World Transactions on Engineering and Technology Education
Vol.2, No.3, 2003

A virtual laboratory exercise to teach the development of industrial computer applications

George Hassapis & Niovi Pavlidou

Aristotle University of Thessaloniki
Thessaloniki, Greece

ABSTRACT: This paper addresses the issue of teaching the development of industrial control applications by utilising industrial computers. These computing devices have a modular hardware architecture that is able to receive a varying number of interfaces for sampling signals from industrial-type sensors and updating signals, which drive a varying number of actuators. They also run under the control of real-time operating systems. The development involves the elaboration of hardware and the software architectures, code generation and software testing. Teaching is done through the realisation of a process control system by using a professional suite of software tools.

INTRODUCTION

The widespread introduction of industrial computers into process plants has posed new requirements on control education [1][2]. These requirements call for combining skills in hardware configuration and software development of these specialised devices with those in traditional control system analysis and design.

Various educators believe that most of the curricula in control systems theory and engineering cover quite adequately the analysis and design aspects, but they do not expose students to the broader and practical issues of a complete control system [3]. In order to bring control systems education a step closer to real life, curricula must be enhanced with work that deals with the implementation of a control system application, which involves the use of an industrial computer.

The work presented in this article proposes an exercise that addresses all of the practical issues encountered in a real-life application of an industrial computer and a way of teaching all the phases of such a development. It is based on the use of the *IsaGraf* software engineering workbench, which is a suite of tools that professionals use to develop applications for this class of computing devices [4]. This work may be considered complementary to that presented elsewhere, which involves the use of professional process simulators and Computer-Aided Control Systems Design (CACSD) software to train students in control systems engineering [5-8].

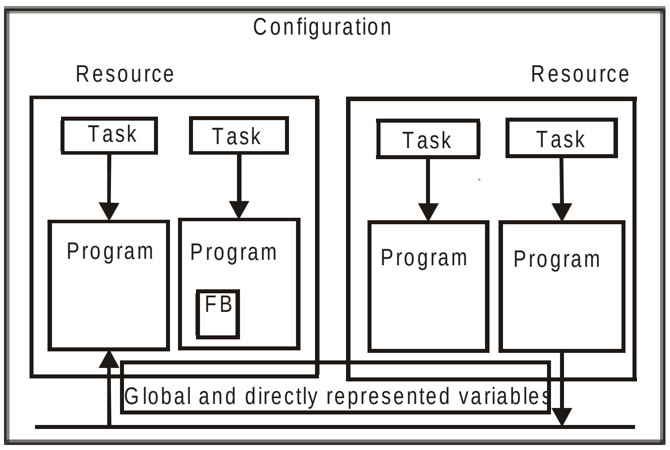
ARCHITECTURE AND PROGRAMMING OF A SOFT LOGIC COMPUTER

Industrial computers are characterised by a modular hardware architecture that consists of a small number of processing boards and a large number of input and output modules. Processing boards and input/output modules are placed on a number of frames that communicate with each other and are distributed in the industrial complex. Each frame is equipped with a back plane bus and slots, which the various modules are plugged in. Input modules are of various types, each type being able to sample either analogue or discrete signals. The analogue signals emanate from devices that measure physical quantities, such as temperature, level, flow, etc. The discrete signals come from devices that identify logical states of physical variables, such as the lowest or highest level of liquid in a vessel. Output modules provide the analogue or discrete signals for adjusting controlled variables to their desired values.

Depending on its type, an input or output module may receive or supply a specific number of similar signals from the sensing or to the actuating devices. The circuitry of each module used to receive or transmit each one of these signals is usually called a channel of the module. Typical modules are available for 8, 16 or 32 channels of discrete signals or 2 to 8 channels of analogue signals.

A real-time operating system manages the use of the computer resources, schedules the execution of the entities of the application software and provides the routines for sampling and updating the input and output modules. The sampling takes place by assigning symbolic addresses to each channel of each module. It also provides the environment to execute the application software according to the model recommended in the IEC1131-3 standard [9].

This model defines the way that software should be structured and run, and defines its main entities. It also specifies the syntax and semantics of a unified suite of programming languages that can realise these entities in software. A schematic presentation of the model is illustrated in Figure 1.



FB: Function Block

Figure 1: Software model.

According to this model, the software for a particular control problem is characterised by its architecture, which consists of resources, tasks and programs. A resource is the operating system abstract view of the computer platform, a task is a scheduling facility that executes either periodically or in response to a state change of a particular Boolean variable, while a program is the largest software entity that can be declared at the resource level.

Furthermore, a resource is characterised by its configuration, which includes the definition of the type of the processing boards, the placement in the frame of input/output modules and the assignment of symbolic addresses to the channels of the input/output modules. A program contains input, output, internal variable declarations, a body of data processing instructions and a class of software entities that are called function blocks. A function block allows a specified algorithm or set of actions to be applied to a given set of data in order to produce a new set of output data. Control over the execution of different programs and function blocks is achieved by assigning them to different tasks.

APPLICATION DEVELOPMENT METHODOLOGY

On the basis of the above model, one can write down a list of actions that the development of software for an industrial computer requires. These actions may be summarised in the following list.

- Configuring the hardware platform;
- Architecturally describing the application software;
- Coding the software entities of the application software;
- Testing the application software.

The configuration of the hardware has been explained in the previous section. Once this action is completed, the architecture of the software has to be described. This can be done by the constructs of a language as defined in the IEC 1131-3 standard, this involves practically the description of the software decomposition to tasks, programs and function blocks. It also involves the assignment of the program and function block execution to the defined tasks and the declaration of the common data objects, that is the data structures and the variables that can be accessed by any program or function block of the application software.

The coding of the software entities involves the writing of the source code of each program, function block and task of the software. Source code writing for each individual entity can be undertaken in any of the languages supported by the IEC 1131-3 standard. The last step in this work involves the compilation and correction of syntactic errors of all of the coded entities and the library routines that may have been included in these entities.

Testing the software involves verifying that the execution of the developed software will perform according to the logical functions that the designer has conceived. A simulation of the software execution on a host computer is a well-established technique for a first-level testing.

The efficient implementation of all of the above-mentioned methodological phases of the application development would require the availability of supporting software tools. *IsaGraf* is a suite of such tools, which has been developed and built to support the IEC 1131-3 recommendations [4]. One of these tools is the so-called Project Manager, which supports the implementation of the configuration of the platform and the description of the software architecture. A number of editors comprise the other set of tools, which allow the writing of the source code of the individual entities of the software in different languages. A multi-language compiler, a debugger and a target system simulator are included in this suite of tools.

The use of the *IsaGraf* suite for teaching the development of an application of an industrial computer through the study of a specific process control problem is presented below.

THE PROCESS CONTROL PROBLEM

A schematic diagram of a chemical process with its control instrumentation is shown in Figure 2. The process involves the formation of a liquid solution with specific pH value by mixing water with hydrochloric acid solution.

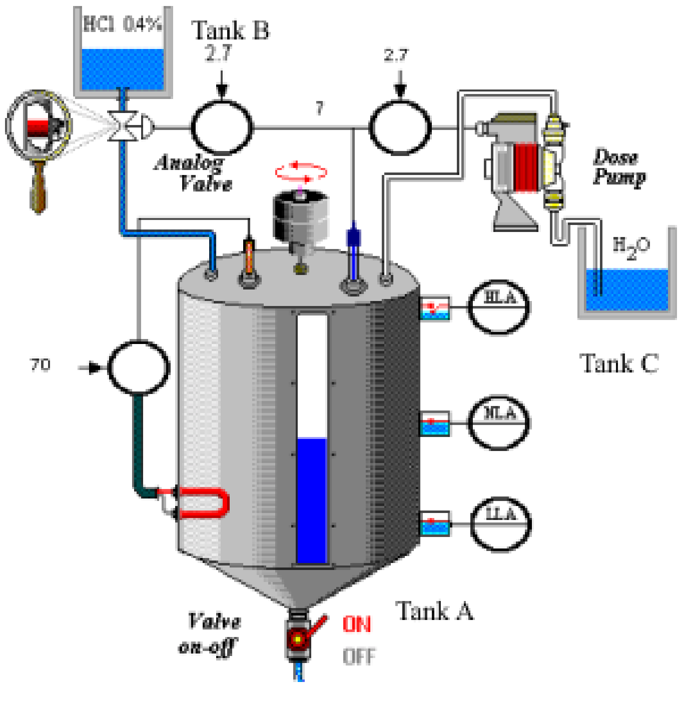


Figure 2: The chemical process of the pH adjustment of a liquid solution.

Initially, water is fed into tank A at a quantity that is equal to half of the tank's volume. Tank B contains a hydrochloric acid solution with a concentration equal to 0.4% of the overall liquid volume. Tank C is filled with water. It is required to form a homogeneous solution that will have a pH value of 2.7 at a temperature of 70ºC. The process of forming the homogeneous solution involves adjusting the volume of the hydrochloric solution and the water poured into tank A from tanks B and C, and the heating and stirring of the mixture. A mixer, which is located at the top of tank A, stirs the mixture. An electric heater, which is situated at the bottom of tank A, is used to heat the mixture.

The process operation must be controlled by a system that performs the following functions:

- Control of the flow rate of the hydrochloric acid solution with the purpose of increasing the pH value.
- Control of the flow rate of water from tank C when the pH value needs to be decreased.
- Control of the solution temperature in tank A in parallel with the control of the pH value.
- Control of the stirring process of the solution in tank A.
- Identification and alarm annunciation of abnormal operating conditions, such as tank A flooding or liquid solution level being below a minimum.
- Monitoring the flow rates of the hydrochloric acid solution and the water, the pH value and the solution temperature by using bar graphs and analogue value indicators.
- Raising alarms for conditions of low liquid level (LLA) and high liquid level (HLA) in tank A.
- Bypassing the automatic mode of operation and manually controlling the flows at the outlets of tanks A, B and C and the temperature in tank A.

The control loops, the alarms, the monitoring of variables and the manual controls are realised in the following way:

- To initiate control of the liquid flow from tank B, a loop is formed by inserting a pH measurement electrode in tank A, by placing an analogue valve with a linear characteristic curve at the outlet of tank B and controlling its position through the use of a Proportional plus Integral (PI) algorithm.
- To control the water flow rate from tank C, a dosimetric pump with 2-lit/pulse-flow rate is placed at the outlet of tank C.
- To drain the contents of tank A, an on/off solenoid valve is connected to the bottom outlet of tank A.
- A J-type thermocouple is assumed to monitor the temperature changes of the solution in tank A. A heater is switched on by a silicon-controlled rectifier every time the temperature drops below the desired value and is switched off when the temperature exceeds the desired value.
- Two level switches for identifying the lowest and highest level of the liquid solution in tank A have been placed at the key positions of tank A that have been marked with the LLA and HLA symbols.
- Four on/off switches are used to set the mode of operation of the following actuators from manual to automatic and vice versa:

  - The analogue valve that controls the flow of the hydrochloric solution;
  - The dosimetric pump that controls the flow of water from tank C;
  - The solenoid valve at the bottom of tank A;
  - The heater.

THE TEACHING APPROACH TO THE APPLICATION DEVELOPMENT

In the teaching approach considered here, the student is guided to implement the process control functions by applying the methodology for the software development already presented above. For the implementation of the control functions, the student is provided with semi-tailored solutions that are required to be completed and tested.

Hardware Configuration

With regard to the computer configuration phase, a virtual computer frame is given and the student is asked to select from a library of input and output modules to fit the needs of the application and then insert them to the appropriate slots of the frame. The selection and insertion can be undertaken by simple drag-and-draw actions of the mouse in the environment of the project management tool of the *IsaGraf* suite. The library contains simulated versions of input and output modules with 8, 16 and 32 digital channels, modules with two analogue channels and modules for displaying and inserting alphanumeric data with up to 32 characters.

In order to simulate the modules, a C-code program was prepared that defines a data structure with the number of the channels of the virtual module and the type of data that each channel may accept, ie Boolean, integer, short or long real number, message or time type of data.

Referring to the specific application of the pH control, it can be easily observed in Figure 2 that the soft logic computer must be able to receive signals from two analogue sensors: the pH electrode and the temperature thermocouple. Also, it must be able to receive discrete signals from the following:

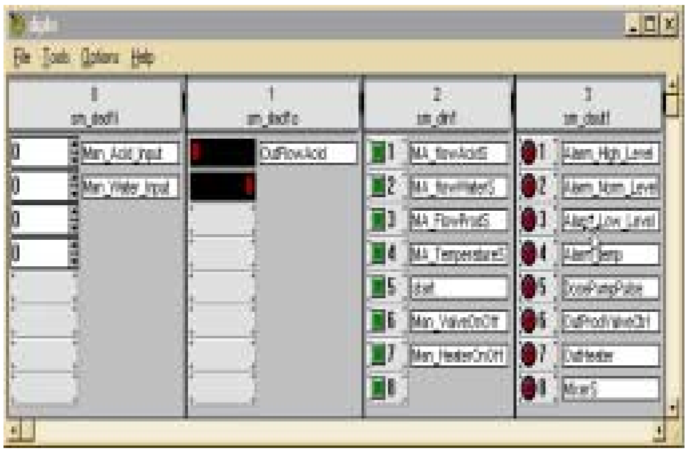
- The three level switches in tank A;
- The switches that select the manual or automatic mode of operation of the control loops of the hydrochloric acid and water flows, the temperature in tank A, and the flow of the product from the bottom of the tank A;
- The start/stop switch of the process.

It should provide one analogue output for the control of the analogue valve and eight discrete outputs as follows:

- The control of the dosimetric pump;
- The heater;
- The solenoid valve at the bottom of the tank;
- The temperature alarm;
- The alarm indicator of a low solution level (LLA);
- The alarm indicator of a high solution level (HLA);
- The indicator of the normal level condition (NLA).

Furthermore, the appropriate alphanumeric display modules must be included for the display of the flows of the hydrochloric acid solution and the water, for the display of the temperature and the pH values. A module of alphanumeric data entry must be selected in order to insert preset values for the manual control of the analogue and dosimetric pump and the set point value of the PI algorithm.

At this point, the student is expected to select the proper modules from the library of the available modules and insert them into the appropriate slots of the frame. This action must be combined later with the action of assigning the names of particular variables to the channels of each module; this takes place when defining the software architecture. The assigned variables are related with the received and sent signals from and to the process. This combined action is expected to result in a frame like that shown in Figure 3.



0 sm_dad1i	1 sm_dad1o	2 sm_din1	3 sm_dout1
0 Man_Acid_Input	OutFlowAcid	1 MA_flowAcidS	1 Alarm_High_Level
0 Man_Water_Input		2 MA_flowWaterS	2 Alarm_Norm_Level
0		3 MA_FlowProdS	3 Alarm_Low_Level
0		4 MA_TemperatureS	4 Alarm_Temp
		5 start	5 DosePumpPulse
		6 Man_ValveOnOff	6 OutProdValveCtrl
		7 Man_HeaterOnOff	7 OutHeater
		8	8 MixerS

Figure 3: Window for configuring the hardware of the industrial computer.

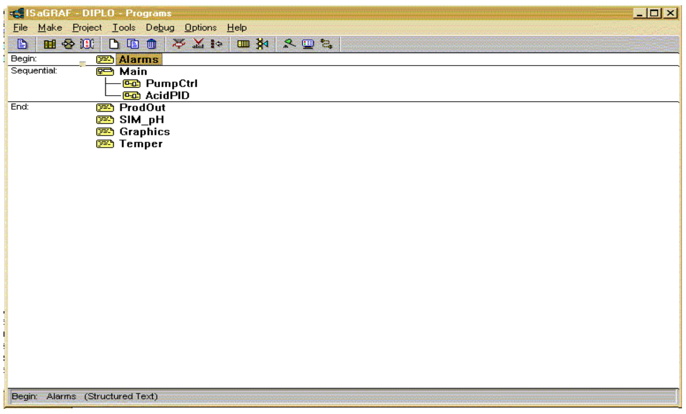
In this figure, the tags shown next to each channel of the module 'sm_din1' correspond to the names of the variables used in the programs to denote the values of signals emanating from the switches that define the manual or automatic mode of operation of the various actuators. The variable names, *MA flowAsidS*, *MA_FlowWaterS*', *MA_FlowProductS*, *MA_TemperatureS*, *Man_ValveOnOff* and *Man_HeaterOnOff* refer respectively to the switches that set in manual or automatic mode of operation the analogue valve, the dosimetric pump, the temperature controller, the solenoid valve and the heater respectively.

The *start* variable denotes the state of the start/stop switch, which initiates and interrupts the whole cycle of the pH adjustment. The module *sm_dout1* provides the eight discrete outputs of the computer. The names of the variables, *DosePumpPulse*, *OutProdValve*, *OutHeater* and *MixerS*, shown in this module, refer respectively to the signals produced for the control of the dosimetric pump, the solenoid valve, the heater, the mixer and for triggering the alarm indicators. When the temperature exceeds a limit, the output with the variable name *Alarm temp* is triggered. When a high or low or normal level of solution is monitored in tank A, the outputs with the variable names *Alarm High Level*, *Alarm Low Level* and *Alarm Norm level* are activated.

The module *sm_dad1o* provide a continuous display of the dynamic flow changes of the hydrochloric acid (OutFlowAcid), the flow of the water (OutFlowWater), the pH value (pH) and the temperature in the tank (Temperature). The module *sm_dad1i* allows the entry of the set point value in the PI control loop of the analogue valve (Set point) and the preset values for the manual control of the position of the analogue valve (Man_Acid_Input) and the dosimetric pump (Man_Water_Input).

Software Architecture

In this phase, the student is provided with a recommended software architecture, which he/she is expected to read and understand by using the project management tool of the *IsaGraf* suite. The tool translates the provided data into the source code of the architecture description language that is recommended in the IEC 1131-3 standard. The window in Figure 4 shows how the architecture is defined by use of the Project Management tool.


ISaGRAF - DIPLO - Programs
File Make Project Tools Debug Options Help

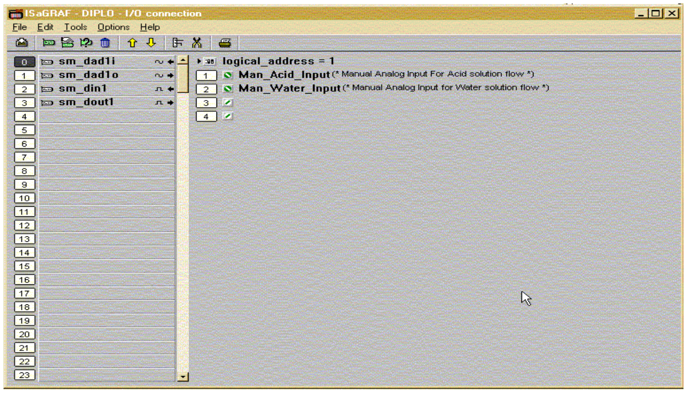
```
Begin:        Alarms
Sequential:   Main
                ├─ PumpCtrl
                └─ AcidPID
End:          ProdOut
              SIM_pH
              Graphics
              Temper
Begin:  Alarms  (Structured Text)
```

Figure 4: Graphical description of the software architecture.

Based on what was stated above, the architecture considers that the computer platform has a single resource, which runs three tasks: the Begin, Sequential and End tasks. The control and monitoring functions have been decomposed to a number of programs, some of them independent while others have a parent-child relation.

Table 1 lists the control or monitoring functions performed by each program. The next step of this phase concerns the definition of the common objects, that is, the various types of variables that are visible by all of the tasks and programs of the software. Again, the student is required to invoke the lists of these variables and identify them.

The last step involves assigning the variable names to the channels of the input and output modules of the computer. This is achieved through the graphical presentation of the addresses of the channels and the writing in the field (next to the address) of each channel the symbolic name of the appropriate variable. Figure 5 provides an example where the assignment of the variable names to the channels of the discrete input card *sm_din1* is shown.


ISaGRAF - DIPLO - I/O connection
File Edit Tools Options Help

```
0  sm_dad1i    logical_address = 1
1  sm_dad1o    1  Man_Acid_Input (* Manual Analog Input For Acid solution flow *)
2  sm_din1     2  Man_Water_Input (* Manual Analog Input for Water solution flow *)
3  sm_dout1    3
4              4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Figure 5: Assignment of variable names to the addresses of the channels of the input and output modules.

Table 1: Description of the functions performed by the software programs.

Program Name	Program Type	Description
Alarms	Parent	LLA and HLA check and annunciation
Main	Parent	Coordinates the call of the *AcidPID* and *PumpCtrl* sub-programs
AcidPID	Child	Implements the PI control of the flow of the hydrochloric acid solution to tank A
PumpCtrl	Child	Implements the flow control of the water from tank C to tank A
ProdOut	Parent	Controls the solenoid valve at the bottom of tank A
SIM_pH	Parent	pH computation - simulation of the pH electrode operation
Graphics	Parent	Imports the graphical presentation of the bar graph and pictures of the virtual instruments, prepared by the *Easy Icons* and *IconMaster* packages Updates the dynamic changes of the values of the measured quantities
Temper	Parent	Temperature computation simulation of the temperature sensor operation

In Figure 5, the selected module, which is denoted by the slot number in black at the left side of the diagram, a list of numbered channels is provided on the right side of the diagram. The student may inscribe next to each channel number the name of the variable that he/she wishes to assign to this channel. At this point, the description of the software architecture is concluded and the student is advised to proceed with writing the source code of the programs that are listed in Table 1. This can be carried out by calling the language editor, which each program is going to be written to, from the Project Management tool.

As an option, instead of writing the code, the student may obtain the full source code of the programs with comments, which can then be studied in detail in order to understand the functions that they perform. The completion of this phase is followed by the compilation phase in which all of the written codes for the architecture description and the programs are converted to executable code.

Control System Testing

The phase to test the logic of the software can be initiated once the syntactically correct executable code is produced. Simulation of the code execution is an option that is offered by the suite of tools. In this option, the developed software runs on a host computer under user-defined scenarios of input values. The simulator on the host computer imitates the way that application software is executed on a target industrial computer and provides either a pre-programmed graphical display of the monitored variables and controls of the process operation or else permits the use of a custom-made display. Through this display, a software execution scenario can be set up and the output results can be monitored by either running the software on an instruction-by-instruction basis, or by inserting breakpoints in the code and run sections of the software. These two modes of software execution allow the student to identify programs and sections of code that do not behave in an expected manner and to modify them accordingly. The student in this phase of software development can obtain the full source code of the programs with comments, which can then be studied and to improve understanding of the functions that they perform. The student is also provided with a display like that shown in Figure 6 and is asked to test the execution of the software.

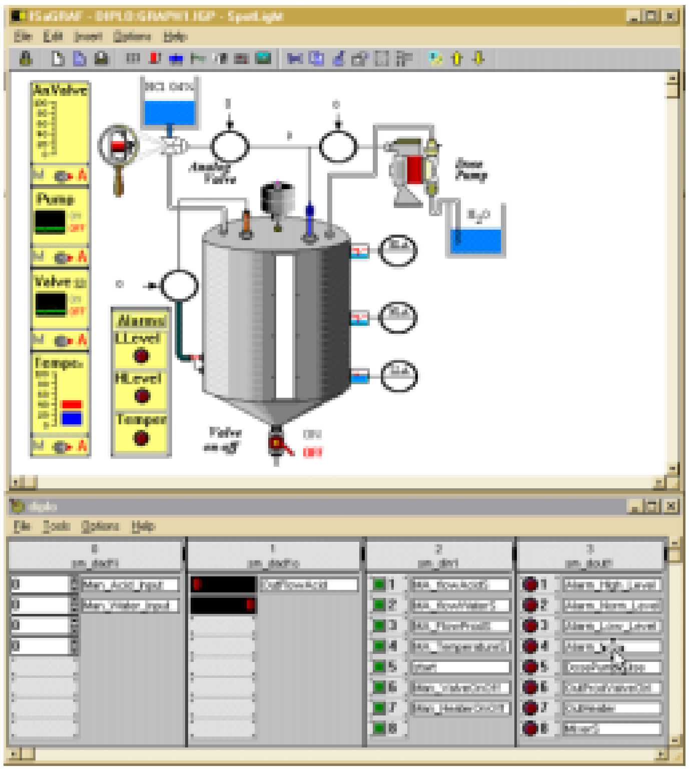


Figure 6: The display of the variables and controls of the process operation.

Although the student can work on any testing scenario, one that is recommended is where the student has to act as a typical plant operator. First, the student has to initiate the start-up procedure and verify that the developed system acts in an expected manner. Next, the student has to run the process in a normal mode of operation and, at the end, he/she has to test the alarm functions. The automatic/manual switches of the analogue valve, the heater, the dosimetric pump and the solenoid valve correspond in the start-up process to the variables *MA_flowAcidS*, *MA_flowWaterS*, *MA_flowProdS* and *MA_TemperatureS* respectively (see Figure 6), which are set to the manual mode of operation. Then, the analogue input *Man_Water_Input* is loaded with the value of 300 strokes, which is a preset value for the dosimetric pump.

In this mode of operation, one expects to see the analogue and the solenoid valves closed, the heater off and tank A to be filled with a volume of 600 litres of pure water from tank B. If the software runs correctly, by pressing the *start* button, the student can see the volume indicator of tank A rise and, when the NLA mark is reached, the dosimetric pump would shut off and the pH indicator would display the value of 7. This is the pH value of the water.

The next operator action would be to adjust manually the flow of the hydrochloric solution in order to form a solution with the desired pH value. This can be realised by loading the analogue input *Man_Acid_Input* with a value of 50% of the analogue valve opening and then observe how the pH value changes as liquid from tank C flows into tank A. When a pH value in the range of 2.5-2.7 is monitored, all of the switches are set from the *manual* to the *automatic* mode of operation. The expected software behaviour to be observed, after a short period of time, is that the pH value of the solution reaches a value in the range of 2.68-2.72, the temperature of the solution reaches a value in the range of 69-71ºC, and the mixer operation to start. The mixer operation continues until the operator gives the command to drain the contents of tank A. This command can be given by setting the manual/automatic switch of the solenoid valve to the *on* position.

During the stirring period, the pH and temperature values continue to have values in the previously mentioned ranges, as long as the level of the liquid solution in tank A is below the HLA mark. In order to test the operation of the alarm functions, the start-up procedure can be repeated but, this time, a value greater than 600 litres (eg 650 litres) is loaded into the *Man_Water_Input*. One should expect to see the level of the liquid in tank A to reach a point above the HLA mark, while the alarm indicator *Hlevel* flashes. After draining tank A to below the mark LLA, one should expect to see the alarm indicator *Llevel* to flash.

By performing the tests described here, or other tests that the student thinks of, one may verify the correctness of the developed software. The reader may carry out the experiment described in this article by accessing the site mentioned in Ref. [10].

CONCLUSIONS

This article has presented an exercise for teaching the development methodology of an industrial computer application. The exercise concerns the start-up, normal mode of operation and the alarm annunciation of abnormal operating conditions of a process that is used to adjust the pH of a chemical solution.

The exercise, through the use of a suite of software tools, demonstrates how one can implement the four phases of the methodology. The respective work involves the computer hardware configuration, the description of the software architecture, the writing of the code of the individual entities of the software, as well as its testing, by simulating its execution on a host computer.

REFERENCES

1. MotioNet.com Inc., Industrial Computer Directory (2001), http://www.motionnet.com/cgibin/search.exe?a=cat&no=213
2. Crispin, A.J., *Programmable Logic Controllers and their Engineering Application*. London: McGraw-Hill (1997).
3. Antsaklis, P., Başar, T., DeCarlo, R., McClamroch, N.H., Spong, M. and Yurkovich, S., Report on the NSF/CSS workshop on new directions in control engineering education. *IEEE Control Systems Magazine*, 19, 53 (1999).
4. Isagraf, *Isagraf* IEC 1131-3 Soft Logic, User's Guide. Seyssins: CJ International (1998).
5. Cooper, D. and Dougherty, D., Control station: an interactive simulator for process control education. *Inter. J. of Engng. Educ.*, 17, 276 (2001).
6. Shin, D., Yoon, E.S., Park, S.J. and Lee, E.S., Web-based interactive virtual laboratory system for unit operations and process systems engineering education. *Computers & Chemical Engng.*, 24, 1381-1385 (2000).
7. Blackley, J.J. and Irvine, D.A., Teaching programmable logic controllers using multimedia-based courseware. *Inter. J. of Electrical Engng. Educ*., 37, 305 (2000).
8. Copinga, G.J.C., Verhaegen, M.H.G. and Van de Ven, M.J., Toward a Web-based study support environment for teaching automatic control. *IEEE Control Systems Magazine*, 20, 8 (2000).
9. International Electrotechnical Commission, IEC-1131-3, Programmable Controllers - Part 3: Programming languages. Geneva: International Electrotechnical Commission (1993).
10. Hassapis, G., Industrial Informatics (2002), http://indinf.ee.auth.gr/myweb