Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 5, May 2013, pg.184 – 190

RESEARCH ARTICLE

DESIGNING A VIRTUAL MACHINE FOR IDENTIFICATION OF CARDIAC ARRHYTHMIAS USING LAB VIEW

B. Subha[1], Subha.S.V[2], M.Anitha[3], M.Eniya[4], M.Gaayathri[5]

[1] Lecturer, Department of Biomedical Engineering, PSNA College of Engineering and Technology, Dindigul, India
[2] Assistant Professor, Department of Biomedical Engineering, PSNA College of Engineering and Technology, Dindigul, India
[3] Final year, Department of Biomedical Engineering, PSNA College of Engineering and Technology, Dindigul, India
[4] Final year, Department of Biomedical Engineering, PSNA College of Engineering and Technology, Dindigul, India
[5] Final year, Department of Biomedical Engineering, PSNA College of Engineering and Technology, Dindigul, India

[1] subhabme@gmail.com, [2] subhaanusv1987@gmail.com

Abstract— Various projects have been proposed for acquiring and analysis of ECG signals using different software. To be in advance this work focuses not only on acquiring and analysis of ECG signal but also on identification of cardiac arrhythmias. This would bridge the gap between medical physicians and engineers. Our project is carried out with the help of LabVIEW software (version 8.2). This model work collects the waveform from the affected person, analyzed and particular disease is identified. Initially, The ECG signals are picked from the patient body using the electrodes (surface electrodes). The signal is obtained using ECG amplifier for better amplification which is then fed to PC through NI ELVIS DAQ. Here the waveform in analog form is converted to digital form and is analyzed for detecting the peak intervals of the ECG signal acquired, based on which the identification of cardiac arrhythmias is done by sending them to the loops containing the conditions for cardiac arrhythmias. Based on the results obtained from the analysis of the ECG signal and comparison with the loop conditions, the cardiac arrhythmias are identified and displayed instantly.

Key Terms: - Electrocardiogram; Laboratory Virtual Instrumentation Engineers Workbench; Data Acquisition; Educational Laboratory Virtual Instrumentation Suite

I. INTRODUCTION

Heart diseases have emerged as the number one killer in both urban and rural areas of the country. About 25 percent of death in age group of 25 – 69 occurs because of heart diseases. In urban areas about 32.8 percent deaths occurs because of heart ailments, while this percent in rural areas is 22.9. According to the above statistics diagnosis of cardiac disorders plays a vital role for the survival of human race. Real time monitoring plays an important role in biomedical engineering, particularly in ECG, EMG, EEG, etc. Personal computers have become a standard platform for the needs of various measurement and test, standardization, performance and low cost. Use of PC in so called personal and virtual an instrumentation development enables realization of a new generation of superior devices. With their performance, this is becoming ever higher and with increasing number of software applications they are widely accepted as an essential tool on desk of engineer. At present, there are various technologies for identification of cardiac disorders using hardware components which are time consuming and not at affordable rate. In this procedure, after analyzing the ECG waveform picked from the

patient, it is to be further consulted by a doctor to diagnose the particular disease. Hence we are intended to develop a virtual machine that acquires the ECG signals from the patient through electrodes, analyze them and help in identification of cardiac diseases. Unlike the existing system, this promotes us to perform the diagnosis for more number of patients simultaneously even in the absence of physicians which helps us to recognize the particular disorder.

II. MATERIALS AND METHODS

2.1 VIRTUAL INSTRUMENTATION

Virtual instrumentation is the foundation for the modern laboratory. A virtual instrument consists of a computer, software, and modulator hardware; all combined and configured to emulate the function of traditional hardware instrumentation. It is also called as LabVIEW program. Because their functionality is software-defined by the user, virtual instruments are extremely flexible, powerful and cost-effective.

2.2 FRONT PANEL

It contains a knob for selecting the number of measurements per average, a control for selecting the measurement type, a digital indicator to display the output value, and a stop button. An elaborate front panel can be created without much effort to serve as the user interface for an application.

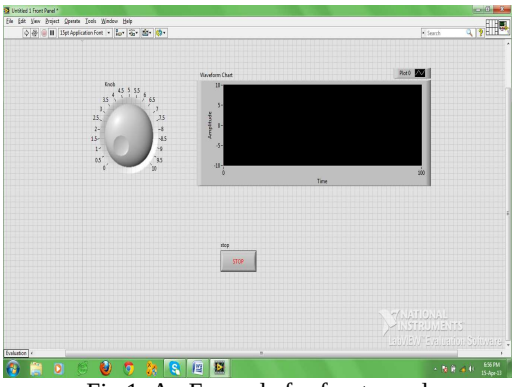

Fig.1: An Example for front panel

2.3 BLOCK DIAGRAM PANEL

The block diagram or source code window holds the graphical source code of LabVIEW VIs. LabVIEW's block diagram corresponds to the lines of text found in a more conventional language like C or BASIC- it is the actual executable code. We can construct the block diagram by writing together objects that perform specific functions. When we place a control or indicator on the front panel, LabVIEW automatically creates a corresponding terminal on the block diagram. By default, we cannot delete a block diagram terminal that belongs to a control or indicator. Although we may try to create a heart's content. The terminal disappears only when we delete its corresponding control or indicator on the front panel.

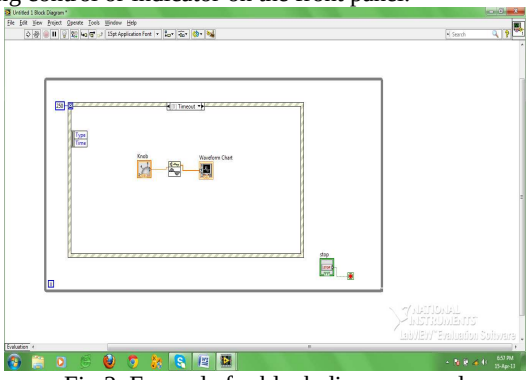

Fig.2: Example for block diagram panel

The outer rectangle structure represents a while loop, and the inner one is a case structure. The icon in the center is a VI, or subroutine, that takes the number measurements per average as input and returns the frequency value as the output. The orange line, or wire, represents the data being passed from control in the VI. The

selection for the measurement type is connected, or wired to the case statement to determine which case is executed. When the stop button is pressed, the while loop stops execution.

LabVIEW is not an interpreted language; it is compiled behind the scenes by LabVIEW's execution engine. Similar to JAVA, the VI's are compiled into an executable code that LabVIEW's execution engine processes during runtime. Every time a change is made to a VI, LabVIEW constructs a wire table for the VI. This wire table identifies elements in the block diagram that have inputs needed for that element to run. Elements can be primitive operators such as addition, or more complex such as a subVI. If LabVIEW successfully constructs all the wire tables, we are presented a solid arrow indicating that the VIs can be executed. If the wire table cannot be created, then a broken arrow is presented for the VIs with a problem, and also for each VI loaded in a memory that requires that VI for execution. LabVIEW run in several subsystems compiles diagrams while we write them. This allows programmers to write code and test it without needing to wait for a compiling process, and programmers do not need to worry about execution speed because the language is not interpreted.

The wire diagrams that are connected do not define an order in which elements are executed. This is an important concept for the programmers to understand. LabVIEW is a data-flow based language, which means that elements will be executed in a somewhat arbitrary order. LabVIEW does not guarantee which order a series of elements is executed in if they are not dependent on each other. A process called arbitrary interleaving is used to determine the order elements are executed in. We may force an order of execution by requiring that elements require output from another element before execution. This is a fairly common practice, and most programmers do not recognize that they are forcing the order of execution. When programming, it will become obvious that some operations must take place before others can. It is the programmer's responsibility to provide a mechanism to force the order of execution in the code design.

2.4 EXPERIMENTAL SETUP

The following block represents the skeletal diagram for acquisition of ECG signal for which disorder is to be determined.

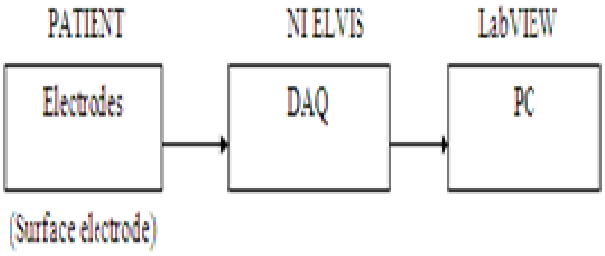

Fig.3: Implementation Method

2.5 SURFACE ELECTRODES

Limb electrodes generally suffer from what is known as motion artifacts caused due to the relative motion at the interface between the metal electrodes and the adjacent layer of electrode jelly. The interface can be stabilized by the use of floating electrodes in which the metal electrode does not make direct contact with the skin. The electrode consists of a light-weight metallic screen or plate held away from the subject by a flat washer which is connected to the skin. Floating electrodes can be recharged, i.e. the jelly in the electrodes can be replenished if desired.

Patten has described spray-on chest electrodes where a conducting spot is developed on the skin by spraying a film of conducting adhesive. Connection with the instrument is established with silver-plated copper wires fixed in the conducting adhesive. The type of electrodes are extremely light-weight and not to make use of electrode jelly. This makes them ideal for use in monitoring the ECG of exercising subjects and aero plane pilots as they give rise to motion artifacts. The contact impedance shown by these electrodes is of the order of 50 KΩ.

Completely flexible electrodes for the long-term monitoring of ECG during space flight are reported by Sandler. These electrodes were made of silver-impregnated silastic rubber and were found to be comfortable to wear. They were also evaluated for use during exercise or prolonged monitoring as may be necessary in an intensive care or coronary care unit.

2.6 DATA ACQUISITION

Data acquisition or DAQ is simply the process of measuring a real-world signal, such as a voltage, and bringing that information into the computer for processing, analysis, storage or other data manipulation.

Physical phenomena represent the real-world signals we are trying to measure, such as speed, temperature, humidity, pressure, flow, start-stop, pH, radioactivity, light intensity and so on.

We can use sensors (sometimes called transducers) to evaluate the physical phenomena and produce electrical signals proportionately. Other examples of sensors include strain gauges, flow meters, and pressure transducers, which measure displacement in a material due to stress, rate of flow, and pressure, respectively. In each case, the electrical signal produced by the sensor is directly related to the physical phenomenon it monitors.

LabVIEW can command DAQ devices to read analog input signals (A/D conversion), generate analog output signals (D/A conversion),read and write digital signals, and manipulate the on-board counters for frequency measurement, pulse generation, quadrature encoder measurements, and so on, to interface with the transducers. In the case of analog input, the voltage data from the sensor goes into the plug-in DAQ devices in the compute, which sends the data into computer memory for storage, processing, or other manipulations. Signal conditioning modules "condition" the electrical signals generated by transducers so that are in a form that the DAQ devices can accept.

To acquire data in our lab using the virtual instrumentation approach, we will need a DAQ device, a computer configured with LabVIEW and DAQ driver software, and some method of connecting our transducer signal to the DAQ device such as a connector block, breadboard, cable, or wire. We may also need signal conditioning equipment, depending on the specifications of our application.

2.7 THE PERSONAL COMPUTER

The computer we use for our DAQ system can drastically affect the maximum speeds at which we can continuously acquire data. As computers continuously improve, our DAQ system can take advantage of the computers enhanced capabilities, including improved real-time processing, the ability to use complex video graphics, and higher streaming-to-disk throughput. As PCs speed continuously increases, DAQ system speed increases as a result. PCs are capable of programmed I/O and interrupt data transfers. Direct Memory Access (DMA) transfers increase the system throughput by using dedicated hardware to transfer data directly into system memory. Using this method, the processor is not burdened with moving data and is therefore free to engage in more complex processing tasks.

2.8 EXECUTING VIs

A LabVIEW program is executed by pressing the arrow or the Run button located in the palette along the top of the window. While the VI is executing, the Run button changes to a black color. All of the items in the palette are displayed during execution of a VI. As we proceed to the right along the palette, we will find the continuous Run, Stop, and Pause buttons. VIs are normally run from the front panel; however, they can also be executed from the block diagram. This allows the programmer to run the program and utilize some of the other tools that are available for debugging purposes.

If the Run button appears as a broken arrow, this indicates that the LabVIEW program or VI cannot compile because of programming errors. When all of the errors are fixed, the broken Run button will be substituted by the regular Run button. LabVIEW has successfully compiled the diagram. While editing or creating a VI, we may notice that the palette displays the broken Run button. If we continue to see this after editing is completed, press the button to determine the cause of the error. An Error List window will appear displaying all of the errors that must be fixed before the VI can compile.

2.9 FILTERS

Filtering is one of the most commonly use signal processing techniques. Signal conditioning systems can filter unwanted signals or noise from the signal we are measuring. Use the noise filter or low rate, or slowly changing, signals, such as temperature, to eliminate higher frequency signals that can reduce signal accuracy. A common use of a filter is to eliminate the noise from a 50 or 60Hz AC power line. Low pass filter of 4 Hz removes the 50 or 60Hz AC noise from signals sampled at low rates. A low pass filter eliminates all signal frequency components above the cut off frequency. Many signal conditioning modules have low pass filters that have software-selectable cut off frequencies from 10Hz to 25 KHz.

2.10 AMPLITUDE & LEVEL MEASUREMENTS

Table.1. PORTS OF AMPLITUDE & LEVEL MEASUREMENTS

PARAMETER	DESCRIPTION
Amplitude Measurements	Contains the following options: ➢ DC – Acquires a DC measurement of signals. ➢ Maximum Peak – Measures the most positive peak in signals ➢ Peak to peak – Measures the most positive peak to the most negative peak in signals
Input Signal	Displays the input signal. If you wire data to the Express VI and run it, Input Signal displays real data. If you close and reopen the Express VI, Input Signal displays sample data until you run the Express VI again.

2.11 PEAK DETECTOR

In order to measure the peaks/valleys LabVIEW uses the Peak Detector from the signal operation found in the Function palette.

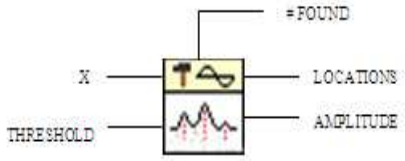

Fig.4: Peak detector

Table.2. PORTS OF PEAK DETECTOR

PARAMETER	DESCRIPTION
# found	# found is the number of peaks/valleys found in the current block of data. # found is the size of the arrays Locations, Amplitudes, and 2nd Derivatives.
X	X is the array of input values that represents the signal to be analyzed. The data can be a single array or consecutive blocks of data. Consecutive blocks of data are useful for large data arrays or for real time processing. Notice that in real time processing, peaks/valleys are not detected until approximately width/2 data points past the peak or valley.
Threshold	Threshold instructs the VI to ignore peaks and valleys that are too small. The VI ignores peaks if the fitted amplitude is less than threshold. The VI ignores valleys if the fitted trough is greater than threshold.
Locations	Locations contain the index locations of all peaks or valleys detected in the current block of data. Because the peak detection algorithm uses a quadratic fit to find the peaks, it actually interpolates between the data points. Therefore, the indexes are not integers. In other words, the peaks found are not necessarily actual points in the input data but may be at fractions of an index and at amplitudes not found in the input array.
Amplitude	Amplitudes contain the amplitudes of peaks or valleys found in the current block of data.

2.12 WAVEFORM CHARTS

A plot is simply a graphical display of X versus Y values. Often, Y values in a plot represent the data value, while X values represent time. The waveform chart, located in the Graph sub palette of the Controls palette, is a special numeric indicator that can display one or more plots of data. The waveform chart has three update modes – strip chart mode, scope chart mode and sweep chart mode.

2.13 SCRIPT NODE

Script nodes are used to execute text – based math scripts in LabVIEW. LabVIEW provides support for certain script nodes that invoke script servers provide by other parties to process scripts, such as the MATLAB Script server. Use input and output variables on math script nodes and other script nodes to pass values between LabVIEW and a math script node, or other script node. The structure of an expression in a script determines whether a variable is an input or an output. We can use the MathScript node to create scripts using the

LabVIEW MathScript syntax or other text-based syntaxes, edit your scripts that we create or load, and process our MathScripts and other text-based scripts by invoking the MathScript RT module engine. MathScript nodes can process many of our text-based scripts created in a MATLAB or compatible environment. However, because the MathScript RT module engine does not support all functions supported by the MATLAB software, some functions in our existing scripts might not be supported. We can implement such functions with a Formula node or another script node.

2.14 BOOLEAN

LabVIEW provides a myriad of switches, LEDs, and buttons for our Boolean controls and indicators. The Boolean palette holds various functions of performing logical operations. All of the functions require Boolean inputs, except for the conversion functions. Boolean data can have one of two states: true or false. A Boolean constant is also provided on this palette. The comparison functions simply compare data values and return a Boolean as a result. We can compare numeric, Boolean, string, array, cluster and character values using these functions.

III. RESULTS AND DISCUSSION

By using the above mentioned functions and performing the task required, we ended up with the following result:

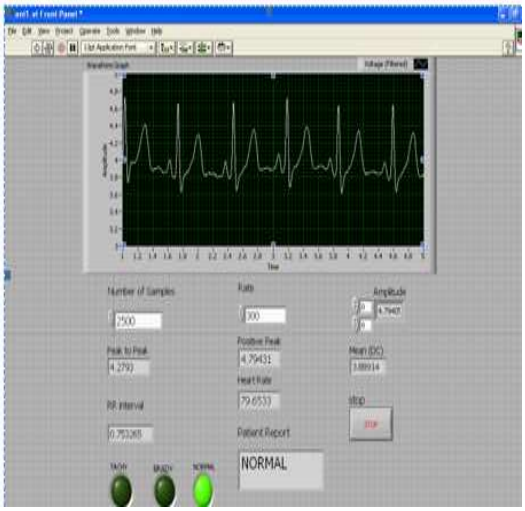

Fig.5. Obtained result

IV. CONCLUSION AND FUTURE ENHANCEMENT

The above developed virtual machine sounds good for regular adults. But this may not suit for sportsman who does exercise more often. This is because the normal heart rate for such person is found to be higher than the regular adult even in the resting state.

In future, ECG Amplifier used in the proposed system, which is operated by using the hardware component, can be eliminated by developing it as an inbuilt tool.

This system can be enhanced similarly for other cardiac disorders, by measuring the various intervals like P-Q, S-T, P-R intervals, and QRS complex. Other parameters such as Temperature, Blood Pressure, and pH of biofluids can also be determined. Apart from ECG, various other Bio-Signals such as EEG, EMG, EOG, etc, can be extracted by using the proposed method. It can be used for more than one patient simultaneously.

Using the Read to measurement file present in the function palette, the readings that are obtained as real – time signal from the patient can even be stored as a database for future retrieval.

REFERENCES

[1] Amit Kumar, Lillie Dewan, Mukhtiar Singh (November 2006) 'Real Time Monitoring System for ECG Signal Using Virtual Instrumentation"- WSEAS transactions on biology and biomedicine, Volume 3.

[2] Channappa Bhyri, Kalpana.V, Hamde, S.T. and Waghmare L.M. (July 2009) "Estimation of ECG features using LabVIEW" - TECHNIA – International Journal of Computing Science and

Communication Technologies, Volume.2, No.1.

[3] Barreto, A. Correia, S. Miranda, J. and Silva, L. (July 2009) 'LabVIEW and MATLAB for ECG acquisition, filtering and processing' - 3rd International Conference on Integrity, Reliability and Failure, Porto/Portugal.

[4] Ahammad, T. Haque, A. N. M. M. Islam, M. K. Khondokar, M. R. H. (June 2012) 'Study and Analysis of ECG Signal Using MATLAB & LABVIEW as Effective Tools' - International Journal of Computer and Electrical Engineering, Volume. 4, No. 3.

[5] Gowthaam, P.R. Indumathi, S. John Kennedy, S. Raaj Kamal, N.R. Vijayakumar, M. (April, 2012) 'Identification of Cardiac Diseases and Remote Monitoring Using LabVIEW' - International Conference on Computing and Control Engineering.

