# Remote Monitoring And Diagnosis of a Mechatronic System

IONUT RESCEANU, MARIUS NICULESCU, NICU GEORGE BIZDOACA, CRISTINA PANA
Department of Mechatronics
University of Craiova
Address Bvd. Decebal Nr. 107
ROMANIA

*Abstract:* The real time evolution of a system is analyzed and the differences between the mathematical models describing the system are providing a feedback indicating a possible fault in the monitored process. All the monitoring activity was done remotely via word wide web using Data Socket Server technology for transmitting and receiving data through the network. The measurement and control are accomplished and integrated by using a computerized data acquisition system and a comprehensive virtual instrument, developed using the LabVIEW application software. In addition, this system allows for easy modification and enhancement of virtual (software) instrument by modification of the software program.

*Key-Words:* - **Remote Monitoring, Data Socket, Virtual Instrumentation, Real-Time, Internet, Control**

## 1 Introduction

**1.Introduction**

Having the capability to control and observe experiments from a remote location has several benefits, including the ability to track and to assist in solving a problem that might arise. The best way to remotely monitor an experiment is via the web with software that is platform independent.

The National Instruments LabVIEW graphical development environment is well-known for data acquisition and instrument control applications. NI LabVIEW includes comprehensive built-in signal processing and analysis capabilities that you can apply to these applications, but also to more general technical computing tasks such as algorithm engineering, simulation, and control.

In the quest to develop the remote capability several options were considered:

The data socket (DS) is a unified end-user application programming interface (API). It consists of a data socket server (DSS) and an API. The advantages of this route are the reliability of the program, the nice Web interface (all the Labview graphical user interfaces (GUI) configuration tools available) and the speed of the data update. The main disadvantage is that in order to be able to view the experiment the web user has to download and install the Run-Time Engine version of Labview (from NI) plus a small Labview executable program written at MIBL, or download the entire application as a self installing program from the MIBL site. The size of the download of the whole application is rather large (> 10 MB), making the process a little difficult for slow computer. In addition, the program needs to be recompiled on a Macintosh machine to have it available for Macintosh users.

Web monitoring with the Labview Web Server was implemented after Labview 6.0 was released. Now version 7.1 is out and it has even better remote access features. Starting with the 6.0 version, any application could be published as a web file rather easy using a builtin Web Server. The Labview Web Server approach has very nice security and user monitoring features, but also some disadvantages. Like the previous method, it would require the user to download and install the Labview Run-Time Engine before the first run. The Labview Web Server comes with a built-in one-client capability, limiting the number of users. Additional licenses can be purchased, but cost could be an issue with this strategy.

But more important, this alternative is not available for Macintosh computer users. However, the simplicity of the approach and the requirement of very little extra programming besides the main application make this option really appealing.

With Labview one can connect to other applications and share data through ActiveX, the Web, DLLs, shared libraries, TCP/IP etc. In addition to Labview, the other software package required was Appletview (Java based), available from Nacimiento Inc. Appletview assists in publishing the Labview instrumentation on the web and makes the web browsers capable of connecting to the applications running on a local computer. The built-in server mediates the communication between the local and the remote computer, as the instruments' outputs are

made available to a remote client. This is accomplished by streaming the data from the local instruments to a Java applet running in a Web browser. The software enables the programmer to configure and shape the user interface according to the needs. The most important advantages of the Labview/Appletview combination are: the end user has access to DAQ systems located anywhere, is language independent and operating system (OS) independent. Using this protocol, data can be sent via theWeb between different machines running Labview.

## 2 Virtual Instrumentation

Labview is a powerful software package that can be used to design user interfaces to interactively control the systems in a graphical environment taking advantage of many programming tools and by creating virtual instruments that mirror real ones allowing remote users to collaborate in real time [1]. This software package is produced by National Instruments and has world wide acceptance and presence at this time. It can be found in most of the research labs and in very many research and development facilities of private businesses. It provides a quick and easy access to instrumentation control and a very large database of drivers for DAQ cards, various computer interfaces (GPIB, serial etc) and instrument drivers. Each Labview program consists of a Front Panel (FP) interface, that contains the controls and data fields, and a Block Diagram (BD) where the real programming flow takes place. Although it is a graphical programming language, its versatility allows the easy incorporation of modules written in other languages (C, Basic etc)

A typical data acquisition (DAQ) system may consist of transducers, signal conditioning hardware, plug-in DAQ boards, and LabVIEW® application software, see Figure 1. Examples may include monitoring and controlling complete measurement or process system, etc. The plug-in DAQ board enables computerized measurement and control of real world analog input-signals (AI, like with an oscilloscope) and generation of analog output-signals (AO, like with a function generator), as well as digital input/output (I/O) signals. The **LabVIEW** (**V**irtual **I**nstrument **E**ngineering **W**orkbench), a graphical programming language by National Instruments, is especially suitable for developing automated instrumentation systems using the PC plug-in data acquisition (DAQ) boards. It may be effectively used for engineering data acquisition, analysis, and presentation. The main LabVIEW advantage over the classical text/script based programming is its graphical interface where the user naturally builds a program by connecting (wiring) built-in component icons, i.e. by drawing the program's algorithm. Another LabVIEW advantage is for use to build computerized or virtual instrumentation since its input/output interface mimics the real-instrument front-panels. It is also full-fledged programming application that integrates advanced data analysis and presentations.

## 3 HOW IT WORKS

In order to make the connection to the Web, it is necessary to build an application with Labview that would collect the data points of interest from the main DAQ program. The data objects to be sent out are only allowed to be of a certain type (string, integer32, single and Boolean type).

The LabVIEW Data Acquisition VIs are located on the Data Acquisition palette and the DAQmx – Data Acqusition palette. The Data Acquisition palette contains the traditional NI-DAQ VIs. The DAQmx - Data Acquisition palette contains the VIs for NI-DAQmx. The DAQmx - Data Acquisition palette contains all of the VIs necessary to perform analog I/O, digital I/O, and counter/timer operations. The VIs are organized so that the most common operations can be performed using the VIs. You can configure a task to perform a very specific function by using the Property Nodes in the palette. Many applications that do not require advanced timing and synchronization can be performed by using the DAQ Assistant Express VI.

### 3.1 Remote Operation Using DataSocket Technology

We implemented remote operation using National Instruments DataSocket communication protocol. The DataSocket Server application runs on the local robot control PC. The DataSocket Server accepts published data from applications and broadcasts data to subscribing applications. All data passed between the remote and local VIs is channeled through the DataSocket Server.

### 3.2 Implementing DataSocket Communication

DataSocket communication relies on the DataSocket Server, which may or may not run on the same computer as client applications. With default permissions on the DataSocket Server, we have

access only to the computer it runs on. To achieve communication outside that computer, we can use the DataSocket Server Manager to configure the Server to accept published data from the computer or host running a client application, and to allow the host that is running a subscribing client application to read data. To use remote communication outside the local area network (LAN), as in this application, it is necessary to run the DataSocket Server on a computer with a static IP address.

DataSocket is a programming tool that enables you to read, write, and share data between applications and/or different data sources and targets across the network. DataSocket can access data in local files and data on HTTP and FTP servers. If you use general purpose file I/O functions, TCP/IP functions, and FTP/HTTP requests to transfer data, you must write separate code for each protocol. With DataSocket you need very little or no code to transmit and receive data over the Internet.

DataSocket is available both for LabVIEW and Measurement Studio and runs on Linux, Macintosh, and Windows. The only caveat is the DataSocket server is still Windows based.

DSTP is an application-layer protocol for transferring measurement data to and from a dstp server called a DataSocket server.

It is implemented on top of TCP, and hence provides connection-oriented communication between the server and the client.

In other words, clients maintain a session during communication with the server. During the session the server keeps track of client-connection information.

Clients providing the measurement data to a DataSocket server are referred to as publishers or writers.

Clients consuming the measurement data provided by the publishers are referred to as subscribers or readers.

System participating in a dstp data exchange usually consists of three components, the DataSocket server, a publisher, and subscribers.

A publisher acquires data from a local or remote data acquisition device and sends it to the server.

The server may be located on the same machine (known as the localhost) or remotely on the Internet.

Subscribers who have an interest in the published data can subscribe to receive the data from the server.

Complex applications may require more than one publisher or more than one server.

DataSocket allows to send data over a network to and from variety of software platforms without worrying about the low-level implementation details.

The "DataSocket" server is an external program that manages TCP/IP connections , handling different data types ( integers , floats , strings , and Booleans, as well as arrays of these)

DataSocket has two main pieces that works together: -The "DataSocket" server; -The "DataSocket" API for clients.

The DataSocket server is standalone application that handles client connections.

The DataSocket Server is a lightweight, stand-alone component with which programs using the DataSocket API can broadcast live measurement data at high rates across the Internet to several remote clients concurrently

DataSocket Server simplifies network TCP programming by automatically managing connections to clients.

Broadcasting data with the DataSocket Server requires three "actors" – a publisher, the DataSocket Server, and a subscriber. A publishing application uses the DataSocket API to write data to the server.

A subscribing application uses the DataSocket API to read data from the server.

Both the publishing and the subscribing applications are "clients" of the DataSocket Server.

The three actors can reside on the same machine, but more often the three actors run on different machines.

The ability to run the DataSocket server on another machine improves performance and provides security by isolating network connections from your measurement application.

The DataSocket API is implemented as an ActiveX control, a LabWindows/CVI C library, and a set of LabVIEW VIs, so you can use it in any programming environment.

The DataSocket API automatically converts the user's measurement data into a stream of bytes that is sent across the network.

The subscribing DataSocket application automatically converts the stream of bytes back into its original form.

Learning the DataSocket API is simple. It consists of four basic actions (open, read, write, and close) that are similar to standard file I/O calls. You can use the same DataSocket API in your programs to read data from:

Data items on HTTP servers ,Data items on FTP servers ,Local files ,Data items on DSTP servers

# 4 The concepts of model based fault detection and isolation

## 4.1 Residue generation

A traditional fault detection method is that of limit verification, comparing process variables with actual limits. When the variables exceed the limits, a fault situation is indicated. Although simple, this method has a major disadvantage: process variables may change with different operating states. Therefore this method depends on operating state of the process. On the other hand, residual signals are quantities representing the inconsequence between the variables of the actual system and mathematical model. They do not depend on operating state of the system and they only answer to faults, which makes us prefer limit verification method.

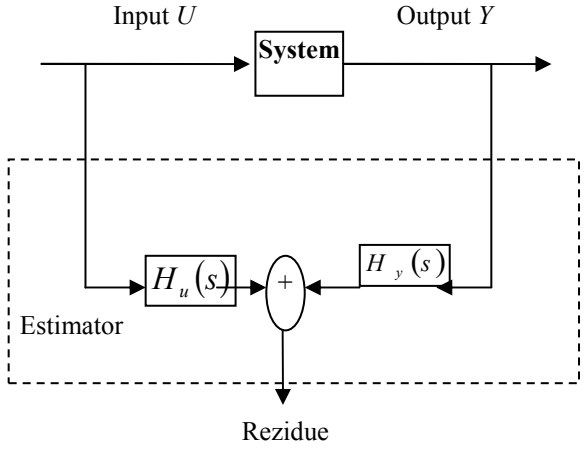


Fig. 1. General structure of a residue generator

The simplest approaching for residue generation is using the system duplicate, i.e. a simulator that is identical with the original is realized. The residue is the difference between the simulated and the actual output. The disadvantage of this method is that the stability of the simulator can not be guaranteed when monitored system is instable. A direct extension to simulator based residue generation is that of placing the simulator on an output estimator, as you can see in the following figure:

This structure is mathematically expressed as:

$$r(s) = \begin{bmatrix} H_u(s) & H_y(s) \end{bmatrix} \begin{bmatrix} U(s) \\ Y(s) \end{bmatrix} = \\ = H_u(s)U(s) + H_y(s)Y(s) \qquad (1)$$

where $H_u(s)$ and $H_y(s)$ represent transfer matrixes which are feasible using stable linear systems. These can be seen as state estimators that use input and output, respectively.

In order to make the residue to become zero for the faultless case:

$$H_u(s)U(s) + H_y(s)Y(s) = H_u(s)U(s) + \\ H_y(s)G(s)U(s) = 0 \qquad (2)$$

the next condition must be respected:

$$H_u(s) + H_y(s)G_u(s) = 0 \qquad (3)$$

Eq. (2) is a generalized representation of all residue generators. Residue generator design requires choosing transfer matrixes $H_u(s)$ şi $H_y(s)$. Usual residues are generated using analytical approaches like design observers, using parameters estimation techniques or parity equations based on analytical redundancy.

For realizing fault detection the residue generator must have fault detectability and isolability properties.

# 5 Application

Our application uses DataSocket Transport Protocol (DSTP) for reading and writing data within network. When using DSTP, DataSocket Server communicates with any virtual instrument (VI); this way virtual instrument can publish or subscribe data. We can identify different types of data through label names, which appeals URL address.

The server mediates the communication between local and portable computer; this way the outputs of the instrument are available for a portable client. Final user can access DAQ systems, no matter their location. Using this protocol one can use web for transmitting data between different computers running Labview.

Equipment used for monitoring the system: NI USB 6008 acquisition board, for capturing system's signals; as monitoring equipment: Labview; as data transmission instrument: WebServer -DataSocket.

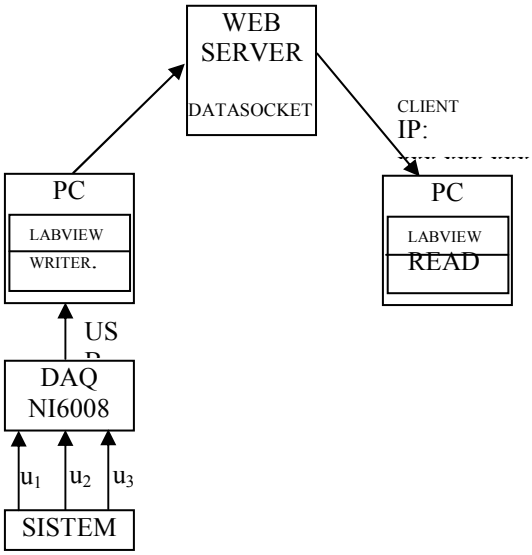


Fig. 2 Operational diagram of the application

A mathematical model is attached to the system. The main idea is to permanently compare mathematical model M2 with real process P. We also introduce a partial mathematical model M1 of the process, which takes as input function an intermediate voltage generated by the real process, in order to allow fault localization.
r(t) vector contains the differences – the residue between real system P and mathematical model M. If a certain preset threshold value is exceeded, an alarm corresponding to the system point where the difference appeared is generated.
We first monitor the application without considering any eventual fault.
Reader application receives information from DataSocket Server. Input voltage of the system, $u_1$, is visualised, and also intermediate voltage $u_2$ and output voltage $u_3$ aquisitioned from phisical system.
$u_3$ voltage, resulted from mathematical model is visualised in simulation chart – $u_3$, and residual vector r(t), resulted from the difference between real voltage $u_3$ and the voltage provided by the mathematical model is visualised in error chart $u_3$-r(t).
Fault operation: the resistance $R_2$ has been modified and adjusted to a superior value comparing to $R_1$ – A3 alarm turns on when a residue that is superior a 0.3V threshold appears – this threshold is considered maximum limit that is allowed for this system.
A step input signal is applied. The evolution of the physical system and mathematical model when this stimulus is applied is obvious. For short periods of time – we talk about milliseconds – an exceeding of 0.3V imposed threshold for residual vector can be noticed. The cause is a delay in mathematical model calculus correlated with sampling rate.

In the following figures results obtained by DS Writer.vi. şi DS Reader.vi are presented. The first one takes over the signals from acquisition board, i.e. the three voltages $u_1$, $u_2$ şi $u_3$ and transmits them to DataSocketServer. The second one connects itself to server from where it receives aquisitioned data.

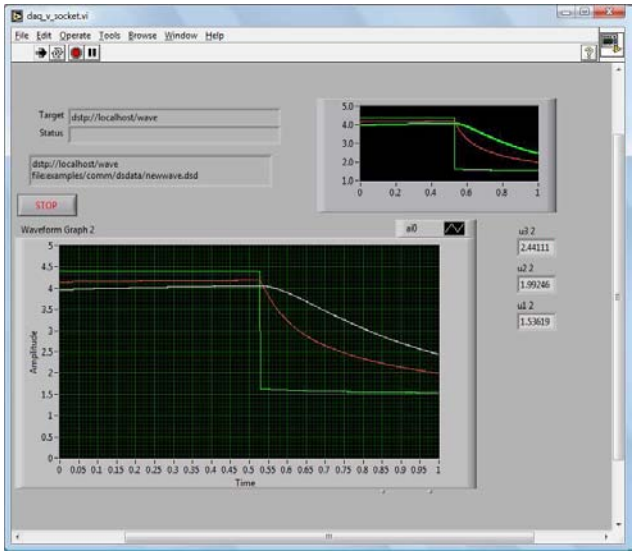


Fig.3 *Writer.vi* Labview front panel for monitoring the system

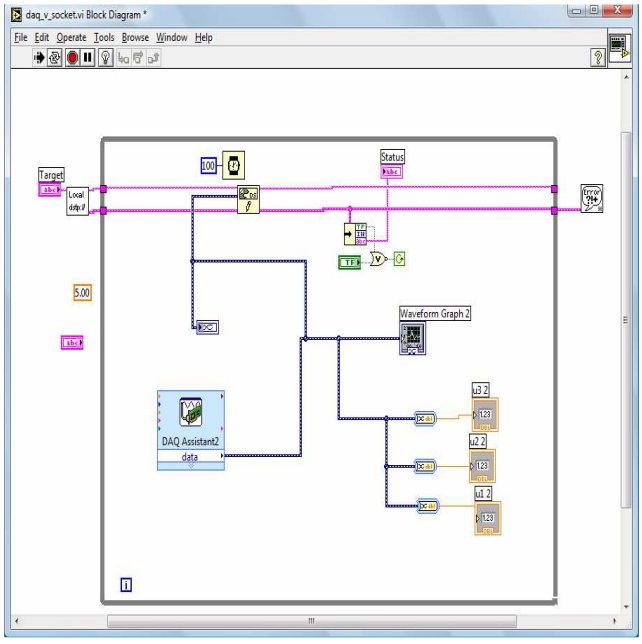


Fig.4 *Writer.vi* Labview front panel for transmiting the data through the Data Socket Sever

Reader.vi is the client application and is receiving themonitored data through the Data Socket Sever.
The real time data is compared with the output of the mathematical model attached to the system.

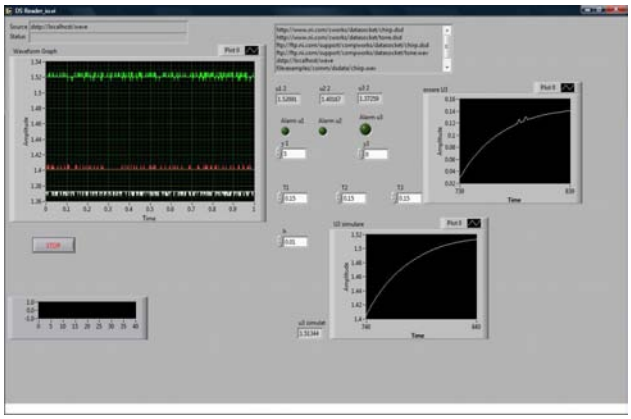

Fig.5 Reader.vi Labview front panel for monitoring the system

## 5 Conclusion

One of the objectives of this project is to utilize the latest powerful, yet inexpensive, technological developments: sensors and transducers, data acquisition and control integrated boards, computers and application software, for research. The designed, computerized measurement and data acquisition system, accomplishes the following objectives: acquire measured data with high speed and accuracy; interactively process and analyze measured data for immediate use or future post-processing; provide interactive and accurate feed-back process control and interactively displays the raw/measured and processed/analyzed data in graphical and/or numerical forms.

In addition, such a system allows for easy modification and enhancement of the so called "virtual (software) instrument" by modification of the software program. It is important to emphasize that functionality and quality of a virtual instrument is practically limited by our creativity.

We studied the real time evolution of the monitored system and analyzed the differences between the mathematical model describing the system. All the monitoring activity was done remotely via word wide web using Data Socket Server technology for transmitting and receiving data through the network.

*References:*

[1]M. Kostic, Data Acquisition and Control for an Innovative Thermal Conductivity Apparatus Using LabVIEW Virtual Instrument, *Laboratory Robotics and Automation Journal,* Vol.10, No.2, pp.107-111, Wiley, 1998

[2]Wells, L and J. Travis, "LabVIEW for Everyone," Prentice Hall PTR, Upper Saddle River, NJ, 1997.

[3] http://sine.ni.com/ -"Customer Solutions Remote Control of a Rhino Robot Using LabVIEW 6i and DataSocket Technology"

[4]DataSocket Overview http://zone.ni.com/devzone/conceptd.nsf/webmain/14F3B2BC11811DC186256A9D0068B2D1?opendocument&node=DZ52047_US

[5]Dave BAKER: Remote Panels in LabVIEW 7.1 – Distributed Application Development LabVIEW Technical Resource

[6]Frank, P. M. (1990) *Fault Diagnosis in Dynamic System Using Analytical and Knowledge Based Redundancy - A survey and some new results*, Automatica, vol.26, no.3, pag.459 - 474.

[7]Ginn B, Bruel & Kjaer (2000) Practical applications of intelligent test systems. Proceedings of the Seventh International Congress on Sound and Vibration, Vol. VI, Germany, pp 3221–3228

[8]Tse P, Peng YH, Yam R (2001) Wavelet analysis and envelope detection for rolling element bearing fault diagnosis – their effectiveness