ISSN: 1692-7257 - Volume 2 – Issue 48 - 2026

ORIGINAL

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Development of an integrated industrial classification and control system with machine vision and PLC automation

Desarrollo de un sistema integrado de clasificación y control industrial con visión artificial y automatización PLC

Diego Alejandro Landinez Quintero[1]
Francisco Ernesto Moreno García[1]
Wlamyr Palacios Alvarado[2]

[1] **Universidad Francisco de Paula Santander, Facultad de Ingeniería, Programa de Ingeniería Electrónica, Cúcuta, Norte de Santander, Colombia.**

[2] **Universidad Francisco de Paula Santander, Facultad de Ingeniería, Programa de Ingeniería Industrial, Cúcuta, Norte de Santander, Colombia.**

Received: 24 March 2026. Revised: 13 May 2026. Accepted: 12 June 2026. Published: 06 July 2026.
Corresponding Author: Diego Alejandro Landinez Quintero

How to cite: D. A. Landinez Quintero, F. E. Moreno García, and W. Palacios Alvarado, "Desarrollo de un sistema integrado de clasificación y control industrial con visión artificial y automatización PLC", *Revista Colombiana de Tecnologías de Avanzada (RCTA)*, vol. 2, no. 48, jul. 2026, art. no. 4390, doi: 10.24054/rcta.v2i48.4390

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Abstract:

This article presents the design and implementation of a prototype for the identification, manipulation, and transport of objects, aimed at simulating an industrial process for classification and quality control. The system integrates a robotic arm, the YOLO detection model, ArUco markers, a PLC-controlled conveyor module, and 3D-printed figures, programmed primarily in Python. The methodology consisted of developing each system separately and then integrating them into a single graphical interface. The robotics and machine vision segments were developed entirely in Python; the PLC was programmed in TIA Portal and connected to Python using Snap7. System evaluation involved repeatability tests for manipulation and transport, and measurements were taken of the YOLO model's classification and segmentation accuracy, as well as the accuracy of the ArUco markers.

Keywords: robotics, artificial vision, PLC.

Resumen:

Este artículo presenta el diseño e implementación de un prototipo para la identificación, manipulación y transporte de objetos, orientado a simular un proceso industrial de clasificación y control de calidad. El sistema integra un brazo robótico, el modelo de detección YOLO, marcadores ArUco, un módulo de transporte controlado por PLC y figuras impresas en 3D, programado principalmente en Python. La metodología consistió en desarrollar cada sistema por separado, y posteriormente integrarlo en una sola interfaz gráfica, el segmento de robótica y visión artificial se desarrollaron completamente en Python; el PLC se programó en TIA Portal y se conectó a Python mediante Snap7. Para la evaluación del sistema se realizaron pruebas de repetibilidad en la manipulación y el transporte, y se midió la precisión de clasificación y segmentación del modelo YOLO, así como la exactitud de los marcadores ArUco.

Palabras clave: robótica, visión artificial, PLC.

INTRODUCTION

Industrial automation has evolved significantly thanks to advances in artificial intelligence, robotics, and programmable logic controllers (PLCs) [1]-[5]. These technologies have enabled academic labs and small businesses to access tools previously reserved for large industrial environments, reducing the gap between applied research and productive practice [6]-[12]. However, its implementation still faces barriers associated with initial costs, infrastructure requirements, and the complexity of integrating heterogeneous technologies into the same system [13]-[15].

In this context, one of the main challenges in academic and prototyping scenarios is to coherently articulate visual perception, robotic manipulation, and industrial control. Although there are isolated developments in each of these areas, the integration of a computer vision system, robotics, and a transport module controlled by PLC continues to represent a technical and methodological challenge, especially when working with teams from different brands, protocols, and programming environments. Therefore, it is relevant to explore architectures that allow coordinating these subsystems in a unified experimental environment, oriented towards functional validation and training in technologies associated with Industry 4.0 [16]-[21].

Against this backdrop, this article presents the design and implementation of a prototype that simulates an industrial classification and quality control process, combining a robotic arm made up of Dynamixel servomotors, a machine vision system based on YOLO and a Lucas-Nuelle conveyor module controlled by a Siemens S7-1200 PLC, integrating all the systems by means of Python. The objective is to verify the technical feasibility of integrating tools from different areas and brands into a single system that includes classification, manipulation and transport, creating a low-cost, open-source prototype that offers an educational tool for experimenting with detection parameters, manipulation strategies and device configuration.

As general background, an exploratory literature search was carried out mainly in Google Scholar, in Spanish and English, in order to identify works related to the proposed integration. In a first stage, the specific names of the equipment and tools used in the prototype were used as keywords, including OpenBot v1, Lucas-Nuelle Cyber physical conveyor system with PLC, YOLO and Python. Since no background was found that combined exactly these components, the search was subsequently broadened to more general terms, such as robotic arm, conveyor module, PLC, machine vision and Python. Finally, given that no works were found that jointly integrated the three areas into a single architecture either, relevant contributions in robotics, machine vision and PLC were reviewed separately. The selection of documents was based on their thematic relevance, their conceptual proximity to the proposal and their usefulness for supporting both the experimental development and the writing of the article.

In the field of robotics, the background [22]-[29], a pedagogical approach was found that combines simulation, open source, and accessible hardware to facilitate adoption and experimentation; graphical interfaces and simulation environments are used to familiarize users with the robot's kinematics, reducing the risk of harm to both the user and the equipment, and demonstrating the feasibility of controlling robots with free tools such as Python and specific vendor libraries, for example Dynamixel SDK, creating a system that can be controlled both locally and remotely without incurring costly licenses. The integration of embedded solutions, such as Raspberry Pi, ESP32-CAM, and simple sensors, for perception and control tasks, was also investigated. Overall, these works demonstrate that the combination of simulators, open-source languages, and economic platforms constitutes a valid solution for creating prototypes and preparing the student before using the robot in physics.

In computer vision, the YOLO family was selected for its wide adoption in real-time detection applications and for its balance between accuracy, inference speed, and computational complexity—features especially valuable in prototypes that must run on resource-limited hardware. Recent literature shows that YOLO has evolved as a single-stage architecture with good overall performance, and that its lightweight variants are suitable for embedded or low-cost scenarios where latency and efficient resource usage are determining factors. Furthermore, recent comparative studies indicate that, compared to two-stage architectures like Faster R-CNN, YOLO usually offers shorter inference time, while alternatives like SSDs present different trade-offs between speed and accuracy. In this work, in addition to these technical advantages, their ease of use and implementation in a conventional laptop was prioritized, which was consistent with the practical constraints of the prototype [30]-[38].

Regarding PLCs and industrial communications, the background [39]-[46], working with the Siemens S7-1200 PLC, show that it is feasible to integrate plant control with open-source solutions, implementations were found that communicate PLCs with computers via profinet, Modbus, or Snap7 for monitoring and control with errors below 1%, and intermediaries such as Raspberry Pi or Node-RED are proposed to create remote access and reduce costs in TIA Portal licenses. These experiences confirm the viability of the S7-1200 as a robust element for managing sensors and actuators, and highlight the technical feasibility of establishing a joint PLC and Python system for control, acquisition, and visualization in educational environments and industrial prototypes.

METHODOLOGY

For the development of this project, four phases were defined: three phases that focus on each area separately, and a final phase that integrates all the systems into a single graphical interface.

The first phase corresponds to the OpenBot v1 robotic arm from Robotics 4.0 (see Fig. 1). The technical manual of the Dynamixel AX-12 servomotors that make it up is studied in detail to extract critical parameters such as torque, speed, communication protocols and operating limits. The robot is controlled with Python using these servos' own library, dynamixel_sdk. First, a simple code is developed that reads the position of each servo without enabling torque, in order to establish position limits and avoid collisions with the structure. Subsequently, the execution of pre-established trajectories is implemented by means of JSON files, which are structured as a list of steps, where each one contains the servo ID, the speed, the position and the delay. Finally, it is developed to the point of becoming a complete graphical interface, designed mainly with Tkinter, focused on facilitating the design, editing and validation of sequences, allowing the trajectory to be adjusted due to physical changes in the arrangement of the elements.

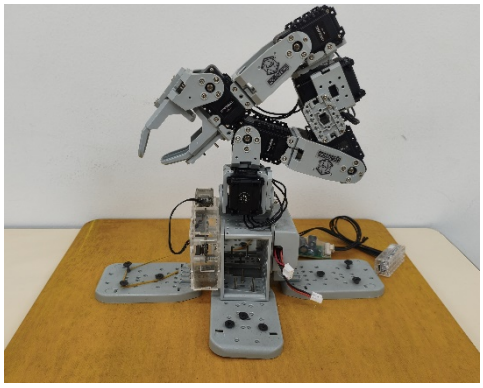

Fig. 1. *OpenBot v1 robotic arm.*
Source: own elaboration.

The robotic arm has 6 degrees of freedom, can be controlled from a computer via USB, and programmed in environments such as Matlab, LabView, C++, Java, Python, and ROS[47]. No further relevant information was found about the complete robot, so the servomotors were investigated, whose technical specifications include a resolution of 0.29°, a rotational range from 0 to 300°, a stopping torque of 1.5 N·m at 12 V and 1.5 A, a no-load speed of 59 rpm at 12 V, and an operating voltage between 9.0 and 12.0 V, with 11.1 V as the recommended value[48], It is important to remember that this information applies only to each servomotor individually. No specific load capacity was found in the consulted documentation, so during the tests only light objects were used, validating the operation through complete trajectories of intake, transport, and release.

The second phase corresponds to machine vision; simple geometric figures are designed and printed with the Snapmaker Original (see Fig. 2), varying not only the shape but also the color and size; these figures are chosen as the objective due to their ease of labeling. The training dataset consists of 810 images, of which 625 are allocated to training and 185 to validation, the dataset distribution is detailed in Table 1; additionally, 120 independent images taken after training are used to test the model.

For labeling, the labelme tool is used in segmentation mode; this tool exports to JSON, so the labelme2yolo library is used to convert the result to the TXT format used by YOLO. PyTorch is used to enable the GPU, along with YOLO's own library called Ultralytics. ArUco markers are integrated to obtain the pixel/cm ratio and thereby calculate the surface area of the figures.

Table 1: Dataset

Figure	**Square**					
Color	Red			Blue		
Length (cm)	2	3	4	2	3	4
# Images	100	49	28	100	49	28
Figure	Triangle					
Color	Red			Blue		
Length (cm)	2	3	4	2	3	4
# Images	130	62	36	130	62	36

Source: own elaboration

The dataset images and the system tests are carried out in a closed laboratory with artificial lighting. The images are taken with the Fischertechnik USB camera at a distance of 28 cm between the camera and the objects, and are captured at a resolution of 634x634 pixels, although YOLO automatically resizes them to 640x640 pixels during training and inference.



Fig. 2. *Snapmaker Original 3D printer.*
Source: own elaboration.

The third phase corresponds to the conveyor module (see Fig. 3). Because it is controlled with a Siemens S7-1200 PLC, it is configured and programmed with TIA Portal. First, the sensors and actuators are verified individually, including the limit switches, the switches and the motor. A simple program is developed that allows direct control from Python; the only logic in this part is a latch to enable the motor and restrictions on the limit switches to prevent the slide holder from being thrown off. The project is configured to allow external communication, and Snap7 is used from Python. Three main sequences are created: one to take the slide holder to the right edge, another similar one for the left side, and one that centers it. Since there is no central limit switch, this process is carried out by taking it to the left edge and then activating forward motion to the right for a predetermined time.

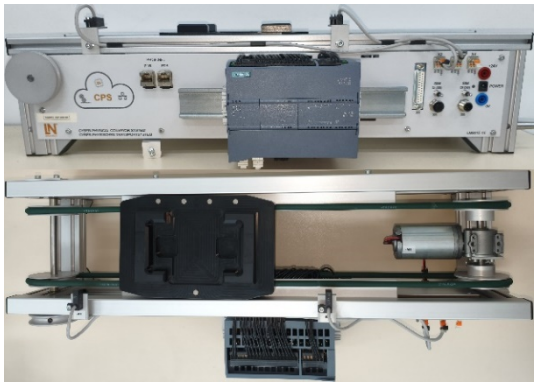

Fig. 3. *Lucas-Nuelle Cyber physical conveyor system with PLC.*
Source: own elaboration.

The fourth phase corresponds to the unified graphical interface. With the systems completed separately, they are integrated into a single application that includes the entire sequence necessary to simulate an industrial process. It is designed to be used by the end user, without the need to go deeply into the technical operation, while at the same time facilitating its modification and expansion since it is open source. The main feature of this graphical interface is the ability to select an objective, establishing its shape, color and size; in addition, it is possible to choose between two destinations, left and right. When a figure is recognized as valid according to how the user has configured it, the robot moves the figure from the base to the slide holder, the module transports it to the corresponding destination and, after a time, it returns to the center.

The information flow between the subsystems is organized in three stages. First, the camera continuously monitors the scene and, when it detects a valid objective that matches the selection made by the user in the interface, the second stage, corresponding to the robotic arm, is activated. In this phase, in addition to the start command, the class and size of the detected figure are transmitted; with this information the side length of the object is calculated and one of three predefined trajectories is selected, which modify the degree of closure of the gripper during grasping. Once the trajectory has been executed, the third stage is activated, corresponding to the PLC, to which the start command is sent together with the belt direction previously defined in the vision phase. With this signal, the PLC moves the slide holder to the corresponding destination and, after a predetermined time, centers it again, thus completing the operating cycle.

During development, Visual Studio Code 1.110 was used as the programming environment; Python 3.14.3 as the main language; dynamixel_sdk 4.0.3 for controlling the servomotors; tkinter 8.6.15 for the graphical interface; Pillow 12.1.1 for

image handling; Python's standard json module for saving and loading sequences; pyserial 3.5 for USB communication; labelme2yolo 0.3.2 for converting the annotation format; PyTorch 2.10.0 to enable the GPU for training; ultralytics 8.4.19 for training and inference of the YOLO model; opencv-python 4.13.0.92 for using the ArUco tags; numpy 2.4.2 for numerical operations; and python-snap7 2.1.0 for communication with the PLC. For labeling, labelme 5.11.3 with Python 3.9.13 was used, due to incompatibilities with more recent versions of Python. Finally, the PLC was programmed in TIA Portal 15.1.

To evaluate the performance of the prototype, metrics were defined for each phase: repeatability of the robotic arm, for which the sequence is executed and the success of picking up, moving and releasing the figure from the base to the conveyor module is measured; precision in YOLO classification, and accuracy of the area measurement with YOLO and ArUco; and the variation of the slide holder position when performing the centering sequence. These metrics make it possible to quantify the performance of the prototype and to propose future improvements for its scaling or educational application.

RESULTS

Robotic arm manipulation

The manual was reviewed and the control table of the AX-12 servomotor was identified; it allows access to up to 48 parameters, of which some are read-only (R) and most are read/write (RW). Of all these, only those necessary for the basic operation of the robotic arm were used; these are presented in Table 2.

Table 2: Control table

Name	Address	Access	Values
Enable torque	24	RW	0-1
Objective position	30	RW	0-1023
Movement speed	32	RW	0-1023 servo 0-2047 wheel

Source: own elaboration

With the parameters established, correct communication was verified using a simple code that does not enable torque and only prints the position of each servo. The dynamixel_sdk library provides all the functions needed to handle the servos; only the COM port to which the robot is connected has to be defined. Sequence execution was then implemented using JSON-type files organized as a list of steps, each step containing four key values: the servo ID, the speed, the target position and the delay; with this, the robot's ability to move objects was verified.

Lastly, a graphical interface (see Fig. 4) was developed for more intuitive use, controlling the position and speed of each servomotor by means of sliders. The main purpose of this application is to create, modify and validate sequences, so in addition to the basic controls, the state of each servo can be saved to create the sequence steps, execute that sequence, save the sequence to a JSON file and load a sequence from that file.



Fig. 4. *Graphical interface for the robotic arm.*
Source: own elaboration.

To evaluate this phase, a repeatability test was performed: the transport sequence of the figures was executed 120 times, representing 10 repetitions for each figure. The results are presented in Table 3. Overall, 114 successful executions out of 120 attempts were recorded, corresponding to a success rate of 95 %. A standard error of 1.99 % was calculated. These results show high repeatability and reduced variability of the transport system.

Table 3: Repeatability evaluation

Figure	**Square**						**Triangle**					
Color	Red			Blue			Red			Blue		
Length (cm)	2	3	4	2	3	4	2	3	4	2	3	4
Successes (N)	10	10	10	10	10	10	9	9	9	10	9	8

Source: own elaboration

The configurations with the lowest success rate were the triangles, which is attributed to their pointed geometry. Unlike squares, which have flat edges, triangles have a narrow vertical side with a smaller useful area. Consequently, when the piece is not perfectly centered with respect to the gripper, it slips and is pushed instead of being properly grasped.

Neural network for object recognition

It was decided to use 3D printing to create the objectives tailored to the equipment; the PLA used as material is also lightweight, an important aspect since the robot does not have much strength. The shape was designed as simple geometric figures with depth, a square and a triangle; they were printed in two colors, red and blue, and in three sizes, with side lengths of 2 cm, 3 cm and 4 cm. The result is shown in Fig. 5.

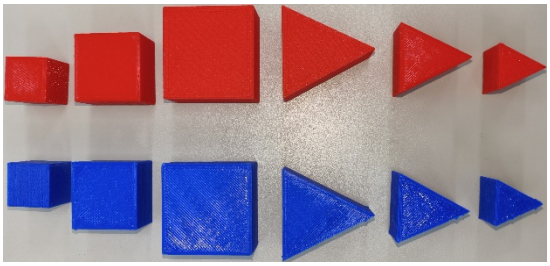

Fig. 5. *3D figures.*
Source: own elaboration.

Before training the model, pytorch was installed to enable the use of the GPU, since the CPU is used by default and performs worse in this type of task; to use YOLO, the Ultralytics library was installed. The exact YOLO model used was version 11, the most recent stable version at the time of carrying out this project, with size n, the smallest, at 2.9 million parameters, and in segmentation mode, since in addition to the classification the mask is required to determine the area. It was trained for 15 epochs with a batch size of 2; the training lasted 15.48 minutes.

Figure 6 shows that the precision, recall and mAP curves increase rapidly during the first epochs and then stabilize. Near the middle of the training, the improvements are less notable, which justifies the selection of 15 epochs as a balance point between minimal learning and a risk of overfitting. The normalized confusion matrix in Figure 7 also shows a marked concentration along the main diagonal, with low values in the relation with the background.

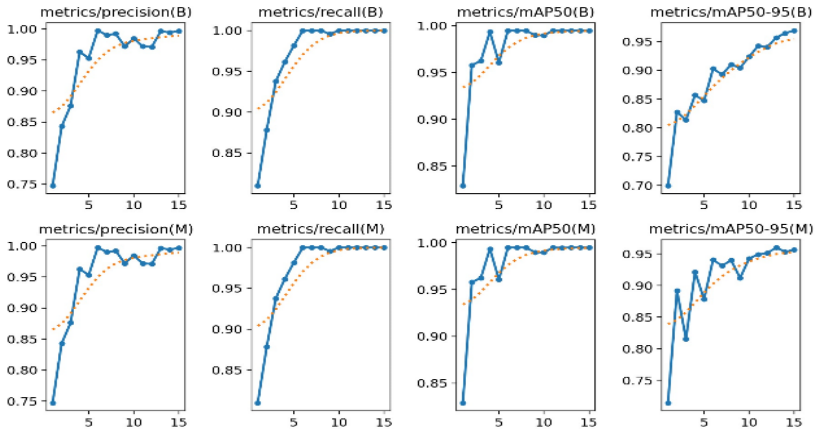

Fig. 6. *Training curves.*
Source: own elaboration.

In the final evaluation for box detection, the model reached precision(B)=0.99672, recall(B)=1, mAP50(B)=0.995 and mAP50-95(B)=0.96917; whereas for segmentation it obtained precision(M)=0.99672, recall(M)=1, mAP50(M)=0.995 and mAP50-95(M)=0.95665. These values reflect a high performance in the identification of the figures, both in localization and in segmentation. In Figure 7, the relation with the diagonal shows that CuadradoRojo has 0.94 in its class, while the other

figures reach 1; in the relation with the background, TrianguloAzul has 0 and the other classes 0.33.

Lastly, the ArUco markers were implemented; with these, the pixel/cm ratio is obtained and, together with the mask provided by YOLO, the surface area of the figures is determined. To evaluate this phase, a total of 120 measurements were made, 10 per figure; the test had two parts, one was the YOLO classification and evaluating whether it was correct or incorrect, the other was the area measurement and its error with respect to the real measurement; to avoid fluctuations, the area was recorded for five seconds and the average was taken. The classification had a precision of 100%, so only the area error will be detailed; the results are presented in Table 4.

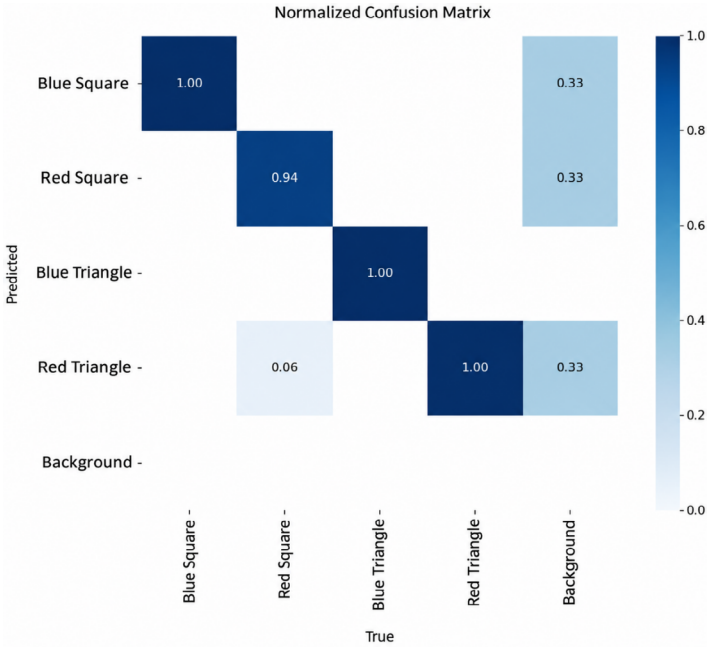

Fig. 7. *Normalized confusion matrix.*
Source: own elaboration.

Table 4: Area precision

Figure	**Square**					
Color	Red			Blue		
Length (cm)	2	3	4	2	3	4
Average area (cm^2)	3,96	8,91	15,62	3,98	8,86	15,28
Average error (%)	2,77	1,84	2,35	2,59	2,91	4,51
Maximum error (%)	6,37	2,92	3,95	4,80	6,05	8,12
Figure	Triangle					
Color	Red			Blue		
Length (cm)	2	3	4	2	3	4
Average area (cm^2)	1,92	4,28	7,68	1,91	4,31	7,62
Average error (%)	4,20	4,89	4,02	4,42	4,27	4,70
Maximum error (%)	10,33	7,55	6,59	8,91	6,99	7,29

Source: own elaboration

The sensors and actuators of the module were characterized: it has two limit switches, one for each end, a motor with two gears, one for each direction, and two switches; the PLC was set to IP 192.168.10.1, in rack 0 and slot 1, and the necessary options were configured to allow external communication.

Programming of the conveyor module

The TIA Portal project was programmed in Ladder (see Fig. 8); the program is divided into three sections: the first is the latch, which enables the use of the motor and uses the switches to start and stop the latch; the second is the actuators, where a virtual variable controls the motor, one variable for each gear, and if the limit switch is detected the motor stops automatically; the third is the sensors, where the state of each limit switch is copied to a virtual variable.

The main logic was then programmed in Python, using Snap7 to connect it to the PLC. In addition to the initial connection, three sequences were created: move right, which activates the gear in that direction until the corresponding limit switch is detected, at which point it stops the motor; move left, which performs the same function as the previous one but in the opposite direction, that is, to the left; center, which executes the move left sequence, and when it is completed activates the right gear for a predetermined time and then deactivates it; this was done in that way because the module does not have a

central limit switch.

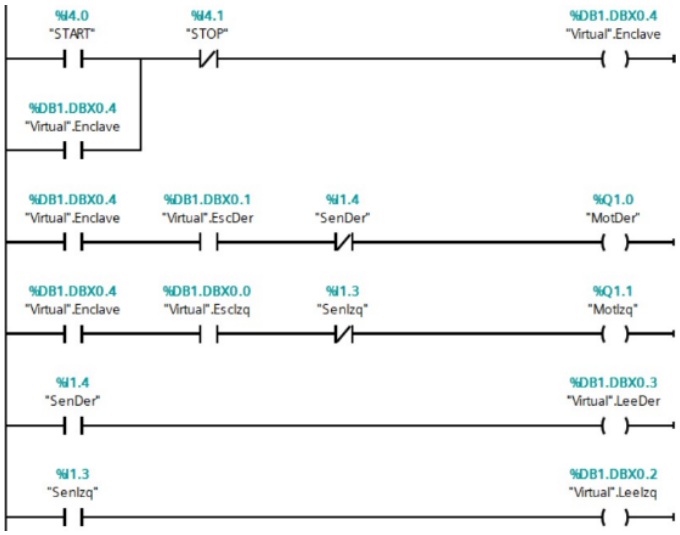

Fig. 8. *TIA Portal project.*
Source: own elaboration.

To evaluate this phase, the repeatability of the centering was measured; for this purpose, the sequence was executed 30 times, taking as a reference the position of one of the ends of the slide holder and measuring each time the distance between the reference and the new position of the slide holder, obtaining an average of 0.673 cm and a maximum of 1 cm. Although there is a small variation between executions, it remains within a reduced margin with respect to the total size of the slide holder, so it does not significantly affect the overall operation of the system nor compromise the operation of the prototype.

Development of the final application

Once each system had been tested separately, they were integrated into a single graphical interface that centralizes the configuration and operation of the prototype; it was designed to be simple to use, so that the user can become familiar with it quickly. The application (see Fig. 9) has two main configurations: the figure class, where one of the figures (blue square, red square, blue triangle or red triangle) can be selected for each direction (right and left); and the size, where two values are entered, the target area in square centimeters and the error margin in percentage. In addition to these configurations, it also allows changing the camera resolution, cropping the image, changing the type of ArUco and a button that opens the help window.

The program flow consists of three main sections: the start, where communication with the devices is established and the graphical interface is built; the centering, where the robot is sent to the predetermined position and the slide holder is centered; and finally the main logic, a loop where the user's choice is continuously retrieved and compared with the detected figure; if they match, the process begins, and once finished the cycle starts again.

The loop consists of waiting for the ArUco detection to calculate the pixel/cm ratio, then waiting for the detection of a single figure and checking that it remains over time to avoid false positives; once verified, the class and size are compared with the objectives selected by the user, and if they match, the side length is calculated from the target area to select the appropriate robot sequence and execute it; upon completing the maneuver the robot returns to its predetermined position. Finally, the belt activates the corresponding direction and, after a time, it centers and restarts the cycle. This algorithm is represented in Fig. 10.

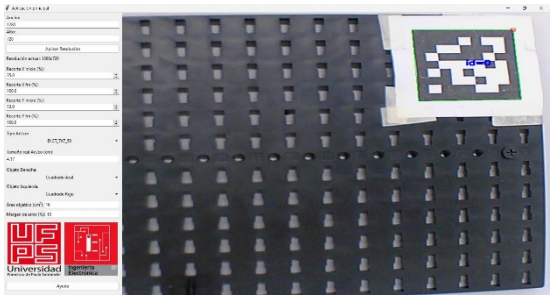

Fig. 9. *Final application.*
Source: own elaboration.

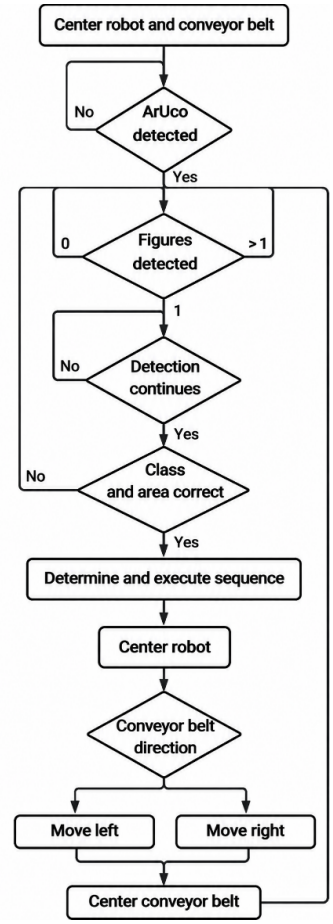

Fig. 10. *Flowchart of the system.*
Source: own elaboration.

CONCLUSIONS

The prototype had an approximate investment of 25,740,000 COP, considering the OpenBot v1 robotic arm, the Fischertechnik camera, the Snapmaker Original 3D printer, the PLA material and the conveyor module. This value is high for a basic academic setup, but it is still lower than a commercial industrial cell with similar capabilities. Moreover, its main contribution lies not only in its cost, but in the possibility of having a functional, flexible and reconfigurable environment for testing, training and validation of strategies.

The YOLO model presented a classification precision of 100 %; however, this result is only applicable to this specific case, under controlled lighting conditions. Even so, during real-time operation occasional failures were recorded; for example, when positioning the figures, the system even identified hands as if they were figures. Therefore, in a real environment it would be necessary to adapt a specific space so that the system has optimal performance. The area measurement errors with YOLO and ArUco were on average between 2 % and 4.5 %, with the maximum error throughout the test being 10.33 %; this precision is sufficient to correctly separate the different sizes.

The communication between the PLC and Python was stable and efficient, thus taking advantage of the robustness of the S7-1200 to handle the sensors and actuators and the flexibility of Python to create complex algorithms. The sequence of centering the slide holder, being performed by software, creates a variation in position, this being at most 1 cm, which I consider acceptable for the applications of this project.

The system confirmed the viability of the prototype by achieving a functional flow of detection, validation, manipulation, and transport. However, during its development and validation, limitations were identified related to sensitivity to lighting conditions, the detection of foreign objects in the field of view, variation in the gripping of pointed figures, and dependence on software-based centering in the conveyor module. These do not fully compromise the system, but they do reveal opportunities for improvement. As future work, we propose expanding the dataset with more diverse scenarios, optimizing the grip of the robotic arm, incorporating more robust measurements, and implementing a dedicated sensor for centering the conveyor module.

REFERENCES

[1] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business and Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, Aug. 2014, Accessed: Aug. 26, 2025. [Online]. Available: http://dx.doi.org/10.1007/s12599-014-0334-4

[2] J. Werheid et al., "Machine vision in manufacturing SMEs: a review," *Discover Applied Sciences*, vol. 7, no. 5, pp. 1–23, May 2025, Accessed: Aug. 26, 2025. [Online]. Available: https://doi.org/10.1007/s42452-025-06923-4

[3] N. Hütten, M. Alves Gomes, F. Hölken, K. Andricevic, R. Meyes, and T. Meisen, "Deep Learning for Automated Visual Inspection in Manufacturing and Maintenance: A Survey of Open- Access Papers," *Applied System Innovation*, vol. 7, no. 1, p. 11, Jan. 2024, Accessed: Aug. 26, 2025. [Online]. Available: https://doi.org/10.3390/asi7010011

[4] A. Bin Rashid and M. A. K. Kausik, "AI revolutionizing industries worldwide: A comprehensive overview of its diverse applications," *Hybrid Advances*, vol. 7, Dec. 2024, doi: 10.1016/J.HYBADV.2024.100277.

[5] J. T. Licardo, M. Domjan, and T. Orehovački, "Intelligent Robotics—A Systematic Review of Emerging Technologies and Trends," *Electronics (Basel).*, vol. 13, no. 3, Jan. 2024, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/electronics13030542

[6] R. Ben Ruben, C. Rajendran, R. Saravana Ram, F. Kouki, H. M. Alshahrani, and M. Assiri, "Analysis of barriers affecting Industry 4.0 implementation: An interpretive analysis using total interpretive structural modeling (TISM) and Fuzzy MICMAC," *Heliyon*, vol. 9, no. 12, p. e22506, Dec. 2023, Accessed: Aug. 26, 2025. [Online]. Available: https://doi.org/10.1016/j.heliyon.2023.e22506

[7] J. Ekong, V. Chauhan, J. Osedeme, S. A. Niknam, and R. Nguyen, "A framework for Industry 4.0 workforce training through project-based and experiential learning approaches," ASEE Annual Conference and Exposition, Conference Proceedings, Aug. 2022, Accessed: Aug. 26, 2025. [Online]. Available: https://peer.asee.org/a-framework-for-industry-4-0-workforce-training-through-project-based-and-experiential-learning-approaches.pdf

[8] B. Salah, S. Khan, M. Ramadan, and N. Gjeldum, "Integrating the Concept of Industry 4.0 by Teaching Methodology in Industrial Engineering Curriculum," *Processes*, vol. 8, no. 9, p. 1007, Aug. 2020, Accessed: Aug. 26, 2025. [Online]. Available: https://doi.org/10.3390/pr8091007

[9] A. Bondin and J. P. Zammit, "Education 4.0 for Industry 4.0: A Mixed Reality Framework for Workforce Readiness in Manufacturing," *Multimodal Technologies and Interaction*, vol. 9, no. 5, p. 43, May 2025, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/mti9050043

[10] I. Ahmad et al., "Inclusive learning using industry 4.0 technologies: addressing student diversity in modern education," *Cogent Education*, vol. 11, no. 1, Dec. 2024, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.1080/2331186X.2024.2330235

[11] M. Ryalat, N. Almtireen, G. Al-refai, H. Elmoaqet, and N. Rawashdeh, "Research and Education in Robotics: A Comprehensive Review, Trends, Challenges, and Future Directions," *Journal of Sensor and Actuator Networks*, vol. 14, no. 4, p. 76, Jul. 2025, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/jsan14040076

[12] M. Vukovic, O. Jorg, M. Hosseinifard, and G. Fantoni, "Low-Cost Digitalization Solution through Scalable IIoT Prototypes," *Applied Sciences*, vol. 12, no. 17, p. 8571, Aug. 2022, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/app12178571

[13] B. Ismaili, S. Bakkali, and S. Benriyene, "The Employability of Engineers in the Era of Industry 4.0," *Engineering Proceedings*, vol. 97, no. 1, p. 35, Jun. 2025, Accessed: Aug. 26, 2025. [Online]. Available: https://doi.org/10.3390/engproc2025097035

[14] D. N. Ege, J. R. Johannessen, S. E. Kildal, C. V. Amundsen, M. F. Berg, and M. Steinert, "Successfully Prototyping Industry 4.0 for Adoption in Smes: Guidelines and Case Study on a Low-Cost Vision-Based Measurement Systems for Slate Manufacturing," Jul. 2024, Accessed: Aug. 26, 2025. [Online]. Available: https://dx.doi.org/10.2139/ssrn.4884856

[15] M. E. Latino, "A maturity model for assessing the implementation of Industry 5.0 in manufacturing SMEs: learning from theory and practice," *Technol. Forecast. Soc. Change*, vol. 214, May 2025, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.1016/J.TECHFORE.2025.124045

[16] M. Mikhail and W. Sealy, "Configuration of a PLC Controlled Articulated Robot for Autonomous Vision Inspection Applications," LACCEI International Multi-Conference for Engineering, Education, and Technology, no. 1, Jun. 2022, Accessed: Jun. 18, 2026. [Online]. Available: http://dx.doi.org/10.18687/LACCEI2022.1.1.347

[17] M. Maślanka, D. Jancarczyk, and J. Rysinski, "Integration of Machine Vision and PLC-Based Control for Scalable Quality Inspection in Industry 4.0," *Sensors*, vol. 25, no. 20, Oct. 2025, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/s25206383

[18] Y. Li, P. Lai, T. Yang, and Z. Li, "Design and development of a machine-vision-based positioning compensation system for industrial robots," *Discover Applied Sciences*, vol. 8, no. 2, pp. 174-, Dec. 2025, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.1007/s42452-025-08196-3

[19] M. Ayaida, N. Messai, F. Valentin, and D. Marcheras, "TalkRoBots: A Middleware for Robotic Systems in Industry 4.0," *Future Internet*, vol. 14, no. 4, Mar. 2022, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/fi14040109

[20] B. Dafflon, N. Moalla, and Y. Ouzrout, "The challenges, approaches, and used techniques of CPS for manufacturing in Industry 4.0: a literature review," *The International Journal of Advanced Manufacturing Technology*, vol. 113, no. 7, pp. 2395–2412, Feb. 2021, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.1007/s00170-020-06572-4

[21] V. Fonseca, R. Barbosa, and F. Pereira, "Interoperability in Industrial Robotics: A Literature Review and Conceptual Path Toward a Universal Robot Protocol," *Applied Sciences*, vol. 16, no. 11, May 2026, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/app16115217

[22] J. S. Nieto Solano and S. A. Toro Mendoza, "Desarrollo de una interfaz gráfica interactiva para el robot OpenBotv V1 en el entorno de MATLAB," Universidad Distrital Francisco José de Caldas, 2022. Accessed: Jul. 09, 2025. [Online]. Available: http://hdl.handle.

net/11349/30190

[23] G. A. Yerbabuena Torres, "Diseño e implementación de una interfaz de comunicación para el control de posición de un brazo robótico de forma local y remota," Escuela Politécnica Nacional, 2024. Accessed: Jul. 10, 2025. [Online]. Available: http://bibdigital.epn.edu.ec/handle/15000/25344

[24] C. A. Valdiviezo Romero and R. A. Valladares Yanqui, "Implementación de un robot humanoide con reconocimiento de objetos por color y forma," Universidad Politécnica Salesiana, 2023. Accessed: Jul. 10, 2025. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/24123

[25] C. Chronis and I. Varlamis, "FOSSBot: An Open Source and Open Design Educational Robot," *Electronics (Basel).*, vol. 11, no. 16, Aug. 2022, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/electronics11162606

[26] J. Vega and V. Pérez, "G-ARM: An open-source and low-cost robotic arm integrated with ROS2 for educational purposes," *Multimed. Tools Appl.*, vol. 84, no. 33, Mar. 2025, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.1007/s11042-025-20748-8

[27] R. J. Trute, C. S. Zapico, A. Christou, D. Layeghi, S. Craig, and M. S. Erden, "Development of a Robotic Surgery Training System," *Front. Robot. AI*, vol. 8, Jan. 2022, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3389/frobt.2021.773830

[28] J. Vega and J. M. Cañas, "Open Vision System for Low-Cost Robotics Education," *Electronics (Basel).*, vol. 8, no. 11, Nov. 2019, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/electronics8111295

[29] J. Vega and J. M. Cañas, "PiBot: An Open Low-Cost Robotic Platform with Camera for STEM Education," *Electronics (Basel).*, vol. 7, no. 12, Dec. 2018, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/electronics7120430

[30] D. L. Moreira Ramos, "Aplicación de un modelo de reconocimiento de objetos utilizando Yolo (you only look once)," Universidad Estatal Península de Santa Elena, 2021. Accessed: Jul. 09, 2025. [Online]. Available: https://repositorio.upse.edu.ec/handle/46000/5755

[31] X. Jiménez Gómez, "Detección y análisis de datos sobre especies exóticas en biomas diferenciados en apoyo a la biodiversidad empleando la herramienta YOLO y microcontroladores," Universidad de La Coruña, 2024. Accessed: Jul. 09, 2025. [Online]. Available: http://hdl.handle.net/2183/39543

[32] A. Schcolnik-Elias, S. Martínez-Díaz, J. E. Luna-Taylor, and I. Castro-Liera, "Detección de armas tipo pistola mediante el uso de redes convolucionales con una arquitectura tipo YOLO y estereoscopía," *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, vol. 11, pp. 196–204, Sep. 2023, Accessed: Jul. 09, 2025. [Online]. Available: https://doi.org/10.29057/icbi.v11iEspecial2.10727

[33] M. Liaqat Ali, Z. Zhang, S. Tomassini, and M. Ali Akber Dewan, "The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection," *Computers*, vol. 13, no. 12, Dec. 2024, Accessed: Jun. 17, 2026. [Online]. Available: https://doi.org/10.3390/computers13120336

[34] P. Mittal, "A comprehensive survey of deep learning-based lightweight object detection models for edge devices," *Artif. Intell. Rev.*, vol. 57, no. 9, pp. 242-, Aug. 2024, Accessed: Jun. 17, 2026. [Online]. Available: https://doi.org/10.1007/s10462-024-10877-1

[35] W. Y. Feng, Y. F. Zhu, J. T. Zheng, and H. Wang, "Embedded YOLO: A Real-Time Object Detector for Small Intelligent Trajectory Cars," *Math. Probl. Eng.*, vol. 2021, no. 1, Jan. 2021, Accessed: Jun. 17, 2026. [Online]. Available: https://doi.org/10.1155/2021/6555513

[36] P. Vilcapoma et al., "Comparison of Faster R-CNN, YOLO, and SSD for Third Molar Angle Detection in Dental Panoramic X-rays," *Sensors*, vol. 24, no. 18, Sep. 2024, Accessed: Jun. 17, 2026. [Online]. Available: https://doi.org/10.3390/s24186053

[37] B. Karacaoglu and M. E. Sahin, "Optimized YOLO architectures for efficient Kiwi detection in precision agriculture on embedded systems," *Sci. Rep.*, vol. 15, no. 1, Dec. 2025, Accessed: Jun. 17, 2026. [Online]. Available: https://doi.org/10.1038/s41598-025-32770-9

[38] M. C. Luculescu, L. Cristea, A. Laszlo Boer, M. Gupta, F. I. Tiberiu Petrescu, and L. M. Ungureanu, "Artificial Vision System for Autonomous Mobile Platform Used in Intelligent and Flexible Indoor Environment Inspection," *Technologies (Basel).*, vol. 13, no. 4, Apr. 2025, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/technologies13040161

[39] F. D. Romero Herrera, "Diseño e implementación de un sistema de control y monitoreo de parámetros eléctricos dentro de procesos industriales mediante software libre y comunicación Modbus," Escuela Superior Politécnica de Chimborazo, 2022. Accessed: Jul. 07, 2025. [Online]. Available: https://dspace.espoch.edu.ec/handle/123456789/17704

[40] W. A. Fernández Lojano and D. F. Idrovo Vazconez, "Desarrollo de un sistema de monitoreo y control remoto para un variador de frecuencia Delta integrado a un PLC S7-1200 de Siemens por medio de una pasarela Inteligente basado en Raspberry Pi," Universidad Politécnica Salesiana del Ecuador, 2024. Accessed: Jul. 08, 2025. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/27554

[41] D. Macko, Ľ. Hrmo, J. Hrbček, and P. Nagy, "Real-Time Control System for Model Railway Based on SIMIS W Interlocking System," *Applied Sciences*, vol. 15, no. 1, Dec. 2024, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/app15010180

[42] D. Ionescu et al., "Communication and Control of an Assembly, Disassembly and Repair Flexible Manufacturing Technology on a Mechatronics Line Assisted by an Autonomous Robotic System," *Inventions*, vol. 7, no. 2, Jun. 2022, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/inventions7020043

[43] Q. T. Dao, L. T. Nguyen, T. K. Ha, V. H. Nguyen, and T. A. Nguyen, "Investigation of Secure Communication of Modbus TCP/IP Protocol: Siemens S7 PLC Series Case Study," *Applied System Innovation*, vol. 8, no. 3, Jun. 2025, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/asi8030065

[44] F. J. Folgado, D. Calderón, I. González, and A. J. Calderón, "Review of Industry 4.0 from the Perspective of Automation and Supervision Systems: Definitions, Architectures and Recent Trends," *Electronics (Basel).*, vol. 13, no. 4, Feb. 2024, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/electronics13040782

[45] M. Barton, R. Budjac, P. Tanuska, I. Sladek, and M. Nemeth, "Advancing Small and Medium-Sized Enterprise Manufacturing:

Framework for IoT-Based Data Collection in Industry 4.0 Concept," *Electronics (Basel).*, vol. 13, no. 13, Jun. 2024, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/electronics13132485

[46] G. Caiza and R. Sanz, "An Immersive Digital Twin Applied to a Manufacturing Execution System for the Monitoring and Control of Industry 4.0 Processes," *Applied Sciences*, vol. 14, no. 10, May 2024, Accessed: Jun. 18, 2026. [Online]. Available: https://doi.org/10.3390/app14104125

[47] Robotics 4.0, "E-Robotics 4.0." Accessed: Jun. 19, 2026. [Online]. Available: https://robotics40.com/e-robotics-4-0/

[48] DYNAMIXEL, "AX-12A." Accessed: Aug. 29, 2025. [Online]. Available: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

FUNDING

No financing

CONFLICT OF INTEREST

None

AUTHORSHIP CONTRIBUTIONS

Drafting – original draft: Diego Alejandro Landinez Quintero, Francisco Ernesto Moreno García, Wlamyr Palacios Alvarado.

Writing–review and editing: Diego Alejandro Landinez Quintero, Francisco Ernesto Moreno García, Wlamyr Palacios Alvarado.