Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

SIMULACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL EN UNA ESTACIÓN MULTI-ROBOTS ABB

Autor:

Heras García, Alejandro

Tutor(es):

Herreros López, Alberto

Departamento de Ingeniería de Sistemas y Automática

Valladolid, Mayo 2025.

Resumen

En este proyecto se va a desarrollar la implementación de una estación multi-robot para el transporte, clasificación y manipulación de una serie de piezas geométricas que simularan las diferentes piezas que pueden ser procesadas en un proceso industrial.

Para poder desarrollar este proceso es necesario simular tanto la estación de trabajo con su funcionamiento, como la obtención de imágenes de las piezas para su posterior análisis.

En el proyecto se generarán diferentes piezas que serán transportadas por una cinta hasta un punto donde se tomara una imagen de la misma. Esa imagen será analizada y en función de las características de la pieza, esta será clasificada y el robot que gestione ese tipo de pieza la colocará en la posición que le corresponda.

Palabras clave

ABB, Robotstudio, Matlab, OPC, Multi-robot, visión artificial, análisis de imágenes, cinta transportadora.

Abstract

This project focuses on the implementation of a multi-robot station for the transportation, classification, and manipulation of a series of geometric parts that simulate the different components typically processed in an industrial environment.

To successfully carry out this process, it is necessary to simulate both the workstation and its operation, as well as the acquisition of images for later analysis.

In the project, various parts will be generated and transported along a conveyor belt to a point where an image of each part will be captured. This image will then be analyzed, and based on the characteristics of the part, it will be classified and placed in the appropriate location by the robot responsible for handling that specific type of part.

Keywords

ABB, Robotstudio, Matlab, OPC, Multi-robot, artificial vision, image analysis, conveyor belt.

ÍNDICE

Índice de figuras:

2. Introducción

En la actualidad, la robótica industrial está adquiriendo un papel cada vez más relevante en los entornos de producción. Cada vez más fábricas están comenzando a implantar soluciones robóticas como una herramienta de apoyo a los operarios, especialmente en aquellas tareas que implican un alto grado de repetitividad o que pueden suponer un riesgo para la salud a largo plazo. La automatización de estos procesos no solo busca aumentar la eficiencia y la productividad, sino también mejorar las condiciones laborales, minimizando la exposición de los trabajadores a actividades que pueden derivar en lesiones por movimientos repetitivos, estrés físico o fatiga acumulada.

Dentro de este contexto, el presente proyecto se centra en el desarrollo y simulación de una tarea de manipulación de piezas geométricas. Se trata de un tipo de trabajo que, en un entorno de fabricación real, sería un excelente candidato para su automatización mediante el uso de robots industriales. Automatizar la manipulación de piezas no solo incrementa la velocidad de producción, sino que también garantiza una mayor precisión y uniformidad en las operaciones, aspectos fundamentales para mantener altos estándares de calidad en cualquier planta de producción.

La detección y clasificación de las piezas se llevará a cabo a través de la simulación de un sistema de visión artificial. Para ello, se realizará una captura de pantalla del entorno de trabajo, la cual será posteriormente analizada mediante un programa desarrollado en Matlab. Este programa procesará los datos de la imagen para extraer información relevante acerca de las características de las piezas, tales como su forma y posición. Una vez procesada la imagen, el sistema se comunicará con el controlador de la estación de trabajo mediante una conexión OPC (OLE for Process Control). A través de esta comunicación, el controlador recibirá los datos necesarios para identificar el tipo de figura detectada, su orientación y su ubicación dentro del área de trabajo, permitiendo así tomar decisiones sobre el brazo robótico más adecuado para realizar la manipulación de la pieza.

La incorporación de sistemas de visión artificial en la industria moderna ofrece un abanico muy amplio de aplicaciones. Entre sus principales ventajas se encuentran la mejora de la trazabilidad de los productos a lo largo de toda la cadena de suministro, la detección temprana de productos defectuosos que no cumplen con las especificaciones técnicas, así como la capacidad de clasificar e identificar piezas de manera automática. Estas funcionalidades no solo optimizan el control de calidad, sino que también permiten una gestión mucho más eficiente de los recursos de producción, reduciendo tiempos de parada y aumentando la fiabilidad de los procesos.

A medida que la tecnología avanza, los sistemas de visión artificial se han convertido en una herramienta indispensable para la industria 4.0, contribuyendo de forma decisiva a la transformación digital de las fábricas. Gracias a la combinación de robótica, sistemas de control avanzados y visión artificial, es posible crear entornos de producción más flexibles, inteligentes y adaptativos, capaces de responder de manera ágil a las necesidades cambiantes del mercado. En este sentido, el presente proyecto se enmarca en una tendencia global hacia la automatización y digitalización de los procesos productivos, demostrando la viabilidad y los beneficios de integrar soluciones basadas en visión artificial en estaciones de trabajo robotizadas.

3. Motivación personal

La elección del tema de este Trabajo de Fin de Grado surge del interés personal por las tecnologías emergentes y su aplicación en entornos industriales automatizados. Desde el inicio de mis estudios universitarios, siempre he sentido una gran atracción hacia la innovación tecnológica y su impacto en los procesos industriales. En particular, la simulación de un sistema de visión artificial en una estación multi-robots ABB representa una oportunidad única para integrar de forma práctica los conocimientos adquiridos a lo largo de la carrera, y de esta forma, comprobar cómo la teoría aprendida en las aulas puede convertirse en soluciones funcionales para el mundo real.

El proyecto desarrollado permite aplicar de manera directa los contenidos estudiados en asignaturas clave como Visión Artificial, Robótica y, en menor medida, Comunicación Industrial. Cada una de estas áreas ha proporcionado las herramientas teóricas y prácticas necesarias para abordar los diferentes desafíos que plantea el proyecto, y su integración ha resultado fundamental para lograr una simulación realista y operativa. La Visión Artificial ha sido esencial para el procesamiento de imágenes y la identificación de piezas, mientras que la Robótica ha permitido programar los movimientos de los robots y diseñar estrategias de manipulación eficientes. Por otro lado, la Comunicación Industrial ha brindado los conocimientos básicos para establecer la conexión entre el sistema de análisis de imágenes y el entorno de control de los robots.

Durante el desarrollo del trabajo, se simula un entorno industrial en el que una cinta transportadora traslada piezas geométricas hasta un punto de captura de imagen. En este punto, se utiliza MATLAB para analizar las imágenes obtenidas, procesando las características de las piezas como su forma, tamaño o posición. Una vez clasificadas, en el entorno de simulación RobotStudio de ABB, se modela la actuación de distintos brazos robóticos que manipulan las piezas y las distribuyen según su clasificación en las posiciones correspondientes. Esta combinación de tecnologías refleja de forma clara el potencial de la automatización avanzada y permite experimentar de manera práctica con sistemas que actualmente están transformando la industria manufacturera.

El principal motor de motivación en la realización de este proyecto ha sido la posibilidad de observar cómo conceptos teóricos y técnicas abstractas cobran vida en un entorno virtual y funcional. Poder ver materializado el flujo completo de captura, procesamiento, toma de decisiones y acción robótica, ha supuesto una experiencia sumamente enriquecedora. A lo largo del proyecto, han surgido numerosos retos técnicos, especialmente en la integración entre distintos softwares, el establecimiento de comunicaciones fiables y el ajuste de los algoritmos de procesamiento de imagen para lograr resultados precisos y consistentes. Enfrentar y superar estos desafíos ha supuesto un aprendizaje

invaluable, que difícilmente se habría podido adquirir exclusivamente mediante el estudio teórico.

Además, este Trabajo de Fin de Grado representa una excelente preparación para futuros entornos laborales, en los que la automatización inteligente y la integración de tecnologías avanzadas jugarán un papel protagonista. La demanda de profesionales capacitados en el uso de herramientas como MATLAB, RobotStudio y protocolos de comunicación industrial no deja de crecer en sectores como la automoción, la logística, la electrónica o la fabricación avanzada. Haber trabajado en un proyecto que combina estos aspectos proporciona una base sólida para afrontar futuros retos profesionales en entornos dinámicos, automatizados y altamente competitivos.

Por último, la elección de este tema también responde al deseo personal de contribuir, en un futuro próximo, a la transformación digital de la industria. Considero que el desarrollo de soluciones automatizadas basadas en visión artificial y robótica no solo mejora la productividad y la calidad de los procesos, sino que también promueve entornos de trabajo más seguros y sostenibles. Poder formarme y especializarme en este campo me motiva enormemente, ya que implica trabajar en la vanguardia tecnológica y participar activamente en la evolución hacia la industria del mañana.

En definitiva, este proyecto no solo ha supuesto una oportunidad para aplicar conocimientos técnicos, sino que ha sido una experiencia profundamente motivadora, que refuerza mi interés por continuar especializándome en el ámbito de la automatización industrial y la robótica avanzada, campos que sin duda marcarán el futuro de la ingeniería.

4. Estado del arte

4.1. Robótica

4.1.1. Introducción a la Robótica

La robótica es una rama multidisciplinaria que integra conocimientos de mecánica, electrónica, informática y control automático con el objetivo de diseñar y construir sistemas robóticos. Estos sistemas, conocidos como robots, están diseñados para ejecutar tareas de forma autónoma o con supervisión parcial, lo que permite aumentar la productividad, la precisión y la seguridad en múltiples ámbitos. El progreso en esta área ha sido impulsado por innovaciones en sensores, mecanismos de actuación, algoritmos de control e inteligencia artificial (Barrientos, Peñín, Balaguer, & Aracil, 2007).

Con el paso del tiempo, el uso de la robótica se ha diversificado más allá del entorno industrial, donde se emplea principalmente para mejorar la eficiencia y disminuir los costes de producción. Hoy en día, también tiene aplicaciones relevantes en sectores como la medicina, especialmente en intervenciones quirúrgicas de alta precisión, o en misiones espaciales, donde los robots pueden desenvolverse en condiciones extremas. Esta sinergia tecnológica permite que los robots trabajen en entornos hostiles o inaccesibles para el ser humano, demostrando su enorme potencial para transformar industrias y contribuir significativamente al bienestar social.

4.1.2. Historia de la Robótica

La robótica, tal como la entendemos hoy, es el resultado de un proceso histórico de evolución tecnológica que ha integrado múltiples disciplinas a lo largo del tiempo. Desde la antigüedad hasta la actualidad, la humanidad ha perseguido el sueño de crear máquinas capaces de replicar ciertas capacidades humanas. Esta aspiración ha pasado de los mitos y las invenciones mecánicas rudimentarias a sistemas sofisticados dotados de inteligencia artificial, sensores y capacidad de toma de decisiones autónoma. La historia de la robótica no solo ilustra el desarrollo técnico de los autómatas y robots, sino también el avance del pensamiento científico y de la ingeniería en general (Barrientos, Peñín, Balaguer, & Aracil, 2007).

Orígenes históricos: mitos y autómatas mecánicos

Las raíces de la robótica se encuentran en los relatos mitológicos y en los primeros experimentos mecánicos. Civilizaciones antiguas como la griega, la egipcia y la china ya concebían figuras animadas o dispositivos automáticos. Por ejemplo, en la mitología griega, el dios Hefesto forjaba autómatas de oro para ayudarle en su trabajo, mientras que el mito de Pigmalión representaba la

creación artificial con cualidades humanas. Aunque estos relatos son alegóricos, reflejan una fascinación temprana por la idea de crear seres artificiales.

En la realidad histórica, los primeros autómatas fueron construidos durante la antigüedad y la Edad Media. Herón de Alejandría, en el siglo I a.C., describió diversos mecanismos automáticos en sus tratados, como puertas que se abrían solas mediante sistemas hidráulicos o teatrillos automáticos. Más adelante, en el mundo islámico medieval, figuras como Al-Jazari (siglo XIII) diseñaron mecanismos complejos como relojes automáticos y autómatas músicos.

Durante el Renacimiento y los siglos posteriores, el interés por los autómatas continuó. Leonardo da Vinci diseñó un caballero mecánico que podía moverse mediante un sistema de poleas y engranajes. Ya en el siglo XVIII, relojeros e ingenieros europeos como Jacques de Vaucanson y Pierre Jaquet-Droz desarrollaron muñecos mecánicos que escribían, tocaban instrumentos o simulaban acciones humanas con sorprendente realismo. No obstante, estos dispositivos eran puramente mecánicos y carecían de cualquier forma de control automático, por lo que no pueden considerarse robots en sentido moderno.

Figura 1. Autómata de Pierre Jaquet-Droz

Revolución industrial: automatización e ingeniería de control

La llegada de la Revolución Industrial en el siglo XVIII marcó un punto de inflexión en la historia de la tecnología y de la robótica. El desarrollo de la máquina de vapor, la mecanización de los procesos productivos y la aparición

de fábricas dieron lugar a un nuevo paradigma: la automatización de tareas repetitivas a través de maquinaria. Aunque estas máquinas no eran robots, sí introdujeron conceptos esenciales como el control automático y la sustitución del trabajo manual por sistemas mecánicos.

Durante los siglos XIX y XX, surgieron los fundamentos de la teoría de control, esencial para el posterior desarrollo de los robots. James Clerk Maxwell formuló en 1868 los principios del control automático en sistemas físicos, lo que permitiría décadas más tarde el desarrollo de servomecanismos y reguladores. Al mismo tiempo, los avances en electricidad y electrónica posibilitaron el diseño de sistemas de control más precisos y adaptables, base fundamental de la automatización moderna.

En paralelo, el concepto de robot comenzaba a tomar forma en el ámbito cultural. El término "robot" fue utilizado por primera vez por el escritor checo Karel Čapek en su obra teatral *R.U.R. (Rossum's Universal Robots)*, estrenada en 1921. En la obra, los robots eran seres artificiales creados para trabajar en lugar de los humanos, una visión que, aunque ficticia, anticipaba los dilemas éticos y sociales asociados al desarrollo de la robótica.

La robótica moderna: del brazo manipulador a la inteligencia artificial

La robótica como disciplina científica comenzó a consolidarse en la segunda mitad del siglo XX, especialmente a partir de la década de 1950, cuando se combinaron los avances en electrónica, teoría de control y computación. En esta etapa temprana, los primeros desarrollos se centraron en manipuladores industriales que podían repetir trayectorias de manera automática. Uno de los hitos más importantes fue la creación del Unimate, el primer robot industrial, diseñado por George Devol y Joseph Engelberger. Este robot fue instalado en una fábrica de General Motors en 1961 para realizar tareas de manipulación de piezas calientes, marcando el inicio de la automatización robotizada en la industria.

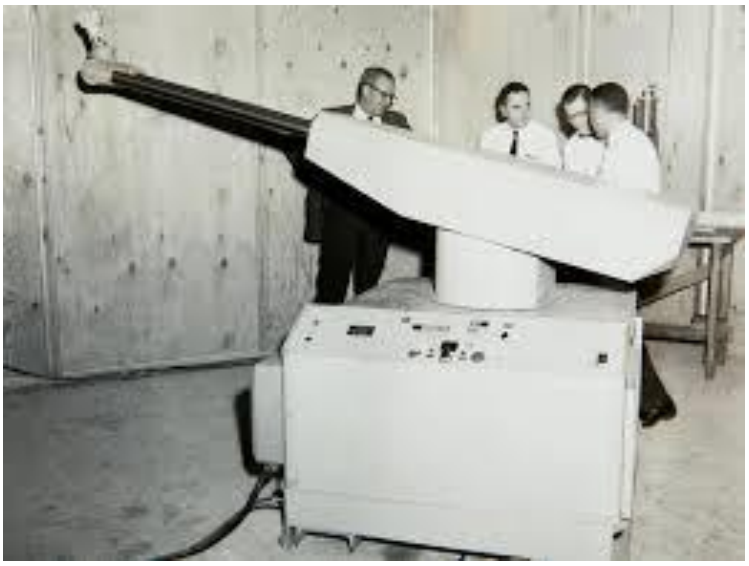

Figura 2. Presentación de Unimate

Desde entonces, el desarrollo de la robótica industrial ha estado estrechamente ligado a las necesidades de la producción. En sus primeras generaciones, los robots eran dispositivos programables de forma muy limitada, que seguían trayectorias predefinidas sin capacidad de adaptación al entorno. Sin embargo, con la incorporación de sensores y control en lazo cerrado, fue posible mejorar la precisión y flexibilidad de estos sistemas.

En paralelo, se fueron desarrollando nuevas estructuras robóticas: robots cartesianos, cilíndricos, esféricos, SCARA (Selective Compliance Assembly Robot Arm) y robots antropomórficos (de 6 grados de libertad), que son los más utilizados hoy en día en la industria. Cada una de estas configuraciones fue pensada para satisfacer diferentes requisitos de espacio de trabajo, precisión, velocidad y capacidad de carga.

Figura 3. Robot SCARA

En la década de 1980, se produjeron avances significativos en la integración de la robótica con sistemas informáticos más potentes. La aparición del lenguaje de programación RAPID (en el caso de ABB) y otros similares permitió desarrollar aplicaciones más complejas y adaptables. Al mismo tiempo, el campo de la inteligencia artificial comenzaba a explorarse en robótica, aunque con muchas limitaciones tecnológicas en aquella época.

En esta etapa, surgieron también aplicaciones fuera del ámbito industrial, como la robótica médica, la robótica móvil, la robótica de servicio y la robótica espacial. La NASA, por ejemplo, desarrolló robots como el Mars Rover para la exploración planetaria, y en medicina comenzaron a aparecer sistemas robóticos quirúrgicos asistidos por ordenador, como el Da Vinci Surgical System.

Robótica contemporánea: integración inteligente y entornos colaborativos

A partir de los años 2000, la robótica ha entrado en una fase de madurez tecnológica e integración con otras ramas como la inteligencia artificial, el aprendizaje automático, la visión por computador y la interconectividad digital. Este avance ha dado lugar a robots más autónomos, con capacidad para percibir su entorno, adaptarse a condiciones cambiantes y tomar decisiones en tiempo real.

Uno de los desarrollos más relevantes de las últimas décadas es la aparición de los robots colaborativos o *cobots*, que pueden trabajar en estrecha proximidad con los seres humanos sin necesidad de barreras físicas. Estos robots están dotados de sensores y algoritmos de seguridad que permiten detener su movimiento ante el contacto con personas u obstáculos. Esta nueva generación de robots ha abierto nuevas posibilidades de automatización en entornos donde la intervención humana sigue siendo necesaria, como tareas de montaje, inspección o empaquetado.

Asimismo, la robótica móvil ha experimentado un gran crecimiento. Robots autónomos capaces de desplazarse por entornos no estructurados, como los vehículos de reparto, los robots de limpieza o los sistemas de transporte interno en almacenes, son cada vez más comunes gracias a los avances en mapeo, navegación autónoma y localización simultánea (SLAM).

En el campo académico y de investigación, los esfuerzos se centran hoy en resolver retos como la manipulación de objetos deformables, la planificación dinámica de trayectorias, la interacción hombre-máquina o el desarrollo de robots sociales. También han emergido nuevas áreas como la biorrobótica, que busca replicar capacidades animales o humanas a través de sistemas artificiales, y la nanorrobótica, orientada a intervenciones a escala microscópica, especialmente en medicina.

Figura 4. Robot Da Vinci

Futuro de la robótica: desafíos y oportunidades

El futuro de la robótica se presenta como un campo lleno de oportunidades, pero también de importantes retos éticos, sociales y tecnológicos. A medida que los robots adquieren mayores capacidades, surge la necesidad de definir marcos normativos que regulen su uso, aseguren la protección de los trabajadores y garanticen un desarrollo justo de la automatización. Cuestiones como la pérdida de empleo, la seguridad de los sistemas autónomos o la responsabilidad en caso de fallos, están en el centro del debate actual.

Además, es necesario seguir trabajando en aspectos técnicos como la autonomía energética, la miniaturización, la mejora de algoritmos de percepción y razonamiento, y la interacción natural con los usuarios. También se prevé una mayor integración de los robots en entornos conectados, donde puedan interactuar con otros dispositivos inteligentes mediante tecnologías como el Internet de las Cosas (IoT) y los sistemas ciberfísicos.

Desde el punto de vista educativo, la robótica también está jugando un papel fundamental. El aprendizaje de conceptos de programación, control y diseño robótico se ha integrado en niveles educativos diversos, desde la educación primaria hasta la universidad. Esta tendencia apunta a una sociedad cada vez más familiarizada con las tecnologías robóticas y con un creciente número de profesionales capacitados para diseñarlas, programarlas y gestionarlas.

4.1.3. Robótica Industrial

Un robot industrial se define como un manipulador multifuncional, reprogramable y controlado automáticamente, diseñado para mover materiales, piezas, herramientas u otros dispositivos a través de movimientos programados y variables con el fin de realizar tareas diversas en entornos industriales. Esta definición enfatiza la flexibilidad, la capacidad de adaptación a distintas aplicaciones y la posibilidad de ser reprogramado sin necesidad de modificaciones físicas en su estructura, lo que permite al robot industrial integrarse en distintos procesos productivos y adaptarse a las exigencias cambiantes de la industria moderna. (Barrientos, Peñín, Balaguer, & Aracil, 2007).

Aplicaciones de los robots industriales

Los robots industriales se han expandido en múltiples sectores productivos, desde la industria pesada hasta ámbitos más especializados como la microelectrónica o la biotecnología. Esta versatilidad se debe a su capacidad para adaptarse a una amplia gama de tareas mediante el uso de diferentes herramientas, sistemas de control y sensores. A continuación, se detallan algunas de las principales aplicaciones:

1. Manipulación de materiales

Una de las aplicaciones más extendidas es la manipulación de piezas y productos. Los robots son capaces de mover cargas desde una estación a otra, ordenar productos en palés (paletizado) o retirar elementos de una línea (despaletizado). Estas tareas, que implican movimientos repetitivos y potencialmente peligrosos, se automatizan con facilidad, reduciendo el esfuerzo físico del personal humano.

2. Montaje y ensamblaje

El ensamblaje de componentes en líneas de producción se ha beneficiado enormemente del uso de robots. Estos sistemas pueden realizar tareas de unión, atornillado o encaje con gran precisión y repetibilidad. En sectores como la automoción o la electrónica, esta aplicación permite fabricar productos con tolerancias muy ajustadas, manteniendo altos niveles de calidad.

3. Soldadura

La soldadura, tanto por puntos como por arco, es otra tarea comúnmente automatizada. Los robots de soldadura trabajan a gran velocidad, garantizando cordones de soldadura uniformes y reduciendo las imperfecciones. Además,

mejoran las condiciones de seguridad al minimizar la exposición del operario al calor, chispas y humos.

4. Pintura y recubrimientos

La aplicación automatizada de pintura industrial exige precisión y uniformidad. Los robots pintores están programados para cubrir superficies de forma homogénea, incluso en geometrías complejas. Esto no solo mejora el acabado superficial, sino que también reduce el desperdicio de material y los riesgos de exposición a productos químicos.

5. Procesamiento de materiales

En tareas como el corte, fresado o taladrado, los robots industriales permiten operaciones precisas en múltiples materiales, incluyendo metal, plástico o vidrio. Al automatizar estos procesos se logra mayor flexibilidad frente a las máquinas herramienta tradicionales, ya que los robots pueden adaptarse con facilidad a cambios en el diseño de la pieza.

6. Control de calidad e inspección

Los robots también se utilizan en el control automatizado de calidad, integrando sensores y sistemas de visión artificial para detectar defectos superficiales o dimensionales. Esta inspección automática permite garantizar productos conforme a los estándares y facilita una trazabilidad continua en el proceso de fabricación.

7. Aplicaciones especiales

En ambientes peligrosos, como zonas radiactivas, cámaras estériles o atmósferas explosivas, los robots industriales pueden operar sin riesgo para los seres humanos. Asimismo, en entornos donde se requiere extrema limpieza o precisión, como en la fabricación de chips o fármacos, los robots garantizan resultados consistentes y condiciones higiénicas óptimas.

Características generales de los robots industriales

Los robots industriales están diseñados con una serie de características técnicas que definen su comportamiento, alcance operativo y tipo de aplicaciones posibles. Según Barrientos et al. (2007), estas características permiten clasificar y seleccionar el robot más adecuado para cada tarea:

1. Grados de libertad

Cada robot tiene un número determinado de *grados de libertad*, que se corresponde con los ejes que permiten su movimiento en el espacio. Un robot con seis grados de libertad, por ejemplo, puede mover su herramienta en cualquier punto y orientación del espacio tridimensional. Esta flexibilidad es fundamental para tareas complejas de ensamblaje, soldadura o pintura.

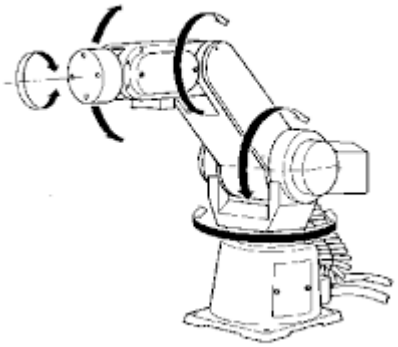

Figura 5. Ejemplo de Robot para mostrar los grados de libertad

2. Capacidad de carga

La capacidad de carga representa el peso máximo que puede transportar el robot, incluyendo su herramienta y cualquier pieza que manipule. Existen robots ligeros (menos de 5 kg) ideales para electrónica, y robots pesados (más de 500 kg) diseñados para levantar motores o estructuras de gran tamaño.

3. Área de trabajo

El *alcance* de un robot indica la distancia máxima desde su base hasta el extremo de su define el espacio en el que el extremo del robot puede operar sin tener en cuenta la herramienta. Es crucial para asegurar que el robot pueda alcanzar todos los puntos necesarios para completar su tarea.

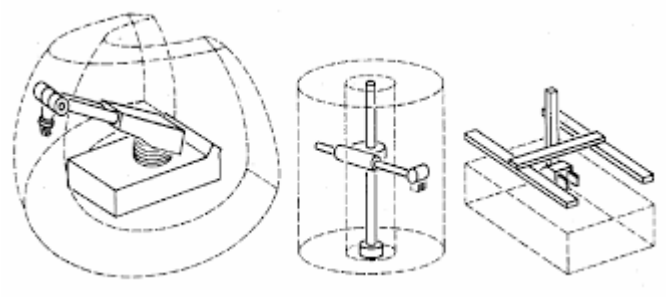

Figura 6. Ejemplos de diferentes tipos de robot con sus areas de trabajo

4. Repetitividad y precisión

La *repetitividad* mide la capacidad del robot para posicionarse en el mismo punto exacto tras múltiples movimientos. Es una característica crítica para procesos como el ensamblaje o la soldadura. La *precisión*, en cambio, se refiere a la capacidad de alcanzar un punto específico por primera vez. Aunque ambos conceptos están relacionados, es común priorizar la repetitividad en entornos industriales.

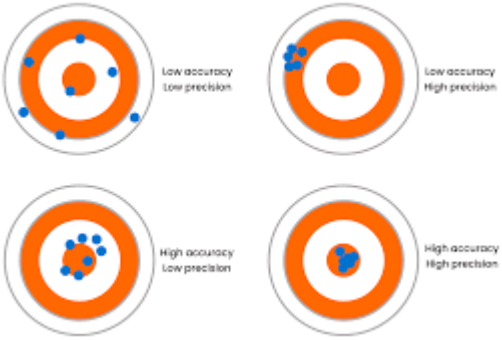

Figura 7. Explicación grafica de Repetitividad y precisión

5. Velocidad

La velocidad de trabajo de un robot afecta directamente a la productividad. Los robots industriales modernos pueden realizar movimientos complejos en fracciones de segundo. Sin embargo, esta velocidad debe gestionarse cuidadosamente para evitar problemas de seguridad o desgaste prematuro.

6. TCP (Tool Center Point)

Es el punto central de la herramienta del robot. Las coordenadas de este punto serán almacenadas en el programa.

7. Controlador y programación

El controlador es el sistema que dirige los movimientos del robot. Puede estar basado en microprocesadores dedicados o incluso en sistemas embebidos de propósito general. La programación de robots industriales puede realizarse mediante *flex pendant* (dispositivo manual), mediante lenguaje de programación específico (como RAPID en ABB o KRL en KUKA), o por simulación en entornos virtuales. Según Barrientos et al. (2007), la programación eficiente es clave para maximizar el rendimiento del robot y minimizar los tiempos de inactividad.

8. Sensores y realimentación

La integración de sensores permite que los robots industriales se adapten al entorno. Sensores de fuerza, proximidad, visión o temperatura aportan

información en tiempo real que el controlador utiliza para modificar trayectorias o ajustar esfuerzos. Esta *realimentación* es esencial en procesos como el ensamblaje con tolerancias estrechas o la manipulación de objetos deformables.

4.1.4. Clasificación de robots industriales

Los robots industriales pueden clasificarse según distintos criterios que reflejan sus características estructurales, funcionales y de aplicación. A continuación, se presentan las clasificaciones más relevantes.

Atendiendo a la generación

La generación de un robot está determinada por el nivel tecnológico disponible en el momento de su desarrollo. La transición entre generaciones ocurre cuando se produce un avance significativo en las capacidades del robot (Barrientos, Peñín, Balaguer, & Aracil, 2007). Según esta perspectiva, se reconocen las siguientes generaciones (Fraile Marinero, 2023):

- **Primera generación**: robots programados para ejecutar secuencias de tareas de forma fija, sin capacidad de percepción ni adaptación al entorno. Carecen de sensores, lo que limita su funcionalidad a entornos completamente estructurados.
- **Segunda generación**: incorporan sensores que les permiten captar información limitada del entorno, como posición de objetos o fuerzas de contacto, lo que les permite realizar ajustes en sus movimientos.
- **Tercera generación**: se programan mediante lenguajes de alto nivel o lenguaje natural. Son capaces de planificar tareas y tomar decisiones de forma autónoma gracias a funcionalidades inteligentes integradas.

Atendiendo al área de aplicación

Los robots industriales pueden dividirse según el ámbito en el que se utilizan y las tareas que desempeñan (Barrientos, Peñín, Balaguer, & Aracil, 2007):

- **Robots para automatización de la producción**: diseñados para procesos industriales repetitivos, como soldadura, pintura, manipulación de piezas, montaje, etc. Se caracterizan por su precisión, velocidad y fiabilidad en líneas de producción automatizadas.
- **Robots de servicios**: se aplican en tareas auxiliares como inspección, mantenimiento, logística o manipulación en entornos colaborativos. Contribuyen a mejorar la eficiencia operativa y la seguridad de los trabajadores.

Atendiendo al tipo de actuadores incorporados

El tipo de actuador determina las características dinámicas del robot: fuerza, precisión, velocidad, consumo energético, entre otras. Según el tipo de actuador, se distinguen tres categorías principales (Barrientos, Peñín, Balaguer, & Aracil, 2007):

- **Actuadores eléctricos**: ofrecen una alta precisión de posicionamiento y facilidad de control. Son comunes en tareas de montaje, ensamblaje y manipulación de componentes pequeños.
- **Actuadores hidráulicos**: proporcionan gran potencia, por lo que son ideales para trabajos de carga pesada, prensado o manipulación de piezas voluminosas en la industria pesada.
- **Actuadores neumáticos**: ligeros, rápidos y económicos, adecuados para tareas sencillas que requieren ciclos de trabajo breves y repetitivos.

Atendiendo a la configuración de sus ejes

La configuración de los ejes de un robot afecta directamente a su capacidad de movimiento, su flexibilidad y su alcance. Las principales configuraciones mecánicas son las siguientes (Barrientos, Peñín, Balaguer, & Aracil, 2007):

- **SCARA (Selective Compliance Assembly Robot Arm)**: poseen cuatro grados de libertad, con flexibilidad en el plano horizontal y rigidez en el eje vertical. Son ideales para aplicaciones de ensamblaje de alta velocidad.
- **Angulares o antropomórficos**: simulan la estructura del brazo humano mediante articulaciones rotacionales. Son versátiles y permiten movimientos complejos.
- **Cartesianos**: también conocidos como robots de pórtico, se mueven en ejes lineales X, Y y Z. Son apropiados para tareas que requieren desplazamientos rectilíneos y repetitivos.
- **Cilíndricos**: combinan movimientos lineales y rotacionales alrededor de un eje vertical. Su configuración es útil para operaciones de ensamblaje en espacios limitados.
- **Esféricos o polares**: articulados mediante un sistema de coordenadas polares. Su estructura les permite operar en un volumen esférico, útil en tareas de acceso amplio.

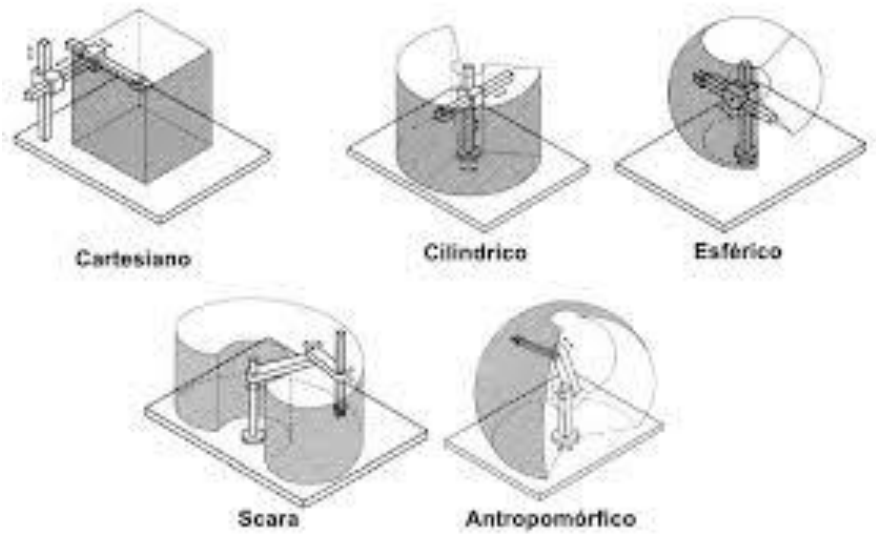

Figura 8. Ejemplo de diferentes Robots respecto a la configuración de sus ejes

Atendiendo al tipo de control

La clasificación por tipo de control se basa en cómo se gestionan los movimientos del robot, de acuerdo con la norma ISO 8373 y la definición de la Federación Internacional de Robótica (IFR) (Barrientos, Peñín, Balaguer, & Aracil, 2007):

- **Robot secuencial**: cada movimiento se ejecuta de manera independiente y consecutiva, bajo un esquema conocido como control punto a punto (PTP). Es típico en manipuladores neumáticos simples.
- **Robot controlado por trayectoria**: los movimientos de sus ejes están sincronizados siguiendo trayectorias definidas mediante interpolación. Se emplean en aplicaciones donde se requiere continuidad y precisión en el desplazamiento.
- **Robot adaptativo**: incorpora sensores y algoritmos que le permiten modificar sus acciones en función de la información del entorno. Puede emplear visión artificial o sensores táctiles para adaptarse a situaciones cambiantes.
- **Robot teleoperado**: controlado a distancia por un operador humano. Este tipo de robot es ideal en ambientes peligrosos o inaccesibles, como zonas radiactivas o misiones de exploración.

4.2. Visión artificial

4.2.1. Introducción a la visión artificial

En la era de la digitalización industrial y la automatización inteligente, la visión artificial ha emergido como una de las tecnologías más transformadoras dentro del ámbito de la ingeniería, la robótica y la industria 4.0. Esta disciplina, que combina el procesamiento de imágenes, inteligencia artificial y sensores ópticos, permite a las máquinas interpretar y comprender su entorno visual de forma automatizada, emulando el sentido de la vista humana. La capacidad de "ver" e interpretar información visual convierte a los sistemas informáticos y robóticos en agentes capaces de tomar decisiones basadas en datos visuales, lo que amplía significativamente sus posibilidades operativas (Wikipedia, 2025).

La visión artificial (o visión por computador) se apoya en técnicas matemáticas y computacionales para adquirir, procesar, analizar e interpretar imágenes del mundo real, con el objetivo de extraer información útil para diversas aplicaciones. Esta información puede ser utilizada para inspeccionar productos, guiar robots, controlar procesos de producción, identificar defectos, verificar dimensiones, reconocer patrones, entre muchas otras funciones. Gracias a los avances en sensores ópticos, hardware de procesamiento y algoritmos de inteligencia artificial, la visión artificial ha alcanzado una madurez tecnológica que la convierte en una herramienta clave para la competitividad industrial.

En los últimos años, el auge de la automatización avanzada ha impulsado la adopción de sistemas de visión artificial en prácticamente todos los sectores productivos. Desde la automoción hasta la farmacéutica, pasando por la electrónica, la logística y la alimentación, los sistemas de visión permiten aumentar la calidad, reducir los costes de producción, mejorar la trazabilidad y dotar a las máquinas de una mayor autonomía y adaptabilidad. Esto es especialmente importante en el contexto de la industria 4.0, donde la flexibilidad, personalización y eficiencia energética son factores críticos (Rapid Innovation, 2024).

El objetivo de este trabajo es ofrecer una visión amplia y profunda sobre la tecnología de visión artificial, explorando sus orígenes, evolución y principales aplicaciones industriales. Este conocimiento es esencial para ingenieros, técnicos y desarrolladores que buscan integrar soluciones de visión en sistemas automatizados o robóticos. Además, el estudio se complementa con un enfoque en la simulación, que permite probar y validar algoritmos de visión sin necesidad de disponer de los equipos físicos, facilitando el diseño, la programación y la puesta en marcha de sistemas complejos de manera segura, económica y eficiente.

La visión artificial no es solo una herramienta tecnológica, sino un habilitador fundamental de nuevas capacidades en los sistemas de producción inteligentes. Su integración en los procesos industriales representa un paso hacia una manufactura más inteligente, flexible y competitiva, alineada con las necesidades del presente y del futuro.

4.2.2. Historia de la visión artificial

El desarrollo de la visión artificial ha sido un proceso progresivo que ha evolucionado junto con el avance de la informática, los sensores y las técnicas de procesamiento digital de imágenes. La historia de esta disciplina está marcada por importantes hitos tecnológicos y conceptuales, que han permitido pasar de simples experimentos académicos a sistemas comerciales de alta precisión utilizados en fábricas y centros de producción (Wikipedia, 2025).

Origenes y primeras investigaciones

La visión artificial como campo de estudio comenzó a tomar forma en la década de 1950, en paralelo con el desarrollo de la inteligencia artificial. En sus inicios, los investigadores intentaron crear sistemas capaces de interpretar imágenes mediante algoritmos rudimentarios. Uno de los proyectos más citados es el “Summer Vision Project” del MIT en 1966, que pretendía dotar a un ordenador de la capacidad de describir una escena visual simple. Aunque los resultados fueron limitados, sentaron las bases para el desarrollo posterior de la disciplina.

En los años 70 y 80, el interés por la visión artificial creció con la aparición de los primeros ordenadores personales y el desarrollo de algoritmos más sofisticados para el procesamiento de imágenes. Se introdujeron técnicas como la detección de bordes, análisis de contornos, segmentación de imágenes y reconocimiento de patrones. Estos avances permitieron las primeras aplicaciones prácticas en industrias que requerían inspección de productos (Wikipedia, 2025).

Avances técnicos en la década de 1990

La década de 1990 fue clave para el despegue de la visión artificial como tecnología aplicable a la industria. La mejora en la calidad y la velocidad de las cámaras digitales, junto con el aumento en la capacidad de procesamiento de los ordenadores, permitió que los sistemas de visión comenzaran a implementarse en entornos industriales reales. Durante estos años, se desarrollaron bibliotecas de software como OpenCV, que democratizaron el acceso a herramientas de visión artificial para investigadores y desarrolladores.

Se popularizó el uso de cámaras CCD y CMOS, que ofrecían mayor resolución y menor coste. También se empezaron a implementar sistemas de visión 2D en

líneas de producción para tareas como control de calidad, lectura de códigos de barras y detección de defectos superficiales (Wikipedia, 2025).

Incorporación de la visión 3D y los sistemas inteligentes

Durante los años 2000, la visión artificial incorporó nuevos enfoques como la visión estereoscópica, la fotogrametría y los escáneres láser, permitiendo la obtención de información tridimensional del entorno. Esto abrió nuevas posibilidades para el guiado de robots, la metrología sin contacto y la reconstrucción de modelos digitales en 3D. Al mismo tiempo, la integración de sensores ópticos con robots industriales empezó a estandarizarse en plataformas de automatización de grandes fabricantes.

También se desarrollaron los primeros sistemas inteligentes, capaces de adaptarse a variaciones del entorno gracias a algoritmos de aprendizaje automático. Estos sistemas eran más robustos y menos sensibles a condiciones de iluminación o posicionamiento, lo que aumentó su fiabilidad y redujo su dependencia de configuraciones fijas (Wikipedia, 2025).

Revolución del deep learning y visión artificial moderna

La gran revolución de la visión artificial llegó con la irrupción del deep learning en la década de 2010. Las redes neuronales convolucionales (CNNs) demostraron ser extremadamente eficaces para tareas de clasificación de imágenes, detección de objetos y segmentación semántica. Gracias a la disponibilidad de grandes conjuntos de datos y al desarrollo de GPUs, los sistemas de visión basados en aprendizaje profundo alcanzaron niveles de precisión similares e incluso superiores a los del ojo humano en tareas específicas.

Actualmente, la visión artificial se apoya en frameworks como TensorFlow, PyTorch o YOLO para entrenar modelos capaces de operar en tiempo real. Además, tecnologías como la visión aumentada, la computación en el edge (edge computing) y los sistemas embebidos están llevando la visión artificial a nuevos dispositivos, desde teléfonos móviles hasta vehículos autónomos o robots colaborativos (cobots).

4.2.3. Aplicaciones Industriales de la visión artificial

La visión artificial tiene un impacto directo en la eficiencia, calidad y seguridad de los procesos industriales. A continuación, se describen las aplicaciones más relevantes en el entorno industrial moderno (EDS Robotics, 2020; Wikipedia, 2025):

Inspección y control de calidad

Una de las aplicaciones más frecuentes es la inspección visual automática de productos. Los sistemas de visión pueden verificar dimensiones, colores, texturas, formas, presencia o ausencia de componentes, entre otros parámetros. Esto permite detectar defectos de fabricación, garantizar la calidad del producto final y reducir los costes asociados a devoluciones o reprocesos (Ultralytics, 2024).

Ejemplo: En una línea de producción de botellas, un sistema de visión puede verificar si las etiquetas están correctamente colocadas, si las tapas están bien cerradas y si no hay impurezas en el líquido.

Guiado de robots

La visión artificial permite a los robots localizar objetos en entornos no estructurados y calcular su orientación para manipularlos con precisión. Este guiado visual (visual servoing) es fundamental en tareas de ensamblaje, pick and place, soldadura o corte.

Los sistemas pueden emplear cámaras montadas en el robot (eye-in-hand) o en el entorno (eye-to-hand), combinadas con procesamiento 2D o 3D para determinar la posición exacta de las piezas (Wikipedia, 2025).

Clasificación y selección automática

Los sistemas de visión pueden clasificar productos por características visuales como color, forma o tamaño, y enviarlos a diferentes líneas de producción o empaquetado. Esto se aplica, por ejemplo, en la industria alimentaria, el reciclaje y la farmacéutica (Wikipedia, 2025).

Ejemplo: En una planta de reciclaje, cámaras detectan y separan materiales según su color y composición, mejorando la eficiencia de la clasificación automática.

Lectura de códigos y trazabilidad

Los sistemas de visión se utilizan para la lectura de códigos de barras, códigos QR y textos mediante reconocimiento óptico de caracteres (OCR). Esta funcionalidad es crucial para garantizar la trazabilidad de los productos, desde la materia prima hasta el cliente final (Wikipedia, 2025).

La lectura se puede realizar incluso a altas velocidades y con códigos parcialmente dañados, lo que supera las limitaciones de los lectores láser tradicionales.

Medición y metrología sin contacto

La visión artificial permite realizar mediciones precisas sin necesidad de contacto físico. Se puede verificar el cumplimiento de tolerancias dimensionales, ángulos, radios de curvatura o alineación de componentes, de forma rápida y automatizada (EDS Robotics, 2020).

Esta función es muy útil en la industria automotriz y aeroespacial, donde se requiere un control dimensional estricto y repetitivo.

Verificación de montaje

Durante el ensamblaje de productos, la visión puede asegurar que cada pieza ha sido colocada correctamente. Se pueden detectar errores como falta de tornillos, polaridades incorrectas o piezas mal orientadas. Esta verificación puede ser realizada por estaciones fijas o por robots colaborativos equipados con cámaras (Wikipedia, 2025).

Control de procesos en tiempo real

La visión artificial permite monitorizar procesos productivos en tiempo real, detectando condiciones anómalas como roturas, atascos, desviaciones de temperatura o presión. Esto facilita la implementación de sistemas de mantenimiento predictivo, que anticipan fallos y evitan paradas no planificadas (Ultralytics, 2024).

Seguridad y prevención de riesgos

En entornos donde operan máquinas peligrosas, la visión artificial puede contribuir a la seguridad mediante la detección de presencia humana en zonas restringidas. Los sistemas pueden frenar o detener el funcionamiento de una máquina si detectan un operario en una zona peligrosa (Wikipedia, 2025).

Integración en sistemas inteligentes

La visión artificial no opera de forma aislada, sino que se integra cada vez más en sistemas ciberfísicos interconectados. Puede trabajar junto con sensores IoT, algoritmos de inteligencia artificial, big data e interfaces hombre-máquina, contribuyendo a la creación de fábricas inteligentes, adaptativas y sostenibles (Rapid Innovation, 2024).

4.2.4. Clasificación de los sistemas de visión artificial en el entorno industrial

Actualmente, en el ámbito industrial se utilizan diversas tecnologías de visión artificial adaptadas a diferentes niveles de complejidad y requerimientos. A continuación, se describen los sistemas más comunes empleados en entornos productivos (EDS Robotics, 2020; Elecproy, 2024).

- **Sensores de visión**

Se trata de dispositivos compactos diseñados para resolver tareas concretas y de baja complejidad. Superan las capacidades de los sensores fotoeléctricos tradicionales al ofrecer salidas binarias (como correcto/incorrecto) con mayor fiabilidad. Estos sensores integran óptica, iluminación, procesamiento y comunicación en un solo cuerpo, lo que los hace ideales para instalaciones rápidas y económicas.

Principales aplicaciones:

- Verificación de calidad básica.
- Confirmación de presencia o ausencia de componentes.
- Inspección de ensamblajes simples.

- **Cámaras inteligentes y sistemas integrados de visión**

Estas soluciones constituyen un paso adelante respecto a los sensores de visión. Incorporan sensores de imagen de alta resolución y procesadores internos potentes, lo que permite llevar a cabo tareas de inspección detalladas. Gracias a su capacidad de procesamiento autónomo, no requieren de un PC externo y pueden integrarse fácilmente en líneas automatizadas.

Principales aplicaciones:

- Control de calidad de precisión en procesos de montaje.
- Verificación de piezas complejas.
- Interacción con sistemas automatizados para una evaluación integral.

- **Sistemas de visión avanzados**

Representan las soluciones más completas en el ámbito de la visión artificial industrial. Utilizan hardware especializado y software avanzado capaz de procesar grandes volúmenes de datos en tiempo real. Están diseñados para operaciones donde se exige un control exhaustivo, análisis complejos y respuesta inmediata.

Principales aplicaciones:

- Control de calidad en sectores críticos como automoción, aeroespacial y electrónica.
- Inspección simultánea de múltiples parámetros de proceso.
- Operaciones en entornos que requieren precisión extrema y máxima fiabilidad.

• Sistemas de visión estéreo

Esta tecnología emplea dos cámaras dispuestas en diferentes posiciones para capturar imágenes desde múltiples ángulos. A partir de esta información, se genera una reconstrucción tridimensional del objeto, permitiendo calcular con exactitud distancias y profundidades. Su uso es clave en aplicaciones donde la percepción del entorno en 3D resulta esencial.

Principales aplicaciones:

- Navegación autónoma de robots móviles.
- Medición precisa de geometría en piezas complejas.
- Evaluación volumétrica y detección de defectos de forma.

• Visión tridimensional (3D)

Los sistemas de visión 3D capturan y procesan datos espaciales para construir modelos volumétricos de los objetos inspeccionados. Utilizan cámaras con sensores de profundidad o técnicas de proyección estructurada, entre otras. Esto permite obtener información geométrica completa del objeto, más allá de su apariencia superficial.

Principales aplicaciones:

- Modelado y análisis de estructuras tridimensionales.
- Inspección de productos con formas irregulares.
- Aplicaciones en entornos de realidad aumentada o realidad virtual.

• Visión térmica

Los sistemas térmicos utilizan cámaras infrarrojas para detectar la radiación de calor emitida por los objetos. Esta tecnología permite visualizar patrones térmicos invisibles al ojo humano, lo que resulta útil en condiciones de baja visibilidad, así como en tareas de diagnóstico térmico.

Principales aplicaciones:

- Detección de sobrecalentamientos o fallos eléctricos incipientes.

- Localización de personas o fuentes de calor en entornos adversos.
- Monitorización en áreas con humo, polvo o sin iluminación visible.

• Visión multiespectral

A través del uso de cámaras capaces de capturar diferentes bandas del espectro electromagnético (incluyendo ultravioleta e infrarrojo), estos sistemas permiten analizar materiales y estructuras más allá del rango visible. Esta capacidad es especialmente útil para detectar composiciones, contaminantes o defectos internos.

Principales aplicaciones:

- Evaluación de la calidad y composición de productos.
- Detección temprana de plagas o deterioro en industrias alimentarias y agrícolas.
- Inspección de capas superficiales y subsuperficiales de materiales.

• Visión por computadora

Este enfoque se basa en algoritmos avanzados y modelos matemáticos para procesar e interpretar imágenes digitales. Utiliza técnicas de aprendizaje automático, inteligencia artificial y análisis de patrones para realizar tareas automatizadas con alta precisión. Es una de las ramas más potentes de la visión artificial moderna.

Principales aplicaciones:

- Reconocimiento de objetos y patrones complejos.
- Seguimiento de movimientos y análisis conductual.
- Inspección predictiva basada en el análisis de imágenes históricas.

Cada una de estas tecnologías responde a necesidades específicas de la industria, y su elección dependerá de múltiples factores: nivel de precisión requerido, velocidad de producción, condiciones del entorno, presupuesto y capacidad de integración con otros sistemas. La combinación de varios de estos sistemas en una solución integral es una tendencia en auge, en línea con las exigencias de la industria 4.0 y la automatización inteligente.

4.3. Comunicación industrial

4.3.1. Introducción a la comunicación industrial

La comunicación industrial constituye un componente esencial dentro de los sistemas de automatización y control de procesos. En un entorno donde la eficiencia, la seguridad y la precisión son determinantes, la correcta interconexión entre dispositivos, sensores, actuadores, controladores y sistemas de supervisión es crucial para garantizar el correcto funcionamiento de las plantas industriales. A través de la comunicación industrial, es posible intercambiar información entre los distintos niveles de la estructura de automatización, desde el nivel de campo hasta el nivel de gestión (Moreno & Hernández, 2021).

Con el auge de la Industria 4.0, la conectividad ha adquirido un papel aún más relevante. Hoy en día, no solo se exige velocidad y fiabilidad en la transmisión de datos, sino también interoperabilidad, escalabilidad y capacidad de integración con tecnologías emergentes como el Internet de las Cosas (IoT), el análisis de datos en tiempo real y la inteligencia artificial. Este nuevo paradigma requiere redes de comunicación más robustas, seguras y adaptables.

El presente resumen tiene como objetivo ofrecer una visión detallada de la comunicación industrial, abordando sus fundamentos, estructuras y estándares más relevantes. Para ello, se analizarán las principales redes de comunicación industrial, se describirá la pirámide CIM, se examinarán las topologías de red utilizadas en entornos industriales y se estudiarán los protocolos de comunicación más destacados. Este enfoque proporcionará una base sólida para comprender cómo fluye la información dentro de los sistemas industriales modernos y qué tecnologías lo hacen posible.

4.3.2. Pirámide CIM

La pirámide CIM (Computer Integrated Manufacturing) es una representación jerárquica de los niveles funcionales de una planta industrial. Permite comprender cómo se interrelacionan los diferentes sistemas y cómo fluye la información entre ellos (Groover, 2016). Está compuesta por cinco niveles, desde la base (nivel físico) hasta la cúspide (nivel empresarial).

Niveles de la pirámide CIM

- Nivel 0: Nivel físico (Sensores y actuadores)

En este nivel se encuentran los dispositivos que interactúan directamente con los procesos físicos: sensores de temperatura, presión, presencia, motores, válvulas, etc. La comunicación se basa en señales analógicas o digitales.

- Nivel 1: Control

Este nivel está compuesto por PLCs, controladores distribuidos (DCS) o CNCs. Se encarga de ejecutar las rutinas de control en tiempo real, basándose en las señales recibidas del nivel 0.

- Nivel 2: Supervisión

Incluye sistemas SCADA o HMI, que permiten la monitorización y supervisión de los procesos industriales. Se recogen datos en tiempo real para visualizar alarmas, tendencias y estados del sistema.

- Nivel 3: Gestión de la producción (MES)

Aquí se encuentran los sistemas MES (Manufacturing Execution Systems), que enlazan el mundo de la automatización con la gestión empresarial. Su función es coordinar órdenes de producción, seguimiento de lotes, trazabilidad, etc.

- Nivel 4: Nivel empresarial (ERP)

Representa la capa más alta de la pirámide, donde se planifican y gestionan los recursos empresariales: compras, ventas, logística, finanzas, etc. Los sistemas ERP interactúan con el nivel 3 para tomar decisiones informadas basadas en los datos del proceso.

Conexión entre niveles

La pirámide CIM no solo organiza la información jerárquicamente, sino que también establece las necesidades de comunicación entre niveles. Mientras que los niveles bajos requieren transmisión en tiempo real (determinismo), los niveles superiores priorizan la integridad y disponibilidad de los datos. Este modelo sigue siendo vigente y sirve como base para entender cómo integrar tecnologías emergentes en la industria.

Figura 9. Ejemplo grafico de la pirámide CIM

4.3.3. Topologías de red industrial

La **topología de red** define la forma en la que están interconectados los dispositivos en una red de comunicación. En el ámbito industrial, la elección de la topología es un factor crítico, ya que influye directamente en la fiabilidad, el mantenimiento, la escalabilidad y el rendimiento de la red (Petruzzi & Antonelli, 2022).

Las topologías más comunes son:

- **Topología en línea (bus)**

Todos los dispositivos están conectados en serie a lo largo de un mismo cable de comunicación. Es común en buses de campo como Profibus o Modbus RTU.

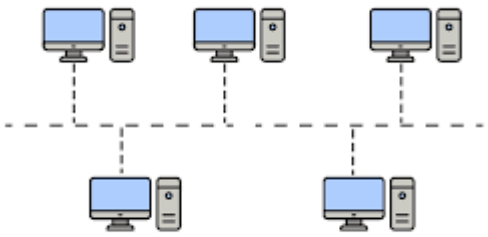

Figura 10. Ejemplo de topología en línea

Ventajas:

- Simple y económica.
- Fácil de extender.

Desventajas:

- Un fallo en el bus afecta a toda la red.
- Limitación en longitud y número de nodos.

• **Topología en estrella**

Cada dispositivo se conecta a un nodo central, como un switch o concentrador. Es la estructura típica en redes Ethernet.

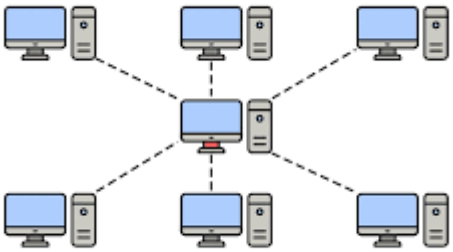

Figura 11. Ejemplo de topología en estrella

Ventajas:

- Fácil mantenimiento.
- Aislamiento de fallos.

Desventajas:

- Dependencia del nodo central.
- Mayor uso de cableado.

• **Topología en anillo**

Los dispositivos se conectan formando un lazo cerrado. Muy utilizada en redes que requieren alta disponibilidad, como Ethernet/IP o Profinet con redundancia.

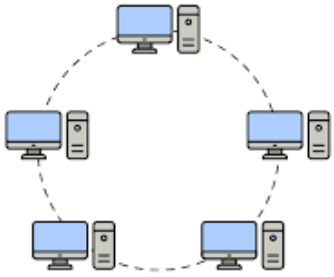

Figura 12. Ejemplo de topología en anillo

Ventajas:

- Alta fiabilidad mediante redundancia.
- Soporta mecanismos de recuperación ante fallos.

Desventajas:

- Mayor complejidad de configuración.
- Coste más elevado.

• Topología en árbol

Combina características de estrella y bus. Es adecuada para instalaciones grandes y jerarquizadas.

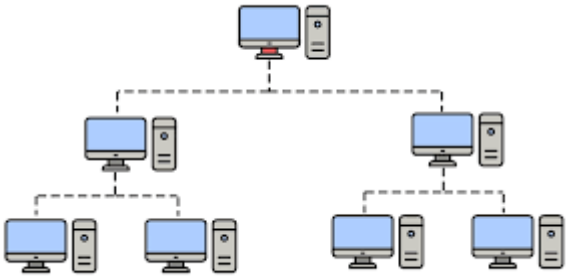

Figura 13. Ejemplo de topología en árbol

• Topología en malla

Cada nodo está conectado con varios otros, lo que proporciona redundancia total. Usada en redes críticas o inalámbricas.

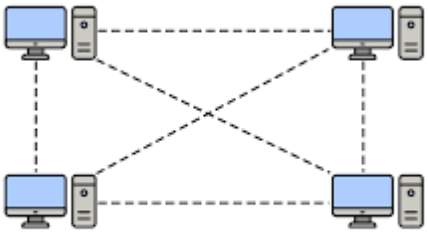

Figura 14. Ejemplo de topología en malla

Ventajas:

- Alta tolerancia a fallos.
- Rutas múltiples disponibles.

Desventajas:

- Complejidad en el diseño.
- Mayor coste de implementación.

4.3.4. Protocolos de comunicación industrial

Los protocolos industriales son conjuntos de reglas que definen cómo se realiza la comunicación entre dispositivos. Existen múltiples protocolos, cada uno con características específicas, dependiendo del tipo de red, fabricante y aplicación. Estos son los diferentes tipos que hay:

- **Protocolos abiertos**

Están estandarizados y son independientes del fabricante, lo que permite interoperabilidad entre dispositivos. Ejemplos: Modbus, CANopen, Profinet, EtherCAT.

- **Protocolos propietarios**

Son desarrollados por fabricantes específicos, como S7 (Siemens) o MELSEC (Mitsubishi), y están optimizados para sus productos, aunque limitan la compatibilidad.

- **Protocolos en tiempo real**

Diseñados para aplicaciones que requieren comunicación determinista, como Profinet RT, EtherCAT o SERCOS III.

Principales protocolos industriales

- **Modbus**

Uno de los protocolos más antiguos y extendidos. Existen dos versiones principales:

- Modbus RTU: comunicación serial (RS-485), modo maestro/esclavo.
- Modbus TCP/IP: versión sobre Ethernet, más rápida y moderna.

- **Profibus y Profinet**

Desarrollados por Siemens. Profibus usa RS-485, mientras que Profinet se basa en Ethernet.

- Profibus-DP: comunicación rápida con dispositivos de campo.
- Profinet RT: permite aplicaciones de tiempo real.
- Profinet IRT: para aplicaciones isócronas.

- CANopen

Basado en el protocolo CAN, utilizado principalmente en robótica y automoción. Ofrece alta fiabilidad, gestión de errores y bajo coste.

- EtherNet/IP

Versión industrial del protocolo Ethernet. Utiliza el protocolo CIP (Common Industrial Protocol) para comunicación entre dispositivos de diferentes niveles.

- EtherCAT

Protocolado por Beckhoff, destaca por su alta velocidad y determinismo. Los datos se procesan “en vuelo”, sin necesidad de almacenamiento intermedio.

- OPC UA

Protocolo orientado a objetos para la interoperabilidad entre sistemas industriales y de gestión. Es clave en entornos de Industria 4.0 y digitalización.

5. Herramientas utilizadas

5.1. Software ABB

El presente Trabajo de Fin de Grado se centra en el desarrollo de tareas específicas a través del uso de robots industriales, para lo cual es imprescindible contar con un entorno que permita su programación y simulación de manera precisa y eficaz.

En este contexto, ABB Robotics representa una referencia destacada dentro del sector de la automatización industrial, no solo por la fiabilidad y adaptabilidad de sus robots, sino también por su conjunto de herramientas orientadas a la simulación y control de sistemas robóticos. Entre estas herramientas se encuentran RobotWare, utilizado para la gestión y ejecución de movimientos por parte del controlador, y RobotStudio, destinado a la simulación integral del entorno de trabajo. Ambas plataformas resultan fundamentales para el diseño, validación y puesta en marcha de procesos automatizados en entornos virtuales (ABB, 2023a).

En el desarrollo de este proyecto se emplearán la versión 2023 de RobotStudio y la versión 6.15.03 de RobotWare, garantizando así compatibilidad con los estándares actuales de ABB. La elección de esta empresa y su ecosistema de software se basa en su reconocido prestigio internacional y en su consolidada trayectoria como líder en soluciones de robótica industrial, lo cual respalda su idoneidad para el desarrollo de las aplicaciones requeridas en esta propuesta.

5.1.1. Robotstudio

RobotStudio es el entorno de simulación y programación offline desarrollado por ABB para sus robots industriales, y constituye una de las herramientas más potentes y versátiles dentro del ecosistema de automatización ofrecido por esta compañía. Gracias a RobotStudio, es posible diseñar, programar, simular y optimizar celdas robotizadas completas desde un ordenador, sin necesidad de tener el robot físicamente presente. Esta capacidad de trabajo en entorno virtual permite ahorrar tiempo en la puesta en marcha, reducir riesgos en la implementación, optimizar el rendimiento del sistema y facilitar la formación y el mantenimiento de los operadores. La herramienta se ha convertido en un estándar de facto en la programación de robots ABB y juega un papel fundamental dentro del concepto de ingeniería digital en el contexto de la Industria 4.0 (ABB, 2023a).

El principio fundamental sobre el que se basa RobotStudio es la programación offline, que consiste en crear programas y simulaciones del robot fuera del entorno de producción. Esto permite realizar pruebas, ajustes y validaciones

sin detener la línea de fabricación ni ocupar el robot real, lo que se traduce en una considerable mejora de la productividad. RobotStudio logra esta capacidad gracias al uso del Virtual Controller, una réplica virtual exacta del controlador IRC5 que gobierna a los robots ABB. Esta réplica funciona como si fuera un controlador físico, ejecutando el mismo código RAPID, interpretando las mismas señales de entrada y salida, y simulando con precisión el comportamiento del robot en tiempo real. De esta manera, cualquier programa creado y probado en RobotStudio puede ser transferido directamente al robot físico con la certeza de que funcionará de forma idéntica (ABB, 2022).

RobotStudio permite trabajar en tres dimensiones mediante un entorno gráfico intuitivo en el que se puede diseñar y construir una celda de trabajo completa. Esto incluye no solo el robot, sino también el utillaje, los sensores, los sistemas de visión, las mesas de trabajo, los transportadores, y cualquier otro componente que forme parte del proceso industrial. El usuario puede importar modelos CAD en formatos estándar como STEP o IGES, definir zonas de trabajo, establecer relaciones cinemáticas entre los elementos y simular la interacción del robot con su entorno. Además, es posible detectar colisiones, validar trayectorias, calcular tiempos de ciclo y realizar análisis de alcance y accesibilidad. Todo esto permite anticipar posibles problemas antes de que se presenten en la instalación real, reduciendo costes y acelerando los tiempos de puesta en marcha (ABB, 2023b).

Una de las funcionalidades más destacadas de RobotStudio es su compatibilidad total con el lenguaje de programación RAPID, el lenguaje nativo de los robots ABB. Desde la propia interfaz de RobotStudio, el usuario puede escribir, editar y depurar código RAPID, asignarlo a tareas específicas, ejecutarlo paso a paso o en modo continuo, y observar cómo se comporta el robot en el entorno simulado. Asimismo, es posible monitorizar variables, definir rutinas, incluir interrupciones, llamadas a funciones o manejar estructuras de datos complejas. Todo esto en sincronización con el Virtual Controller, lo que garantiza que el comportamiento del código en la simulación será idéntico al del robot físico.

Además de la programación RAPID, RobotStudio permite trabajar con otras funcionalidades avanzadas, como la gestión de trayectorias complejas (mediante curvas 3D o puntos de control), la creación de sistemas de coordenadas personalizados, la integración con sistemas de visión artificial, y la programación de múltiples robots en la misma celda. Esta última función, conocida como Multimove, permite coordinar el trabajo de varios robots que comparten el mismo espacio, garantizando que no se interfieran entre sí y optimizando las secuencias de operación para maximizar el rendimiento global. También se puede integrar con sensores externos o sistemas PLC mediante protocolos de comunicación como OPC, Profinet o Ethernet/IP, lo que convierte

a RobotStudio en una plataforma ideal para diseñar sistemas de automatización complejos.

Otra funcionalidad clave de RobotStudio es la posibilidad de realizar simulaciones en tiempo real y generar animaciones. Esto permite observar cómo se comportará el robot durante un ciclo de trabajo completo, analizar el movimiento desde distintos ángulos, y documentar el proceso para presentar informes o justificar decisiones de diseño. Estas simulaciones también son útiles en la formación de operadores y programadores, ya que permiten experimentar con el entorno del robot sin riesgo de dañar equipos ni comprometer la seguridad de las personas. De hecho, ABB ofrece licencias educativas de RobotStudio que son ampliamente utilizadas en centros de formación técnica, universidades y programas de ingeniería.

RobotStudio también ofrece módulos adicionales que extienden sus capacidades, entre ellos destacan RobotStudio Machining PowerPac, orientado a tareas de mecanizado y fresado con robot; ArcWelding PowerPac, para aplicaciones de soldadura por arco; PickMaster, para tareas de picking y colocación en entornos de alta velocidad; y VisionStudio, que permite integrar y programar sistemas de visión en el entorno de simulación. Estos módulos están pensados para facilitar el desarrollo de aplicaciones específicas, proporcionando bibliotecas, asistentes y plantillas que reducen significativamente el tiempo de desarrollo. Gracias a estos complementos, RobotStudio no solo sirve para la programación de robots, sino que se convierte en una plataforma integral de ingeniería de automatización.

En cuanto a la conectividad, RobotStudio permite importar y exportar fácilmente datos y proyectos, tanto al controlador físico IRC5 como a sistemas externos. La sincronización entre el entorno virtual y el robot real puede realizarse mediante conexión por Ethernet, USB o a través de redes industriales. Además, se pueden cargar y descargar programas RAPID, backups del sistema, configuraciones de herramientas, bases de datos de usuarios, y más. Esta flexibilidad facilita la migración de proyectos, el mantenimiento preventivo y la actualización de sistemas existentes sin interrumpir el flujo de producción.

En el ámbito de la Industria 4.0, RobotStudio representa una herramienta fundamental para la ingeniería digital, ya que permite simular y optimizar procesos antes de su implementación física, conectarse con otras herramientas de gestión y producción, y generar datos que pueden ser utilizados en análisis predictivos o de mejora continua. Por ejemplo, es posible integrar RobotStudio con plataformas de simulación de plantas completas, sistemas MES, ERP o gemelos digitales, proporcionando una visión completa del rendimiento del sistema antes de realizar inversiones físicas. Asimismo, se puede utilizar en entornos colaborativos donde ingenieros de distintas áreas

trabajan de manera simultánea en el diseño de la celda robotizada, desde diferentes ubicaciones geográficas.

Una ventaja muy significativa de RobotStudio es su capacidad de reducir el tiempo de inactividad de los robots. En lugar de detener una línea de producción para probar un nuevo programa, este se puede desarrollar y validar en RobotStudio, y una vez verificado, transferirse al robot real en cuestión de minutos. Este enfoque mejora la eficiencia, reduce errores humanos y contribuye a mantener altos niveles de calidad en los productos manufacturados. También permite realizar mantenimiento virtual, estudiar alternativas de layout, rediseñar procesos productivos y responder con mayor agilidad a los cambios en la demanda del mercado.

No obstante, el uso de RobotStudio también presenta ciertos desafíos. Su correcta utilización requiere formación específica, tanto en el uso del software como en el lenguaje RAPID. Aunque la interfaz es bastante intuitiva, el dominio de sus herramientas más avanzadas exige experiencia y conocimientos técnicos en automatización, programación y robótica. Asimismo, para realizar simulaciones con alta fidelidad, es necesario contar con modelos CAD detallados y una descripción precisa del entorno físico, lo que puede requerir un esfuerzo adicional en la etapa de diseño. Otro aspecto a tener en cuenta es la compatibilidad entre versiones, ya que algunas funciones pueden variar dependiendo del modelo de robot o la versión del software RobotWare instalada en el controlador IRC5.

En resumen, RobotStudio es una herramienta imprescindible para el desarrollo de soluciones robóticas eficientes, seguras y adaptadas a los requerimientos actuales de la industria moderna. Su enfoque basado en simulación, su integración con el controlador IRC5 mediante el Virtual Controller, y su compatibilidad con el lenguaje RAPID, lo convierten en el entorno ideal para la programación, verificación y optimización de robots ABB. Ya sea para diseñar nuevas celdas robotizadas, mejorar procesos existentes, formar personal técnico, o experimentar con nuevas tecnologías como visión artificial o robótica colaborativa, RobotStudio ofrece un entorno potente, versátil y orientado al futuro. Su uso adecuado permite mejorar la productividad, reducir costes de integración, minimizar errores y asegurar una transición fluida hacia la automatización avanzada. Por todo ello, RobotStudio representa una pieza clave dentro del ecosistema digital de ABB y una herramienta estratégica para cualquier ingeniero que desee desarrollar soluciones robóticas en entornos industriales exigentes.

Al iniciar RobotStudio, la interfaz de usuario se presenta de la siguiente manera:

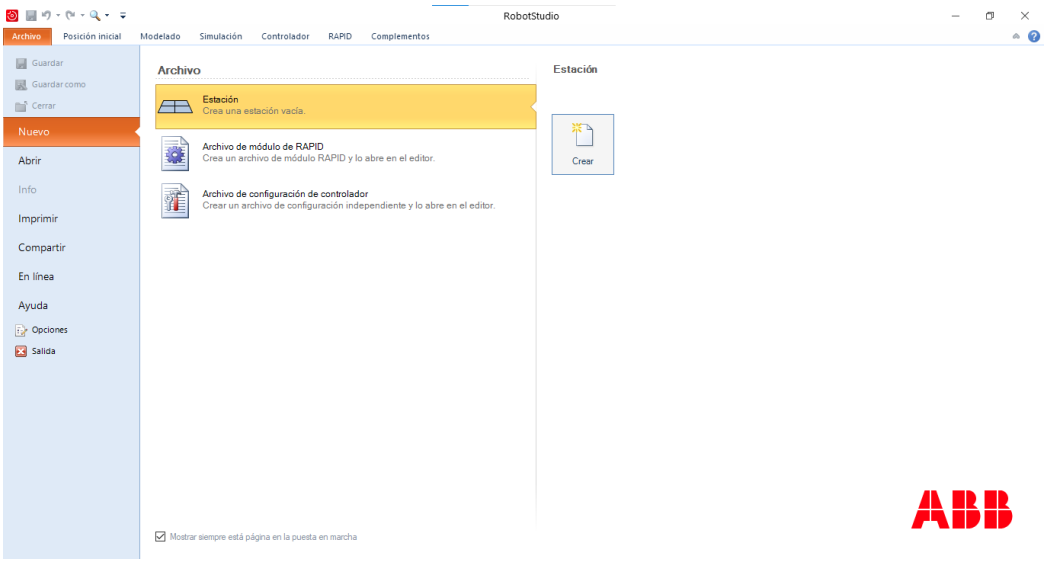

Figura 15. Interfaz al iniciar Robotstudio 2023

Pero el entorno de trabajo es el siguiente:

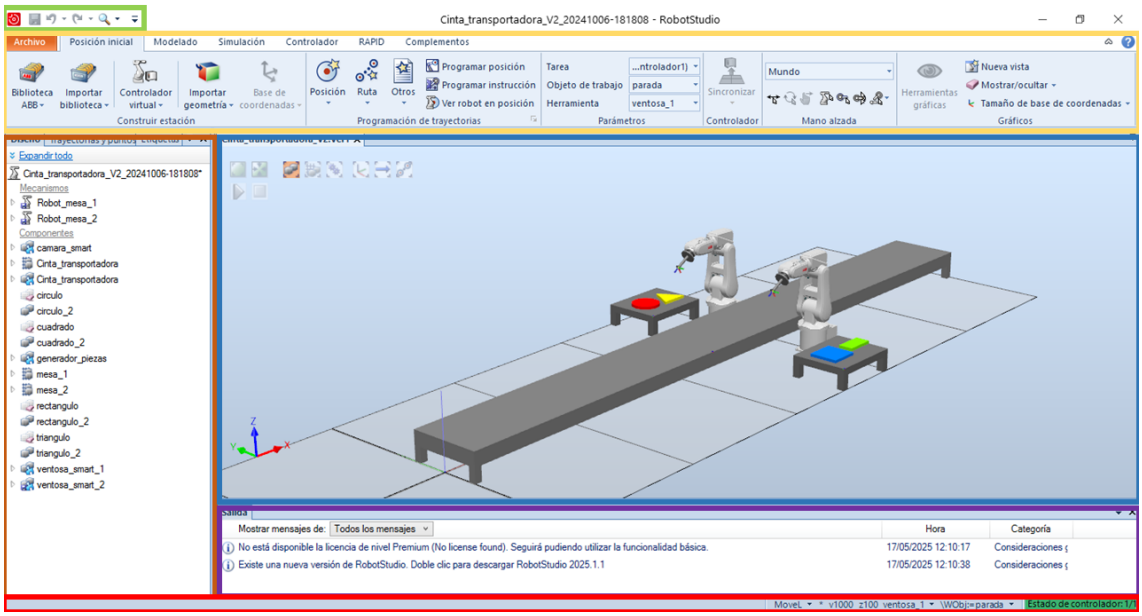

Figura 16. Entorno de trabajo RobotStudio 2023

Aquí podemos encontrar diferentes áreas:

- Barra de herramientas: Aquí tenemos las funciones básicas como guardar, cerrar o búsqueda rápida.
- Barra de pestañas: aquí están organizadas las diferentes acciones que proporciona RobotStudio organizadas por pestaña:
    - Archivo: incluye funciones para configurar una nueva estación, crear un sistema robótico, conectar con un controlador, guardar la estación en diversos formatos, etc.

- Posicion inicial: Aquí se gestionan el diseño de la estación, la posición de los componentes y la generación de trayectorias.

Figura 17. Pestaña posición inical en RobotStudio 2023

- Modelado: Tiene las funciones necesarias para crear y organizar componentes.

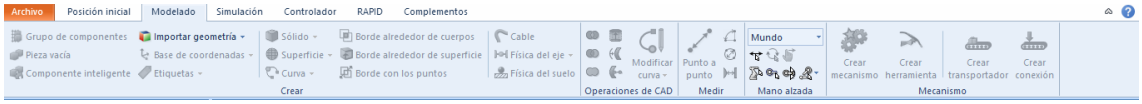

Figura 18. Pestaña Modelado en RobotStudio 2023

- Simulación: En esta pestaña se gestionan todas las funciones necesarias para llevar a cabo la simulación.

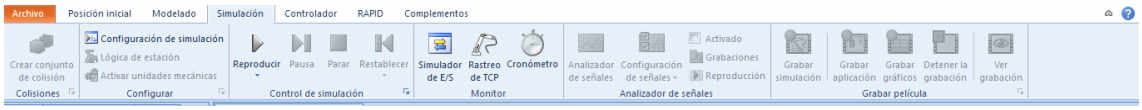

Figura 19 Pestaña Simulación en RobotStudio 2023.

- Controlador: incorpora acciones para la sincronización, configuración y asignación de tareas al controlador virtual, así como para manejar controladores reales.

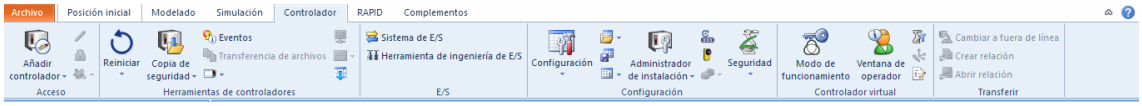

Figura 20. Pestaña Controlador en RobotStudio 2023

- Rapid: Funciones propias del editor de código Rapid.

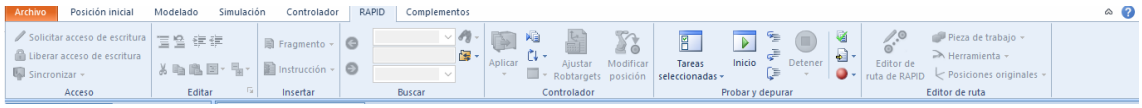

Figura 21. Pestaña Rapid en RobotStudio 2023

- Complementos: módulos y extensiones adicionales de RobotStudio.

- Ventana de navegación: En este área se muestra una vista jerarquica de los diferentes elementos asociados específicamente a cada pestaña de la barra de pestañas.

- Ventana de la estación: Muestra la visualización de la estación sobre la que se esa trabajando, su diseño y los movimientos de los componentes y los robots.

- Ventana de Salida: Muestra los eventos producidos en la estación, tales como avisos errores, inicio y fin de la simulación, etc.

- Ventana de Estado: muestra información en tiempo real sobre la estación, tales como el estado del controlador, los valores del sistema de coordenadas, etc.

5.1.2. Rapid

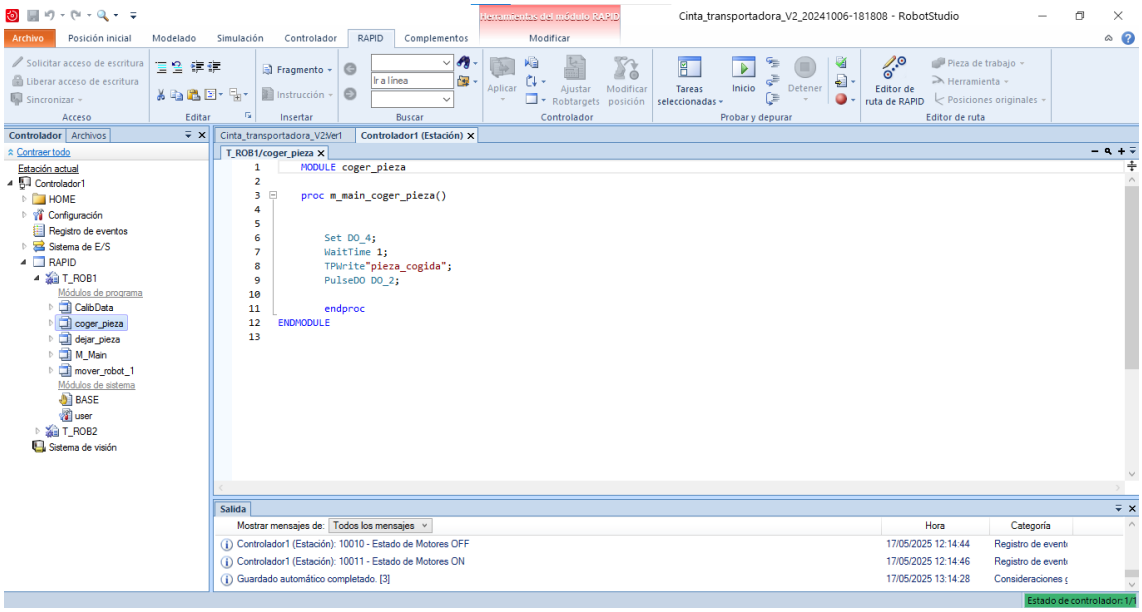

Figura 22. Pantalla de edición de código de Rapid en RobotStudio 2023

RAPID es el lenguaje de programación desarrollado por ABB para sus robots industriales. Se trata de un lenguaje estructurado, de alto nivel, diseñado específicamente para facilitar la programación, control y coordinación de los movimientos y tareas de los robots en entornos industriales. Su sintaxis, clara y cercana a lenguajes como Pascal o C, permite a los usuarios definir rutinas, configurar herramientas, interactuar con sensores y ejecutar tareas automatizadas con gran precisión (ABB, 2023a).

Una de las principales ventajas de RAPID es su integración nativa con el ecosistema de ABB, particularmente con el controlador IRC5 y con el entorno de simulación y programación RobotStudio. Esto permite desarrollar, depurar y probar programas de manera offline antes de trasladarlos al entorno real, reduciendo tiempos de implementación y evitando posibles errores de ejecución (ABB, 2023b).

El lenguaje RAPID está organizado en módulos que contienen procedimientos, funciones, interrupciones, rutinas y variables. Un programa típico se compone de varios módulos, cada uno orientado a una parte específica del proceso: definición de herramientas, posiciones (targets), rutinas de movimiento, control de señales, entre otros.

RAPID incluye una variedad de tipos de datos, tanto primitivos como estructurados. Entre los más utilizados se encuentran:

- num: Números reales para realizar cálculos y establecer velocidades.
- bool: Valores lógicos que permiten tomar decisiones condicionales.
- string: Cadenas de texto para gestionar información escrita.
- robtarget: Coordenadas y orientaciones espaciales que definen las posiciones del robot.
- tooldata y wobjdata: Configuraciones de herramientas y objetos de trabajo necesarios para tareas de precisión.

Además de movimientos, RAPID permite gestionar entradas y salidas digitales para interactuar con el entorno. Esto resulta esencial para tareas como el manejo de ventosas, actuadores o la lectura de sensores. También es posible definir condiciones de espera, sincronización entre robots y respuestas a eventos mediante interrupciones o bucles de control.

5.1.3. ABB IRC5 OPC

En el contexto de la automatización industrial moderna, la necesidad de integrar distintos sistemas y dispositivos dentro de un entorno de producción ha llevado al desarrollo de estándares de comunicación capaces de proporcionar interoperabilidad, flexibilidad y eficiencia. Uno de estos estándares ampliamente utilizados es OPC (OLE for Process Control), que facilita el intercambio de datos entre dispositivos de control, como controladores lógicos programables (PLCs) y controladores de robots, con aplicaciones de software externas, como sistemas SCADA, MES o herramientas de análisis (OPC Foundation, 2022).

Dentro de este ecosistema de automatización, destaca el sistema de control IRC5 de ABB, que permite operar y gestionar los robots industriales de esta compañía. ABB, como uno de los principales fabricantes de soluciones de automatización a nivel mundial, ha integrado la tecnología OPC en sus controladores IRC5, permitiendo que datos esenciales del robot sean accesibles de manera segura y en tiempo real desde sistemas externos. Esta combinación se conoce como ABB IRC5 OPC, y representa una herramienta poderosa para supervisar, controlar e integrar robots ABB en procesos industriales complejos (ABB, 2022).

El controlador IRC5 es la quinta generación de sistemas de control desarrollados por ABB para gestionar sus robots industriales. Este controlador fue introducido en el mercado en 2004 y ha sido el estándar desde entonces para todos los modelos de robots ABB, gracias a su capacidad de controlar múltiples ejes con alta precisión, su arquitectura robusta y flexible, y su compatibilidad con distintos lenguajes y protocolos de comunicación. El IRC5 está diseñado para ofrecer un control avanzado de movimiento, facilidad de integración con otros dispositivos industriales y una experiencia de usuario eficiente a través del lenguaje de programación RAPID y la interfaz gráfica

RobotStudio. Gracias a estas capacidades, el IRC5 se adapta a aplicaciones en sectores tan diversos como la automoción, la electrónica, la alimentación, la metalurgia y la logística.

Por su parte, OPC es un conjunto de especificaciones de comunicación que permite el intercambio de datos entre dispositivos industriales y aplicaciones informáticas de forma estandarizada. Inicialmente desarrollado en los años noventa por la OPC Foundation, OPC surgió como una solución para los problemas de comunicación entre sistemas de distintos fabricantes. Las primeras versiones, conocidas como OPC Classic, se basaban en tecnologías Microsoft como COM/DCOM, lo que limitaba su uso a entornos Windows y presentaba desafíos de seguridad. Con el tiempo, se desarrolló OPC UA (Unified Architecture), una evolución más moderna, segura e independiente de plataforma, que amplió el alcance de esta tecnología. En esencia, OPC actúa como un puente entre los datos generados por un dispositivo industrial y las aplicaciones que los necesitan, sin importar el fabricante ni el sistema operativo. El servidor OPC es el componente que se comunica directamente con el hardware (como el controlador IRC5) y expone la información al exterior mediante una interfaz estándar, mientras que los clientes OPC son aplicaciones que consumen estos datos para visualización, control o análisis.

En el caso del ABB IRC5, el soporte para OPC se implementa a través de un servidor OPC que puede ejecutarse en un ordenador externo conectado al controlador, o en ciertos casos, directamente dentro del entorno de software proporcionado por ABB, como RobotWare. Este servidor OPC permite acceder a diferentes tipos de datos generados por el robot y por el programa RAPID que lo gobierna. Entre los datos disponibles se incluyen las posiciones del efector final, los estados de las entradas y salidas digitales, las variables internas del programa, el estado del sistema (si está operativo, en error, o en modo automático), así como alarmas, eventos y otros parámetros críticos para el diagnóstico y la supervisión de la célula robotizada. De este modo, una aplicación cliente OPC puede conectarse al servidor y obtener en tiempo real todos estos datos para ser visualizados en una interfaz HMI, ser almacenados en una base de datos para análisis posterior o incluso ser utilizados para tomar decisiones automáticas dentro de una arquitectura de control descentralizado.

La arquitectura típica de un sistema ABB IRC5 OPC incluye el controlador IRC5, un ordenador con el servidor OPC instalado y una o varias aplicaciones cliente OPC. La comunicación entre el IRC5 y el servidor se realiza a través de Ethernet, lo que permite una conexión rápida y eficiente. Por su parte, los clientes OPC pueden ser sistemas SCADA, plataformas MES, aplicaciones de control de calidad, software de mantenimiento predictivo, u otros componentes del sistema de gestión industrial. Esta arquitectura modular facilita la escalabilidad y flexibilidad del sistema, permitiendo que múltiples clientes accedan simultáneamente a los datos del robot, sin afectar su funcionamiento. Además,

ABB proporciona herramientas para configurar qué variables RAPID serán accesibles desde el servidor OPC, permitiendo un control detallado sobre qué información se comparte y cómo se estructura.

Una de las principales ventajas de utilizar ABB IRC5 OPC es la interoperabilidad con otros sistemas. Al tratarse de un estándar ampliamente adoptado en la industria, OPC permite que el robot ABB se integre sin problemas con equipos y software de distintos fabricantes. Esto reduce la necesidad de desarrollar interfaces personalizadas, disminuye los costos de integración y mejora la flexibilidad del sistema. Otra ventaja importante es la supervisión en tiempo real. Gracias a OPC, es posible monitorear el comportamiento del robot desde una sala de control, detectar anomalías, registrar ciclos de operación, y generar alarmas ante fallos o desviaciones en el proceso. Esta capacidad es especialmente útil en entornos donde la continuidad operativa es crítica, como en la industria automotriz o farmacéutica.

Asimismo, el uso de ABB IRC5 OPC permite registrar datos operativos que luego pueden ser analizados con técnicas de inteligencia artificial o análisis estadístico. Esta información es valiosa para la mejora continua, la planificación de mantenimiento, la trazabilidad de productos y la optimización de procesos. También se facilita la integración de los robots con sistemas de control superiores como ERP o plataformas de Industria 4.0, lo que habilita una planta verdaderamente conectada y automatizada. Por ejemplo, se pueden ajustar parámetros de operación de los robots desde una aplicación web basada en OPC UA, o se puede sincronizar el trabajo del robot con el estado de otras máquinas dentro de la línea de producción.

A pesar de sus ventajas, la implementación de OPC en el entorno ABB IRC5 no está exenta de desafíos. Uno de ellos es la configuración inicial, que requiere conocimientos técnicos tanto del entorno RAPID como de la arquitectura OPC. Es necesario definir correctamente las variables a exponer, configurar las conexiones de red, asegurarse de que no existan bloqueos por firewalls, y en algunos casos, gestionar licencias específicas del software de ABB. También hay que prestar especial atención a la seguridad, sobre todo si se utilizan versiones OPC Classic, que no cuentan con cifrado ni autenticación robusta. En estos casos, se recomienda migrar a OPC UA, que ofrece mecanismos modernos de seguridad como certificados digitales, control de acceso por usuario y encriptación de datos. Otra consideración importante es la latencia en la comunicación. Aunque OPC permite obtener datos en tiempo real, la frecuencia de actualización depende del hardware y del volumen de información. Por eso, se deben definir cuidadosamente los ciclos de muestreo y los parámetros de calidad de servicio, para evitar sobrecargar la red o el sistema de control.

En cuanto a las aplicaciones prácticas, ABB IRC5 OPC se utiliza en múltiples sectores industriales. En la industria automotriz, se emplea para integrar robots de soldadura o ensamblaje con sistemas SCADA que supervisan la calidad del proceso en tiempo real. En la industria electrónica, permite coordinar robots de montaje con estaciones de inspección óptica, registrando los datos de producción para análisis posterior. En el sector de la alimentación, facilita el control de robots de paletizado o empaquetado, integrándolos con sistemas de trazabilidad. En entornos logísticos, se utiliza para sincronizar el trabajo de robots con transportadores, almacenes automáticos y sistemas de gestión de pedidos. En todos estos casos, el uso de OPC como capa de integración garantiza que los datos fluyan de manera consistente, segura y estandarizada.

En conclusión, ABB IRC5 OPC representa una solución integral para la integración de robots ABB en entornos de automatización avanzada. Su capacidad para comunicar el controlador IRC5 con múltiples aplicaciones mediante un estándar abierto permite lograr sistemas más eficientes, conectados y fáciles de mantener. A través del servidor OPC, los datos generados por el robot pueden ser monitorizados, analizados y utilizados para mejorar la calidad del proceso, reducir tiempos de parada y aumentar la productividad. La estandarización, la interoperabilidad, la escalabilidad y la posibilidad de integrar inteligencia artificial o análisis avanzado de datos convierten a ABB IRC5 OPC en una herramienta esencial dentro de la Industria 4.0. Sin embargo, su implementación debe realizarse cuidadosamente, prestando atención a aspectos como la seguridad, la configuración de red y la gestión de variables. Con una planificación adecuada, esta tecnología puede ofrecer grandes beneficios a cualquier sistema industrial que emplee robots ABB.

El entorno de trabajo es el siguiente:

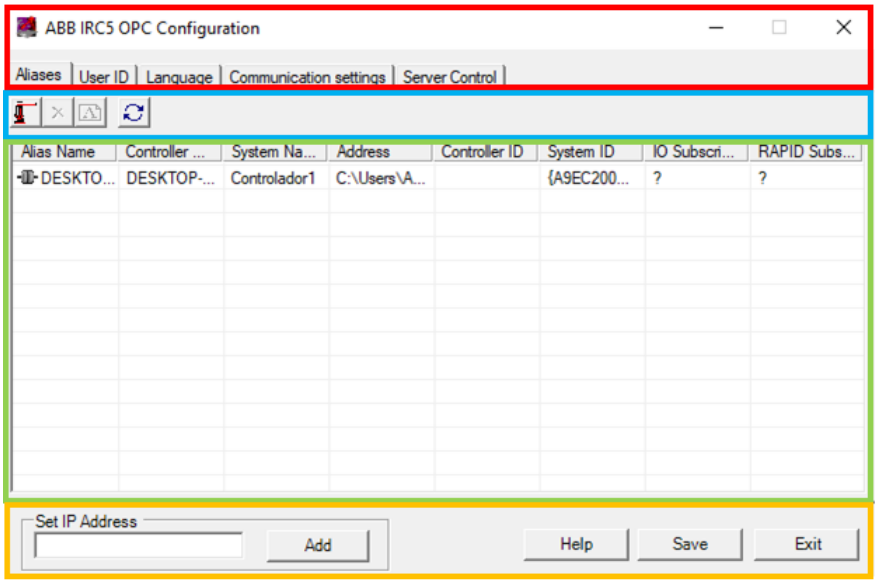

Figura 23. Entorno de trabajo de ABB IRC5 OPC

- Menu de pestañas: Contiene las pestañas principales de configuración:

  - Aliases: Donde se crean y gestionan los alias (nombres simbólicos) para los controladores conectados.
  - User ID: Configura los datos del usuario para acceder al controlador.
  - Language: Permite cambiar el idioma de la interfaz.
  - Communication settings: Define cómo se comunican los controladores con el servidor OPC.
  - Server Control: Controla el estado del servidor OPC (iniciar, detener, etc.).

- Barra de herramientas: Incluye botones de acción rápida:

  - Nuevo alias: Agrega una nueva entrada para un controlador IRC5.
  - Eliminar alias: Elimina el alias seleccionado.
  - Actualizar lista: Refresca la lista de controladores detectados o configurados.

- Lista de Alias/configuraciones: Muestra todos los controladores configurados y su información:

  - Alias Name: Nombre simbólico dado al controlador.
  - Controller Name/System Name/Address: Información del sistema y ruta al archivo de configuración.
  - Controller ID/System ID: Identificadores únicos del controlador.
  - IO Subscription / RAPID Subscription: Indica si están activadas las suscripciones a señales IO y al programa RAPID.

- Configuracion de dirección IP: Permite agregar un nuevo controlador manualmente:

  - Set IP Address: Introduces la IP del controlador IRC5.
  - Add: Añade el controlador a la lista.
  - Help / Save / Exit: Botones para obtener ayuda, guardar cambios o salir del programa.

5.2. Matlab

MATLAB es un entorno de programación de alto nivel desarrollado por MathWorks, ampliamente utilizado en entornos académicos, industriales y científicos para el cálculo numérico, análisis de datos, simulación y desarrollo de algoritmos. Su nombre proviene de *Matrix Laboratory*, y refleja su capacidad inicial para manipular matrices, una de sus estructuras de datos fundamentales. A lo largo de las décadas, MATLAB ha evolucionado hacia una plataforma multifuncional, con amplios toolboxes (bibliotecas) que extienden sus capacidades en campos como el procesamiento de señales, control automático, visión por computadora, aprendizaje automático y, especialmente, el análisis de imágenes (MathWorks, 2023a).

El análisis de imágenes digitales es una de las áreas donde MATLAB ha adquirido gran relevancia, gracias a su toolbox especializado llamado Image Processing Toolbox. Este conjunto de funciones permite a los usuarios leer, procesar, analizar, visualizar y manipular imágenes de forma sencilla, estructurada y eficiente. Debido a su potencia y versatilidad, MATLAB se ha convertido en una herramienta habitual en proyectos de visión artificial, inspección automática, reconocimiento de patrones, biometría, medicina, robótica, y automatización industrial.

El análisis de imágenes en MATLAB comienza generalmente con la adquisición o lectura de una imagen digital desde una fuente externa, como un archivo, una cámara o un sistema de visión. Esta operación se realiza mediante la función imread(), que permite cargar imágenes en formatos comunes como JPEG, PNG, BMP o TIFF. Una vez cargada, la imagen se representa internamente como una matriz de píxeles, en escala de grises o a color, dependiendo del tipo de imagen (MathWorks, 2023b).

Una de las primeras etapas del análisis es el preprocesamiento, que incluye operaciones como la conversión a escala de grises (rgb2gray()), la normalización de la intensidad, la mejora del contraste (imadjust() o histeq()), la eliminación de ruido (mediante filtros como imfilter() o medfilt2()), y el recorte o escalado de la imagen. Estas acciones preparan la imagen para ser segmentada o analizada de manera más efectiva.

Uno de los procesos más importantes en el análisis de imágenes es la segmentación, que consiste en dividir la imagen en regiones que contengan objetos o características relevantes. En MATLAB, este proceso puede realizarse mediante diversas técnicas: segmentación por umbral (imbinarize()), detección de bordes (edge() con operadores como Sobel, Canny o Prewitt), agrupamiento mediante k-means (imsegkmeans()), o incluso técnicas basadas en deep learning cuando se trata de imágenes más complejas.

Una vez segmentada la imagen, es posible identificar y etiquetar las regiones mediante bwlabel() o bwconncomp(), y luego analizar sus propiedades morfológicas con la función regionprops(). Esta última permite extraer información como el área, perímetro, orientación, forma, centroides, eje mayor y menor, circularidad, y otros parámetros que ayudan a clasificar o identificar los objetos presentes.

Este análisis es esencial en aplicaciones como el conteo de células en microscopía, la detección de defectos en productos industriales, o la clasificación de piezas en líneas de ensamblaje automatizadas. La segmentación precisa también es crítica en tareas médicas como la localización de tumores o estructuras anatómicas en imágenes de resonancia magnética o tomografías.

Una de las fortalezas más notables de MATLAB es su capacidad de visualización gráfica. El entorno permite mostrar imágenes fácilmente con imshow() y superponer sobre ellas contornos, marcas, vectores o resultados del análisis. También se pueden generar histogramas de intensidad (imhist()), gráficos de líneas, mapas de calor o gráficos tridimensionales para representar datos extraídos de las imágenes (MathWorks, 2023a).

Además del análisis visual, MATLAB permite realizar un análisis estadístico cuantitativo de las imágenes procesadas. Por ejemplo, puede calcular la media de intensidad, la varianza, la entropía (entropy()), la textura mediante matrices de co-ocurrencia (graycomatrix()), o aplicar transformadas como la de Fourier (fft2()) o de Hough (hough()) para detectar patrones más complejos como líneas o círculos (MathWorks, 2023a).

Estos métodos no solo proporcionan una descripción matemática del contenido visual, sino que también permiten alimentar sistemas de clasificación, redes neuronales, o algoritmos de aprendizaje supervisado y no supervisado desarrollados dentro de MATLAB.

Otra ventaja clave de MATLAB es su capacidad de automatizar el procesamiento de imágenes a gran escala. Mediante scripts o funciones personalizadas, el usuario puede procesar cientos o miles de imágenes de forma secuencial, extrayendo información y guardando resultados

automáticamente. Esto es especialmente útil en proyectos de inspección industrial, donde se deben analizar grandes volúmenes de datos en poco tiempo.

El entorno también permite crear interfaces gráficas de usuario (GUI) con App Designer o GUIDE, lo cual facilita la implementación de aplicaciones de análisis visual interactivo sin necesidad de conocimientos avanzados de programación.

MATLAB permite la integración con otros lenguajes como Python, C/C++, Java o Simulink. También es compatible con protocolos de comunicación como OPC, que permiten el intercambio de datos entre MATLAB y sistemas externos (como PLCs o robots industriales). Esta capacidad es especialmente relevante cuando el análisis de imágenes se utiliza como entrada para la toma de decisiones automatizadas, como en el caso de sistemas de visión artificial que determinan la posición de una pieza en una estación de trabajo robótica.

La interfaz de Matlab de usuario se muestra asi:

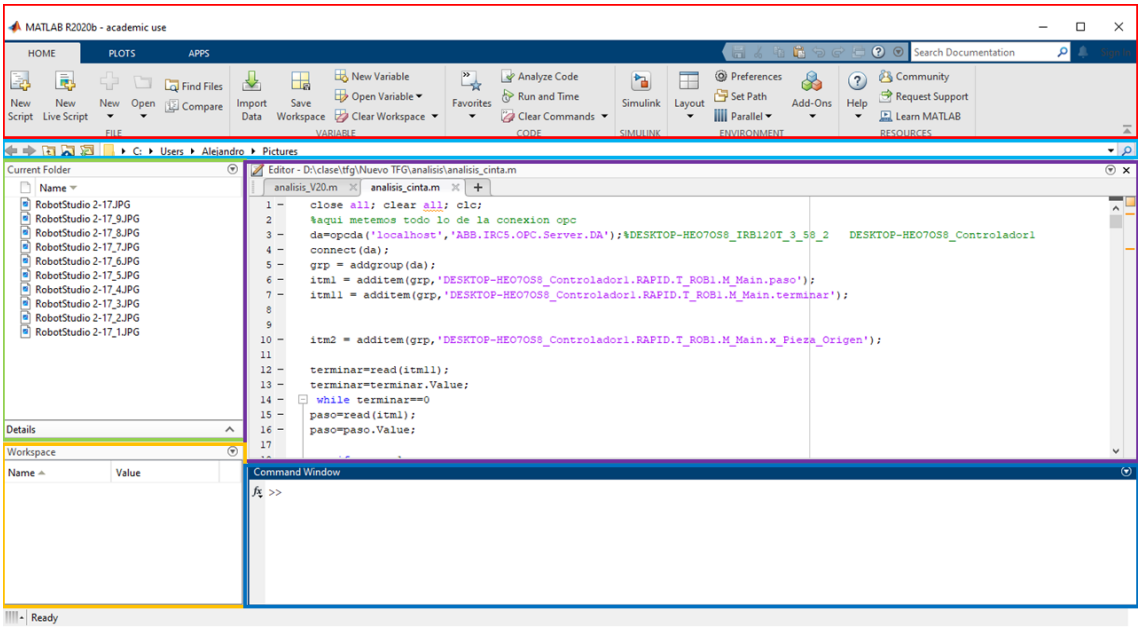

Figura 24. Interfaz de MATLAB

Se muestran diferentes ventanas:

- Cinta de herramientas: Desde aquí puedes iniciar nuevos scripts, guardar tu trabajo, ejecutar código o ajustar la configuración del entorno. Agrupa herramientas y comandos principales organizados en pestañas como:
    - **HOME:** Crear scripts, abrir archivos, importar datos, gestionar variables, analizar código, etc.

    - **PLOTS / APPS:** Crear gráficos o acceder a aplicaciones integradas de MATLAB.
    - **ENVIRONMENT**: Configurar el entorno de trabajo (por ejemplo, cambiar el directorio o ver las preferencias).
- Barra de navegación de la carpeta actual: Esta barra te muestra la ruta del directorio actual de trabajo, es decir, la carpeta en tu sistema donde MATLAB está buscando y guardando archivos de forma predeterminada. Desde aquí se puede navegar o cambiar manualmente el directorio de trabajo.
- Editor de código: Aquí es donde se programan y editan los scripts. El texto en color te ayuda a identificar comentarios, funciones y comandos.
- Current folder: Muestra los archivos disponibles en el directorio de trabajo actual. Sirve para acceder rápidamente a imágenes, scripts y datos necesarios para tus proyectos.
- Workspace: Muestra las variables activas en la memoria mientras se ejecuta el código. Aquí se pueden ver y comprobar valores de variables durante la ejecución del programa.
- Command window: Aqui se pueden ejecutar comandos de MATLAB de forma directa. Tambien se puede probar funciones rápidamente, hacer cálculos, ejecutar scripts o depurar errores.

6. Estación

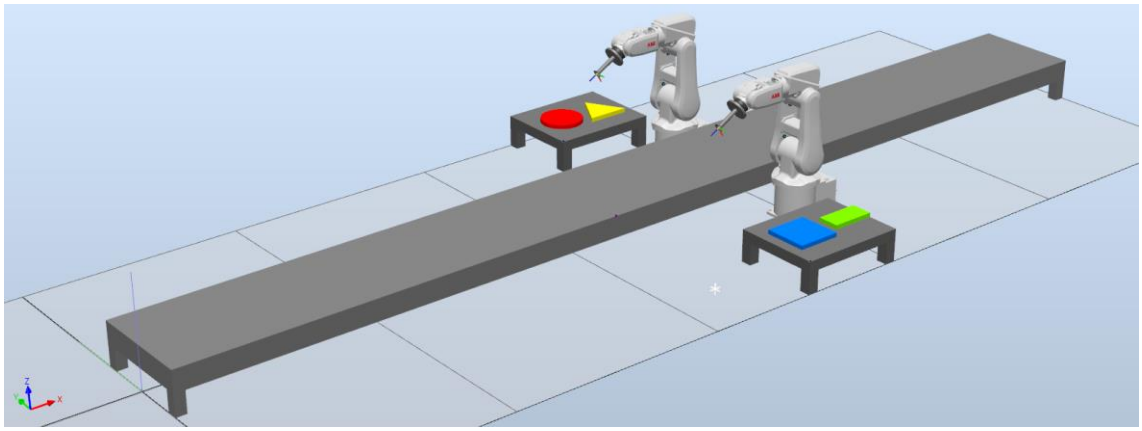

Figura 25. Estación de trabajo

Para este proyecto hemos creado una estación en la que tenemos 2 robots IRB 120T ya que son los que hemos usado en la asignatura de robótica. Estos dos robots están controlados por un solo controlador IRC5, que hace que se coordinen los dos.
También hemos utilizado una serie de objetos y componentes inteligentes para poder llevar a cabo las acciones que queremos, mediante la lógica de la estación.

6.1. Lógica de la estación

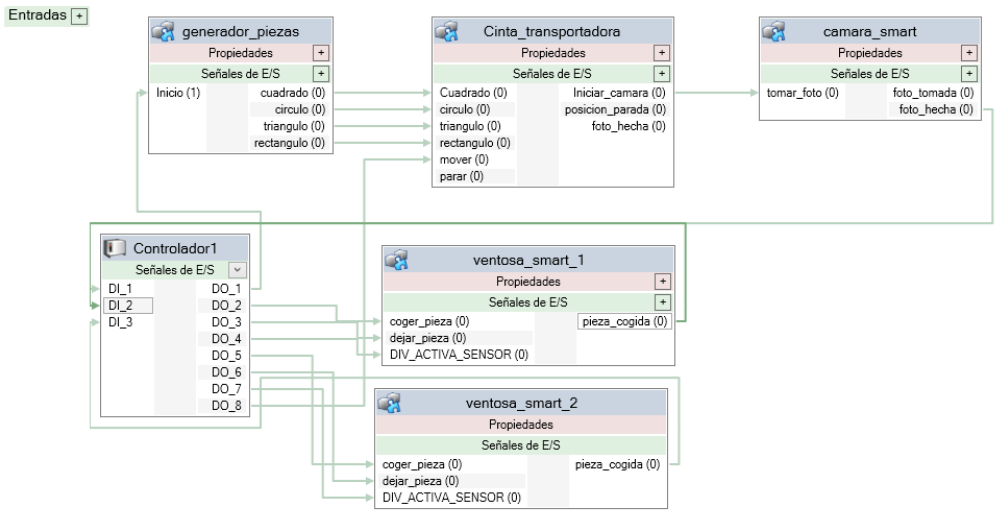

Figura 26. Lógica de la estación

Aquí vemos que el proceso tiene dos partes diferenciadas, la generación de las piezas, su transporte y la toma de la foto, y por otra parte tenemos el movimiento del robot y el uso de las herramientas.
Para la primera parte, vemos que mediante la señal DO_1 del controlador damos inicio al proceso. Esto nos permite de manera secuencial generar la

pieza, transportarla, hacer la foto, una vez hecha enviar una señal de vuelta a DI_1 indicando que ya hemos hecho la foto.
Para la segunda parte ya veremos que la mayor parte del proceso usaremos Rapid, pero para controlar las herramientas que usan los robots usaremos las señales de salida del controlador DO_2, DO_3, DO_4, DO_5, DO_6 y DO_7, y las señales de entrada DI_2 y DI_3. Estudiaremos esto más adelante.

6.2. Componentes Inteligentes

6.2.1. Generador de piezas

Para generar las piezas de forma aleatoria hemos utilizado un componente inteligente en el cual mediante un generador de números aleatorios entre 0 y 4 y una serie de comparadores generamos 4 salidas que se corresponden con el cuadrado, circulo, triángulo y rectángulo.

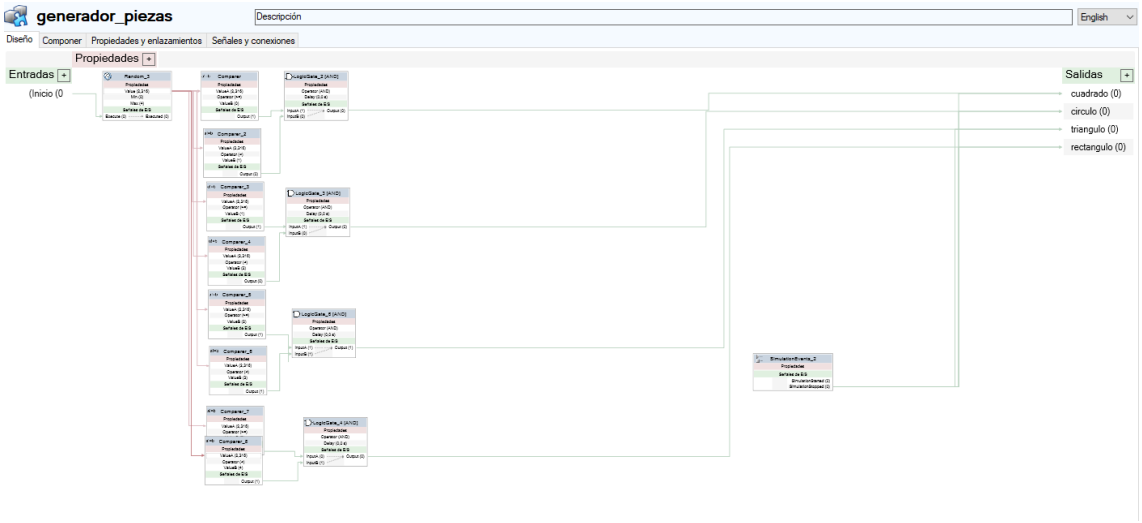

Figura 27. Lógica del generador de piezas

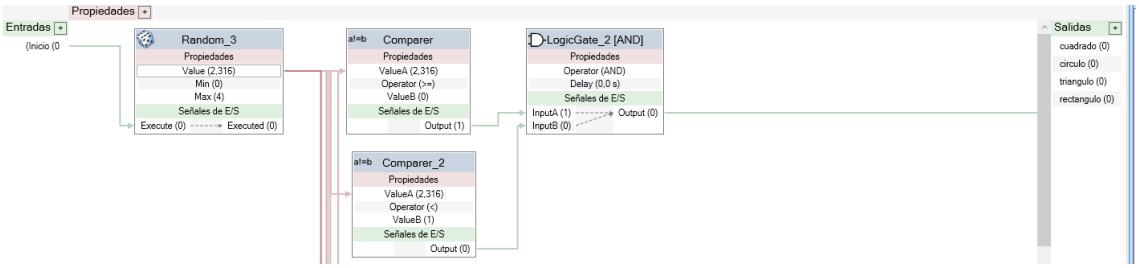

Figura 28. Detalle de la lógica del generador de piezas

6.2.2. Cinta transportadora

Ahora para generar las piezas y desplazarlas hemos creado un componente inteligente llamado cinta_transportadora, la cual es una mesa sobre la que aparecen las diferentes piezas y se transportan hasta el punto de hacer la foto.

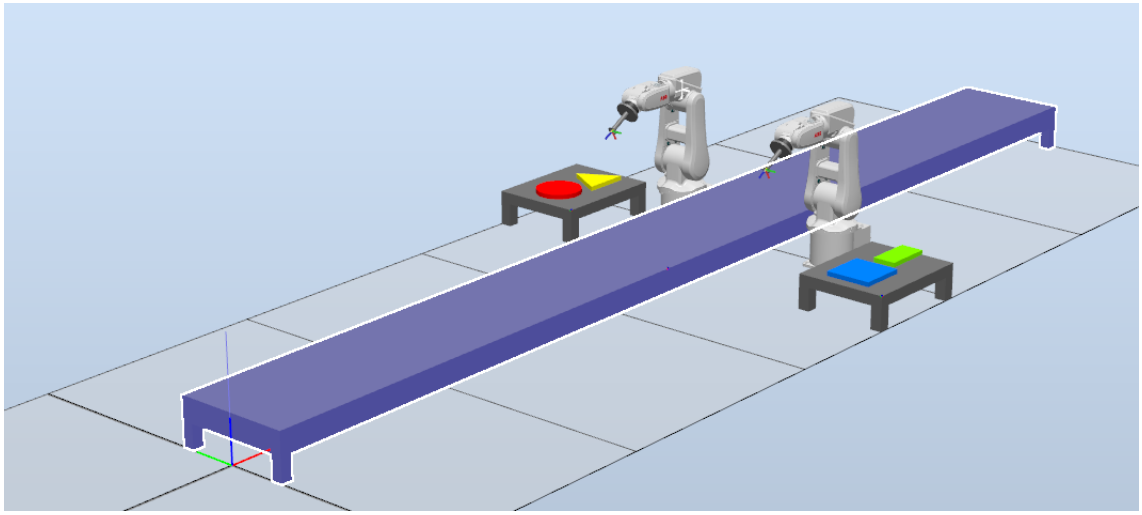

Figura 29. Cinta Transportadora

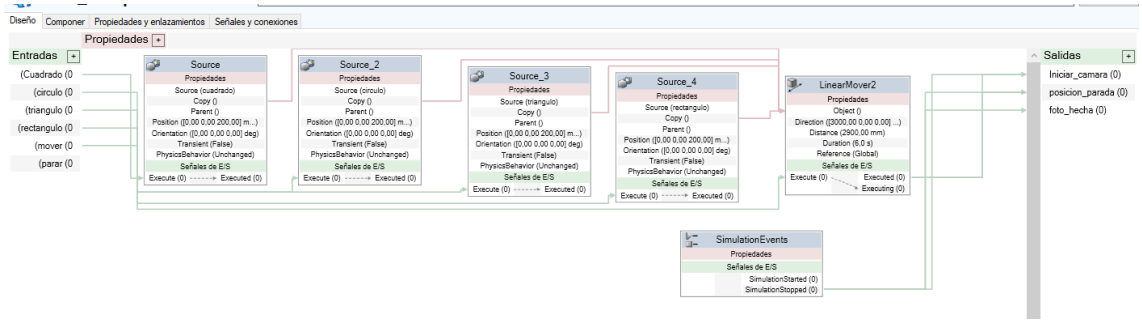

Figura 30. Lógica de la cinta transportadora

Aquí recibimos las señales del generador de piezas, que es el que nos indica que tipo de figura debe crearse. Esta señal activa el source correspondiente que genera una copia de la pieza indicada, al principio de la cinta transportadora. Una vez creada se vincula a un LinearMover que desplazará la pieza hasta la posición de tomar la foto.

Cuando la pieza llega a esa posición se lanza la señal para indicar que podemos tomar la foto.

6.2.3. Cámara Smart

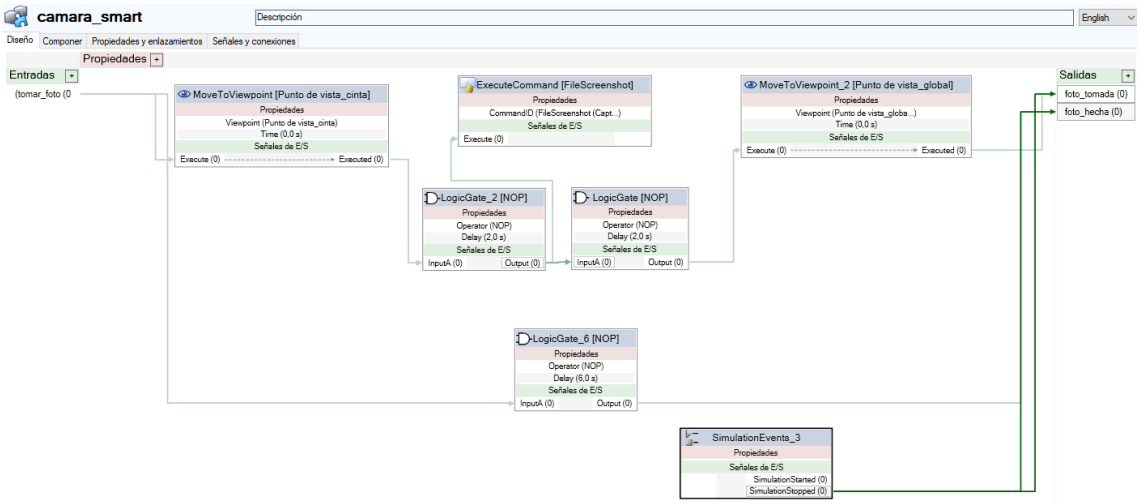

Figura 31 Lógica de la cámara smart.

Para que la cámara Smart se active necesitamos que llegue la señal de que la pieza está en la posición necesaria. Una vez recibida esta señal lo que hacemos es poner la vista de la simulación en un plano occipital que nos permite tener una buena visión de la cinta transportadora y el objeto que hay en ella. Este plano tiene que haber sido definido previamente y para movernos hasta el tenemos que usar el componente MoveToViewpoint.

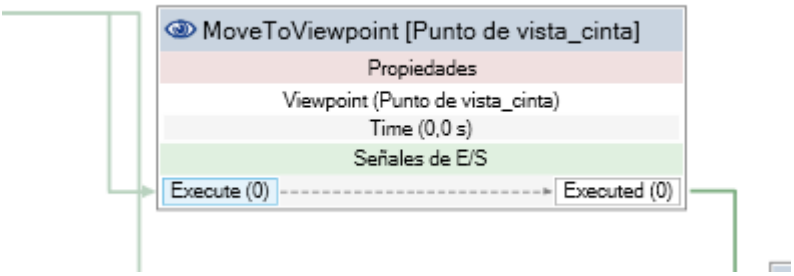

Figura 32. Detalle del componente moveToViewpoint

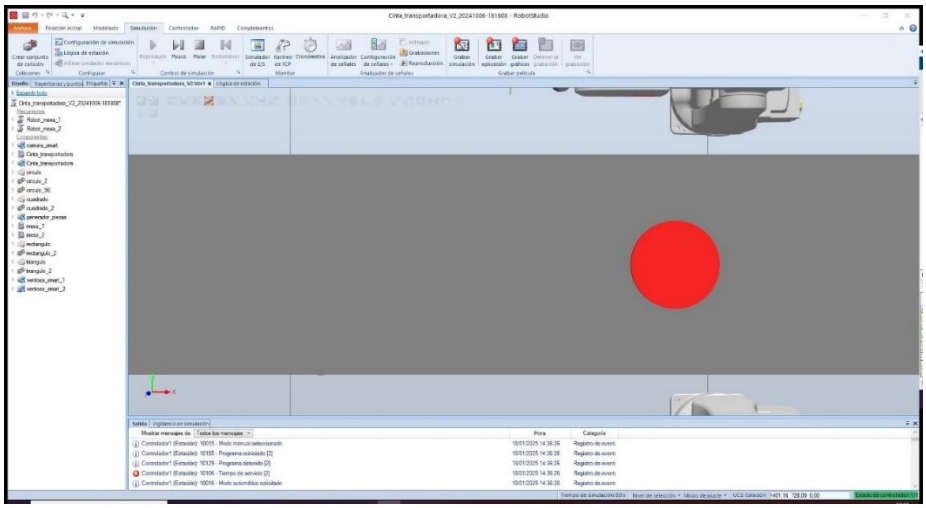

Figura 33. Detalle de la vista de la foto

Después de un delay de 2 segundos, para que no haya problemas con la imagen por ir muy deprisa, lo que hace el sistema es hacer una captura de pantalla que se guardará en la carpeta imágenes del ordenador, simulando así un sistema de visión artificial. Para ejecutar esta acción es necesario usar el componente ExecuteCommand con la instrucción de FileScreenshot.

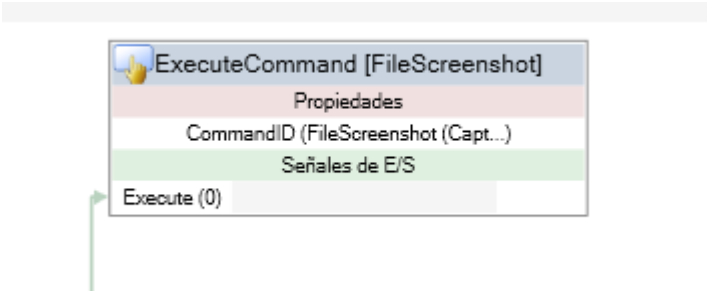

Figura 34. Detalle del componente ExecuteCommand[FileScreenshot]

Después habrá otro delay de 2 segundos para dar tiempo a que la imagen se guarde sin problemas, y una vez cumplido ese tiempo volveremos a un plano general de la estación que previamente hemos definido.
Finalmente, cuando la foto ha sido hecha mandamos una señal al controlador para que continúe el proceso.

6.2.4. Herramientas

Para las herramientas hemos creado 2 objetos inteligentes iguales, uno para cada robot, son las ventosas_smart.

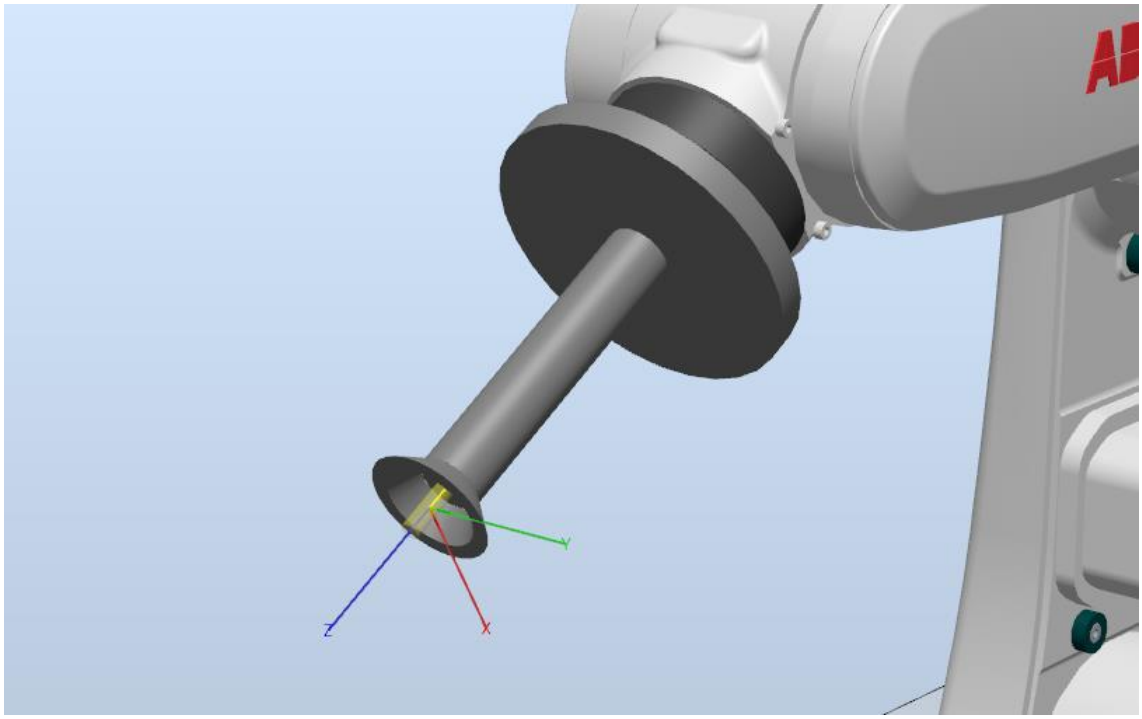

Figura 35 Herramienta utilizada por los robots

La herramienta simula una ventosa que podría ser utilizada en algún proceso industrial.

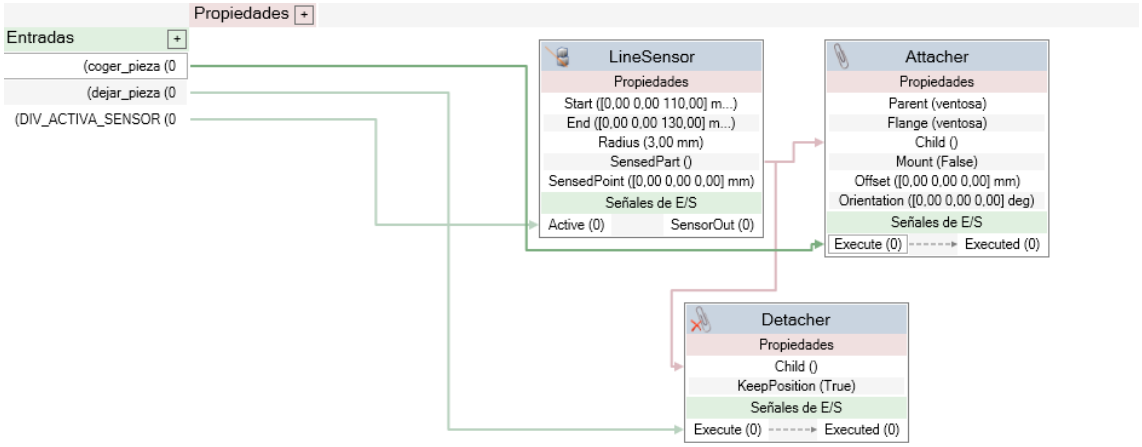

Figura 36. Lógica de la herramienta

Para la lógica hemos utilizado un Linear sensor que sirve para detectar la pieza que queremos mover.
Después usamos un attacher que hace que esa pieza se fije a la herramienta, es decir, como si activáramos la ventosa en el mundo real y se cogiera la pieza.
Para desvincular esa pieza de la herramienta tenemos que usar un detacher, que sería como desactivar la ventosa para dejar la pieza.
Estos elementos se activan mediante las señales de entrada de coger_pieza, dejar_pieza y DIV_ACTIVA_SENSOR que se corresponden con las señales de salida del controlador que habíamos visto anteriormente.

7. Análisis del código de Rapid

Para el análisis del código de Rapid, tenemos que diferenciar dos partes, una para el gobierno del robot 1 y de la sincronización y otra para el control del robot 2.

7.1. Robot 1

7.1.1. Módulo CalibData

T_ROB1/coger_pieza* | T_ROB1/CalibData*

```
MODULE CalibData

    PERS tooldata ventosa:=[TRUE,[[0,0,120],[1,0,0,0]],[1,[0,0,1],[1,0,0,0],0,0,0]];
    PERS tooldata ventosa_1:=[TRUE,[[0,0,120],[1,0,0,0]],[1,[0,0,1],[1,0,0,0],0,0,0]];
    PERS wobjdata parada:=[FALSE,TRUE,"",[[900,750,200],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];

        CONST robtarget Target_10:=[[468.276906599,50,533.999883878],[0.500000005,0,0.866025401,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_10:=[[800,250,100],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_20:=[[800,250,20],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS wobjdata mesa_1:=[FALSE,TRUE,"",[[500,-150,150],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
    CONST robtarget punto_m1_2:=[[150,200,120],[0,0,1,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_m1_1:=[[150,200,40],[0,0,1,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_m1_4:=[[350,150,120],[0,0,1,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_m1_3:=[[350,150,40],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget base_parada_10:=[[0,0,0],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PROC Path_10()
        MoveJ Target_10,v1000,z100,ventosa\WObj:=wobj0;
        MoveJ punto_10,v1000,z100,ventosa\WObj:=parada;
        MoveL punto_20,v1000,z100,ventosa\WObj:=parada;
        MoveL punto_10,v1000,z100,ventosa\WObj:=parada;
        MoveJ Target_10,v1000,z100,ventosa\WObj:=wobj0;
    ENDPROC
    PROC mover_coger()
        MoveJ Target_10,v1000,z100,ventosa\WObj:=wobj0;
        MoveJ punto_10,v1000,z100,ventosa\WObj:=parada;
        MoveL punto_20,v1000,z100,ventosa\WObj:=parada;
        MoveL punto_10,v1000,z100,ventosa\WObj:=parada;
        MoveJ Target_10,v1000,z100,ventosa\WObj:=wobj0;
        MoveL base_parada_10,v1000,z100,ventosa_1\WObj:=parada;
    ENDPROC
    PROC dejar_circulo()
        MoveJ Target_10,v1000,fine,ventosa_1\WObj:=wobj0;
        MoveJ punto_m1_2,v500,z0,ventosa_1\WObj:=mesa_1;
        MoveL punto_m1_1,v500,fine,ventosa_1\WObj:=mesa_1;
        MoveL punto_m1_2,v500,fine,ventosa_1\WObj:=mesa_1;
        MoveJ Target_10,v500,fine,ventosa_1\WObj:=wobj0;
    ENDPROC

    PROC dejar_tri()
        MoveJ Target_10,v500,fine,ventosa_1\WObj:=wobj0;
        MoveJ punto_m1_4,v500,fine,ventosa_1\WObj:=mesa_1;
        MoveL punto_m1_3,v100,fine,ventosa_1\WObj:=mesa_1;
        MoveL punto_m1_4,v100,fine,ventosa_1\WObj:=mesa_1;
        MoveJ Target_10,v500,fine,ventosa_1\WObj:=wobj0;
    ENDPROC
ENDMODULE
```

En este módulo recogemos todos los datos necesarios para luego ejecutar nuestro proceso.

Tenemos diferentes tipos de datos:

- Tooldata: es el objeto donde guardamos los valores de nuestra herramienta. En nuestro caso usaremos ventosa_1
- Wobjdata: los work object data son objetos donde guardamos los datos del objeto de trabajo, es decir, la zona donde vamos a trabajar. Nosotros tenemos:
    - Parada: es la zona de parada de la cinta, donde está la pieza esperando a ser recogida.
    - Mesa_1: es la zona de la mesa 1 donde deberemos dejar la pieza, en este caso es la mesa que contiene al círculo y al triangulo.

– Robtarget: son los diferentes puntos que hemos definido para poder mover el tcp del robot a través de ellos:

- Target_10:

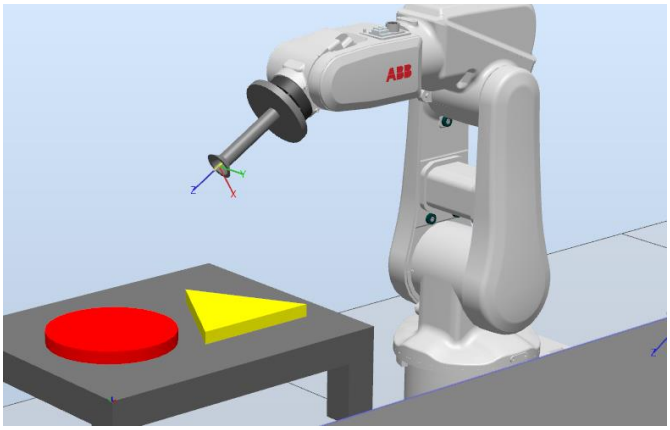

Figura 37 Robot 1 en posición Target_10

- Punto_10:

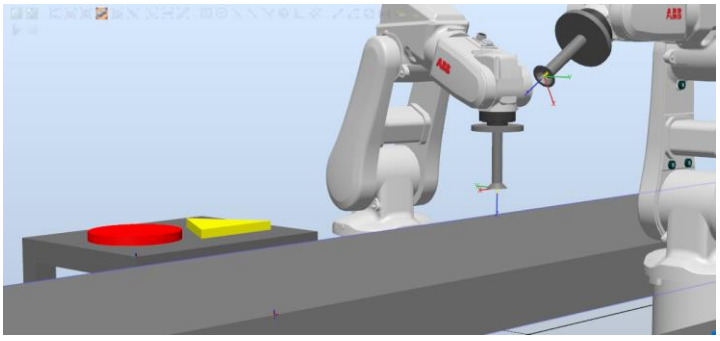

Figura 38 Robot 1 en posición Punto_10

- Punto_20:

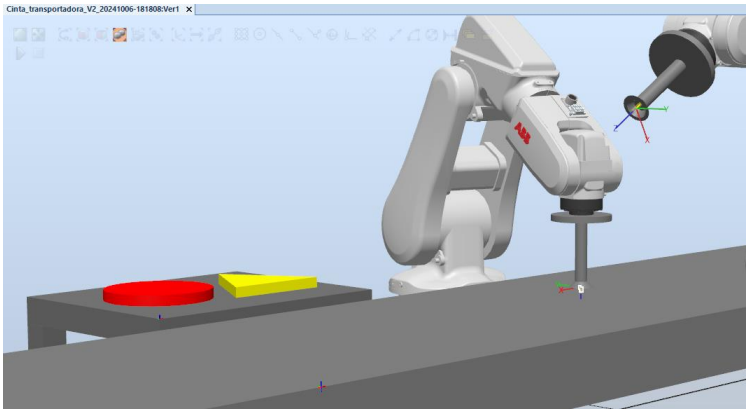

Figura 39 Robot 1 en posición Punto_20

- Punto_m1_1:

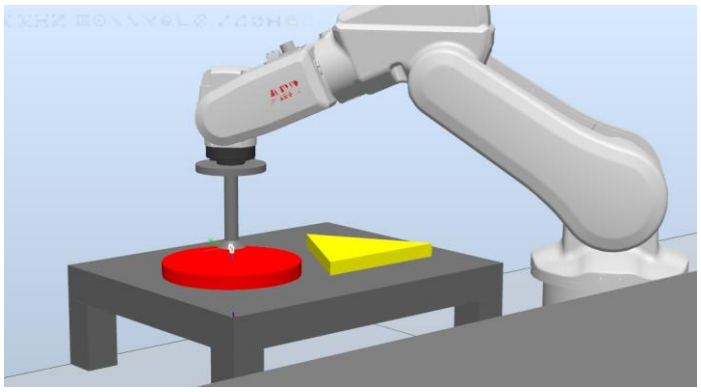

Figura 40 Robot 1 en posición Punto_m1_1

- Punto_m1_2:

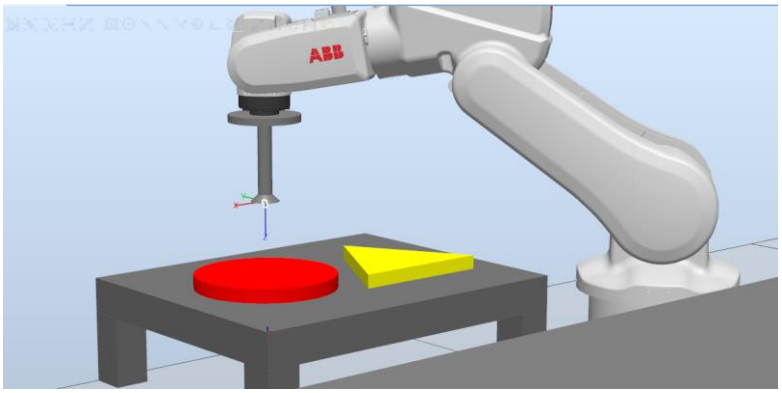

Figura 41 Robot 1 en posición Punto_m1_2

- Punto_m1_3:

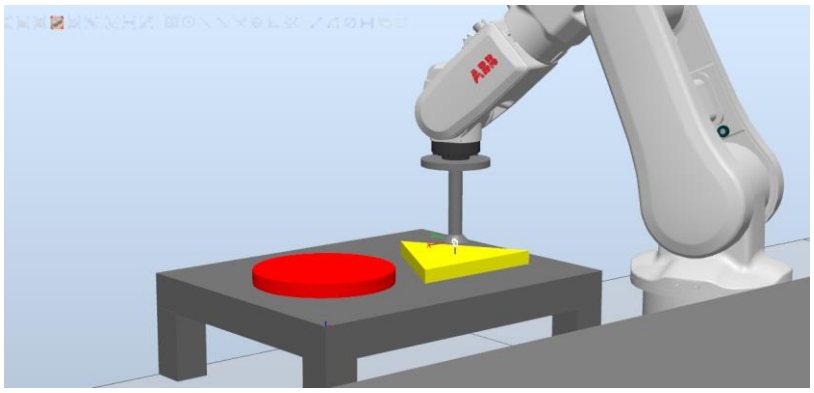

Figura 42 Robot 1 en posición Punto_m1_3

- Punto_m1_4:

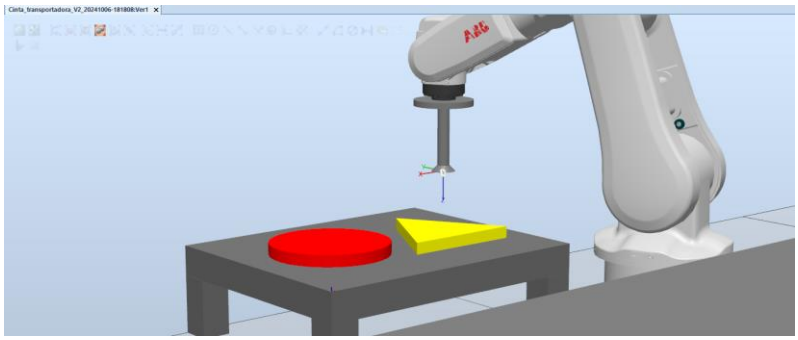

Figura 43 Robot 1 en posición Punto_m1_4

7.1.2. Módulo M_Main

Este es el modulo principal del proceso y desde el cual llamamos al resto de funciones y módulos.

```
1   MODULE M_Main
2       CONST speeddata MinVel:=v50;
3       CONST speeddata MaxVel:=v500;
```

Lo primero que hacemos es definir las velocidades máxima y mínima a la que va a ir nuestro robot.

```
5       PERS num entrada_1;
6       PERS num robot:=0;
7       PERS num paso:=1;
8       PERS num pieza_n:=4;
9       PERS num x_Pieza_mesa:=0;
10      PERS num y_Pieza_mesa:=0;
11      PERS num h_Pieza_mesa:=20;
12
13      PERS num x_Pieza_cinta:=0;
14      PERS num y_Pieza_cinta:=0;
15
16      PERS num terminar:=0;
17      PERS num x_Pieza_Origen{4}:=[0,0,0,0];
18      PERS num h_circ:=0;
19      PERS num h_tri:=0;
20      PERS num h_cuad:=40;
21      PERS num h_rect:=120;
22      PERS num m:=0;
23      PERS num n:=0;
24      PERS num robot_movido:=0;
```

Después definimos las variables que vamos a usar en el todo el modulo.

```
26      PROC main()
27          TPErase;
28           Init_signals;
29           TPWrite "Inicio del Programa";
30
```

Primero borramos todos los mensajes que se han mandado a la flexpendant, y llamamos a la función Init_signals en la que se inicializan todas las señales y variables.

```
31          WHILE TRUE DO
32        IF paso=0 then
33          Set DO_1;
34          Set DO_8;
35          WaitDI DI_1,1;
36          Reset DO_8;
37       entrada_1:=DI_1;
38       TPWrite "entrada_1:"\Num:=entrada_1;
39       TPWrite "DI_1:"\Num:=DI_1;
40
```

A continuación entramos en un bucle controlado por la variable paso que nos servirá como semáforo entre Rapid y Matlab.

Ponemos a 1 la señal DO_1, que es la que da inicio al proceso de generar las piezas. También ponemos a 1 la señal DO_8 que es la que activa el LinearMover que hace que las piezas se muevan por la cinta.

Después de esto esperamos a recibir la señal DI_1, que es la que nos indica que ya se ha tomado la foto de la pieza en la cinta, la guardamos en la variable entrada_1, y reseteamos la señal de salida DO_8.

```
41          IF (entrada_1=1) THEN
42              TPWrite "hemos entrado en el if";
43              paso:=1;
44              WaitUntil(paso=2);
45              pieza_n:=x_Pieza_Origen{4};
46             x_Pieza_cinta:=x_Pieza_Origen{1};
47             y_Pieza_cinta:=x_Pieza_Origen{2};
48              TPWrite "hemos entrado en el if";
49              TPWrite "entrada_1:"\Num:=pieza_n;
```

Si la entrada_1 está a 1 entramos en el if donde lo primero que haremos será poner la variable paso a 1 y esperar a que se ponga a 2. Esta variable es el semáforo con Matlab. Cuando está a 1, le toca al programa del análisis grafico estudiar la foto y ver que pieza tenemos. Una vez tiene los datos necesarios, los manda de nuevo a Rapid y cambia el valor de paso a 2 para poder continuar con el proceso.

Los datos que recibimos son pieza_n que nos indica con un valor numérico que tipo de pieza es:

- Pieza_n=1 -> mover robot 1 al círculo.
- Pieza_n=2 -> mover robot 1 al triangulo.
- Pieza_n=3 -> mover robot 2 al cuadrado.
- Pieza_n=4 -> mover robot 2 al rectángulo.

```
IF pieza_n=1 THEN
    TPWrite "mover robot 1 al circulo";

    h_circ:=h_circ+20;
    x_Pieza_mesa:=150;
    y_Pieza_mesa:=150;
    h_Pieza_mesa:=h_circ-100;
    robot:=1;
ELSEIF pieza_n=2 THEN
    TPWrite "mover robot 1 al triangulo";
    h_tri:=h_tri+20;
    x_Pieza_mesa:=350;
    y_Pieza_mesa:=157.735;
    h_Pieza_mesa:=h_tri-100;
    robot:=1;
ELSEIF pieza_n=3 THEN
    TPWrite "mover robot 2 al cuadrado";
    h_cuad:=h_cuad+20;
    x_Pieza_mesa:=150;
    y_Pieza_mesa:=200;
    h_Pieza_mesa:=h_cuad-100;

    robot:=2;
ELSEIF pieza_n=4 THEN
    TPWrite "mover robot 2 al rectangulo";
    h_rect:=h_rect+20;
    x_Pieza_mesa:=400;
    y_Pieza_mesa:=200;
    h_Pieza_mesa:=h_rect-100;
    robot:=2;
ENDIF
```

Una vez tenemos identificada que pieza es, indicamos que robot se debe mover con la variable robot.

```
IF robot=1THEN
    TPWrite "mover robot 1";
    IF pieza_n=1 THEN
       TPWrite "mover robot 2 al circulo";
       circulo;
       robot_movido:=1;
    ENDIF
    IF pieza_n=2 THEN
       TPWrite "mover robot 2 al triangulo";
       triangulo;
       robot_movido:=1;
    ENDIF
ELSEIF robot=2THEN
    WaitUntil robot_movido=1;
ENDIF
```

Si tenemos que la variable robot es igual a 1 y la pieza_n es igual a 1 entonces tenemos que el robot 1 es el que tiene que moverse al círculo. Para ello llamamos a la función círculo. Después indicamos que el robot ya se ha movido con la variable robot_movido.

Si tenemos que la variable robot es igual a 1 y la pieza_n es igual a 2 entonces tenemos que el robot 1 es el que tiene que moverse al triangulo. Para ello

llamamos a la función triangulo. Después indicamos que el robot ya se ha movido con la variable robot_movido.

Si por el contrario tenemos que la variable robot es igual a 2 tenemos que esperar a que el robot 2 se mueva.

```
97              x_Pieza_mesa:=0;
98              y_Pieza_mesa:=0;
99              h_Pieza_mesa:=0;
100             x_Pieza_cinta:=0;
101             y_Pieza_cinta:=0;
102             robot:=0;
103             robot_movido:=0;
104
105           ENDIF
106           TPWrite"terminar";
107
108 !                 WaitTime 120;
109 !                 terminar:=1;
110 !                 stop;
111         Reset DO_1;
112         Reset DO_8;
113         ENDIF
114           ENDWHILE
115     ENDPROC
```

Una vez se ha movido el robot correspondiente reinicializamos las variables y terminamos el proceso.

Función Init_signals():

```
118     PROC Init_signals()
119         Reset DO_1;
120         Reset DO_2;
121         Reset DO_3;
122         Reset DO_4;
123         Reset DO_5;
124         Reset DO_6;
125         Reset DO_7;
126         Reset DO_8;
127 !        Reset DI_1;
128 !        Reset DI_2;
129 !        Reset DI_3;
130 !        mesa_1:=[FALSE,TRUE,"",[[2500,600,150],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
131 !        mesa_2:=[FALSE,TRUE,"",[[2500,-1000,150],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
132 !        zona_parada:=[FALSE,TRUE,"",[[2100,-250,200],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
133         robot:=0;
134         paso:=0;
135         terminar:=0;
136         x_Pieza_Origen:=[0,0,0,0];
137         x_Pieza_mesa:=0;
138         y_Pieza_mesa:=0;
139         h_circ:=0;
140         h_tri:=0;
141         m:=0;
142         n:=0;
143         x_Pieza_cinta:=0;
144         y_Pieza_cinta:=0;
145         robot_movido:=0;
146 !        y_Pieza_Origen:=[0];
147 !        ori_Pieza_Origen:=[0];
148     ENDPROC
```

Esta función sirve para inicializar todas las señales y variables del proceso M_Main.

7.1.3. Módulo mover_robot_1

Dentro del módulo mover_robot_1 tenemos las funciones para mover dicho robot para que coja la pieza y la lleve a su posición, siempre y cuando sea un círculo o un triángulo.

Función circulo:

```
25      PROC circulo()
26          TPWrite"Es un circulo";
27          MoveJ Target_10,v200,fine,ventosa\WObj:=wobj0;
28          MoveJ punto_10,v100,fine,ventosa\WObj:=parada;
29          MoveL punto_20,v50,fine,ventosa\WObj:=parada;
30          m_main_coger_pieza;
31          MoveL punto_10,v50,fine,ventosa\WObj:=parada;
32
33          MoveJ Target_10,v200,fine,ventosa\WObj:=wobj0;
34          MoveJ punto_m1_2,v100,fine,ventosa_1\WObj:=mesa_1;
35          MoveL punto_m1_1,v50,fine,ventosa_1\WObj:=mesa_1;
36          m_main_dejar_pieza;
37          MoveL punto_m1_2,v50,fine,ventosa_1\WObj:=mesa_1;
38          MoveJ Target_10,v200,fine,ventosa_1\WObj:=wobj0;
39          robot:=0;
40      ENDPROC
```

En este proceso el robot 1 se moverá a la posición Tarjet_10 que es la inicial si no está ya. Después se moverá a la posición punto_10 donde se colocara encima de la pieza a una altura para poder después realizar la aproximación de manera más lenta, hasta alcanzar la posición punto_20.

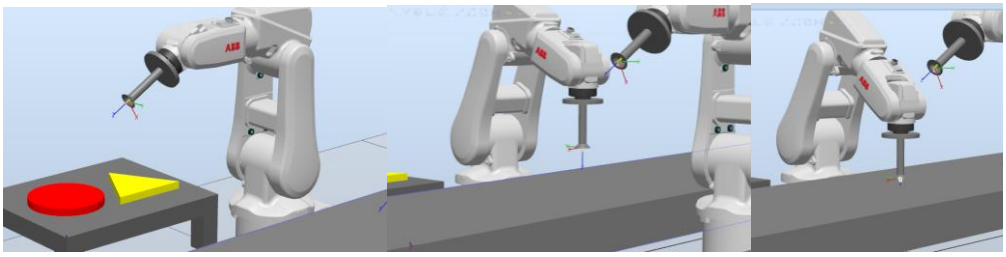

Figura 44 Secuencia del movimiento del robot 1 para coger la pieza

Una vez hemos alcanzado esa posición llamamos a la función m_main_coger_pieza. Cuando hemos cogido la pieza volvemos a mover el robot a las mismas posiciones pero a la inversa, es decir, pasamos de la posición punto_20 a la posición punt_10, y de ahí volvemos a Tarjet_10.

A partir de ahí procederemos a dejar la pieza en su sitio. Primero nos moveremos a la posición punto_m1_2, para después bajar hasta alcanzar la posición para dejar la pieza, punto_m1_1. Una vez en esa posición llamamos a la función m_main_dejar_pieza. Cuando hemos dejado la pieza hacemos el

recorrido inverso hasta llegar a la posición inicial Tarjet_10 y ponemos la variable robot a 0.

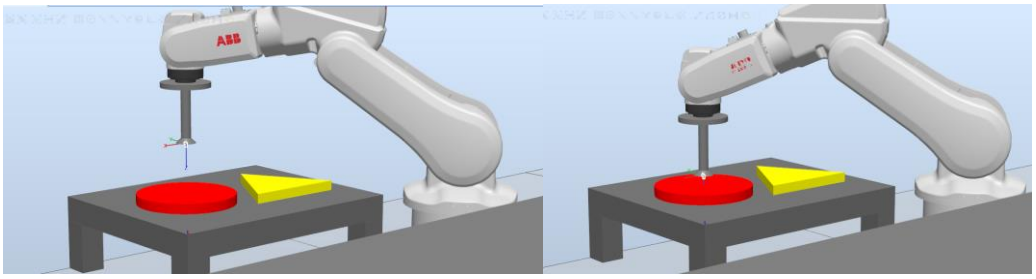

Figura 45 Secuencia del movimiento del robot 1 para dejar el círculo

Función triangulo:

```
42      PROC triangulo()
43          TPWrite"Es un triangulo";
44          MoveJ Target_10,v200,fine,ventosa\WObj:=wobj0;
45          MoveJ punto_10,v200,fine,ventosa\WObj:=parada;
46          MoveL punto_20,v50,fine,ventosa\WObj:=parada;
47          m_main_coger_pieza;
48          MoveL punto_10,v50,fine,ventosa\WObj:=parada;
49
50          MoveJ Target_10,v200,fine,ventosa\WObj:=wobj0;
51          MoveJ punto_m1_4,v100,fine,ventosa_1\WObj:=mesa_1;
52          MoveL punto_m1_3,v50,fine,ventosa_1\WObj:=mesa_1;
53          m_main_dejar_pieza;
54          MoveL punto_m1_4,v50,fine,ventosa_1\WObj:=mesa_1;
55          MoveJ Target_10,v200,fine,ventosa_1\WObj:=wobj0;
56          robot:=0;
57
58      ENDPROC
59
```

El principio de la función es igual que en círculo, es decir pasamos de tarjet_10 a punto_10, de ahí a punto_20, cogemos la pieza y volvemos a tarjet_10 pasando por punto_10 otra vez. A partir de ahí nos movemos hacia la mesa para dejar la pieza en su sitio. Para ello vamos al punto_m1_4, y después nos acercamos despacio a la altura donde dejaremos la pieza, punto_m1_3. Dejamos la pieza llamando a la función m_main_dejar_pieza. Una vez la hemos soltado hacemos el camino inverso hasta tarjet_10 y ponemos la variable robot a 0.

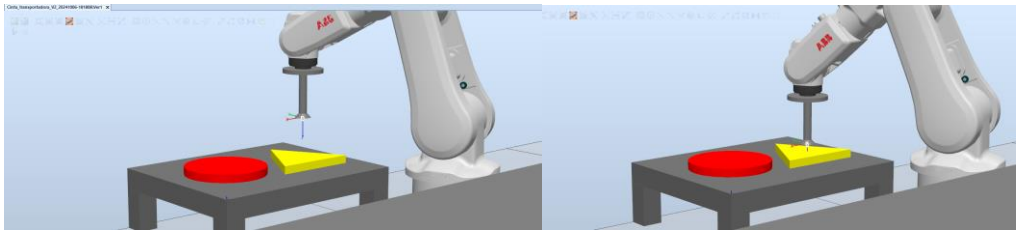

Figura 46 Secuencia del movimiento del robot 1 para dejar el triángulo

7.1.4. Módulo coger_pieza

Dentro de este módulo tenemos la función m_main_coger_pieza.

```
MODULE coger_pieza

    proc m_main_coger_pieza()


        Set DO_4;
        WaitTime 1;
        TPWrite"pieza_cogida";
        Set DO_4;
        PulseDO DO_2;

        endproc
ENDMODULE

```

Aquí ponemos a uno la señal DO_4 que es la que activa el sensor de la herramienta y mandamos un pulso en DO_2 que es la señal del attacher.

7.1.5. Módulo dejar_pieza

Dentro de este modulo tenemos la función m_main_dejar_pieza.

```
MODULE dejar_pieza

    proc m_main_dejar_pieza()
        Reset DO_2;
!            PulseDO\PLength:=1,DO_3;
            Set DO_3;
       WaitTime 1;
        Reset DO_4;
        TPWrite"pieza_dejada";
        endproc
ENDMODULE

```

En esta funcion reseteamos la señal DO_2 que es la que va al attacher de la herramienta y activamos DO_3 que es la del dettacher. Despues esperamos 1 segundo y desactivamos DO_4 que es la señal que va al sensor de la herramienta.

7.2. Robot 2

7.2.1. Módulo CalibData

```
MODULE CalibData


    PERS tooldata ventosa:=[TRUE,[[0,0,120],[1,0,0,0]],[1,[0,0,1],[1,0,0,0],0,0,0]];
    PERS tooldata ventosa_2:=[TRUE,[[0,0,120],[1,0,0,0]],[1,[0,0,1],[1,0,0,0],0,0,0]];
    PERS wobjdata mesa_2:=[FALSE,TRUE,"",[[500,1450,150],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
    PERS wobjdata zona_parada:=[FALSE,TRUE,"",[[2100,-250,200],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
    PERS robtarget posicion_inicial_rob_2:=[[31.7231,550,384],[0,0.866025,0,-0.5],[0,0,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
    PERS robtarget origen_zona_parada:=[[600,250,100],[0,0.707107,0.707107,0],[0,0,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
    PERS robtarget origen_mesa_2:=[[0,0,100],[0,0.707107,0.707107,0],[0,0,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
    PERS robtarget origen_cinta:=[[0,0,100],[0,0.707107,0.707107,0],[0,0,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
    PERS robtarget posicion_1:=[[798.471,154.653,0],[0,1,0,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS robtarget Target_10:=[[431.723122473,-200,334],[0,0.866025404,0,-0.5],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS wobjdata parada_2:=[FALSE,TRUE,"",[[900,750,200],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];
    CONST robtarget Target_20:=[[468.276877527,950,534],[0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_30:=[[800,250,100],[0,0,1,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_40:=[[800,250,20],[0,0,1,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_m2_2:=[[150,200,100],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_m2_1:=[[150,200,40],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_m2_4:=[[400,200,100],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget punto_m2_3:=[[400,200,40],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PROC Path_20()
        MoveL posicion_inicial_rob_2,v1000,z100,ventosa_2\WObj:=mesa_2;
        MoveL posicion_1,v1000,z100,ventosa_2\WObj:=zona_parada;
        MoveL origen_zona_parada,v1000,z100,ventosa_2\WObj:=zona_parada;
        MoveL origen_mesa_2,v1000,z100,ventosa_2\WObj:=mesa_2;
        MoveL origen_cinta,v1000,z100,ventosa_2\WObj:=zona_parada;
        MoveL Target_10,v1000,z100,ventosa_2\WObj:=zona_parada;
    ENDPROC
    PROC coger_2()
        MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
        MoveJ punto_30,v100,fine,ventosa_2\WObj:=parada_2;
        MoveL punto_40,v100,fine,ventosa_2\WObj:=parada_2;
        MoveL punto_30,v100,fine,ventosa_2\WObj:=parada_2;
        MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
    ENDPROC

    PROC dejar_cuad()
        MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
        MoveJ punto_m2_2,v200,fine,ventosa_2\WObj:=mesa_2;
        MoveL punto_m2_1,v100,fine,ventosa_2\WObj:=mesa_2;
        MoveL punto_m2_2,v100,fine,ventosa_2\WObj:=mesa_2;
        MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
    ENDPROC
    PROC dejar_rect()
        MoveJ Target_20,v500,fine,ventosa_2\WObj:=wobj0;
        MoveJ punto_m2_4,v200,fine,ventosa_2\WObj:=mesa_2;
        MoveL punto_m2_3,v100,fine,ventosa_2\WObj:=mesa_2;
        MoveL punto_m2_4,v100,fine,ventosa_2\WObj:=mesa_2;
        MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
    ENDPROC
ENDMODULE
```

En este módulo recogemos todos los datos necesarios para luego ejecutar nuestro proceso.

Tenemos diferentes tipos de datos:

- Tooldata: es el objeto donde guardamos los valores de nuestra herramienta. En nuestro caso usaremos ventosa_1
- Wobjdata: los work object data son objetos donde guardamos los datos del objeto de trabajo, es decir, la zona donde vamos a trabajar. Nosotros tenemos:
    - Parada: es la zona de parada de la cinta, donde está la pieza esperando a ser recogida.
    - Mesa_2: es la zona de la mesa 2 donde deberemos dejar la pieza, en este caso es la mesa que contiene al cuadrado y al rectángulo.

– Robtarget: son los diferentes puntos que hemos definido para poder mover el tcp del robot a través de ellos:

- Tarjet_20:

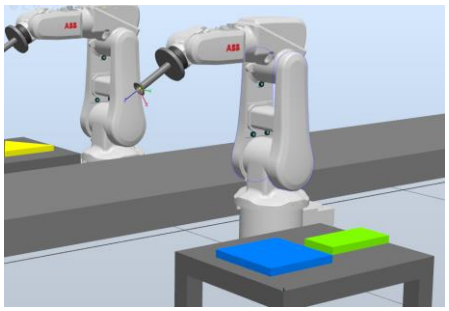

Figura 47 Robot 2 en posición Target_20

- Punto_30:

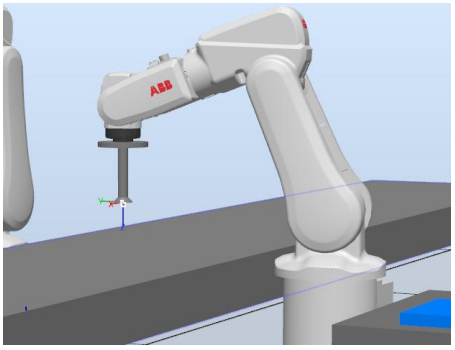

Figura 48 Robot 2 en posición Punto_30

- Punto_40:

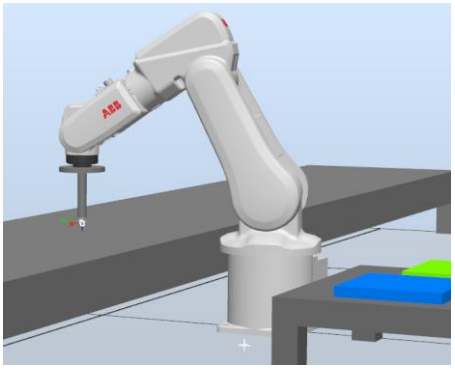

Figura 49 Robot 2 en posición Punto_40

- Punto_m2_1:

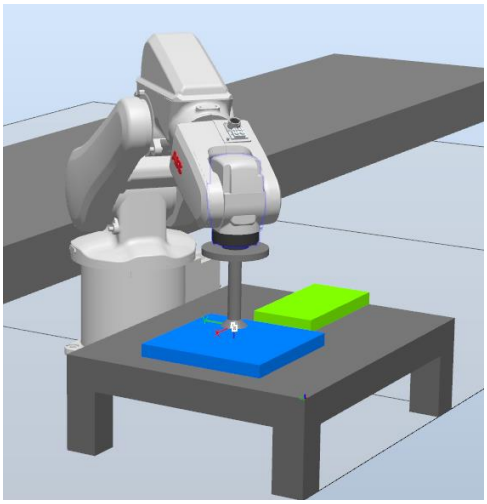

Figura 50 Robot 2 en posición Punto_m2_1

- Punto_m2_2:

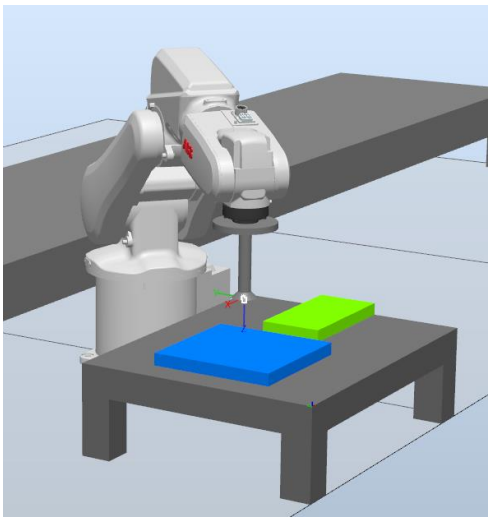

Figura 51 Robot 2 en posición Punto_m2_2

- Punto_m2_3:

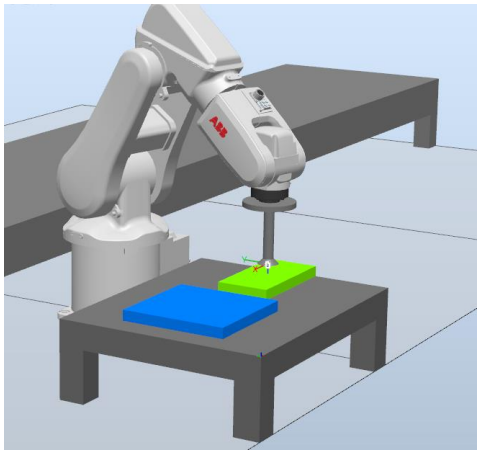

Figura 52 Robot 2 en posición Punto_m2_3

- Punto_m2_4:

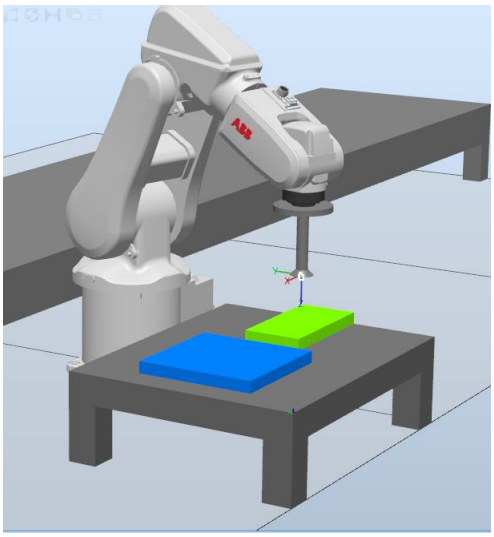

Figura 53 Robot 2 en posición Punto_m2_4

7.2.2. Módulo M_main2

Este módulo es el principal del robot 2.

```
MODULE M_main2
    PERS num robot:=0;
    PERS num pieza_n:=4;
    PERS num x_Pieza_mesa:=0;
    PERS num y_Pieza_mesa:=0;
    PERS num h_Pieza_mesa:=20;
    PERS num terminar:=0;
    PERS num x_Pieza_Origen{4}:=[0,0,0,0];
    PERS num h_cuad:=40;
    PERS num h_rect:=120;
    PERS num i:=0;
    PERS num j:=0;
    CONST speeddata MinVel:=v50;
    CONST speeddata MaxVel:=v500;
    PERS num robot_movido:=0;

    PROC main()
        robot:=0;
        WHILE TRUE DO
            IF robot=2 THEN
                TPWrite "mover robot 2";
                IF pieza_n=3 THEN
                    TPWrite "mover robot 2 al cuadrado";
                    cuadrado;
                    robot_movido:=1;
                ENDIF
                IF pieza_n=4 THEN
                    TPWrite "mover robot 2 al rectangulo";
                    rectangulo;
                    robot_movido:=1;
                ENDIF
                robot_movido:=1;
            ENDIF
        ENDWHILE
    ENDPROC
```

Este modulo es más sencillo que el del robot 1 porque no gestiona toda la parte de la comunicación ni de la sincronización.

Primero inicializamos las variables que vamos a utilizar. Despues entramos en la función main, en la que hay un bucle infinito. Si el main del robot 1 da el valor 2 a la variable robot, entramos en un if en el cual elegiremos a donde se tiene que mover el robot dependiendo de la variable pieza_n. Si es 3 llamaremos a la funcion cuadraro, la cual llevara la pieza hasta la posicion del cuadrado y una vez acabe pondremos la variable robot_movido a 1 para devolverle el control al proceso main del robot 1. Si pieza_n es 4 llamaremos a la funcion rectangulo y haremos lo mismo que en la anterior.

7.2.3. Módulo mover_robot_2

```
25      PROC cuadrado()
26          TPWrite"Es un cuadrado";
27          MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
28          MoveJ punto_30,v100,fine,ventosa_2\WObj:=parada_2;
29          MoveL punto_40,v50,fine,ventosa_2\WObj:=parada_2;
30          m_main_coger_pieza;
31          MoveL punto_30,v50,fine,ventosa_2\WObj:=parada_2;
32          MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
33
34          MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
35          MoveJ punto_m2_2,v200,fine,ventosa_2\WObj:=mesa_2;
36          MoveL punto_m2_1,v50,fine,ventosa_2\WObj:=mesa_2;
37          m_main_dejar_pieza;
38          MoveL punto_m2_2,v50,fine,ventosa_2\WObj:=mesa_2;
39          MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
40          robot:=0;
41
42      ENDPROC
43
44      PROC rectangulo()
45          TPWrite"Es un rectangulo";
46          MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
47          MoveJ punto_30,v100,fine,ventosa_2\WObj:=parada_2;
48          MoveL punto_40,v50,fine,ventosa_2\WObj:=parada_2;
49          m_main_coger_pieza;
50          MoveL punto_30,v50,fine,ventosa_2\WObj:=parada_2;
51          MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
52
53          MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
54          MoveJ punto_m2_4,v100,fine,ventosa_2\WObj:=mesa_2;
55          MoveL punto_m2_3,v50,fine,ventosa_2\WObj:=mesa_2;
56          m_main_dejar_pieza;
57          MoveL punto_m2_4,v50,fine,ventosa_2\WObj:=mesa_2;
58          MoveJ Target_20,v200,fine,ventosa_2\WObj:=wobj0;
59          robot:=0;
```

En este módulo tenemos las funciones cuadrado y rectángulo que sirven para coger la pieza y llevarla a la posición correspondiente.

En cuadrado movemos el robot desde tarjet_20 hasta punto_30, donde en un movimiento vertical y lento moveremos el robot hasta la posición punto_40 para poder coger la pieza llamando a la función coger_pieza.

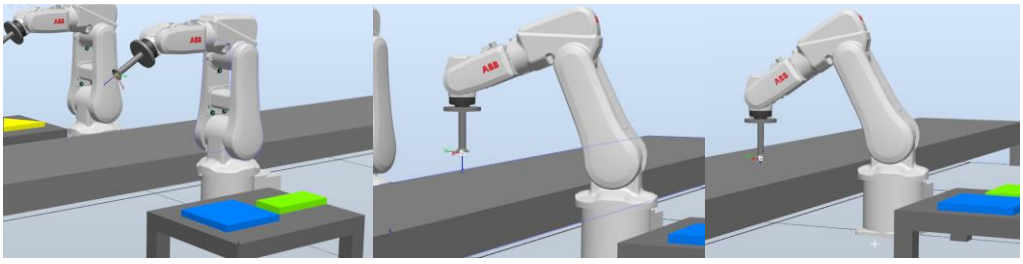

Figura 54 Secuencia del movimiento del robot 2 para coger la pieza

Una vez tenemos la pieza cogida nos movemos hasta punto_m2_2 pasando por tarjet_20, para después en un movimiento vertical y lento acercarnos hasta la posición punto_m2_1 donde dejaremos la pieza con la función dejar_pieza.

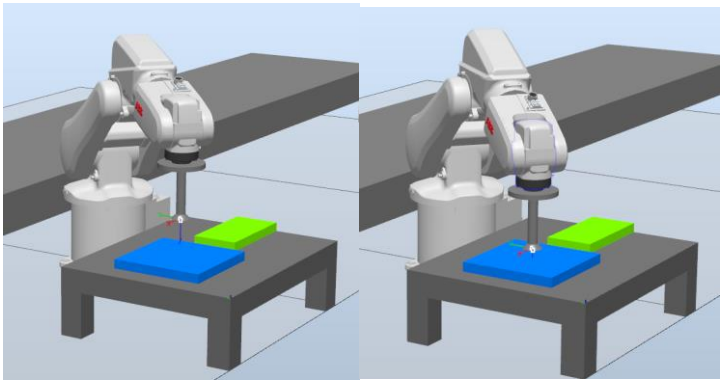

Figura 55 Secuencia del movimiento del robot 2 para dejar el cuadrado

Después haremos el camino inverso hasta tarjet_20 para dejar el robot en la posición inicial.

En rectángulo movemos el robot desde tarjet_20 hasta punto_30, donde en un movimiento vertical y lento moveremos el robot hasta la posición punto_40 para poder coger la pieza llamando a la función coger_pieza.

Una vez tenemos la pieza cogida nos movemos hasta punto_m2_4 pasando por tarjet_20, para después en un movimiento vertical y lento acercarnos hasta la posición punto_m2_3 donde dejaremos la pieza con la función dejar_pieza.

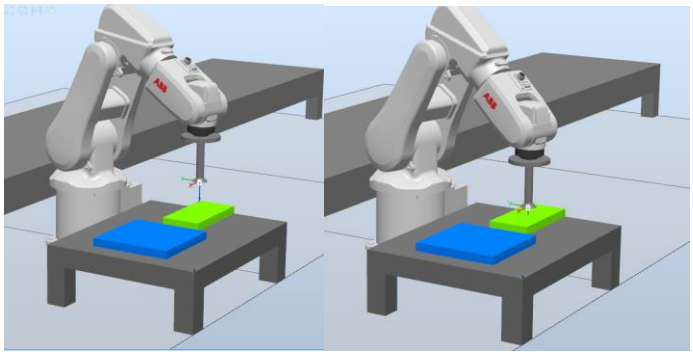

Figura 56 Secuencia del movimiento del robot 2 para dejar el rectángulo

Después haremos el camino inverso hasta tarjet_20 para dejar el robot en la posición inicial.

7.2.4. Módulo coger_pieza

```
1   MODULE coger_pieza
2
3       proc m_main_coger_pieza()
4
5
6           Set DO_7;
7           WaitTime 1;
8           TPWrite"pieza_cogida";
9           PulseDO DO_5;
10          endproc
11  ENDMODULE
12
```

En este módulo tenemos la función m_main_coger_pieza en la cual ponemos a 1 la señal DO_7, que es la que nos activa el sensor de la herramienta y mandamos un pulso en DO_5 que es la señal del attacher.

7.2.5. Módulo dejar_pieza

```
1   MODULE dejar_pieza
2
3       proc m_main_dejar_pieza()
4
5
6
7           Reset DO_5;
8   !     PulseDO\PLength:=1,DO_6;
9           Set DO_6;
10         WaitTime 1;
11          Reset DO_7;
12          TPWrite"pieza_dejada";
13          endproc
14  ENDMODULE
15
```

En este modulo tenemos la funcion m_main_dejar_pieza, donde reseteamos la señal DO_2 que es la que va al attacher de la herramienta y activamos DO_3 que es la del dettacher. Despues esperamos 1 segundo y desactivamos DO_4 que es la señal que va al sensor de la herramienta.

8. Análisis del código de Matlab

En este apartado estudiaremos el código de Matlab, el cual se encarga del análisis gráfico de las imágenes.

```
close all; clear all; clc;
%aqui metemos todo lo de la conexion opc
da=opcda('localhost','ABB.IRC5.OPC.Server.DA');%DESKTOP-HEO7OS8_IRB120T_3_58_2   DESKTOP-HEO7OS8_Controlador1
connect(da);
grp = addgroup(da);
itm1 = additem(grp,'DESKTOP-HEO7OS8_Controlador1.RAPID.T_ROB1.M_Main.paso');
itm11 = additem(grp,'DESKTOP-HEO7OS8_Controlador1.RAPID.T_ROB1.M_Main.terminar');


itm2 = additem(grp,'DESKTOP-HEO7OS8_Controlador1.RAPID.T_ROB1.M_Main.x_Pieza_Origen');

```

En las primeras líneas del código de Matlab establecemos la conexión OPC vinculando las variables “itm1” y “itm11” con las variables “paso” y “terminar” de Rapid. Estas nos servirán de semáforos para saber cuándo tiene que esperar un programa y cuando tiene que esperar el otro o para saber cuándo termina la simulación.

También vincularemos la variable “itm2” con “x_Pieza_Origen”. Aquí usaremos un array para poder pasar los datos de la pieza de Matlab a Rapid.

```
  terminar=read(itm11);
  terminar=terminar.Value;
 while terminar==0
paso=read(itm1);
paso=paso.Value;

    if paso==1
```

Ahora crearemos un bucle en el cual se ejecuta el código hasta que Rapid no nos envíe la señal de que la simulación se ha terminado.

Si “terminar” se mantiene a cero seguiremos en el bucle en el cual se lee la variable “paso” de Rapid continuamente hasta que esta vale 1, que es lo que nos indica que la pieza está en posición y la captura de pantalla se ha hecho correctamente.

```
ruta=pwd;
addpath(genpath('C:\Users\Alejandro\Pictures'));
cd C:\Users\Alejandro\Pictures;
dirimagenes=dir('*.jpg');
tam=length(dirimagenes);
[O, map]=imread(dirimagenes(tam).name);
cd (ruta);
```

Cuando tenemos la captura hecha y podemos empezar el análisis gráfico, lo primero que hay que hacer es coger la imagen de la carpeta “C:\Users\Alejandro\Pictures”, utilizando este pad para ello y cogiendo la última imagen que se ha guardado en la carpeta y guardándola en la variable “O”.

```
27 -    OR=imcrop(O,[534.5 232.5 590 300]);
28 -    figure,subplot(2,2,1), imshow(O),title('cinta');
29 -           subplot(2,2,2), imshow(OR),title('cinta Recortada');
30
```

Mediante el comando imcrop recortamos esa imagen para quedarnos solo con la parte que nos interesa y lo almacenamos en “OR”.

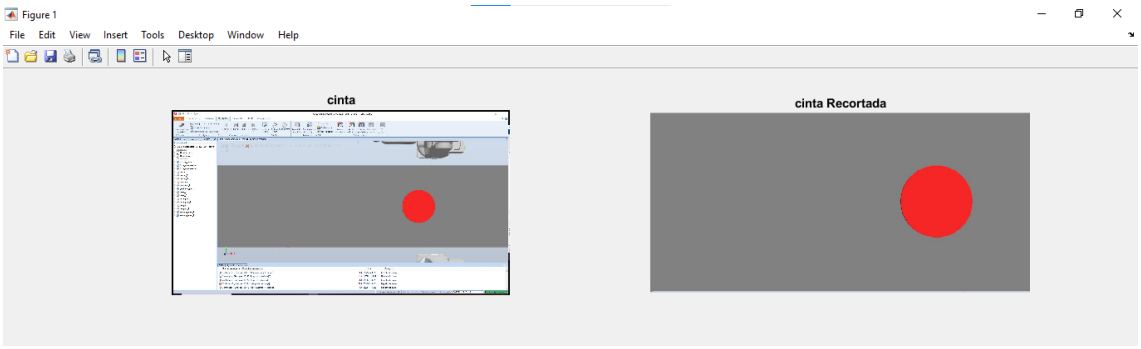

Figura 57 Figura de Matlab donde se muestra la imagen que nos ha dado RobotStudio, y la Imagen recortada

```
31      %seccionar por color
32
33
34      %cinta
35
36 -    numColors = 2;
37
38 -    L = imsegkmeans(OR,numColors);
39 -    B = labeloverlay(OR,L);
40 -    lab_OR = rgb2lab(OR);
41 -    ab = lab_OR(:,:,2:3);
42 -    ab = im2single(ab);
43 -    pixel_labels = imsegkmeans(ab,numColors);
44 -    B2 = labeloverlay(OR,pixel_labels);
45 -    figure,subplot(2,2,1), imshow(B),title('Cinta por regiones');
46 -           subplot(2,2,2), imshow(B2),title('Cinta');
```

Ahora, mediante el comando imsegkmeans vamos a coger la imagen recortada y dividirla en regiones según el color, que nosotros previamente hemos limitado a dos porque nuestra imagen es una pieza de color sobre un fondo gris que es la cinta. A cada región se le asigna una etiqueta y con el comando laveloverlay lo que hacemos es fusionar la imagen de entrada, OR, con un color diferente para cada etiqueta distinta de cero en la matriz de etiquetas L.

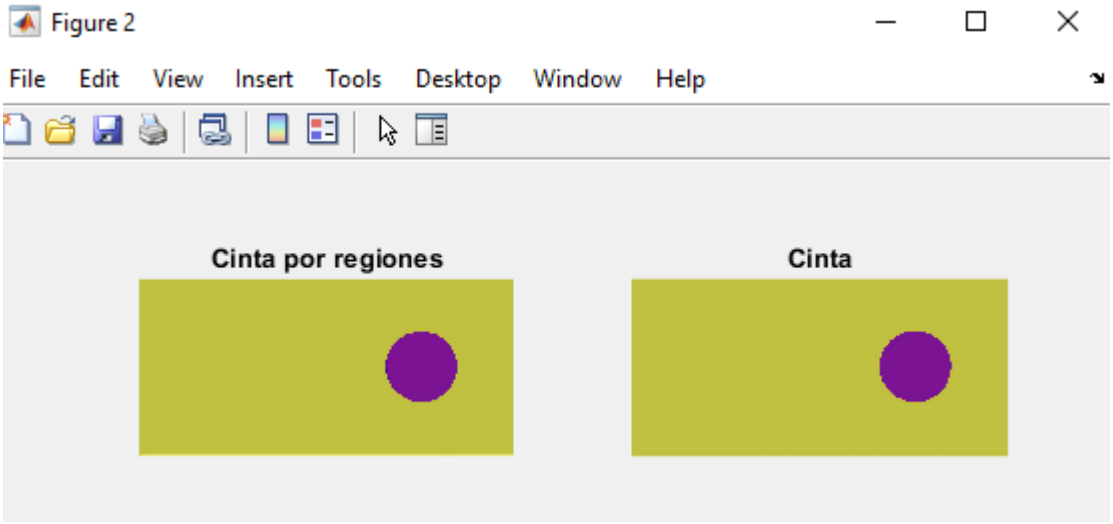

Figura 58 Figura de Matlab donde se aprecia la imagen dividida en regiones

```
47 -        pieza=B2;
48          %%Si la imagen no está en escala de grises, se convierte
49 -        if ndims(pieza)==3
50 -           pieza=rgb2gray(pieza);
51 -        else pieza=pieza;
52 -        end
```

Si la pieza no está en escala de grises, la convertimos con “rgb2gray”.

```
54 -        A =~ imbinarize(pieza, graythresh(pieza)); % binarización
55 -        A = bwareaopen( A , 500 );
56 -        dd=strel('disk',7,4);%hacemos la imagen mas uniforme
57 -        C=imclose(A,dd);
58 -        J = imcomplement(C);
```

Una vez en escala de grises binarizamos la imagen con “imbinarice”, para después hacer más uniforme la imagen con “bwareaopen”, “strel” y “imclose”. Con “imcomplement” invertimos los colores de nuestra imagen binarizada.

```
59 -        objetos=bwconncomp(C,8);%cuenta num objetos
60 -        datos=regionprops(objetos,'All');%localiza datos
61 -        fprintf('datos del objeto de origen');
62 -        Z = imbinarize(pieza, graythresh(pieza));
63
```

Ahora con “bwconncomp” contamos el número de objetos de la imagen, y con “regionprops” guardamos los datos de cada objeto en la variable “datos”.

```
64 -        numobjetos=objetos.NumObjects;
65 -        centroids = cat(1, datos.Centroid);
66 -        perimetro=datos.Perimeter;
67 -        circularidad=datos.Circularity;
68 -        orientacion=datos.Orientation;
69 -        area=datos.Area;
```

Guardamos el centroide, el perímetro, la circularidad la orientación y el área en variables para poder usarlo más tarde.

```
71 -      figure,subplot(3,3,1), imshow(pieza),title('Cinta gris');
72 -          subplot(3,3,2), imshow(A),title('Cinta blanco y negro');
73 -          subplot(3,3,3), imshow(C),title('Cinta blanco y negro uniforme');
74 -          subplot(3,3,4), imshow(J),title('Cinta complementada');
75 -          hold on;
```

Mostraremos ahora las imágenes.

```
77 - rectangle('Position',[datos.BoundingBox(1),datos.BoundingBox(2),datos.BoundingBox(3),datos.BoundingBox(4)],'EdgeColor','r','LineWidth',2 );
78 - plot(datos(1).Centroid(1),datos(1).Centroid(2),'-+r');
79 - fprintf(' centroide: %d \n,  %e \n orientacion: %f \n perimetro: %f \n',datos.Centroid(1),datos.Centroid(2),datos.Orientation,datos.Perimeter);
80 - Template_cinta=imcrop(C,[datos(1).BoundingBox(1) datos(1).BoundingBox(2) datos(1).BoundingBox(3) datos(1).BoundingBox(4)]);
81
82 -      subplot(3,3,5), imshow(Template_cinta),title('template');
```

En la última mostraremos nuestro objeto recuadrado y con el centroide marcado en rojo.

Una vez tenemos delimitado esa parte de la imagen la recortaremos para ver mejor nuestro objeto.

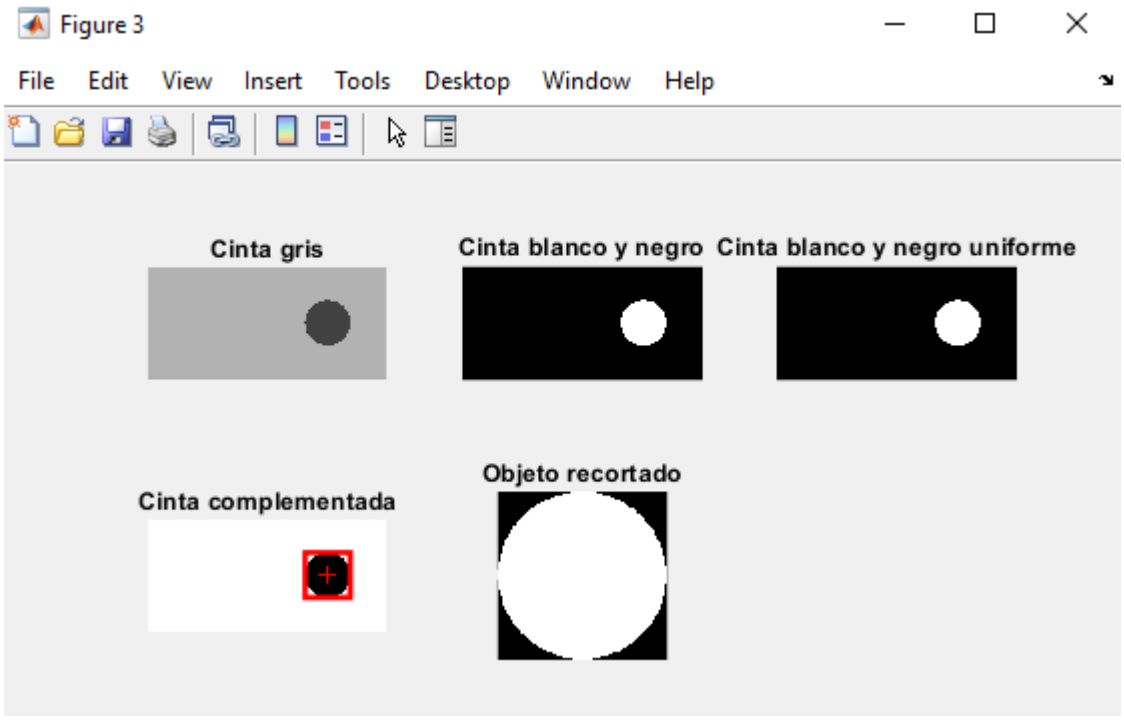

Figura 59 Figura de Matlab donde se aprecia la imagen en grises, la cinta en blanco y negro, la cinta en blanco y negro uniformada, la cinta complementada y el objeto recortado

Una vez tenemos la imagen de la figura y los datos procedemos a analizarlos para saber que figura es y que debemos enviar al robot.

Para saber que figura es nos bastará con estudiar el perímetro, la circularidad y el área.

```
84 -      if perimetro>540 && perimetro<555 && circularidad<1 && datos.Area>17900 && datos.Area<18100
85            %rectangulo
86 -          n_pieza=4;
87 -          fprintf('rect');
88
```

```
95 -        elseif perimetro>=550 && perimetro<570 && circularidad<1 && datos.Area>15700 && datos.Area<15800
96              %triangulo
97 -            n_pieza=2;
98 -            fprintf('tri');
```

```
106 -           elseif perimetro>=580 && perimetro<595 && circularidad>1
107                 %circulo
108 -               n_pieza=1;
109 -               fprintf('circ');
```

```
111 -           elseif perimetro>=720 && perimetro<735 && circularidad<1 && datos.Area>35000
112                 %cuadrado
113 -               fprintf('cuad');
114 -               n_pieza=3;
```

Ahora que ya sabemos que pieza es, sus características y su posición lo mandamos a Rapid para que siga la ejecución del proceso.

```
125 -   x(:,1)=[datos.Centroid(1)*1000/1382,datos.Centroid(2)*500/744,cinta_orientacion,n_pieza];
126     % y(:,1)=[0,0,0,datos.Centroid(2)*500/744];
127     % ori(:,1)=[0,0,0,cinta_orientacion];
128 -   write(itm2,x)
129     % write(itm3,y)
130     % write(itm4,ori)
131 -   paso=2;
132 -   write(itm1,paso)
133 -       end
134 -   end
135
136 -   disconnect(da);
137 -   delete(da);
```

Guardamos en el array “x” los datos necesarios y lo mandamos a Rapid con la variable “itm2”, para después cerrar el semáforo que tenemos con la variable “paso”, que también enviamos a Rapid para que siga el proceso en su lado.

Si finalmente se termina el proceso procedemos a cerrar la comunicación.

9. Conclusiones

La simulación desarrollada en este trabajo de fin de grado es el resultado de la combinación de los conocimientos adquiridos en las asignaturas de Visión Artificial, Robótica y Comunicación Industrial y su aplicación práctica. La experiencia adquirida durante la realización de este trabajo no solo ha reforzado las bases sino que también ha proporcionado una visión práctica y compleja de la combinación de diferentes sistemas así como de los retos y oportunidades presentes en el ámbito de la automatización avanzada.

Además, este proyecto ha puesto de manifiesto la importancia de una buena planificación y diseño de sistemas de automatización, en los que la coordinación entre diferentes tecnologías (visión, control, robótica) es clave para dar respuesta a los problemas complejos en la industria.

La simulación desarrollada en este trabajo de fin de grado ha permitido integrar conocimientos teóricos y prácticos en un entorno funcional, en el que la visión artificial y la robótica trabajan de forma conjunta para lograr la clasificación y manipulación de piezas.

A lo largo de este proyecto se ha comprobado y mostrado la viabilidad de aplicar algoritmos de análisis de imagen mediante MATLAB en un flujo automatizado, así como la efectividad de la comunicación con la plataforma RobotStudio para la ejecución de movimientos en función del resultado del análisis visual.

Este proyecto sienta una base sólida para futuras investigaciones o desarrollos más complejos, e invita a seguir explorando la aplicación de la inteligencia artificial en entornos industriales reales.

Este trabajo ejemplifica de manera sencilla, visual y dinámica de numerosos sistemas presentes en las industrias de hoy en día en fábricas multinacionales y tan punteras como Renault en la que aplican una tecnología similar para la clasificación de piezas en las cadenas de montaje.

Al tratarse de un ejemplo sencillo también podría aplicarse en el ámbito educativo para explicar el funcionamiento de sistemas complejos presentes en la industria a personas cuyo futuro vaya a implicar la participación en cualquiera de sus formas (diseño y programación, utilización, mantenimiento, etc.) en estos complejos sistemas.

10. Referencias y bibliografía

ABB. (2022). *RobotStudio® User Manual*. ABB Robotics. https://new.abb.com/products/robotics/es/robotstudio (03-02-2025)

ABB. (2023a). *RobotStudio and RobotWare 6.15 Documentation*. ABB Robotics. https://new.abb.com/products/robotics/es (03-02-2025)

ABB. (2023b). *RobotStudio PowerPacs and Add-ins*. ABB Robotics. https://developercenter.robotstudio.com/ (07-02-2025)

Barrientos, A., Peñín, L. F., Balaguer, C., & Aracil, R. (2007). *Fundamentos de robótica* (2.ª ed.). McGraw-Hill.

EDS Robotics. (2020). *Aplicaciones de visión artificial en la industria*. https://edsrobotics.com/vision-artificial (20-03-2025)

Elecproy. (2024). *Clasificación de sistemas de visión artificial industrial*. https://elecproy.com/blog/vision-artificial-industrial (20-03-2025)

Groover, M. P. (2016). *Automatización, producción y manufactura moderna* (4.ª ed.). Pearson Educación. (20-03-2025)

MathWorks. (2023). *Image Processing Toolbox™ User's Guide*. https://www.mathworks.com/help/images/ (13-03-2025)

Moreno, J., & Hernández, A. (2021). *Redes industriales y protocolos de comunicación en automatización*. Alfaomega.

Petruzzi, F., & Antonelli, G. (2022). *Industrial Communication Networks and Fieldbuses*. Springer. (20-03-2025)

Rapid Innovation. (2024). *La visión artificial en la industria 4.0: tendencias y oportunidades*. https://www.rapidinnovation.io/blog/vision-industrial (20-03-2025)

Ultralytics. (2024). *YOLOv5 Documentation*. https://docs.ultralytics.com/ (20-03-2025)

Wikipedia. (2025). *Visión por computadora*. Wikipedia, La enciclopedia libre. https://es.wikipedia.org/wiki/Visión_por_computadora (20-03-2025)