CHAPTER 1

INTRODUCTION TO LabVIEW FOR BIOMEDICAL INSTRUMENTATION
CHAPTER – I
INTRODUCTION TO LabVIEW FOR BIO-MEDICAL INSTRUMENTATION

1.1 INTRODUCTION TO LabVIEW AND SALIENT FEATURES:

LabVIEW (short for Laboratory Virtual Instrumentation Engineering Workbench) \(^1\) is a system design software that provides engineers and scientists with the tools needed to create and deploy measurement and control systems through unprecedented hardware integration. LabVIEW helps the user to solve problems, accelerates the user productivity, and gives the user confidence to continually innovate. LabVIEW is a programming language which allows simple interfacing with measurement hardware. With its unique graphical programming environment, libraries containing knowledge of thousands of devices and useful tool-kits, acquiring data is very easy and easy to control and it can acquire data from any instrument over any bus. The user can automate measurements from several devices, analyze the acquired data and can create custom reports, all in a matter of minutes. LabVIEW helps the user to avoid spending hours learning how to take measurements from a particular device. With its help, the user can focus on the results rather than a process of obtaining them. Developing measurement systems can be faster with LabVIEW with the graphical system design approach. Within a single software environment, LabVIEW provides unparalleled integration with NI or third-party data acquisition hardware, extensive signal processing libraries, and user interface controls purpose-built for user to visualize the measurement of data.

The programming language used in labVIEW, also referred to as G, is a dataflow programming language \(^2\). Program execution is determined by the structure of a
graphical block on which the programmer connects different functioning nodes by drawing wires. These wires propagate variables and any node can execute as soon as all input data become available. Since this might be the case for multiple nodes functioning simultaneously, G is inherently capable of parallel execution of codes. Multi-processing and multi-threading hardware is automatically exploited by the built in scheduler, which multiplexes multiple OS threads over the nodes which are ready for execution.

The dataflow completely defines the execution sequence, and that can be fully controlled by the programmer. Thus, the execution sequence of the labVIEW graphical syntax is well defined with any textually coded language such as C, Visual Basic, etc. Furthermore, labVIEW does not require type definition of the variables; the wire type is defined by the data-supplying node. LabVIEW supports polymorphism and the wires automatically adjust to various types of data.

LabVIEW ties the creation of user Interfaces (also called front panel) into the development cycle. LabVIEW programs are called Virtual Instruments (VI’S) because they have the look and feel of physical systems or instruments.

One benefit of LabVIEW over other development environments is the extensive support, which it gives for accessing instrumentation hardware. Drivers and abstraction layers for many different types of instruments and buses are included and are available for inclusion. These present themselves as graphical nodes. The abstraction layers offer standard software interfaces to communicate with hardware devices. The provided driver interfaces save program development time.
In terms of performance, LabVIEW includes a compiler that produces native code for the CPU platform. The graphical code is translated into executable machine code by interpreting the syntax and by compilation of codes. The LabVIEW syntax is strictly enforced during the editing process and compiled into the executable machine code when requested to run or upon saving. In the latter case, the executable and the source code are merged into a single file. The executable runs with the help of the LabVIEW run-time engine, which contains some precompiled code to perform common tasks that are defined by the G language. The run-time engine reduces compile time and also provides a consistent interface to various operating systems, graphic systems, hardware components, etc. The run-time environment makes the code portable across platforms. Generally, LV code can be slower than equivalent compiled C code, although the differences often lie more with program optimization than inherent execution speed.

LabVIEW uses a patented data flow programming model that frees you from the linear architecture of text-based programming languages. Because the execution order in LabVIEW is determined by the flow of data between nodes\(^3\), and not by sequential lines of text, and user can create block diagrams that execute multiple operations in parallel. Consequently, LabVIEW is a multitasking system capable of running multiple execution threads and multiple VIs in parallel.

LabVIEW is a data acquisition software package commonly used with hardware acquisition boards, LabVIEW has many features for data acquisition and processing of either measured data or simulated signals.

LabVIEW is an application that lets you interface a computer with an experiment. It is extremely powerful, allowing you to generate and measure analog and digital
voltages as well as control the timing of such operations. In order to have such a powerful application, there must be a huge amount of flexibility in what the user can program.

The LabVIEW platform provides specific tools and models to solve specific applications ranging from designing signal processing algorithms to making voltage measurements and can target any number of platforms from the desktop to embedded devices – with an intuitive, powerful graphical paradigm.

1.1.1 ADVANTAGES OF LabVIEW:

a. Interfacing

A key benefit of LabVIEW over other development environments is the extensive support for accessing instrumentation hardware. Drivers and abstraction layers for many different types of instruments and buses are included and are available for inclusion. These present themselves as graphical nodes. The abstraction layers offer standard software interfaces to communicate with hardware devices. The provided driver interfaces save program development time. The sales pitch of National Instruments is so popular, that even people with limited coding experience can write programs and deploy test solutions in a reduced time frame when compared to more conventional or competing systems. A new hardware driver topology (DAQmxBase), which consists mainly of G-coded components with only a few register calls through NI Measurement Hardware DDK (Driver Development Kit) functions, provides platform independent hardware access to numerous data acquisition and instrumentation
devices. The DAQmxBase driver is available for LabVIEW on Windows, Mac OS X and Linux platforms.

b. Code Compilation

In terms of performance, LabVIEW includes a compiler that produces native code for the CPU platform. The graphical code is translated into executable machine code by interpreting the syntax and by compilation of codes. The LabVIEW syntax is strictly enforced during the editing process and compiled into the executable machine code when requested to run or upon saving. In the latter case, the executable and the source code are merged into a single file. The executable runs with the help of the LabVIEW run-time engine, which contains some precompiled code to perform common tasks that are defined by the G language. The run-time engine reduces compile time and also provides a consistent interface to various operating systems, graphic systems, hardware components, etc. The run-time environment makes the code portable across platforms. Generally, LV code can be slower than equivalent compiled C code, although the differences often lie more with program optimization than inherent execution speed.

c. Large Libraries

Many libraries with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, etc., along with numerous graphical interface elements are provided in several LabVIEW package options. The number of advanced mathematic blocks for functions such as integration, filters, and other specialized capabilities usually associated with data capture from hardware sensors is immense.
Fig 1.1: Schematic diagram of an instrument system based on LabVIEW
In addition, LabVIEW includes a text-based programming component called MathScript with additional functionality for signal processing, analysis and mathematics. MathScript can be integrated with graphical programming using "script nodes" and uses a syntax that is generally compatible with MATLAB.

d. Code Re-use

The fully modular character of LabVIEW code allows code reuse without modifications: as long as the data types of input and output are consistent, two sub VIs are interchangeable. The LabVIEW Professional Development System allows creating stand-alone executables and the resultant executable can be distributed a number of times. The run-time engine and its libraries can be provided freely along with the executable.

A benefit of the LabVIEW environment is the platform independent nature of the G code, which is (with the exception of a few platform-specific functions) portable between the different LabVIEW systems for different operating systems (Windows, Mac OS X and Linux). National Instruments is increasingly focusing on the capability of deploying LabVIEW code onto an increasing number of targets including devices like Phar Lap or VxWorks OS based LabVIEW Real-Time controllers, FPGAs, Pocket PCs, PDAs, and Wireless sensor network nodes.

e. Educational Objectives

It is difficult to design laboratory exercises using complex scientific equipment and still have the student learn the concepts being demonstrated. Using LabVIEW to facilitate the data generation and collection changes the focus from learning how to use the equipment to learning the physiological concepts being presented in the lab. Because LabVIEW allows designing simple, easy-to-use interfaces, however, care must be taken
not to make the virtual instruments (VIs) so simple to use that students may not see the
details of the measurement process. Hence instructors must ensure that students do not
leave the lab thinking that measuring physiologic phenomena means you click once on a
button and the data magically appears.

f. Countless Applications

LabVIEW applications are implemented in many industries worldwide including
automotive, telecommunications, aerospace, semiconductor, electronic design and
production, process control, biomedical, and many others. Applications cover all phases
of product development from research to design to production and to service. By
leveraging LabVIEW throughout an organization the user can save time and money by
sharing information and software.

1.1.2 BENEFITS OF LabVIEW[^4]:

- **Graphical user Interface**: Design professionals use the drag-and-drop user interface
  library by interactively customizing the hundreds of built-in user objects on the controls
  palette.

- **Drag-and-drop built-in functions**: Thousands of built in functions and IP including
  analysis and I/O, from the functions palette to create applications easily.

- **Modular design and hierarchical design**: They run modular LabVIEW VIs by
  themselves or as subVI and easily scale and modularize programs depending on the
  application.

- **Multiple high level development tools**: develop faster with application specific
  development tools, including the LabVIEW state chart Module, LabVIEW control
  Design and Simulation Module and LabVIEW FPGA Module.
- **Professional Development Tools:** Manage large, professional applications and tightly integrated project management tools; integrated graphical debugging tools; and standardized source code control integration.

- **Multi platforms:** The majority of computer systems use the Microsoft Windows operating systems. LabVIEW works on other platforms like Mac OS, Sun Solaris and Linux. LabVIEW applications are portable across platforms.

- **Reduces cost and preserves investment:** A single computer system equipped with LabVIEW is used for countless applications and purposes – it is a versatile product. Complete instrumentation libraries can be created for less than the cost of a single traditional commercial instrument.

- **Flexibility and Scalability:** Engineers and scientists have needs and requirements that can change rapidly. They also need to have maintainable, extensible solutions that can be used for a long time. By creating virtual instruments based on powerful development software such as LabVIEW, the user can perfectly design an open framework that seamlessly integrates software and hardware.

- **Connectivity and instrument control:** LabVIEW has ready-to-use libraries for integrating stand-alone instruments, data acquisition devices, motion control and vision products, GPIB/IEEE 488 and serial/RS-232 devices, and PLCs to build a complete measurements and automation solution.

- **Open Environment:** LabVIEW provides the tools required for most applications and is also an open development environment. This open language takes advantage of existing code; can easily integrate with legacy systems and incorporate third party software with .NET, active, DLLs, objects, TCP, web services and XML formats.
- **Distributed environment**: Can easily develop distributed applications with LabVIEW, even across different platforms. With powerful server technology the user can offload processor-intensive routines to another machine for faster execution, or create remote monitoring and control applications.

- **Visualization capabilities**: LabVIEW include a wide variety of built-in visualization tools to present data on the user interface of the virtual instrument as chart, graphs, 2D and 3D visualization. Reconfiguring attributes of the data presentation, such as colours, font size, graph types, and more can be easily performed.

- **Rapid development with express technology**: Use configuration-based Express VIs and I/O assistants to rapidly create common measurement applications without programming by using LabVIEW signal Express.

- **Complied language for fast execution**: LabVIEW is a complied language that generates optimized code with execution speeds comparable to compiled C and develops high-performance code.

- **Simple application distribution**: use the LabVIEW applications builder to create executables (.EXE) and shared libraries (DLLs) for deployment.

- **Target Management**: Easily manage multiple targets, from real-time to embedded devices including FGPAs, microprocessors, microcontrollers, PDAs and touch panels.

- **Objects-oriented designs**: Use objects-oriented programming structures to take advantages of encapsulation and inheritance to create modular and extensible code.

- **Algorithm Design**: develop algorithms using math-oriented textual programming and interactively debug.m file script syntax with LabVIEW Math Script.
1.2 HISTORICAL DEVELOPMENT OF LabVIEW:

LabVIEW which was originally released for the Apple Macintosh in 1986 and the graphical language used at the heart called “G”. The idea at that time was to revolutionize the measurement and automation industry, and the technology brought about the virtual instrument - helping engineers and scientists to customize measurement systems to suit their needs. It was not until 1992 that LabVIEW was available for platforms other than the Macintosh and since then it has undergone many revisions. LabVIEW is commonly used on a variety of platforms including Microsoft Windows, various versions of UNIX, Linux, and Mac OS X.

LabVIEW made its appearance as an interpreted package on the Apple Macintosh in 1986. LabVIEW 2 was released in 1990, and was a compiled package as against an interpreted package. The first reasonably stable graphical environment (Windows 3.0) made its appearance only in 1990.

In 1992, LabVIEW for Windows and LabVIEW for Sun based on the new portable architecture were introduced. LabVIEW 3 arrived in 1993 for Macintosh, Windows and Sun operating systems. LabVIEW 3 programs written on one platform could run on another. In 1999, LabVIEW became available for the Linux platform as well. LabVIEW 4, released in 1996, featured a more customizable development environment so that users could create their own workspace to match their industry, experience level and development habits. In addition, LabVIEW 4 added high-powered editing and debugging tools for advanced instrumentation systems, as well as OLE-based connectivity and distributed execution tools.
Networking support on smaller systems was first introduced in Version 5. LabVIEW 5 and 5.1 (in 1999) continued to improve on the development tool by introducing a built-in web server, a dynamic programming and control framework (VI server), integration with ActiveX, and easy sharing of data over the Internet with a protocol called Data socket. In 2000, LabVIEW 6 (sometimes called 6i) provided both an easy and intuitive programming interface. In 2001, LabVIEW 6.1 introduced event-oriented programming, remote Web control of LabVIEW.

Version 7 has expanded the horizon to make it simpler for the inexperienced user by providing various Express utilities and Assistants. LabVIEW has two closely related products-Bridge VIEW and LabVIEW RT (for real time applications). Bridge VIEW can also be called LabVIEW industrial. The RT module of LabVIEW was originally designed to support distributed computing and real-time applications. LabVIEW RT is a hardware and software combination that allows you to take portions of your LabVIEW code and download them to be executed on a separate controller board with its own real-time operating system. In certain respects LabVIEW RT and the development of FPGA (Field programmable Gate Array) support in LabVIEW overlap. It is possible to program FPGA modules to carry out many tasks which may be assigned to RT systems.

National Instruments released LabVIEW 8.5, the latest version of the graphical system design platform for test, control and embedded system development. LabVIEW 8.5 simplifies multicore as well as FPGA – based application development with its intuitive parallel dataflow language. NI labVIEW 8.6 includes the platform installation DVDs, Block Diagram Cleanup, Quick Drop and Web services creation. In addition to these added tools to the LabVIEW development system, we can explore key new features
in the LabVIEW modules and toolkits, including industrial function blocks in the LabVIEW Real-time Module IP.

With the enhanced releases of LabVIEW graphical programming software, system-level engineers as well as domain experts with little to no embedded expertise can accurately work with systems of increased complexity and scale, thereby, drastically reducing the time from concept to prototype.

<table>
<thead>
<tr>
<th>LabVIEW Version</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>LabVIEW 1.0 (for Macintosh)</td>
<td>October 1986</td>
</tr>
<tr>
<td>LabVIEW 2.0</td>
<td>January 1990</td>
</tr>
<tr>
<td>LabVIEW 2.5 (first release for Sun &amp; Windows)</td>
<td>August 1992</td>
</tr>
<tr>
<td>LabVIEW 3.0 (Multiplatform)</td>
<td>July 1993</td>
</tr>
<tr>
<td>LabVIEW 3.0.1 (first release for Windows NT)</td>
<td>1994</td>
</tr>
<tr>
<td>LabVIEW 3.1</td>
<td>1994</td>
</tr>
<tr>
<td>LabVIEW 3.1.1 (first release with &quot;application builder&quot; capability)</td>
<td>1995</td>
</tr>
<tr>
<td>LabVIEW 4.0</td>
<td>April 1996</td>
</tr>
<tr>
<td>LabVIEW 4.1</td>
<td>1997</td>
</tr>
<tr>
<td>Product Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>LabVIEW 5.0</td>
<td>February 1998</td>
</tr>
<tr>
<td>LabVIEW RT (Real Time)</td>
<td>May 1999</td>
</tr>
<tr>
<td>LabVIEW 6.0 (6i)</td>
<td>7/26/2000</td>
</tr>
<tr>
<td>LabVIEW 6.1</td>
<td>12/4/2001</td>
</tr>
<tr>
<td>LabVIEW 7.0 (Express)</td>
<td>April 2003</td>
</tr>
<tr>
<td>LabVIEW PDA module first released</td>
<td>May 2003</td>
</tr>
<tr>
<td>LabVIEW FPGA module first released</td>
<td>June 2003</td>
</tr>
<tr>
<td>LabVIEW 7.1</td>
<td>2004</td>
</tr>
<tr>
<td>LabVIEW Embedded module first released</td>
<td>May 2005</td>
</tr>
<tr>
<td>LabVIEW 8.0</td>
<td>September 2005</td>
</tr>
<tr>
<td>LabVIEW 8.20 (native Object Oriented Programming)</td>
<td>August 2006</td>
</tr>
<tr>
<td>LabVIEW 8.2.1</td>
<td>2/21/2007</td>
</tr>
<tr>
<td>LabVIEW 8.5</td>
<td>2007</td>
</tr>
<tr>
<td>LabVIEW 8.6</td>
<td>7/24/2008</td>
</tr>
<tr>
<td>Product</td>
<td>Release Date</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>LabVIEW 8.6.1</td>
<td>12/10/2008</td>
</tr>
<tr>
<td>LabVIEW 2009 (32 and 64-bit)</td>
<td>8/4/2009</td>
</tr>
<tr>
<td>LabVIEW 2009 SP2</td>
<td>1/8/2010</td>
</tr>
<tr>
<td>LabVIEW 2010</td>
<td>8/4/2010</td>
</tr>
<tr>
<td>LabVIEW 2010 f2</td>
<td>9/16/2010</td>
</tr>
<tr>
<td>LabVIEW 2010 SP1</td>
<td>5/17/2011</td>
</tr>
<tr>
<td>LabVIEW for LEGO MINDSTORMS (2010 SP1)</td>
<td>August 2011</td>
</tr>
<tr>
<td>LabVIEW 2011</td>
<td>6/22/2011</td>
</tr>
<tr>
<td>LabVIEW 2011 SP1</td>
<td>3/1/2012</td>
</tr>
<tr>
<td>LabVIEW 2012</td>
<td>August 2012</td>
</tr>
</tbody>
</table>

Table 1.1: History of LabVIEW
1.3 LabVIEW PROGRAMMING PROCESS AND TECHNIQUES:

LabVIEW, also referred to as G, is different from most other general-purpose programming languages in two major ways. First, programming is performed by wiring together graphical icons on a diagram, which is then compiled directly to machine code so the computer processors can execute it. The second main differentiator is that G code developed with LabVIEW executes according to the rules of data flow instead of the more traditional procedural approach found in most text-based programming languages like C and C++. Dataflow languages like G promote data as the main concept behind any program. Dataflow execution is data-driven, or data-dependent. The flow of data between nodes in the program, not sequential lines of text, determines the execution order. LabVIEW provides a variety of features and tools ranging from interactive assistants to configurable user-defined interfaces; it is differentiated by its graphical, general-purpose programming language (known as G) along with an associated integrated compiler, a linker, and debugging tools. LabVIEW programs/subroutines are called virtual instruments (VIs). Each VI has three components: a block diagram, a front panel and a connector pane. The last is used to represent the VI in the block diagrams of other, calling VIs. Controls and indicators on the front panel allow an operator to input data into or extract data from a running virtual instrument. However, the front panel can also serve as a programmatic interface. Thus a virtual instrument can either be run as a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the given node through the connector panel. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.
1.3.1 Block Diagram:

The block diagram is a pictorial representation of the program code. It corresponds to the lines of text found in text based programming language, and is, the actual executable code. The block diagram is built by wiring together objects that perform specific functions.

The block diagram contains this graphical source code. Front panel objects appear as terminals on the block diagram. The components of block diagram belong to one of three classes of objects.

Nodes : program execution elements.

Terminals : parts through which data passes between block diagram

Wires : data paths between terminals.

Nodes : nodes are analogous to statements, functions and subroutines in text based Programming languages. There are three types of nodes functions, sub VI Nodes and Structures.

Functions are the built in nodes for performing elementary operations such as File I/O, adding numbers or string formatting. Functions are the fundamental operating element of a block diagram.

Sub VI node are the VI that a user can design and later call from the diagram of another VI.

Structures are the icons that are used to control the program flow (such as FOR loop and WHILE loop).
Fig 1.2 : LabVIEW Block Diagram
Fig 1.3: LabVIEW Front Panel
1.3.2 Front Panel:

The front panel is the interactive interface of the VI. It contains the graphical interface with which a user interacts. The front panel can house various graphical objects ranging from simple buttons to complex graphs.

1.3.3 Connector Pane:

The connector pane is a set of terminals that correspond to the controls and indicators of that VI, similar to the parameter list of a function call in text-based programming languages. The connector pane defines the inputs and outputs a user can wire to the VI so that he can use it as a subVI. A connector pane \cite{5} receives data at its input terminals and passes the data to the block diagram code through the front panel controls and receives the results at its output terminals from the front panel indicators.

The user can designate which inputs and outputs are required, recommended, and optional to prevent users from forgetting to wire subVI terminals.

For terminal inputs, required means that the block diagram on which a user placed the subVI will be broken if he does not wire the required inputs. For terminal inputs and outputs, recommended and optional means is that wiring of the terminals is sometimes optional and sometimes compulsory. If user does not wire the terminals, the block diagram on which he placed the subVI does not generate any warnings.

Inputs and outputs of VIs in vi.lib are already marked as Required, Recommended, or optional. LabVIEW sets inputs and outputs of VIs a user creates to Recommended by default. A user sets a terminal setting to required execution only if the VI has the input or output to run properly.
Fig 1.4: Connector Pane
To designate which inputs and outputs are required or recommended, and optional, right-click a terminal on the connector pane and select This Connection Is from the shortcut menu. A user can configure LabVIEW to set all new terminals to configure on the connector pane required instead of recommended. Select Tools » Options » Front Panel and place a checkmark in the Connector pane terminals default to required checkbox. This option also applies to connections made using the Edit » Create SubVI menu item.

1.3.4 Controls and Indicators:

Controls and indicators are the interactive input and output terminals of the VI, respectively. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates. Objects for the Front Panel are dragged over from the Controls Palette (this palette can be opened from the View menu as long as the Front Panel window is active and it disappears when the Block Diagram is selected). Controls supply the VI with data and Indicators display data from the VI. When a control or indicator is dragged on to the Front Panel, an object corresponding to it automatically appears on the Block Diagram. Objects for the Block Diagram are dragged over from the Functions palette (which becomes available when the Block Diagram is active). Once all the required objects have been dragged on to the Front Panel and Block Diagram, wiring them together. Each object on the Block Diagram has one or more input or output terminals, and it is good programming practice to make sure each is wired properly, even if the program runs with a terminal left unwired. Wiring is done by
dragging and clicking the wiring tool from terminal to terminal. Dragging the tool creates
the wire, while clicking tacks down the wire at that point and lets a user to draw the tool
in another direction. Once the wiring is completed, the VI is saved in the usual way (with
.vi as the extension).

One complication is that there are many different types of data, and each object
produces and expects certain ones. LabVIEW uses labels and colors to make this easier,
but it is a part of the programming that needs careful attention. The types you will be
using at first are 16 bit signed integers (I16), single and double precision real floating
point numbers (SGL and DBL), Boolean states (e.g., true or false), and I/O name controls
(Channel Types, which users have LabVIEW generate for them). A user must remember
to save the developed VI.

1.4 LabVIEW FOR MEDICAL INSTRUMENT AND DEVELOPMENT:

Producing safe, high-quality devices for patient care is a primary concern,
but reduced development time is also critical. By tightly integrating hardware, software,
validation, and reporting tools, LabVIEW, by National Instruments consists of the most
popular and powerful tools available and provides the best solution for rapidly
developing and testing complex medical devices. Acquisition of the signal can be
handled through built-in procedures, and LabVIEW’s ability to easily create a user
interface is second to none. Analysis of the signal received can be readily performed by
readymade procedures which can be obtained from National Instruments by the more
industrious, and intermediate programmer. In all, for an intermediate programmer,
LabVIEW provides a clear and easy-to-use method for obtaining, analyzing, and
displaying the signal desired.
National Instruments hardware and software products are used in a broad variety of biomedical applications such as biophysics cellular physiology, tissue bath recordings, exercise physiology, biomechanics, and lab automation. This series of Webcasts explores a variety of topics around the challenges in designing and validating these types of medical applications.

The medical instrument which is developed using LabVIEW is called the virtual medicine instrument \[^6\]. This instrument makes the best of active computer resource, matches with oneness of constructive instrument hardware and proprietary software, and achieves all the functions of the traditional medical instrument as well as some fantastic specific functions that cannot be carry out on the traditional medicine instrument.

A virtual medical instrument adds a set of software and hardware on the general-purpose computer, while a user operates this computer as if operating special traditional medicine equipment designed by himself. The appearance of the virtual medical instrument technology breaks through the mode, that traditional medicine instrument is defined by manufacturer and a user cannot change it. Given the adequate space to exert ability and imagination, a user can design his own medical instrument system to satisfy various applications.

The virtual medical instrument is the testing platform based on the computer soft and hardware, it can replace the traditional medical instrument, such as electrocardiograph, electroencephalograph, polygraph \[^7\], and it also can be used in auto control and data processing system, to build up special medical instrument system. A virtual medical instrument is composed of a computer, application software and medical instrument hardware. No matter which kind it is, the virtual medical instrument system is
sure to be made up of these three aspects that the application software and the medical instrument hardware are installed into a platform based on a notebook computer, a table computer or a workstation (even a PDA). The virtual medical instrument puts the computer hardware resources and medical instrument hardware together organically through software, and then links the powerful count processing ability of the computer with the measuring and controlling ability of the medical instrument hardware, thereby the cost and volume of the medical instrument hardware is reduced and the data can be displayed.

The biomedical signals acquired from the human body are frequently very small, often in the millivolts range, and each has its own processing needs. For instance, electroencephalography signals are in the microvolt range and have many frequency components. Obviously these biomedical signals require processing before they can be analyzed. LabVIEW contains the tools, from Fast Fourier transforms (FFTs) to digital filters\[^8\], to do the job. In order to do frequency analysis, a complex signal must first be broken down into its frequency components. One of the most common ways to do this is with an FFT. In order to facilitate this type of analysis, LabVIEW comes with built in FFTs that make the process of component separation quickly and easily. In addition, biomedical signals, being extremely small in amplitude, are prone to being overwhelmed by noise. To combat this, it is necessary to run the acquired signal through a set of filters\[^9\]. This can be done externally to the computer using standard hardware filtering devices. However, after the signal reaches the computer, it can still contain noise. Another way to solve the noise problem is to use the digital filters provided with LabVIEW. LabVIEW offers the choice of Butterworth, Bessel, Chebyshev, and Chebyshev II digital filters\[^10\].
With a few adjustments, these filters can be configured for almost any design that is needed.

Analysis of the biomedical signals can also be easily done in LabVIEW. The graphical nature of LabVIEW allows even the beginning programmer to easily write programs to analyze data without having to worry about the syntax problems associated with most programming languages. Once again, the user just drags subprograms onto the diagram screen to set up a working program. These modules can be configured to do any kind of manipulation required with just a few clicks of a mouse button. In addition to the analysis modules that the programmer writes, National Instruments offers a wide variety of analysis libraries, including ones for biomedical signals.

LabVIEW can be an efficient alternative to stand alone medical instrument, and as the speed and reliability of the system increases, there will be more and more of virtual medical instrument systems available.

The components necessary for a LabVIEW based acquisition and analysis system are inexpensive and readily available. The experience necessary to program this type of system in LabVIEW is small, and the number of libraries available from National Instruments is growing.

1.5 PURPOSE AND SCOPE OF THE PRESENT STUDY:

A virtual instrument is software packed graphically to have the look and feel of a physical instrument. The screen looks like the front panel of an instrument with knob, slides, and switches. LabVIEW provides a library of controls and indicators for a user to create and customize the look of the front panel. LabVIEW programs are composed of sets of graphical functional blocks with interconnected wiring.
Virtual Instrumentation is a new step in modern instrumentation. LabVIEW based platform provides great flexibility to design various kinds of instruments. In medical instrumentation, to monitor the patient more number of instruments are required. This occupies a lot of space and critical electrical connections in the environment. This will lead a complex environment around the patient. To eliminate this kind of atmosphere, the labVIEW based instrumentation is the right option. The analog front end instrumentation remains same with medical equipment but whereas signal conditioning(processing), data storage, data display, data processing, setting clinical standard, closed loop monitoring, effective alarming systems, and user friendly operation like many features can be incorporated with LabVIEW.

Basically LabVIEW requires a computer as a medium. The LabVIEW is effectively implemented to integrate many medical types of equipment with VI’s. Because of the user controls that are available in technology through the software, the flexibility of virtual instrumentation is unmatched by traditional instrumentation. The modular, hierarchical programming environment of virtual instrumentation is inherently reusable and reconfigurable. These advantages of virtual instrumentation are the basis for increased performance and reduced costs. Due to the great flexibility of LabVIEW programming user based application can also be expanded. The LabVIEW system is capable of presenting the virtual monitoring like ECG, Pulseoximeter, and EEG can be developed very easily. So the purpose of design and development of all the above instruments with LabVIEW is to eliminate the complex instruments around the patient.

Virtual instrumentation brings many advantages over “conventional” instrumentation. Generally, virtual instruments are more flexible and scalable as they can
be easily reconfigured in software. Moreover, standard interfaces allow seamless integration of virtual instruments in distributed system. Virtual instrumentation significantly decreases the price of an instrument based on mass-produced general-purpose computing platforms and dedicated sensors for a given application.

Virtual instrumentation is rapidly entering biomedical field. Many of the general virtual instrumentation concepts may be directly used in biomedical measurements, but biomedical measurements have their own specific features that must be taken into account. Therefore, although it is widely used in many biomedical solutions, the virtual instrumentation is not common in critical clinical applications. Keeping in mind complexity of biomedical phenomena, bringing virtual instrumentation closer to critical biomedical applications will require more testing and more extensive list of developed solutions.

Medical instrument which goes along with the development of electronic measurement instrument has gone through from the simulated instrument, intelligent instrument to virtual instrument. The advantage of the virtual medical instrument technology lies in the user-defined system, agile functions and easy construction, so it is used widely in many fields, especially in the scientific research, exploitation, medical measurement, medical detection, and medical signal process and so on. It has powerful functions to carry out all of the functions of the traditional medical instruments from data waveform display, logic analysis, spectral analysis, signal generate to medical image display\textsuperscript{[11]}. Matching with the medical sensor and software, it can measure various medical parameters such as body temperature, pulse wave, blood pressure; It has not only operation agility, but also complete graph interface, so new users can master the
operation rules without any training and it is easy to construct the automatic measurement system with high speed.
REFERENCE


2. Virtual Instrumentation - Changing the Face of Design, Measurement and Automation
   By Jayaram Pillai, Managing Director, National Instruments, India


4. Virtual Instrumentation Using LabVIEW- Jovitha Jerome


8. E.M.O'Brien, LabVIEW usage as part of the biomedical engineering senior design experience, 0-7803-7612-9/02 2002IE, p. 2599.

