Low-Cost Industrial Controller based on the Raspberry Pi Platform

Gustavo Vieira†, José Barbosa∗, Paulo Leitão∗ and Lucas Sakurada‡

†Federal Center of Technology Education Celso Suckow da Fonseca (CEFET), Rio de Janeiro, RJ, Brasil
Email: gustavommrv@hotmail.com
‡Federal Technological University of Paraná (UTFPR), Campus Campo Mourão, Brasil
Email: lucas-sakurada@hotmail.com

Abstract—Programmable Logic Controllers (PLCs) still are the state-of-the-art regarding the industrial automation control, but the Industry 4.0 advent is imposing new requirements, e.g., related to the capability to acquire and process data on real-time at the edge computational layer. On the other hand, the current availability of cheaper and more powerful processors opens new windows to develop low-cost and more advanced industrial controllers aligned with the Industry 4.0 principles. In this context, an important challenge is to improve the current state-of-the-art PLCs by taking into consideration the low-cost but powerful computational boards that will allow to embed IoT technologies and data analytics. This work describes the development of a low-cost but powerful industrial controller based on the use of the single-board computer Raspberry Pi, which allows executing logic control programs codified in IEC 61131-3, IEC 61499, or even in Java or Python, while maintaining the industrial requirements. The proposed platform was experimentally used to control an automation process based on a Fischertechniks platform.

Keywords: Cyber-Physical Systems, Industrial automation, Industrial Internet of Things, Programmable Logic Controller.

I. INTRODUCTION

The third industrial revolution, occurred in the late 1960s, was characterized by the introduction of robotics and ICT in industry to automate the production. Programmable Logic Controllers (PLCs) are dedicated computers responsible for controlling industrial automation machines [1] that replaced the old electromechanical systems and address the strong industrial automation requirements. PLCs were the most distinctive elements of that revolution, contributing to the improvement of efficiency and flexibility, and the reduction of maintenance costs, still being widely used in the current factories.

The fourth industrial revolution is surpassing problems that arose in the previous revolution, such as the dynamic and real-time capability to respond and adapt to condition changes. Taking advantage of the capability to analyze the huge amount of data available on the shop floor, it is possible to support monitoring, diagnosis, prediction, planning and optimization tasks. Industry 4.0, based on Cyber-Physical Systems (CPS) and several disruptive technologies, such as the Internet of Things (IoT), Big data, cloud computing and Artificial Intelligence (AI), is promoting the modernization of traditional factories by merging the physical and digital spheres that are distributed among a network. In this context, hardware and software platforms are required to support the distribution of intelligence and data analysis by cloud, fog and edge computational layers, allowing them to execute AI algorithms but also compatible with the industrial IoT technologies. The inclusion and combination of different electronic assets are possible at a higher level than was not possible before the outbreak of the newest technologies in the communication field. As examples, the availability of cheaper and more powerful processors and newest wireless networks, are allowing fast data management through accessible and smart machines leading to the creation of intelligent industrial CPS.

In this sense, the development of IoT devices operating on the edge computational level assumes a crucial role in the current days. A possible approach is to use single-board computer platforms, such as Raspberry Pi or MyRIO, which present low-cost but proper processing and communication capabilities, supporting the acquisition of data, the execution of low-level control and also the execution of some preprocessing data analytics. These new IoT devices, addressing the industrial requirements, can constitute the innovative version of PLCs, that are low-cost and offers a more powerful computational platform, aligned with the Industry 4.0 principles.

Having this in mind, the challenge is to take advantage of such open-source hardware and software platforms to achieve easy reconfigurability, enhanced autonomy, fast response, diversity of control programming languages, and interoperability between diversified control systems, without relaxing the endurance of harsh industrial environments.

This paper focuses on the development of a low-cost industrial controller system based on the single-board computer Raspberry Pi, that offers all characteristics imposed by industrial environments, such as robustness and reliability, providing at the same time a low-cost and powerful platform to control automation systems, running applications developed in different languages, e.g., the traditional IEC 61131-3 [2], the modern IEC 61499-4 [3], as well as accommodate others more
advanced, e.g. Multi-Agent Systems (MAS) through the use of the Java Agent Development Framework (JADE) framework [4], or data analytics codified in the Python language.

The rest of the paper is organized as follows. Section II describes the state of the art related to the low-cost automation field and Section III presents the architecture for the proposed low-cost industrial controller based on the Raspberry Pi platform. Section IV discusses the programming environments that can be deployed in the developed industrial controller, with particular emphasis on IEC61131-3, IEC61499 and MAS. Section V describes the use of the developed low-cost industrial controller to control an automation case study. Finally, Section VI rounds up the paper with the conclusions and points out some future work.

II. RELATED WORK

Low-cost automation is a strategic approach of engineering solutions deployed to reduce costs through existing resources on the market. The main goal is to increase the productivity and flexibility while decreasing waste with cost-effective and straightforward automation. Low-cost automation can be regarded as low-cost hardware but not necessarily with low performance. The system automation is low-cost when compared to the flagship devices on the market. However, they are disposed to accomplish tasks in environments and activities where their capacity for processing is well designed to handle, also exploiting deeply the capabilities that the hardware offers.

This methodology is also closely linked to the decentralization of control by using less powerful devices than the ones used at cloud level, which are spread throughout the shop floor, being capable to collect and process data and to communicate with each other. This feature improves the system robustness and flexibility since the process relies not only on one important-centered device. Furthermore, the modular architecture allows the direct replacement of the inoperative equipment [5].

The simple implementation of the automation system, the reduced need for programming knowledge and the application of open-source platforms are essential features in this low-cost perspective. While getting rid of proprietary communication and processing, the open-source advent allows the quick development and deployment of applications able to run in inexpensive yet reliable hardware. As an example, following those requirements, the OpenPLC software is an open-source and runtime environment that can be embedded in different systems on a chip, such as ESP8266, Arduino and Raspberry Pi [6], constituting an attractive solution for the real-time management of PLC routines running IEC 61131-3 programs. Similarly, FORTE is available to be embedded in multiple computational boards, among which are the BeagleBone Black and the Raspberry Pi as the most versatile options, supporting the execution of IEC 61499 applications.

Among the leading low-cost automation solutions, the Unipi and Iono Pi controllers [7], based on the Raspberry Pi platform, feature I/O modules that combine several digital and analog input lines, interfaces like 1-Wire, Wiegand, and power relay outputs. In the automation field, they are employed both in industrial and residential domains where robustness is a prerequisite, for instance, applied to data acquisition and control, home and building automation and environmental monitoring. The hardware is compatible with open and proprietary frameworks available for the Raspberry Pi platform. However, these approaches are mainly applied in laboratory, as described in [6], [8] and [9], being its applicability in industrial environments practically residual.

Some limitations regarding the application and use of PLCs are due to the creation of closed control ecosystems by each manufacturer. In order to hinder the transition between devices from different PLC manufacturers, the development of non-backward languages between control devices is observed. The open-source concept aims to break these barriers in both the hardware and software domains. Additionally, the need to embed intelligence and data analytics in such IoT devices requires the use of low-cost industrial controllers that support new programming languages, such as Java and Python, but also the traditional IEC61131-3 languages.

III. ARCHITECTURE OF THE LOW-COST INDUSTRIAL CONTROLLER

The architecture of the proposed low-cost industrial controller, represented in Figure 1, relies on the concept of an accessible controller through an inexpensive CPU and expansion board that plays the role of the I/O interface. Aiming to optimize the space used by the hardware, the circuitry was designed to fit properly into the region conceived for the expansion board development of the Raspberry Pi single board computer.

Figure 1. Architecture of the low-cost industrial controller.

The low-cost industrial controller was conceived to use inexpensive but matured and powerful components. The core component is the Raspberry Pi 3 Model B+ that constitutes
an efficient, robust and powerful computational platform to execute the logical control program that can be codified in different programming languages. An industrial shield, top-connected to the Raspberry Pi platform through the 40-pin GPIO, is in charge of interfacing inputs and outputs to the industrial controller brain, performing an essential function in a PLC to endure the harsh industrial environments. Particularly, it lodges all electronic components which receive the input signals and transmit the commands to the output depending on the logic embedded in the core.

The designed industrial shield board comprises eight digital ports, four of which are used as inputs and four as outputs, being possible to attach several boards in a stack to provide access to more GPIOs from the Raspberry Pi. The inputs ports are mainly composed of optocouplers that provide galvanic isolation between potential high power triggers and the low voltage CPU. Meanwhile, the output ports are formed by relays that provide magnetic isolation from the loads to the master system and ensure no overcharge. In fact, an outer short circuit can not affect the proper operation or breakdown of the controller. This separation is essential to maintain reliability, emphasizing the robustness in harsh environments that can have poor electrical maintenance.

Figure 2 illustrates the schematic representation of the input and output circuits applied and replicated throughout the available GPIO pins of the Raspberry Pi. The external input voltage supplies the optocoupler entrance, which activates the infrared LED inside the optocoupler, sending a signal to the Raspberry Pi by the phototransistor at the opto’s output. At the Raspberry Pi’s output, the 3.3V pin provides enough current to activate the relay that isolates the 24V output from 3.3V Raspberry Pi’s logic level.

Another critical tool in the low-cost industrial controller is the development and programming environment, being necessary to consider a Real-Time Environment (RTE) but also an Integrated Development Environment (IDE) that consists of a source code editor that provides the conditions to a seamless formulation of scripts or graphics programming. Since one of the assumptions in this work is to develop a platform that can support different programming languages, it was considered several tandem solutions: PLCopen Editor and Open PLC for the IEC 61131-3 programming languages, 4DIAC and FORTE for the IEC 61499 programming language, and Eclipse and Java for MAS programming. These programming environments, deeply described in the following section, serve as interaction areas between operator and PLC, including the inner deployment of the code on the machine [10].

IV. PROGRAMMING ENVIRONMENT

The programming environment consists of frameworks that make possible the implementation of the control logic. These soft environments offer a graphical user interface that enables the proper development of the circuit logic and facilitate the simple integration for the systems to be controlled since they deliver tools needed for the smooth communication between the parts of the whole system.

A. Programming with IEC 61131-3

The PLCopen is encoded by following the IEC 61131-3 standard [2] through the use of Program Organization Units (POUs), which can be present in form of programs, function blocks and functions [10]. As illustrated in the flow chart represented in Figure 3, the process from building the code to deploy the code follows some steps.

Initially, the logic is developed in one IEC 61131-3 language. The program can then be saved in the XML format regulated by the IEC 61131-10 that allows the exchange of code developed between any device compatible with an IEC 61131-3 programming language. Through a converter module, the code is converted to ST (Structured Text) and then sent to a back-end compiler belonging to the IDE. As soon as the code is compiled for C++, it is remotely deployed to the OpenPLC environment embedded in the Raspberry Pi [11].

The OpenPLC platform, embedded in the Raspberry Pi, is able to receive the control logic in the ST format from the IDE,
and to run the script in the kernel. Through a web graphic interface, the OpenPLC will access remotely the terminal where the PLCOpen is hosted, download the structured text archive, save it in its repository and compile the script. The interface allows to start and stop the routine and also the management of previously downloaded scripts.

Working accordingly to the IEC 61131-3 standard, the framework enables intercompatibility among different devices, including open-source hardware. Since the OpenPLC adopts the IEC 61131 standard strictly, the lockdown related to the proprietary communication is overcome.

B. Programming with IEC 61499

The programming environment for IEC 61499 uses the 4DIAC IDE. Initially, a system containing the application and configuration is created. At the application application level, the logic is developed using the function blocks compliant to the IEC 61499-2 standard [12]. Within the system configuration, the device to be used is chosen, and therefore, the developed logic is mapped to the resource contained in the device. The resource defines how the application will be initialized and whether it will communicate with other devices. After this last step, the code can then be deployed to the FORTE RTE embedded on the physical device.

4DIAC is a software provided by Eclipse that offers a solution for the distributed control domain, consisting of the 4DIAC IDE and FORTE as the RTE. That framework is codified accordingly to the IEC 61499-4 standard, which splits the information propagated through the logic between data and events. Designed as function blocks, the logic circuit has the events on the top triggering its functionality and the data on the bottom, storing important values that route the way logic will follow. Both the IDE and RTE communicate with each other using wireless whenever connected to the same net. 4DIAC has already the Raspberry Pi as a native device in the system, and features a convenient monitoring interface that allows the user to verify the logic control running in real-time. Watches can be added to any function block input or output to support the debugging process.

C. Programming with Multi-Agent Systems

The use of MAS technology allows decentralizing the intelligence through autonomous and cooperative agents, contributing to the realization of industrial CPS. JADE (Java Agent DEvelopment Framework) [13] is a framework based on the JAVA programming language that simplifies the development of MAS solutions, providing several functionalities, such as graphical and debug tools, libraries to define the agents behavior, the yellow pages service and mechanisms for the communication between the agents. The message sniffer, which is responsible for tracking the messages exchanged between agents, and the introspector agent, which aims to monitor the life cycle of an agent, are examples of debug tools.

The development of a MAS solution requires to use an IDE, e.g., Eclipse or NetBeans, to facilitate its implementation and to avoid possible syntax errors. After verifying the correct functioning of the agent behavior using the IDE, it is possible to embed the developed program in a computational platform, such as Raspberry Pi, through a JAR file. When considering the developed low-cost industrial controller, it is necessary to access the I/Os of the Raspberry Pi. For this purpose, the PI4J I/O API (https://pi4j.com/1.1/) is integrated in the agent code to access the Raspberry’s GPIOs. This library provides the required functions to manage and control the automation process in a flexible approach.

V. EXPERIMENTAL IMPLEMENTATION

The proposed low-cost industrial controller was used to control an automation system for testing and validation purposes.

A. Description of the Case Study

The case of study deals with the control of the Fischertechnik punching machine, illustrated in Figure 4. As a stable scan time, it was chosen 50 ms for unbiased performance purpose on each programming environment.

![Figure 4. Industrial controller performing the control of the punching machine.](image)

The punching machine comprises four inputs and four outputs, fitting the industrial shield I/O resources. The first light sensor, placed at the begin of the conveyor, detects the arrival of the part in the punching machine, which should turn on the motor to transport the part along the conveyor until it reaches the final position, detected by the second light sensor. At this moment, the second motor brings the punching tool way down to operate the part. When the bottom limit sensor is actuated, the tool is stopped, but the piece is maintained under the tool pressure. After some seconds the piece is released, the puncher comes back to its original position reaching the top limit sensor, and the belt’s motor starts running backward, returning to its starting point.

B. Programming the Logic Control using IEC 61131-3

The developed logic control for the experimental implementation of the automation process used the Function Block
diagram language (one of the IEC 61131-3 languages) and is illustrated in Figure 5.

The blocks represent the software encapsulation and are arranged in a particular order to achieve the desired control. The upper section, corresponding to the function blocks horizontally positioned between the variables PHOTO_1 and OUT_1, represents the control of the running process related to the sensitization of the initial light sensor and the drive of the conveyor belt’s motor. The lower section corresponds to the control between the sensitization of the second belt’s light sensor and the sensors presented in the vertical structure of the punching machine including its motor. Both sections are interconnected so that the beginning or ending of one section triggers the other.

As the complexity and scalability of the control increases, the need for a large number of Function Blocks is inevitable, and consequently a high number of connections between them makes the control prone to fail and difficult to maintain.

The program was codified using the PLCOpen editor and deployed in the OpenPLC RTE as described in Section IV.A.

C. Programming the Logic Control using IEC 61499

The IEC 61499 application program for the case study was developed in the 4DIAC editor and is illustrated in Figure 6. Programming using 4DIAC provides a more homogeneous and easy way to read, being the number of function blocks required for the same control significantly lower in IEC 61499 when compared to 61131-3 programming language. The interconnection of each input and output function block is denoted most clearly, the data and events are explicit and complementary concepts, and new function blocks can be created containing the desired control characteristics for each application as well as can be encapsulated in other blocks.

![Figure 5. Control program developed in IEC 61131-3 language.](image)

![Figure 6. Control program developed in IEC 61499 language.](image)

The blocks named PHOTO_1, PHOTO_2, DOWN_S and UP_S are related to sensors that are responsible for collecting information about each input section of the automation process. Some triggers take part in processing segments of the automation system as well as in the activation of actuators through the function blocks mapped to the Raspberry Pi outputs.

As previously described, the developed control program was
later deployed in the FORTE RTE environment embedded in the Raspberry Pi platform.

D. Programming the Logic Control using JADE

Another way to develop the logic control of the automation process is through the implementation of a software agent. This process involves some steps: development, deployment, and control of the agent’s behaviour. In the first step, it was used the Eclipse IDE to support the agent code development, avoiding the syntax errors and testing the code before to be embedded in the Raspberry Pi. The agent code was implemented in the JADE framework based on the Java programming language. The developed agent uses a cyclic behavior that allows to perform tasks cyclically and thus always be able to detect events or receive external commands, and execute actions accordingly.

Furthermore, in order to control the industrial shield connected to Raspberry Pi, it was necessary to use the PI4J library. This library offers some functions that allow controlling the inputs/outputs of the Raspberry Pi. In this case by using the GPIO triggers, while listen the state changes in any pin and will perform a pre-defined action. The following piece of the Java code illustrates the access of the agent to I/Os.

```java
//create a GPIO control trigger on the input pin (Input1)
Input1.addTrigger(new GpioSetStateTrigger(Input1.HIGH, pinOutput[0], PinState.LOW));

//function called if Input1 detect some event
public void call() throws Exception {
    Input1.addTrigger(new GpioCallbackTrigger(new Callable() {
        @Override
        public void call() throws Exception {
            Output[0].high();
            Output[1].low();
            return null;
        }
    });
}
```

After this step, the code was compiled in a JAR format and then deployed in the Raspberry Pi. Finally, the embedded controller composed by an industrial shield and the agent running in the Raspberry Pi platform, can control and manage the automation process, aided by the intelligent capabilities of the agent running in the low-cost industrial controller.

VI. CONCLUSIONS

In the Industry 4.0 advent, CPS are based on networks of components that integrate cyber and physical parts and are able to be connected and take their own decisions, becoming smarter devices, machines, systems and products. From device to device, the importance of enabling communication between control devices, even more in a decentralized way, is the natural direction of technology development from now on in the face of the enormous benefits of greater flexibility, reconfiguration and low-cost from this new reality being established through Industry 4.0. In this context, an important challenge is to improve the current state-of-the-art PLCs by taking into consideration the low-cost but powerful computational boards that will allow to embed IoT technologies and data analytics.

Having this in mind, this work presents a low-cost industrial controller, based on the single-board computer Raspberry Pi, that offers all characteristics imposed by industrial environments, such as robustness and reliability, and provides the capability to run applications developed in different languages, e.g., IEC 61131-3, IEC 61499 and Java. This industrial controller comprises industrial shields that, due to the galvanic insulation, provide an interface between the controller and the automation process, protecting the control board from the high currents that can circulate in the external environment.

The developed industrial controller was applied to control an automation system using three different industrial programming languages: 61131-3 (using OpenPLC), IEC 61499 (using 4DIAC IDE and FORTE RTE) and Java (using Eclipse IDE). The achieved results show the great potential to use these low-cost but powerful computational platforms in industrial automation, addressing the industrial requirements but being able to execute preprocessing algorithms and run, in a easy manner, application programs codified in different programming languages.

Future work mainly involves the execution of additional experimental tests, particularly analysing its endurance to harsh industrial environments, e.g., regarding to temperature, humidity and magnetic interference, and the development of other types of industrial shields that perform the control of different I/O ports, e.g., analog ports, as well as the implementation of commercially established industrial communication protocols, such as OPC UA, Ethernet/IP and Modbus.

REFERENCES

[1] A. A. Bryan and E. A. Bryan, *Programmable Controllers: Theory and Implementation*, Industrial Text Company, 2nd ed., 1997.
[2] IEC, “IEC 61131-3:201, International Electrotechnical Commission,” https://webstore.iec.ch/publication/4552.
[3] IEC, “IEC 61499-4:2013, International Electrotechnical Commission.” https://webstore.iec.ch/publication/5508.
[4] JADE, “JAVA Agent DEvelopment Framework is an Open Source Platform for Peer-to-Peer Agent Based Applications,” 2019. Last accessed 18 February 2019.
[5] H.-H. Erbe, “Manufacturing with Low Cost Automation,” *IFAC Proceedings Volumes*, vol. 36, no. 7, pp. 95 – 100, 2003.
[6] T. Alves and T. Morris, “OpenPLC: An IEC 61131–3 Compliant Open Source Industrial Controller for Cyber Security Research,” Computers & Security, vol. 78, pp. 364 – 379, 2018.
[7] U. technology, “Unipi technology.” https://www.unipi.technology.
[8] J. Yan and V. V. Vyatkin, “Distributed Execution and Cyber-physical Design of Baggage Handling Automation with IEC 61499,” in Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN’11), pp. 573–578, July 2011.
[9] M. V. García, F. Pérez, I. Calvo, and G. Morán, “Building Industrial CPS with the IEC 61499 Standard on Low-cost Hardware Platforms,” in Proceedings of the IEEE International Conference on Emerging Technology and Factory Automation (ETFA’14), pp. 1–4, Sep. 2014.
[10] E. Tisserant, L. Bessard, and M. de Sousa, “An Open Source IEC 61131-3 Integrated Development Environment,” in 2007 5th IEEE International Conference on Industrial Informatics, vol. 1, pp. 185–187, IEEE, 2007.
[11] M. de Sousa and A. Carvalho, “An IEC 61131-3 Compiler for the MatPLC,” in Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA’05), vol. 1, pp. 485–490 vol.1, Sep. 2003.
[12] IEC, “IEC 61499/2:2012, International Electrotechnical Commission,” https://webstore.iec.ch/publication/5507.
[13] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Chichester, England: NJ: John Wiley, 2007.