Tema 7. Programación y Control de Sistemas Robóticos

Bienvenido/a, en este tema abordarás uno de los pilares fundamentales de la robótica moderna: la **programación y el control de sistemas robóticos**. Comprenderás cómo el software da vida a los robots, cómo se estructuran los algoritmos que gobiernan su comportamiento y qué herramientas profesionales se utilizan en la industria actual. A lo largo del tema, trabajarás con conceptos teóricos sólidos, ejemplos prácticos detallados y casos de aplicación real que te prepararán para ejercer con competencia en entornos robóticos reales.

01

Concepto y fundamentos

Qué es la programación robótica y su relación con el movimiento del robot.

02

Lenguajes y algoritmos

Tipos de lenguajes, estructuras algorítmicas: secuencias, condicionales y bucles.

03

Control y sensores

Control de movimiento, trayectorias, velocidad y la interacción con sensores.

04

Rol profesional y práctica

Casos prácticos desarrollados paso a paso, herramientas digitales y autoevaluación.

Concepto de Programación Robótica

La programación robótica es el proceso mediante el cual un profesional define, de forma precisa y estructurada, el conjunto de instrucciones que un robot debe ejecutar para llevar a cabo una tarea determinada. No se trata simplemente de escribir código: programar un robot implica comprender profundamente la mecánica del sistema, los actuadores que generan el movimiento, los sensores que recogen información del entorno, y la lógica que conecta todos estos elementos en un comportamiento coherente y seguro.

Desde sus orígenes, la robótica dependió de la programación para trasladar la intención humana a movimientos físicos. En los primeros robots industriales de los años 60 y 70, la programación se realizaba mediante paneles de control físicos y programación por enseñanza directa (*teach pendant*), en la que el operario movía manualmente el robot hasta las posiciones deseadas y el sistema las almacenaba en memoria. Con el tiempo, los lenguajes de programación especializados permitieron abstraer esa programación y escribir secuencias complejas desde un ordenador, sin necesidad de manipular el robot directamente.

Hoy en día, la programación robótica abarca desde entornos gráficos accesibles —utilizados en robótica educativa y en robots colaborativos— hasta lenguajes de alto nivel orientados a objetos que se emplean en aplicaciones de inteligencia artificial aplicada a la robótica. Sea cual sea el nivel de abstracción, el principio fundamental es el mismo: el robot solo actúa según lo que se le ha programado, y la calidad del programa determina directamente la calidad y seguridad del comportamiento robótico.

Importancia de la programación en robótica

La programación es el alma del robot. Sin ella, el hardware —por muy sofisticado que sea— es inerte. A través del software, el profesional en robótica convierte un conjunto de motores, engranajes y sensores en un sistema autónomo capaz de realizar tareas repetitivas con una precisión y velocidad que supera la capacidad humana. En la industria manufacturera, por ejemplo, los robots soldadores o de ensamblaje pueden repetir el mismo movimiento miles de veces con una tolerancia de décimas de milímetro, gracias a programas cuidadosamente elaborados y verificados.

Más allá de la repetibilidad, la programación permite la adaptabilidad. Los robots modernos pueden modificar su comportamiento en tiempo real en función de las señales que reciben de sus sensores, lo que les permite operar en entornos dinámicos e impredecibles. Esta capacidad de respuesta inteligente —programada— es la base de conceptos como la robótica colaborativa, la robótica de servicio y los sistemas autónomos.

Relación entre software y movimiento del robot

El movimiento de un robot es el resultado directo de las instrucciones enviadas por el software al sistema de control, que a su vez genera señales eléctricas hacia los actuadores —motores, servomotores, cilindros neumáticos o hidráulicos—. En un robot de brazo articulado, por ejemplo, cada articulación posee un motor con encoder que informa continuamente de su posición angular. El software lee esa posición, la compara con la posición objetivo programada y envía la señal correctiva necesaria para alcanzarla. Este proceso ocurre cientos de veces por segundo en un bucle de control denominado **lazo cerrado** o *closed-loop control*.

> **Concepto clave:** El **lazo cerrado de control** es el mecanismo fundamental que permite al robot corregir desviaciones en tiempo real, garantizando precisión y seguridad en sus movimientos. Comprenderlo es esencial para cualquier profesional en robótica.

La programación también define parámetros cinemáticos como la velocidad máxima de cada articulación, la aceleración y deceleración (para proteger tanto el robot como la pieza trabajada), y la interpolación del movimiento entre puntos —lineal, circular o por puntos intermedios—. Todos estos parámetros quedan codificados en el programa y pueden ajustarse según las necesidades de cada proceso productivo.

Lenguajes de Programación en Robótica

Para programar un robot existen múltiples aproximaciones, que se pueden clasificar en tres grandes categorías: los **lenguajes gráficos**, los **lenguajes basados en texto** y los **lenguajes específicos para robots**. Conocer las diferencias entre ellos, sus ventajas e inconvenientes y sus contextos de aplicación es fundamental para que puedas elegir la herramienta adecuada según el tipo de robot y la tarea a programar.

Lenguajes gráficos

Los lenguajes gráficos, también denominados lenguajes de programación por bloques o por diagrama de flujo, representan las instrucciones como elementos visuales que se conectan entre sí arrastrándolos sobre una pantalla. El ejemplo más conocido en el ámbito educativo es **Scratch**, desarrollado por el MIT, aunque existen versiones derivadas orientadas específicamente a la robótica educativa como **Scratch for Arduino**, **mBlock** (para robots Makeblock), o el entorno de programación de **LEGO MINDSTORMS EV3**.

La principal ventaja de los lenguajes gráficos es su accesibilidad: permiten que personas sin experiencia en programación textual puedan crear programas funcionales de forma intuitiva, enfocándose en la lógica del comportamiento sin tener que preocuparse por la sintaxis. Son ideales para la enseñanza de conceptos algorítmicos básicos y para la fase de prototipado rápido. Sin embargo, presentan limitaciones cuando se requieren comportamientos complejos, ya que la representación visual puede volverse confusa y difícil de mantener en proyectos grandes.

Lenguajes basados en texto

Los lenguajes de programación textuales clásicos también se utilizan ampliamente en robótica. **Python** es actualmente uno de los más populares en robótica de investigación y desarrollo, gracias a su sintaxis clara y a las abundantes librerías disponibles, como ROS (Robot Operating System) con Python, o PyRobot. **C y C++** siguen siendo los lenguajes de referencia para aplicaciones que requieren máximo control sobre el hardware y tiempos de respuesta en tiempo real, especialmente en microcontroladores como Arduino o en sistemas embebidos industriales.

Estos lenguajes ofrecen gran flexibilidad y potencia, pero exigen al programador un mayor dominio técnico. La gestión de la memoria, el control de interrupciones y la sincronización de procesos son aspectos críticos que debes conocer al trabajar con C o C++ en entornos robóticos de producción.

Lenguajes específicos para robots

Los fabricantes de robots industriales han desarrollado sus propios lenguajes de programación, optimizados para sus arquitecturas y sistemas de control. Estos lenguajes específicos integran de forma nativa los conceptos propios de la robótica —puntos de posición, movimientos interpolados, zonas de trabajo, señales de entrada/salida— facilitando la programación sin necesidad de gestionar el hardware a bajo nivel.

RAPID (ABB)

Lenguaje de ABB Robotics. Sintaxis estructurada con módulos, procedimientos y funciones. Muy utilizado en la industria automovilística.

KRL (KUKA)

KUKA Robot Language. Basado en estructuras similares a Pascal. Permite control preciso de movimientos y señales E/S del robot KUKA.

Karel (FANUC)

Lenguaje de FANUC, derivado de Pascal. Ofrece control total de la lógica del robot y se integra con el entorno ROBOGUIDE.

URScript (Universal Robots)

Lenguaje script de los cobots UR. Sintaxis sencilla orientada a la programación de robots colaborativos en entornos flexibles.

Como profesional en robótica, deberás familiarizarte con al menos uno de estos lenguajes específicos, ya que son los que encontrarás en las instalaciones industriales reales. La documentación técnica del fabricante y los entornos de simulación como RobotStudio o ROBOGUIDE son tus aliados para aprender y practicar antes de trabajar sobre el robot físico.

Algoritmos Aplicados a Robots

Un algoritmo es una secuencia finita, ordenada y precisa de instrucciones que describe cómo resolver un problema o realizar una tarea. En robótica, los algoritmos son el esqueleto de cualquier programa: definen exactamente qué debe hacer el robot, en qué orden, bajo qué condiciones y cuántas veces. Comprender las estructuras algorítmicas fundamentales es imprescindible para que puedas desarrollar programas robóticos eficientes, claros y libres de errores.

Secuencias de instrucciones

La estructura algorítmica más básica es la **secuencia**: un conjunto de instrucciones que se ejecutan una tras otra, en el orden en que están escritas. Por ejemplo, un programa de movimiento lineal podría secuenciar: mover a posición A → cerrar pinza → mover a posición B → abrir pinza → volver a posición inicial. Cada instrucción se ejecuta completamente antes de pasar a la siguiente. La claridad y el orden en la secuencia son críticos: un error en la disposición de las instrucciones puede causar colisiones, daños al robot o a la pieza trabajada.

Condicionales

Las estructuras **condicionales** permiten al robot tomar decisiones en función del estado del sistema o de las señales recibidas. La instrucción *IF-THEN-ELSE* es el ejemplo paradigmático: "SI el sensor de proximidad detecta un objeto ENTONCES ejecutar la rutina de recogida, DE LO CONTRARIO continuar el movimiento de búsqueda". Las estructuras condicionales son fundamentales para dotar al robot de comportamiento adaptativo, ya que le permiten responder de forma diferente ante distintas situaciones del entorno.

Bucles de repetición

Los **bucles** o estructuras de repetición permiten ejecutar un bloque de instrucciones múltiples veces sin necesidad de reescribirlo. Existen principalmente dos tipos: los bucles con contador determinado (*FOR* o *REPEAT N TIMES*), que repiten la instrucción un número fijo de veces, y los bucles condicionales (*WHILE* o *UNTIL*), que repiten la instrucción mientras se cumpla —o hasta que se cumpla— una condición determinada. En robótica industrial, los bucles son omnipresentes: un robot de ensamblaje que trabaja en una línea de producción ejecuta continuamente el mismo ciclo de operaciones en un bucle que solo se interrumpe por una señal de parada o por una condición de error.

Este esquema visual resume las tres estructuras algorítmicas fundamentales que gobiernan el comportamiento de cualquier robot programable. Dominar su combinación te permitirá crear programas robóticos capaces de enfrentarse a situaciones complejas y variables del entorno industrial.

Ejemplo visual: Algoritmo de movimiento

Imagina un robot que debe recoger piezas de una cinta transportadora y depositarlas en una caja. El algoritmo combinaría las tres estructuras: un **bucle** exterior que repite el ciclo mientras haya piezas disponibles, una **condicional** que verifica si el sensor de presencia detecta la pieza antes de activar la pinza, y una **secuencia** que ordena los movimientos de aproximación, agarre, transporte y depósito. Esta combinación estructurada es el corazón de cualquier programa robótico funcional en entorno industrial.

Control de Movimiento Robótico

El control de movimiento es la disciplina que se ocupa de cómo el robot desplaza sus articulaciones y efector final desde una posición hasta otra, siguiendo criterios de precisión, velocidad y seguridad. No basta con indicar al robot "ve al punto B": debes especificar **cómo** debe llegar, a qué velocidad, con qué aceleración y siguiendo qué trayectoria. Cada uno de estos parámetros tiene un impacto directo en la calidad del proceso productivo y en la vida útil del robot.

Trayectorias

Una trayectoria define el camino que sigue el extremo del robot —la herramienta o efector— en el espacio tridimensional. Existen varias modalidades de interpolación de trayectoria que debes conocer como profesional:

Movimiento punto a punto (PTP)

El robot se desplaza de un punto a otro por la trayectoria más rápida posible para cada articulación, sin garantizar el camino del efector final. Se usa cuando la trayectoria no es crítica, solo la posición de llegada. Ideal para manipulación de piezas.

Interpolación lineal (LIN)

El efector final describe una línea recta entre el punto de partida y el de llegada. Esencial en operaciones de soldadura, corte o dispensado de adhesivo, donde la trayectoria del extremo debe ser exactamente recta.

Interpolación circular (CIRC)

El efector final describe un arco de circunferencia pasando por tres puntos definidos: inicio, punto intermedio y final. Se utiliza en operaciones de soldadura circular, mecanizado de perfiles curvos o movimientos de seguimiento de contornos.

Velocidad

La velocidad de movimiento en un robot se puede especificar en unidades de velocidad lineal del efector (mm/s o m/s) para movimientos interpolados, o en porcentaje de la velocidad máxima de cada articulación para movimientos PTP. Una velocidad demasiado alta puede provocar vibraciones, imprecisiones en los puntos de llegada, desgaste prematuro del robot y riesgo de colisión si hay obstáculos no previstos. Una velocidad demasiado baja reduce la productividad del sistema. El ajuste óptimo de la velocidad es siempre fruto del equilibrio entre productividad, calidad y seguridad.

En sistemas modernos, los programas robóticos incluyen perfiles de velocidad con rampas de aceleración y deceleración (trapezoidal o tipo S-curve) que suavizan los cambios bruscos de velocidad, protegen los mecanismos del robot y mejoran la calidad del proceso. Como profesional, deberás ser capaz de ajustar estos perfiles según las especificaciones de cada aplicación.

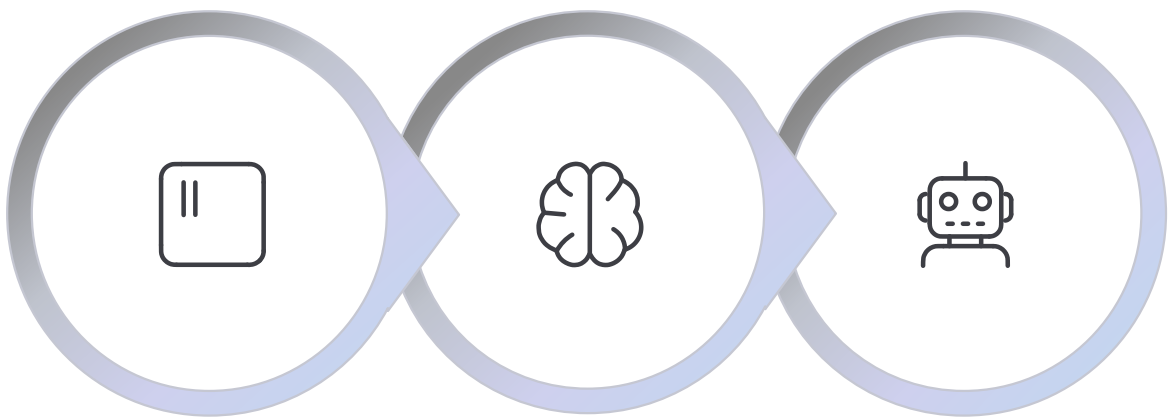
Precisión de movimientos

La **precisión** o exactitud posicional es la capacidad del robot de alcanzar con exactitud la posición programada. La **repetibilidad**, por su parte, es la capacidad de volver a esa posición de forma consistente tras múltiples ciclos. En robots industriales modernos, la repetibilidad puede ser de ±0,02 mm o inferior. Esta precisión depende de la calidad mecánica del robot, de los encoders de los motores y del algoritmo de control implementado en el software.

En la programación, la precisión también se controla mediante el parámetro de **zona de parada** (zone o blend): si se programa una zona de parada exacta, el robot detiene completamente el movimiento antes de iniciar el siguiente; si se programa una zona de paso o zona de mezcla, el robot suaviza el giro en el punto intermedio sin detenerse, lo que aumenta la velocidad del ciclo a costa de una pequeña desviación de trayectoria en ese punto.

Interacción entre Sensores y Programación

La integración de sensores en los sistemas robóticos transforma al robot de una máquina de movimientos predefinidos en un sistema capaz de percibir su entorno y reaccionar de forma inteligente. Esta capacidad de **percepción-decisión-actuación** es el fundamento de la robótica adaptativa y autónoma, y su correcta implementación depende directamente de cómo se programa la interacción entre los datos del sensor y la lógica de control.



Este ciclo de tres fases —lectura, decisión y ejecución— se repite continuamente durante el funcionamiento del robot, permitiéndole responder en tiempo real a los cambios en su entorno de trabajo.

Lectura de sensores

El primer paso en la interacción sensor-programa es la **lectura** del valor del sensor. Dependiendo del tipo de sensor, esta lectura puede ser digital (el sensor devuelve 1 si detecta presencia o 0 si no detecta) o analógica (el sensor devuelve un valor numérico continuo que representa, por ejemplo, la distancia en milímetros o la intensidad de luz). En la programación, se asigna a cada sensor una variable o dirección de entrada, y el programa lee periódicamente ese valor o reacciona a un cambio mediante una interrupción.

Los sensores más comunes en robótica industrial incluyen: **sensores de proximidad inductivos y capacitivos** (detectan presencia de objetos metálicos o materiales en general), **sensores de visión artificial** (cámaras que identifican posición, orientación y características de las piezas), **sensores de fuerza/par** (miden las fuerzas ejercidas en la herramienta, fundamentales en robótica colaborativa) y **sensores ultrasónicos y LiDAR** (miden distancias para navegación en robots móviles).

Toma de decisiones

Una vez leído el valor del sensor, el programa evalúa ese valor mediante estructuras condicionales y toma una decisión. Por ejemplo: "SI la distancia al objeto es inferior a 50 mm ENTONCES reducir velocidad a 50 mm/s; SI la distancia es inferior a 10 mm ENTONCES detener el movimiento y activar la pinza". La calidad del programa de decisión determina la robustez del sistema: un programa bien diseñado contemplará todos los estados posibles del sensor —incluidos estados de error, señales fuera de rango y situaciones de emergencia— y definirá una respuesta apropiada para cada uno.

Ejecución de acciones

Tras la decisión, el robot ejecuta la acción programada: un movimiento, la activación de una salida digital (para controlar una pinza neumática, una ventosa, un indicador luminoso), la modificación de un parámetro de velocidad o la llamada a una subrutina de tratamiento especial. La ejecución debe ser rápida y segura, y el programa debe garantizar que el robot no intentará ejecutar acciones incompatibles entre sí o peligrosas para el entorno.

Ejemplo práctico: Programación de respuesta ante detección de objeto

Considera un robot de paletizado equipado con un sensor de proximidad en la pinza. El programa incluye el siguiente comportamiento: el robot se desplaza hacia la posición de recogida; cuando el sensor detecta la pieza a menos de 20 mm, reduce la velocidad al 20% y activa el modo de control de fuerza; cuando la pinza contacta con la pieza (confirmado por el sensor de fuerza), se activa el vacío de la ventosa; el programa espera la confirmación de la señal de vacío conseguido antes de iniciar el movimiento de elevación. Si la señal de vacío no se recibe en 2 segundos, el robot detiene el ciclo y emite una alarma. Este ejemplo ilustra cómo la integración de múltiples sensores y la lógica de programación adecuada crea un proceso robusto y seguro en la realidad industrial.

Rol Profesional en Programación Robótica

Como profesional en robótica, tu función no se limita a escribir código. Tu rol abarca un conjunto de competencias técnicas y organizativas que van desde la comprensión profunda de los algoritmos hasta la supervisión continua del funcionamiento del sistema en producción. Esta visión integral es la que distingue a un profesional cualificado de un simple operario.

Interpretación de algoritmos

Antes de poder programar o modificar un sistema robótico, debes ser capaz de **leer e interpretar algoritmos** existentes. En la práctica industrial, frecuentemente te encontrarás trabajando con programas escritos por otros técnicos o heredados de instalaciones anteriores. La capacidad de seguir la lógica de un programa, identificar qué hace cada bloque de instrucciones, detectar posibles errores o ineficiencias y proponer mejoras es una competencia crítica. Para desarrollarla, es esencial practicar regularmente con la lectura de código, los diagramas de flujo y la documentación técnica de los robots.

La interpretación de algoritmos también incluye la capacidad de traducir las especificaciones del proceso —proporcionadas por el departamento de producción o ingeniería— en un programa robótico concreto. Esta traducción requiere entender tanto el lenguaje técnico del proceso productivo como el lenguaje de programación del robot, actuando como puente entre ambos mundos.

Configuración de sistemas robóticos

La configuración de un sistema robótico comprende todas las tareas previas a la ejecución de un programa: la **calibración del robot** (definición del sistema de coordenadas de la herramienta —TCP— y del sistema de coordenadas del objeto de trabajo), la configuración de las entradas y salidas digitales para los periféricos (pinzas, sensores, señalizaciones), la definición de las zonas de seguridad y la carga del programa en el controlador. Una configuración incorrecta puede causar que el robot trabaje en coordenadas erróneas, lo que se traduce en movimientos imprecisos, colisiones o daños a los equipos.

Supervisión del funcionamiento

Una vez que el sistema está en producción, el profesional en robótica debe realizar una **supervisión continua** del funcionamiento: verificar que los ciclos de trabajo se ejecutan correctamente, monitorizar los registros de alarmas y errores, analizar los datos de rendimiento (tiempos de ciclo, tasa de errores, consumo energético) y actuar rápidamente ante cualquier anomalía. Los sistemas robóticos modernos ofrecen interfaces de monitorización remota que permiten supervisar múltiples robots desde un único puesto, facilitando la gestión eficiente de instalaciones complejas.

Interpretar

Leer, analizar y comprender algoritmos y programas robóticos existentes.

Configurar

Calibrar, parametrizar y poner a punto el sistema antes de su operación.

Supervisar

Monitorizar el funcionamiento, detectar anomalías y optimizar el rendimiento.

Mantener

Actualizar programas, corregir errores y adaptar el sistema a nuevas tareas.

CASO PRÁCTICO 1

Programación de Movimiento Básico de un Robot

A continuación se desarrolla de forma completa y detallada el proceso de programación de un movimiento básico en un robot industrial de brazo articulado. Este caso práctico te mostrará paso a paso cómo trasladar una especificación de tarea real a un programa ejecutable, desde la definición del movimiento hasta la verificación en el controlador. El robot utilizado como referencia es un brazo articulado de 6 ejes con controlador compatible con lenguaje RAPID (ABB), aunque los conceptos son extrapolables a cualquier plataforma.

Contexto de la tarea

El robot debe recoger una pieza metálica de una posición fija (Posición A) y depositarla en una bandeja de salida (Posición B), ambas posiciones conocidas y fijas en el espacio de trabajo. La tarea requiere que el robot se aproxime verticalmente a la pieza para evitar colisiones laterales, cierre la pinza neumática sobre ella, la eleve, se desplace hasta Posición B y la deposite de forma suave.

1 Paso 1 — Definir el movimiento

El primer paso consiste en **definir con precisión todas las posiciones** que el robot debe alcanzar durante la tarea. Estas posiciones se denominan *targets* o *robtargets* en RAPID. Para este caso, necesitarás definir como mínimo los siguientes puntos:

HomePos: Posición de reposo del robot, generalmente en el centro del espacio de trabajo, con el brazo en posición segura, alejado de todos los obstáculos. El robot debe empezar y terminar siempre en esta posición.

ApproachA: Posición de aproximación sobre Posición A, a una altura de 100 mm por encima de la pieza. Esta posición intermedia evita colisiones durante el desplazamiento hacia la pieza.

PickA: Posición exacta de recogida, donde el efector (pinza) entra en contacto con la pieza. Se define con la herramienta orientada verticalmente hacia abajo.

ApproachB: Posición de aproximación sobre Posición B, a 100 mm de altura sobre la bandeja de depósito.

PlaceB: Posición exacta de depósito sobre la bandeja de salida.

La definición de estas posiciones se realiza mediante programación por enseñanza (*teach-in*): con el robot en modo manual, el operario desplaza el brazo hasta cada posición con el mando de control (*teach pendant*) y registra las coordenadas. Alternativamente, en entornos virtuales como RobotStudio, se pueden introducir las coordenadas numéricamente o mediante la interfaz 3D.

2 Paso 2 — Crear el algoritmo

Con las posiciones definidas, el siguiente paso es **estructurar la lógica del programa**. El algoritmo debe secuenciar correctamente todos los movimientos y acciones del proceso. En lenguaje RAPID, el programa principal (procedimiento `PROC main()`) contiene la secuencia de instrucciones en el orden correcto:

1. Mover el robot a HomePos en movimiento PTP (instrucción `MoveJ`).
2. Mover el robot a ApproachA en movimiento lineal a velocidad media (`MoveL`).
3. Mover el robot a PickA en movimiento lineal a velocidad baja (aproximación lenta para precisión).
4. Esperar 0.3 segundos para estabilización mecánica (`WaitTime`).
5. Activar la pinza cerrando la salida digital DO_Pinza (`Set DO_Pinza`).
6. Esperar la confirmación de la señal de pinza cerrada desde la entrada digital DI_PinzaCerrada (`WaitDI DI_PinzaCerrada, 1`).
7. Elevar el robot a ApproachA en movimiento lineal.
8. Mover el robot a ApproachB en movimiento PTP a velocidad alta.
9. Bajar a PlaceB en movimiento lineal a velocidad baja.
10. Esperar 0.3 segundos.
11. Abrir la pinza (`Reset DO_Pinza`).
12. Elevar el robot a ApproachB.
13. Regresar a HomePos.

Este algoritmo incluye señales de espera de confirmación (puntos 6 y 10), lo que garantiza que el robot no continúa hasta verificar que la acción anterior se ha completado correctamente. Esta práctica es fundamental para la seguridad y robustez del proceso.

3 Paso 3 — Ejecutar el programa

Antes de ejecutar el programa en modo automático, debes seguir un protocolo de **verificación y puesta en marcha** riguroso:

Verificación en simulación: Si dispones de RobotStudio u otro entorno de simulación, ejecuta primero el programa en el modelo virtual del robot. Verifica visualmente que todas las trayectorias son correctas, que no hay colisiones con el entorno y que la lógica de señales funciona como se espera.

Ejecución en modo manual a velocidad reducida: Con el robot real, ejecuta el programa en modo manual (T1 o T2 según la normativa de seguridad) a velocidad reducida (máximo 250 mm/s). Mantén el dispositivo de habilitación (*enabling device*) pulsado durante toda la ejecución y observa atentamente cada movimiento. Si detectas cualquier anomalía, suelta el dispositivo inmediatamente para detener el robot.

Validación de posiciones: Verifica que el robot alcanza correctamente cada posición programada, que la pinza cierra y abre en los momentos correctos, y que las señales de confirmación funcionan.

Ejecución en modo automático: Una vez validado el programa, cierra el perímetro de seguridad y ejecuta el programa en modo automático. Realiza al menos 10 ciclos completos supervisados antes de considerar el programa validado para producción. Documenta el programa y las posiciones en el sistema de gestión documental del taller.

CASO PRÁCTICO 2

Programación de Robot con Sensor de Proximidad

Este segundo caso práctico desarrolla un escenario más complejo en el que el robot debe adaptar su comportamiento en función de las señales de un sensor de proximidad. La integración de sensores en la programación robótica es uno de los aspectos más demandados en la industria actual, y su dominio es un factor diferenciador clave para el profesional en robótica. El robot utilizado es un brazo articulado de 6 ejes con sensor de proximidad inductivo instalado en la herramienta portapieza, con salida digital conectada al controlador.

Contexto de la tarea

El robot trabaja en una estación de inspección. Su tarea es recorrer una secuencia de posiciones de inspección sobre una bancada de trabajo, verificar en cada posición si hay una pieza presente mediante el sensor de proximidad, y ejecutar acciones diferenciadas según el resultado: si hay pieza, recogerla y depositarla en la bandeja de piezas conformes; si no hay pieza, continuar al siguiente punto de inspección. Al finalizar el recorrido completo, el robot regresa a la posición de inicio.

1 Paso 1 — Detectar el objeto

El primer elemento del programa es la rutina de **lectura y validación del sensor de proximidad**. En RAPID, la señal del sensor está mapeada a la entrada digital `DI_Sensor`. La lectura se realiza mediante la función `DInput(DI_Sensor)`, que devuelve 1 si el sensor detecta un objeto y 0 si no lo detecta.

Sin embargo, en la práctica industrial, las señales de los sensores pueden presentar rebotes (*bouncing*) o lecturas transitorias erróneas, especialmente al acercarse a la pieza. Para evitar falsos positivos, el programa implementa una **validación por tiempo**: la instrucción `WaitDI DI_Sensor, 1\MaxTime:=0.5` espera hasta 0,5 segundos la confirmación de la señal. Si la señal se activa y se mantiene activa durante al menos 100 ms (verificado mediante una segunda lectura tras ese tiempo), se considera que la pieza está presente. Si la señal no se confirma en 0,5 segundos, se asume ausencia de pieza y se continúa al siguiente punto.

Adicionalmente, el programa registra en una variable de tipo array (`VAR bool pieza{6}`) el estado de detección en cada uno de los 6 puntos de inspección, creando así un mapa del estado de la bancada que puede ser utilizado para estadísticas de producción o para repetir la inspección en puntos donde se produjo una detección dudosa.

2 Paso 2 — Programar la respuesta

La lógica de respuesta se implementa mediante una **estructura condicional anidada dentro de un bucle de iteración** sobre los 6 puntos de inspección. El código (en pseudocódigo RAPID) es el siguiente:

```
FOR i FROM 1 TO 6 DO
MoveJ Approach{i}, v500, fine, tool0;
MoveL Inspect{i}, v100, fine, tool0;
IF DInput(DI_Sensor) = 1 THEN
WaitTime 0.1;
IF DInput(DI_Sensor) = 1 THEN
pieza{i} := TRUE;
Set DO_Pinza;
WaitDI DI_PinzaCerrada, 1;
MoveL Approach{i}, v100, fine, tool0;
MoveJ PlaceCONFORME, v300, fine, tool0;
Reset DO_Pinza;
ENDIF
ELSE
pieza{i} := FALSE;
ENDIF
ENDFOR
```

Este programa utiliza el bucle FOR para recorrer sistemáticamente los 6 puntos de inspección. La condicional anidada doble implementa la validación por tiempo de la señal del sensor. Las variables de estado (`pieza{i}`) permiten la trazabilidad del proceso.

3 Paso 3 — Ejecutar la acción

La fase de ejecución sigue el mismo protocolo de verificación del Caso Práctico 1, con pasos adicionales específicos para sistemas con sensores:

Verificación del sensor desconectado del robot: Antes de montar el sensor en el robot, verifica su funcionamiento básico conectándolo a una fuente de alimentación y al controlador, y comprobando que la señal `DI_Sensor` cambia de estado correctamente al acercar una pieza metálica a la distancia de detección especificada en el manual del sensor (típicamente 5-10 mm para sensores inductivos estándar).

Ajuste de la posición de inspección: La altura de la posición de inspección (`Inspect{i}`) debe ajustarse para que la pieza a detectar quede exactamente dentro del rango de detección del sensor cuando el robot está en esa posición. Normalmente, la distancia entre el sensor y la superficie de la pieza debe ser del 60-80% de la distancia nominal de detección del sensor para garantizar una lectura fiable.

Simulación de estados de pieza presente y ausente: Durante la validación manual, prueba ambos estados: coloca piezas en algunas posiciones y deja otras vacías, y verifica que el robot responde correctamente en todos los casos. Prueba también situaciones intermedias (pieza ligeramente desplazada o inclinada) para comprobar la robustez del programa.

Registro y análisis de datos: Tras la validación, conecta el sistema a la interfaz de supervisión y verifica que los datos de estado de las variables `pieza{i}` se registran correctamente en el historial de producción. Esta capacidad de trazabilidad es fundamental en entornos de calidad industrial.

Cuadro Comparativo: Programación Gráfica vs. Programación Textual

Uno de los aspectos que debes dominar como profesional en robótica es saber cuándo utilizar cada tipo de lenguaje de programación. La elección entre un entorno gráfico y un lenguaje textual no es trivial: depende del tipo de robot, la complejidad de la tarea, el perfil del equipo y las exigencias de mantenimiento del programa. A continuación se presenta una comparativa detallada de ambos enfoques.

Criterio	Programación Gráfica	Programación Textual
Facilidad de aprendizaje	Alta. Intuitiva, visual, sin requisitos de sintaxis. Accesible para principiantes y no programadores.	Media-Alta. Requiere conocimiento de sintaxis y lógica de programación. Curva de aprendizaje mayor.
Capacidad de expresión	Limitada. Difícil de gestionar lógicas complejas, condicionales múltiples o gestión de datos avanzada.	Alta. Permite expresar cualquier lógica, gestionar estructuras de datos complejas y llamadas a funciones.
Velocidad de desarrollo	Rápida para tareas simples. El arrastrar bloques es más lento que escribir código para expertos.	Rápida para expertos. Posibilidad de reutilización de código mediante funciones y módulos.
Mantenimiento	Fácil para programas pequeños. Puede volverse complejo y difícil de navegar en programas grandes.	Excelente con buenas prácticas (comentarios, modularización). Escalable para proyectos grandes.
Integración con hardware	Limitada. Depende de los bloques predefinidos por el fabricante. Menor acceso directo al hardware.	Total. Permite gestionar señales E/S, interrupciones, temporizadores y registros de hardware directamente.
Ámbito de aplicación	Robótica educativa, cobots, prototipado rápido, operadores sin perfil técnico profundo.	Robótica industrial, aplicaciones de alta precisión, integración de sistemas complejos, programación avanzada.
Ejemplos	Scratch, mBlock, LEGO MINDSTORMS EV3, PolyScope (UR en modo gráfico), Blockly.	RAPID (ABB), KRL (KUKA), Karel (FANUC), URScript, Python+ROS, C/C++.
Depuración de errores	Visual e intuitiva para errores de lógica simple. Difícil en programas complejos.	Potente con herramientas de depuración, puntos de ruptura, registros de variables y trazas de ejecución.

Como conclusión práctica: en el ámbito profesional industrial, la programación textual mediante los lenguajes específicos del fabricante es el estándar. Sin embargo, el conocimiento de entornos gráficos es valioso para la comunicación con perfiles no técnicos, para la formación inicial y para el prototipado rápido de soluciones en cobots y robots de servicio. Un profesional completo domina ambos enfoques y sabe cuándo aplicar cada uno.

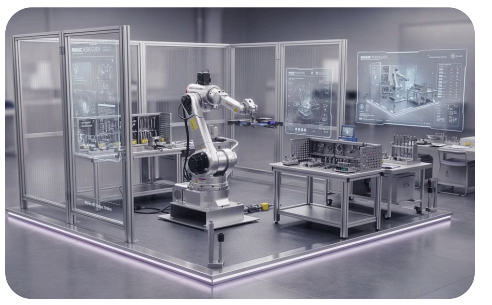
Herramientas Digitales para Programación y Simulación Robótica

El uso de herramientas digitales de simulación y programación es hoy en día una competencia fundamental para cualquier profesional en robótica. Estas plataformas te permiten diseñar, programar y verificar el comportamiento del robot en un entorno virtual antes de implementarlo en el robot real, reduciendo drásticamente los riesgos de colisión, los tiempos de puesta en marcha y los costes de desarrollo. A continuación se describen las tres herramientas más relevantes del mercado industrial.

ABB RobotStudio

Fabricante: ABB Robotics. Entorno de programación offline y simulación 3D completa para robots ABB. Permite programar en lenguaje RAPID, simular trayectorias y verificar alcanzabilidad y tiempos de ciclo. Incluye biblioteca de robots ABB, herramientas y elementos de célula estándar. La función de *AutoPath* permite generar automáticamente trayectorias sobre geometrías 3D importadas en formato CAD.

Es el estándar de facto para la programación de robots ABB en la industria y una herramienta imprescindible si trabajas con equipos de esta marca. Dispone de versión de evaluación gratuita.



FANUC ROBOGUIDE

Fabricante: FANUC Corporation. Suite de simulación para robots FANUC que abarca aplicaciones de manipulación, soldadura, pintura y mecanizado. Permite programar en Karel y TP (Teach Pendant language), simular ciclos completos y verificar colisiones. La herramienta *CycleTime Estimation* permite optimizar los tiempos de ciclo antes de la implementación real.

ROBOGUIDE es especialmente valorado en la industria automovilística y en aplicaciones de soldadura por arco, donde la precisión de la simulación es crítica para la calidad del proceso.

KUKA.Sim

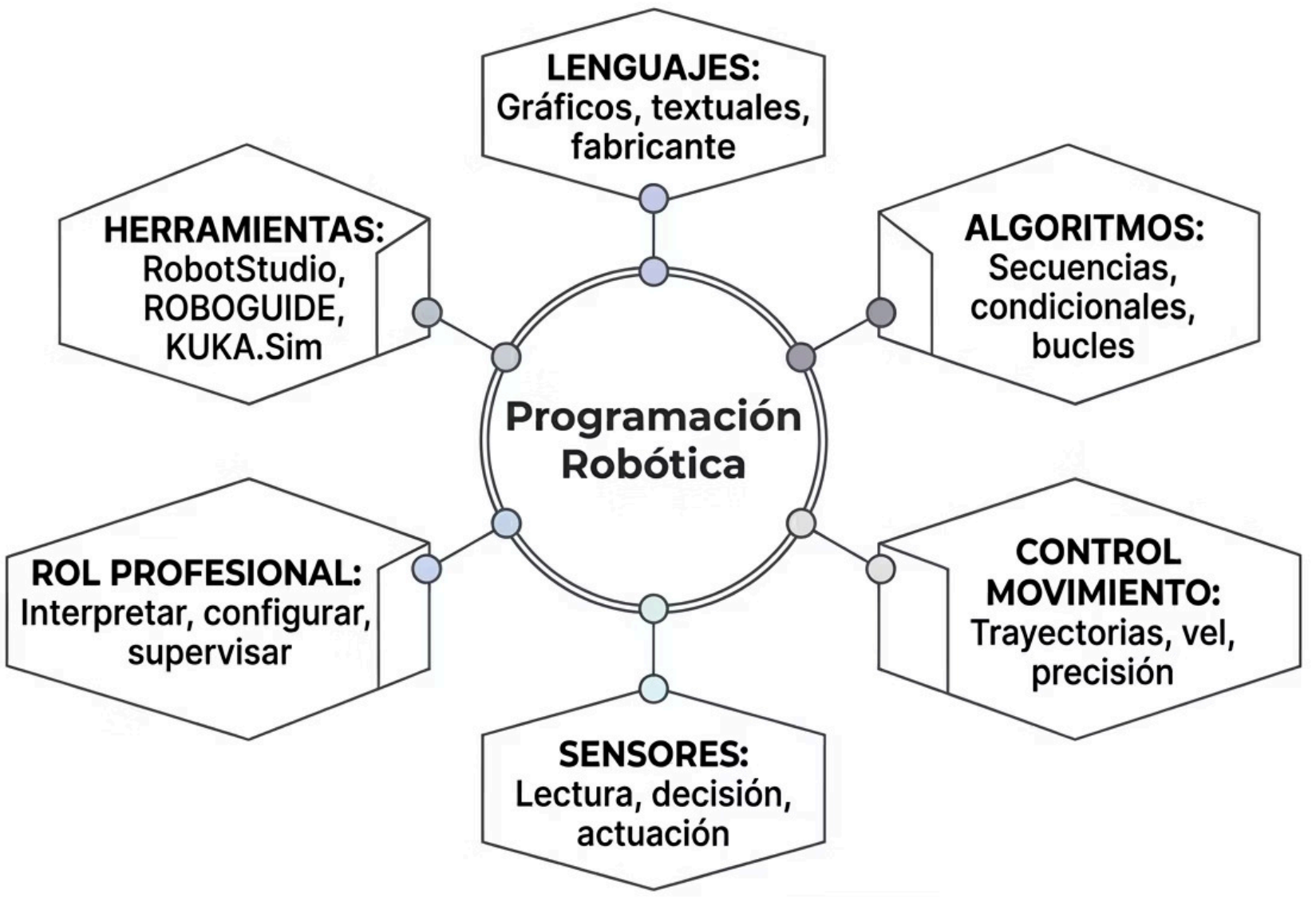
Fabricante: KUKA AG. Plataforma de planificación y simulación de instalaciones robóticas completas con robots KUKA. Permite modelar células de trabajo en 3D, programar en KRL, verificar alcanzabilidad y detectar colisiones. Integra el concepto de *gemelo digital* (digital twin), que permite mantener sincronizado el modelo virtual con el robot real durante toda su vida útil.

Más información en: **https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/software**

Recomendación profesional: Dedica tiempo a practicar con la versión de evaluación de al menos una de estas herramientas. La simulación offline es una competencia cada vez más demandada por las empresas del sector y un elemento diferenciador en tu perfil profesional.

Resumen Visual de Conceptos Clave

A continuación encontrarás un esquema visual que sintetiza los conceptos fundamentales del Tema 7, organizados en sus áreas temáticas principales. Utiliza este resumen como mapa mental de referencia para repasar el contenido del tema y para preparar las actividades de autoevaluación.



Este mapa conceptual te muestra de un vistazo cómo los seis grandes bloques del tema están interconectados y centrados todos en torno al concepto nuclear de la programación robótica. Observa cómo el rol profesional ocupa un espacio transversal: el profesional en robótica aplica conocimientos de lenguajes, algoritmos, control de movimiento y sensores de forma integrada, utilizando herramientas digitales especializadas para hacerlo de forma eficiente y segura.

6	3	±0,02…	3
Ejes articulados	**Tipos de lenguaje**	**Repetibilidad**	**Estructuras algorítmicas**
Grados de libertad de un robot industrial estándar, cada uno programable independientemente.	Gráfico, textual y específico de fabricante: los tres grandes categorías de programación robótica.	Precisión típica de un robot industrial moderno, resultado de una buena programación y calibración.	Secuencia, condicional y bucle: los bloques fundamentales de cualquier programa robótico.

Recomendaciones y Lectura Complementaria

Para profundizar en los contenidos de este tema y consolidar tu formación en programación y control robótico, se recomienda la consulta de los siguientes recursos:

Lectura recomendada

Manual de robótica.

Recurso completo y accesible que aborda los fundamentos de la programación robótica desde una perspectiva práctica y educativa. Especialmente recomendado para comprender la transición entre la robótica educativa y la industrial.

Acceder al manual completo (PDF)

Documentación técnica oficial

Consulta siempre la documentación técnica del fabricante del robot con el que trabajes. Los manuales de programación de ABB (RAPID), KUKA (KRL) y FANUC (Karel) están disponibles en los portales de soporte de cada fabricante y son la referencia definitiva para la programación profesional.

Práctica con simuladores

Descarga e instala ABB RobotStudio (versión de evaluación gratuita) y realiza los tutoriales oficiales disponibles en la web de ABB. Esta es la forma más efectiva de aplicar los conceptos aprendidos en este tema en un entorno seguro y sin coste de equipamiento.

Ejercicios de Verdadero o Falso — Autoevaluación sobre la lectura recomendada

Tras la lectura del manual recomendado, responde a las siguientes afirmaciones indicando si son verdaderas (V) o falsas (F). Las respuestas correctas se incluyen a continuación de cada enunciado para que puedas comprobar tu comprensión.

No	Afirmación	Respuesta correcta
1	La programación de un robot consiste únicamente en indicarle las posiciones a las que debe moverse, sin necesidad de especificar la velocidad ni el tipo de movimiento entre ellas.	**FALSO.** La programación incluye también la definición de la velocidad, el tipo de interpolación de trayectoria (PTP, LIN, CIRC), los parámetros de herramienta y las acciones sobre las salidas digitales, entre otros aspectos. Solo indicar posiciones produce programas incompletos e inseguros.
2	Los lenguajes de programación gráfica, como los basados en bloques, son adecuados para introducirse en la programación robótica y para aplicaciones educativas, aunque presentan limitaciones en entornos industriales complejos.	**VERDADERO.** Los lenguajes gráficos facilitan el aprendizaje y el prototipado rápido, pero la industria profesional requiere los lenguajes específicos de cada fabricante (RAPID, KRL, Karel) para acceder a toda la funcionalidad del sistema de control del robot.
3	Los sensores en un sistema robótico solo tienen la función de detectar la presencia o ausencia de objetos, sin poder influir en la lógica ni en el movimiento del robot durante la ejecución del programa.	**FALSO.** Los sensores son elementos activos en la programación robótica moderna: sus señales son leídas por el programa, evaluadas mediante condicionales y utilizadas para modificar en tiempo real la trayectoria, la velocidad, la activación de herramientas o la selección de subrutinas de actuación del robot.