WEB BASED TELEMATICS APPLICATION USING OPEN-SOURCE TECHNOLOGIES

Ionut Dinulescu, Dorin Popescu,
Gabriel Terejanu, Andras Marinescu

University of Craiova, Faculty of Automation, Computers & Electronics
Dept. of Automation and Mechatronics
107, Decebal Street, 200440, Craiova, Romania
E -mail: dorinp@robotics.ucv.ro

Abstract: The aim of the work was to develop the hardware and software structures to be used in an e-teaching context. A web based application has been developed, which links a programmable logic controller (PLC) with a computer for programming via Internet. The main requirement of our application is to provide a friendly user interface that allows PLC application developers to control the devices from anywhere in the world via a thin client that does not need the installation of any additional software on the user side. After considering several alternatives, it has been opted for web based open-source technologies that considerably reduced the implementation costs while accomplishing the other project requirements.

Keywords: telematics, web based application, programmable logic controller, open-source technologies.

1. INTRODUCTION

The high cost necessary to carry out experiments (for didactical purposes) with manipulators and robots in various environments led to the development of remote facilities where the physical system, sensors and operating environment can be at a great distance. Some researchers achieved an analysis of the current challenges in Internet tele-programming and tried to find the possible solutions under the current environment, where there are some unsolved common problems associated with this new technology such as limited bandwidth and unreliable signaling (Backes *et al.*, 2000; Belmonte and Sanchez, 2001; Taylor and Dalton, 1997).

Remote laboratories are laboratory experiments that run remotely via a web interface. Usually, either the student can set some parameters on the web, then a software interface converts those parameters to a form that is accepted by the local computer running the experiment or he can tele-operate an equipment (Berntzen *et al.* 2001; Ewald *et al.,* 2000; Grange *et al.*, 2000; Hutzel, 2001; Popescu and Schilling, 2003; Schilling *et al.*, 1997).

We proposed more, namely the student can achieve a program, which can run on the PLC. The student has thorough freedom to choose what experiment wants to do. The goal is remote programming of a PLC, which controls a robotic manipulator for use in tele-education. The robotic manipulator is controlled through PLC and Internet connection.

Web-based tele-robotic systems have only become available on the Internet in the last decade. In the beginning they utilized a CGI interface to access the robots (Backes *et al.*, 2000; Schilling *et al.*, 1997). With the introduction of Java and its integration into web browsers, developers could create tele-operated systems that sustained an interactive link to the robot during its execution. Many new tele-robotic systems were created that gave the user much more control and provided functionality, which under CGI could have never been possible (Amin *et al.*, 2001; Marin and Sanz, 2001; Popescu and Schilling, 2003).

The Internet lab technology offers the students the opportunity to work with sophisticated equipment, of the kind they are more likely to find in an industrial setting, and which may be too expensive for most faculties to purchase.

This paper gives a solution for tele-programming of a certain PLC. It tries to show an aspect of higher education and training based on Information Technology. The final purpose of the remote laboratory is to allow the student to take full control of the equipment, in order to fulfill the task required by the teacher.

2. PLC AND TELE-ROBOTIC SYSTEM

The system consists of an IDEC IZUMI Programmable Logic Controller, a modular robotic manipulator, a network video camera and a computer.

The IDEC IZUMI FA1J is a simple micro-PLC with 64 digital inputs, 64 digital outputs, 46 counters and 80 timers. The achieved editing software for IDEC IZUMI PLC permits writing programs in statement list. Our application has a help menu about instructions of IDEC IZUMI PLC and about the way to write programs.

The manipulator is a 3-axis manipulator type translation-translation-translation manufactured with pneumatic components (Fig. 1). The pick-and-place manipulator has various discrete positions for its gripper. The positions are determined by discrete signals - "on" causes the manipulator to move to one extreme position and "off" moves it to the other extreme position. The manipulator has five powered pneumatic solenoids. If all solenoids are off, no air is applied to the manipulator's actuators.

Traditionally, a manipulator is connected to a PLC via digital I/O or specific hardware interfaces. In this case, the manipulator and PLC programs can only communicate at a very low level (Popescu, 2001).

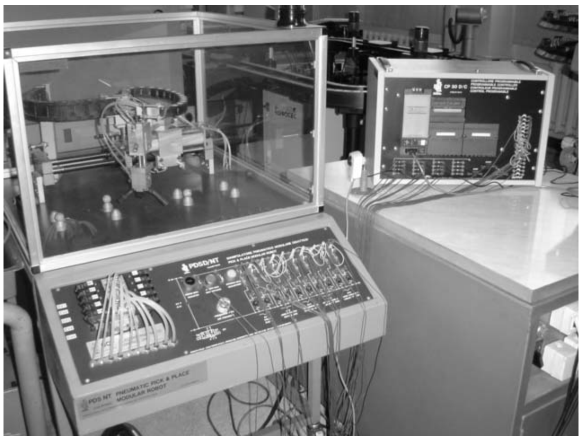
Fig. 1. PLC and robotic manipulator

The control of the manipulator is made with the IDEC IZUMI PLC (Fig. 1). The application illustrates how a PLC may be used to control a manipulator from a long distance.

3. TELEMATICS APPLICATIONS

The goal of this project in the remote programming area is to discover and develop the system by combining network technology with capabilities of PLCs and manipulators. The use of Internet technology for remote programming application offers the advantage of low-cost deployment. There is no longer a requirement for expensive purpose built equipment at each operator's location. Almost every computer connected to the Internet can be used to control a tele-operable device. The downside is the limitation of varying bandwidth and unpredictable time delays. These Internet features should be defined and considered before designing an efficient remote programming system. Besides that, several functional requirements should also be defined before designing any tele-operable system.

This web-based application allows users to perform own experiments remotely from another computer. Using a standard web browser and a connection to the Internet, the user can write his program for the PLC. After this, the user uploads his program and the results can be viewed on screen. The experiments have video streams available for visual monitoring.

3.1 Performance requirements and technologies used

The main requirement of this application is to provide a friendly user interface that allows PLC application developers to control the devices from anywhere in the world via a thin client that does not need the installation of any additional software on the user side.

Another requirement is related to the costs that are involved during the project development (e.g. development tools, hardware, etc.).

After considering several alternatives, it has been opted for web based open-source technologies that considerably reduced the implementation costs while accomplishing the other project requirements.

Thus, a bunch of open-source solutions were chosen for the application development, such as: Java, PHP, MySQL, Apache, JavaScript and HTML (McLaughlin, 2000; Welling and Thomson, 2004; Hall, 2000). In order to speed up the development process and eliminate software bugs, additional open source libraries and frameworks have also been used. The role of` each one within the whole system will be explained in the next paragraphs.

3.2 Application architecture

The general system structure is described in Fig. 2. The only software needed on the user side is a conventional web browser that can run JavaScript scripts and Java applets. This application has been tested and optimized to work on the most popular web browsers including Internet Explorer and FireFox, while other ones are currently being tested. The Apache Server is responsible for the application administration module, that involves users management, PLC program editor, etc. The functionality provided by this server is implemented by making intensive use of PHP, HTML and JavaScript. The MySQL database is mainly used for storing data about registered users, questions that will be asked within tests, and other persistent data.

The Tomcat Server maintains the communication with the PLC. This module handles user's requests such as sending a program to the PLC and monitoring the PLC's status (variables, counters and timers). The Java based technologies that are used here include Turbine, Velocity and Servlets (Kurniawan, 2002).

An AXIS web camera is used for sending live video of the process to the user's browser. This camera model includes a built-in WEB server running under Embedded Linux that transmits the video stream either via an ActiveX control or a Java applet. While the ActiveX transmission method gives better results, the Java applet is a good choice for browsers that don't support ActiveX technology (such as Opera or Konqueror).

3.3 Implementation details

Model-View-Controller design pattern
The server application that is running under Tomcat Server has been developed according to the Model-View-Controller design pattern (MVC). MVC is a modern architecture suited for interactive applications that provides a way of breaking the code in three parts: the Model, the View and the Controller. The relationship between them is presented in Fig. 3 (Buschmann *et al.*, 1996).

The Model component is the core of the application. This maintains the state and data that the application represents. When significant changes occur in the Model, it updates all of its views. The Controller receives input from the user and manipulates the Model. The user interface is represented by the View component that needs to be a registered view with the model.

There are several advantages using the MVC design pattern, including the following ones:
- clarity of design: the public methods in the model stand as an API for all the commands available to manipulate its data and state. By glancing at the model's public method list, it should be easy to understand how to control the model's behavior.
- multiple views: the application can display the state of the model in a variety of ways, and create/design them in a scalable, modular way;
- ease of growth: controllers and views can grow as the model grows; and older versions of the views and controllers can still be used as long as a common interface is maintained;

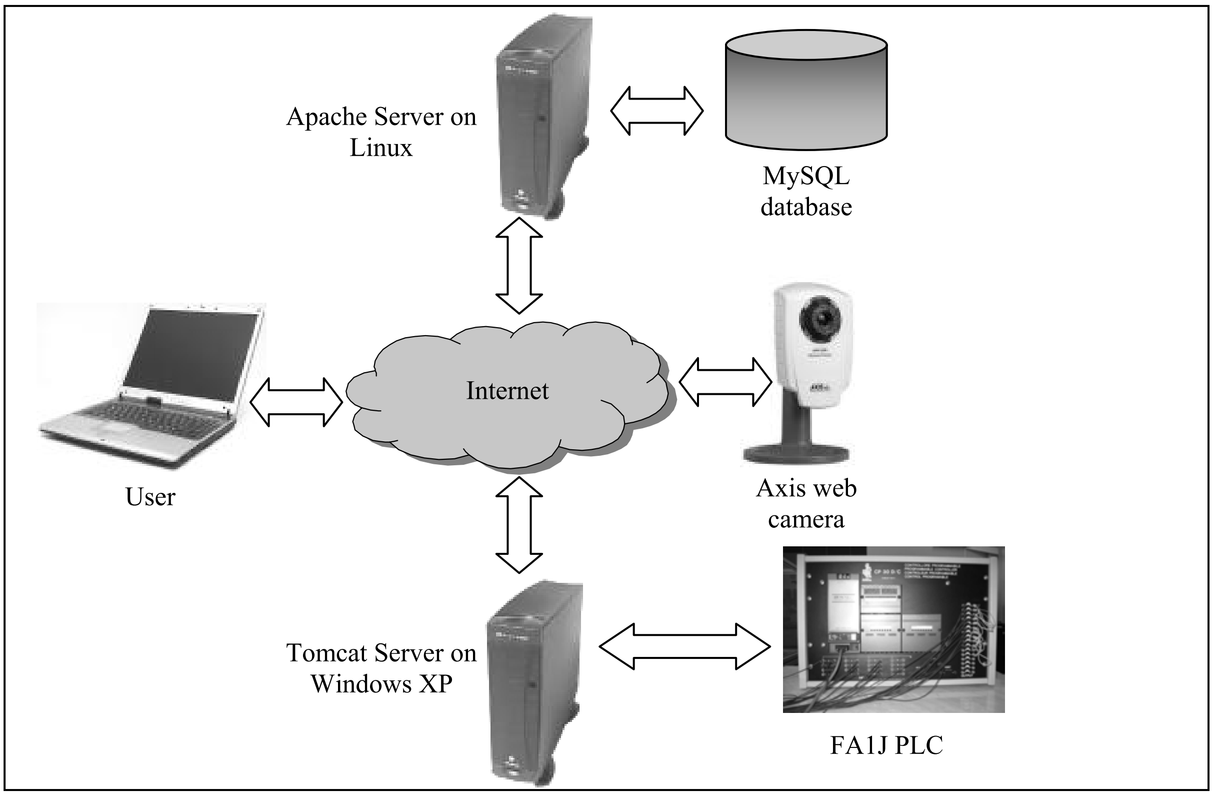

Fig. 2. General system architecture

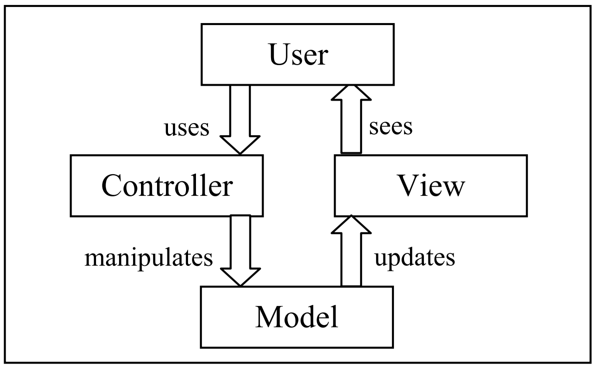

Fig. 3. Model-View-Controller

Our application's Controller is implemented by the Turbine framework. The core Turbine components that implement the Controller are Turbine Servlet and Action Event Handlers. Actions are used within Turbine to handle user input that requires interaction with the Model. An action event can be assigned to a button on a HTML page. For example, the user can press on a Start/Stop button for controlling the PLC. This button is assigned a Turbine action whose handler function will send the appropriate command to the PLC via the serial port. Similarly, there is an action that is fired every time the user sends a program to the PLC, by pressing the Send button.

The physical PLC device has a corresponding software object (the FA1JComm singleton) inside the application that handles all the communication with the PLC. The FA1JComm object has the role of Model component within the MVC architecture. FA1JComm contains public methods for different FA1J operations: sending a program, starting and stopping the PLC, monitoring variables, timers and counters, etc. Each of these methods is associated with a Turbine action event handler contained in the Controller part.

The response that is displayed inside the user's web browser is the View component that has been implemented by using Jakarta's Velocity framework. This is an open-source Java-based template engine that allows generation of web pages from predefined templates. Velocity separates Java code from the web pages, making the web site easier to maintain and allows web designers to work in parallel with Java developers, without making any changes to the Java code. Velocity is the perfect implementation of the View component and it is a viable alternative to JSP technology that implies mixing Java and HTML code.

Program editor

The module that allows the users to write their own FA1J programs has been written using the PHP language. PHP has been chosen because it has a very good built-in support for parsing character strings, which is indispensable for a language compiler. The translator takes the list of instructions from the HTML form, parses each line and, if there are no errors, it transforms the line into FA1J machine code. Checking for errors is also done on the fly, while the user is editing the program in the web form, thus eliminating the number of errors that are checked by the compiler, and reducing compilation time.

One major phase during the project analysis was to establish the machine code correspondences of each FA1J instruction. The compiler handles 2 bytes instructions in the format *MNEMONIC OPERAND*, as well as 4 bytes instructions like JMP, CNT and TIM. Currently not all of the instructions have been implemented, but the way the compiler is written, allows easy addition of new instructions, by only including them in the list of existing ones, with minimal changes of the source code. This approach allows fast implementation of different instruction sets for other PLC's that are available in the laboratory with less programming effort, by using the existing PHP code.

Serial communication

A RS-232 serial link is used for transmitting the program to the PLC and for monitoring its status. FA1J uses an asynchronous communication based on binary commands, acknowledgement codes and reply messages that are exchanged between the PLC and the PC. Establishing the communication protocol used by FA1J was a very time consuming process due to the lack of documentation on this subject. A simple C++ was created for monitoring the RX and TX channels of the serial port while the software delivered with FA1J was sending different commands to it. This allowed us to figure out what codes need to be transmitted when our software wants to send a program or to monitor different internal variables, counters or timers.

One more issue that has been encountered is happening when sending a program to the FA1J. In order to eliminate the communication errors, FA1J requires the communicating software to send it a checksum value along with the instructions in machine code format. After the program is transmitted, FA1J recalculates the checksum based on the data that has been received, compares it with the value that has been sent by the software, and decides whether the program is valid or not. The main problem was to figure out the algorithm used for the checksum calculation.

After we have solved these two problems mentioned above, we were able to implement the communication module in Java. Due to the fact that Java applications do not have direct access to the underlying hardware, we were enforced to use a Windows dynamic library that enabled us communicate with the Intel 8250 UART interface. Although it was the only way to get access to the serial port, this method bounds our Java application to the Windows operating system, and requires modifications in the code in order to port it to a free environment such as Linux.

Next, we wrapped all the PC-PLC commands in the FA1JComm singleton pattern that has only one instance through the entire application, due to the fact that the serial port cannot be opened more than once at the same time by different processes. Also, this approach is suitable for a multi-user environment like the WEB, only one user being able to send data to the PLC at a time. Taking into account that the serial port is not a preemptable resource, we are using classic synchronization objects like semaphores and Java built-in monitors to avoid simultaneous access to the serial port.

Monitor applet
A Java applet has been developed and embedded into the web page, which enables the user to watch the status of FA1J internal relays, timers and counters. The applet is integrated in the MVC architecture as a different View object. The communication with the Tomcat server is accomplished by using Java's powerful serialization support. A problem that has been encountered is related to the real time capabilities of the applet: the delay time that occurs during the transmission of the PLC's status depends on the speed of the network connection. Currently, optimizations are made in order to decrease such delays, but they will not be ever eliminated because of hardware limitations

3.4 Website specifications

The application is created around only one main file, 'index.php'. It is the one who handles the other additional classes and templates according 'op' parameter. In fact, the 'index.php' file is a big 'switch' structure which takes actions according the value passed to 'op' parameter.

In fact, the application contains many additional files grouped in three main categories:

- *program file*s are the main files of the application that include other files, that compute data and take decisions
- *classes* are a special type of files; each of them performs actions that handle table's data.
- *templates* are files that contain almost entirely HTML code that represent the design of the site; these files contain only a few PHP or JavaScript instructions necessary for a dynamic content.

The site map looks like this:

HOME
|- **admin**
|- **config**
|- **documentatie**
|- **images**
|- **js**
|- **lab**
|- **libs**
|- **menu**
|- **templates**
|- **translator**
|- index.php

The application works like this: the script receives some data, computes it and then reads the template file and replaces the variables with corresponding data creating a HTML content which will be output to the user's browser.

The script also uses sessions for security reasons. The session keeps the username, the encrypted password and the group to which the user belongs. The user and password verification is done by the *correct_login* function from *common.php* library. The test is relatively simple. If the username and password stored in session are not found in the users table, it means that the user is not correctly logged and he is redirected to the login page.

After the user passes the login phase he can select the PLC model and then he has to go to the test section. The test section selects three random questions related to the PLC model. If the answer is wrong then the script selects again three questions from the database, that are usually different from the previous three.

If the test is successfully passed then the user can enter in the programming mode that allows him to view and program the PLC in real time. The editor is displayed in a popup window.

If the logged user has administrator rights, then he can get into the administrator panel which allows him to add, edit and delete questions and answers as well as team mates. Here he can also activate users accounts, read contact messages and manage them. Something that is worth to be mentioned here is that the script has a built-in mail client created around the *contact* table. The *contact* table allows it to mark the read messages so the administrator can easily see which messages haven't yet been read.

So when the administrator clicks on a message then the script sets the value "1" in the *read* field of the corresponding record in the *contact* table.

4. CONCLUSION

This project is currently being used at University of Craiova (in the PLC class). Such remote laboratory experiment method enables the student to use laboratory equipment, which is not usually available to the students.

The application has been tested inside the faculty where a computer has been connected to Internet that the user had used to program the PLC in order to control the manipulator which was in another laboratory.

Students can watch the manipulator via the application. For the majority of remote experiments via Internet in robotics, the student can teleoperate a robot or set some parameters on the web and then he can watch the robot movements. We performed more, namely the student can write a program, which can run on the PLC in order to move the robot as he wants.

The application is provided with friendly GUI. The exercises help students to understand and compare different manipulator control methods using a PLC.

REFERENCES

Amin S., M. Zakaria, N. Majid, L. Siong, L. Horvath and J. Tar (2001). Internet-based telerobotics: UTM's experince and future direction, *The 10th International Conference on Advanced Robotics, ICAR 2001*, Budapest, Hungary, pp. 313-319.

Backes, P., K Tso, J. Norris, G. Tharp, J. Slostad, R. Bonitz and K. All (2000). Internet based operations for the Mars polar lander mission, *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, pp. 2025-2032.

Belmonte Bermudez G. and M.A. Perez Sanchez (2001). Robots Tele-programming, *1st Workshop on Robotics Education and Training, RET 2001*, Weingarten, Germany, pp. 19-24.

Berntzen R., J.O. Strandman, T. Fjeldly and M. Shur (2001). Advanced Solutions for Performing Real Experiments over the Internet, *Int. Conference on Engineering Education*, Oslo, 6B1, pp. 21-26.

Buschmann F., R. Meunier, H. Rohnert, P. Sommerlad, M. Stal (1996). *Pattern Oriented Software Architecture*, John Wiley & Sons.

Ewald H. and G.F. Page (2000). Performing Experiments by Remote Control Using the Internet, *Global J. of Engineering. Education*, **Vol. 4**, No. 3.

Flanagan D. (2001). *Java Script – The Definitive Guide*, O'Reilly.

Grange, S., T. Fong and C. Baur (2000). Effective vehicle teleoperation on the world wide web, *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, pp. 2007-2012.

Hall M. (2000). *Core Servlets and Java Server Pages*, Prentice Hall.

Hutzel W. (2001). Creating a Virtual HVAC Laboratory for Continuing/Distance Education, Int. *Conference on Engineering Education*, Oslo, 6B1, pp. 11-14.

Kurniawan B. (2002). *Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions*, New Riders Publishing.

Marin R. and P. Sanz (2001) The UJI Telerobotic Training System, *1st Workshop on Robotics Education and Training*, RET 2001, Weingarten, Germany, pp. 25-30.

McLaughlin B. (2000). *Java and XML*, O'Reilly.

Popescu D. (2001). *Programmable logic controllers*, Ed. Sitech, Craiova.

Popescu D. and K. Schilling (2003). TOM – A Remote Laboratory for Mobile Robots Education, *7th International Conference on Intelligent Engineering Systems*, Assiut, Egipt, pp. 447-453.

Schilling, K., H. Roth and R. Lieb (1997). Teleoperations of rovers – from Mars to education, *Proceedings IEEE International Symposium on Industrial Electronics*, 1, pp. 257-262.

Taylor, K. and B. Dalton (1997). Issues in Internet telerobotics, *FSR 97 International Conference on Field and Service Robotics*, pp. 37-42.

Welling L., L. Thomson (2004). *PHP and MySQL Web Development* (3rd Edition), Sams.

***, FA-1Junior Series, *Programmable Controller - Users Manual.*

***, FA1J/FA2Junior Series*, 1:1 Personal Computer Link System - Users Manual.*