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

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v2i48.4390

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], PhD. Francisco Ernesto Moreno García[1]
PhD. 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.*

Correspondence: *diegoalejandrolq@ufps.edu.co, femgarcia@ufps.edu.co*

Received: *march 24, 2026.* ***Accepted:*** *june 12, 2026.* ***Published:*** *july 06, 2026.*

How to cite: *D. A. Landinez Quintero, F. E. Moreno García, and W. Palacios Alvarado, "Development of an integrated industrial classification and control system with machine vision and PLC automation", RCTA, vol. 2, n.º. 48, pp. 45–56, jul. 2026. Recovered from https://ojs.unipamplona.edu.co/index.php/rcta/article/view/4390*

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.

1. INTRODUCTION

Industrial automation has significantly evolved due to advancements in artificial intelligence, robotics, and programmable logic controllers (PLCs) [1]-[5]. These technologies have allowed academic laboratories and small businesses to access tools previously reserved for large industrial environments, narrowing the gap between applied research and productive practice [6]-[12]. Nevertheless, their implementation still faces barriers associated with initial costs, infrastructure requirements, and the complexity of integrating heterogeneous technologies within a single system [13]-[15].

In this context, one of the primary challenges in academic and prototyping scenarios lies in the coherent articulation of visual perception, robotic manipulation, and industrial control. Although isolated developments exist in each of these fields, integrating a machine vision system, robotics, and a PLC-controlled transport module remains a technical and methodological challenge, particularly when aiming to work with equipment from different brands, protocols, and programming environments. Therefore, it is pertinent to explore architectures that enable the coordination of these subsystems within a unified experimental environment, oriented toward functional validation and training in technologies associated with Industry 4.0 [16]-[21].

In light of this scenario, this article presents the design and implementation of a prototype that simulates an industrial classification and quality control process, combining a robotic arm composed of Dynamixel servomotors, a YOLO-based machine vision system, and a Lucas-Nuelle transport module controlled by a Siemens S7-1200 PLC, integrating all systems via Python. The objective is to verify the technical feasibility of integrating tools from different fields and brands into a single system encompassing classification, manipulation, and transport. This creates a low-cost, open-source prototype that serves as an educational tool for experimenting with detection parameters, manipulation strategies, and device configurations.

As a general background, an exploratory literature review was conducted primarily on Google Scholar in both Spanish and English to identify studies related to the proposed integration. In the initial stage, specific names of the equipment and tools used in the prototype were selected as keywords, including OpenBot v1, Lucas-Nuelle Cyber physical conveyor system with PLC, YOLO, and Python. Since no literature combining these exact components was located, the search was subsequently expanded to more general terms such as robotic arm, transport module, PLC, machine vision, and Python. Finally, given that no studies were found integrating all three areas into a single architecture, relevant contributions in robotics, machine vision, and PLCs were reviewed separately. Document selection was based on thematic relevance, conceptual proximity to the proposal, and utility in supporting both the experimental development and the writing of the article.

In the field of robotics, the literature [22]-[29] highlighted a pedagogical approach combining simulation, open-source code, and accessible hardware to facilitate adoption and experimentation. Graphical interfaces and simulation environments are utilized to familiarize users with robot kinematics, reducing the risk of damage to both users and equipment, while demonstrating the feasibility of controlling robots with open-source tools like Python and manufacturer-specific libraries (e.g., Dynamixel SDK). This creates a system that can be controlled both locally and remotely without incurring expensive licensing fees. The integration of embedded solutions, such as Raspberry Pi, ESP32-CAM, and simple sensors for perception and control tasks, was also investigated. Taken together, these works demonstrate that combining simulators, open-source languages, and affordable platforms constitutes a valid solution for prototyping and preparing students before operating physical robots.

In machine vision, the YOLO family was selected due to its wide adoption in real-time detection applications and its balance of accuracy, inference speed, and computational complexity—features particularly valuable for prototypes running on resource-constrained hardware. Recent literature shows that YOLO has evolved into a single-stage architecture with good overall performance, and its lightweight variants are suitable for embedded or low-cost scenarios where latency and efficient resource utilization are critical factors. Furthermore, recent comparative studies indicate that compared to two-stage architectures like Faster R-CNN, YOLO

typically offers shorter inference times, while alternatives like SSD present different trade-offs between speed and accuracy. In this work, alongside these technical advantages, priority was given to its ease of use and implementation on a conventional laptop, which aligned with the practical constraints of the prototype [30]-[38].

Regarding PLCs and industrial communications, existing literature [39]-[46] working with the Siemens S7-1200 PLC shows that it is feasible to integrate plant control with open-source software solutions. Implementations communicating PLCs with computers via Profinet, Modbus, or Snap7 for monitoring and control achieved error rates under 1%, and intermediaries such as Raspberry Pi or Node-RED have been proposed to establish remote access and reduce TIA Portal licensing costs. These experiences confirm the reliability of the S7-1200 as a robust element for managing sensors and actuators, highlighting the technical feasibility of establishing a joint PLC-Python system for control, data acquisition, and visualization in educational settings and industrial prototypes.

2. METHODOLOGY

Four phases were defined for the development of this project: three phases focused on each individual area separately, and a final phase integrating all systems into a single graphical user interface (GUI).

The first phase corresponds to the OpenBot v1 robotic arm from Robotics 4.0 (see Fig. 1). The technical manual of the constituent Dynamixel AX-12 servomotors was studied in detail to extract critical parameters such as torque, speed, communication protocols, and operational limits. The robot is controlled in Python using the manufacturer's native servo library, dynamixel_sdk.

First, a basic script was developed to read each servo's position without enabling torque, aiming to set positional limits and prevent structural collisions. Subsequently, the execution of pre-established trajectories was implemented using JSON files, structured as a list of steps where each step contains the servo ID, speed, target position, and delay. Finally, this code was expanded into a complete graphical interface designed primarily with Tkinter, focused on facilitating sequence design, editing, and validation, thereby allowing trajectory adjustments due to physical alterations in the component layout.

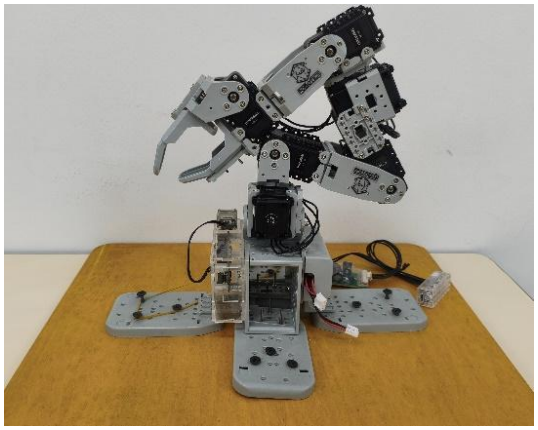

***Fig. 1**. OpenBot v1 robotic arm*
***Source:** compiled by the authors.*

The robotic arm has 6 degrees of freedom, can be controlled from a computer via USB, and can be programmed in environments such as Matlab, LabView, C++, Java, Python, and ROS [47]. No further relevant information was found regarding the complete robot assembly, so individual servomotor specifications were investigated. Their technical specifications include a resolution of 0.29°, a rotation range from 0 to 300°, a stall torque of 1.5 N·m at 12 V and 1.5 A, a no-load speed of 59 rev/min at 12 V, and an operating voltage between 9.0 and 12.0 V, with 11.1 V being the recommended value [48]. It is important to note that this information applies strictly to each individual servomotor. Since no specific payload capacity was identified in the reviewed documentation, testing was conducted exclusively with lightweight objects, validating operation through complete pick, place, and release trajectories.

The second phase corresponds to machine vision. Simple geometric shapes were designed and 3D-printed using a Snapmaker Original (see Fig. 2), varying in shape, color, and size; these shapes were chosen as targets due to their ease of labeling. The training dataset comprises 810 images, with 625 allocated for training and 185 for validation. The dataset distribution is detailed in Table 1; additionally, 120 independent images taken after training were used for model testing. Labeling was performed using the labelme tool in segmentation mode. Because this tool exports data in JSON format, the labelme2yolo library was used to convert the annotations into the TXT format required by YOLO. PyTorch was utilized to enable GPU acceleration, alongside YOLO's native library, Ultralytics. ArUco markers were integrated to obtain the pixel-to-centimeter ratio, enabling the calculation of the surface area of the shapes.

Table 1: *Dataset*

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

Source: *compiled by the authors.*

Dataset image acquisition and system testing were conducted in a enclosed laboratory under artificial lighting. Images were captured using a Fischertechnik USB camera at a distance of 28 cm from the objects, with a resolution of 634x634 pixels, although YOLO automatically resizes them to 640x640 pixels during the training and inference processes.

Fig. 2*. Snapmaker Original 3D Printer.*
Source: *compiled by the authors.*

The third phase corresponds to the transport module (see Fig. 3). Since it is controlled by a Siemens S7-1200 PLC, it was configured and programmed using TIA Portal. First, individual sensors and actuators were verified, including limit switches, toggle switches, and the motor. A straightforward program was developed to allow direct control from Python; the only logic implemented in this section was an interlock to enable motor operation and limit switch constraints to prevent the object carrier from derailing. The project was configured to allow external communication, utilizing Snap7 from Python. Three primary sequences were created: one to move the object carrier to the right edge, a similar one for the left side, and one to center it. Given the absence of a central limit switch, the centering process is performed by moving the carrier to the left edge and subsequently activating the rightward motion for a predetermined duration.

Fig. 3*. Lucas-Nuelle Cyber-physical conveyor system with PLC.*
Source: *compiled by the authors*

The fourth phase corresponds to the unified graphical user interface. Once the individual systems were completed separately, they were integrated into a single application that encompasses the entire sequence required to simulate an industrial process. It is designed for end-users, requiring no deep technical knowledge of the internal operations, while remaining easy to modify and expand due to its open-source nature. The primary feature of this GUI is the ability to select a target object by defining its shape, color, and size, alongside selecting between two destinations: left or right. Upon recognizing a valid shape according to the user's configuration, the robot transfers the shape from the base to the object carrier, the module transports it to the corresponding destination, and after a set time, centers itself again.

The data flow among the subsystems is organized into three stages. First, the camera continuously monitors the scene. When it detects a valid target that matches the user's selection on the interface, the second stage corresponding to the robotic arm is triggered. In this phase, besides the start command, the class and size of the detected shape are transmitted; this information is used to calculate the object's side length and select one of three predefined trajectories, which adjust the gripper's closure degree during grasping. Once the trajectory is executed, the third stage corresponding to the PLC is activated. The PLC receives the start command along with the conveyor direction previously defined during the vision phase. With this signal, the PLC shifts the object carrier to the designated destination and centers it again after a predetermined duration, completing the operational cycle.

During development, the following tools were utilized: Visual Studio Code 1.110 as the IDE; Python 3.14.3 as the main programming language; dynamixel_sdk 4.0.3 for servomotor control; 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 annotation formats; PyTorch 2.10.0 to enable GPU acceleration for training; ultralytics 8.4.19 for training and inference of the YOLO model; opencv-python 4.13.0.92 for processing ArUco tags; numpy 2.4.2 for numerical operations; and python-snap7 2.1.0 for communicating with the PLC. For labeling, labelme 5.11.3 was used with Python 3.9.13 due to incompatibilities with newer Python versions. Finally, the PLC was programmed in TIA Portal 15.1

To evaluate prototype performance, specific metrics were defined for each phase: robotic arm repeatability (measuring the success rate of picking, moving, and releasing the shape from the base to the transport module); YOLO classification accuracy and surface area measurement accuracy using YOLO and ArUco; and the position variance of the object carrier during the centering sequence. These metrics quantify prototype performance and suggest future improvements for educational or industrial scaling.

3. RESULTS

3.1. Robotic Arm Manipulation

The manufacturer manual was reviewed, identifying the AX-12 servomotor control table, which grants access to up to 48 parameters—some being read-only (R) and the majority being read/write (RW). Only the parameters necessary for the basic operation of the robotic arm were used, as shown in Table 2.

Table 2: *Control table*

Name	Address	Access	Values
Torque enable	24	RW	0-1
Goal position	30	RW	0-1023
Moving speed	32	RW	0-1023 servo 0-2047 wheel

Source: *compiled by the authors*

With the parameters established, correct communication was verified using a basic script that bypassed torque enablement and simply printed each servo's current position. The dynamixel_sdk library provides all the necessary functions to handle the servos, requiring only the definition of the COM port to which the robot is connected.

Subsequently, sequence execution was implemented using JSON-formatted files organized as a list of steps. Each step contains four key values: servo ID, speed, target position, and delay; this verified the robot's ability to manipulate objects. Finally, a graphical interface was developed for a more intuitive user experience, controlling each servomotor's position and speed via sliders (see Fig. 4). The primary purpose of this application is to create, modify, and validate sequences; thus, besides basic controls, it allows saving the status of each servo to generate sequence steps, executing said sequence, saving it to a JSON file, and loading a sequence from a file.

Fig. 4*. Graphical interface for the robotic arm.*
Source: *compiled by the authors.*

To evaluate this phase, a repeatability test was performed by executing the transport sequence 120 times, representing 10 repetitions for each specific shape. The results are presented in Table 3. Overall, 114 successful executions were recorded out of 120 attempts, corresponding to a success rate of 95%. A standard error of 1.99% was calculated. These results demonstrate high repeatability and reduced variability within 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: *compiled by the authors*

It was observed that the configurations with the lowest success rates corresponded to triangles, which is attributed to their sharp geometry. Unlike squares, which feature flat edges, triangles have a narrow vertical side with less usable surface area. Consequently, if the part is not perfectly centered

relative to the gripper, the gripper slips and pushes the object rather than securely clamping it.

3.2. Object Recognition Neural Network

It was decided to use 3D printing to fabricate customized targets tailored to the equipment's dimensions. Furthermore, the PLA material used is lightweight, which is a critical factor since the robot possesses limited lifting force. The targets were designed as simple geometric shapes with depth: a square and a triangle; they were printed in two colors (red and blue) and three lateral lengths (2 cm, 3 cm, and 4 cm). The results are shown in Fig. 5.

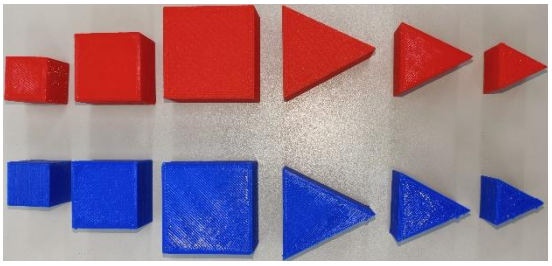

Fig. 5*. 3D-printed shapes.*
Source: *compiled by the authors***.**

Prior to training the model, PyTorch was configured to enable GPU utilization, as the system defaults to the CPU, which exhibits inferior performance in this class of tasks. To implement YOLO, the Ultralytics library was installed. The specific YOLO model deployed was version 11, the most recent stable release at the project's inception. The chosen model size was 'n' (nano)—the smallest variant, containing 2.9 million parameters—configured in segmentation mode, as the system requires the instance mask to calculate the surface area in addition to classification. Training was performed over 15 epochs with a batch size of 2, lasting 15.48 minutes.

Figure 6 illustrates that the precision, recall, and mAP curves increase rapidly during the initial epochs before stabilizing. Around the midpoint of the training process, performance gains become negligible, justifying the choice of 15 epochs as an optimal trade-off between baseline learning and overfitting risks. The normalized confusion matrix in Figure 7 further substantiates this performance, displaying a distinct concentration along the main diagonal and low background correlation values.

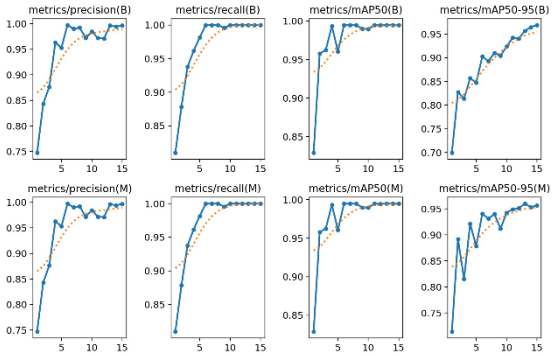

Fig*. 6. Training curves.*
Source: *compiled by the authors.*

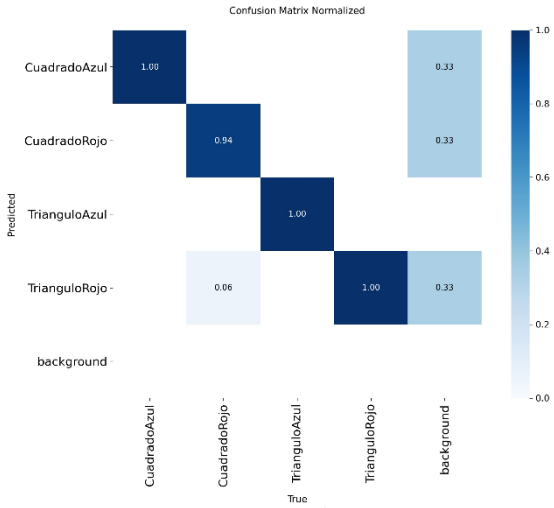

Fig. 7. *Normalized confusion matrix.*
Source: *compiled by the authors*

In the final evaluation for bounding-box object detection, the model achieved: $Precision(B) = 0.99672$, $Recall(B) = 1$, $mAP50(B) = 0.995$, and $mAP50-95(B) = 0.96917$. Meanwhile, for instance segmentation, it obtained: $Precision(M) = 0.99672$, $Recall(M) = 1$, $mAP50(M) = 0.995$, and $mAP50-95(M) = 0.95665$. These metrics reflect high performance in shape identification regarding both localization and segmentation. In Figure 7, the diagonal relation indicates that the 'Red Square' class scored 0.94, while all other shapes reached 1.0. Regarding background correlation, 'Blue Triangle' scored 0, while the remaining classes scored 0.33. Finally, ArUco markers were implemented to obtain the pixel-to-centimeter conversion ratio, which, combined with the YOLO segmentation mask, determines the surface area of the shapes.

To evaluate this phase, 120 total measurements were taken (10 per shape). The test evaluated two metrics: YOLO classification accuracy (correct vs. incorrect) and surface area measurement error relative to the

actual dimensions. To eliminate fluctuations, area data was recorded over five seconds and averaged. Classification achieved 100% accuracy; therefore, only the surface area calculation error is detailed in Table 4.

Table 4: *Area Measurement Accuracy*

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

Fuente: *elaboración propia*

3.3. Transport Module Programming

The module's sensors and actuators were fully characterized; it features two limit switches (one at each end), a motor with dual-direction control (forward and reverse), and two physical toggle switches. The PLC was configured with the static IP address 192.168.10.1, residing in Rack 0, Slot 1, with external communication options enabled.

The TIA Portal project was programmed using Ladder Logic (see Fig. 8). The program is divided into three sections: the first handles the motor interlocking logic to enable operation, using the physical switches for start and stop signals; the second manages the actuators, where virtual variables control the motor directions, ensuring that the motor stops automatically if a limit switch is triggered; the third handles the sensors, mapping the real-time status of each physical limit switch to a corresponding virtual variable. Subsequently, the core control sequence logic was programmed in Python

Subsequently, the main logic was 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 movement in that direction until it detects the corresponding limit switch, 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; and center, which executes the move left sequence, and when completed, activates the right movement for a predetermined time and then deactivates it. This was done in this way because the module does not have a center limit switch.

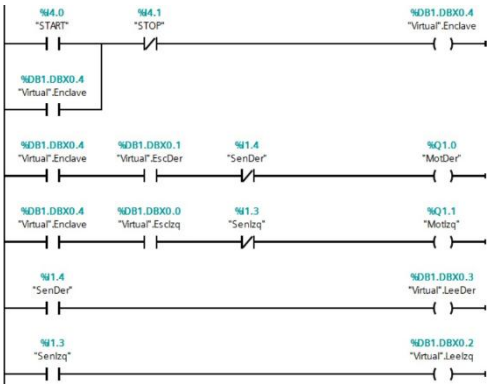

Fig. 8. *Ladder logic program diagram in TIA Portal.*
Source: *compiled by the authors.*

To evaluate this phase, the repeatability of the centering was measured. The sequence was executed 30 times, using the position of one end of the slide as a reference point. Each time, the distance between the reference point and the new position of the slide was measured. An average of 0.673 cm and a maximum of 1 cm were obtained. Although there is a small variation between executions, it remains within a small margin relative to the total size of the slide, and therefore does not significantly affect the overall functioning of the system or compromise the operation of the prototype.

3.4. Unified System Integration

Once each system was tested separately, they were integrated into a single graphical interface, which centralizes the configuration and operation of the prototype. It was designed for ease of use, allowing the user to quickly become familiar with it. The application (see Fig. 9) has two main settings: the shape class, where one of the shapes (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 margin of error as a percentage. In addition to these settings, it also allows changing the camera resolution, cropping the image, changing the ArUco type, and includes a button that displays the help window.

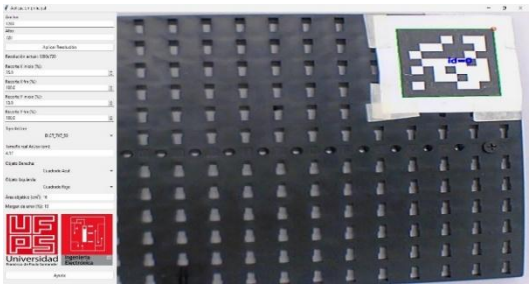

Fig. 9. *Unified system graphical user interface.*
Source: *compiled by the authors.*

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 object carrier is centered; and finally, the main logic, a loop where the user's selection is continuously extracted and compared with the detected shape. If they match, the process begins, and once finished, the cycle starts again. The loop consists of waiting for the ArUco to detect the shape and calculate the pixel/cm ratio. Then, it waits for the detection of a single shape and checks that it remains constant over time to avoid false positives. Once verified, the shape's class and size are compared with the targets selected by the user. If they match, the lateral length is calculated from the target area to select the appropriate robot sequence and execute it. Upon completion of the maneuver, the robot returns to its predetermined position. Finally, the conveyor belt activates the corresponding direction, and after a short time, the robot centers itself and the cycle restarts. This algorithm is represented in Figure 10.

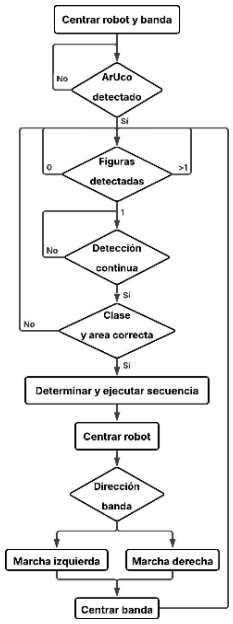

Fig. 10*. System flow diagram* ***Source:*** *compiled by the authors.*

4. CONCLUSIONS

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

The YOLO model demonstrated 100% classification accuracy; however, this result is only applicable to this specific case under controlled lighting conditions. Even so, occasional errors were recorded during real-time operation; for example, when positioning the figures, the system sometimes identified hands as figures. Therefore, in a real-world environment, a specific space would need to be adapted for the system to perform optimally. The area measurement errors with YOLO and ArUco averaged between 2% and 4.5%, with a maximum error of 10.33% throughout the test; this accuracy is sufficient for correctly separating the different sizes.

Communication between the PLC and Python was stable and efficient, leveraging the robustness of the S7-1200 for handling the sensors and actuators and the flexibility of Python for creating complex algorithms. The software-based centering sequence of the object holder creates a slight variation in position, with a maximum variation of 1 cm, which I consider acceptable for the applications of this project.

The system confirmed the prototype's viability 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, detection of foreign objects in the field of view, variations in gripping pointed objects, and the dependence on software-based centering in the transport module. These issues completely compromise the system, but they do highlight opportunities for improvement. Future work includes expanding the dataset with more diverse scenarios, optimizing the robotic arm's grip, incorporating more robust measurements, and implementing a dedicated sensor for centering the transport 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/