# A VIRTUAL LABORATORY FOR REAL-TIME MONITORING OF CIVIL ENGINEERING INFRASTRUCTURE

*Kevin Amaratunga*[1] *and Raghunathan Sudarshan*[2]

***Abstract*** *¾ In this paper, we describe the design and implementation of a remote virtual laboratory for monitoring physical infrastructure over the Internet. The context of this work is Project I-Campus at MIT, which includes several efforts to provide remote access to laboratory facilities. Our focus on building a virtual laboratory was to study the behavior of a flagpole under wind loading. Such a laboratory can be used for illustrating concepts in structural dynamics, signal processing and sensor technologies. We discuss some of the issues that were encountered in designing the software and hardware components of our outdoor laboratory and its associated Web interface. We also provide examples of applications where live data published on the Web is processed using Java applets. The Web provides a framework within which the field data and computer simulations are integrated to provide a set of hands-on tools for teaching engineering principles. Our system has also provided a platform for similar virtual laboratories, which are being developed at MIT.*

*Index Terms ¾ Civil engineering infrastructure, Remote virtual laboratories, Web-based monitoring.*

## INTRODUCTION

Virtual laboratories for engineering education often take on the form of a traditional laboratory apparatus that is accessed remotely over the Internet. This approach fits the traditional model of a laboratory as a controlled environment for conducting experiments. Virtual laboratories of this kind are frequently used to extend the reach and availability of test equipment, which is usually a limited resource because it is too expensive to replicate or because access must be restricted for security or safety reasons. Providing a web interface to existing laboratory infrastructure was in fact, the primary motivation behind many of the I-Labs being developed as part of Project I-Campus at MIT. Some of the laboratories that have been developed under this framework include the Microelectronics WebLab (http://weblab.mit.edu) for studying the characteristics of semiconductor devices and the Heat Exchanger laboratory (http://heatex.mit.edu) for studying chemical process control.

While these are valid examples of virtual laboratories, there are significant educational advantages in expanding our earlier definition to include remote access to real-world engineering systems that cannot be easily studied in a traditional laboratory setting. With a real-world system, one can observe and appreciate the interplay between the various components of the system as well as the higher levels of uncertainty associated with practical operating environments. However, this might not be feasible in many instances, (e.g. civil engineering and aerospace structures), because the scale and complexity of the system make it impossible to appreciate its intricate nature through direct observation alone. Providing a virtual interface to such physical systems can lend deeper insight because different physical or functional aspects of the system may be juxtaposed in an educationally significant manner within the virtual environment. Studying a real physical system through a virtual environment also allows for observations of the actual behavior to be seamlessly integrated with simulated behavior. For example, one might wish to initially observe the response of a structure under real loading and subsequently predict how the structure fails when the loads reach extreme values.

In this article, we describe the implementation of such a virtual interface around a real-world system and in particular, address the problem of monitoring in real-time, the deformation of a 102 ft (31 m) tall flagpole subjected to wind loading. The structure is located close to Boston's Charles River, in an area where wind conditions have historically been a significant factor in the design of buildings. We also describe the implementation of the web interface to our laboratory (accessible at http://flagpole.mit.edu) that enables clients anywhere on the Internet to access real-time and archived measurements of the response of the structure. Some of the unique hardware and software challenges that had to be addressed during the implementation of such an outdoor laboratory are discussed in the subsequent sections.

### Objectives

The intent behind our effort was twofold. The first major objective was to expose students to recent advances in sensors, communications and information technology, which play an increasingly important role in the design, operation, maintenance and emergency management of large-scale infrastructure projects. This objective was accomplished by involving students in the design and development of the virtual laboratory, in part through class projects. The need

[1] Kevin Amaratunga, Massachusetts Institute of Technology, 77 Massachusetts Avenue, 1-274, Cambridge, MA 02139 kevina@mit.edu
[2] Raghunathan Sudarshan, Massachusetts Institute of Technology, 77 Massachusetts Avenue, 1-249, Cambridge, MA 02139 darshan@mit.edu
Sponsored by the MIT-Microsoft Alliance as part of Project I-Campus.

for such exposure was based on the observation that students frequently have highly specialized knowledge in a particular domain, (e.g. software, sensors, signal processing, structures or transportations systems,) but only a limited appreciation for how these specialties can effectively work together in practice. This is supported by the observation that even when large infrastructure projects are instrumented with sensors, there tend to be inadequate tools for collecting and processing the large amounts of data involved.

The second objective was that once built, our laboratory would be used for teaching purposes. The Flagpole WebLab provides real-time measurements of acceleration at various points on the structure. A task that students might perform using the acceleration data is to determine the modal frequencies of the structure, which in turn can be used to estimate its material or geometric properties. In fact, good agreement has been observed between the modal frequencies observed in practice and those predicted with a finite element model. Similarly, students might analyze the acceleration data to estimate damping, to estimate and remove noise (the noise in this case is partly electrical noise from the sensors and partly mechanical noise from the flag and the flagpole cord), or to detect unusual wind patterns that suggest an approaching storm. A parallel data collection system is the photovoltaic weather station (http://pvbase.mit.edu/index.html), which collects data for wind speed and direction. This, together with the flagpole data collection system provides students with the ability to study cause and effect.

We believe that apart from its evident didactic benefits, the framework that we have developed for our laboratory also has important practical applications in real-life situations. For example, there has been an increasing trend in recent years towards the design of motion sensitive structures, where deflection rather than strength is the governing criterion. Such motion sensitive structures are controlled by active control devices, which apply a compensating force to keep the response of the structure under acceptable limits (see [1]). Therefore, accurate real-time measures of the response of the structure are essential inputs to the active control algorithm. Another emerging application of real-time monitoring is in the detection of distress in structures, be they nuclear reactors, historical monuments or tunnels. Using an interconnected network of sensors, one has the ability to predict and even prevent mishaps due to structural failure, which might otherwise lead to catastrophic damage to life and property. These ideas are in fact fundamental to the concept of an I-City (see [7]), which envisages a large-scale network of sensors around an entire metropolis linked to a web-based monitoring and diagnostic system. It has been proposed that such a system could be invaluable during emergencies by helping to significantly reduce the impact of catastrophic events.

### Outline

The outline of the rest of the paper is as follows: In the next section, we describe the hardware components of the project in detail. In particular, we discuss the sensors and data acquisition system used and address issues such as powering the sensors, interfacing the sensors with the data acquisition systems and resolution and sampling rates for the different sensor modules. We also discuss many steps taken to ensure long-term durability and accuracy of the sensors.

The section after that discusses the implementation of the software framework of our effort, which can be divided into its server and client components. The server side components consist of data acquisition and data archival programs and the client side components consist of Java applets and Microsoft .NET controls, which process real-time and archived data in various ways. These software tools are supplemented by educational simulations that illustrate concepts in structural dynamics and signal processing. We then present a number of scenarios in which the tools developed as part of our laboratory have been, or can be utilized in a classroom setting. The infrastructure that we have developed for the "Flagpole WebLab" is generally applicable and therefore, we describe how this framework can be extended to other situations. Finally, we present our conclusions and suggest avenues for further investigation.

## HARDWARE ARCHITECTURE OF THE FLAGPOLE WEBLAB

### Sensors

For our virtual laboratory, we chose to monitor the accelerations along two perpendicular axes at three points (labeled #1, #2 and #3 in Figure 1) along the length of the flagpole.

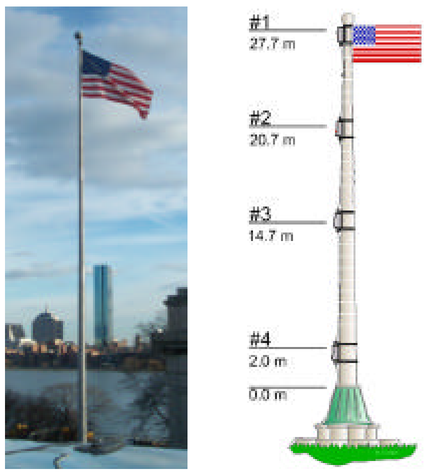


FIGURE. 1
THE FLAGPOLE AGAINST THE BOSTON SKYLINE (LEFT) AND LOCATION OF THE ACCELEROMETERS (RIGHT).

The locations of these points were chosen as the points of maximum deflection of the first three modes of the flagpole, as determined from a finite-element simulation. In addition, a thermocouple was used to monitor the ambient temperature at the flagpole's base.

For measuring accelerations, three CXL02LF3 accelerometers manufactured by Crossbow Technology were used. A few salient characteristics of these lightweight, low power triaxial accelerometers are summarized in Table I.

TABLE I
SALIENT CHARACTERISTICS OF CXL02LF3 ACCELEROMETERS

| | |
|---|---|
| Sensitivity | 1 V/g |
| Acceleration range | ± 2g |
| Supply voltage | 5 V |
| Zero acceleration voltage | 2.5 ± 0.15 V |

The accelerometers were secured inside durable watertight aluminum containers and connected to the data acquisition system via shielded cables. Since an ideal solution in an outdoor environment would consist of sensors with integrated microprocessors and short range wireless data transmission capabilities, a solution that was considered was the CrossNet wireless node, CN1100, from Crossbow (with power consumption of under 1W, a range of 100m and a sampling frequency of 500Hz). However, the version of this Bluetooth-based solution available during the installation of the sensors did not meet our range specifications. A solution that has subsequently become available from Crossbow is the MICA motes wireless sensor network. Each sensor node of this research prototype has an on-board microprocessor, which runs the UC Berkeley TinyOS operating system. The power consumption is on the order of 100mW with a radio range of 200ft.

## Data Acquisition System

A data acquisition system is used to collect data from the sensors, preprocess it in some manner (for instance by amplifying or band-limiting the signal from the sensors), digitize the signals using an analog to digital converter, and transmit the data samples in some manner to a host computer where it can be further processed. Normally, many of these functions are carried out by dedicated data acquisition cards based on the PCI or PCMCIA architectures, which directly plug into a data acquisition computer. However, this solution is not suitable for an outdoor laboratory such as ours. Instead, we used the FieldPoint distributed data acquisition system manufactured by National Instruments.

A typical FieldPoint installation (shown in Figure 2) consists of the following components:

- One or more *Sensor Input Modules*, which interface with different types of sensors. These modules perform various functions such as filtering the data and digitizing the output from the sensors, and,
- A *Network Interface Module*, which is connected to the sensor input modules via a high-speed bus. This module also connects to a host computer through an Ethernet or a serial (RS-232) link. The network interface module also provides power to all the other sensor input modules.

One of the advantages of a FieldPoint installation is that new sensor modules can be added easily and the installation can be configured using a simple software tool. An additional advantage is that since each FieldPoint installation can function independently, a single computer may be used to acquire data from many such distributed installations. The host computer accesses data from a FieldPoint sensor bank by periodically polling the corresponding network interface module (This is referred to as an *advise operation*).

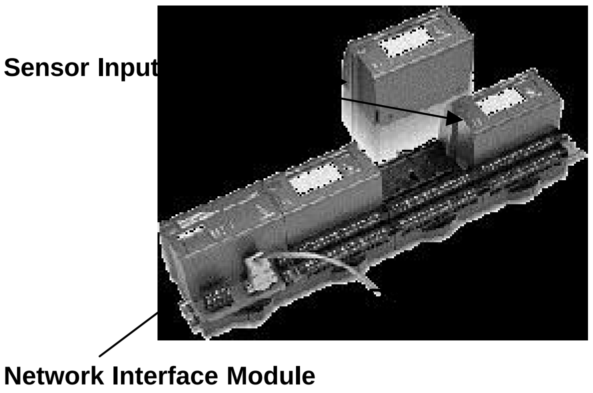


FIGURE. 2
A TYPICAL FIELDPOINT INSTALLATION.

In selecting appropriate sensor input and network interface modules for a FieldPoint installation, attention must be paid to the desired sampling rate and resolution. Unlike plug-in data acquisition cards, where the sampling rate is governed only by the analog to digital converter in the card, the effective sampling rate of a FieldPoint installation is governed both by the sampling rate of the sensor input modules as well as the network throughput rate of the network interface module [6]. Therefore, the attainable sampling rates for FieldPoint installations are usually much lower than those for PCI based measurement systems and can be found in [4]. For our monitoring project, acceleration and temperature measurements were required at 10 and 160 millisecond intervals respectively. Hence, the following FieldPoint configuration was used:

- The three accelerometers were connected to three dual channel voltage input modules (FP-AI-V10), which have a sampling rate of 2.8 milliseconds, a voltage range of 0 to 10 Volts and a 12 bit resolution. These modules in turn connected to a common terminal base (FP-TB-10). Initially, we had experimented with FP-AI-110 analog voltage input modules, but these did not have sufficiently high sampling rates.

- The thermocouple was connected to a thermocouple input module (FP-TC-120),
- The network interface module used was FP-1000, which is a serial communication module with a data transfer rate of 115.2 kilobits/sec. Here again, we had initially experimented with FP-1600 Ethernet modules, with the intention of making the data acquisition system wireless by replacing the ethernet cable with a pair of ethernet to (802.11b) wireless converters. While this approach worked for relatively low sampling rates, it was unsatisfactory for our application because the Ethernet module lacked an onboard buffer, which resulted in each transmitted packet containing only one sample per sensor channel. This problem has been addressed in newer Ethernet modules, such as the FP-2000.

The sensors were installed on the flagpole and wired to the data acquisition system in mid-February 2002 (see Figure 3 for photographs taken during the installation process). In mid-March, an underground cable was laid between the data acquisition system and an indoor computer. Our system has been operational since without interruption.

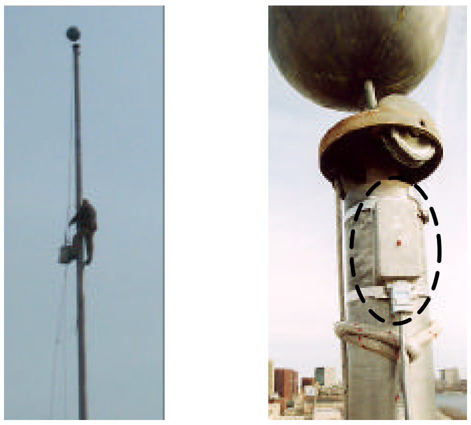


FIGURE. 3
INSTALLATION OF THE SENSORS (LEFT) AND ONE OF THE ACCELEROMETER BOXES (RIGHT, CIRCLED).

## SOFTWARE FOR DATA COLLECTION, ARCHIVAL AND RETRIEVAL

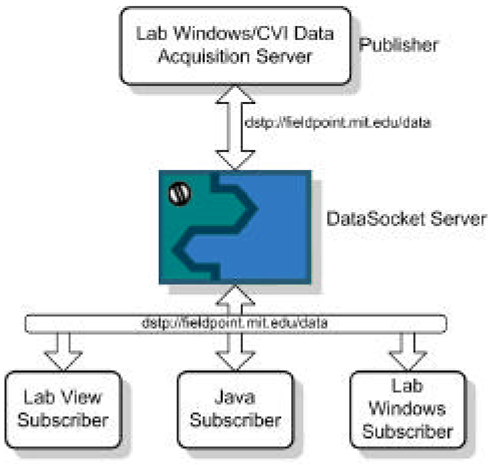
Once the hardware (sensors and data acquisition system) was installed, it was necessary to write software to perform the following functions on the host computer:

- Poll the instrument periodically and obtain data from the different sensor input modules,
- Make real-time data accessible to clients on the Internet, and,
- Archive the data and handle requests for historical records.

All the software components written for our project can be accessed at http://flagpole.mit.edu/software.html.

### Data Collection and Dissemination

The first two tasks were easily accomplished using a software solution from National Instruments called LabWindows/CVI. This environment provides C libraries, which can be used to connect to a FieldPoint installation and extract data from the network interface module. National Instruments also provides an API known as DataSockets (see [5]) for sharing data in real-time between clients in different platforms. Instead of looking at the source of data as a *server*, and the application accessing it through the network as a *client*, the DataSocket API has the notion of a *publisher* and a *subscriber*, respectively. In addition, there is a DataSocket server, which is a program handling all the communication between the two. The publisher first binds to a unique URL on the DataSocket server and publishes data to that URL. Various subscribers can then access the data published by reading data from the same URL (see Figure 4). The DataSocket API thus handles many of the low-level implementation details and provides a convenient mechanism for transferring data among different programs on different platforms.


Lab Windows/CVI Data Acquisition Server
Publisher
dstp://fieldpoint.mit.edu/data
DataSocket Server
dstp://fieldpoint.mit.edu/data
Lab View Subscriber
Java Subscriber
Lab Windows Subscriber
FIGURE. 4
EXCHANGING DATA USING DATASOCKETS.

### Data Archival

Due to the high volume of acquired acceleration data, it was found that archiving it on the same machine placed a heavy burden on the CPU, which adversely affected the data acquisition process. Hence, it was decided to adopt a distributed solution with two machines, one of which served as a data acquisition server and the other which served as a database server. However, this solution in turn lead to a few difficulties, which are discussed subsequently.

The archived acceleration and temperature records are stored in a Microsoft SQL Server 7.0 database. However, we realized that archiving the entire acceleration record in the database was very expensive in terms of storage space and retrieval time. Therefore, it was decided to store acceleration data only for the past 24 hours in the database;

data older than a day is purged from the database and stored in zip files. Whenever data older than a day is required, it is extracted from the zip files and uploaded into the database. Storing the temperature data did not pose this problem because it is sampled less often; moreover, since ambient temperature varies rather slowly, only one sample out of 100 is actually stored in the database.

The program which archives the data and purges acceleration data older than a day is implemented in Java. It subscribes to the data published on the DataSocket server by the data acquisition program. The acceleration records are first buffered in a text file. At the end of each minute, this text file is inserted into the database and added to a zip archive. It was found that this sequence of operations was far more efficient than directly inserting each record into the database from the Java program. At the end of each hour, a new zip file for the acceleration records is created, and data older than 24 hours is purged from the database. The temperature data is directly inserted into the database every 16 seconds (i.e., 100 times less often than it is sampled).

One of the difficulties we experienced due to the distributed nature of our system was that there were now two points of failure: the data acquisition server and the database server. Of the two, the former was observed to be more critical, and recovery from such failures required manual intervention. To make crash recovery automatic, a thread was implemented in the data archival program that periodically checks the status of connection with the data acquisition server, and starts up or shuts down the archival program depending on whether the publisher is online or not.

### Data Retrieval from the Database

One of the goals of creating the server-side software was to ensure that it would be easy for students to write their own programs (i.e. Java applets or .NET controls) to access real-time and archived data. While it is fairly straightforward to access archived data from Java programs using the JDBC (Java Database Connectivity) API (see [2]), it was decided to write our own interface over JDBC to provide a convenient mechanism for data access. The implementation of this layer was necessary for another important reason: by default applets running within a browser can make network connections only to the host from which they are served. Hence it would not be ordinarily possible for an applet to acess both real-time and archived data simultaneously, since these are served from different machines. However, our database interface runs as a standalone application on the data acquisition server and is hence not bound by this restriction. Our interface also validates all queries and ensures that only a reasonable amount of data is queried at a time from the database.

The database access layer was written using the Java Remote Method Invocation (RMI) API (see [8]). This collection of libraries allows Java programs on one machine to invoke methods on Java objects existing on other machines. The RMI layer implemented for our project provides two methods for accessing acceleration and temperature data respectively. Each of these methods validates the input and executes the query on the database. The result from running the query is then sent back to the Java applet.

In addition to the RMI interface, archived data from the database is also made available through a servlet (accessible at http://flagpole.mit.edu/jsp/DBInterface.jsp). While this is not as interactive as an applet (the interaction is via HTML forms), it does not require a Java virtual machine on the client web browser and is often more convenient.

## SOFTWARE APPLICATIONS FOR THE VIRTUAL LABORATORY

As mentioned in the introduction, one of the goals of building our virtual laboratory was to enhance the comprehension of concepts in structural dynamics, sensor technology and signal processing using a real-world structural system as a laboratory model. In this section, we describe a few software applications, consisting primarily of Java applets, which were developed towards this goal. These software tools can be divided into two categories: the first category consisting of applets which process real-time data in different ways and the second category consisting of applets which illustrate concepts in structural dynamics and signal processing using numerical simulations. In addition, we also made it convenient for students to write their own programs to process real-time and archived data.

Applets which access and process real-time and archived data can be found at http://flagpole.mit.edu/realtime.html. In these programs, real-time data is accessed using a Java (or .NET) implementation of the DataSocket API and archived data is accessed using the Remote Method Invocation layer described in the previous section. Some of the applets that belong to this category are:

- **DataSocket Reader Applet** (http://flagpole.mit.edu/datasocketreader_frames.html). This applet simply reads real-time acceleration and temperature data published by the data acquisition program and plots the acquired data on a time axis. It also displays temperature variation over the past 24 hours. While this applet only acquires and displays raw data, it can be used as a template for implementing many data processing algorithms such as denoising or data compression.
- **Wavelet Decomposition Applet** (http://flagpole.mit.edu/wavedec_frames.html). Wavelet decomposition of a signal, see Figure 5, provides a joint time and frequency localization capability that is superior to a windowed Fourier transform. This applet performs a three level wavelet decomposition of the raw acceleration data to identify frequency bands in the signal with maximum energy. Wavelet transforms have

the advantage that they can help identify frequencies of interest and the time of occurrence almost in real time. This applet also performs a fast Fourier transform of the data every 1024 samples to identify the dominant mode of vibration and illustrate the tradeoffs involved in using the two methods. The wavelet decomposition applet can be further extended to process the signal in different ways. An applet which denoises the signal in real-time by doing a wavelet decomposition can be found at http://flagpole.mit.edu/denoise_frames.html. .

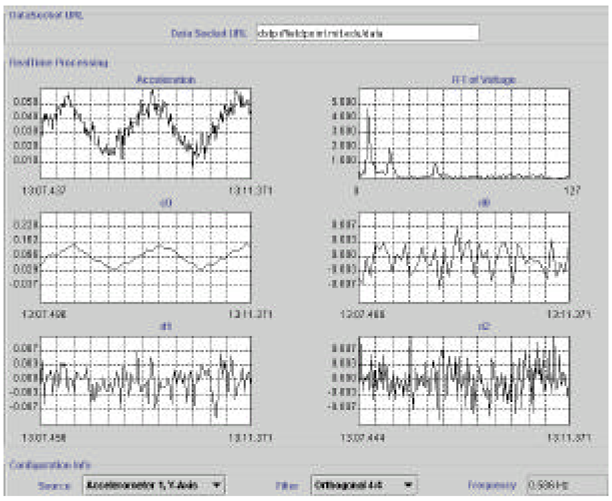


FIGURE. 5
SCREENSHOT OF THE WAVELET DECOMPOSITION APPLET.

In addition to the above, applets which illustrate concepts in structural dynamics and signal processing can be found at http://flagpole.mit.edu/education.html. Some of these applets include:

- **Tuned Mass Damper Applet** (http://flagpole.mit.edu/applets/TunedMassDamper/TunedMassDamper_plugin_1.3.html). This applet simulates the response of a building with a tuned mass damper system. Unlike viscous dampers, tuned mass dampers are inertial systems, that is, the damping is provided by a mass moving out of phase with the building. This applet can be used to illustrate the principles behind optimal design of such a tuned mass damper for a building subjected to wind and earthquake loads. Figure 6 illustrates the working of the tuned mass damper applet.

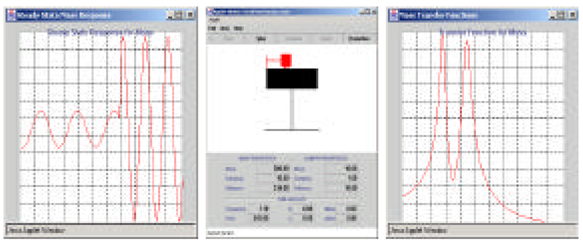


FIGURE. 6
THE TUNED MASS DAMPER APPLET.

- **Mohr's Circle Applet** (http://flagpole.mit.edu/applets/mohr2/mohr2_plugin_1.3.html). This applet is an interactive simulation of the Mohr's circle, which is a graphical tool to determine the stresses along different planes passing through a point.

Many of these software tool were developed as class projects for graduate level courses on Wavelets and Filterbanks and Software Engineering. The simulation tools were used in several courses in the Department of Civil and Environmental Engineering such as Civil Engineering Materials, Structural Dynamics and Motion Based Design.

## SUMMARY AND CONCLUSIONS

In this article, we described the design and implementation of a virtual, web-accesible interface to a real-world struture. We also described the utilization of our laboratory in an engineering curriculum to enhance the comprehension of physical principles.

The framework developed as part of our project has been succesfully applied to other similar virtual laboratories, such as one involving the measurement of hydrological parameters in wells around the MIT campus. In the future, we hope to extend the scope of our project to other larger civil engineering infrastructure systems.

## ACKNOWLEDGEMENT

The support from Microsoft Corporation through Project I-Campus at MIT is gratefully acknowledged. We would like to thank all students who contributed software to our project. We would also like to acknowledge the technical assistance received from National Instruments.

## REFERENCES

[1] Connor, J.J., "Introduction to Structural Motion Control", *http://moment.mit.edu/r_textbook.asp*, 2001.

[2] Hamilton, G., and Cattel, R., "JDBC™: A Java SQL API, Version 1.20", *Sun Microsystems Inc.*, 1998.

[3] Microsoft Corporation, ".NET Framework", *http://www.microsoft.com/net*, 2002.

[4] National Instruments, "FieldPoint Benchmarks", *ftp://ftp.ni.com/support/fieldpoint/Server/fp16_rates.pdf*, 1998.

[5] National Instruments, "Integrating the Internet into Your Measurement System: DataSocket Technical Overview", *National Instruments White Paper*, No. 1680, 1999.

[6] National Instruments, "Sampling Rates of FieldPoint Modules", *National Instruments Knowledge Base*, No. 1O3CJ7US, 1999.

[7] Sudarshan, R., "A Web-Based Virtual Laboratory for Monitoring Physical Infrastructure", *SM Thesis, Department of Civil and Environmental Engineering*, *MIT*, 2002.

[8] Sun Microsystems, "Java ™ Remote Method Invocation Specification", Sun Microsystems Inc., CA.