A Cost Effective Solution for Development Environment for Data Acquisition, Monitoring and Simulation of PLC Controlled Applications

Ognjen Bjelica and Srđan Lale

***Abstract* — It is very important to test and monitor the operation of Programmable Logic Controller (PLC) in real time (online). Nowadays, conventional, but expensive monitoring systems for PLCs, such as Supervisory Control and Data Acquisition (SCADA) systems, software and hardware simulators (or debuggers), are widely used. This paper proposes a user friendly and cost-effective development environment for monitoring, data acquisition and online simulation of applications with PLC. The purpose of this solution is to simulate the process which is controlled by the PLC. The performances of the proposed development environment are presented on the examples of washing machine and dishwasher simulators.**

***Keywords* — Data acquisition, monitoring, PLC, simulation.**

I. Introduction

PROGRAMMABLE Logic Controller (PLC) is a special form of microprocessor-based controller which is used to control machines and processes [1]. PLCs can be used with a wide range of control systems, which vary widely in their nature and complexity. Due to its electrical properties and especially programming simplicity, a PLC is one of the most essential parts of control systems in automation industry.

After programming the PLC, before starting a real process, the operator has to verify if the program is correct, i.e. if the PLC correctly performs the predefined control task. Therefore, the operator monitors the states at input and output port of the PLC. A conventional way is to monitor all the states and program variables online, during the operation of PLC, with the same programming environment which was used for programming the PLC. In order to have a better insight into the process which is controlled by the PLC, it is best to develop a physical educational model of the system, where the operator manually enters the inputs of the PLC and monitors its output states. However, this can take a long time and more costs depending on the complexity of the system. Also, this solution isn't modular because it is difficult and time consuming to rebuild the model if some changes are necessary.

The most complex system for monitoring and control of industrial processes which is widely used nowadays is Supervisory Control and Data Acquisition (SCADA) system [2], [3]. This system requires the use of communication protocols between the client on one side and PLC and other parts of control system on the other side. Although the SCADA is a standard monitoring and data acquisition system in industry, it isn't easy to make, so it isn't appropriate in simple control systems, when it is necessary to make a simple cost-effective monitoring system in order to test whether the PLC performs its predefined task correctly.

In recent time there have been many research efforts dealing with real time simulation of different applications with PLCs. For example in [4] the software simulation is presented for a train control system which is based on serial communication between a PC and a PLC. In [5] authors used a virtual model of Tetra Pak Filling machine for the validation and demonstration of PLC program.

This paper proposes a user friendly, simple and cost-effective development environment which can be used for data acquisition, monitoring and simulation of PLC controlled applications. Using software simulations can also improve learning experience of PLC programming so it is suitable for wide audience. This solution differs from similar ones because a PC is not directly coupled with a PLC (it uses an interface board instead of PLC dependent communication). So in theory any PLC could be used in this way without modifying simulation model on PC.

The proposed system consists of two parts: the electronic board for interface between the PLC and PC and a software simulator which is a Visual Basic .NET (VB.NET) application.

This paper is organized as follows. Section II describes the electronic board for interface between the PLC and PC. The PC application is described in Section III. Experimental verification of proposed data acquisition, monitoring and simulation system is given in Section IV. Section V is the conclusion.

II. The Interface Board Between PLC and PC

The proposed development environment is intended to

Paper received March 3, 2014; revised May 12, 2014; accepted May 13, 2014. Date of publication July 31, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zorica Nikolić.

This paper is a revised and expanded version of the paper presented at the 21th Telecommunications Forum TELFOR 2013.

Ognjen Bjelica is with the Faculty of Electrical Engineering, University of East Sarajevo, Vuka Karadzica 30, 71123 East Sarajevo, BIH (phone: 00387-57-342788; e-mail: ognjen.bjelica@etf.unssa.rs.ba).

Srđan Lale is with the Faculty of Electrical Engineering, University of East Sarajevo, Vuka Karadzica 30, 71123 East Sarajevo, BIH (phone: 00387-57-342788; e-mail: srdjan.lale@etf.unssa.rs.ba).

work as follows. The operator uses a software simulator to activate certain inputs of the PLC, while the PC application receives the information about output states of the PLC and it shows them graphically, i.e. in animation. This process is realized through an interface board which communicates with the PC and controls/reads the input/output ports of the PLC. The block diagram of the interface board is shown in Fig. 1.

The common power supply DC voltage for all PLCs is +12 V or +24 V. The outputs signals of PLC (+12 V/+24 V for a logical "1" and 0 V for a logical "0") are connected to the interface board at the input of optocouplers, which are used for galvanic isolation between the PLC and microcontroller on the interface board. Except for galvanic isolation, which is very important in these applications, optocouplers were used for the adjustment of voltage levels for the microcontroller because the power supply voltage of microcontroller is typically +5 V or +3.3 V. In this way the output signals from the PLC are brought to the corresponding inputs of the microcontroller via optocouplers, so the microcontroller gets information about the output states of the PLC. This information is sent by the microcontroller to the PC over a serial RS-232 communication protocol and animated in a software simulator. Also, the operator activates, using the software simulator, the control inputs of the PLC (e.g. START and STOP signals) over the serial port of the PC and interface board.

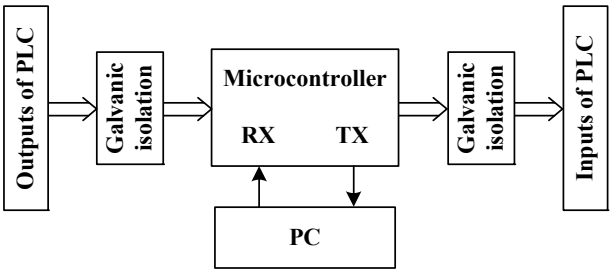

Fig. 1. The block diagram of the interface board.

The result is a simple system for monitoring and testing the operation of the PLC, where the user can start and stop the control process easily over a graphical interface and monitor its progress in real time.

The prototype of the interface board is shown in Fig. 2.

III. PLC Programs

The PLC algorithms for both washing machine and dishwasher are in principle the same. In both cases the PLC first reads inputs (input ON/OFF for power and START signal for the beginning of machine's process). A high logic value of START signal activates the timer with a predefined value of machine's process step time. After reaching the time limit of the timer, the specified counter is incremented, the timer resets and this process is continued until the end of machine's process. The current value of the counter determines the position in machine's working cycle that is used for specifying the outputs of the PLC. Depending on the states of counter and outputs, several working phases are indicated on the Human Machine Interface (HMI) display of the PLC and in the PC application. When the counter reaches some predefined final value, PLC sends a sign to stop the operation of the process and depending on its input signals (ON/OFF and START) it can begin a new working cycle. The described algorithm is shown in Fig. 3.

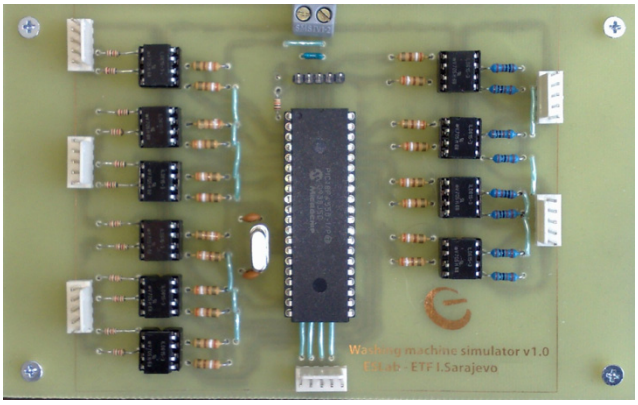

Fig. 2. The prototype of the interface board.

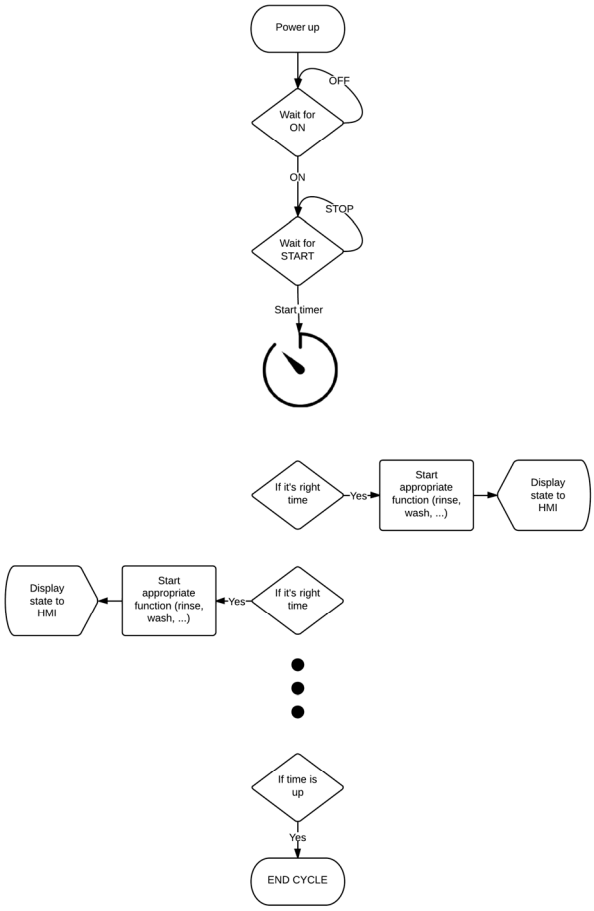

Fig. 3. The flowchart of washing machine and dishwasher.

Generally, this is the most common way for solving time based problems. Of course, this algorithm is just for the PLC which is unaware of its surrounding environment – meaning that the PLC program does not need any kind of modification whether it's working in a real or simulated system.

IV. Software Simulators

Process of developing a software simulator is quite simple because of the used serial communication. The

interface board uses two kinds of serial packages, as it is shown in Fig. 4. The first is for setting input ports and the second is for getting the states of output ports of PLC.

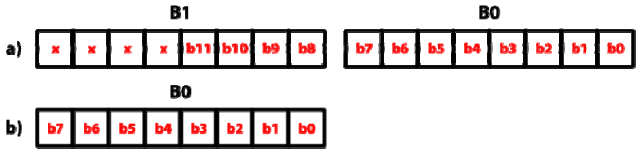

Fig. 4. a) Packet for getting output states of PLC which consists of two bytes. b) Packet for setting input ports of PLC.

So far, two software simulators have been developed just to show the principle. One is a washing machine simulator and the other is a dishwasher simulator. Software developers can use a development platform they are most familiar with. The earlier mentioned washing machine and dishwasher simulators were developed using VB.NET and Microsoft .NET framework. VB.NET was used because it is English like language so it is easier to read and understand. End users can use any language they wish – feel most comfortable with. The structure of .NET framework is shown in Fig. 5. The Windows Presentation Foundation (WPF) was chosen For Graphical User Interface(GUI) because of its many qualities, but mostly because of good animation support [6], [7].

Developing a software simulator using the WPF made this process almost trivial. The process consists of making a couple of animations (which require 0-lines of code using Microsoft Blend) and then writing a simple code which calls them if a particular state changes.

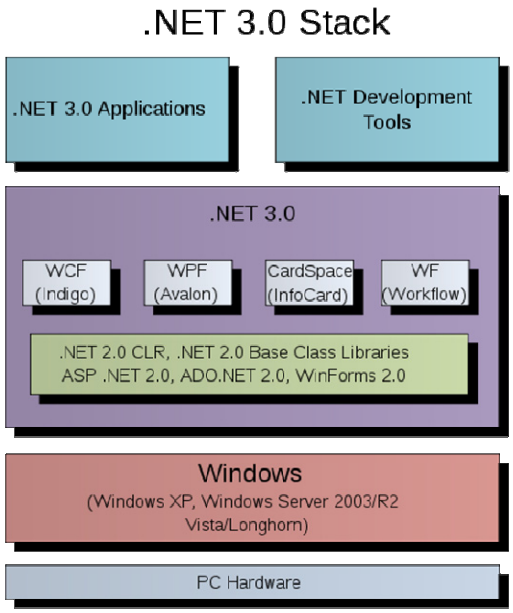

Fig. 5. The structure of .NET framework.

For example, the GORENJE PG 202 Q washing machine [8] and the Whirlpool dishwasher [9] use a timing diagram which means that changes in the program are known in advance and happen in particular time intervals. The implementation of timing diagram is left to the PLC programmer. What is only important for the simulator are the states of the machine (for example, is the heater on, or is the motor turning in either direction, etc.). After determining which states need to be animated, the next step is making animations. As mentioned earlier, the WPF was used for this process but it is not limited to it. For example, Flash or HTML5 with CSS3 could be used for making animations [10]. Fig. 6 and Fig. 7 display a finished user interface (UI) for a washing machine and dishwasher simulator, respectively. Unfortunately, animations cannot be seen in these figures. Common controls (ON/OFF buttons, buttons for changing programs, display, etc.) are also implemented and animated. Even close in rotating drum is animated. As it's demonstrated in these simple real-life devices, practically there is no limitation on what can be simulated in this way. It's worth mentioning that simulation itself doesn't need to be complicated (rotations and translation are good enough).

Fig. 6. User interface for washing machine simulator (everything is animated except for door opening).

Fig. 7. User interface for dishwasher simulator (everything is animated except for door opening).

Animation is followed by writing a simple code which gets and/or sets the states of output/input ports and, based on that, a corresponding animation starts. A pseudo code would be:

```
if PLCoutputport.Motor = ON then
   startDrumAnimation()
else
   stopDrumAnimation()
end if
```

The code for both simulators is simple because it does not simulate the entire washing machine and dishwasher but only simple parts of these machines like a drum on or drum off. The entire logic behind the simulator is presented in Fig. 8.

Using this approach it is possible to develop very simple simulators (such as a bulb, simple conveyer belt, parking lot, etc.), but it is also possible to develop very complex simulations with ease. In this way a software simulator can be developed even by people who have not much experience in software development. This aspect is

important for most students, teaching assistants, professors and engineers because they can focus on programming PLCs. Also it is worth mentioning that some parts could be generalized in order to be reused for other simulations.

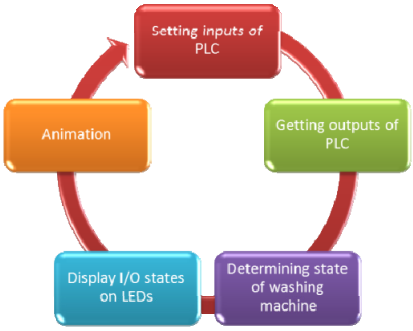

Fig. 8. Steps in software simulation.

V. Overview Of Entire System

The proposed development environment, as is shown in Fig. 9, consists of PC, PLC and interface board. The two main parts are:

- interface board,
- software simulator.

The interface board has the following key components:

- microcontroller (Microchip PIC18F4550) [11],
- optocouplers for galvanic isolation (Vishay ILD615-3) [12],
- connector for UART communication (which can be used to connect the interface board to the PC using RS- 232, USB, bluetooth or other desired protocol) – in Fig. 9 is shown the usage of additional RS-232 communication board.

In the above examples, UNITRONICS Vision 120™ OPLC™ (Graphic Operator Panel & Programmable Logic Controller) was used [13], but with no modification what so ever in software simulator, other PLCs can be used instead. The main reason why this is possible is because the system is modular and the simulator just simulates primitive parts, not the entire logic as it was stated before. Also, this solution does not require the usage of particular programming environment for PLC, instead the user can choose the programming environment based on his needs and wishes, because the interface board only uses the standard inputs and outputs of PLC. In these examples the PLC programs are implemented in VisiLogic software environment, in the form of ladder diagrams.

The response time of the entire system is limited by the following factors:

- speed of microcontroller (used on the interface board),
- baud rate of UART communication,
- logical delays of interface board (optocouplers),
- response time of developed PC application (this can vary from a multiple factor such as PC itself, used language, etc.).

For instance, the used microcontroller can easily execute its task in less than 1 ms; UART set up on 115200 baud can transfer 2B in ~0.1 ms; logical delays of interface boards are in the order of µs or ns so they are negligible; the response time of PC application is usually less than the response time of microcontroller (once again this can vary). When summed up, all these delays are of the order of a couple of ms, which is acceptable in most scenarios.

Fig. 9. The complete development environment for monitoring, data acquisition and simulation of PLC controlled applications.

VI. Conclusion

The presented solution can significantly improve the learning experience of PLC programming with minimal funds. Instead of building complex and expensive models, software simulation can and should be used. This approach has many pros and these are just some of them:

- easier to develop,
- cheaper to develop,
- once a simulator is developed it can be copied and used by thousands of students/engineers,
- safer to use and harder to destroy,
- easier to transport and set up,
- can be used in distance learning and virtual laboratories.

Also, this solution can be used for monitoring and data acquisition of PLC controlled processes, without using complex communication protocols. Using this approach can significantly speed up the development of PLC programs because a programmer can simulate his code while he is programming it and detect bugs in earlier stages, without stopping the real control process which is unacceptable in many situations.

There are many cons to this approach, but probably the most important thing about this solution is that it is vendor independent and cheap to implement.

References

[1] W. Bolton, "Programmable Logic Controllers," 5th ed., Newnes, 2009 ISBN 978-1-85617-751-1, Chapter 1.

[2] Galloway, B.; Hancke, G.P., "Introduction to Industrial Control Networks," Communications Surveys & Tutorials, IEEE , vol.15, no.2, pp.860,880, Second Quarter 2013.

[3] S. G McCrady, "Designing SCADA Application Software: A Practical Approach," 1st ed., Elsevier, 2013, ISBN 978-0124170001.

[4] K.M. Liu; D.M. Jiang, "PLC used in the train control simulation system," Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on , vol.2, no., pp.308,311, 25-27 May 2012.

[5] T. Erlandsson, M. M. Rahaman, "Testing and verifying PLC code with a virtual model of Tetra Pak Filling Machine," Master thesis report, Report No. EX016/2013, Chalmers University of Technology, Gothenburg, Sweden 2013.

[6] M. MacDonald, "Pro WPF 4.5 in VB," 3rd ed., Apress, 2008, ISBN 978-1590598221.

[7] A. Troelsen, "Pro VB 2008 and the .NET 3.5 Platform," 1st ed., Apress, 2012, ISBN 978-1430246831.

[8] Gorenje, "GORENJE PG 202 Q service manual" [Online], accessible at: *http://www.gorenje.com/*, [last time accessed on 15.09.2013.].

[9] Whirlpool, "Whirlpool Dishwasher service manual", [Online], accessible at: *http://www.whirlpool.com/*, [last time accessed on 18.02.2014.].

[10] E. Castro, "HTML5 & CSS3 Visual QuickStart Guide," 7th ed., Peachpit Press, 2011, ISBN 978-0321719614.

[11] Microchip, "PIC18F2455/2550/4455/4550 Data Sheet," [Online], accessible at: *http://www.microchip.com/*, [last time accessed on 15.09.2013].

[12] Vishay, "Optocoupler, Phototransistor Output (Dual, Quad Channel)," [Online], accessible at: *http://www.vishay.com/*, [last time accessed on 15.09.2013].

[13] Unitronics, "V120-22-UN2 Graphic Operator Panel & Programmable Logic Controller," [Online], accessible at: *http://www.unitronics.com/*, [last time accessed on 15.09.2013].