AC 2012-5448: PLATFORM INDEPENDENT INTERFACE FOR REMOTE LABORATORY EXPERIMENTS

Mr. Bo Cao, University of Houston
Dr. Gangbing Song, University of Houston
Xuemin Chen, Texas Southern University
Mr. Daniel Osakue, Texas Southern University

Platform Independent Remote Laboratory Experiments

Abstract

A remote laboratory experiment is an online experiment that requires only external input through the internet to control. These days, online experiments have not been widespread in the use of engineering curriculum because of their complexity in both development and use. We advocate the development and use of a set of protocols to control an experiment through a mixture of the Web 2.0 technologies to provide users with an interactive interface that does not require any additional software to be installed. This dynamic interface will be compatible with most of the currently available and commonly used web browsers and operating systems available. A scheduler server will used for scheduling between the client and the experiment in order to provide scheduling and a database for the experiment to store its information. This scheduler server can handle many experiments at once in order to provide students with a more complete experience in the remote laboratory experiment field.

Introduction

Engineering curriculums have always included hands-on experience as part of education. Starting in secondary education, sometimes even as low as 5th grade, students have performed experiments in their science classes in order to receive a more complete understanding of what it is that they are learning. These experiments escalate in difficulty and equipment needed throughout a student's education. When a student reaches college and becomes an undergraduate, equipment has become thousands of dollars each and individual space needed for each student has reached a small room. It has become no longer possible to fit more than 20-30 students in a room to perform these experiments, and different sessions have to be made to accommodate an entire class. The cost and maintenance of such laboratories have been a huge burden for schools both in terms of money and time[1,3,7].

One of the ways schools have dealt with this is what we call "virtual laboratories". These laboratories are purely mathematical in nature and provide students with a system that only uses numbers to indicate changes that they make. The students are not provided with any other visual confirmation of what it is that they are doing other than these numbers[5]. Our solution to this is a mixture of the traditional and the virtual laboratories into what we call the remote laboratory experiment[2,9].

Remote laboratory experiments are real experiments conducted virtually by students through the internet. These experiments use a real set of equipment and instrumentation, located in a remote location, such as a room in the school, and can be controlled from any terminal that has an interface to control it. These terminals can be anything with internet access, even a smart phone with 4G wireless access.

One of the more popular tools right now in the field is the National Instrument's Laboratory Virtual Instrument Engineering Workbench (LabVIEW) software[4,6,8]. LabVIEW is ideal for rapid prototyping of an experiment and was designed with control and interfacing in mind. Many

industrials current use LabVIEW software to control their equipment, including the aerospace industry. One of the things that LabVIEW does not do too well in, however, is remote control of their systems. LabVIEW has provided a runtime engine for users to download on their web browsers to control their interfaces. This runtime engine is currently around 210mbs and growing fast throughout the versions. The worst part about these runtime engines is that they are not backward compatible so the user must use the version of the engine that the interface was designed with. Also, because of their size and nature, these runtime engines require administrator rights on the computer to install. Even with laptop computers becoming more popular these days, many students still choose to use the school's computer room as a means of accessing the internet in breaks between classes. These computers do not allow you the rights to install these runtime engines and therefore cannot use the LabVIEW interfaces. Furthermore, any updates to the LabVIEW front panel may result in errors in the runtime engine.

Recently, LabVIEW has included a new feature in their software to interface with RESTful web services. REST (Representational State Transfer) allows the experiment to communicate with the outside world by more than just LabVIEW interfaces, it provides an architecture for autonomous control of such experiments through personally-designed interfaces using other means.

This is where our remote experiment interface comes in. Using a pure JavaScript environment, our interface provides for a unique experience that requires no add-ons or plug-ins other than a web browser and a high speed internet connection.

Methodology

The goal of our project is to provide users with an interface that will work in any Internet-enabled web browser without the need to install any software. The project will include three sections: client side, web server, and experiment server (see Figure 1). The client side will communicate their commands to the experiment server through the web server, which acts as a medium for control and data-basing. When the clients log in their web browser on our website where the web server is located, they will be asked to sign in using their given username/password. This username/password combination will be set to a default value for each student, using their student id number as the username and password. Upon logging in for the first time, the student can then change their password to a value they desire with certain restrictions. This password is encoded in MD5 before sending for security purposes. The MD5 message-digest algorithm is a cryptographic hash function that produces a 128-bit hash value. On this website, students can use a calendar to sign up for two hours per day on the experiment that they desire to use. When it is time for their timeslot, students will be given a link to access the experiment interface.

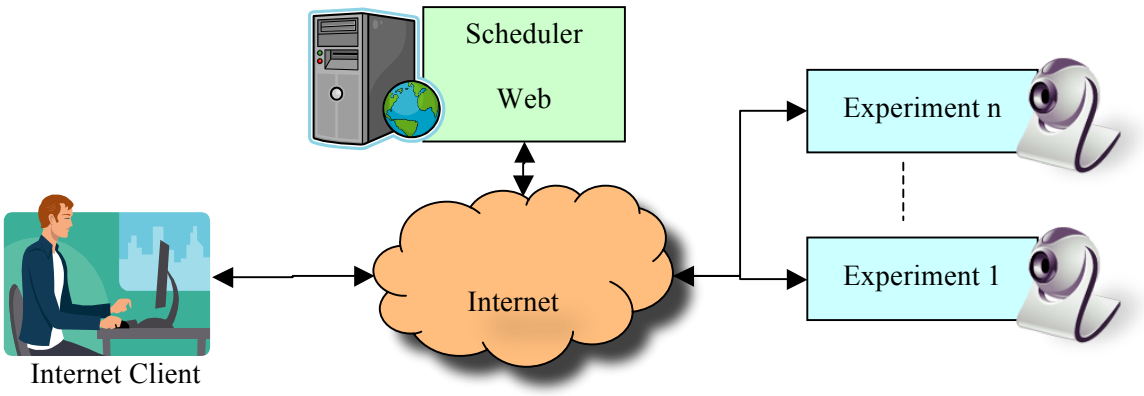

Figure 1: Client, Web Server, Experiment Server

For a prototype of this interface, we used a simple experiment with one output, the Carbon Nanofiber Beam Experiment. This experiment uses a carbon nanocomposite paper made ferromagnetic with electrodeposition of nickel particles and coated with an outer layer of PDMS (polydimethysiloxane), a clear silicone gel. This piece of paper is then put in between two coils of metals which will induce a magnetic field on them. The strength of the magnetic field, and therefore the displacement of the paper from the center position, will depend on the strength of current provided to the coils. Measuring this displacement is a laser displacement sensor.

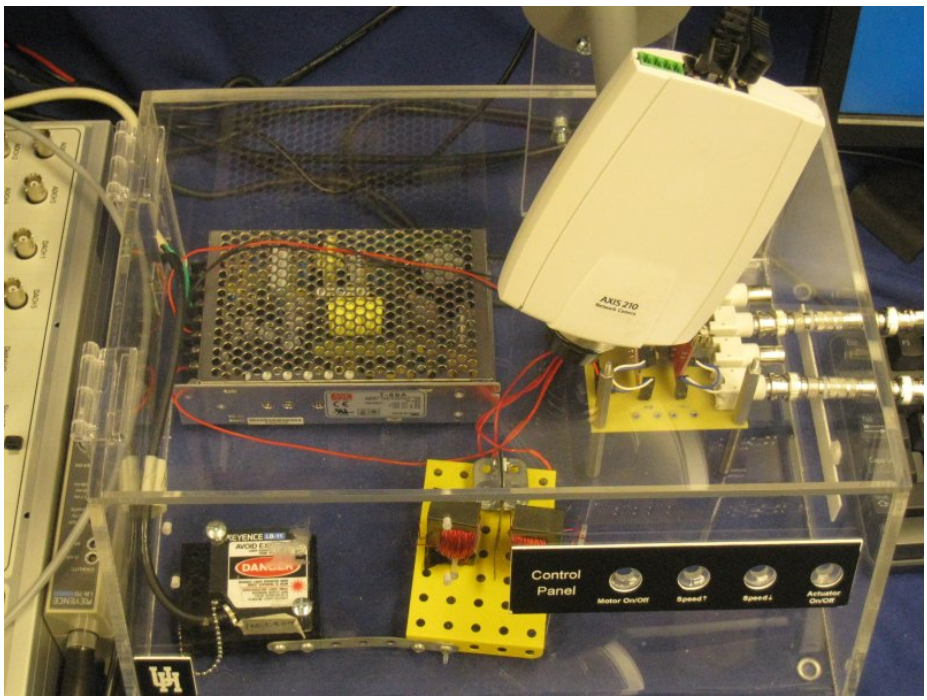

Figure 2: Carbon Nanofiber Beam

The interface itself it fairly simple and easy to understand, there is a “Start” and “Stop” button, which will initially display start and will change to stop once the user presses it. Pressing this button starts up a connection with the web server which will then automatically relay to the user a live webcam feed on the experiment. Also, this button will then activate the link which allows the web server to relay the experiment parameters from the interface to the experiment server. These parameters are the changing variable in the experiment itself and by modifying them, the user will see visual changes in the experiment webcam and dynamic changes in the experiment chart. These experimental parameters can include but is not limited to: “frequency of input value, amplitude of input value, offset of input value, and sensor bias”. The interface itself checks the values of the parameters before sending them to the web server, invalid selections such as letters or values that are too high/low are not accepted and will produce an error. These values are also checked on the experiment side in LabVIEW for double safety.There is shown on the interface itself a chart of the displacement of the experiment, its only output besides time. Also included is a record button that will log the values of displacement to a default log file that the user can later use. The logging of the values will continue until the user presses the “Stop” button. This log file can be opened in Microsoft Excel for users to plot and look at their data over time in more detail.

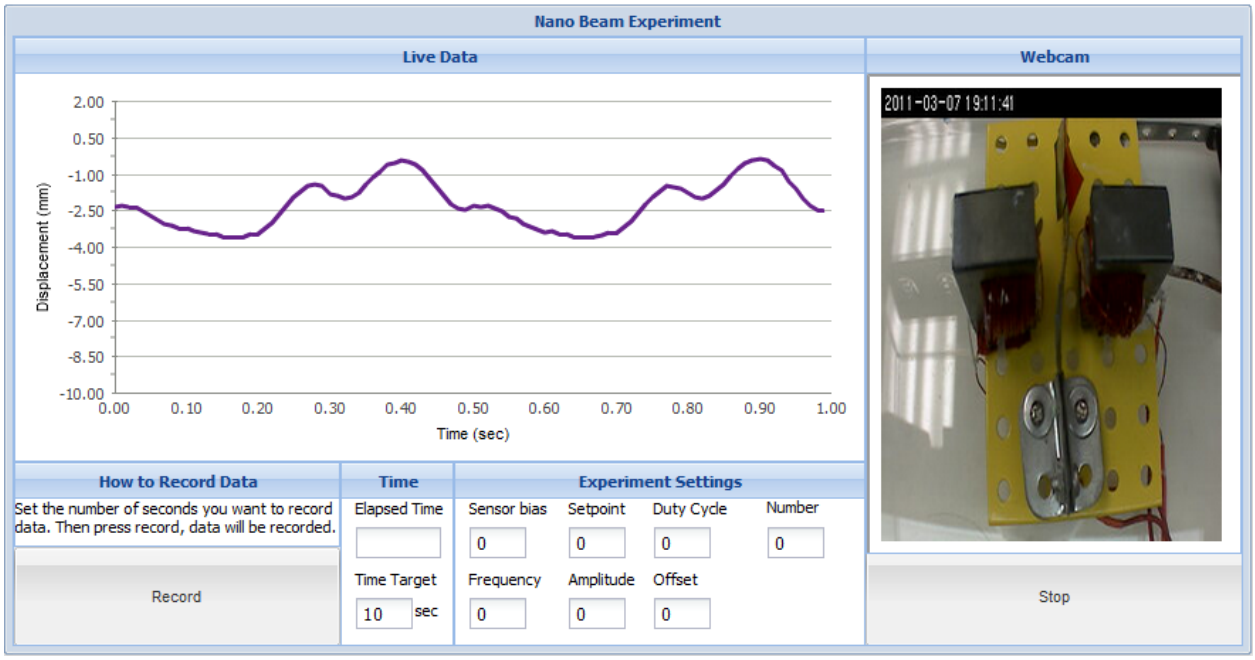

Figure 3: User Interface

The web server that will act as a medium between the experiment and the user is a Linux based machine with a Python server that uses phpMyAdmin as the database controller. PhpMyAdmin is a tool written in PHP that is intended to handle the administration of MySQL databases over the internet. The experiment sends its information first to the MySQL database, which will then store the information and send it out periodically to the user interface.

On the experiment side, the entire control and interface is done in LabVIEW. Since the students will not be able to see the LabVIEW front panel of the experiment, little modification was

needed on the front panel side of the Carbon Nanofiber Beam experiment. The original interface, modified slightly for debugging purposes, is shown below.

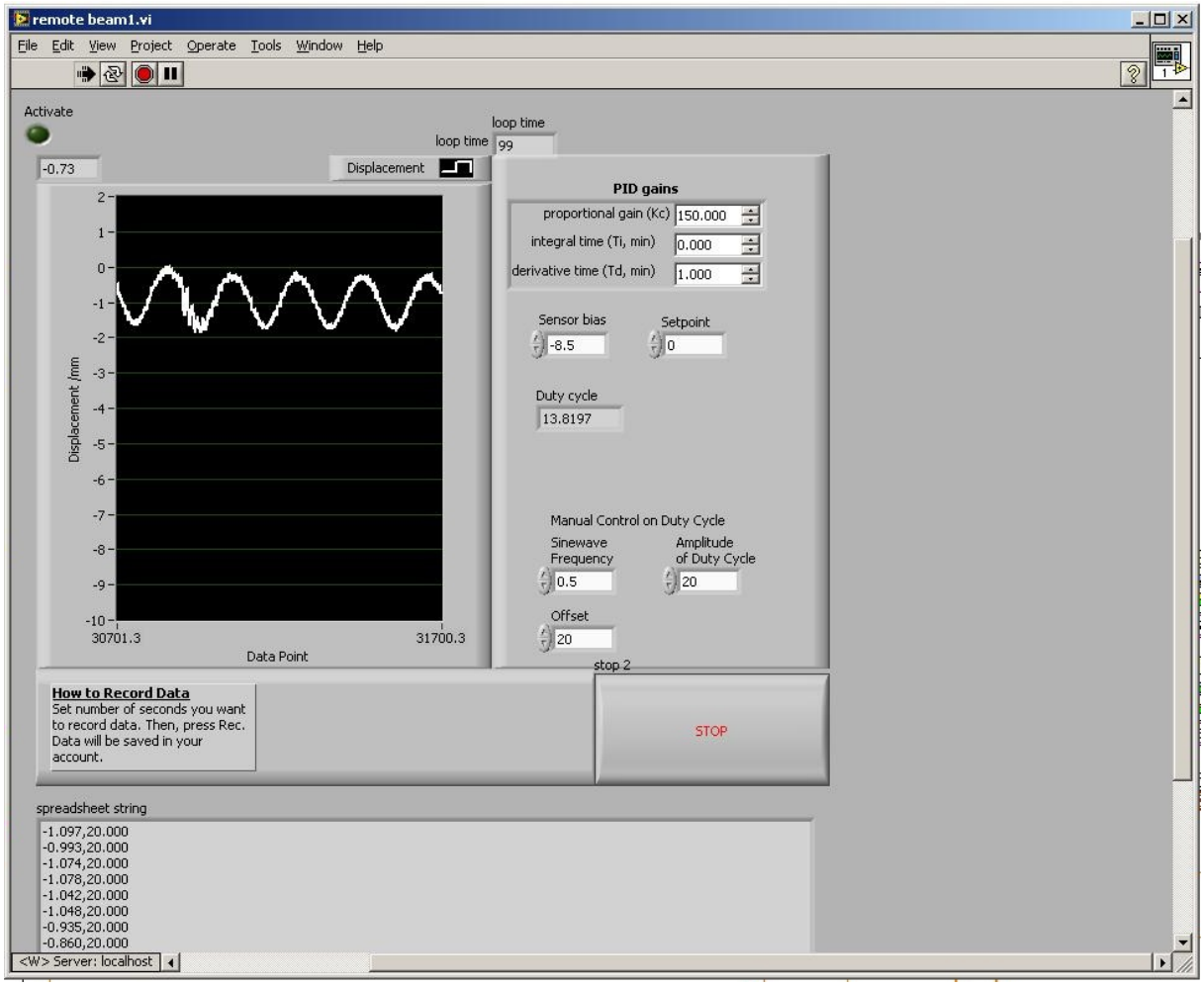

Figure 4: LabVIEW interface

Technologies

Each of the three sections uses different technologies in order to communicate with each other. There is a framework consisting of five levels on how the remote laboratory works.

Table 1: Framework levels and technologies

Level	Name	Technology/Protocol
1	Client – User Interface	JavaScript, JSON, Ext JS
2	Data Protocol – Low Speed	TCP Sockets
3	Server - Web Service	WSGI, REST, MySQL, PHP
4	Data Protocol – High Speed	TCP Sockets
5	Experiment Server	LabVIEW

Th2 Table 1 shows the technologies and protocols that are used in the framework of the remote laboratory experiment. On Level 1, the client uses the user interface to control the experiment. This interface is built completely using JavaScript, which will run on any modern web browser. JSON (JavaScript Object Notation) is a lightweight data-interchange format for JavaScript. JSON is built upon two structures: a collection of name/value pairs (object) and an ordered list of values (array). The interface communicates with the web server using JSON format, which the web server will relay to the experiment server. Ext JS is a JavaScript library useful for building interactive web applications using techniques such as Ajax, DHTML, and DOM scripting. In the previous version, Ext JS 3.0, the chart-making functions still used Flash to create the charts, which made this version not truly platform independent, as Apple OS and systems do not support Flash. Ext JS 4.0, the newest version, will allow for the creation of charts using SVG (Scalable Vector Graphics) and VML (Vector Markup Language), both of which are fully capable of being rendered on any modern web browser.

SVG is a family of specifications based on an XML file format for describing two dimensional vector graphics, it is an open standard that has been fully developed since 1999. SVG images are fully described in XML text file and are supported and directly rendered in all major modern web browsers, including Mozilla Firefox, Internet Explorer 9, Google Chrome, Opera, and Apple Safari. How SVG works is that while a common bitmap is composed of a set of dots, which when enlarged may lost their quality, SVGs are composed of a set of shapes, so scaling is not an issue for SVGs.

VML is also an XML language used to produce vector graphics. VML is an extension of XML1.0. VML supports the markup of vector graphics the same way that HTML does textual information. In VML, the content is composed of paths using connected lines and curves. VML is commonly used by Microsoft Office applications within their “save as html” option. VML is not natively supported by most modern web browsers, which uses SVG instead.

On Level 3 of the framework, WSGI (Web Services Gateway Interface) is used as an interface between the web server core software and the web applications that allows building web services faster than the old CGI (Common Gateway Interface) using the Python language. REST(Representational state transfer) provides a way for the server to handle packets from the user interface. REST uses the standard HTTP methods, such as GET, POST, PUT, and DELETE, to interact with the user interface. HTTP defines nine methods (sometimes referred to as "verbs") indicating the desired action to be performed on the identified resource. What this resource

represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server. Each of these verbs generates an action on the server depending on the method and the input parameters. The most commonly used verb in our interface is POST, which, when included with embedded data, will be used as a request to update data in the experiment. Because this level only requires minor amounts of data to be sent, only a low data speed protocol is needed. The table below will show some of the commonly used ways the interface uses these commands[10].

Table 2: REST methods and retrieval of data

Client Request			**Server Response**	
Resource	**Parameters**	**Method**	**Action**	**Response Data**
Experiment				
/exp	/expID	GET	Gets all experiment information	Returns all experiment information JSON
User				
/user	/userID	GET	Gets all user information	Returns all user information JSON
Client Request			**Server Response**	
Resource	**Parameters**	**Method**	**Action**	**Response Data**
Experiment Data				
/expLive	/expID	GET	Get Data Array from Exp.	Returns Data Array JSON
/expLive	/expID /userID /active	POST	Case: {1-Start, 0-Stop} Update database. Inform Experiment.	Returns acknowledgement
/expLive	/expID	POST	Case: {parameters}	Changes parameters on experiment

At the fifth level of the framework, the experiment server is completed entirely in LabVIEW. LabVIEW will acquire data from the experiment and parse it into an array. This array will then be sent through a TCP socket to the MySQL database located on the web server. This information will be stored in the database and sent periodically to the user. This will require a high bandwidth because of the webcam information and high amounts of data being sent. MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. MySQL is written in C and C++ and the parser is written in Yacc. MySQL works on many different system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, Mac OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

Conclusion

The Carbon Nanofiber Beam experiment at the University of Houston was modified from a LabVIEW runtime engine to this Platform Independent Interface. The client uses a user interface developed entirely in JavaScript to ensure that it runs on any modern web browser with high speed internet access. A Linux machine hosting a Python web server was set up to act as a medium for communication between the client and the experiment server while still maintaining a secure level of communication. With this interface, no add-ons or plug-ins will need to be installed on any computer, and anyone with a web browser and internet access will be able to use the interface to control an experiment remotely.

Acknowledgments

This work is partially supported by the National Science Foundation under Grant Numbers EEC-0935208, EEC-0935008, and DUE-0942778.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Bibliography

[1] Ambrose, S. A., & Amon , C. H. (1997). Systematic design of a first-year mechanical engineering course at Carnegie Mellon University. Journal of Engineering Educations, 173-181.
[2] Boehringer, D., Jeschke, S., & Richter, T. (2009). Lila - A European Project on Networked Experiments. Paper presented at the Sixth International Conference on Remote Engineering and Virtual Instrumentation.
[3] Corter, J., Nickerson, J., Esche, S., & Chassapis, C. (2004). Remote vs. Hands-On Labs: A Comparative Study. Paper presented at the 34th ASEE/IEEE Frontiers in Education Conference.
[4] Duro, N., Dormido, R., Vargas, H., Dormido-Canto, S., et al. (2008). An Integrated Virtual and Remote Control Lab: The Three-Tank System as a Case Study. Computing in Science & Engineering, 10(4), 50-59.
[5] Felder, R. M., & Brent, R. (2005). Understanding Student Differences. Journal of Engineering Education, 21(1), 166-177.
[6] Jeschke, S., Richter, T., & Sinha, U. (2008, Oct. 2008). Embedding Virtual and Remote Experiments Into a Cooperative Knowledge Space. Paper presented at the 38th ASEE/IEEE Frontiers in Education Conference, Saratoga Springs, NY.
[7] Jing, M., & Jeffrey, V. N. (2006). Hands-on, simulated, and remote laboratories: A comparative literature review. ACM Comput. Surv., 38(3), 7.
[8] Olmi, C., Song, G., & Mo, Y. L. (2007). An innovative and multi-functional smart vibration platform. Smart Mater. Struct., 16, 1302–1309.
[9] Song, G., Olmi, C., & Bannerot, R. (2007). Enhancing Vibration and Controls Teaching with Remote Lab Experiments. Paper presented at the Proceedings of the 2007 ASEE Annual Conference & Exposition.
[10] Olmi, C., Cao, B., Chen, X.., & Song, G., (2011). A Unified Framework for Remote Laboratory Experiments. Paper presented at the Proceedings of the 2011 ASEE Annual Conference & Exposition.