Proceedings of the XI DINAME, 28th February-4th March, 2005 - Ouro Preto - MG - Brazil
Edited by D.A. Rade and V. Steffen Jr. © 2005 - ABCM. All rights reserved.

COUPLING SENSING HARDWARE WITH DATA INTERROGATION SOFTWARE FOR STRUCTURAL HEALTH MONITORING

Charles R. Farrar
Los Alamos National Laboratory
MS T-001
Los Alamos, NM 87545 USA
farrar@lanl.gov

David W. Allen
Los Alamos National Laboratory
MS T-001
Los Alamos, NM 87545 USA
dallen@lanl.gov

Steven Ball
Motorola Labs
Los Alamos Research Park
4200 West Jemez Road, Suite 300
Los Alamos, NM 87544 USA
hamster@snurkle.net

Michael P. Masquelier
Motorola Labs
Los Alamos Research Park
4200 West Jemez Road, Suite 300
Los Alamos, NM 87544 USA
Mike.Masquelier@motorola.com

Gyuhae Park
Los Alamos National Laboratory
MS T-001
Los Alamos, NM 87545 *USA*
gpark@lanl.gov

Abstract. *The process of implementing a damage detection strategy for aerospace, civil and mechanical engineering infrastructure is referred to as structural health monitoring (SHM). The authors' approach is to address the SHM problem in the context of a statistical pattern recognition paradigm. In this paradigm, the process can be broken down into four parts: (1) Operational Evaluation, (2) Data Acquisition and Cleansing, (3) Feature Extraction and Data Compression, and (4) Statistical Model Development for Feature Discrimination. These processes must be implemented through hardware or software and, in general, some combination of these two approaches will be used. This paper will discuss each portion of the SHM process with particular emphasis on the coupling of a general purpose data interrogation software package for structural health monitoring (DIAMOND II) with a modular wireless sensing and processing platform that is being jointly developed with Motorola Labs. More specifically, this paper will address the need to take an integrated hardware/software approach to developing SHM solutions..*

Keywords: *sensing, data Interrogation, software, system integration*

1. Introduction

The process of implementing a damage detection strategy for aerospace, civil and mechanical engineering infrastructure is referred to as structural health monitoring (SHM). Here damage is defined as changes to the material and/or geometric properties of these systems, including changes to the boundary conditions and system connectivity, which adversely affect the system's current or future performance. Our approach is to address the SHM problem in the context of a statistical pattern recognition paradigm (Farrar, Doebling and Nix, 2001). In this paradigm, the process can be broken down into four parts: (1) Operational Evaluation, (2) Data Acquisition, (3) Feature Extraction, and (4) Statistical Model Development for Feature Discrimination. When one attempts to apply this paradigm to data from "real-world" structures, it quickly becomes apparent that data cleansing, normalization, fusion and compression, which can be implemented with either hardware or software, are inherent in Parts 2-4 of this paradigm. A more detailed overview of the SHM process can be found in (Worden and Dulieu-Barton, 2004). These processes must be implemented through hardware or software and, in general, some combination of these two approaches will be used.. The authors believe that all approaches to SHM, as well as all traditional non-destructive evaluation procedures (e.g ultrasonic inspection, acoustic emissions, active thermography) can be cast in the context of this statistical pattern recognition paradigm. To date, however, there have been a limited number of technology development efforts that have approached the SHM problem in an integrated manner where all portions of the paradigm are addressed for a particular application. Instead, most efforts have focused exclusively either on the sensing technology or the data interrogation algorithms. (Doebling, et al., 1996; Sohn, et al., 2004a).

This paper will present a summary of an integrated hardware/software SHM system that is being jointly developed by Los Alamos National Laboratory and Motorola Labs to address the shortcoming described above. First, each portion of the SHM statistical pattern recognition paradigm will be briefly summarized. This discussion will be followed by a summary of a coupled hardware/software system that is intended to address most portions of the statistical pattern recognition paradigm for SHM.

2. The Statstical Pattern Recogntion Paradigm for Structural Health Monirtoing

2.1 Operational Evaluation

The first step in the development of a SHM system is referred to as operational evaluation. Operational evaluation attempts to answer four questions regarding the implementation of a damage detection capability: 1.) What are the life-safety and/or economic justification for performing the SHM? 2.) How is damage defined for the system being investigated and, for multiple damage possibilities, which cases are of the most concern? 3.) What are the conditions, both operational and environmental, under which the system to be monitored functions? and 4.) What are the limitations on acquiring data in the operational environment? Operational evaluation begins to set the limitations on what will be monitored and how the monitoring will be accomplished. This evaluation starts to tailor the damage detection process to features that are unique to the system being monitored and tries to take advantage of the unique characteristics.

2.2. Data Acquisition

The data acquisition portion of the SHM process involves selecting the excitation methods, the sensor types, number and locations, and the data acquisition/storage/processing/transmittal hardware. The actual implementation of this portion of the SHM process will be application specific. Economic considerations will play a major role in making decisions about the type and extent of the data acquisition system that can be deployed. For real-world applications the ruggedness and long-term stability of the data acquisition system will also be a concern.

A fundamental premise regarding data acquisition and sensing is that these systems do not measure damage. Rather, they measure the response of a system to it operational and environmental loading. Depending on the sensing technology deployed and the type of damage to be identified, the sensor reading may be more or less directly correlated to the presence and location of damage. Data interrogation procedures are the necessary components of a SHM system that convert the sensor data into information about the structural condition. Furthermore, to achieve successful SHM the data acquisition system will have to be developed in conjunction with these data interrogation procedures.

2.3. Data Normalization

As it applies to SHM, data normalization is the process of separating changes in sensor reading caused by damage from those caused by varying operational and environmental conditions. (Farrar, Sohn, Worden, 2001) Because data can be measured under varying conditions, the ability to normalize the data becomes very important to the damage detection process. One of the most common procedures is to normalize the measured responses by the measured inputs. When environmental or operational variability is an issue, the need can arise to normalize the data in some temporal fashion to facilitate the comparison of data measured at similar times of an environmental or operational cycle. This normalization may require additional types of measurements (e.g. temperature) to be made. Sources of variability in the data acquisition process and with the system being monitored need to be identified and minimized to the extent possible. In general, not all sources of variability can be eliminated. Therefore, it is necessary to make the appropriate measurements such that these sources can be statistically quantified. Variability can arise from changing environmental and test conditions, changes in the data reduction process, and unit-to-unit inconsistencies.

2.4. Data Cleansing

Data cleansing is the process of selectively choosing data to pass on to, or reject from, the feature selection process. The data cleansing process is usually based on knowledge gained by individuals directly involved with the data acquisition. As an example, an inspection of the test setup may reveal that a sensor was loosely mounted and, hence, based on the judgment of the individuals performing the measurement, this set of data or the data from that particular sensor may be selectively deleted from the feature selection process. Signal processing techniques such a filtering and re-sampling can also be thought of as data cleansing procedures.

2.5. Feature Extraction

A damage-sensitive feature is some quantity extracted from the measured system response data that indicates the presence of damage in a structure. Identifying features that can accurately distinguish a damaged structure from an undamaged one is the focus of most SHM technical literature (Doebling, et al, 1996, Sohn, et al., 2004a). Fundamentally, the feature extraction process is based on fitting some model, either physics-based or data-based, to the measured system response data. The parameters of these models or the predictive errors associated with these models then become the damage-sensitive features. An alternate approach is to identify features that directly compare the sensor waveforms or spectra of these waveforms. Many of the features identified for impedance-based and wave propagation-based SHM studies fall into this category (Park, et al., 2004, and Sohn, et al., 2004b).

One of the most common methods of feature extraction is based on correlating observations of measured quantities with the first-hand observations of the degrading system. Another method of developing features for damage detection is to apply engineered flaws, similar to ones expected in actual operating conditions, to systems and develop an initial understanding of the parameters that are sensitive to the expected damage. The flawed system can also be used to validate that the diagnostic measurements are sensitive enough to distinguish between features identified from the undamaged and damaged system. The use of analytical tools such as experimentally-validated finite element models can be a great asset in this process. In many cases the analytical tools are used to perform numerical experiments where the flaws are introduced through computer simulation. Damage accumulation testing, during which significant structural components of the system under study are subjected to a realistic degradation, can also be used to identify appropriate features. This process may involve induced-damage testing, fatigue testing, corrosion growth, temperature cycling, etc. to accumulate certain types of damage in an accelerated fashion. Insight into the appropriate features can be gained from several sources and is usually the result of information obtained from some combination of these sources.

2.6. Data Fusion

Data fusion is the process of combining information from multiple sensors in an effort to enhance the fidelity of the damage detection process. Inherent in many feature selection processes is the fusing of data from multiple sensors and condensation of these data. Common examples of data fusion include the extraction of mode shapes from sensor arrays and the averaging of spectral quantities to remove noise from the measurements. Additional data fusion procedures focus on establishing other types of correlations (or quantifying loss of correlation) between different sensors in an effort to identify the presence and location of damage.

2.7. Data Compression

The operational implementation and diagnostic measurement technologies needed to perform SHM produce more data than traditional uses of dynamic response information. A condensation of the data is advantageous and necessary when comparisons of many feature sets obtained over the lifetime of the structure are envisioned. Also, because data will be acquired from a structure over an extended period of time and in an operational environment, robust data reduction techniques must be developed to retain feature sensitivity to the structural changes of interest in the presence of environmental and operational variability. To further aid in the extraction and recording of quality data needed to perform SHM, the statistical significance of the features should be characterized and used in the compression process.

2.8. Statistical Model Development

The portion of the SHM process that has received the least attention in the technical literature is the development of statistical models for discrimination between features from the undamaged and damaged structures. Statistical model development is concerned with the implementation of the algorithms that operate on the extracted features to quantify the damage state of the structure. The algorithms used in statistical model development usually fall into three categories. When data are available from both the undamaged and damaged structure, the statistical pattern recognition algorithms fall into the general classification referred to as *supervised learning*. *Group classification* and *regression analysis* are categories of supervised learning algorithms. *Unsupervised learning* refers to algorithms that are applied to data not containing examples from the damaged structure. *Outlier* or *novelty detection* is the primary class of algorithms applied in unsupervised learning applications. All of the algorithms analyze statistical distributions of the features to enhance the damage detection process.

The damage state of a system can be described as a five-step process, along the lines of that proposed by Rytter, 1993, that answers the following questions: 1.) Is there damage in the system (existence), 2.)?Where is the damage in the system (location)?; 3.) What kind of damage is present (type)?; 4.) How severe is the damage (extent)?; and 5.) How much useful life remains (prognosis)?

Answers to these questions in the order presented represent increasing knowledge of the damage state. When applied in an unsupervised learning mode, statistical models are typically used to answer questions regarding the existence and location of damage. When applied in a supervised learning mode and coupled with analytical models, the statistical procedures can be used to better determine the type of damage, the extent of damage and remaining useful life of the structure. The statistical models are also used to minimize false indications of damage. False indications of damage falls into two categories: (1) *False-positive* damage indication (indication of damage when none is present), and (2) *False-negative* damage indication (no indication of damage when damage is present). Errors of the first type are undesirable as they will cause unnecessary downtime and consequent loss of revenue as well as loss of confidence in the monitoring system. More importantly, there are clear safety issues if misclassifications of the second type occur. Many pattern recognition algorithms allow one to weight one type of error above the other, this weighting may be one of the factors decided at the operational evaluation stage.

3. A Coupled SHM Hardware/Software System

As the SHM field grows and matures, the question of “can a structure’s health be monitored?” is being replaced by “which permutation of sensing technology, data cleansing, data compression, data normalization, feature extraction, and statistical discrimination procedures yields the best results for the problem outlined in the operational evaluation of a structure?” Next, one must ask how a process can be developed, tested and deployed on a “real-world” structure. The authors have attempted to provide an answer to these questions by developing a modular software toolbox for cataloging feature extraction, data normalization, data cleansing, data fusion and statistical discrimination algorithms. This software is referred to as DIAMOND II. DIAMOND II permits the, rapid assembly of SHM data interrogation processes, easy modification of the process as more data is analyzed, and the embedding of these processes in remote monitoring hardware developed by Motorola Labs. The Motorola hardware is referred to by the acronym WiSHM, which stands for Wireless Structural Health Monitoring system. This hardware was specifically designed for structural health monitoring applications. The fundamental design philosophy for both the hardware and the software was to develop a system that can be easily adapted to a variety of SHM applications. To meet this goal, the hardware was designed so that it could interface with a variety of sensors and it could also drive actuators in an effort to facilitate “active” sensing. Processing and memory capabilities were designed around the DIAMOND II software. The software was designed to facilitate the comparison of different SHM processes as it is envisioned that process development will require an iterative approach. The subsequent portions of this paper describe the hardware and software in more detail.

3.1 The DIAMOND II Software

DIAMOND II is a modular software package developed by staff at LANL (Allen, 2004) to interface with an integrated hardware system assembled by Motorola Labs. This software fills a current void in the SHM community by bringing a variety of different SHM algorithms together in a single software package. This software provides an efficient and adaptable set of tools with which to develop the data interrogation portion of the SHM process.

The software described herein is comprised of two pieces. The first is a client to allow graphical construction of data interrogation processes. The second is node software for remote execution of processes on Motorola’s WiSHM. The client software is created around a catalog of data interrogation algorithms compiled over several years of research at LANL. The development of this software required encapsulating the DIAMOND II algorithms into independent interchangeable functions and providing a streamlined mechanism to facilitate the continual expansion of the function catalog as new algorithms are developed. The client software also includes methods for interfacing with the node software over an Internet connection. Once communications are established with Motorola’s WiSHM node, either by Ethernet or wireless communications, the client software can upload a developed process to the node. The node software has the ability to run the processes and return results. When this software is integrated with Motorola’s WiSHM, it can be used to create a distributed SHM network where the individual nodes perform the monitoring function and then send a system state indicator to a centralized monitoring facility.

3.1.1 Client Software

The SHM team at LANL conceived the idea that grew into the Graphical Linking and Assembly of Syntax Structure, (GLASS) software. The concept was for software that would allow a user to assemble statistical pattern recognition functions into a SHM process in “plug and play” manner. The project developed from simple graphical interfaces to a modern piece of software that provides easy user interaction, expandability, and is easy to maintain.

The original LANL toolbox, DIAMOND (Doebling, 1997), is a graphical-user-interface (GUI) driven MATLAB^{TM} toolbox for experimental modal analysis, finite element model updating and damage identification based on changes in modal properties. This toolbox was developed in the mid 1990’s by staff and students in LANL’s Engineering Analysis group. A shift in paradigm from global modal parameter based damage identification (Doebling, et al., 1996) to statistical pattern recognition based SHM (Farrar, Doebling and Nix, 2001), required the development of a new software tool. The DIAMOND II module and GLASS technology have been created to meet this demand. DIAMOND II, like its predecessor, is a collection of functions based in MATLAB^{TM} that are assembled to provide SHM data interrogation tools. These tools can be categorized as “Data Collection,” “Data Cleansing and Normalization,” “Feature Extraction,” and “Statistical Discrimination.” Functions from these categories are assembled to form a SHM process. The DIAMOND II MATLAB^{TM} algorithms have been encapsulated so that each is a stand-alone function with defined inputs and outputs. The functions are also based on a single data structure allowing them to be assembled in a “plug and play” manner.

With this new catalog of plug and play functions, an interface is needed to allow simple assembly of a SHM process. The GLASS client platform facilitates construction of new processes by allowing drag and drop of MATLAB^{TM}, C, or JAVA functions into a workspace. Variable types, values, and descriptions are displayed and dragging output variables from one function to the input variables of another easily links the two functions. Once assembled, a process can be run in its entirety or selected functions can be run as needed. Processes can then be saved and stored for use in the future, executed remotely, or embedded into microprocessors. GLASS is developed in the JAVA programming language to allow for cross-platform compatibility, and to incorporate modular design allowing for future expansion. In GLASS, DIAMOND II is one of several modules containing data interrogation functions. Other modules include a hardware integration module, a utilities module, and an experimental modal analysis module.

3.1.2 Development of GLASS Software Technology

The first GLASS software versions were developed using the MATLAB^{TM} GUI environment. It was quickly realized that a more powerful language was needed to capture the required functionality. JAVA was chosen because of the ability to compile the software independent of a computing platform, ease of development, and JAVA's Object Oriented (OO) language structure. The use of an OO language allows development of reusable objects, decreasing development and revision time.

OO software emulates abstracted objects from the real world. In the case of the GLASS software, the objects to model come from the functions, and the organization of these functions. The following is a bottom up listing of the objects that are emulated in the GLASS software:

Variables: Each variable object is assigned a name and type corresponding to the MATLAB^{TM} workspace. The object also has a value and description with which it is associated. An experimentally measured 1024-point acceleration time history is an example of an array variable.

Functions: Function objects encapsulate the ability to execute MATLAB^{TM}, C or JAVA code. Function objects also contain information pertaining to its description, authorship, purpose, and input and output variables. An algorithm to perform a Fast Fourier Transform (FFT) is an example of a function.

Categories: In the DIAMOND II module, the categories reflect the statistical pattern recognition paradigm for SHM. In GLASS, a tabbed pane represents each category and displays the contained functions. The FFT function might be part of the feature extraction category within DIAMOND II.

Module: A module is the top level of organization in GLASS. DIAMOND II has been developed as a module to allow future expansion and easier assembly of processes. Each module consists of multiple categories. Other modules add functionality in different aspects of structural dynamics such as model validation and uncertainty quantification or experimental modal analysis.

The GlassComponent is an abstract object that contains properties and methods that are common to many of the GLASS objects. Object oriented programming is useful because of the concept of inheritance. The Module, Category, and Function objects all inherit all of the properties and methods of the abstract GlassComponent. This inheritance also allows Modules, Categories, and Functions to be treated interchangeably on an abstract level. For example, the Glass client requesting the name of a GlassComponent does not make any distinction as to whether the component is a Module, Category, or Function.

There is a problem with creating a JAVA interface to MATLAB^{TM}. JAVA can communicate by evaluating strings in the MATLAB^{TM} workspace, but MATLAB^{TM} has no direct way to interact with JAVA objects created outside of the MATLAB^{TM} workspace. Creating the dataHandler object in the MATLAB^{TM} workspace allows the JAVA object access to the MATLAB^{TM} workspace variables. Other JAVA objects outside of the MATLAB^{TM} workspace can then query this JAVA object. This object allows results to be retrieved from the MATLAB^{TM} workspace and the appropriate Variable object to have its value updated.

With the link between the JAVA and MATLAB^{TM} workspaces established, functions can be run, variables retrieved, expressions evaluated, and MATLAB^{TM} commands executed. This method provides an extraordinary tool that allows graphical manipulation of objects via a JAVA interface and subsequent computational execution in MATLAB^{TM}. JAVA provides a far superior user interface than the native MATLAB^{TM} GIU toolbox because of the drag and drop functionality, threading, and advanced user controls such as the process tree.

Once MATLAB^{TM} functions became executable, JAVA classes and C functions were easily encapsulated in the same JAVA function framework. This framework allows functions written in the different languages to be integrated into a single process, sharing variable values and functionality.

With GLASS Technology, there exists the ability to create functions in MATLAB^{TM}, JAVA or C and then categorize, visually assemble, and have them execute in a process. The next section discusses the assembly of a process from individual functions.

3.1.3 Graphically Prototyping Algorithms

In GLASS, functions are categorized as belonging to a Module and a Category. This hierarchy allows a separation of functions by developer, project, or method. For example, the DIAMOND II Module is a collection of functions developed by many people in an effort to use the statistical pattern recognition paradigm to tackle the SHM problem. This module is then broken into the categories representing the steps followed to analyze data using time series analysis.

Another module, hardware integration, contains functions for accessing a data acquisition board to collect data and functions for broadcasting data or results over a network.

GLASS modules can be created, stored, and shared among users. Functions from different modules may also be combined together to form new processes. This modular approach was taken in an effort to reduce the number of functions re-written by individuals when various functions (e.g. importing a specific file type) have already been written. To assemble a new process, functions are selected from the categories and placed in the workspace. Functions can be re-ordered or inserted at anytime. Functions are then linked by their input and output variables in a cascading fashion.

The first step to assembling a process is data collection. When the data (e.g. measured acceleration time histories) are collected live from a DSP board described below in Section 3.2.2, a JAVA class for communicating with the DSP board is used. The function collectDSPdata starts the process. In this function, the number of data points, sampling frequency, and IP address of the hardware are specified.

Next data cleansing is performed. Because the DSP board used for data collection is a custom and experimental board, there is a small transient response at the beginning of all the samples taken. The subset_data function is the next step in the process and is used to truncate the sample, removing the initial transient data.

A damage sensitive feature must be extracted from the time series. In this example, features are based on fitting a time series model to measured acceleration time-histories. The residual error that results when this model is used to subsequently predict future data sets is considered the damage sensitive feature.

Finally, statistical modeling for feature discrimination is obtained using an X-bar control chart. This test is used to determine when there are significant changes in the residual error features. The result of this statistical test is then broadcast by a JAVA function over the network to other clients. Figure 1 shows the GLASS GUI implementation of this process.

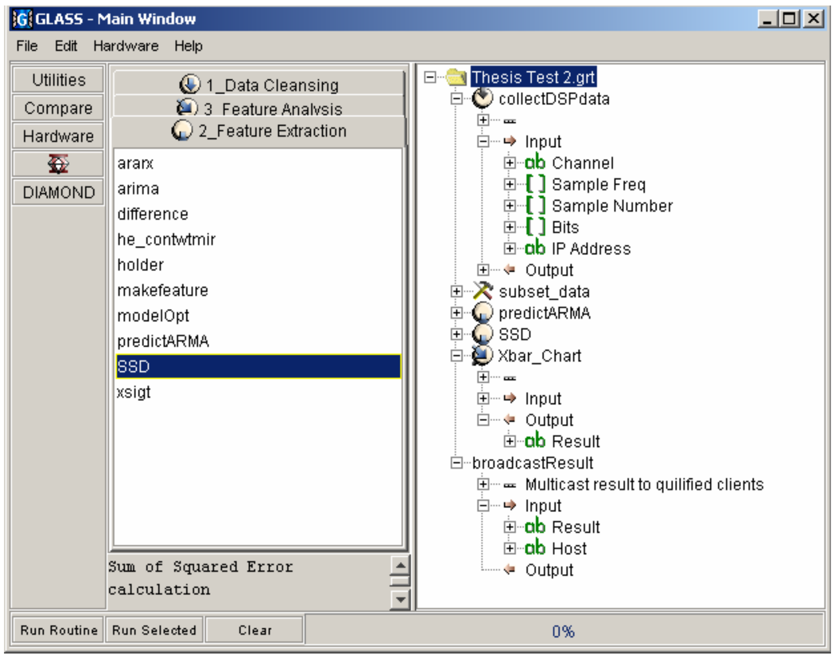

Figure 1. The GLASS GUI showing an algorithm assembled for performing a sequential probability ration test on features extracted from data obtained on a laboratory frame structure.

Once assembled, algorithms can be run in their entirety, or in selected sequences. The idea is that once an algorithm has been assembled, and run once, small changes to parameters should not require the entire sequence to be run again, only affected functions need to be rerun and the final results recalculated. New functions can also be dragged into the workspace and results recalculated to compare and contrast two methods. Once a process has been created, it can be saved for future use or passed on to other individuals. GLASS Technology has been developed to be an open ended and cooperative endeavor that will save time and promote understanding of different approaches to SHM.

3.1.4 GLASS Node Software

Originally, the system design called for the GLASS client software to embed a developed process directly to a digital signal processing (DSP) chip. In researching the process of embedding developed MATLAB^{TM} functions onto a DSP it was found that no simple solution was available. Some tools existed for targeting the MATLAB^{TM} functions for

DSP chips; however, the conversion produces excessive code and linking external math libraries proved to be difficult. The conclusion was made that the functions would need to be re-written in C if a SHM process were to be easily embedded on a DSP directly from GLASS. Because of the time already invested in functions developed in MATLAB^{TM} code, rewriting the functions in C was undesirable. The solution then was to implement the full single board computer (SBC) into Motorola's WiSHM. The SBC solution allows the Linux operating system to directly run MATLAB^{TM} in its entirety without these functions being converted to C. Communication with the GLASS node is accomplished from the GLASS client through a TCP/IP socket. This connection allows the WiSHM and the process development platform to be physically separated but connected through a local area network (LAN), Wireless LAN, or over the Internet.

The GLASS node software is both similar too and yet very different from the GLASS client. Where the user of the GLASS client is an engineer assembling SHM processes, the user of the GLASS node is a GLASS client. Because the user is another piece of software, the GUI portion of the software that allows a human user to graphically communicate with the software needs to be replaced. Pieces of software communicate with each other through a communications protocol where a set of commands and responses that designate actions.

Because the development of the original GLASS client is object oriented (OO) based, many of the objects could be reused for the node software. The OO development also allows the process and function objects to be easily transferred over an Ethernet socket, allowing the node to share and run objects created on the client.

The node software is also designed to reside on an independent piece of hardware, such as the WiSHM or a remote desktop, and continuously run. Once a client has connected to a node, loaded a process, and set it to run, the node will dispatch a thread that will allow the process to run repeatedly until a stop command is sent from a client.

3.1.5 Client integration

The original GLASS client was designed solely for constructing SHM processes on a desktop. Now, however, integration in the client is added to allow communication with the GLASS node software to share constructed processes. A "Hardware" menu (Figure 2) is added to facilitate GUI handling of operations such as opening a connection to the hardware, uploading a constructed process, starting and stopping the process remotely, and finally receiving results broadcasted over the network.

To facilitate the communication between the GLASS client and node, a simple command and response communications protocol is created. An example of an exchange between the client and node to establish a connection and start a process is similar to communication between people: Each step in this communication is a command from the client followed by a response from the node. Without a "+OK" response, the client will abort and ask the user if they would like to try again. This command and response is implemented to prevent deadlock between the two programs. Similarly, if the node receives a command that is not expected or the received object is not valid it will return a "-ERR" response notifying the client that there was a problem with the communication.

Each step in this dialog process is also tied to an action in the GLASS client hardware menu. To send the Open_Connection command, the user selects "Open Connection" in the Hardware menu. Thereby the user controls each step of the process allowing a connection to be established and possibly several versions of the process to be uploaded before the Start_Process command is sent. Confirmation of each command is displayed in the progress bar at the bottom of the screen (Figure 2). While multiple clients can access a single node, the protocol is setup to allow only a single client to connect at one time. This connection mode means that while connected, a client has dedicated and exclusive access to the hardware node.

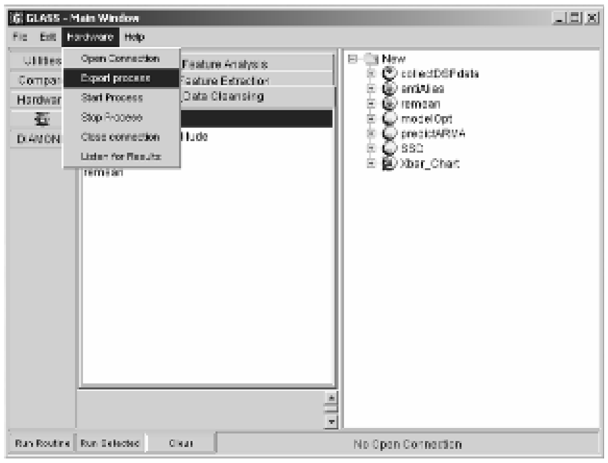

Figure 2. - Screen shot capturing the added Hardware drop down menu. The menu items displayed correspond to commands that can be sent to the node. Notice that confirmation or errors of the commands sent are displayed in the process bar at the bottom.

3.1.6 Communication of results

Once the data are collected and processed a result needs to be returned to a client, mobile device, a display, and/or a central monitoring device. The result is broadcast over a socket opened on a specific port. The difference is that while the above communications were limited to a dedicated connection between a client and the node, the result can be broadcast to multiple recipients simultaneously. Broadcasting of results to multiple recipients is possible using user datagram protocol (UDP) and the JAVA multicasting functionality. This "multicasting" means that several clients can all receive the broadcasted result at the same time and without each making a direct connection to the device. Once a result is calculated, the node will broadcast the result. Any devices that are connected to the appropriate multicast group will receive the result. Results are typically in the form of 1, 0, or -1 representing the state as damaged, undecided, or undamaged respectively. It is up to the receiving program or device to interpret the result. The GLASS client is modified to listen for such broadcast results and to display them as a change in the progress bar at the bottom of the screen. Another simple program can be run on the node or client that listens for a result and then records the result with a time stamp to a text file. A simple program to change the state of a green, yellow or red LED cluster, or to show a result on a pocket PC device can also be created.

3.2 The Wireless Integrated Structural Health Monitoring Hardware

To develop a true integrated SHM system, the data interrogation processes must be transferred to embedded software and hardware that incorporates sensing, processing, and the ability to return a result either locally or remotely. Most off-the-shelf solutions currently available, or in development, (see Allen 2004) have a deficit in processing power that limits the complexity of the software and SHM process that can be implemented.. Also, many integrated systems are inflexible because of tight integration between the embedded software, the hardware, and sensing.

To implement computationally intensive processes such as those developed with the DIAMOND II software, a single board computer (SBC) was selected to provide true processing power in a compact form. Also included in the integrated system is a Motorola Labs developed digital signal processing (DSP) board with six analog to digital converters (ADC) providing the interface to a variety of sensing modalities. Finally, a Motorola wireless network board provides the ability for the system to relay structural information to a central host, across a network, or through local hardware. Each of these hardware parts are built in a modular fashion and loosely coupled through the transmission control protocol (TCP) or UDP, Internet protocols (IP). By implementing a common interface, changing or replacing a single component does not require a redesign of the entire system.

By allowing processes developed in the GLASS client to be downloaded and run directly in the GLASS node software, this system becomes the first hardware solution where new processes can be created and loaded dynamically. This modular nature does not lead to the most power optimized design, but instead achieves a flexible development platform that is used to find the most effective combination of algorithms and hardware for a specific SHM problem. Optimization for power is of secondary concern and will be the focus of follow-on efforts.

The hardware is designed in modular boards. The PC-104 specification is implemented for determining the size and design of each of the hardware boards. Drivers are written for each of the hardware portions that allow communication between the hardware boards through a common protocol over a TCP socket as is seen in Figure 3. This communication setup allows the back-end server to communicate with hardware in an encapsulated form. TCP also allows each hardware portion to be accessed individually over an Ethernet connection for testing while the GLASS node software is running in emulation on a desktop platform.

3.2.1 Single board computer

For the processing center of the WiSHM a SBC (Figure 4) is used to provide powerful processing capabilities. The SBC houses a 133 MHz Pentium™ processor, 256 Mb of RAM, and a Compact Flash (CF) card slot that acts as the hard drive. The SBC can support serial, Ethernet and USB communication with other hardware. Developed by Micro/Sys, the SBC adheres to the PC-104 standard and is easily linked to other hardware via the PC-104 bus, or the previously described connections.

3.2.2 Sensing board

The sensing board utilizes a Motorola DSP56858 chip (www.motorola.com) for reading the ADCs and communicating with the SBC. A DSP is necessary for sampling the ADCs because of the sampling speed requirements. The DSP is an optimized package able to sample and return samples in four seconds for 1024 samples. The SBC would not be fast enough to read the ADCs and buffer results by itself. The six ADCs have a maximum sampling speed of 200 kHz. The DSP board communicates with the SBC, or other command sources, over the serial port through a TCP socket. For example, a command to sample from the DSP board is sent to port 5255 from the CPU, this command is then received on port 5255 (See Fig. 3) and relayed to the serial port and the command is then received by the DSP. The TCP socket is implemented to remove dependence on the serial port. In the future, if the DSP board were to implement the PC-104 bus, the only interfaces that need to be rewritten would be the TCP socket-serial port interface, not the complex code that actually sends the commands.

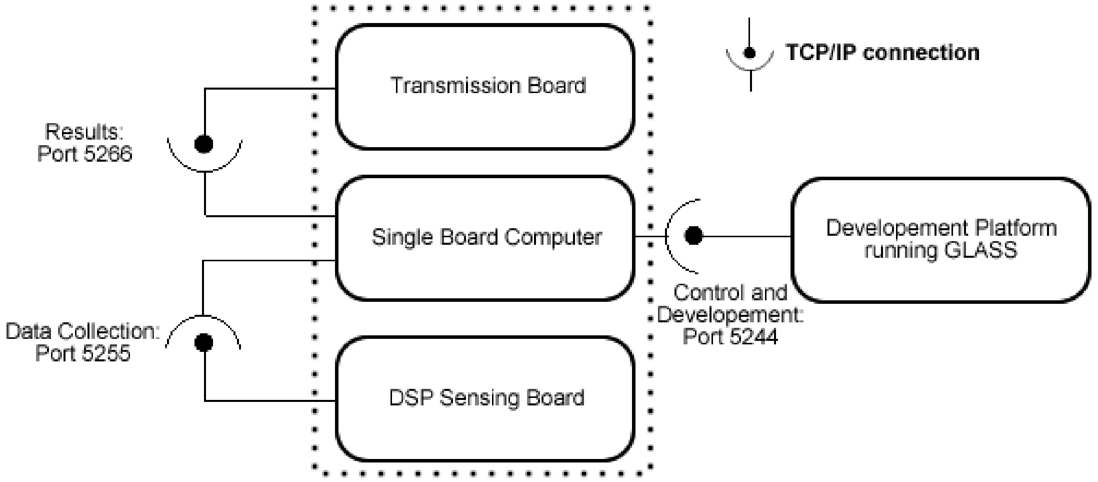
TCP/IP connection
Transmission Board
Results: Port 5266
Single Board Computer
Developement Platform running GLASS
Control and Developement: Port 5244
Data Collection: Port 5255
DSP Sensing Board

Figure 3. An overview of the hardware configuration showing the modular approach and using Ethernet protocols for connecting modules.

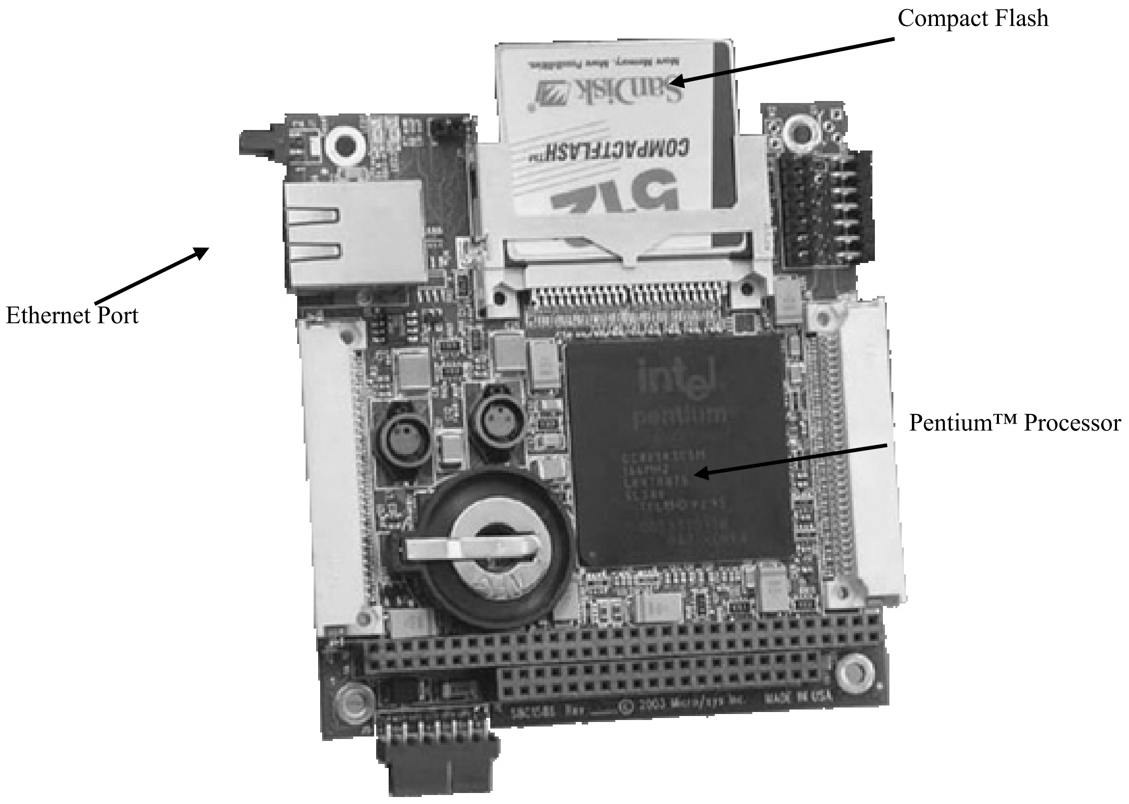

Figure 4. Single board computer (actual size) showing the processor, the compact flash drive, and Ethernet port.

3.2.4 Transmission board

The neuRFon™ transmission board developed by Motorola Labs provides a wireless access point to the WiSHM. The neuRFon™ board adheres to the IEEE 802.15.4 standard and is designed to be a self-organizing network. For example, if several boards are located within transmission range of each other, a network will be created and data dynamically routed along the most efficient path to a host node. The host node provides connectivity with an external network. The advantage to this network arrangement is if one node becomes disabled or more nodes are added, the network can dynamically reconfigure. The wireless transmission board is shown in Figure 5. Again, the wireless board is attached to the SBC through a TCP socket, allowing the neuRFon™ board, another wireless solution, or simply an Ethernet cable to act as a sending/receiving gateway.

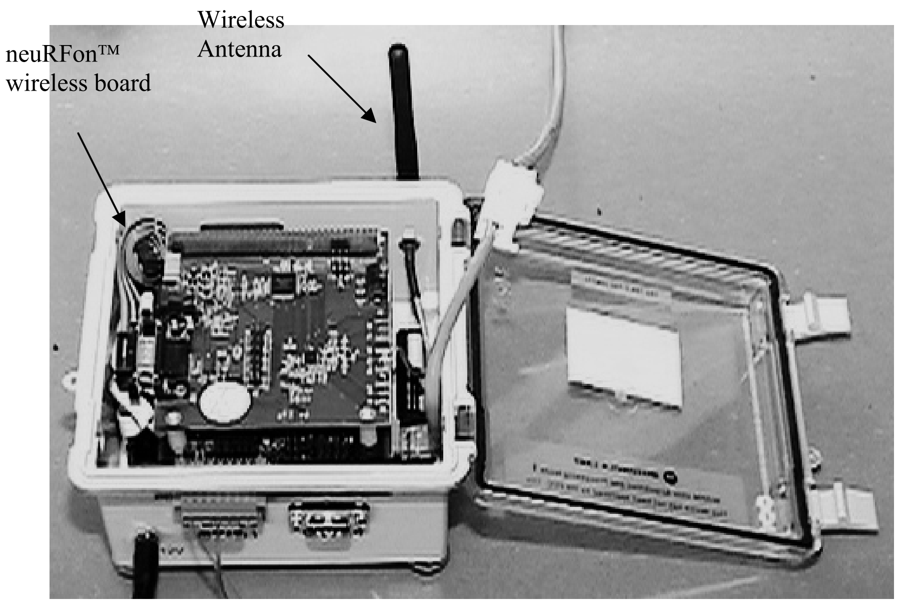

Figure 5. The Motorola neuRFon™ wireless communication board displayed on the prototype WiSHM system.

4. Summary

Current integrated health monitoring systems rely on processes statically loaded onto the monitoring node before the node is deployed in the field. The primary contribution of the work reported herein is a software paradigm that allows processes to be created remotely and uploaded to the node in a dynamic fashion over the life of the monitoring node without taking the node out of service. From a hardware perspective, the WiSHM offers many unique features that make it applicable to a wide range of SHM applications. This system is one of the first integrated data acquisition, telemetry and processing systems that has been designed specifically for SHM. The processing capability of this system is more substantial than that found in most other embedded systems being used for SHM and was designed around the DIAMOND II SHM data interrogation software. Also, the sampling rates allow this system to be used for low-frequency global system monitoring as well as for high-frequency local monitoring (see Park et al., 2003). Although not elaborated on in this paper, the WiSHM can also drive up to for actuators thus facilitating an active sensing necessary for many wave-propagation approaches to SHM. A design decision made early on during the development of this hardware was to assume that AC power would be available and to assume that it would not be necessary to time synchronize sensors associated with different WiSHM nodes. There are numerous SHM applications where these choices will not be detrimental to the SHM hardware deployment. However, the authors acknowledge that there are other applications that will necessitate a smaller physical size to the WiSHM and that will not have AC power readily available. The LANL/Motorola research team is beginning the process of planning the next generation system that will address these issues.

5. Acknowledgements
This research was funded through the Laboratory Directed Research and Development program, entitled "Damage Prognosis Solution," at Los Alamos National Laboratory.

6. References

Allen, D. W. (2004) "Software for Manipulating and Embedding Data Interrogation Algorithms into Integrated Systems: Special Application to Structural Health Monitoring, " M.S. Thesis, Dept. of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Doebling, S. W., er al. (1996) "Damage Identification and Health Monitoring of Structural and Mechanical Systems From Changes in their Vibration Characteristics: A literature Review," Los Alamos National Laboratory report LA-13070-MS.

Doebling, S.W., Farrar, C.R., and Cornwell, P.J. (1997) "DIAMOND: A Graphical User Interface Toolbox for Comparative Modal Analysis and Damage Identification," in Proc. of Sixth International Conference on Recent Advances in Structural Dynamics, Southampton, UK, pp. 399-412.

Farrar, C. R., S. W. Doebling and D. A. Nix (2001) "Vibration-Based Structural Damage Identification" *Philosophical Transactions of the Royal Society: Mathematical, Physical & Engineering Sciences*, **359**, No. 1778, pp. 131 – 149.

Farrar, C. R. H. Sohn and K. Worden (2001) "Data Normalization: A Key to Structural Health Monitoring," Proc. Of the Third International Structural Health Monitoring Workshop, Stanford, CA

Park, G., et al., (2003) "Overview of Piezoelectric Impedance-Based Health Monitoring and Path Forward, *Shock and Vibration Digest*, **35**(6) pp 451-463.

Rytter, A. (1993) "Vibration based inspection of civil engineering structures," Ph. D. Dissertation, Dept. of Building Technology and Structural Eng., Aalborg Univ., Denmark.

Sohn, H., et al, (2004a) *"A Review of Structural Health Monitoring Literature from 1996-2001,"* Los Alamos National Laboratory report LA-13976-MS..

Sohn, H., et al., (2004b) "Wavelet-Based Signal Processing for Detecting Delamination in Composite Plates," *Journal of Smart Material and Structures*, **13**(1) pp 153 - 160.

Worden, K. and J. M. Dulieu-Barton (2004) "An Overview of Intelligent Fault Detection in Systems and Structures," *Int. J. of Structural Health Monitoring*, **3** (1) pp. 85–98.