UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

RAPORT DE CERCETARE

Controller Logic Programabil (PLC) pentru Aplicații Industriale

Nicolae-Andrei Vasile

Coordonator științific:
Conf. Dr. Ing. Dan Stefan Tudose

BUCUREŞTI

2022

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT

RESEARCH REPORT

Programmable Logic Controller (PLC) for Industrial Applications

Nicolae-Andrei Vasile

Thesis advisor:
Conf. Dr. Ing. Dan Stefan Tudose

BUCHAREST

2022

TABLE OF CONTENTS

ABSTRACT

This report presents a basic overview of the industrial automation scene, alongside the fundamental component of it, The Programmable Logic Controller (PLC). We are aiming to provide a general idea of the state on which the domain is currently part of, discoveries, present approaches and trends, and future implementations and ideas.

The paper is structured in two main chapters: Introduction, where an initial description of the aspects discussed is given, and State of the Art, where we explain more in-depth the means of the industrial automation domain.

1 INTRODUCTION

1.1 Industrial Automation

Industrial automation represents the use of control systems, such as robots and specialized computers to replace human operation. It handles all the necessary processes and machineries in order to perform the given tasks, following a strict set of rules, usually abstracted by an implemented software program.

Industrial automation has been a focus of research attention in both academia and industry alike for several decades. Due to that, strong expectations are in place in the context of increased productivity, improved part quality, reduced costs, and relaxed part design constraints. The basis for these expectations is two–fold. First, machining process automation, if exercised strategically and advantageously, can perform consistently for large batch production or flexibly for small batch jobs. Secondly, process automation can be set up to autonomously tune the parameters in pursuit of desirable performance (tolerance, finish, cycle time, etc.), thereby bridging the gap between product design and process planning while reaching beyond the human operators' capability.

The success of manufacturing process automation hinges primarily on the effectiveness of process monitoring and control systems.

1.2 History

Before 1970's relays were used to control the automation processes in industrial environment. They are electro-mechanical devices used for switching higher loads. One of the limitations, such as hard wiring, configuration wearing out due to changes in wiring at each design, and so on.

As the automation processes grew more and more in complexity, debugging relay failures and changing functionality by modifying the sequence of operations became extremely time-consuming and costly because of the required rewiring, so the need for a better replacement was imminent at that time.

In 1968, Hydramatic Division of General Motors Corporation (GM) specified the design criteria for the first PLC, as described in Erickson's paper [1]. Since then, programmable logic controllers (PLC) have been the primary workhorse of industrial automation [2].

1.3 Programmable Logic Controller (PLC)

A programmable logic controller (PLC) is an electronic device used in many industries to monitor and control building systems and production processes. Unlike PCs and smartphones, which are designed to perform any number of roles, a PLC is designed to perform a single set of tasks, except under real-time constraints and with superior reliability and performance.

The architecture of the PLC is like a general-purpose computer, containing a central processing unit (CPU), power supply and input and output (I/O) modules. In fact, some of the early PLCs were computers with special I/O. However, as K. T. Erickson states in his work also [1], some important characteristics distinguish PLCs from general purpose computers. They can be placed in an industrial environment that has extreme temperatures, high humidity, electrical noise, electromagnetic interference, and mechanical vibration, making them suitable for industrial environments.

Further, the software for the PLCs can be developed using a variety of programming languages standardized in IEC 61131-3 (IEC standard for programmable controllers), and contains two textual ones: Structured Text (ST) and Instruction List (IL); and two diagrammatic ones: Function Block Diagrams (FBD) and Ladder Logic Diagrams (LLD), as described in articles [1], [2] and [3]. The ladder logic is by far the most common programming language used for developing industrial applications for PLCs [1].

1.4 Overview

In this paper we are to analyze and document the development, approaches and integration of programmable logic controllers in the industrial scene. Categories of interest are the used standards, such as IEC61131-3 and IEC 61499, the differences between them and what impact do they have in PLC design and development, programming languages (ex.: Ladder Logic, Structured Text) and methods of organizing and creating embedded software for industrial applications, hardware architectures and the general state of the industrial scene.

Further, we aim to design and develop a lightweight programmable logic controller with usages in various industrial applications following the mandatory standards and procedures relying on the existent approaches as well.

In the next chapter, the State-of-the-Art in industrial automation using PLCs is described more in-depth.

2 STATE OF THE ART

2.1 Hardware Architecture

Although they have roughly the same sort of components found in many other computer systems, PLCs operate quite differently. A PLC operating cycle, consists of the following (please note that the order can differ):

- Reading and storing the current value of each input.
- Sequentially executing the instructions in program memory, while storing any updated variables or outputs to data memory.
- Changing all physical outputs to match the output table values stored in data memory.

The main components of a typical hardware architecture of a PLC include, according to "*Programmable Logic Controllers*" [1]:

- *Central Processing Unit (CPU)*: the CPU is responsible with the implemented logic execution for a given system. The operations are executed sequentially, according to the data provided by external machines, sensors, and so on.
- *Input and Output (I/O) modules*: the I/O connects various external devices to the PLC to read and process data, for the inputs, and to change the outputs according to the logic of the application. They provide as well means to program and debug the device using various tools and protocols.
- *Power supply*: the power supply provides power to all the components of the systems for them to work properly

In a small PLC, all the described parts are enclosed in a single compact unit, in contrast to larger PLCs, that have the possibility of acquiring them separately [1].

2.2 Standards and Norms

As Vyatkin describes in his paper [3], standards and norms are very influential in the design and development of industrial automation software. They provide the technical rules that ensure the safety and performance, giving the basis as well for conformance testing. We discuss the relevant ones in this section, the most important being *IEC 61499* and *IEC 61131-3*, as described in „*Formal methods in PLC programming*" [2] as well.

- *IEC 61131-3*

  The IEC 61131-3 is the most successful standard for the industrial control software regarding PLC Programming, according to articles [2] and [3]. Various implementations of industrial applications are using it as the main source of rules and norms, such as papers [4], [5] and [6].

- *IEC 61499*

  The IEC 61499 reference architecture has been formed to facilitate the use of distributed automation intelligence in systems with decentralized logic, according to [3] and [7].

According to article [3], other notable mentions are *ISA 88/95*, intended to provide solutions to *application configuration* and *system integration* problems, *IEC 61804*, describes the specification and requirements of distributed process control systems based on function blocks, and *IEC 61850*, which addresses the interfacing issues and standardizes communication to avoid the use of vendor-specific protocols in power systems automation.

„*IEC 61499 as Enabler of Distributed and Intelligent Automation*" [7] discusses in part about the relationship between standards *IEC 61499* and *IEC 61131-3*. For example, ISaGRAF shows that it is possible to develop distributed control applications using both standards for PLC Programming.

2.3 Programming Languages

The programming languages that are to be discussed in this section conform to the international standard IEC 61131-3 for PLC Programming. As stated by Erickson [1], there are four defined languages that can be used: ladder logic, sequential function charts, function blocks and a text language (Structured text - C-like).

As many papers [1], [5], [6] and [8] show that the ladder logic is the most prevalent language. The ladder logic symbology was developed from the relay ladder logic wiring diagram. One of the reasons why the ladder logic programming language is mostly used, is because it is easy to learn [1], and the PLC software is commonly monitored and maintained by users, not by developers [3], giving it a huge advantage.

2.4 Relay Ladder Logic

Vyatkin states as well in his work [1] that the ladder logic diagram is only a symbolistic representation of the computer program. In order to properly explain how the logic ladder diagrams work, we need to introduce basic components to work with. The fundamental symbols count to 3, and are as following [1]:

- Normally open (NO) contacts
- Normally closed (NC) contacts
- Output (relays coils)

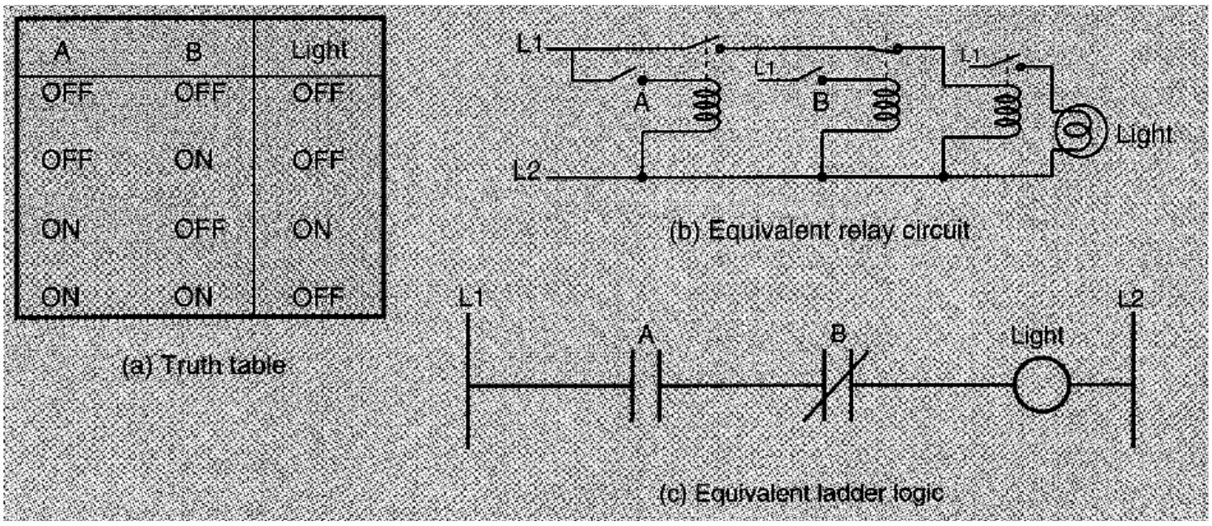
For the symbols to compose a ladder diagram, the „rung" concept needs to be defined. A rung represents the horizontal line in a diagram, between the two power rails, that contain characteristic symbols in order to create the program logic [1], [8] and [9].

Normally open (CO) are the type of contacts that do not facilitate a continuous path unless they are energized by an outer source. On the other hand, the normally closed (NC) contacts usually connect the two heads in normal state, the path being disrupted when the contact is energized. As for the output (relay coils), they are usually energized when the rung they are part of have a continuous closed path from one power rails to another [1], [8] and [9].

The logic of a usual ladder diagram flows from the left power rails to the right power rail, and from the top rung to the bottom one.

Further, we will take an example from „*Programmable Logic Controllers*" [1] to describe the workflow of an actual diagram. Let us say that we want to energize an output with two contacts: A and B. The A contact is given as normally open (NO), and the B contact as normally closed (NC) and the output should be ON when A is ON, and B is OFF.

The image below shows the truth table for out example (a), the equivalent relay logic (b) and the equivalent ladder logic (c).



A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	ON
ON	ON	OFF

Figure 2-1. Logical NOT in ladder logic [1]

The standard symbols for NO and NC contacts and output (relay coil) are used. The ladder consists of only one rung that implements the described logic. In normal state, A contact does not connect it's two pins, being normally open, in contrast to contact B. Therefore, when A is energized and B is left in normal state, a continuous path is created from one power rail to another, turning on the output.

2.5 Petri Nets

Traditionally, the RLL was developed to smooth the transition from relay control systems to PLCs. It is used as a way of capturing the sequence of operations executed by the system's control software, specify the I/O procedures of the Programmable Logic Controller (PLC) [8] and [9].

Many papers [2], [8] and [9] drew attention to the fact that, the larger the control system, the more difficult it is to determine the initial design specifications (how the system operates) by examining the control logic. The ladder logic diagrams grew more and more in complexity making the troubleshooting process extremely difficult. In addition, ladder logic diagrams are limited only to control the correspondent system [9].

Petri Nets came in the scene as a solution for the limitations of RLLs described above. Zhou and Twiss describe Petri Nets in their work [8] as being a graph-related model and a visually graphical tool designed for modeling, analysis, performance evaluation, and control of discrete event systems. Further, the capability of modeling sequential, asynchronous, and concurrent events that drive an industrial process is mentioned also in "*Comparing Ladder Logic Diagrams and Petri Nets for Sequence Controller Design Through a Discrete Manufacturing System*" [9] as well.

Petri Nets display great flexibility and understandability of the process, exceeding today's standard RLL programming techniques. Therefore, they hold a promise as a solution to modern industrial control problems, [8] and [9].

Figure 2-2 presents a basic overview of the PN model in contrast with ladder logic diagrams.

2.6 Software Engineering in Industrial Automation

The complexity and importance of software has become more and more relevant in the industrial automation domain. It is stated by Vyatkin in his work [3] that the ratio of software has increased from 20% up to 40% in the last decade, predicting a shift of the main activities of automation towards software engineering. The increasing relevance and dependability of software is sustained by "*A Real-Time Service-Oriented Architecture for Industrial Automation*" [10] also.

Software requirements are a very important part of a system design. Papers [3] and [10] describe the following as being quite important in developing industrial applications:

- *Interoperability*: The system characteristic to be able to communicate with others
- *Heterogeneity*: The capability to be diverse in content
- *Scalability*: The capacity to be changed in scale
- *Dependability*: The characteristic of systems to be reliable in any condition
- *Distribution*: The ability to decentralize the tasks over a set of distributed units

All designs take in consideration those characteristics and customize their approach and architecture according to the priority of each requirement. Various designs are proposed in this domain, but we will present three of the most important ones, as [2], [3], [10] describe as well:

- *Formal models*

  The purpose of formal models is to create dependable software for industrial systems with properties that can be guaranteed by the design, according to [3].

  Based on formal methods themselves, they support the need for high quality solutions and the application of PLC in safety-critical tasks to verify and validate certain characteristics of programs, such response times, and others [2]. In this way, the reliability of the systems in the given conditions is assured.

  However, the practical use of formal models in industrial practice is not yet common [3].

- *Multi-agent architectures*

  The multi-agent architectures put accent on modularity, scalability and flexible software design. Structure-wise, they offer an alternative solution based on decentralization of functions over a set of distributed entities [3].

  However, the existent agent architectures do not always fulfill the practical industrial requirements of a plant. It can easily reach the point where the resource usage and response time requirements become unattainable because of the overhead created by the model.

- *Service-oriented architectures*

  Service-oriented architectures came as a proposed solution to the high demand of for efficiency in machine reconfiguration and reduced time-to-market of new products [3] and [10]. These models ensure a high level of interoperability between the system's components, distribution for units of the systems, contributing to scalability and flexibility as well.

  According to [10], a bottleneck in today's industrial systems which affects efficiency and flexibility is represented by the network infrastructure between the components. Standardized protocols are making way into the industrial scene, such as Modbus, Ethernet, and so on, providing a simpler workflow for engineers. In today's systems, most of the communication networks are proprietary.

3 CONCLUSION

In this paper we have cover a basic overview of the industrial automation domain and programmable logic controllers (PLCs), current problems that we are facing today and proposed solutions to them.

Since 1970’s, PLCs have been a fundamental part in automation processes and raised attention in both industrial and academic scene, representing a research topic for decades now. Further, software engineering is expected to grow in importance in automation due to the need of more complex and reliable systems, therefore the specified domain is a large and growing area with a rich body of knowledge.

Future work consists of continuing documenting various approaches for the issues that today’s industry and academic institutions are facing, ultimately, to build a programmable logic controller on our own, using the techniques that fit our requirements and purposes most.

4 BIBLIOGRAPHY

[1] K. T. Erickson, "Programmable Logic Controllers," *IEEE Potentials,* vol. 15, no. 1, pp. 14-17, 1996.

[2] G. Frey and L. Litz, "Formal methods in PLC programming," in *IEEE International Conference on Systems, Man and Cybernetics*, Nashville, TN, USA, 2000.

[3] V. Vyatkin, "Software Engineering in Industrial Automation: State-of-the-Art Review," *IEEE Transactions on Industrial Informatics,* vol. 9, no. 3, pp. 1234-1249, 2013.

[4] D. Schütz, A. Wannagat, C. Legat and B. Vogel-Heuser, "Development of PLC-Based Software for Increasing the Dependability of Production Automation Systems," *IEEE Transactions on Industrial Informatics,* vol. 9, no. 4, pp. 2397-2404, 2013.

[5] O. G. Bellmunt, D. Montesinos-Miracle, A. Sumper, S. Galceran-Arellano and A. Sudrià-Andreu, "A Distance PLC Programming Course Employing a Remote Laboratory Based on a Flexible Manufacturing Cell," *IEEE Transactions on Education,* vol. 49, no. 2, pp. 278-284, 2006.

[6] M. G. Ioannides, "Design and Implementation of PLC-Based Monitoring Control System for Induction Motor," *IEEE Transactions on Energy Conversion,* vol. 19, no. 3, pp. 469-476, 2004.

[7] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," *IEEE Transactions on Industrial Informatics,* vol. 7, no. 4, pp. 768-781, 2011.

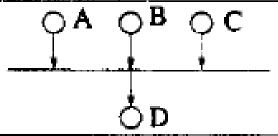
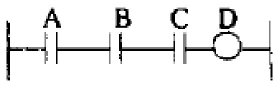
[8] M. Zhou and E. Twiss, "Design of Industrial Automated Systems Via Relay Ladder Logic Programming and Petri Nets," *IEEE Transactions on Systems, Man and Cybernetics - part C: Applications and Reviews,* vol. 28, no. 1, pp. 137-149, 1998.

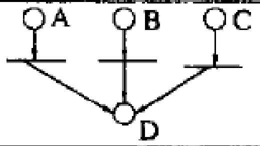
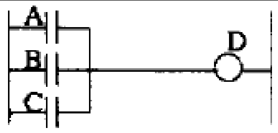
[9] K. Venkatesh, M. Zhou and R. J. Caudill, "Comparing Ladder Logic Diagrams and Petri Nets for Sequence Controller Design Through a Discrete Manufacturing System," *IEEE Transactions on Industrial Electronics,* vol. 41, no. 6, pp. 611-619, 1995.

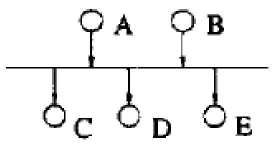
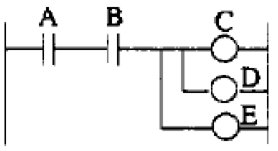
[10] T. Cucinotta, A. Mancina, G. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo and F. Rusinà, "A Real-Time Service-Oriented Architecture for Industrial Automation," *IEEE Transactions on Industrial Informatics,* vol. 5, no. 3, pp. 267-276, 2009.

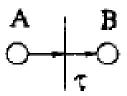
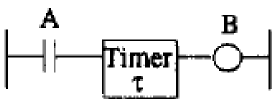
5 ANEXES

Logic constructs	Petri nets	Ladder logic diagrams
Condition or the status of a system element	○ Place	No explicit representation
An activity	——— Transition	No explicit representation
Flow of information or material.	——→ Directed arc	No explicit representation
Objects such as machines, robots, pallets, etc.	⊙ Token(s) in place(s)	No explicit representation

Logical AND
IF A = 1 and B = 1 and C = 1
THEN D = 1
		

Logical OR
IF A = 1 or B = 1 or C = 1
THEN D = 1
		

Concurrency
IF A = 1 and B = 1
THEN C = 1 and D = 1
and E = 1
		

Time delay
IF A = 1
THEN delay "τ time units"
B = 1
		

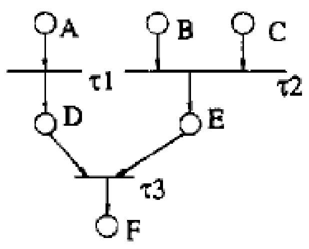
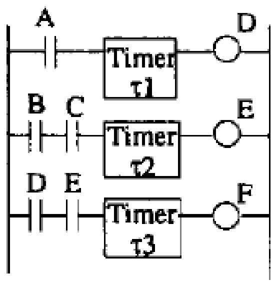
Synchronization
IF A = 1
THEN
delay "τ1 time units"; D = 1
IF B = 1 and C = 1
THEN
delay "τ2 time units"; E = 1
IF D = 1 and E = 1
THEN
delay "τ3 time units"; F = 1
		

Figure 5-1. Control logic representation by PN's and LLD's