Web-Based Distance Experiments: Design and Implementation

Timothy Chang[1] and Douglas Hung[2]
[1]*Department of Electrical & Computer Engineering, NJIT, Newark, NJ 07102USA*
[2]*Department of Computer & Information Science, NJIT, Newark, NJ 07102, USA*

Abstract: As part of the effort to enhance retention and improve learning, web-based distance experiments have been designed and implemented so that full-time and part-time students may carry out their experimental work from home or other convenient sites on a flexible schedule. This project differs from other existing web-based distance learning experiments in the following way: First, our experiments are based on a Texas Instruments TMS320C31 floating point Digital Signal Processor (DSP) which is capable of very high bandwidth, real-time signal processing, independent of the host PC's configuration. Second, Java is used to implement the Internet communication so that multi-user, near real-time interactive access can be achieved. Third, an Application Programming Interface has been written for the DSP to facilitate platform independent programming. Finally, the users have the option to upload their code segments to the DSP via the Internet for to customize the experiment without posing security problems on the server.

Keywords: Distance experiments, web-based system, real-time control, digital signal processing

1. Introduction

Among the various factors affecting the performance of students in urban institutions, the limited availability of laboratory facilities is one of the most critical. This is especially so for part time students who are constrained by their work schedule and ability to come on campus on a regular basis. With the increase in bandwidth through high speed and ISDN data lines, web-based distance experiments can play a significant role in supporting the learning process by providing a 24/7 laboratory with real experiments, not "sanitized" simulations, so that the students can draw realistic conclusions and gain insight for real-life problem solving. General advantages of web-based distance experiments include the ability to:

1. Enhance student learning.
2. Improve schedule flexibility.
3. Reduce laboratory operating expenses and equipment breakdown costs.
4. Introduce research experiments into teaching.

A number of web-based distance experiments are already in place. For example, Resource Center for Engineering Laboratories at the University of Tennessee at Chattanooga offers distance experiments in process control and dynamics with Labview interface[1]. Oregon State University's "Second Best to Being There" network application allows user to remotely control a robot arm [2]. The Automated Internet Measurement Laboratory established at Rensselaer Polytechnic Institute and at the Norwegian University of Science and Technology [3] provides a remote measurement module for semiconductor characterization.

While these approaches are meritorious and work well for the intended applications, it is the objective of this work to devise a general-purpose, distance experimental system that possesses the following characteristics:

1. Efficient interactive/multi-user operations.
2. Machine/platform independent.
3. Secure operations.
4. Graphics User Interface compatible.
5. High processing bandwidth with true real-time capabilities.

To meet these requirements, several hardware (PC and digital signal processor) and software (Java and Application Programming Interface) components are utilized and integrated. These components are described in Sections 2-4 below.

2. Internet Communi cation

Most existing web projects are written in CGI/Perl since CGI programs provide strong server-side control. However, for web-based experiments where hardware reliability and safety issues are of primary concern, CGI can no longer be used. Java programs, on the other hands, provide strong client-side control. Through graphical user interfaces, clients can interactive with the server in real time. On the security issue, Java language has been deliberately made to protect against invasion of computer resources. One further advantage of using Java is that Java programs (applets) use sockets as their backbone. In the case of Java program, the socket connection can be maintained as long as the applet is active. Thus, multi-user, intra, inter group interaction can be readily implemented. The students visit the html page in the server-side by accessing its URL address. The html interpreter interprets this page and an applet is then being requested. The applet interpreter deciphers the Java byte-codes and shows the result. Since the socket connection keeps active, the data will be continuously manipulated and the changes will be shown as soon as they are made. A server applet has been generated to handle upload and setting experimental parameters. A client applet handles client-side user interface and deciphers input settings. The overall configuration is shown in Figure 1 while multi-user access configuration is shown in Figure 2:

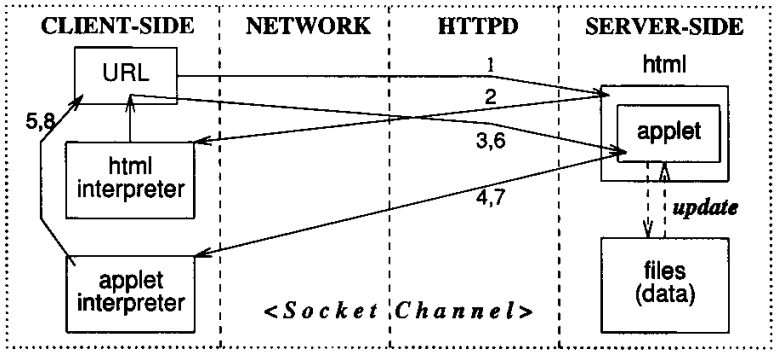

Figure 1: Client/server configuration

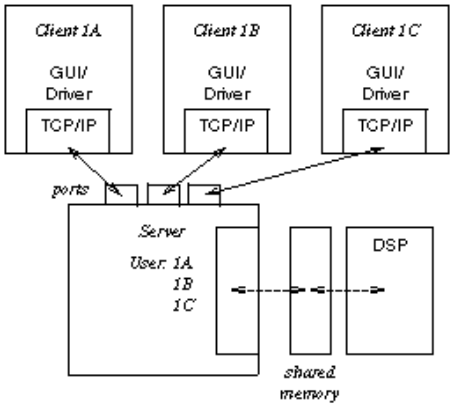

Figure 2: Multi-user access configuration

It should be noted that:

1. All members of the experiment can input their own parameters through the provided graphic user interface. Users can discuss their opinions through the interface. Any designed member or the captain of the term can issue commands to initiate the experiment. All the involved users can view the output of the DSP board.
2. The parameters or pre-coded messages are download to the shared memory and then the server applet initiates an API application to compile the coded messages or to convert the parameters to an executable code which will be uploaded to our DSP experiment circuit.

3. A control panel will then shown on the screen of the one who initiates the process. The generated code can be loaded into the DSP circuit by click `load` button and the experiment can be initiated or stopped simply by click the `go` or `stop` button.
4. The DSP board will continue to dump outputs to coded addresses of the shared memory, so that the server can upload the responses and sends them to update the client sides, such that the progressing of the experiment can be view in a real-time manner.

3. Pilot Experiment and Digital Signal Processing

3.1 The Pilot Experiment:

The pilot experiment, shown in Figure 3, is a Flexible Beam Control System where the beam position and vibrations are to be regulated by a digital control system whose algorithm and parameters are to be designed and implemented by the individual student groups.

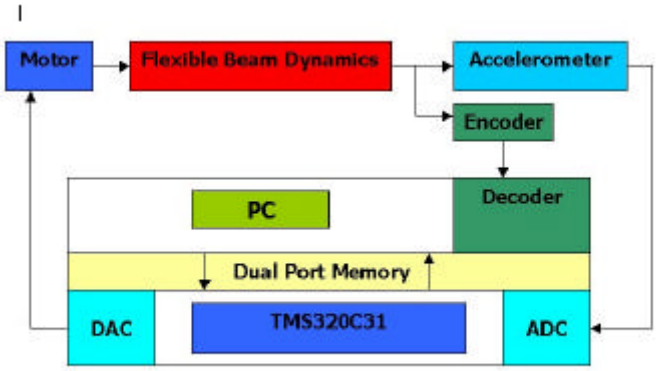

Figure 3: Pilot Experiment: Flexible Beam Control System

3.2 The PC

The PC is a standard Intel Pentium based system. The DSP card and the encoder counter card are installed in the PC as ISA add-on cards. No real time O/S is used or is necessary. Timing and synchronization are achieved in the DSP system.

3.3 The Digital Signal Processor

The DSP system used for this project was the 'Dalanco-Spry' Data Acquisition and Signal Processing Board - Model 310B. Block diagram of the Model 310B is shown in Figure 4 below:

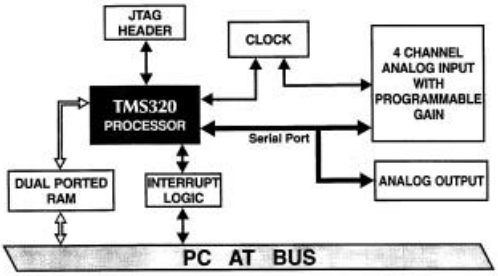

Figure 4: Architecture of the Model 310B DSP Card

The DSP board has a Texas Instruments' TMS320C31 DSP chip running at 50MHz, two 12-bit DAC, a 14-bit ADC with a four channel differential multiplexer, and 128k words of memory. The memory on the DSP board is dual ported, i.e. it is accessible at any time to the DSP as well as to the PC via the bus interface. The ADC and the DAC are however accessible only to the DSP. Any data from the ADC and to the DAC must pass through the C31. The DAC is capable of outputting at a maximum rate of 140kHz. The ADC has a maximum conversion rate of 300kHz

and has a programmable gain amplifier that gives a software programmable gain, ranging from 1 to 1000, facilitating the handling of signals with small amplitudes.

3.4 Encoder Interface

The encoders are mounted on the motor shaft. The encoder board is basically a set of counters. The input signal is filtered by a digital filter before entering the counters. The digital filter has a programmable cut-ff frequency, which may be optimized to suit each application. The filtered inputs are applied to 24-bit counters.

4. Software Development

The TI DSP software development environment supports the Common Object File Format (COFF) and ANSI C language, thus significantly simplifying code development on the PC for the DSP. The Apache server was used to handling communication while servlet server handling read and write to the server.
The software tools used in this project include:

- Apache web server and Apache JServ servlet server
- TI C compiler, assembler, linker for floating point C3x/C4x DSP.
- Borland C/C++ Complier for the PC.
- Dalanco-Spry debugger, loader, and library functions.

4.1 Web Interface

The experiment starts by first run the sefver CentralServer by

```
% java CentralServer
```

Then a client access to the following web-page:

```
<FORM METHOD=POST ACTION="/servlets/ExpPage">
<NPUT TYPE=SUBMIT VALUE="Enter">
</form>
```

When the submit buttom is clicked, the servlets server calling ExpPage.class which in return calling other classes and then creates a web page with which the main GUI interface was set. Each instance of ExpPage communicates with the CentralServer and each of which is recognized by the server as a port number. That is, the input and the response will be coded as IP address of the host with the designed port number. Thus, multiple clients for the Central Server are readily accommodated.

4.2 C Programming on the DSP

Central to the C code development for the DSP is the suitable utilization of the Common Object File Format (COFF) where each object file consists of a file header, section headers, raw data, relocation information, line numbers, a symbol table, and a string table. For the C31, the most common sections are .text (code), .data (initialized data),.bss (uninitialized variables), and name sections such as the interrupt vector table. Input object files are combined by the linker to create an executable output file according to the memory and section directives specified in the linker command file. In the linker command file, the interrupt jump table is supplied by vecs.obj which is loaded at locations 0 to 3Fh. The function c30int.obj handles all interrupts except RESET which is vectored to _ c_ int00. Linkage of this label to boot.ob j in rts30.lib enables the following tasks to be executed: 1) definition of .stack section for system stack, 2) setting up initial stack and frame pointer, 3) initialization of global variables by copying data from .cinit to .bss section, 4) setting up page pointer to .bss, and 5) calling main() to begin execution.

4.3 System Interaction

The software programs in this project ran simultaneously on both platforms: the control algorithms were executed on the DSP. The programs on the PC performed the functions of data handling and web interface. The position of the encoder is read from the encoder interface card by the PC, which is then transfer into the memory of the DSP by the PC. The DSP then accesses this position and then calculates the servo command. The hardware interface to the encoder card and the DSP card is through the PC-ISA bus. Figure 5 shows the data flow.

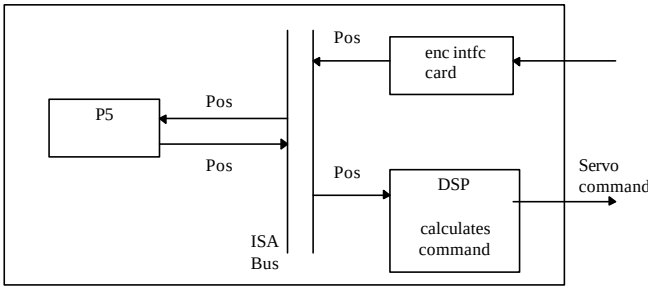

Figure 5: Data Flow Between the PC and the DSP

4.4 API for the Digital Signal Processing Card

The Dalanco-Spry Model 310 B Data Acquisition and Signal Processing card has four analog input channels and two analog output channels. An API is developed to facilitate platform independent programming. Specific functions in the API now discussed:

The Dalanco Board needs initialization at startup. The ADC connects to the serial port on the DSP. So the serial port and timer must be set up, and also the latch on the Dalanco Spry Board must be set up. For this the *InitDSP()* function is implemented. The function call prototype is

void InitDSP(void);

The output to the analog channels is written via the DAC. The function call for this is *WriteDAC(..)*. The C prototype for this is

int WriteDAC(int value, int channel);

To input analog values from the ADC the function *ReadAdc()* is called. The prototype for this function is

int ReadAdc(int channel);

This reads the value from the ADC for the voltage applied on the specified channel.

Finally, a number of interface routines are used for general I/O handling. These are:

- Grab: transfers data from DSP to PC in real time.
- int320: sends interrupt INT0 to the C31
- go320: runs the C31
- hlt320: halts the C31
- sendio: sends data from PC to the C31.
- recevio: receives data from DSP to PC.

5. Conclusions

This paper addressed the design and implementation of a web-based experiment. It differs from other existing distance experiments in that it provides efficient multi-user, near real-time interaction and a very high real-time signal processing bandwidth. A combination of Java and application programming interface are used to provide platform independent programming for the users.

6. Selected References

[1] Henry, J., "Running Laboratory Experiments via the World Wide Web," Session 3513 at the 1998 National Meeting of ASEE, Seattle, Washington.

[2] Aktan, B., Bohus, C.A., Crowl, L.A., Shor, M.H, " Distance learning applied to control engineering laboratories," IEEE Transactions on Education, Vol. 39, pp 320 – 326, Aug. 1996.

[3] Shen, H. Xu, Z., Kristiansen, V., Strom, O, and Shur, M, "Conducting Laboratory Experiments over the Internet," IEEE Transaction on Education, Vol42, pp. 180-185, August, 1999.