A tecnologia ActiveX fornece um padrão estabelecido pela Microsoft para comunicação entre linguagens ou tecnologias chamadas OLE e COM. O desenvolvimento baseado em componentes, COM (*Component Object Model*) é um paradigma de programação estabelecido pela Microsoft, assim como o OLE (*Object Linking and Embedding*), um mecanismo que aplicativos podem utilizar para apresentar dados de outros aplicativos, ou com recursos destes, para um cliente. O LabVIEW pode ser considerado um componente Active X e o trabalho com esta tecnologia nada mais é que uma disposição Servidor/Cliente, conforme vimos em TCP/IP, onde estes elementos estão representados por linguagens ou plataformas de programação distintas.

A ferramenta ActiveX pode ter diferentes funcionalidades dentro do LabVIEW e dentre essas, podemos citar algumas muito importantes como usar o LabVIEW como um cliente ActiveX para

iniciar o Excel. O LabVIEW também pode ser usado como um servidor além de permitir que componentes fiquem ligados ao seu painel frontal.

O LabVIEW fornece VIs para a utilização da ferramenta ActiveX e estas estão disponíveis na paleta de funções em *Communication>>ActiveX*.

Figura 120 – Acesso a paleta de funções ActiveX.

12.2.1 Utilizando os métodos do Excel (VB)

Quando acessamos o Excel por meio da interface ActiveX estamos acessando seus métodos já que este programa os disponibiliza para aplicações avançadas. Para entender um pouco melhor como esta interface funciona vamos criar um pequeno programa no Excel através da interface Visual Basic (VB) disponível no próprio pacote Microsoft ™ que executa operações de leitura e escrita em células de uma planilha. Na planilha denominada Plan1, através do menu *Ferramentas>Macro* acesse o editor do Visual Basic do Excel .

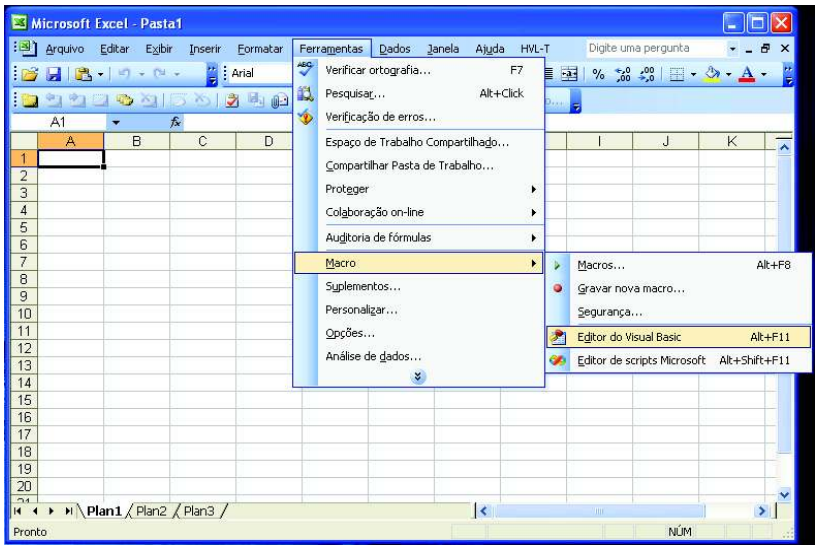


Figura 121 – Acesso as Macros do Excel no editor Visual Basic.

No editor Visual Basic vamos criar um Módulo por meio do menu *Inserir>Módulo*. Com o módulo 1 criado vamos criar uma macro por meio do menu *Ferramentas>Macros...* Crie a macro "Teste" que será incorporada ao módulo criado.

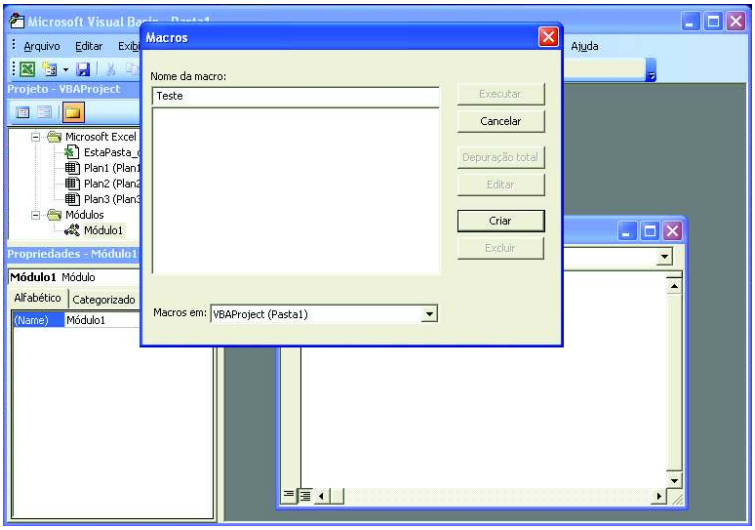


Figura 122 – Acesso as Macros do Excel no editor Visual Basic.

A macro cria automaticamente um método definido no VB por uma "Sub ()". Este método ou sub-rotina pode então ser utilizado para acessar dados ou mesmo incluir dados em qualquer planilha disponível dentro daquele arquivo ou mesmo externamente. Isto requer algum conhecimento de programação em VB e um grande conhecimento de como cada método do Excel opera. Para nosso exemplo, queremos apenas ter uma ideia de como o LabVIEW irá facilitar estes propósitos com sua linguagem.

Dentro da "Sub Teste ()" iremos instanciar a planilha Plan1 de forma bem simples, digitando "Plan1." dentro do método. Estaremos visualizando então todos os métodos disponíveis para a planilha, dentre os quais vamos trabalhar o "Range" que estabelece diretamente, por meio de uma *string*, um endereçamento de célula por coordenadas de Coluna e Linha. Estes métodos são igualmente acessíveis na programação com o LabVIEW.

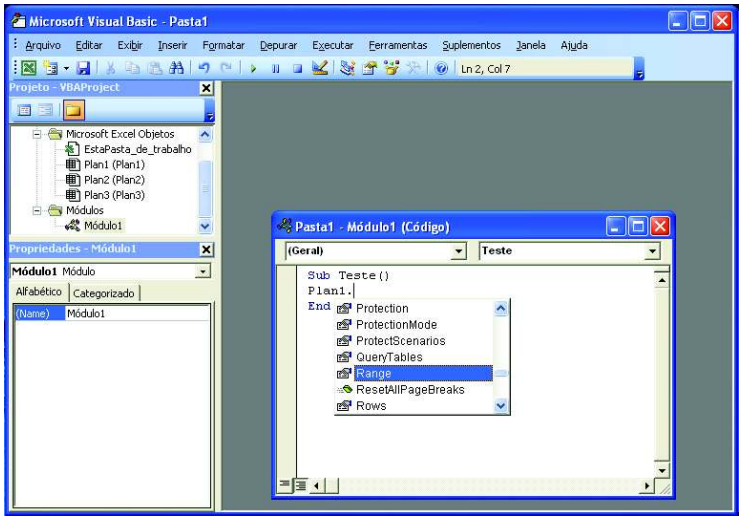
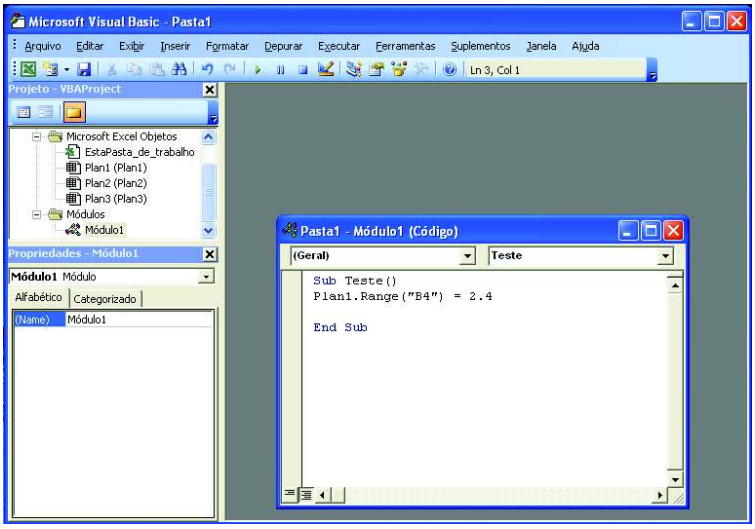


Figura 123 – Acesso as Macros do Excel no editor Visual Basic.

Vamos então escrever na célula “B4” do Excel o valor 2,4.



Executando a macro pelo menu *Executar* ou pelos botões de execução veremos o resultado. Da mesma forma podemos fazer diversas operações disponibilizadas pelos métodos desta planilha.

Agora vamos inserir manualmente na planilha na célula “A5” o valor 3,5. No módulo 1 vamos escrever uma rotina que lê este valor e o retorna a célula “C5” o quadrado do valor contido em “A5”.

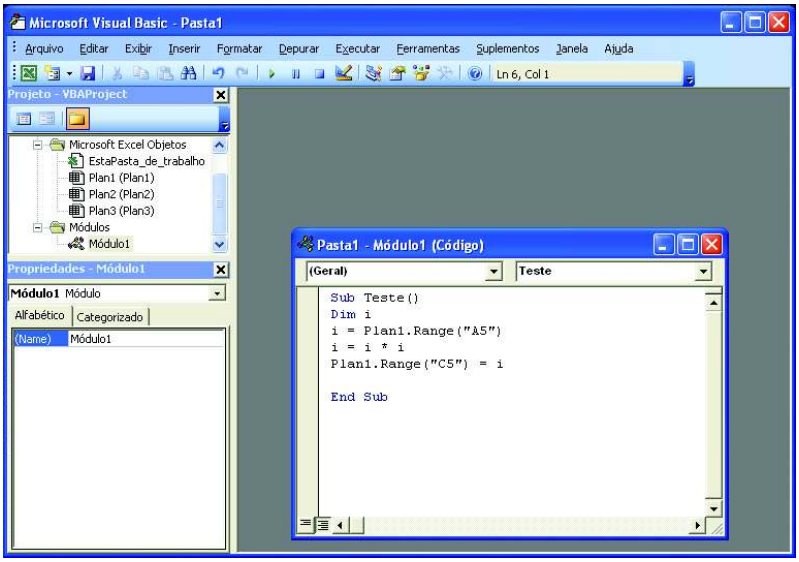


Figura 124 – Acesso as Macros do Excel no editor Visual Basic.

É possível ver a partir daqui que o trabalho com o editor VB requer conhecimento mínimo de linhas de comando em VB e irá requerer grande conhecimento e habilidade do programador quando precisarmos trabalhar com maior complexidade de dados e operações. Outra limitação é o acesso a periféricos como placas de aquisição ou mesmo portas. Esta integração é muito facilitada quando temos como interface o LabVIEW. Neste caso em particular ele pode funcionar como ponte entre a aquisição de dados e o preenchimento da planilha automaticamente.

Devido a padronização de procedimentos e registros na plataforma Excel ou mesmo Word, Access e outros utilitários da Microsoft, um sistema que já apresenta funcionalidade pode ser automatizado ou ainda aperfeiçoado utilizando-se como interface o LabVIEW.

12.2.2 LabVIEW - Excel / ActiveX

Agora vamos explorar como o LabVIEW interpreta e executa funções na plataforma Excel utilizando o Active X. Primeiramente crie uma "blank VI". Selecione na paleta de funções, *communication (ou conectivity)>>Active X>>Automation open function* e clique com o botão direito do mouse para selecionar uma função. Em seguida, selecione *ActiveX Class*. Neste momento, aparecerá uma janela.

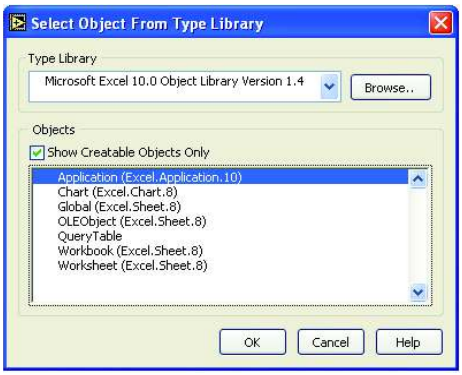


Figura 125 – Acesso aos Objetos na biblioteca do LabVIEW.

Você irá verificar uma diversidade imensa de plataformas de interação, ou elementos Active X, que o LabVIEW pode estabelecer comunicação para transferência de dados. Selecione a plataforma Microsoft Excel e o objeto *Application* para montarmos nosso primeiro aplicativo. A partir daí observe no painel frontal que seu código já possui o vínculo com um aplicativo Excel.

Agora selecione na paleta de funções *communication (ou connectivity)>>Active X* a VI *Property Node*. Conecte o vínculo criado anteriormente a esta VI e em seguida clique com o botão direito do mouse sobre este objeto e selecione *Properties>>Workbooks*. Clicando novamente com o botão direito do mouse sobre o "Property Node", selecione *Add Element* e, em seguida, escolha o método *Visible*. Selecione este elemento e altere sua propriedade de entrada e saída clicando em *Change to write.*

Conecte as saídas e entradas de erro entre os objetos (VIs utilizadas). Selecione, na paleta de funções, *communication>>Active X>>Invoke Node*. Conecte também esta VI aos outros dois objetos, ligando agora seu vínculo ao Workbook da VI Property Node. A VI Invoke Node executará um método, portanto, clicando com o botão direito do mouse, selecione *Methods>>Open*.

Após a abertura e operação Active X as referencias do servidor devem ser fechadas, tanto para o Workbook (planilha) quanto para o Aplicativo, portanto, selecione na paleta de funções, *communication>>Active X>>Close reference*.

A figura abaixo apresenta o diagrama de blocos para uma VI que abre uma referencia no Excel e, em seguida, inicia o Excel;

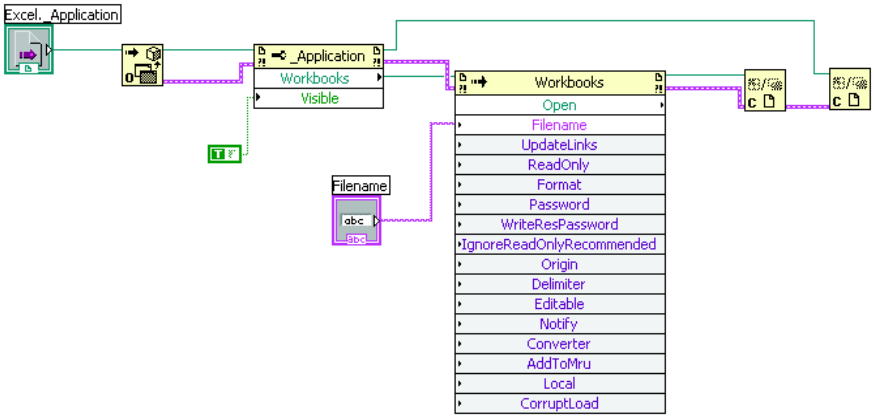


Figura 126 – Exemplo de acesso a um aplicativo Excel no diagrama de blocos.

Indique em "*File name*" o nome de um arquivo que se deseje abrir anteriormente à execução desta VI. Esta ação está limitada a condições específicas de endereçamento de arquivo, requerendo que o mesmo seja detalhadamente descrito na string Filename para sua correta localização.

Exercício 24: Enviando dados de medições para o Excel

Na figura abaixo é apresentada uma planilha do Excel onde temos campos específicos que devem ser preenchidos. Crie uma VI que possa simular aquisição de dados de uma forma de onda de tensão e, em seguida calcule os parâmetros indicados na planilha. Esta VI deve simular, em cada vez que rodar, 10 formas de onda diferentes.

	A	B	C	D	E	F	G
1							
2		Tensão Máxima	RMS	Pico a Pico			
3	1				Desvio padrão (Tensão Máx) =		
4	2				Desvio padrão (RMS) =		
5	3				Desvio padrão (Pico a Pico) =		
6	4						
7	5						
8	6						
9	7						
10	8						
11	9						
12	10						
13							
14							

Figura 127 – Exercício de utilização do ActiveX para comunicação LabVIEW - Excel.

12.3 *Simple Mail Transfer Protocol* (SMTP)

Antes de desenvolvermos uma VI para trabalhar com protocolos de transferência de e-mails é necessário entendermos um pouco do funcionamento do SMT. Normalmente um protocolo com o termo "simples" deve ser tudo, menos simples. O SMTP não é um protocolo de difícil manipulação desde que conheçamos alguns de seus comandos principais e sua metodologia básica de operação.

O SMTP é definido em RFC (*Request For Comments* 811), que é um padrão para Internet. Basicamente, cada comando que enviamos provocará uma resposta do servidor. Estas respostas do servidor consistem basicamente de três dígitos numéricos e um texto. Neste protocolo estamos mais preocupados com o primeiro dígito que varia de dois a cinco. Respostas com início 2 são positivas, ou seja, fizemos algo certo ao enviar uma informação ao servidor e o mesmo está nos permitindo continuar. Um número 3 indica que efetuamos uma ação aceitável, mas que alguma informação está faltando, ou seja, a informação está incompleta. Indicadores 4 e 5 representam erros que deveremos trabalhar, ou seja não fizemos a coisa correta. Por esta breve introdução podemos perceber que o trabalho com máquinas de estado para o tratamento de protocolos SMTP. Faremos códigos, no entanto, utilizando outras técnicas, já que máquinas de estado não é objetivo deste módulo intermediário.

Um ID de usuário válido deve ser informado para que uma ação tenha sucesso. Uma mensagem com código 250 indica, por exemplo, que o usuário é válido e pode enviar uma mensagem (e-mail). O endereçamento da mensagem vem em seguida, e isto é efetuado com "RCPT TO:<*email address*>". Novamente o servidor deverá responder com um código 250. Para preencher o conteúdo da mensagem, o comando DATA é enviado e deve aguardar uma resposta 354 do servidor, que indicará aceitação do comando, mas o mesmo não será concluído até que enviemos a sequência <CRLF>.<CRLF>. Após esta ação estamos prontos para enviar o corpo da mensagem e o servidor não irá responder até que enviemos outra combinação "*carriage return, line feed*". Neste momento, o servidor responderá com outro código 250, e então finalizamos o envio da mensagem. Enviamos por fim um comando QUIT e o servidor responderá com 220 efetivando o desligamento da linha.

O maior problema no LabVIEW é que nenhum suporte para autenticação é fornecido para VIs do grupo SMTP mail. Desta forma, a utilização direta deste grupo de VIs fica limitado a liberação por IP de cada usuário, definido nas diretrizes do servidor SMTP designado, que irá habilitar o acesso por endereço físico sem identificação de usuário e senha.

Um maior detalhamento dos comandos e tratamento de erros em protocolo SMTP podem ser obtidos pelo LabVIEW Help, acessando o conteúdo SMTP Email *Reply Codes*.

Para exercitar o uso destas VIs vamos utilizar a VI SMTP Email *Send Message* configurando suas entradas de dados e efetuando o envio de e-mail com a mensagem teste.

Uma forma de nos familiarizarmos com uma VI para trabalho com protocolo SMTP e depois partirmos para um desenvolvimento mais aprofundado ou resumido em nossas aplicações é abrindo a VI E-mail *Notification* disponível nos exemplos do LabVIEW.

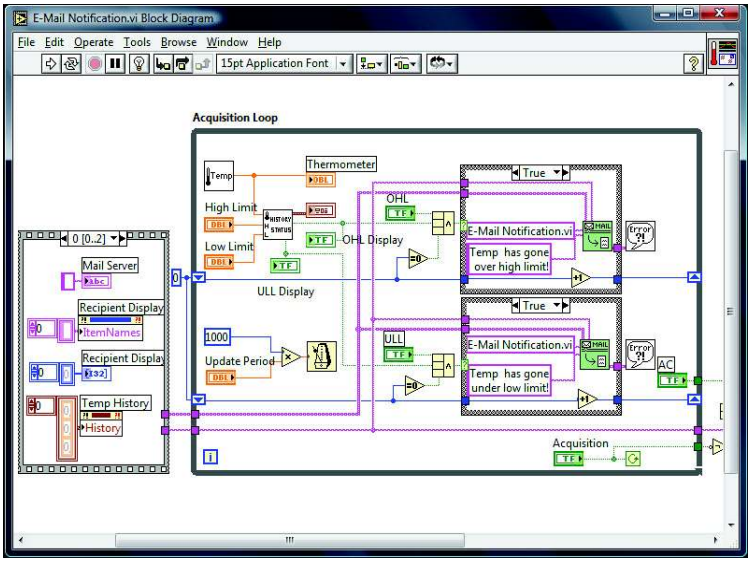


Figura 128 – Diagrama de blocos da VI E-mail *Notification*.

Nesta VI é possível verificar a conexão com um SMTP válido, mas sua efetivação só será possível se o aplicativo estiver instalado em uma máquina com acesso autorizado pelo servidor (IP de acesso autorizado).

12.4 Introdução as bibliotecas VISA no LabVIEW

VISA (*Virtual Instrument Software Architecture*) é uma biblioteca de interface de controle individual para controles VXI, GPIB, RS-232, e outros tipos de instrumentos. Esta biblioteca fornece uma série de padrões para I/O utilizados para todos os drivers de instrumentos no LabVIEW. Utilizando as funções VISA, você pode construir uma VI que servirá como driver de seus instrumentos que possuem os padrões de interface conforme citado.

Uma *string* para descrição do instrumento é utilizada para abrir uma comunicação VISA, selecionando que tipo de I/O será utilizado com o instrumento. A partir daí, funções de escrita e leitura irão gerenciar o processo de comunicação para se estabelecer uma funcionalidade específica. Os drivers destes instrumentos são considerados de interface independente e podem ser utilizados desta forma em qualquer outro sistema.

Vamos abordar aqui dois tipos de interfaces, GPIB e RS-232. A exploração deste tema é, contudo, dependente de cada aplicação e de cada instrumento ou dispositivo que comporta um driver para este padrão de comunicação e o aprofundamento requer conhecimento não só das funções da biblioteca VISA mas do dispositivo em uso.

Normalmente, equipamentos e instrumentos que possuem drivers para comunicação com sistemas de controle possuem protocolos de comunicação disponibilizados por meio de manuais ou documentação técnica, que é parte integrante da documentação acompanhante do mesmo, o que facilita sua operação e a geração de aplicativos de controle e aquisição de dados por meio de linguagens de programação. Os drivers de instrumentos e outros dispositivos que efetuam a comunicação com as funções VISA no LabVIEW efetuarão, desta forma, ações específicas pré-estabelecidas pelo fabricante do dispositivo e não específicas a interface de comunicação.

12.4.1 Interface GPIB

A GPIB (*General Purpose Interface Bus*) é um link, ou sistema de interface, pela qual dispositivos eletrônicos interconectados podem se comunicar. Desenvolvido pela HP para comunicação de sua instrumentação programável, o padrão foi logo implementado em outras aplicações como a comunicação entre microcomputadores e interfaces, devido a sua velocidade máxima de transferência (1 Mbyte/s). Foi então aceita como padrão IEEE 488-1975 e posteriormente ANSI/IEEE 488.2-1987, como mais comumente a conhecemos.

O padrão 488.2 descreve exatamente como o controlador deve gerenciar a GPIB, incluindo as mensagens padrão que devem ser implementadas nos dispositivos certificados, mecanismos de reportagem de erros e informações de status, além de vários protocolos para detectar e configurar dispositivos GPIB interconectados.

A diversidade de hardwares GPIB compatíveis com o LabVIEW para os diversos sistemas operacionais disponíveis é encontrada no Help ou LabVIEW zone na Web. As funções tradicionais para protocolo GPIB no LabVIEW são compatíveis com dispositivos IEEE488 e IEEE488.2 e normalmente são suficientes para a maioria das aplicações.

A primeira e mais fácil maneira de se familiarizar com as VIs VISA para uma comunicação GPIB é a utilização da VI exemplo SRQ *Event Handling*. Esta VI explora os princípios básicos da comunicação GPIB que irão se reproduzir na maioria dos aplicativo que utilizam este padrão de interface.

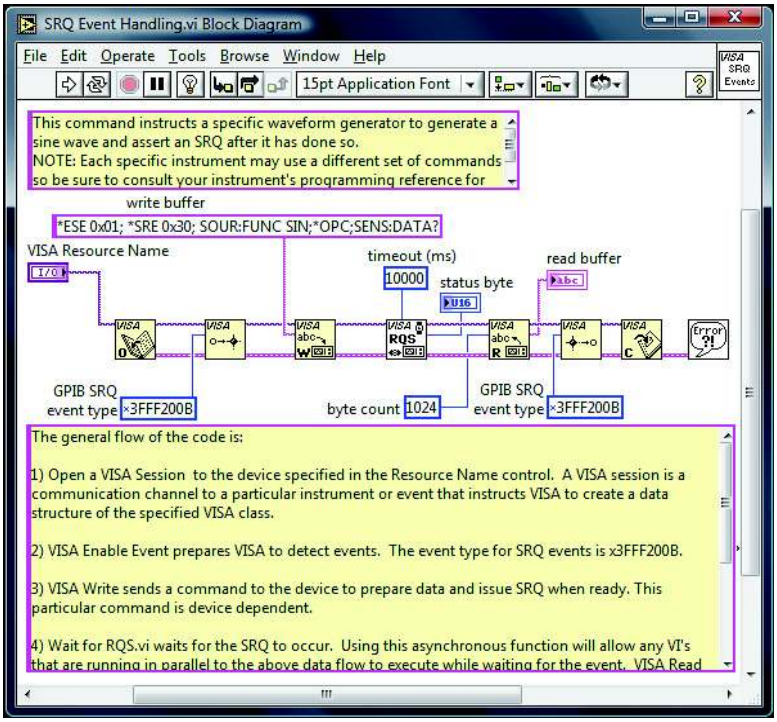


Figura 129 – Diagrama de blocos da VI SRQ *Event Handling*.

Logicamente, para programadores com maior experiência e vivência com este tipo de interface, a programação de seus parâmetros, de forma mais complexa e aprofundada pode ser efetuada no LabVIEW por meio da paleta *Instrument* I/O>>488 (GPIB). Esta paleta traz todas as possibilidades de configuração destas interfaces e seu detalhamento está no LabVIEW Help.

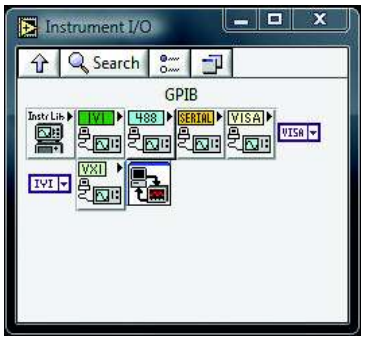


Figura 130 – Acesso a paleta de funções *Instrument I/O* no diagrama de blocos.

12.4.2 Interface RS-232

Este padrão foi originalmente usado para conectar um *teletipo* (equipamento eletromecânico de comunicação assíncrona que usava código ASCII) a um modem. Quando terminais eletrônicos (burros ou não) começaram a ser usados, eram projetados para serem intercambiáveis com as *teletypewriters*, e também suportavam RS-232. A terceira revisão deste padrão (chamada de RS-232C) fora publicada em 1969, em parte para adequar-se às características elétricas destes dispositivos. Deste modo, fora utilizado em diversos tipos de comunicação remota, especialmente por modems. Posteriormente PCs (e outros equipamentos) começaram a utilizar este padrão para comunicação com equipamentos já existentes. Quando a IBM lançou computadores com uma porta RS-232, esta interface tornou-se realmente onipresente. Por muitos anos o padrão para comunicação serial em quase todos os computadores era algum tipo de porta RS-232. Continuou sendo utilizado em grande escala até o fim dos anos 90. Durante este tempo esta foi a maneira padrão para a conexão de modems.

Uma exceção eram os mainframes, que geralmente não se comunicavam diretamente com dispositivos terminais. Estes costumavam ter processadores especializados em E/S conectados a eles, por exemplo, alguns mainframes da IBM possuíam uma unidade de controle de telecomunicação (TCU - *Telecommunication Control Unit*, Unidade de Controle de Telecomunicação) anexados a seus canais multiplexadores. O TCU deveria suportar múltiplos terminais, às vezes centenas. Vários desses TCUs suportavam RS-232 quando necessário, assim como outras interfaces seriais.

Há alguma confusão sobre o que a EIA (*Eletronics Industries Alliance*) padronizou no RS-232. Este padrão apenas especifica características elétricas dos circuitos e a numeração dos pinos. Outras características como o conector em forma de "D", o uso de código ASCII, formato dos dados e comunicação assíncrona não são parte do RS-232, a palavra "padrão", porém, é utilizada geralmente quando todos estas características aparecem juntas, de modo a tornarem-se efetivamente obrigatórias. Foram construídas em torno de 100.000 *teletypewriters* (33-ASR) e milhares de PCs feitos toda semana, todos eles podem atuar como *teletypewriters* virtuais. Uma única característica que era utilizada em *teletypewriters* e que fora abandonada é que uma *teletypewriter* real requer dois bits de parada para trabalhar de modo satisfatório, deste modo um caractere ocupava 11 bits. Por isso *teletypewriters* de 100 palavras por minuto transmitiam a 110 bauds. Hoje em dia utiliza-se apenas um bit de parada. Sendo que trataremos aqui uma simulação de um 33-ASR, não o documento RS-232.

A IBM favoreceu o uso do código EBCDIC de oito bits ao invés do ASCII com sete bits, favoreceu também um formato de transmissão "big endian" ao invés do formato "little endian" do ASCII. A IBM ofereceu suporte a esses outros formatos de modo que, para transmitir caracteres "little endian", o mainframe precisaria somente inverter cada caractere usando uma instrução para tradução de bloco. Os primeiros *teletypewriters* tinham três linhas de teclas e suportavam

somente letras maiúsculas. Elas utilizavam o código Baudot e, geralmente, trabalhavam a taxas de 60 palavras por minuto. Os equipamentos com teclados de quatro linhas, código ASCII e letras maiúsculas e minúsculas começaram a aparecer quando computadores pessoais se popularizaram. Os circuitos integrados de comunicação serial UART, introduzidos no início dos anos 70, continuam sendo emulados por muitos *chipsets* e ainda suportam os primeiros formatos, incluindo 1,5 bits de parada. Contudo tais recursos são raramente utilizados.

A importância de portas seriais começou a decrescer gradualmente quando redes de alta velocidade tornaram-se disponíveis para comunicação PC com PC. Hoje é comum utilizar conexões Ethernet Base 10 ou 100.

Mesmo com o advento de outros padrões com velocidade de comunicação muito superior, o padrão RS-232 permanece em praticamente todos os instrumentos eletrônicos devido a sua simplicidade e estabilidade de comunicação. O LabVIEW possibilita o acesso a este padrão por meio de suas VIs VISA e os resultados são muito interessantes. De forma similar ao exemplo explorado para o padrão GPIB, vamos mostrar uma abertura de comunicação RS-232:

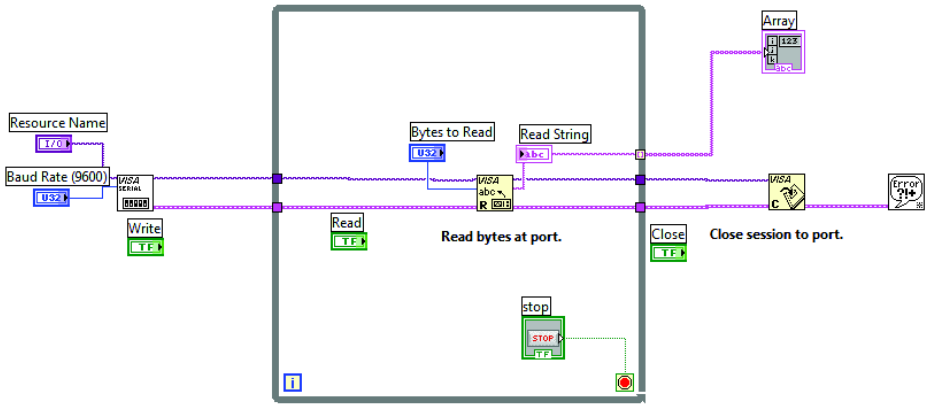


Figura 131 – Exemplo de utilização de VI do grupo VIS no diagrama de blocos.

Este exemplo ilustra a abertura de uma comunicação RS-232 com a VI VISA configure serial port, que apresenta as entradas de dados de todos os parâmetros que necessitamos configurar numa comunicação padrão RS-232:

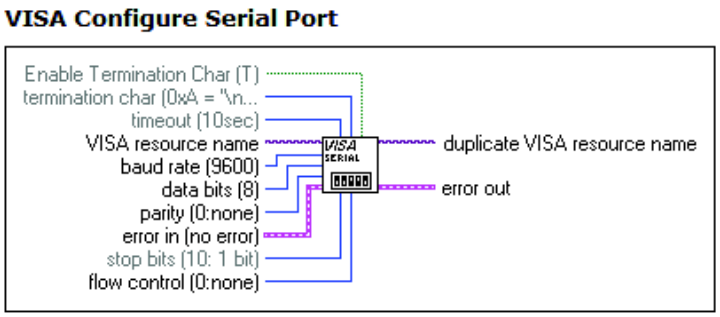


Figura 132 – Configuração da VI VISA serial.

O detalhamento de cada entrada ou parâmetro de operação serial pode ser obtido no LabVIEW Help. As configurações destes parâmetro normalmente são impostas pelo dispositivo que iremos trabalhar na prática.

Um exemplo prático de comunicação serial é a interação com a interface modular CNZ 51-MX. Este kit de desenvolvimento para microcontroladores da família 8051 traz uma porta de acesso para programação e aplicação com interfaces seriais, inclusive em padrão RS-232. Dois programas podem ser testados utilizando-se este dispositivo: Envio de "Banner", ou seja, uma mensagem (*string*) enviada pelo LabVIEW através da porta serial é implementada no display LCD do kit 51-MX; Envio de caracteres para obtenção de código ASCII, onde o kit 51-MX retorna o código ASCII de qualquer caractere enviado.

Para facilitar a implementação deste exemplo vamos utilizar uma VI existente nos exemplos do LabVIEW, a VI Serial Communication.

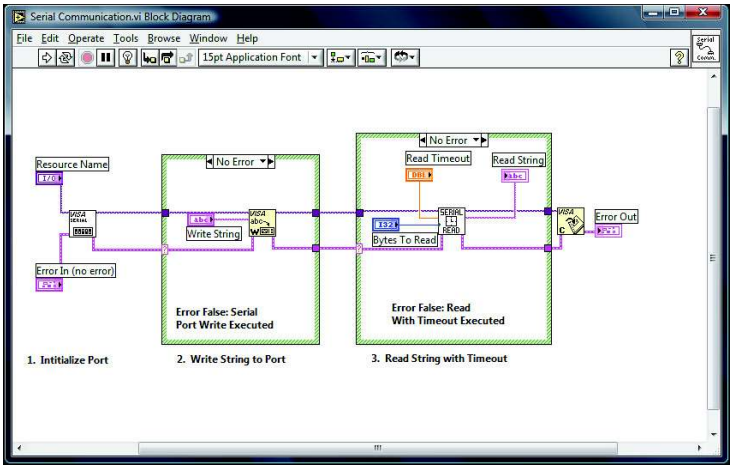


Figura 133 – Exemplo de utilização das VIs do grupo VISA para comunicação serial no diagrama de blocos.

É sempre importante utilizar o *Measurement & Automation*, já explorado anteriormente, para verificarmos se nossas interfaces estão ativas e reconhecidas como tal pelo LabVIEW.

Para versões mais atuais do LabVIEW, como o LabVIEW 8, temos melhorias relacionadas a detecção automática que pode ser verificada ao criarmos um controle de entrada para o "*Resource Name*", que é o indicador de dispositivo para abertura de comunicação da VI VISA Serial.

Utilizando como exemplo esta VI, ou a VI *Basic Serial Write and Read* do LabVIEW 8, vamos comunicar nossa plataforma com este periférico. Mesmo sendo uma aplicação muito simples, ela ilustra o que vamos encontrar na maioria de nossos aplicativos com sistemas, instrumentos ou outros dispositivos com comunicação serial, ou seja, execução de comandos de escrita e leitura, já

definidos pelos fabricantes destes periféricos, que irão disponibilizar comandos e leituras de dados de processos.

13 Considerações Finais

Com o conhecimento básico da estrutura de programação inicial com LabVIEW já é possível vislumbrar inúmeras aplicações laboratoriais. Inúmeros exemplos de aplicações e estudos de casos são apresentados no site da National Instruments, bem como estudos de casos que podem servir de guia para aplicações práticas.

Obviamente a aquisição de dados depende essencialmente das interfaces com dispositivos externos para comunicação do ambiente virtual com o ambiente real, o que é facilitado pela ampla biblioteca de drivers disponibilizados pela NI e por outros fabricantes e fornecedores de equipamentos e instrumentos. Além disso, já podemos simular dados reais ou mesmo utilizar conjuntos de dados reais para nossas simulações, o que consiste em uma importante ferramenta de validação de métodos e resultados, requisito fundamental para laboratórios de pesquisa, ensaios e desenvolvimento tecnológico.

Parte dos exemplos apresentados neste livro podem ser obtidos por meio de download no site do IEE-USP.

14 Referências Bibliográficas

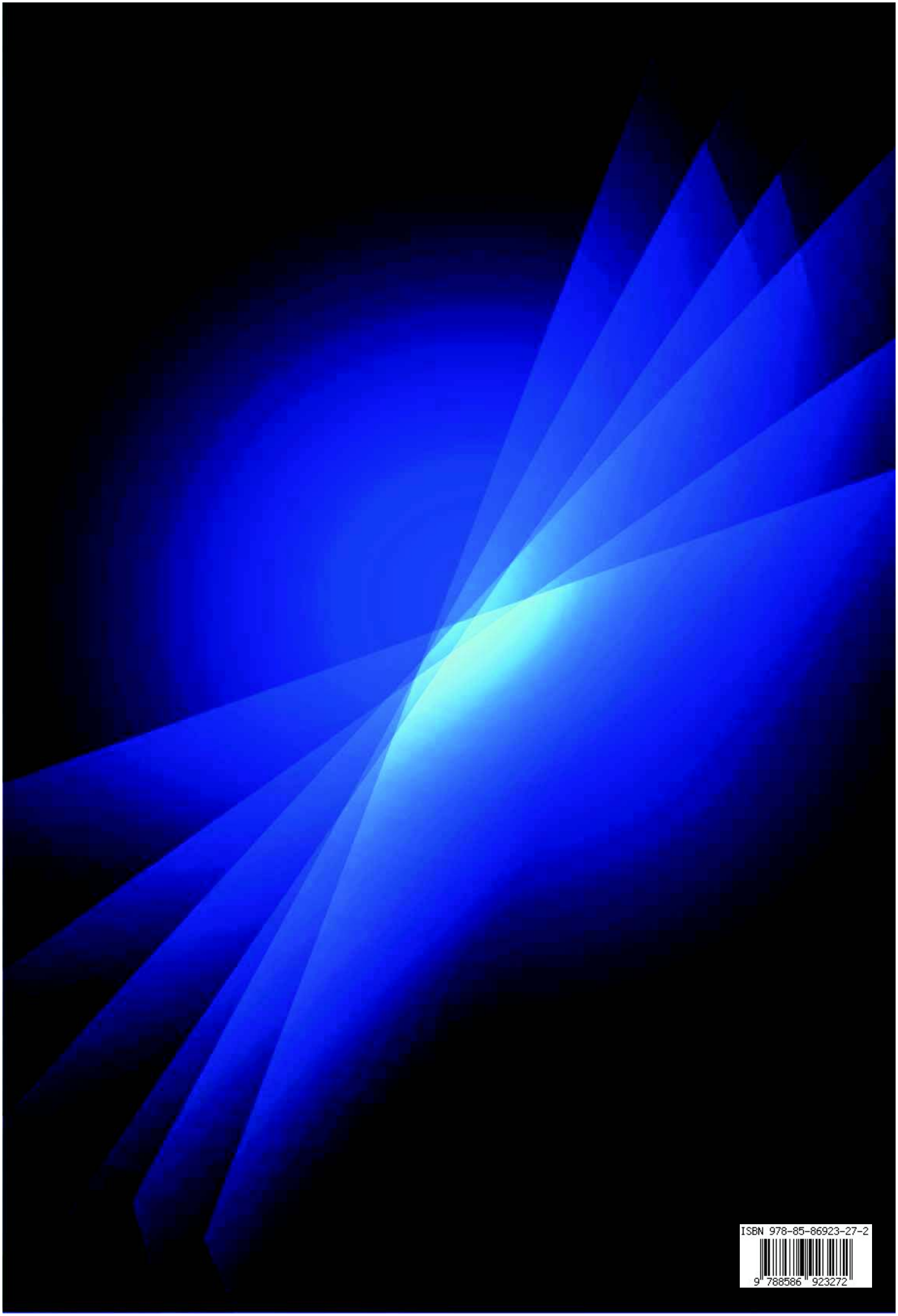
LabVIEW Tutorial Manual. National Instruments Corporation, 1996. 6504 Bridge Point Parkway, Austin.

LabVIEW Function and VI Reference Manual. National Instruments Corporation, 1997. 6504 Bridge Point Parkway, Austin.

LabVIEW Data Acquisition Basics Manual. National Instruments Corporation, 1998. 6504 Bridge Point Parkway, Austin.

Rick Bitter, Taqi Mohiuddin and Matt Nawrocki. LabVIEW advanced programming techniques. CRC Press LLC,2001 – Boca Raton.

Otaviano Helene. Métodos dos Mínimos Quadrados com Formalismo Matricial. Editora Livraria da Física. São Paulo, 2006.


ISBN 978-85-86923-27-2