ISSN: 1692-7257 - Volumen 2 – Número 48 - 2026

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v2i48.4390

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

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

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.*

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

Recibido: *24 marzo 2026.* ***Aceptado:*** *12 junio 2026.* ***Publicado:*** *06 julio 2026.*

Cómo citar: *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", RCTA, vol. 2, n.º. 48, pp. 45–56, jul. 2026. Recuperado de https://ojs.unipamplona.edu.co/index.php/rcta/article/view/4390*

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.

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.

1. INTRODUCCIÓN

La automatización industrial ha evolucionado significativamente gracias al avance de la inteligencia artificial, la robótica y los controladores lógicos programables (PLC) [1]-[5]. Estas tecnologías han permitido que laboratorios académicos y pequeñas empresas accedan a herramientas antes reservadas para grandes entornos industriales, reduciendo la brecha entre la investigación aplicada y la práctica productiva [6]-[12]. No obstante, su implementación sigue enfrentando barreras asociadas con los costos iniciales, los requerimientos de infraestructura y la complejidad de integrar tecnologías heterogéneas en un mismo sistema [13]-[15].

En este contexto, uno de los principales desafíos en escenarios académicos y de prototipado consiste en articular, de forma coherente, percepción visual, manipulación robótica y control industrial. Aunque existen desarrollos aislados en cada una de estas áreas, la integración de un sistema de visión artificial, robótica y un módulo de transporte controlado por PLC sigue representando un reto técnico y metodológico, especialmente cuando se busca trabajar con equipos de distintas marcas, protocolos y entornos de programación. Por ello, resulta pertinente explorar arquitecturas que permitan coordinar estos subsistemas en un entorno experimental unificado, orientado a la validación funcional y a la formación en tecnologías asociadas a Industria 4.0 [16]-[21].

Frente a este panorama, el presente artículo presenta el diseño e implementación de un prototipo que simula un proceso industrial de clasificación y control de calidad, combinando un brazo robótico conformado por servomotores Dynamixel, un sistema de visión artificial basado en YOLO y un módulo de transporte de Lucas-Nuelle controlado por un PLC Siemens S7-1200, integrando todos los sistemas mediante Python. El objetivo es verificar la viabilidad técnica de integrar herramientas de distintas areas y marcas en un solo sistema que incluya clasificación, manipulación y transporte, creando un prototipo de bajo coste y de código abierto, ofreciendo una herramienta educativa que permita experimentar con parámetros de detección, estrategias de manipulación y configuración de dispositivos.

Como antecedente general, se realizó una búsqueda bibliográfica exploratoria principalmente en Google Scholar, en idioma español e inglés, con el fin de identificar trabajos relacionados con la integración propuesta. En una primera etapa se utilizaron como palabras clave los nombres específicos de los equipos y herramientas empleados en el prototipo, incluyendo OpenBot v1, Lucas-Nuelle Cyber physical conveyor system with PLC, YOLO y Python. Debido a que no se localizaron antecedentes que combinaran exactamente estos componentes, la búsqueda se amplió posteriormente a términos más generales, como brazo robótico, módulo de transporte, PLC, visión artificial y Python. Finalmente, dado que tampoco se encontraron trabajos que integraran de forma conjunta las tres áreas en una sola arquitectura, se revisaron por separado aportaciones relevantes en robótica, visión artificial y PLC. La selección de documentos se realizó con base en su pertinencia temática, su cercanía conceptual con la propuesta y su utilidad para sustentar tanto el desarrollo experimental como la redacción del artículo.

En el ámbito de la robótica, los antecedentes [22]-[29], se encontró un enfoque pedagógico que combina simulación, código abierto y hardware accesible para facilitar la adopción y experimentación, se emplean interfaces gráficas y entornos de simulación para familiarizar a los usuarios con la cinemática del robot, reduciendo el riesgo de daños tanto al usuario como al equipo, y demostrando la viabilidad de controlar robots con herramientas libres como Python y las librerías específicas de fabricantes, por ejemplo Dynamixel SDK, creando un sistema que puede ser controlado tanto local como remotamente sin incurrir en costosas licencias. También se investigó la integración de soluciones embebidas, como Raspberry Pi, ESP32-CAM y sensores simples, para tareas de percepción y control. En conjunto, estos trabajos demuestran que la combinación de simuladores, lenguajes de código abierto y plataformas económicas constituye una solución válida para crear prototipos y preparar al estudiante antes de utilizar el robot en físico.

En visión artificial, la familia YOLO se seleccionó por su amplia adopción en aplicaciones de detección en tiempo real y por su equilibrio entre precisión, velocidad de inferencia y complejidad computacional, características especialmente valiosas en prototipos que deben ejecutarse sobre

hardware de recursos limitados. La literatura reciente muestra que YOLO ha evolucionado como una arquitectura de una sola etapa con buen desempeño global, y que sus variantes ligeras son adecuadas para escenarios embebidos o de bajo costo, donde la latencia y el uso eficiente de recursos son factores determinantes. Además, estudios comparativos recientes indican que, frente a arquitecturas de dos etapas como Faster R-CNN, YOLO suele ofrecer menor tiempo de inferencia, mientras que alternativas como SSD presentan compromisos distintos entre rapidez y exactitud. En este trabajo, además de estas ventajas técnicas, se priorizó su facilidad de uso e implementación en un portátil convencional, lo cual resultó coherente con las restricciones prácticas del prototipo [30]-[38].

Respecto a PLC y a comunicaciones industriales, los antecedentes [39]-[46], que trabajan con el PLC Siemens S7-1200, muestran que es factible integrar control de planta con soluciones de software libre, se encontraron implementaciones que comunican PLC con computadores mediante profinet, Modbus o Snap7 para monitorización y control con errores bajo el 1 %, y se proponen intermediarios como Raspberry Pi o Node-RED para crear un acceso remoto y reducir costes en las licencias de TIA Portal. Estas experiencias confirman la viabilidad del S7-1200 como un elemento robusto para gestionar sensores y actuadores, y destacan la viabilidad técnica de establecer un sistema conjunto de PLC y Python para control, adquisición y visualización en entornos educativos y prototipos industriales.

2. METODOLOGÍA

Para el desarrollo de este proyecto se definieron cuatro fases, tres fases que se enfocan en cada área por separado, y una última fase que integra todos los sistemas en una única interfaz gráfica.

La primera fase corresponde al brazo robótico OpenBot v1 de Robotics 4.0 (ver Fig. 1), se estudia en detalle el manual técnico de los servomotores Dynamixel AX-12 que lo conforman, para extraer parámetros críticos como el torque, la velocidad, protocolos de comunicación y límites operativos; el robot se controla con Python con la librería propia de estos servos, dynamixel_sdk. Primero se desarrolla un código simple que lee la posición de cada servo sin habilitar el torque, esto para establecer límites en la posición y evitar colisiones con la estructura, posteriormente se implementa la ejecución de trayectorias preestablecidas mediante archivos JSON, los cuales se estructuran en una lista de pasos, donde cada uno contiene el ID del servo, la velocidad, la posición y el delay. Finalmente se desarrolla hasta convertirla en una interfaz gráfica completa, diseñada principalmente con Tkinter, enfocada en facilitar el diseño, la edición y la validación de secuencias, permitiendo ajustar la trayectoria debido a cambios físicos en la disposición de los elementos.

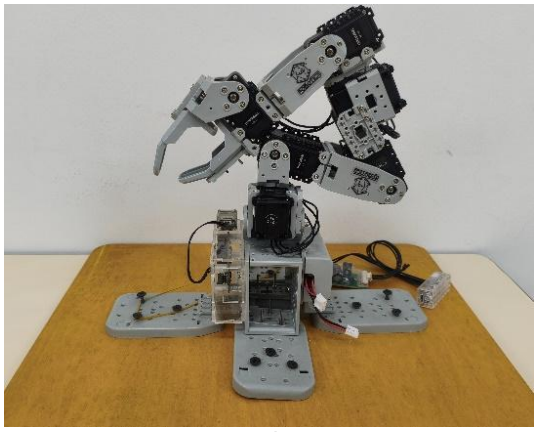

***Fig. 1**. Brazo robótico OpenBot v1.*
***Fuente:** elaboración propia.*

El brazo robótico tiene 6 grados de libertad, puede controlarse desde un computador mediante USB, y programarse en entornos como Matlab, LabView, C++, Java, Python y ROS [47]. No se encontró más información relevante sobre el robot completo, por lo que se indagó en los servomotores, cuyas especificaciones técnicas incluyen una resolución de 0,29°, un rango de giro de 0 a 300°, un torque de parada de 1,5 N·m a 12 V y 1,5 A, una velocidad sin carga de 59 rev/min a 12 V y un voltaje de operación entre 9,0 y 12,0 V, con 11,1 V como valor recomendado [48], es importante recordar que esta información se aplica únicamente a cada servomotor de manera individual. En la documentación consultada no se encontró una capacidad de carga específica, por lo que durante las pruebas se trabajó únicamente con objetos ligeros, validando la operación mediante trayectorias completas de toma, traslado y liberación.

La segunda fase corresponde a la visión artificial, se diseñan figuras geométricas simples y se imprimen con la Snapmaker Original (ver Fig. 2), variando además de la forma el color y el tamaño; se eligen estas figuras como objetivo debido a su facilidad para etiquetarlas. El dataset para el entrenamiento se compone de 810 imágenes, de las cuales 625 se destinan al entrenamiento y 185 a la validación, la distribución del dataset se detalla en la Tabla 1; adicionalmente, para la prueba del modelo se

utilizan 120 imágenes independientes tomadas posteriormente al entrenamiento.

Para etiquetar se utiliza la herramienta labelme en modo segmentación, esta herramienta exporta en JSON así que se utiliza la librería labelme2yolo para convertir el resultado al formato TXT usado por YOLO. Se utiliza PyTorch para habilitar la GPU, y la librería propia de YOLO llamada Ultralytics. Se integran marcadores ArUco para obtener la proporción pixel/cm y con esto calcular el área superficial de las figuras.

Tabla 1: *Dataset*

Figura	**Cuadrado**					
Color	**Rojo**			**Azul**		
Longitud (cm)	2	3	4	2	3	4
# Imágenes	100	49	28	100	49	28
Figura	**Triangulo**					
Color	**Rojo**			**Azul**		
Longitud (cm)	2	3	4	2	3	4
# Imágenes	130	62	36	130	62	36

Fuente: *elaboración propia*

Las imágenes del dataset y las pruebas del sistema se realizan en un laboratorio cerrado con iluminación artificial. Para tomar las imágenes se usa la cámara USB Fischertechnik, la distancia entre la cámara y los objetos es de 28 cm, y las imágenes se capturan con una resolución de 634x634 píxeles, aunque YOLO las redimensiona automáticamente a 640x640 píxeles durante el proceso de entrenamiento e inferencia.

Fig. 2*. Impresora 3D Snapmaker Original.*
Fuente: *elaboración propia.*

La tercera fase corresponde al módulo de transporte (ver Fig. 3), ya que se controla con un PLC Siemens S7-1200 se configura y programa con TIA Portal, primero se verifica individualmente los sensores y actuadores, lo que incluye finales de carrera, interruptores y el motor. Se desarrolla un programa sencillo que permite el control directo desde Python, la única lógica de esta parte es un enclave para habilitar el uso del motor y restricciones de los finales de carrera para evitar tirar el portaobjetos. Se configura el proyecto para permitir la comunicación externa, y desde Python se usa Snap7; se crean tres secuencias principales, una para lleva el portaobjetos al borde derecho, otra similar para el lado izquierdo, y una que lo centra. Ya que no se cuenta con un final de carrera central, este proceso se realiza llevándolo al borde izquierdo y posteriormente activando la marcha derecha un tiempo predeterminado.

Fig. 3*. Lucas-Nuelle Cyber physical conveyor system with PLC.*
Fuente: *elaboración propia.*

La cuarta fase corresponde a la interfaz gráfica unificada, con los sistemas completados por separado se integran en una única aplicación, esta incluye toda la secuencia necesaria para simular un proceso industrial. Se diseña para ser usada por el usuario final, sin necesidad de profundizar en el funcionamiento técnico, pero a la vez facilitando su modificación y ampliación al ser de código abierto. La principal característica de esta interfaz gráfica es poder seleccionar un objetivo, estableciendo su forma, color y tamaño, además se puede elegir entre dos destinos, izquierda y derecha. Al reconocer una figura como valida según halla configurado el usuario, el robot mueve la figura de la base al portaobjetos, el módulo lo transporta al destino correspondiente y después de un tiempo vuelve a centrarse.

El flujo de información entre los subsistemas se organiza en tres etapas. En primer lugar, la cámara permanece monitoreando continuamente la escena y, cuando detecta un objetivo válido que coincide con la selección realizada por el usuario en la interfaz, se activa la segunda etapa correspondiente al brazo robótico. En esta fase, además de la orden de inicio, se transmite la clase y el tamaño de la figura detectada; con esta información se calcula la

longitud lateral del objeto y se selecciona una de tres trayectorias predefinidas, las cuales modifican el grado de cierre de la pinza durante la toma. Una vez ejecutada la trayectoria, se activa la tercera etapa, correspondiente al PLC, al cual se le envía la orden de inicio junto con la dirección de la banda previamente definida en la fase de visión. Con esta señal, el PLC desplaza el portaobjetos hacia el destino correspondiente y, después de un tiempo predeterminado, lo centra nuevamente, completando así el ciclo de operación.

Durante el desarrollo se utilizaron Visual Studio Code 1.110 como entorno de programación; Python 3.14.3 como lenguaje principal; dynamixel_sdk 4.0.3 para el control de los servomotores; tkinter 8.6.15 para la interfaz gráfica; Pillow 12.1.1 para el manejo de imágenes; el módulo estándar json de Python para guardar y cargar secuencias; pyserial 3.5 para la comunicación por USB; labelme2yolo 0.3.2 para convertir el formato de las anotaciones; PyTorch 2.10.0 para habilitar la GPU para el entrenamiento; ultralytics 8.4.19 para el entrenamiento e inferencia del modelo YOLO; opencv-python 4.13.0.92 para usar las etiquetas ArUco; numpy 2.4.2 para operaciones numéricas; y python-snap7 2.1.0 para la comunicación con el PLC. Para el etiquetado se utilizó labelme 5.11.3 con Python 3.9.13, debido a incompatibilidades con versiones más recientes de Python. Finalmente, el PLC se programó en TIA Portal 15.1.

Para evaluar el rendimiento del prototipo se definieron métricas para cada fase: repetibilidad del brazo, se ejecuta la secuencia y se mide el éxito de recoger, mover y soltar la figura desde la base hasta el módulo de transporte; precisión en la clasificación de YOLO, y exactitud de la medición del área con YOLO y ArUco; y la variación de la posición del portaobjetos al realizar la secuencia de centrado. Estas métricas permiten cuantificar el desempeño del prototipo y plantear futuras mejoras para su escalado o aplicación educativa.

3. RESULTADOS

3.1. Manipulación del brazo robótico

Se reviso el manual y se identificó la tabla de control del servomotor AX-12, permite el acceso de hasta 48 parámetros, de los cuales algunos son de lectura (R) y la mayoría son de lectura/escritura (RW). De todos estos solo se utilizaron los necesarios para el funcionamiento básico del brazo robótico, estos se presentan en la Tabla 2.

Tabla 2: *Tabla de control*

Nombre	Dirección	Acceso	Valores
Habilitar torque	24	RW	0-1
Posición objetivo	30	RW	0-1023
Velocidad de movimiento	32	RW	0-1023 servo 0-2047 rueda

Fuente: *elaboración propia*

Con los parámetros establecidos se comprobó la correcta comunicación con un código simple que no habilita el torque y solo imprime la posición de cada servo. La librería dynamixel_sdk ofrece todas las funciones necesarias para manejar los servos, solamente se tiene que definir el puerto COM al que está conectado el robot. Posteriormente se implementó la ejecución de secuencias, para ello se usaron archivos tipo JSON y se organizó como una lista de pasos, cada paso contiene cuatro valores clave, el ID del servo, la velocidad, la posición objetivo y el delay; con esto se comprobó la capacidad del robot para mover objetos.

Por último, se desarrolló una interfaz gráfica (ver Fig. 4) para un uso más intuitivo, controlando la posición y la velocidad de cada servomotor por medio de sliders. El principal propósito de esta aplicación es crear, modificar y validar secuencias, por lo que además de los controles básicos se pude guardar el estado de cada servo para crear los pasos de la secuencia, ejecutar dicha secuencia, guardar la secuencia en un archivo JSON y cargar una secuencia desde ese archivo.

Fig. 4*. Interfaz gráfica para el brazo robótico.*
Fuente: *elaboración propia.*

Para evaluar esta fase se realizó una prueba de repetibilidad, se ejecutó la secuencia de transporte de las figuras 120 veces, lo que representa 10 repeticiones para cada figura. Los resultados se presentan en la Tabla 3. En conjunto, se registraron 114 ejecuciones exitosas de 120 intentos, lo que corresponde a una tasa de éxito del 95 %. Se calculo un error estándar de 1.99 %. Estos resultados evidencian una alta repetibilidad y una variabilidad reducida del sistema de transporte.

Tabla 3: *Evaluación de repetibilidad*

Figura	Cuadrado						Triangulo					
Color	Rojo			Azul			Rojo			Azul		
Longitud (cm)	2	3	4	2	3	4	2	3	4	2	3	4
Éxitos (N)	10	10	10	10	10	10	9	9	9	10	9	8

Fuente: *elaboración propia*

Se observó que las configuraciones con menor tasa de acierto corresponden a los triángulos, lo cual se atribuye a su geometría puntiaguda. A diferencia de los cuadrados, que presentan bordes planos, los triángulos poseen un lado vertical estrecho y con menor área útil. En consecuencia, cuando la pieza no queda perfectamente centrada respecto a la pinza, esta resbala y es empujada en lugar de ser sujetada correctamente.

3.2. Red neuronal para reconocimiento de objetos

Se decidió usar impresión 3D para crear los objetivos a la medida de los equipos, además de que el PLA utilizado como material es ligero, aspecto importante ya que el robot no posee mucha fuerza. Se diseño la forma como figuras geométricas simples con profundidad, un cuadrado y un triángulo; se imprimieron en dos colores, rojo y azul; y en tres tamaños, de longitud lateral 2 cm, 3 cm y 4 cm. El resultado se ve en la Fig. 5.

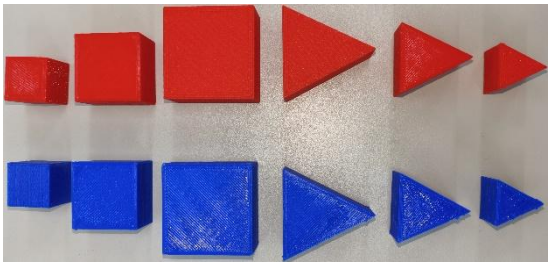

Fig. 5*. Figuras 3D.*
Fuente: *elaboración propia.*

Antes de entrenar el modelo, se instaló pytorch para habilitar el uso de la GPU, ya que por defecto se usa la CPU, la cual rinde peor en esta clase de tareas; para usar YOLO se instaló la librería Ultralytics. El modelo exacto de YOLO utilizado fue: la versión 11, la versión estable más reciente al momento de realizar este proyecto; el tamaño n, el cual es el más pequeño, con 2,9 millones de parámetros; y el modo segmentación, ya que además de la clasificación se requiere la máscara para determinar el área. Se entrenó con 15 épocas y un batch size de 2, el entrenamiento duro 15,48 minutos.

En la Figura 6 se observa que las curvas de precisión, recall y mAP aumentan rápidamente durante las primeras épocas y luego se estabilizan. Cerca de la mitad del entrenamiento, las mejoras son menos notables, lo cual justifica la selección de 15 épocas como un punto de equilibrio entre un aprendizaje mínimo y un riesgo de sobreajuste. La matriz de confusión normalizada de la Figura 7 también respalda muestra una concentración marcada sobre la diagonal principal, y con valores bajos en la relación con el fondo.

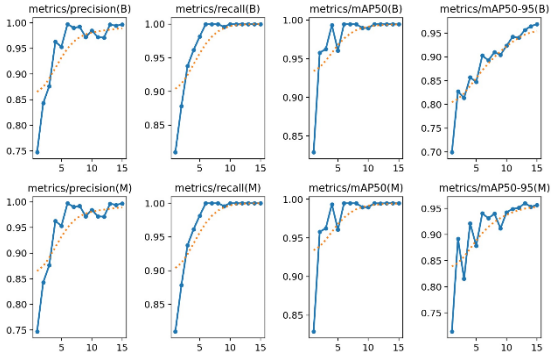

Fig. 6*. Curvas de entrenamiento.*
Fuente: *elaboración propia.*

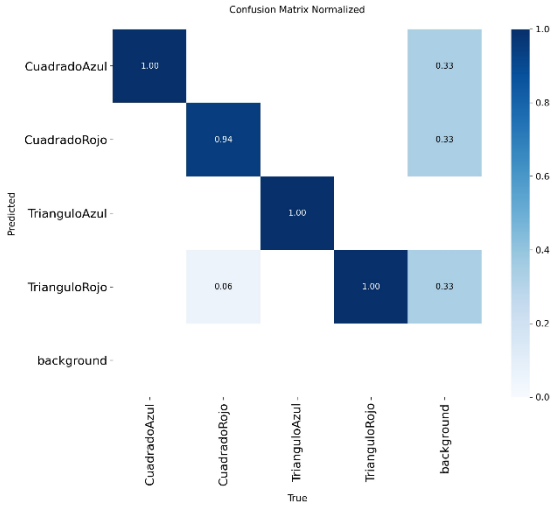

Fig. 7*. Matriz de confusión normalizada.*
Fuente: *elaboración propia.*

En la evaluación final para detección por cajas, el modelo alcanzó precision(B)=0,99672, recall(B)=1, mAP50(B)=0,995 y mAP50-95(B)=0,96917; mientras que para segmentación obtuvo precision(M)=0,99672, recall(M)=1, mAP50(M) =0,995 y mAP50-95(M)=0,95665. Estos valores reflejan un desempeño alto en la identificación de las figuras, tanto en localización como en segmentación. En la Figura 7, la relación con la diagonal muestra que CuadradoRojo tiene 0,94 en su clase, mientras que las demás figuras alcanzan 1;

en la relación con el fondo, TrianguloAzul tiene 0 y las demás clases 0,33.

Por último, se implementó los marcadores ArUco, con estos se obtiene la proporción pixel/cm y junto a la máscara que entrega YOLO se determina el área superficial de las figuras. Para evaluar esta fase se realizaron 120 medidas en total, 10 a cada figura, la prueba tenía dos partes, una fue la clasificación de YOLO y evaluar si era correcta o incorrecta, otra fue la medición del área y su error respecto a la medida real, para evitar fluctuaciones se registraba el área durante cinco segundos y se tomaba el promedio. La clasificación tuvo una precisión del 100%, por lo que solo se va a detallar el error del área, los resultados se presentan en la Tabla 4.

Tabla 4: *Precisión del área*

Figura	**Cuadrado**					
Color	**Rojo**			**Azul**		
Longitud (cm)	2	3	4	2	3	4
Área promedio (cm^2)	3,96	8,91	15,62	3,98	8,86	15,28
Error promedio (%)	2,77	1,84	2,35	2,59	2,91	4,51
Error máximo (%)	6,37	2,92	3,95	4,80	6,05	8,12
Figura	**Triangulo**					
Color	**Rojo**			**Azul**		
Longitud (cm)	2	3	4	2	3	4
Área promedio (cm^2)	1,92	4,28	7,68	1,91	4,31	7,62
Error promedio (%)	4,20	4,89	4,02	4,42	4,27	4,70
Error máximo (%)	10,33	7,55	6,59	8,91	6,99	7,29

Fuente: *elaboración propia*

3.3. Programación del módulo de transporte

Se caracterizaron los sensores y actuadores que posee el módulo, cuenta con dos finales de carrera, uno para para cada extremo, un motor con dos marchas, una para cada sentido, y dos interruptores; el PLC se estableció en la IP 192.168.10.1, en el rack 0 y slot 1, y se configuraron las opciones necesarias para permitir la comunicación externa.

El proyecto en TIA Portal se programó en Ladder (ver Fig. 8), el programa se divide en tres secciones: la primera es de enclave, habilita el uso del motor, utiliza los interruptores para iniciar y detener el enclave; la segunda es de actuadores, una variable virtual controla el motor, una variable para cada marcha, si se detecta el final de carrera el motor se detiene automáticamente; la tercera es de sensores, el estado de los finales de carrera se copia cada uno a una variable virtual.

Posteriormente se programó la lógica principal en Python, utilizando Snap7 para conectarlo con el PLC. Además de la conexión inicial se crearon tres secuencias: mover a la derecha, la cual activa la marcha hacia esa dirección hasta detectar el final de carrera correspondiente, momento en el cual detiene el motor; mover a la izquierda, la cual cumple la misma función que la anterior, pero en sentido contrario, es decir hacia la izquierda; centrar, el cual ejecuta la secuencia mover a la izquierda, cuando se completa activa la marcha derecha durante un tiempo predeterminado y la desactiva, se hizo de esa forma ya que el módulo no posee un final de carrera central.

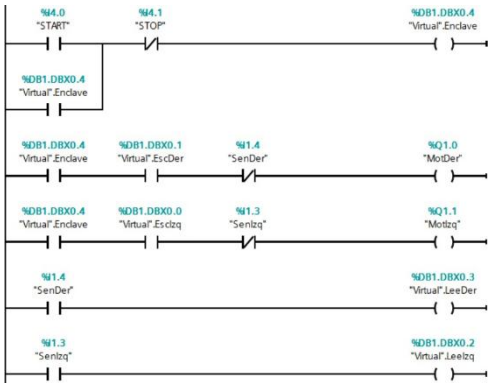

Fig. 8*. Proyecto en TIA Portal.*
Fuente: *elaboración propia.*

Para evaluar esta fase se midió la repetibilidad del centrado, para ello se ejecutó la secuencia 30 veces, tomando como referencia la posición de uno de los extremos del portaobjetos, y midiendo cada vez la distancia entre la referencia y la nueva posición del portaobjetos, se obtuvo un promedio de 0,673 cm y un máximo de 1cm. Aunque existe una pequeña variación entre ejecuciones, esta se mantiene dentro de un margen reducido respecto al tamaño total del portaobjetos, por lo que no afecta de manera significativa el funcionamiento global del sistema ni compromete la operación del prototipo.

3.4. Desarrollo de la aplicación final

Una vez comprobado cada sistema por separado, se integraron en una única interfaz gráfica, que centra la configuración y operación del prototipo, se diseñó para que su uso fuera sencillo, de modo que el usuario se familiarice rápidamente. La aplicación (ver Fig. 9) posee dos configuraciones principales:

la clase de la figura, donde se puede seleccionar para cada dirección (derecha e izquierda) una de las figuras (cuadrado azul, cuadrado rojo, triangulo azul o triangulo rojo); y el tamaño, en la cual se ingresan dos valores, el área objetivo en centímetros cuadrados y el margen de error en porcentaje. A parte de estas configuraciones también permite cambiar la resolución de la cámara, realizar un recorte a la imagen, cambiar el tipo de ArUco y un botón que despliega la ventana de ayuda.

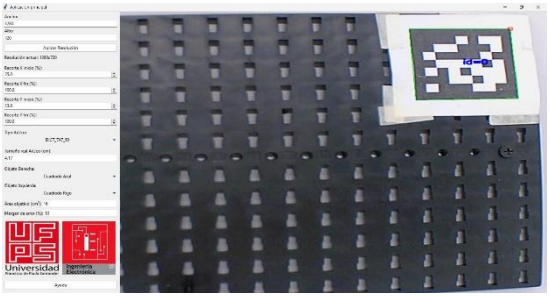

Fig. 9*. Aplicación final.*
Fuente: *elaboración propia.*

El flujo del programa consiste en tres secciones principales: el inicio, donde se establecen las comunicaciones con los dispositivos y se construye la interfaz gráfica; el centrado, donde el robot se envía a la posición predeterminada y el portaobjetos se centra; y finalmente la lógica principal, un bucle donde continuamente se extrae la elección del usuario y se compara con la figura detectada, si coinciden empieza el proceso, y una vez terminado vuelve a empezar el ciclo.

El bucle consiste en esperar la detección del ArUco para calcular la proporción píxel/cm, luego espera la detección de una única figura y comprueba que esta se mantenga en el tiempo para evitar falsos positivos; una vez verificada se compara la clase y el tamaño con los objetivos seleccionados por el usuario, y en caso de coincidir se calcula la longitud lateral a partir del área objetivo para seleccionar la secuencia del robot apropiada y ejecutarla, al completar la maniobra el robot vuelve a su posición predeterminada. Por último, la cinta activa la dirección correspondiente y, tras un tiempo se centra y reinicia el ciclo. En la Fig. 10 se representa este algoritmo.

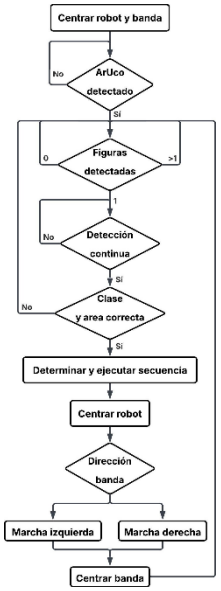

Fig. 10*. Diagrama de flujo del sistema.*
Fuente: *elaboración propia.*

4. CONCLUSIONES

El prototipo tuvo una inversión aproximada de 25.740.000 COP, considerando el brazo robótico OpenBot v1, la cámara Fischertechnik, la impresora 3D Snapmaker Original, el material PLA y el módulo de transporte. Este valor resulta elevado para un montaje académico básico, pero sigue siendo menor que una celda industrial comercial con capacidades similares. Además, su principal aporte no reside únicamente en su costo, sino en la posibilidad de disponer de un entorno funcional, flexible y reconfigurable para pruebas, formación y validación de estrategias.

El modelo YOLO presentó una precisión de clasificación del 100 %, sin embargo, este resultado solo es aplicable a este caso concreto, bajo condiciones controladas de iluminación. Incluso así, durante la operación en tiempo real se registraron fallos puntuales; por ejemplo, al posicionar las figuras, el sistema llegó a identificar las manos como si fueran figuras. Por ello, en un entorno real sería necesario adecuar un espacio específico para que el sistema tenga un rendimiento óptimo. Los errores de la medición del área con YOLO y ArUco fueron en promedio entre el 2 % y el 4,5 %, siendo el error máximo en toda la prueba de 10,33 %; esta precisión es suficiente para separar correctamente los distintos tamaños.

La comunicación entre el PLC y Python fue estable y eficiente, aprovechando de este modo la robustez del S7-1200 para manejar los sensores y actuadores

y la flexibilidad de Python para crear algoritmos complejos. La secuencia de centrar el portaobjetos al realizarse por software, crea una variación en la posición, esta es de máximo 1 cm, lo que considero aceptable para las aplicaciones de este proyecto.

El sistema confirmó la viabilidad del prototipo al lograr un flujo funcional de detección, validación, manipulación y transporte. No obstante, durante su desarrollo y validación se identificaron limitaciones relacionadas con la sensibilidad a las condiciones de iluminación, la detección de objetos ajenos en el campo visual, la variación en el agarre de figuras puntiagudas y la dependencia de un centrado por software en el módulo de transporte. Estas comprometen totalmente el sistema, pero sí evidencian oportunidades de mejora. Como trabajo futuro se propone ampliar el conjunto de datos con escenarios más diversos, optimizar el agarre del brazo robótico, incorporar mediciones más robustas e implementar un sensor dedicado para el centrado del módulo de transporte.

REFERENCIAS

[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/