

Controlling A Java Enabled Pepsi® Vending Machine Over the World Wide Web.

Roger W. Webster, Ph.D. Paul W. Ross, D.E. Timothy M. Bailey
Stacey M. Conrad Michael J. Fiorill John M. Flinchbaugh Eric A. Velkly
Roger.Webster@millersville.edu
Department of Computer Science
Millersville University
Millersville, PA USA 17551

I. Abstract

This paper describes an experimental Java client-server system and a special purpose hardware interface to control a Pepsi® vending machine over the World Wide Web. This system allows users with pre-paid accounts to vend a soda from the Pepsi® machine (without any coins or bills) using a web browser such as Netscape or Internet Explorer. Anyone with a web browser may find out if any of his or her favorite sodas are left in the machine. The server software can also notify the vending corporation, via Internet email, when each brand of soda needs to be restocked. The software includes a Java applet (the client) and a multi-threaded Java server. The computer system, an Intel based Linux machine, runs its own web server at <http://pepsi.millersv.edu/> and is physically located inside the Pepsi® machine.

II. Introduction.

In 1991 the SUN Microsystems corporation funded an internal corporate research project code-named Green. The project's original objective was to develop an object-oriented programming language for software embedded in microprocessor based intelligent consumer electronic devices. The result was James Gosling's computer language called Oak, later called Java because Oak was already taken. The system described in this paper provides an interesting, experimental platform for the study of controlling a real-time electronic device using a Java client-server architecture, one of the original intentions of Gosling's work. A special purpose hardware interface to control a Pepsi® vending machine over the World Wide Web was built. This hardware and its software control the vending machine sensors that dispense the sodas.

III. The Hardware

The complete hardware system of the Pepsi® machine project consists of three components. The first is the on-board Dixie-Narco vending system built into the machine from the factory. The Dixie-Narco system processes all machine functions, including change collection and dispensing, and soda dispensing. The second component is the main computational engine, an Intel 486 computer running Linux, which handles the Internet interfacing, accounting, and control over the hardware subsystem (the third component) for Internet dispensing of the sodas. The computer system runs its own web server at <http://pepsi.millersv.edu/> and is physically located inside the Pepsi® machine. The third component is a special purpose hardware subsystem built at Millersville University, to interface to the Dixie-Narco vending computer. The main function of the hardware subsystem is to control and route the electrical power to the motors, and to decode information from the Intel/Linux machine. The hardware subsystem was assembled using off the shelf electronic components. The subsystem consists of three main components: the ISA Bus Decoder, the Optical Isolator, and the Highpower Controller.

The ISA bus decoder card simply monitors the address lines of the ISA bus on the Intel/Linux machine computer. If the card receives an address from decimal 764 through 767, an appropriate read or write data function is initialized. For the Pepsi® machine only the address 766 needs to be implemented. Sending an ASCII character to this address will cause activation of corresponding dispensing motors. Motor actuator numbers are coded as the following integers: 1, 2, 4, 8, 16, 32, 64, or 128 corresponding to bins numbered [0...7]. The program computes the integer N to send to the

hardware using the formula, $2^x = N$ where x is the number of the soda to be dispensed in the range of [0... 7], inclusive. For example, a user who wishes to dispense a soda from bin number one inside the Pepsi® machine would activate a dispensing motor by sending the integer $N=2$ to address 766. It was interesting to note that during early debugging the server was erroneously sending a number outside the range [0...7]. This causes the activation of multiple motors. This may also cause machine jams and may exceed the physical and electrical characteristics of both the machine and the custom hardware. The server software now prevents any writes to the address 766 of any integer that is not in the list (1,2,4,16,32,64,128).

The second component of the hardware subsystem is the optical isolator board. The primary function of this board is to isolate the data signals from the Dixie-Narco computer from those of the ISA bus decoder. The board consists of eight photocells each having two light emitting diodes (LED) directed toward the cell. The ISA decoder bus controls eight of the LEDs. Upon receiving the ASCII character on address 766, the number's binary representation will light up on the eight corresponding LEDs. These LEDs cause the photocell to conduct a small amount of electricity which powers the transistor array located on the high-power switch board.

The computer must power the LEDs for exactly 500 milliseconds. Sending a pulse substantially longer than the 500-millisecond specification can cause the soda machine to dispense more than one soda. Additionally, sending a pulse substantially shorter than the 500-millisecond specification can result in no product being dispensed at all. After this time an ASCII null must be sent to the board. This causes the motor to rotate far enough to activate the cam sensor switch that monitors the motor rotation. The cam sensor switches are native to the machine. The Dixie-Narco computer uses the switch to monitor soda drops. The Dixie-Narco computer uses the normally closed contacts of these switches. To avoid interference with the Intel/Linux computer, we power the LEDs off the normally open contacts of the switches. Once a given switch is activated, it powers the second LED for its corresponding circuit. This causes the photocell to continue conducting until the motor has rotated far enough to dispense one soda. At that time the switch returns back to its normal positions, the LED turns off, and the motor stops turning. While the power is buffered in the Dixie-

Narco computer, it is not used to power the small relays, as this can exceed the electrical sourcing specifications of the internal buffer chips. The signal from the switches going into the LEDs is a pulse signal with an estimated duty-cycle of 25. Thus we connect the anode of the LED to a five volt regulated positive output from the Dixie-Narco power supply. The cathode is connected in parallel to a current limiting resistor, and then the resistor is connected to the switch. Because of the slow response rate of the photocell, no filters are necessary for the LED to prevent oscillation of the output circuitry.

The photocell is also connected to the input of the tri-state buffer of the ISA card. This allows the Intel/Linux machine computer to monitor the status of the motors on address 766 for processes such as accounting and inventory control. To produce logical output codes, while sending the proper high/low control voltages to the transistor array (described later) the photocell actually puts out a high voltage, and has a 'pull down' resistor connected to ground. The result is querying port 766 will produce a null ASCII character if no motors are spinning, and produce an ASCII integer corresponding to the binary addition of all motors engaged. For example, if motor 4 is rotating, a query would result in the integer 16 which is the result of where $x=4$.

The final board of the subsystem is the highpower controller board. It is the function of this board to control the 120-volt signals that power the eight soda dispensing motors. The board consists of eight 7 amps at 120 VAC relays. The power for the relays comes from the 12-volt supply of the computer. The wires are connected to the 12-volt supply via the ISA decoder card. A sizable capacitor is necessary across the high-power board to prevent data errors in the latches of the ISA card. Also, each relay has a diode across its power terminals to prevent a loss of data integrity throughout the system when the relay's coil is powered down. The high-power controller can be powered by its 120/240 volt power supply as a stand-alone unit. However, powering the transformer up while it is connected to the ISA card can cause damage to various parts of the Intel/Linux machine and ISA decoder card. The relays on the highpower board are not switched directly by the photocells rather, they are buffered using an octal transistor array.

IV. Java Pepsi® Machine Client Software.

The user interface to the Pepsi® machine is a Java applet (client). The Java applet is first loaded into the web browser (the Java class file and images). The Java bytecodes are then interpreted and the code is executed. The front panel of the Java applet (figure 1) is a jpeg picture simulating the front of an actual Pepsi® machine. The right side of the image shows the soda selection buttons. If a particular soda is out of stock its button will have a red light illuminated, as is done on the front panel of an actual Pepsi® machine. The current count of sodas is determined by calling the `GetCurrentCount()` method which queries the actual number of sodas remaining in the Pepsi® machine via a socket to the Pepsi® server (running on `pepsi.millersv.edu`). The `GetCurrentCount()` method connects the client to the server via a socket and creates I/O streams. It then sends the ASCII text keyword "available" to the server through the socket, waits for a response, and reads in the number of sodas remaining in the machine. For example, a string such as `8:20:20:54:8:0:5` shows there are 8 Pepsi, 20 diet Pepsi, 20 slice, 54 Dr. Pepper, 8 Lipton Iced Tea, 0 Mountain Dew, and 5 Schweppes. The user can also click on the coin return icon to show the availability of sodas in the machine (see figure 1).

The main event handler that provides user interaction is the `Action()` method. At the press of any button (Pepsi, Diet Pepsi, Slice, Dr. Pepper, Lipton Iced Tea, Mountain Dew, Schweppes) on the front panel in the applet, the user name and password are requested via a pop-up dialog box (see figure 2). The user name and password is then sent along with the number of the selected button (soda) to the server for verification. The `ReadReturnMessage()` method reads the string returned from the server and parses out the return code (100, 500, 510, 520, and 530). This is performed via an output stream and a socket to the server. When the server returns the reply, the appropriate return message is displayed in a `PopUpWindow` on the client machine. If this return code shows a successful dispensing of a soda on the server then a pop-up affirmation window is shown.

V. The Pepsi® Machine Server Software.

The Pepsi® machine server software coordinates all aspects of the networked Pepsi® machine. The server accepts connections from the Java client to receive commands, verify the account, and send commands to the hardware. The server also



Figure 1. The Java applet showing the availability of sodas in the Pepsi machine.

maintains the list of available sodas. The server code is a multi-threaded Java application. The main server reads a list of counts from a file on disk on start up, initializes a network socket, names and binds the socket to a port number, and sets up a listen queue. The listen queue allows the server to accept one connection at a time and let the others wait without being denied the connection.

After successfully setting up the socket, the server spawns a "watcher" thread to monitor the hardware while the main server can continue executing and accepting connections. Each connection requests information or an action. For example, a client may query the availability of the sodas in the machine, which returns a list from an array of integers maintained within the server. Another example is that the Java client sends the user's login, password, and requested selection over the connection. The syntax of this request is checked, then the login and password are passed to the verify method. Once



Figure 2. Java applet showing the login pop-up window.

verified, the main server will send the command to drop the user's selection. The user's account is then decremented. The server decrements the count of the reported soda upon receiving a message from the watcher. The watcher thread is responsible for reading the data from the hardware and reporting it to the server. The watcher thread polls the hardware for data, and when it sees a value of 1, 2, 4, 8, 16, 32, 64, or 128, it reports that the respective soda number [0 ... 7] has been dropped from the machine. Once it has been determined that a soda has been physically dropped, it then sends "dropped N", where N is the number of the soda that was vended.

Another method implemented in the server is the verify method. This is a simple routine that reads the pepsi.pwd password file, and return the user's account status. It returns the constants INVALID for invalid logins, OUTFECASH for accounts, which haven't enough credit to order, and 0 for valid logins. The password file consists of the login name, encrypted password, and account value. Passwords are encrypted and verified with the standard Unix function call crypt. The write method actually writes to the ISA card in the computer. The program computes the integer N to send to the hardware using

the formula, $2^x = N$ where x is the number of the soda to be dispensed in the range of [0... 7], inclusive.

The server also controls the rotator motors inside the Pepsi® machine. The server's "write" method opens the /dev/port Linux device interface and seeks to the 766 address, writes the number to that port, and then closes the interface. This sets the motor into motion to dispense the desired soda. 500 milliseconds later, it sends a 0 to the interface to relinquish the computer's control of the spinning motor. The server's "decacct" method decrements the user's account after the selection has been successfully requested. It reads and parses the password file, pepsi.pwd, rewriting the user's line with a new value (oldvalue - PRICE).

VI. Conclusion.

This paper has described an experimental Java client-server system and a special purpose hardware interface to control a Pepsi® vending machine over the World Wide Web. This system allows users with pre-paid accounts to vend a soda from the Pepsi® machine (without any coins or bills) using a web browser such as Netscape or Internet Explorer. Anyone with a web browser may find out if any of his or her favorite sodas are left in the machine. The Java server can notify the vending corporation, via email, when each soda needs to be restocked. A special purpose hardware subsystem (see figure 3) built at Millersville University, to interface to the Dixie-Narco vending computer inside the Pepsi® machine has also been described. Further information can be found at our web site: <http://cs.millersv.edu/javapepsi/index.html>.

VII. Acknowledgements.

This project was funded, in part, by the National Science Foundation under grant numbers DUE-9350841 and DUE-9651237, and by the Faculty Grants Committee of Millersville University. The authors would like to thank the following people for technical assistance: Shannon Aldinger, Millersville University computer science student, Howard Wayt of Dixie-Narco Company and Bob Beers of the Pepsi® Corporation. Parts and supplies were provide by: Dr. Ken DeLucca, Department of Industrial Technology, and Dr. Joseph Grosh, Department of Physics, of Millersville University. Many thanks go to Mrs. Bonnie Work, for administrative assistance. Pepsi® Cola Incorporated and Bob Slabinski, director of Student Services at Millersville

University, provided the Pepsi® machine. Special thanks go to Rick Fritz and Mike Ondisco for Internet technical support and setting up the DNS entry pepsi.millersv.edu.

VIII. Bibliography.

Courtois, Todd, "Java 1.1 Networking and Communications", First Edition, Prentice Hall Press, Upper Saddle, New Jersey, 1998.
"Dixie-Narco Series II Electronics Technical Reference Manual", Dixie-Narco Company, San Francisco, CA, 1993.

Flanagan, David, "Java in a Nutshell", First Edition (JDK 1.0), O'Reilly Press, Sebastopol, CA, 1996.

Flanagan, David, "Java in a Nutshell", Second Edition (JDK 1.1) O'Reilly Press, Sebastopol, CA, 1997.

Geary, David M., "Graphic Java 1.1 - Mastering the AWT", Second Edition, Sun Microsystems Press, Mountain View, CA, 1997.

Hartley, Stephen, "Concurrent Programming Using the Java Programming Language", Oxford University Press, New York, N.Y., 1998.

Oaks, Scott and Wong, Henry, "Java Threads", O'Reilly Press, Sebastopol, CA, 1997.

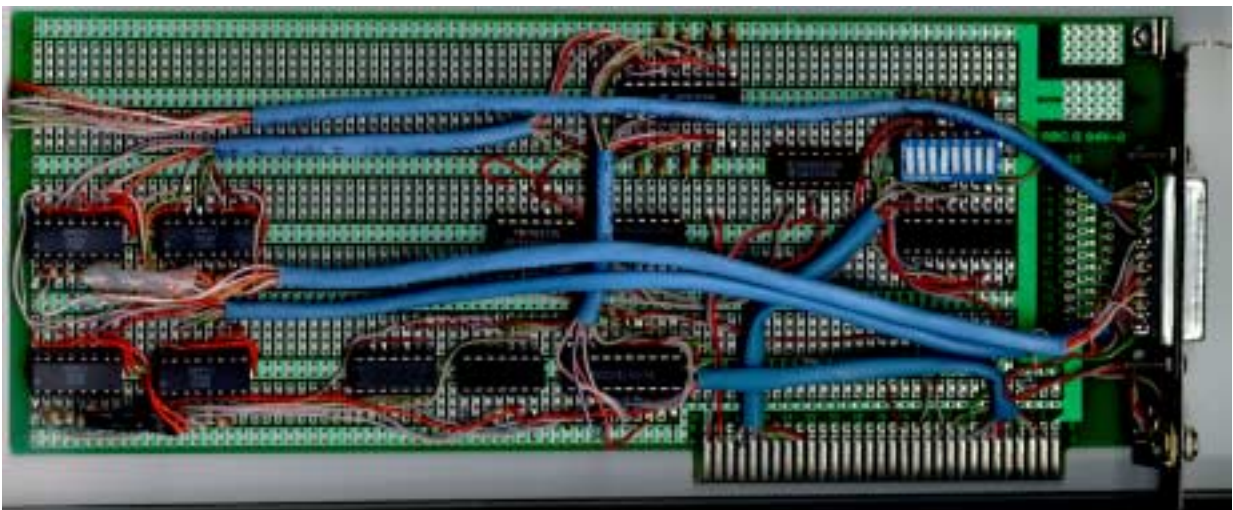


Figure 3. ISA bus decoder hardware interface built at Millersville University to control Pepsi vending machine.