Web based remote control of an electro-pneumatic process

Hannes Laget, Francisco del Valle, Fernando Tadeo
Dpto. Ingeniería de Sistemas y Automática
47005 Valladolid, Spain
fernando@autom.uva.es

Abstract

This paper presents the development of a Web-based tool to teach remotely control of electro-pneumatic process. This tool makes possible to show the basic ideas of electro-pneumatic process without the necessity of going to the laboratory. The tool is based on an Internet server that uses Labview to command two PLCs, that interfaces with the plant through Java.

Keywords

Control Education, Remote laboratories, PLCs, Electro-pneumatical plants, Internet, Java.

Introduction

During the last years, Internet has been increasingly used as a learning tool [1,2]. So, students who cannot attend lessons or need to do extra work, can login and work on-line. During the last years many people have developed different ideas: virtual laboratories [3,4], remote laboratories [5,6,7], Internet based material [8,9,10,4].

This interest on the internet for remote control is not only an academic issue: in industrial environments, many tools offer connecting the process-information to the Internet. This might help a process engineer or operator to supervise the process from any computer, including from remote areas. This means controlling and, if necessary, manipulating some process-parameters.

Although the possibilities of the Internet are extensive, it however also implies some dangers, such as the accessibility of unauthorised people (e.g. hackers) who may cause a lot of damage if a robust security system is not implemented.

This project tries to develop a remote laboratory using electro-pneumatic materials and PLCs. Compared with other remote laboratories [5,6,7 and the references therein], the possibility of remote control is not hampered by safety regulations, or the lack of any additional material (water, chemical products, etc.): intrinsically safe and reliable components (industrial electro-pneumatic components and PLCs) are used.

The plant will be used in the laboratory as a test installation for students doing practical work in automatic control. The PLC’s can be reprogrammed and the possibilities of the process adapted, depending on the level of complexity required by the subject to be studied

1. Tool description

Figure 1 shows the main components of the tool developed.

- The first component is a electro-pneumatic plant, designed with the emphasis on flexibility, so changing the PLC programs makes possible to study different pneumatic plants. To simplify the implementation, and reduce the costs and the floor space taken by the plant, the number of components that is used is limited (3 cylinders and 5/2 control valves). These valves can be operated in two modes: automatic and manual. In the automatic mode, a process-computer will send the control signals: the operator can decide the sequence in which the cylinders are moving, the number of times this sequence has to be repeated, as well as the time a cylinder stands still in an ending-position before returning to the initial position. Thus, the tool can be used to learn basic ideas in pneumatic plants.

- For controlling the plant one or more programmable PLC’s must be used. The PLC selected are Schneider TSX Nano Series, because they offer Modbus connectivity at low cost. The developed program has to support the high level of flexibility of the plant, and the interchange of the process parameters over Modbus protocol. Then, the tool can be used for demonstrations during practical exercises in programming PLC’s.

- As the plant will be remotely controlled, a few sensors are added to assure the safe operation. Other additions are failure procedures, which treat problems in the normal operation of the plant. These additions makes possible to discuss safety considerations in lectures.

- After testing the physical part, a small SCADA system is implemented in LabVIEW 6.i. This program visualizes the process state and parameters, and offers the possibility to control and change some of them. This component can be used by the student to study the development of SCADAs, and the implementation of ModBus and Internet Protocols.

- Finally a remote control is added, to make the process accessible from any computer. This is
done by programming a few small Java programs which handle the process data and makes them available to a Web server. This component makes possible to discuss remote control and Java programming to students.

- Printed Circuit Boards (PCB)
- Line Converter RS 232 -> RS 485
- External power supply and relays
- Manual board

3. The Programmable Logic Computers

The two PLC’s used in this project are:
- TSX 07 31 2408 with 14 inputs, 10 outputs (negative logic transistors), which is used to control the plant.
- TSX 07 31 1628 with 9 inputs, 7 outputs (positive logic transistors), which is used to control the manual command board.

These PLCs were selected because they are extendable, using the Modbus communication protocol.

4. Labview components

Three main components were developed in Labview:
- a) The Control Panel (Graphical User Interface)
- b) The PLC-PC communication algorithms
- c) The supervisory tool
- d) The Internet protocols

These components are now briefly described:
- a) The Control Panel

The control panel, shown in Figure 3, consists of 3 blocks, each of them with a specific function:

3. The Programmable Logic Computers

[12]

The two PLC’s used in this project are:
- TSX 07 31 2408 with 14 inputs, 10 outputs (negative logic transistors), which is used to control the plant.
- TSX 07 31 1628 with 9 inputs, 7 outputs (positive logic transistors), which is used to control the manual command board.

These PLCs were selected because they are extendable, using the Modbus communication protocol.

4. Labview components

Three main components were developed in Labview:
- a) The Control Panel (Graphical User Interface)
- b) The PLC-PC communication algorithms
- c) The supervisory tool
- d) The Internet protocols

These components are now briefly described:
- a) The Control Panel

The control panel, shown in Figure 3, consists of 3 blocks, each of them with a specific function:

a. The input and output signals:

These signals, coming both from the plant as from the manual board, are visualised by indicator lights and give the actual state of the sensors, push buttons and

Figure 1: Tool components

Figure 2: Overview of the plant with the air filter, the electro-valves, the cylinders and the sensors

Building the plant is much more than connecting the sensors and electro valves with the PLC’s and coupling the pneumatic parts with a compressor. To solve mounting problems, the following components were added:

- Printed Circuit Boards (PCB)
- Line Converter RS 232 -> RS 485
- External power supply and relays
- Manual board
electro valves. Their state is determined by the process and therefore can not be changed by the student operating the plant.

b. The control unit

The student can select a few parameters which defines the process executed by the cylinders. To create a "command", he/she can select:

- Which 4 different functions are sequentially executed (cycle)
  - move Cylinder 1 to the middle
  - move Cylinder 1 to the end
  - move Cylinder 2 to the end
  - move Cylinder 3 to the end
  - move none of the three cylinders
- The number of times this cycle has to be executed, limited to 100 repetitions.
- How long Cylinder 1 has to wait in the end position before returning to its initial position, limited to 10 secs.
- How long Cylinder 2 has to wait in the end position before returning to its initial position, limited to 10 secs.

The user can load up to 16 “commands” or “programs”. These programs are stored in the memory of the PLC (registers), and executed in the same order (FIFO). To store a program, the user has to click the “load program” button on the front panel. The user also has the possibility to stop the process and reset it. These last commands are executed immediately.

c. The actual process values

This block contains the value of the number of cycles executed in automatic mode, as well as the number of cycles executed and the number of cycles to be executed in the actual program. Pushing the button “More information and variables” shows a small window with these values. In case one of the sensors, cylinders or the air supply fails, this failure is detected and an emergency stop provoked.

b) Communication PLC-PC

Although the most logical solution would be to connect as I/O extension the second PLC (the one that drives the command panel), as it is not possible to configure the PLC TSX Nano as an I/O extension and a Modbus slave at the same time, it was decided to connect both PLC’s as ModBus slaves of the PC. The inconvenience here is that the process is half as fast as before, and all the data have to be treated by the master. The program implemented in the master (LabVIEW) is then more complex.

c) The Supervisory tool [13]

The main program in LabVIEW or VI consists of a sequence with 2 parts:

1. Once the program is started, the first step initialises the serial part and establishes the communication with the PLC’s.
2. In the second step, a ‘while loop’ is programmed, so this part will be active until the VI is stopped. In this loop, the data-interchange with each PLC’s is repeated sequentially.

In the data interchange with the PLC’s, the LabVIEW program is the Master. Because the two PLC’s cannot communicate among themselves without interference of the Master, the LabVIEW program coordinates the data transfer between them. Sequentially, the main program will read the data from PLC 2 (manual board), write these data to PLC 1 (pneumatic plant), then read the data from PLC 1 and write them to PLC 2. Meanwhile, the program visualises the most relevant of them in the GUI. The user’s commands are added to the data sent to PLC 1. Mainly, the main program is not doing more than converting the data to the right format and sorting them in the correct order to send. The real actions are programmed in subprograms, that are called by the main program. These subprograms have the following tasks:

- Calculating CRC code
- Modbus reading function
- Modbus writing function

d) LabVIEW and the Internet

LabVIEW has a built-in web server, which makes it possible to put the GUI of a LabVIEW VI on the Internet. However, in the LabVIEW Base Package, its possibilities are very limited as it is only possible to visualise the process. If necessary, a module can be added to provide interactive web pages; however this module is not for free. The most important advantage is the simplicity to make the web pages: in just a few mouse clicks (Tools – Web Publishing Tool), the process can be online. Because of the limitations, this option was not implemented [13]

Instead, a communication with Java programs was implemented, as it gives additional flexibility, and makes possible to learn Java programming by the students. The process data are stored in a text file on the hard disk. This file can be read by a Java application, which sends the information to a Java applet in a web page.
Data interchange between LabVIEW and Java programs is based on the interchange of text files. The VI writes the process data in a text file and reads the control commands from another one. At the other side, two Java servers read and write the same text files and make this data accessible to applets running in a web page. The advantages of using Java applets are that they are independent from the computer platform, and the end user does not need a LabVIEW licence.

To adapt the original program to its new needs, two extra sub-VI’s are written. One of them saves the process data in a text file. The other program reads the data from a text file. Once the data are read, the main program writes a series of zeros in the same text file to avoid that the same command will be executed twice.

5. Internet components

a) The servers

Two Java servers provide the data interchange (ServerIn, ServerOut). In fact, only the server to write the process data (ServerOut) is necessary. An applet can not write data directly on the hard disk of another computer, it can however directly access data files to read them. Introducing two servers elevates the security level, defining through which ports the communication will happen, and eventually switch off just one of them if necessary.

Figure 5: Java Server for Viewer Applet

The structure of the two servers is quite similar. Basically, a ServerSocket and a Socket are created. The server registers an available port number. The client asks to connect with the server using this port. First, the ServerSocket initializes the port and waits until a client wants a connection. Then, a Socket is fired, and an OutputStream or InputStream are created over which the server and client interchange their information. Once the transmission is completed, the server closes the connection. When they are running, the servers generate user information.

The ServerOut: this server communicates between the SCADA system and the Viewer Applet. The server waits until the client asks for a connection. Then it opens the text file, reads the process data and writes it in a string. This string is sent to the applet and the connection is closed. Meanwhile the server is running, its status is displayed in a frame.

The ServerIn: this server communicates between the SCADA system and the Control Applet. As well as the ServerOut, the ServerIn waits until the client asks for a connection. Then it reads the data from the applet, and stores them in a string. The connection with the applet is closed, and the data are written in a text file. This server also displays its status in a frame.

b) The applets

Two applets were developed for remote control:

- The Control Applet is used to implement the remote control of the process. In a GUI, the user can insert a command. This command will be sent to a server, which saves the data in a text file. Afterwards this text file will be read by the SCADA system.

- The Viewer Applet visualizes the actual process parameters. Therefore it makes a connection to a server. This server opens a text file with the process data and sends them to the applet. In the GUI of the applet appear the new parameters.

The GUI of the applets is written with the Swing toolbox of Java. This toolbox has a lot of preprogrammed elements, such as boxes, buttons, text fields, etc. and methods for the lay out of a GUI. An ActionListener is related to the buttons. This means that when the button is clicked, the program will execute a task, in this case making a connection with the server and updating the GUI.

The other part of the program code manages the data interchange. A Socket is used to create a connection with the server, and an InputStream or OutputStream is generated. Once the information is sent or received, the Socket is closed.

Control Applet: The GUI of the control applet offers the same possibilities as the LabVIEW control. The user can choose 4 commands, the number of times they are executed and the end time of cylinder 1 and 2 in their end position. Clicking on one of the three buttons (reset, emergency stop or load program), the applet will prepare a data packet. Therefore, it controls the state of the choose boxes (commands), and the content of the text fields (number of cycles, …). If the latter are incorrect, their value will be changed, and set as a default value. Once the data packet is put together, a connection to the server is made, and the information is sent.
In this project, a basic security level is built in, defining user groups which can view or control the plant. For testing the server and the remote control, Netscape 7.0 has been used.

In our project the main problems came from the communication between the different components: The data format in the SCADA system is not the same as a memory word in a PLC. The way to collocate a push button in LabVIEW differs a lot from collocating the same button in a Java applet. The main point of this project is to change the chip every time a new part is started, rather than resolving mathematical equations or making a range of simulations. More Information on the project: www.geocities.com/hlaget

Executing the same project in an industrial environment should be done by a multidisciplinary team. This will avoid spending more time on learning programming languages than on the project itself, and makes it possible to realise the whole project in just a few months. On the other hand, it requires a lot of organisation and supervising, and the manager must have a minimal knowledge of the tasks realised by the members of his team.

Without given strict limitations initially on response time, the process responds in real time at commands given through a web page. This requires a lot of different programs, interchanging strings of information. When this response time becomes important, it could be useful to change a few elements.

In comparison with the GUI in LabVIEW, the failure detection is omitted in the viewer applet. The rest of the indicators stay the same. When starting the program, a connection is made with the server to obtain the latest process information. Afterwards, the user has to push the “refresh” button to actualize the data.

The viewer applet receives the data in a text string. The applet translates the data stream into useful information and adapts the state of the indicators. The colour of the text in the labels changes (red or green when activated), and the numbers of the counters are adapted.

c) Abyss Web Server XI

The Abyss Web Server is a very user-friendly server with a lot of possibilities. On the Internet, the manual of the web server as well as the server itself can be found. In the manual there is a description of how to install the server, how to configure it, how to add user groups and define their access rights and much more [17].

Conclusion

The difficulty of implementing remote-control systems on laboratory plants is not the design of the plant, neither the implementation of the remote control on its own: Each separate part can be quite good defined and does not have a big complexity. However, each part has its specific vocabulary and way of thinking, and its own programming language.
First of all, replacing the two PLC’s by a bigger one should reduce the communication time between the SCADA system and the plant up to 50%. Another advantage could be that the process directly responds to the commands given by an operator using the manual board, without the need of running the SCADA system.

The use of Java applets does not require LabVIEW on the PC of the end user. Surely, the Java programs could be optimised (combining the two servers in one), and perhaps, a Java program can replace all of the LabVIEW programs. This program will be independent from the platform, and does not require a LabVIEW licence. Probably this may result in a cheaper and faster way to work.

Another idea for future projects is adding some components (cylinders, sensors, electrical motor, etc.) at the process thereby enlarging its flexibility. It should also be possible to write a PLC program at home, loading it in a real PLC in the laboratory over Internet and testing the proper working of it as suggested in [18].

Acknowledgments
The authors would like to acknowledge the support of JCYL through grant VA02/06.

References