ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería de Computadores

SIMULADOR CINEMÁTICO DE UN ROBOT MANIPULADOR INDUSTRIAL

KINEMATIC SIMULATOR OF AN INDUSTRIAL ROBOT ARM

Realizado por

M. Dolores Román Romero

Tutorizado por

D. Enrique Domínguez Merino

Cotutorizado por

D. Víctor F. Muñoz Martínez

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, diciembre 2014

RESUMEN

Este trabajo fin de grado trata sobre la implementación de un simulador cinemático de un robot manipulador industrial, orientado al aprendizaje de los principios de programación y desarrollado mediante la herramienta de software matemático MATLAB, dicho simulador debe tener como características principales ser capaz de emular las características de programación que incorporan los lenguajes a nivel robot y resultar fácilmente accesible a los alumnos de las ingenierías.

Asimismo, el simulador tendrá la capacidad de definir los objetos que integran el entorno físico que rodean al robot con el objeto de simular la interacción cinemática del brazo manipulador con dicho entorno.

Para ello, primero se realizará un estudio de los lenguajes de nivel robot, en este caso concreto V+, con el objeto de elaborar un catálogo de funciones y estructuras relevantes, concretamente se trataran las estructuras de datos, funciones del robot, etc. A partir de estos, se elaborarán las especificaciones que debe cumplir el simulador cinemático.

Por último se realizarán unas prácticas sobre el simulador orientadas al aprendizaje y elaboración de los manuales de usuario del mismo.

Palabras clave:

Robot industrial, robótica, simulador, brazo manipulador, MATLAB, cinemática, programación, V+

ABSTRACT

This work deals with the implementation of a kinematic simulation of an industrial robot manipulator, which is oriented to the learning programming principles and it is developed by the mathematical software tool MATLAB, this simulator must be capable to emulate the characteristics of programming languages that are incorporated at robot level and show kind access to students of engineering.

Moreover, the simulator will have the ability to define the objects that make up the physical environment surrounding the robot in order to simulate the kinematic manipulator arm interaction with its environment.

To that end, first a study level robot languages will be made, in this case V +, in order to develop a catalog of relevant functions and structures, that is the addressed data structures, the robot functions, etc. Thus, the specifications that must meet the kinematic simulator are developed.

Finally, some practices will be made with the simulator oriented to the learning and development of user manuals.

Palabras clave:

Industrial robot, robotics, simulator, manipulator arm, MATLAB, kinematics, decoupling, , programming, V+

1. INTRODUCCIÓN

Los robots manipuladores se han hecho imprescindibles en nuestros días, son herramientas poderosas utilizadas con frecuencia en la industria, pero también podemos encontrarlos en otras áreas como por ejemplo la medicina.

Se caracterizan por su capacidad para realizar tareas peligrosas para las personas, trabajos en lugares de difícil acceso, manipulación de objetos pesado o de gran tamaño, tareas que requieren precisión y repetitividad.

Su diseño permite el posicionamiento de un efector final de una manera precisa y flexible. La flexibilidad de un brazo manipulador depende de los grados de libertad posea, cuantos más grados de libertad tenga un brazo manipulador, en más puntos en el espacio y con más orientaciones podrá posicionarse.

Los principios e la programación de los robots industriales abarcan nuevos conceptos y metodologías que no son habituales en los lenguajes de programación convencionales como pueden ser el manejo de las entradas y salidas, la concurrencia, la recuperación de errores, los movimientos del robot y el modelado geométrico del entorno. Gracias al uso de simuladores el aprendizaje de estos conceptos se hace más sencillo al conseguir estos reproducir los movimientos como si de un robot industrial se tratara.

A causa de lo anterior surge la idea de crear un simulador cinemático basado en trabajos previos desarrollados en este ámbito bajo la plataforma MATLAB.

1.1 Objetivo

El objetivo principal de este trabajo es el desarrollo de un simulador de un robot industrial antropomórfico basado en MATLAB, y capaz de emular las características de programación que incorporan los lenguajes a nivel robot. Asimismo, esta plataforma tendrá la capacidad de definir los objetos que componen el entorno físico que rodea al robot con el propósito de simular la interacción cinemática del robot con los elementos de su entorno.

Para ello se analizarán los lenguajes a nivel robot convencionales con el fin de elaborar un catálogo con las funciones y estructuras más importantes que los definen para después implementarles en el simulador

1.2 Contenido de la memoria

La memoria de este trabajo fin de grado se estructura en vario capítulos, en el primero “FUNDAMENTOS DE LA ROBÓTICA” se hace una introducción al mundo de la Robótica, definiendo los conceptos robot y robótica, tipos de robot, así como una descripción detallada de los componentes de un sistema robot.

En los siguientes capítulos se va resolviendo progresivamente el problema del control cinemático de los movimientos en el caso de los robots manipuladores industriales hasta llegar al desarrollo del simulador.

En el capítulo “ROBOTS MANIPULADORES” se define el término robot industrial, profundizando en los sistemas robotizados, su estructura y elementos terminales, también se presentan las distintas estructuras de los robots manipuladores y sus parámetros característicos.

A continuación encontramos el capítulo “CINEMÁTICA DE UN ROBOT MANIPULADOR”, donde se introduce el concepto de localización como herramienta para especificar tanto la posición y la orientación de los objetos que forman el entorno de trabajo del robot como las del elemento terminal de este último, también se definen los distintos modelos que resuelven el problema cinemático: cinemática directa, cinemática inversa y desacoplo cinemático, que simplifica

El capítulo denominado “PROGRAMACIÓN”, describe las distintas técnicas de programación a nivel robot, sus ventajas y las características básicas que debe tener un lenguaje de programación a nivel robot para que se considere óptimo. Se trata la programación explícita de los robots manipuladores, detallando las características de este tipo de lenguajes desde las instrucciones de movimiento a la integración de sensores.

Finalizando con los capítulos encontramos el denominado “SIMULACIÓN CINEMÁTICA” donde se implementa el simulador cinemático del brazo manipulador a partir de las funciones desarrolladas en MATLAB en el presente trabajo.

Además de los capítulos descritos arriba, tenemos el apartado “ANEXOS” donde se incluyen cuatro prácticas comentadas que implementan distintas tareas del brazo manipulador así como, una simulación del mundo de bloques.

Por último las “CONCLUSIONES” a las que se han llegado tras la redacción de la memoria y el desarrollo de este trabajo y las “REFERENCIAS BIBLIOGRÁFICAS” de las fuentes consultadas para la realización de este trabajo.

Plan de trabajo seguido para la elaboración de este trabajo fin de grado:

1. Estudio de los lenguajes de nivel robot convencionales con el objeto de crear un catálogo de funciones y estructuras más relevantes que los definen. En concreto se han analizado las estructuras de datos, funciones del robot, gestión de señales. Con todo ello, se ha elaborado las especificaciones que debe cumplir el simulador cinemático.

2. Selección del brazo robot para simular, construcción del modelo cinemático directo e inverso, para realizar el modelo geométrico tridimensional del brazo manipulador para su representación en MATLAB.

3. Diseño del editor usado para incorporar a la escena del robot los objetos con los que este debe interaccionar, en concreto se han definido cajas, cintas transportadoras y sensores.

4. Elaboración del bucle de simulación, donde se incorporan las funciones definidas en el punto 1.

5. Desarrollo de un conjunto de prácticas sobre el simulador orientado al aprendizaje y documentación de las mismas.

2. FUNDAMENTOS DE LA ROBÓTICA

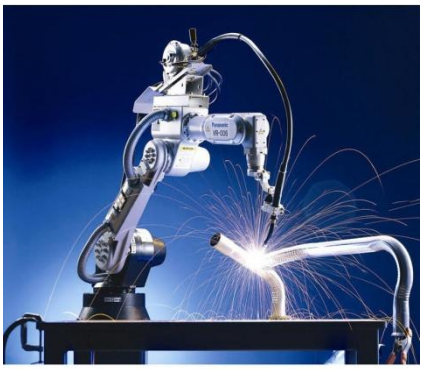
2.1 Definición de robot

¿Qué es un robot? La mayoría de los expertos en robótica dirían que es complicado ofrecer una definición de robot universalmente aceptada.

El término robot viene de la palabra checa *"robota"*, que significa servidumbre o trabajo forzado, si intentamos imaginar un robot nos viene a la mente una máquina con aspecto humano o animal que es capaz de realizar una serie de acciones como moverse, manipular objetos, mostrar un comportamiento inteligente e incluso hablar. Claramente esta imagen está condicionada por la literatura y el cine de Ciencia Ficción, pero la realidad es que un robot puede adoptar una gran variedad de formas (dependiendo de las tareas para las que haya sido diseñado), sin embargo, independientemente de su aspecto, todo robot debe poseer, en mayor o menor grado, las características de **versatilidad** e **inteligencia**, que describimos con más detalle a continuación.

Versatilidad: Capacidad para realizar tareas de distinta índole, incluso aquéllas no previstas en principio por los diseñadores, lo cual implica una considerable auto-adaptabilidad al entorno. La versatilidad va a depender en gran medida de su estructura mecánica por lo que los diseñadores intentan configurar dicha estructura para que cubra, de manera eficiente, el máximo número de aplicaciones posibles.

Inteligencia: Habilidad del robot para tomar decisiones de forma autónoma con el objeto de resolver situaciones imprevistas. Es decir, debe estar capacitado para reconocer su entorno de trabajo y en base a ello, planificar los siguientes movimientos, por ello entendemos que goza de un elevado grado de autonomía y de auto planificación, de manera que es capaz de hacer su tarea sin intervención del operador, tomando las decisiones adecuadas a partir de la información que recaban sus sensores, gracias al programa almacenado en su memoria, en este caso hablamos de robot inteligente.



Estos dos conceptos, nos pueden ayudar a dar una primera definición al término robot, cómo una máquina versátil e inteligente que puede reemplazar a un operador humano, sin embargo, si nos ceñimos a esta definición, cualquier electrodoméstico podría ser considerado un robot, por lo que surge la pregunta de cuál es el grado mínimo de inteligencia que debe poseer dicha máquina para denominarse robot.

En términos generales, un robot para ser considerado como tal, debe presentar alguna de estas capacidades:

-Haber sido creado artificialmente.

-Sentir su entorno.

-Poder manipular objetos.

.Tener cierta inteligencia o habilidad para tomar decisiones basadas en el entorno o en una secuencia pre-programada automática.

-Ser programable.

-Poder moverse en uno o más ejes de rotación o traslación.

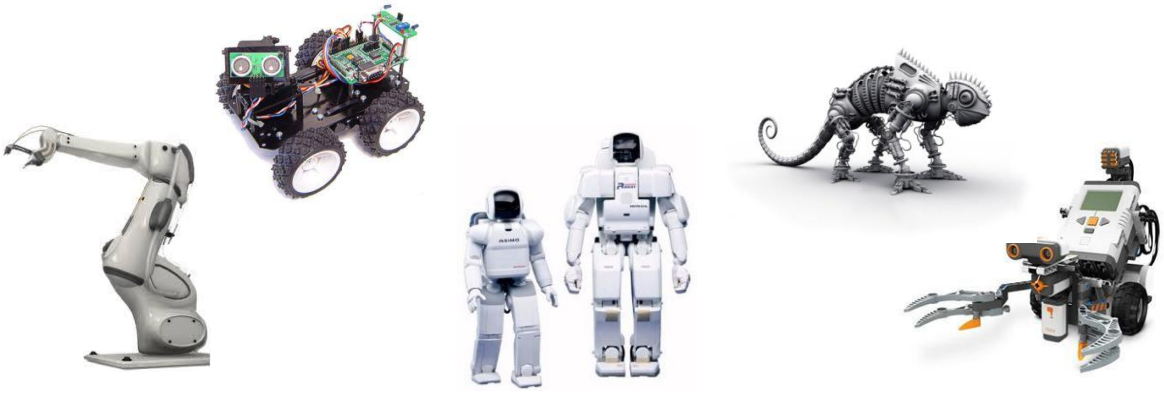
-Realizar movimientos coordinados.

Los robots, en la actualidad, se utilizan en el ámbito industrial (para montar piezas de diversos mecanismos, desplazar grandes pesos y otras tareas), en la medicina (para operar en lugares de difícil acceso) y en el campo militar (para reducir las bajas humanas).

De manera más específica, en el campo industrial, el concepto robot se define como un manipulador programable multifuncional, diseñado para mover material, objetos o dispositivos especializados a través de movimientos variables y programables, para realizar un conjunto de tareas. De esta última idea deriva el concepto de robot manipulador industrial.

2.2 Clasificación de los robots

Existen múltiples formas de clasificar a los robots, por función, tamaño, arquitectura, metodología del diseño, tipo de inteligencia etc.….



Clasificación según su arquitectura:

Poliarticulados: En este grupo se encuentran robots de muy diversa forma y configuración, cuya característica común es la de ser básicamente sedentarios y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas y con un número limitado de grados de libertad. Generalmente se usan en labores industriales.

Móviles: Robots con gran capacidad de desplazamiento, basados en carros o plataformas con un sistema de locomoción de tipo rodante. Su espacio de trabajo no está delimitado o es demasiado extenso para un robot poliarticulados y requieren una capacidad sensorial mayor que estos. Estos robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación.

Androides: Son capaces de reproducir de manera total o parcial la forma y comportamiento cinemático de un ser humano. Actualmente, los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, y destinados, fundamentalmente, al estudio y experimentación.

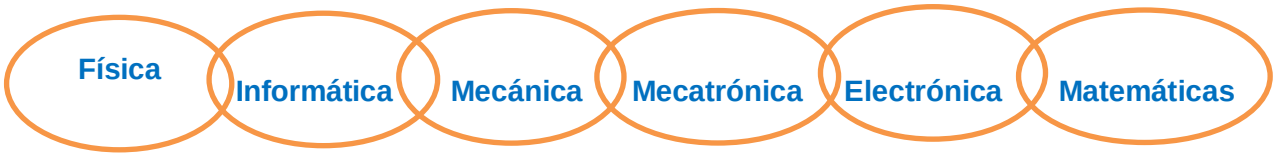
Zoomórficos: Dentro de esta categoría podrían incluirse también a los androides, su característica principal es que poseen un sistema de locomoción que imitan al de algunos seres vivos. Se clasifican en dos grupos principalmente, los caminadores y los no caminadores-en el interior de las venas y arterias humanas o los usados para la revisión del colon.

Híbridos: Corresponden a aquellos de difícil clasificación, cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo, uno de los atributos de los robots móviles y de los robots zoomórficos.

2.3 Definición de Robótica

La Robótica se define como el conjunto de conocimientos teóricos y prácticos que permiten concebir, realizar y automatizar sistemas basados en estructuras mecánicas poliarticuladas, dotados de un determinado grado de inteligencia y destinarlos a la producción industrial o a la sustitución del hombre en múltiples tareas, es decir la robótica abarca tanto el diseño y construcción de robots, hasta su implantación con el fin de resolver problemas de automatización de procesos industriales.

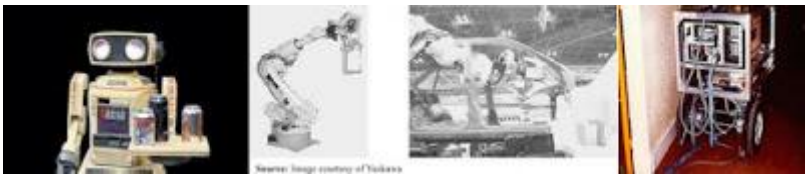
Se basa en diversas áreas de estudio, la mecatrónica, la física y las matemáticas podemos considerarlas como ciencias básicas, aunque también cabe destacar la mecánica, electrónica, informática, inteligencia artificial e ingeniería de control.



Se suelen distinguir cuatro generaciones en el desarrollo de la industria robótica:

- **Primera generación**: **Manipuladores.** Tenían capacidad para almacenar trayectorias de movimiento descritas punto a punto. Esta primera generación de robots era programable, con un sencillo sistema de control y de tipo brazo manipulador.

- **Segunda generación: Robots de aprendizaje**. La segunda generación de robots entra en escena a finales de los años 70. Estos robots cuentan con sensores externos (tacto y visión por lo general) que dan al robot información (realimentación) limitada del mundo exterior.



- **Tercera Generación: Robots con control sensorizado**. Durante esta etapa, que tiene lugar durante los años 80s y 90s, los robots ahora cuentan con controladores (computadoras) que usando los datos o la información obtenida de sensores, obtienen la habilidad de ejecutar las ordenes de un programa escrito en un lenguaje de programación.

- **Cuarta Generación: Robots inteligentes**. Son similares a los anteriores, emplean la inteligencia artificial y hacen uso de los ordenadores más avanzados, estos ordenadores no sólo trabajan con datos, sino que también lo hacen con los propios programas, realizan razonamientos lógicos y aprenden. .

El término Robótica fue acuñado por el escritor Isaac Asimov, quien creo también las 3 leyes de la Robótica:

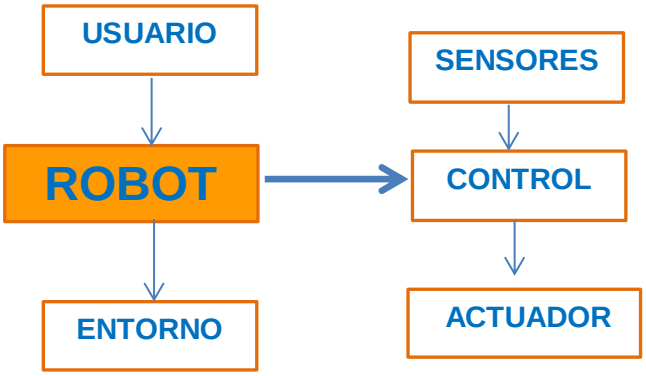
Primera Ley: un robot no debe dañar a una persona o dejar que una persona sufra un daño por su falta de acción.

Segunda Ley: un robot debe cumplir con todas las órdenes que le dicta un humano, con la salvedad que se produce si estas órdenes fueran contradictorias respecto a la Primera Ley**.**

Tercera Ley, establece que un robot debe cuidar su propia integridad, excepto cuando esta protección genera un inconveniente con la Primera o la Segunda Ley**.**

2.4 Componentes de un sistema robot

Después de todo lo expuesto, llegamos a la conclusión de que un robot es una máquina compleja que se compone de dos partes fundamentales, una parte mecánica, encargada de actuar sobre el mundo real, y otra electrónica que constituye la inteligencia del mismo.



Configuración básica de un sistema robot

Teniendo en cuenta que un robot debe realizar determinadas tareas en un entorno concreto podemos hablar de un sistema robot-entorno definiéndolo como un conjunto de subsistemas, podemos clasificar estos subsistemas en:

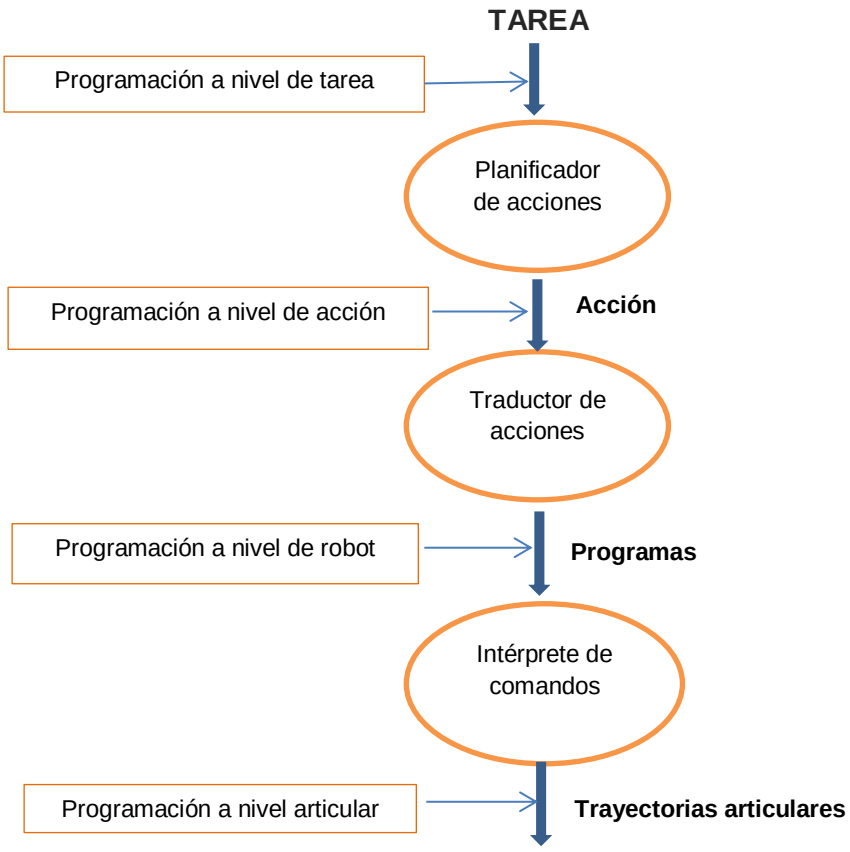
Subsistema de la tarea

Es el encargado de elaborar la trayectoria que trazará el robot para realizar la tarea asignada a partir del programa del usuario. Proporciona la comunicación entre la persona y la máquina a través de un lenguaje, este debe estar diseñando para optimizar tanto la productividad de la programación como la ejecución de la tarea.

El sistema de la tarea define la capacidad de planificación autónoma del sistema robot tanto para definir el nivel de comunicación hombre-robot, como para responder en tiempo real a situaciones no previstas, como detección de fallos y recuperación de planes, estos se refieren al uso de la realimentación sensorial del entorno de trabajo con el objeto de generar acciones que, ante posibles imprevistos, garanticen que la tarea se lleve a buen término.

Los cuatro niveles de programación permitidos en la comunicación persona-máquina son, de tarea, de acción, de robot y articular, siguiendo la secuencia que detallamos a continuación.

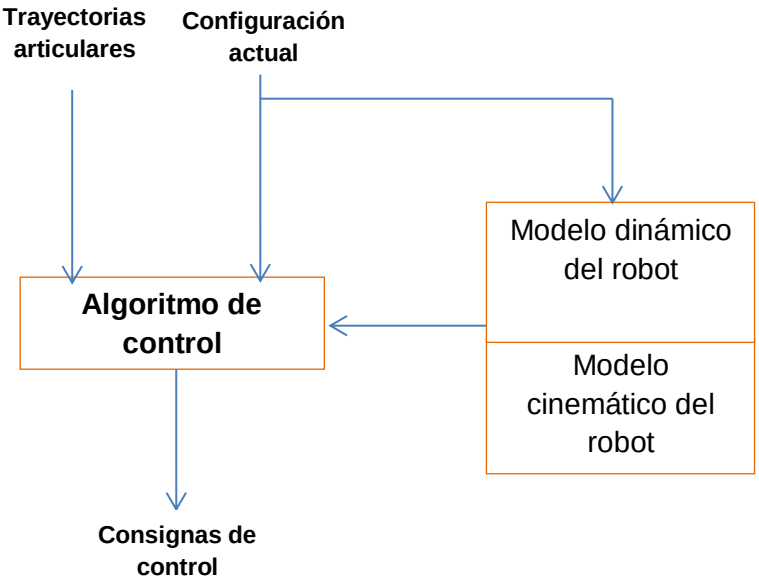
En el nivel tarea el operador describe la tarea en un lenguaje natural, esto es traducido a una secuencia de acciones de movimientos básicos (nivel de acción), estas acciones se traducen una a una al lenguaje propio del robot (nivel robot) y este código servirá para obtener las trayectorias articulares (nivel articular), que actúan de referencia para el **subsistema de control.**



Subsistema de control

El subsistema de control se encarga, de ejecutar, con los mínimos errores posibles, las trayectorias articulares planificadas, incluso ante la presencia de perturbaciones, aquí los comandos de alto nivel son convertidos en referencias para los actuadores físicos.

Para esto, la estrategia de control se configura a partir de los modelos cinemáticos y dinámico de la componente mecánica, de forma que la respuesta temporal de esta última verifique las especificaciones de control impuestas.



La salida de este bucle interno de servocontrol, se encamina hacia el **subsistema actuador** para que las convierta en la energía cinética que moverá la mecánica. Para esta acción, este último subsistema, mediante el uso de la electrónica de potencia, amplifica las consignas de control, a los niveles adecuados para excitar a los actuadores.

Subsistema actuador

Este se compone de los actuadores y los amplificadores de potencia encargados de mover a la estructura mecánica. La naturaleza de estos elementos determina las prestaciones del robot en cuanto a movilidad, capacidad de carga o precisión en sus movimientos. Los actuadores se clasifican en:

-**Neumáticos**. Usan aire comprimido y se caracterizan por su ligereza y su tamaño compacto, sin embargo, tienen la desventaja de que son imprecisos.

-**Hidráulicos**. Son similares al émbolo neumático, pero utiliza aceite en vez de aire. Consiguen movimientos más precisión aunque más lentos, tienen una gran capacidad para manejar grandes cargas pero necesitan estrategias de control más complejas.

-**Eléctricos**. Estos son los más usados en la actualidad, gracias al servocontrol se pueden usar sobre ellos sofisticadas técnicas de regulación automática con la que se consigue altas prestaciones en cuanto a precisión.

Subsistema mecánico.

El subsistema mecánico está compuesto por una serie de elementos rígidos conectados entre sí y que se encargan de actuar sobre su entorno de trabajo por medio de una herramienta especializada en la tarea que tiene que realizar, este puede configurarse de múltiples maneras, como por ejemplo un brazo manipulador, una plataforma móvil, pistola de pintura, fresadoras etc….

Subsistema de proceso

Este describe la forma en la que el robot realiza la tarea especificada, para ello hay que tener en cuenta una serie de cuestiones como son el sorteo de obstáculos, la velocidad de desplazamiento de los elementos mecánicos con el fin de sincronizarlos con otras máquinas, la seguridad de los operarios humanos.

El subsistema de proceso se define como externo al sistema robot y se compone de todos los elementos que se encuentran en el interior del entorno de trabajo.

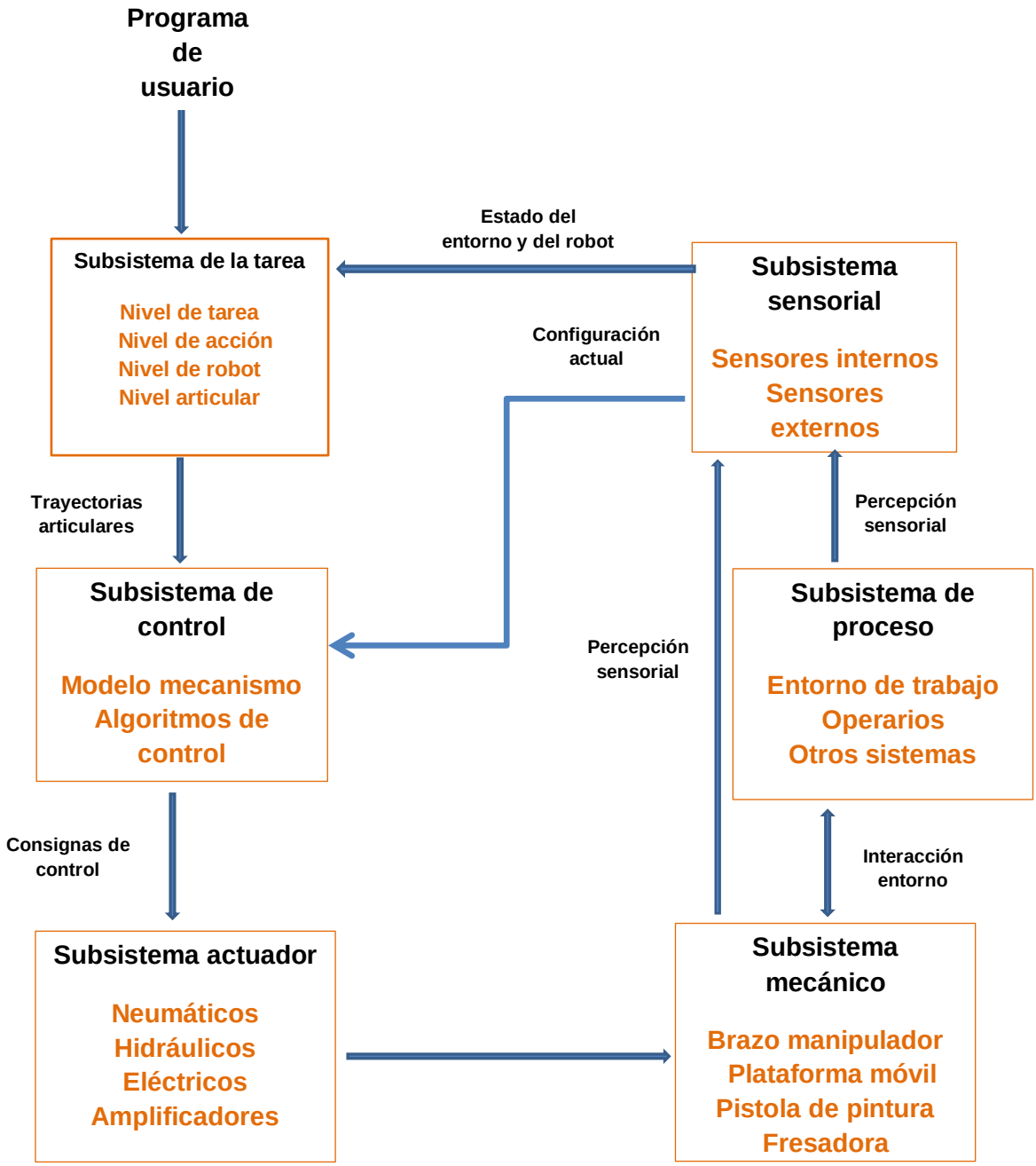
En el caso de robots poliarticulados este subsistema resulta más limitado que en los móviles, facilitando esto la construcción de un mapa exacto para que facilite al sistema de control la realización de la tarea. En este caso se dice que la tarea se realiza en un entorno estructurado y no necesita de mucha capacidad sensorial para realizar la tarea, lo que abarata costes.

Subsistema sensorial

Es el encargado de recopilar todos los datos recogidos por los sensores y procesarlos para que puedan ser interpretados por los subsistemas de la tarea y de control.

A través de este subsistema el robot conoce tanto su estado interno como el del entorno en el que trabaja. Para ello, se encuentra dotado de sensores internos que informan al sistema de control de la configuración que posee la estructura mecánica en cada momento, y externos que suministran datos acerca del ámbito donde el robot realiza la tarea. El subsistema sensorial permite el acceso a unos sensores virtuales, que proporcionan la información a nivel de abstracción requerido. .

En resumen, el proceso de interpretación sensorial es complejo y vulnerable a las condiciones ambientales de trabajo, llevando a extraer falsas conclusiones, este hecho condiciona que los robots industriales poseen un bajo grado de percepción sensorial que compensan estructurando el entorno de trabajo. Por otro lado los sensores externos tienen un coste elevado y una complicada puesta a punto, cuando en las aplicaciones industriales se buscan soluciones sencillas y robustas



ESQUEMA DE LOS SUBSISTEMAS DE UN ROBOT

3. ROBOTS MANIPULADORES

Existen ciertas dificultades a la hora de establecer una definición formal de lo que es un robot industrial.

Según la Asociación de Industrias de Robótica (RIA, Robotic Industry Association):

"Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas, o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas"

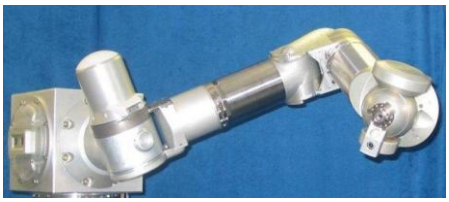
Según la Organización Internacional de Estándares (ISO) que define al robot industrial como:

"Manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas"

Se incluye en esta definición la necesidad de que el robot tenga varios grados de libertad. Una definición más completa es la establecida por la Asociación Francesa de Normalización (AFNOR), que define primero el manipulador y, basándose en dicha definición, el robot:

Manipulador: mecanismo formado generalmente por elementos en serie, articulados entre sí, destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o mediante dispositivo lógico.

Robot: manipulador automático servo-controlado, reprogramable, polivalente, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectoria variables reprogramables, para la ejecución de tareas variadas. Normalmente tiene la forma de uno o varios brazos terminados en una muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Normalmente su uso es el de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material.



Común en todas las definiciones anteriores es la aceptación del robot industrial como un brazo mecánico con capacidad de manipulación y que incorpora un control más o menos complejo.

Un sistema robotizado, en cambio, es un concepto más amplio, engloba todos aquellos dispositivos que realizan tareas de forma automática en sustitución de un ser humano y que pueden incorporar o no a uno o varios robots, siendo esto último lo más frecuente.

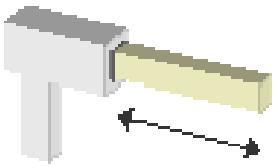
3.1 Estructura de los robots manipuladores

Un manipulador robótico consta de una secuencia de elementos estructurales rígidos, denominados enlaces o eslabones, conectados entre sí mediante juntas o articulaciones que permiten el movimiento relativo de cada dos eslabones consecutivos.

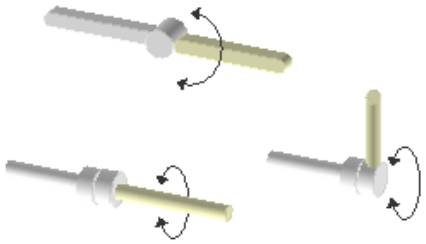
Una articulación puede ser:

- **Lineal** (deslizante, traslacional o prismática), si un eslabón desliza sobre un eje solidario al eslabón anterior.

- **Rotacional**, en caso de que un eslabón gire en torno a un eje solidario al eslabón anterior.



Lineal



Rotacional

El conjunto de eslabones y articulaciones se denomina **cadena cinemática**. Se dice que una cadena cinemática es abierta si cada eslabón se conecta mediante articulaciones exclusivamente al anterior y al siguiente, exceptuando el primero, que se suele fijar a un soporte, y el último, cuyo extremo final queda libre. A éste se puede conectar un **elemento terminal o actuador final**: una herramienta especial que permite al robot de uso general realizar una aplicación particular, que debe diseñarse específicamente para dicha aplicación: una herramienta de sujeción, de soldadura, de pintura, etc. El punto más significativo del elemento terminal se denomina **punto terminal (PT)**. En el caso de una pinza, el punto terminal vendría a ser el centro de sujeción de la misma

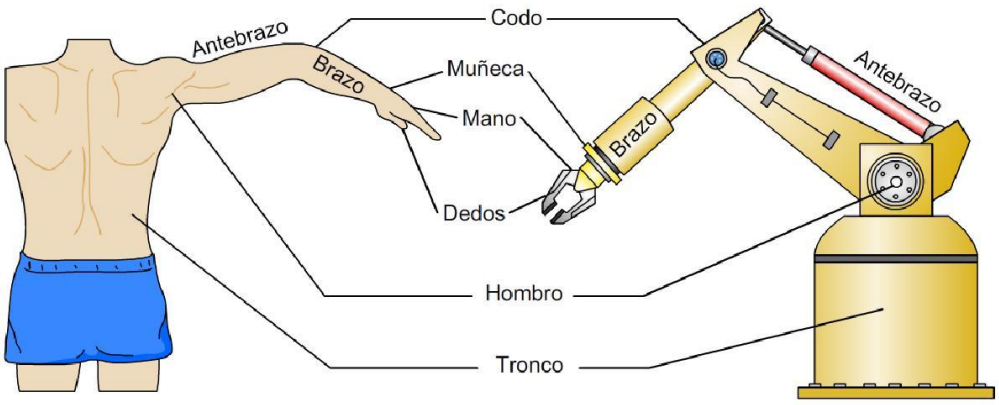
Los elementos terminales pueden dividirse en dos categorías:

- **Pinzas**
- **Herramientas**

Las **pinzas** se utilizan para tomar un objeto, normalmente la pieza de trabajo, y sujetarlo durante el ciclo de trabajo del robot. Hay una diversidad de métodos de sujeción que pueden utilizarse, además de los métodos mecánicos obvios de agarre de la pieza entre dos o más dedos. Estos métodos suplementarios incluyen el empleo de casquillos de sujeción, imanes, ganchos, y cucharas.

Una **herramienta** se utiliza como actuador final en aplicaciones en donde se exija al robot realizar alguna operación sobre la pieza de trabajo. Estas aplicaciones incluyen la soldadura por puntos, la soldadura por arco, la pintura por pulverización y las operaciones de taladro. En cada caso, la herramienta particular está unida a la muñeca del robot para realizar la operación.

A los manipuladores robóticos se les suele denominar también **brazos de robot** por la analogía que se puede establecer, en muchos casos, con las extremidades superiores del cuerpo humano.



Semejanza de un brazo manipulador con la anatomía humana

Se denomina **grado de libertad (g.d.l.)** a cada una de las coordenadas independientes que son necesarias para describir el estado del sistema mecánico del robot (posición y orientación en el espacio de sus elementos). Normalmente, en cadenas cinemáticas abiertas, cada par eslabón-articulación tiene un solo grado de libertad, ya sea de rotación o de traslación. Pero una articulación podría tener dos o más **g.d.l**. que operan sobre ejes que se cortan entre sí.

Distintos grados de libertad de un brazo de robot

Para describir y controlar el estado de un brazo de robot es preciso determinar:

- La posición del punto terminal (o de cualquier otro punto) respecto de un sistema de coordenadas externo y fijo, denominado el sistema mundo.

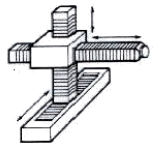
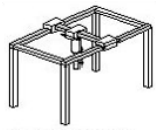
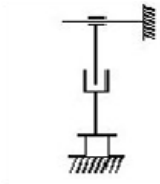
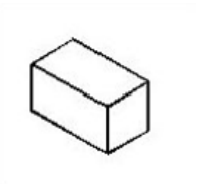
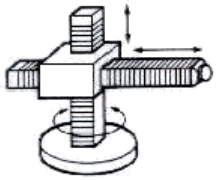
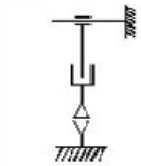
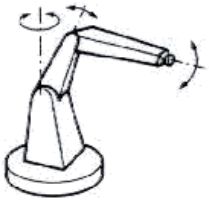
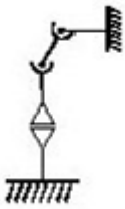
- El movimiento del brazo cuando los elementos actuadores aplican sus fuerzas y momentos.

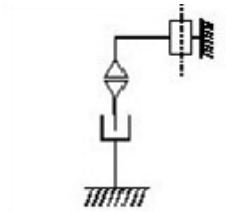
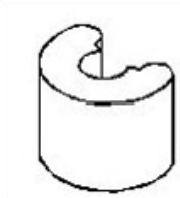
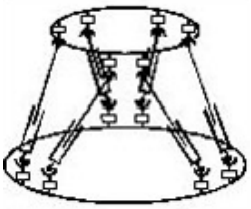
El análisis desde el punto de vista mecánico de un robot se puede efectuar atendiendo exclusivamente a sus movimientos (estudio cinemático) o atendiendo además a las fuerzas y momentos que actúan sobre sus partes (estudio dinámico) debidas a los elementos actuadores y a la carga transportada por el elemento terminal.

3.2 Estructuras mecánicas

Según la geometría de su estructura mecánica, un manipulador puede ser:

- **Cartesiano**, cuyo posicionamiento en el espacio se lleva a cabo mediante articulaciones lineales.
- **Cilíndrico**, con una articulación rotacional sobre una base y articulaciones lineales para el movimiento en altura y en radio.
- **Polar**, que cuenta con dos articulaciones rotacionales y una lineal.
- **Esférico** (o de brazo articulado), con tres articulaciones rotacionales.
- **Mixto**, que posee varios tipos de articulaciones, combinaciones de las anteriores. Es destacable la configuración SCARA (Selective Compliance Assembly Robot Arm)
- **Paralelo**, posee brazos con articulaciones prismáticas o rotacionales concurrentes.

Configuración geométrica	Estructura cinemática	Espacio de trabajo	Ejemplo
cartesianos  tipo cantilever  tipo pórtico			
cilíndrico 			
Polar 			
Esférico 			

SCARA 			
paralelo 			

Configuraciones geométricas, estructura cinemática, espacio de accesibilidad y ejemplos de robots industriales

3.3 Parámetros característicos

Los principales parámetros que caracterizan a los robots industriales son:

- **Número de grados de libertad**. Es el número total de grados de libertad de un robot, dado por la suma de g.d.l. de las articulaciones que lo componen.
- **Espacio de accesibilidad o espacio (volumen) de trabajo**. Es el conjunto de puntos del espacio accesibles al punto terminal, que depende de la configuración geométrica del manipulador.
- **Capacidad de posicionamiento del punto terminal**. Se concreta en tres magnitudes fundamentales: resolución espacial, precisión y repetibilidad, **Capacidad de carga**. Es el peso que puede transportar el elemento terminal del manipulador.
- **Velocidad**. Es la máxima velocidad que alcanzan el PT y las articulaciones.

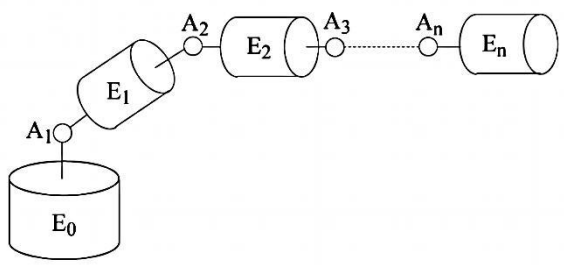
4. CINEMÁTICA DE UN ROBOT MANIPULADOR

Un robot manipulador consta de una secuencia de cuerpos rígidos, llamados eslabones o elementos conectados mediante articulaciones. En un extremo del robot está fijo a la base mientras que el otro está libre y unido a una herramienta.

Podemos esquematizarlo de la siguiente forma:

- Cada pareja articulación-eslabón se le conoce como grado de libertad.
- Un robot con n grados de libertad, tiene n pares de articulaciones- eslabones. La base del robot no se considera parte del robot.
- Las articulaciones-eslabones se enumeran desde la base del robot y hasta la muñeca.
- Para cada par articulación-eslabón se establece un sistema coordenado intermedio que se fija al eslabón y que se mueve conforme el eslabón se mueva.
- En la base del robot se establece el sistema coordenado de referencia {0} y en la muñeca del robot se establece el sistema coordenado móvil {n}.
- Los elementos i e i-1 están conectados por la articulación i.

El conjunto de eslabones y articulaciones se denomina **cadena cinemática**.



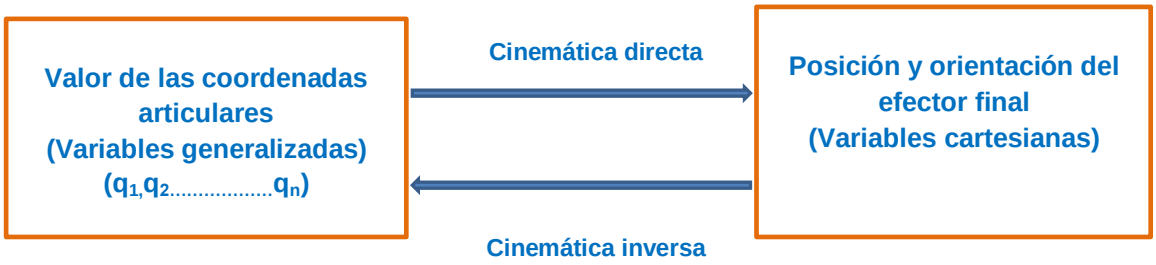
Estructura de cadena cinemática

La **cinemática** es el estudio de su movimiento con respecto a un sistema de referencia o dicho de otra forma, trata del estudio analítico del movimiento (posición, velocidad, aceleración) del robot con respecto a un sistema coordenado fijo como una función del tiempo sin considerar las fuerzas/momentos que originaron dicho movimiento.

Para realizar tareas de manipulación es esencial conocer donde se encuentran ubicados los objetos dentro del área de trabajo del robot y la localización del extremo final del robot manipulador, para ello definimos formalmente los conceptos de:

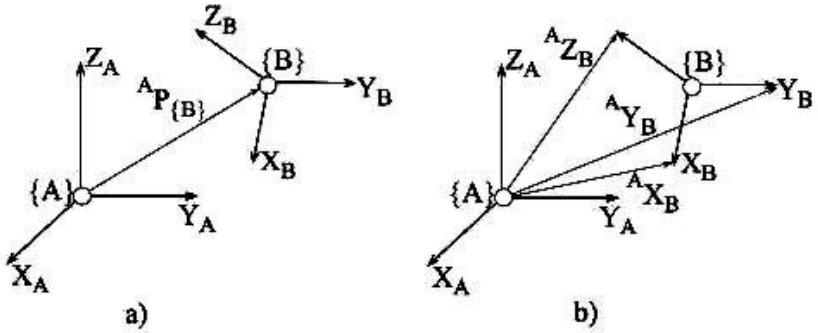
- **Localización de objetos**: Especificación de la ubicación de los objetos del área de trabajo en relación con algún sistema de referencias.

- **Cinemática directa**: Determinación de la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas de referencia, conocidos los ángulos de las articulaciones y los parámetros geométricos de los elementos del robot.



Estas dos cuestiones se resuelven mediante la definición de un conjunto de sistemas de coordenadas locales que se asocian a cada objeto del entorno de interés y a cada uno de los eslabones del robot manipulador.

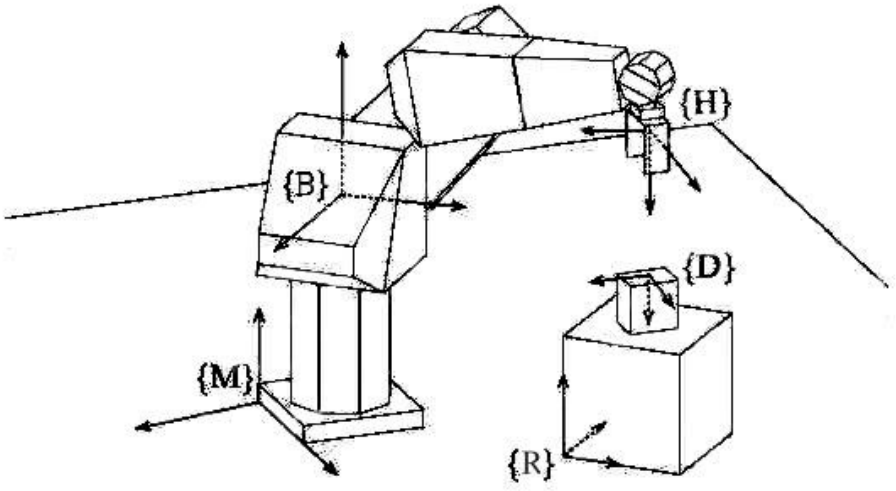
De esta forma, se define la localización como la relación geométrica que se establece entre dos sistemas de coordenadas cartesianas y que define los vínculos de posición y orientación de un sistema respecto al otro.



Localización del sistema {A} con respecto al sistema {B}

4.1 Representación de la posición y la orientación

Como definimos en el apartado anterior, para especificar una tarea concreta de un robot manipulador, debemos de resolver tanto la localización como el problema cinemático directo, para ello, hay que establecer un sistema global de referencias y un conjunto de sistemas de coordenadas locales, asociados a cada uno de los elementos del robot y del entorno que se considere para esta tarea específica, a continuación hay que localizar, de forma directa o indirecta, los sistemas locales con el global de referencia mediante el uso de las matrices de transformación homogéneas.



Tarea de manipulación de un robot

{B} Sistema de coordenadas de la base del robot.

{H} Sistema de coordenadas de la herramienta final del robot.

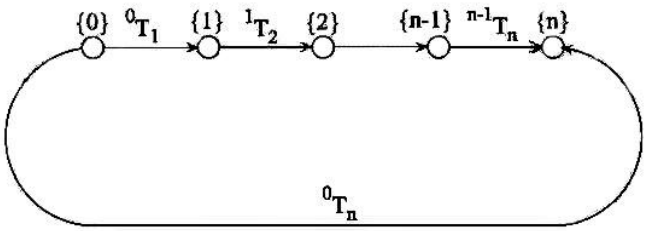
{R} Sistema de coordenadas de la tarea específica.

{D} Localización de agarre.

4.2 Modelo cinemático directo

El modelo cinemático directo de un robot manipulador se define como la matriz transformada homogénea que vincula el sistema ubicado en la herramienta con el que se localiza en la base inmóvil del robot.

El método de Deanvit-Hartenberg se establece como un procedimiento adecuado para el cálculo del modelo cinemático directo. Se basa en asociar a cada elemento de la cadena cinemática del robot manipulador un sistema de coordenadas locales.



Grafo del modelo cinemático directo de un manipulador

Podemos definir una serie de conceptos relacionado con el modelo cinemático directo:

Variable articular: Totalidad de posiciones que puede tomar el grado de libertad asociado a una articulación.

Vector articular: Conjunto ordenado de todas las variables articulares de un robot manipulador.

Espacio articular: Conjunto de todos los vectores articulares asociados a un robot manipulador. Llamado también espacio de coordenadas generalizadas.

Espacio cartesiano: Conjunto de ejes ortogonales dónde se referencia la posición y orientación del extremo final. También denominado como espacio orientado a la tarea o espacio operacional.

Espacio de actuación: Conjunto de valores que deben tomar los actuadores asociados a cada articulación para conseguir un determinado vector articular. En este caso podemos hablar también de espacio de actuación,

4.3 Caracterización de los eslabones de un robot manipulador

Un robot manipulador de n grados de libertad se define como una cadena cinemática formada por n-1 eslabones unidos entre sí por n articulaciones de un solo grado de libertad. La base inmóvil de la cadena se define como eslabón 0 y el último eslabón terminal.

Cada eslabón tiene una relación espacial invariante entre dos articulaciones consecutivas. A continuación vamos a definir cuatro parámetros conocidos como de Denavit-Harenberg, ya que fueron establecidos por estos investigadores en 1.955 para el modelado cinemático de máquinas.

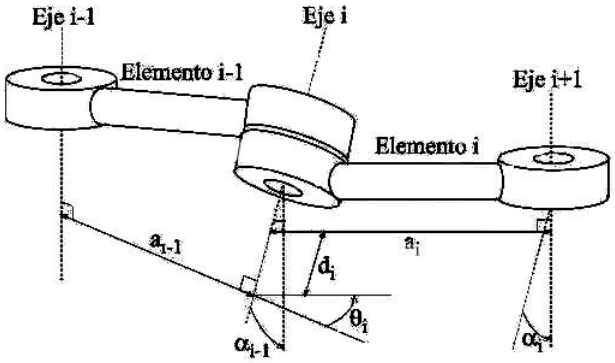
La relación entre dos eslabones consecutivos se define, desde el punto de vista geométrico:

- **Longitud** a_{i-1} La distancia entre los ejes i-1 e i, medida a lo largo de la recta perpendicular que une a los mismos.

- **Torsión** α_{i-1} Ángulo desde el eje i-1 al eje i medido sobre el segmento $a_{i-1,}$ y siguiendo la regla con la mano derecha.

De la misma forma, los parámetros que explicitan la relación geométrica del eslabón i-1 con i, se definen a continuación:

- **Desplazamiento** d_i Distancia a lo largo del eje i desde dónde intersecta a_{i-1}, hasta donde lo hace a_i.
- **Ángulo** θ_i Ángulo desde la prolongación de a_{i-1} hasta la recta a_i, y medido sobre el eje i, y define el grado de libertad en las articulaciones de revolución.



Relación entre dos elementos consecutivos

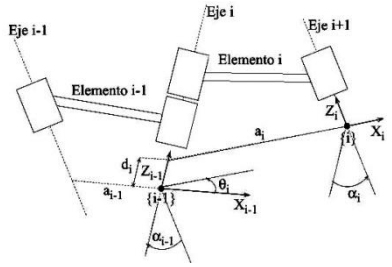
4.4 Construcción del modelo directo

Como se ha comentado en un apartado anterior, la construcción del modelo cinemático de un robot manipulador se basa, en primer lugar, en asignar a cada eslabón de la cadena cinemática un sistema de coordenadas solidaria al mismo, de forma que los parámetros Denavit-Hartenberg conserven la interpretación física que se les ha asignado anteriormente.

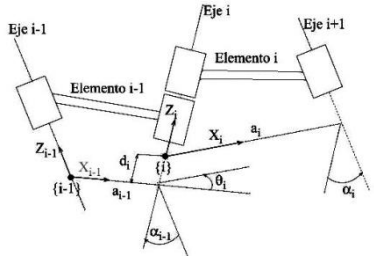
Las dos formas más comunes de asignación de sistemas de coordenadas a cada eslabón del robot manipulador son las denominadas DH1 yDH2.

DH1: Es la más estándar de las dos, consiste en ubicar el sistema i-ésimo solidario con el eslabón homónimo en el extremo más distante al mismo.

DH2: Al contrario que en el anterior, el sistema {i}, solidario al elemento i-ésimo, se ubica en el extremo más próximo de éste, es decir, sobre el eje denominado i.



Asignación de sistemas según la notación DH1



Asignación de sistemas según la notación DH2

Como ejemplo de modelos cinemáticos directos podemos destacar el robot de tipo SCARA y el robot tipo RX90.

Robot tipo SCARA

Robot tipo RX90

4.5 Modelo cinemático inverso

Si el modelo cinemático directo consiste en que a partir de una serie de variables de las articulaciones hallamos la posición y la orientación del extremo final, el modelo cinemático inverso, consiste en que dadas la posición y orientación del extremo final, hallar el vector de variables de articulación La resolución no es sistemática: Depende de la configuración del robot y pueden existir soluciones múltiples.

• Este tipo de solución presenta las siguientes ventajas:

- Posibilidad de resolución en tiempo real (seguimiento de trayectorias).
- Posibilidad de incluir restricciones que garanticen la mejor solución (por ejemplo, límite en los recorridos articulares).
- Posibilidad de simplificaciones.
- Problema: No siempre existe.

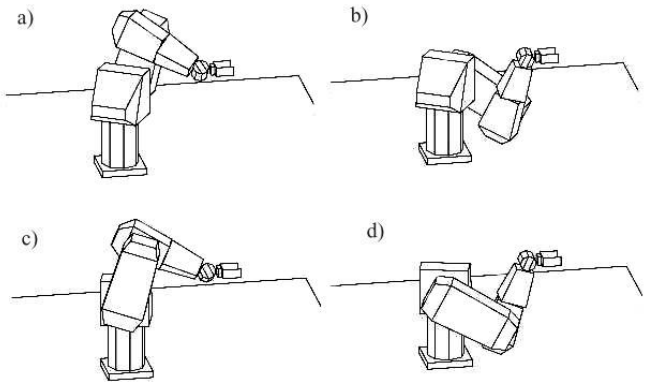
4.6 Desacoplo cinemático

Una de las principales dificultades que nos encontramos en la resolución de los robots manipuladores, es que generalmente nos enfrentamos a sistemas de seis dimensiones. Por esto debemos buscar fórmulas que simplifiquen el problema original. El concepto de desacoplo cinemático se refiere a la posibilidad de resolver de forma independiente la cinemática inversa de la posición y de la orientación.

Los manipuladores con una muñeca de modelo esférico cumplen que los tres primeros eslabones de la cadena cinemática se utilizan de forma exclusiva para posicionar, mientras que la muñeca se usa para alcanzar la orientación requerida.

En el caso concreto del manipulador RX90 podemos alcanzar un punto concreto en el espacio de trabajo de cuatro formas distintas, comenzando por la esquina superior izquierda y teniendo en cuenta la similitud con el brazo humano, sería:

- Brazo izquierdo con codo arriba
- Brazo izquierdo con codo abajo
- Brazo derecho con codo arriba
- Brazo derecho con codo abajo



Soluciones para alcanzar un punto en el espacio

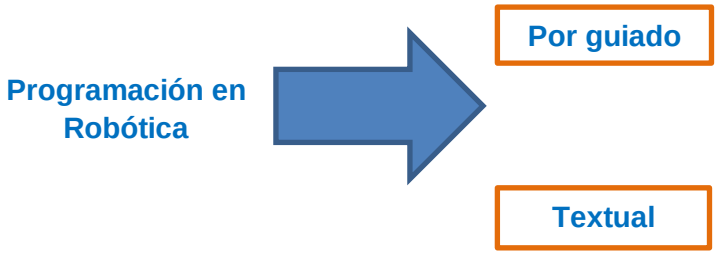
5. PROGRAMACIÓN

Desde el punto de vista de la Robótica, la programación se define como un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas, en nuestro caso concreto consistirá en el conjunto de ordenes necesarias para que poner el robot manipulador en movimiento con objeto de interaccionar, por medio de su efector final, con otros elementos del entorno de trabajo. Dichos desplazamientos no tienen por qué ser inamovibles, pueden depender de las decisiones que tome el sistema basándose en la lectura de los sensores externos, de la comunicación con otros componentes automáticos o incluso con un operador humano.

En el caso de la programación de un robot el tiempo de ejecución es esencial y algunas variables pueden contener datos que dependan de las lecturas de los sensores externos. Por otro lado el programa, además de definir la tarea, debe controlar que esta se realice sin errores. Generalmente el robot no trabaja en solitario, convive con otros elementos como máquinas, cintas transportadoras e incluso otros robots.

La sincronización es fundamental por lo que para que la tarea finalice correctamente si ciclo de movimiento, su respuesta ante eventos externos o la comunicación deben cumplir ciertas restricciones de tiempo real. Por otro lado tenemos que tener en cuenta al interaccionar con el mundo real, pueden ocurrir fallos como la caída del objeto que transportaba la pinza o colisiones por las imprecisiones a la hora de ejecutar un movimiento.

Los programas tienen que tener en cuenta estos imprevistos y ser capaces de recuperar la ejecución de la tarea en la medida de lo posible, cuando ocurran.

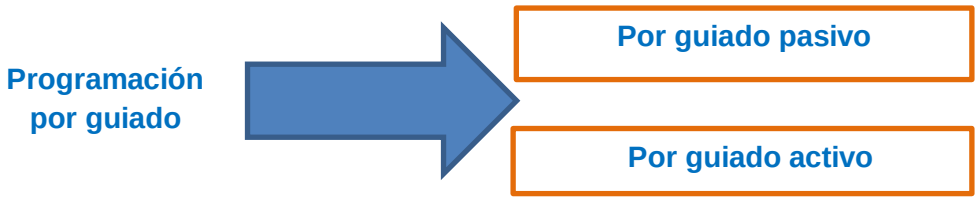


Es importante que la tarea se pueda codificar de manera eficiente y que proporcione facilidades para el desarrollo de la misma como unas herramientas adecuadas para la depuración y validación del programa. Basándonos en lo anterior, podemos definir dos técnicas para la programación de un robot manipulador: **por guiado y textual.**

5.1 Programación por guiado

En este tipo de programación, el propio brazo interviene en el trazado del camino y en las acciones a desarrollar en la tarea de la aplicación. Esta característica determina, inexcusablemente, la programación "on-line".

La programación por guiado se subdivide en dos clases:



5.1.1 Guiado pasivo

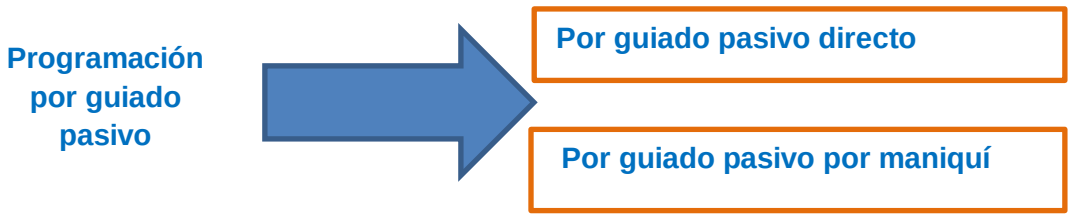
La programación por guiado pasivo se subdivide a su vez en:

- **Guiado pasivo directo**

En este caso, el programador puede tomar el extremo del robot y llevarlo hasta los puntos deseados a través de las trayectorias más adecuadas. La unidad de control del robot registra de manera automática la señal de los sensores de posición de las articulaciones en todos los puntos recorridos.

La técnica de guiado pasivo directo se utiliza, por regla general, en labores de pintura. El operario conduce la muñeca del manipulador o del brazo maestro, determinando los tramos a recorrer y aquellos en los que la pistola debe expulsar una cierta cantidad de pintura. Con esta programación, los operarios sin conocimientos de "software", pero con experiencia en el trabajo a desarrollar, pueden preparar los programas eficazmente.

La programación por guiado pasivo directo tiene pocas posibilidades de edición, ya que, para generar una trayectoria continua, es preciso almacenar o definir una gran cantidad de puntos, cuya reducción origina discontinuidades. El "software" se organiza, aquí, en forma de intérprete.



- **Guiado pasivo por maniquí**

La dificultad física de mover toda la estructura del robot se resuelve a través de este procedimiento. En este caso se dispone de un doble del robot, mientras que éste permanece fuera de línea. El doble posee una configuración idéntica que el robot real, pero es mucho más ligero y fácil de mover. La programación se realiza llevando de la mano a este doble, mientras que la unidad de control muestrea y almacena con cierta frecuencia los valores que toman los sensores de posición de las articulaciones, para su posterior repetición por el robot.

La programación, usando el guiado pasivo por maniquí, consiste en determinar las acciones y movimientos del brazo manipulador, a través de un elemento especial para este cometido. En este caso, las operaciones ordenadas se sincronizan para conformar el programa de trabajo.

5.1.2 Guiado activo

Esta posibilidad permite emplear el propio sistema de accionamiento del robot, controlado desde una botonera o bastón de mando (conocido como joystick) para que sea éste el que mueva sus articulaciones.

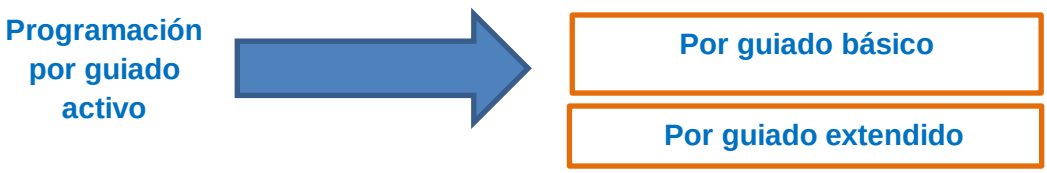
Atendiendo a la potencia del sistema, se habla de guiado básico y guiado extendido.

Guiado Básico

El robot es guiado por los puntos por los cuales se desea que pase durante la fase de ejecución automática del programa. Durante ésta, la unidad de control interpola dichos puntos según determinadas trayectorias. Muchas veces no es posible incluir ningún tipo de estructuras de control dentro del programa, por lo que los puntos son recorridos siempre secuencialmente, en el mismo orden que se programaron. Un ejemplo de este tipo de programación es la utilizada en casi todos los robots de pintura, donde la unidad de control muestrea automáticamente los puntos recorridos por el robot con una frecuencia muy alta.

Guiado Extendido

Permite especificar, junto a los puntos por los que deberá pasar el robot, datos r elativos a la velocidad, tipo de trayectoria, precisión con la que se quiere alcanzar los puntos, control del flujo del programa, atención a entradas/salidas binarias, etc. En este caso, el método guiado de utilizado es el de la botonera o joystick. El guiado por extendido aumenta la potencia del sistema de programación.



Dependiendo del algoritmo de control que se utilice, el robot pasa por los puntos finales de la trayectoria enseñada. Hay que tener en cuenta que los dispositivos de enseñanza modernos no sólo permiten controlar los movimientos de las articulaciones del manipulador, sino que pueden, también, generar funciones auxiliares, como:

- Selección de velocidades

- Señalización del estado de los sensores

- Borrado y modificación de los puntos de trabajo

- Funciones especiales

Al igual que con la programación por guiado pasivo directo, en la que se emplea un elemento de enseñanza, el usuario no necesita conocer ningún lenguaje de programación. Simplemente, debe habituarse al empleo de los elementos que constituyen el dispositivo de enseñanza. De esta forma, se pueden editar programas, aunque como es lógico, muy simples.

La estructura del "software" es del tipo intérprete; sin embargo, el sistema operativo que controla el procesador puede poseer rutinas específicas, que suponen la posibilidad de realizar operaciones muy eficientes.

Ventajas de la programación por guiado:

-Son fáciles de aprender.

-Requieren de un espacio de memoria relativamente pequeño para almacenar la información.

Inconvenientes de la programación por guiado:

-Necesidad de utilizar al propio robot y su entorno para realizar la programación, lo que obliga a sacar al robot de la línea de producción e interrumpir ésta.

- Inexistencia de una documentación del programa.

- Falta de adaptabilidad en tiempo real con el entorno.

- No pueden tratar con facilidad interacciones de emergencia.

- Dificultad de realizar modificaciones en el mismo.

5.2 Programación textual

En este caso la programación está formada por un texto de instrucciones o sentencias, para cuya elaboración no es necesaria la intervención del robot; es decir, permite indicar la tarea al robot a través de un lenguaje de programación específico. Con la programación textual el operador no define, prácticamente, las acciones del brazo manipulador, sino que se calculan, en el programa mediante el uso de las instrucciones textuales adecuadas.

Por ejemplo, en una aplicación de ensamblaje de piezas, en la que se requiere una gran precisión, los posicionamientos seleccionados mediante la programación por guiado no son suficientes, debiendo ser sustituidos por cálculos más perfectos y por una comunicación con el entorno que rodea al sistema.

En la programación textual, la posibilidad de edición es total. El robot, sólo interviene en la puesta a punto final.

Dependiendo del lenguaje utilizado, pueden elaborarse programas de distinta complejidad, con inclusión de saltos condicionales, empleo de bases de datos, posibilidad de creación de módulos operativos intercambiables, capacidad de adaptación a las condiciones del mundo exterior….

Dentro de la programación textual, podemos distinguir dos grupos de características completamente distintas:

5.2.1 Programación textual explicita

En este grupo, el programa consta de una secuencia de órdenes o instrucciones concretas, que van definiendo con rigor las operaciones necesarias para llevar a cabo la tarea.

La programación explicita engloba a los lenguajes que definen los movimientos punto a punto, similares a los de programación por guiado, pero formulado en lenguaje

formal. En este tipo, el tratamiento de las situaciones anormales, colisiones…queda a cargo del programador.

Dentro de la programación explicita, hay dos niveles:

- **Nivel de movimiento elemental**

  Abarca los lenguajes dirigidos a controlar los movimientos del brazo manipulador. Se subdividen a su vez en dos tipos:

  - **Articular.** Cuando el lenguaje se dirige al control de los movimientos de las diversas articulaciones del brazo.
  - **Cartesiano.** Cuando el lenguaje define los movimientos relacionados con el sistema de manufactura, es decir, los del efector final.

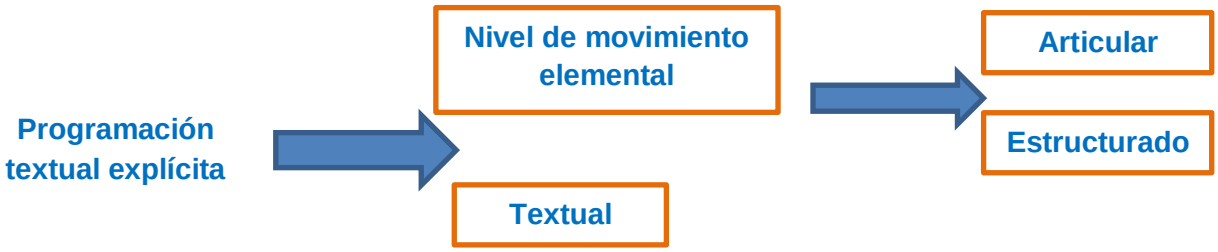
Los lenguajes del tipo cartesiano utilizan transformaciones homogéneas. Esta característica hace al programa muy versátil, independizando a la programación del modelo particular del robot, puesto que un programa, en coordenadas cartesianas, puede utilizarse en otro, con diferentes coordenadas, mediante el sistema de transformación correspondiente.

Por el contrario, los lenguajes de tipo articular indican los incrementos angulares de las articulaciones.

- **Nivel estructurado**

Intenta introducir relaciones entre el objeto y el sistema del robot, para que los lenguajes se desarrollen sobre una estructura formal.

Estos lenguajes describen objetos y transformaciones con objetos, disponiendo, muchos de ellos, de una estructura de datos arborescente.
El uso de lenguajes con programación explícita estructurada aumenta la comprensión del programa, reduce el tiempo de edición y simplifica las acciones encaminadas a la consecución de tareas determinadas.
En los lenguajes estructurados, es típico el empleo de las transformaciones de coordenadas, que exigen un cierto nivel de conocimientos. Por este motivo dichos lenguajes no son populares hoy en día.



5.2.2 Programación textual especificativa

Se trata de una programación del tipo no procesal, en la que el usuario describe las especificaciones de los productos mediante una modelización, al igual que las tareas que hay que realizar sobre ellos.

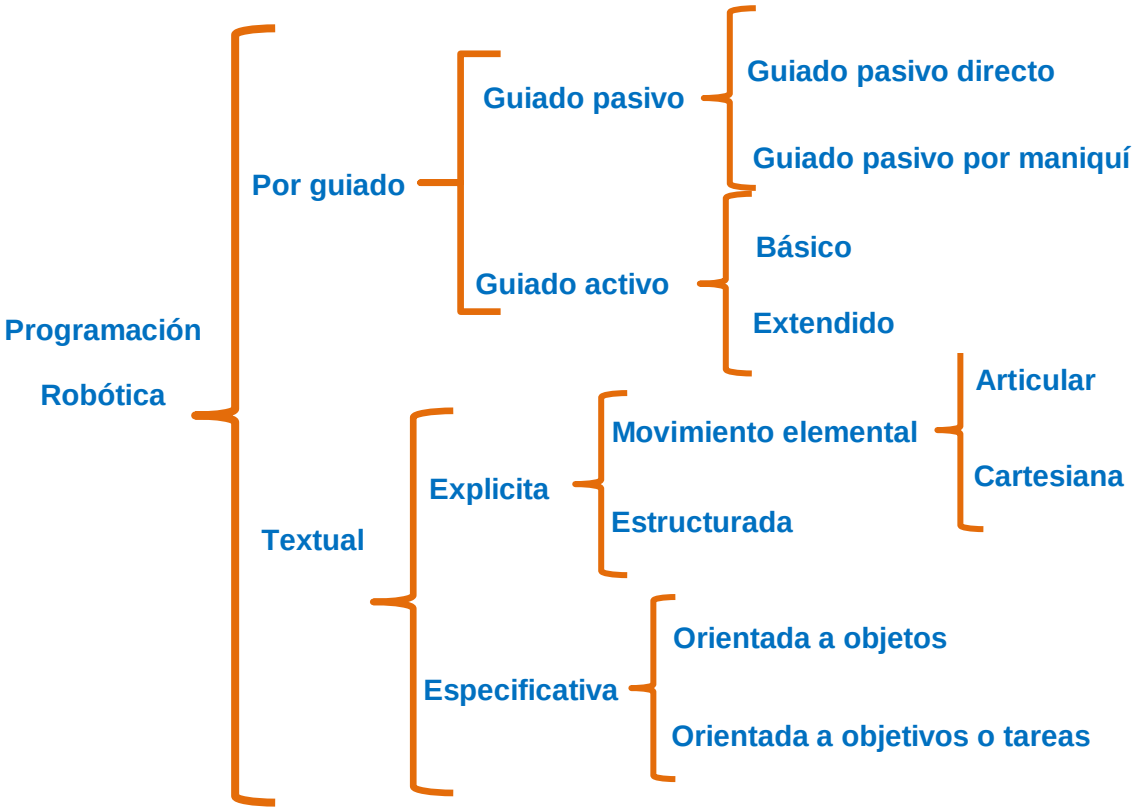
El sistema informático para la programación textual especificativa ha de disponer del modelo del mundo donde se encuentra el robot. Este modelo será, normalmente, una base de datos más o menos compleja, según la clase de aplicación, pero siempre requiere potentes ordenadores para el procesar una gran cantidad de información.

El trabajo de la programación consistirá, simplemente, en la descripción de las tareas a realizar, lo que permite poder realizar trabajos complicados.

Actualmente, los modelos del mundo son del tipo geométrico. Dentro de este tipo programación, hay dos clases dependiendo de si el modelo está orientado a objetos o a objetivos.

Si el modelo se orienta al nivel de los objetos, el lenguaje trabaja con ellos y establece las relaciones entre ellos. Teniendo en cuenta la imprecisión de los cálculos del ordenador y de las medidas de las piezas, se precisa de una ejecución previa, para ajustar el programa al entorno del robot.

Los lenguajes con un modelo del universo orientado a los objetos son de alto nivel, permitiendo expresar las sentencias en un lenguaje similar al usado comúnmente.

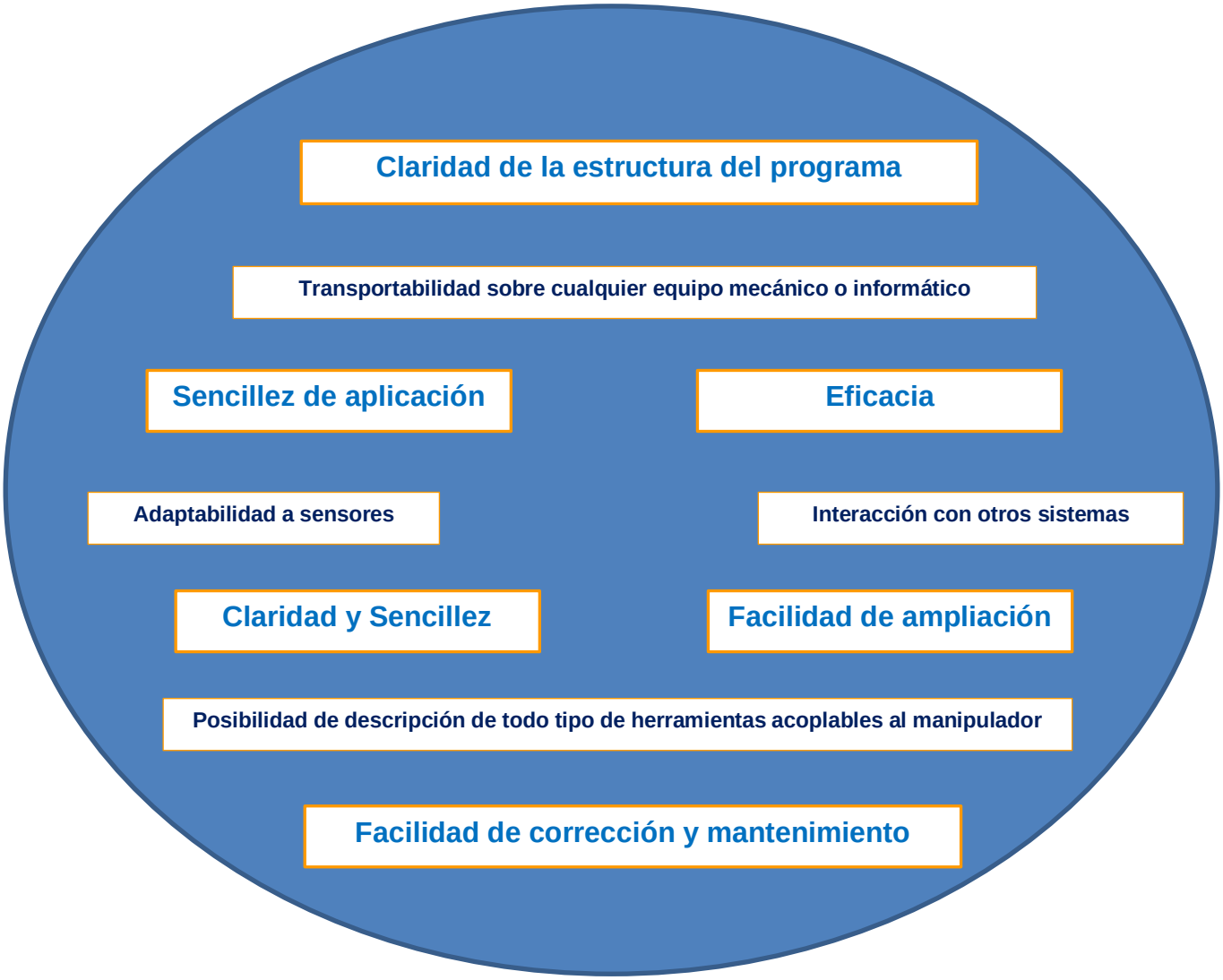


Las características básicas de un sistema óptimo de lenguaje son:

- Claridad y sencillez
- Claridad de la estructura del programa
- Sencillez de aplicación
- Facilidad de ampliación
- Facilidad de corrección y mantenimiento
- Eficacia

A estas características habría que añadir:

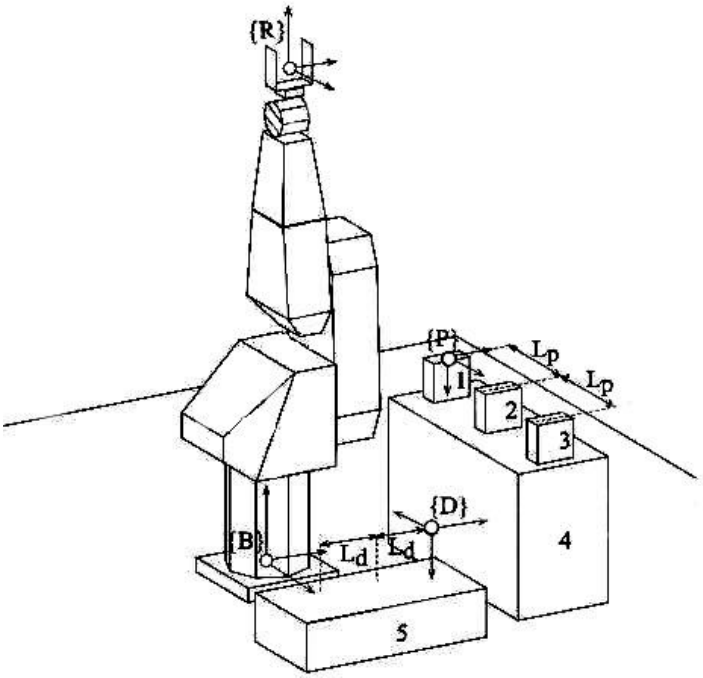
- Transportabilidad sobre cualquier equipo mecánico o informático
- Adaptabilidad a sensores
- Posibilidad de descripción de todo tipo de herramientas acoplables al manipulador
- Interacción con otros sistema



EJEMPLO

A continuación describimos un ejemplo que es el que vamos a utilizar durante el resto del trabajo.

Disponemos de un brazo manipulador RX90 al lado del cual tenemos cinco cajas, numeradas del 1 al 5. Las cajas 4 y 5 actúan a modo de plataforma donde en su parte superior se pueden colocar las cajas 1,2 y 3.



Disponemos también de cuatro sistemas de coordenadas cuyos orígenes se han marcado con círculos y que se utilizarán como referencias para definir los movimientos del brazo robot y sus relaciones:

{B} Sistema asociado a la base del robot, a partir del cual se referirán el resto de las localizaciones empleadas en este ejemplo.

{R} Sistema utilizado para definir la posición de descanso del robot.

{P} Sistema que establece la posición y la orientación para el agarre de la caja número 1 y que se utiliza para definir los mencionados datos de agarre para las cajas 2 y 3.

{D} Sistema que especifica la localización de destino donde se trasladará la caja 1.

Descrito lo anterior nos proponemos la tarea de desplazar las cajas 1, 2 y 3 de la caja 4 a la 5. Con la programación a **nivel de tarea** el operador usaría una sentencia como la siguiente:

Traslada las cajas 1, 2 y 3 que se encuentran sobre la caja 4 a la caja 5

Se observa que el programa está expresado en un lenguaje muy cercano al humano, y usa un modelo cualitativo del entorno; no explica exactamente donde se encuentran las cajas que tiene que trasladar, solo indica una relación (*sobre*) que las localiza dentro del entorno.

En el caso del **nivel de acción,** se supone una acción del robot que podemos identificar como ***apilar caja en destino***, donde el primer parámetro indica el número de la caja afectada y el segundo el destino del movimiento, que en nuestro caso será siempre la caja 5. La tarea antes enunciada se describiría asi:

apilar 1 en 5
apilar 2 en 5
apilar 3 en 5

De nuevo se emplea un modelo cualitativo del mundo, al igual que en el anterior nivel se indica de forma cualitativa que el origen de los desplazamientos se encuentra sobre la caja 4 y el destino sobre la caja 5. Esto hace que a los niveles de programación ***de tarea y de acción*** se les denomina de forma genérica **programación implícita.**

La conversión a lenguaje robot de las acciones descritas arriba, implica el uso del modelo geométrico definido por los sistemas {B}, {R}, {P} y {D} para convertir dichas acciones al ámbito cuantitativo del mundo real.

Se puede convertir las acciones en el siguiente conjunto de instrucciones del robot:

```
funcion APILAR (caja, destino)
        Definir (P,D,R)
Origen:=Desplaza (P,[Lp*(caja-1),0,0])
Destino:=Desplaza (D,[Lp*(caja-1),0,0])
AbrirPinza
Mueve (Origen)
CerrarPinza
Mueve (Destino)
AbrirPinza
Mueve (R)

fin de función
```

En el programa anterior se ha creado la orden ***Definir*** para especificar las localizaciones de {P}, {D} y {R}, ***AbrirPinza*** y **CerrarPinza** para abrir y cerrar el efector final, y la función ***Desplaza*** (S,V) para calcular el resultado de desplazar el sistema S sobre sus propios ejes según el vector de posición V. Para finalizar ***Mueve***(T) se usa para mover físicamente el brazo robot desde la localización actual hasta el destino T.

Cada acción **APILAR** ejecuta un ciclo del brazo robot de coger una caja en el origen calculado, trasladarla hasta el destino especificado y volver a la posición de inicio. La programación a nivel robot se califica como programación explícita porque en ella se

detallan cuantitativamente los orígenes y destino de cada movimiento.

En el caso anterior se ha traducido una tarea especificada en un lenguaje casi natural para el ser humano hasta otro de nivel de programación convencional, Es evidente que para el operador resultaría más fácil que este definiera las ordenes a nivel tarea y que de forma automática se generara el programa nivel robot, pero este sería ineficiente ya que se emplearía más código del necesario y el robot pasa por la posición {R} de inicio en cada ciclo de trabajo. Sería mucho más adecuado programar de forma directa a nivel robot:

```
funcion APILAR2
    Definir (P,D,R)
    Para i=1 hasta 3 hacer
        Origen:=Desplaza (P,[Lp*(caja-1),0,0])
        Destino:=Desplaza (D,[Lp*(caja-1),0,0])
       AbrirPinza
      Mueve (Origen)
     CerrarPinza
    Mueve (Destino)
    AbrirPinza
fin Para
     Mueve (R)
fin de función
```

Con la función ***APILAR2,*** se evita la llamada, de forma repetida, a la función ***APILAR*** con el uso del bucle, y el robot sólo vuelve a la posición de inicio cuando ha terminado su tarea.

5.2.3 Lenguajes de nivel robot

Un programa, codificado en un lenguaje a nivel robot, detalla la secuencia de movimientos del brazo robot, partiendo de las localizaciones definidas por el operador y la información captada por los sensores externos. A este respecto, el lenguaje debe tener en cuenta, la posibilidad del uso de funciones básicas del movimiento del brazo, que permitan definir distintas formas de interaccionar con el entorno, así como el uso de las transformaciones homogéneas como elemento fundamental. Por otra parte, de cara a sincronizarse con otros elementos del entorno, tomar decisiones entre alternativas o posicionar el efector final de forma activa tiene que tener en cuenta la información que proporcionan los sensores tanto internos como externos.

Los lenguajes de nivel robot se pueden concebir como una ampliación de otros lenguajes convencionales de programación, modificando estos de forma que se pueda introducir extensiones, definir nuevas estructuras de datos y funcionalidades. Esta concepción tiene la ventaja de que el programador no parte de cero a la hora de utilizar el lenguaje, simplemente tiene que centrarse en el aprendizaje de nuevas funcionalidades.

Por otra parte, se puede crear un nuevo lenguaje especializado que facilite la programación de tareas robóticas específicas como son el movimiento del brazo o la lectura de datos por medio de los sensores.

En cualquier caso, el entorno de programación influye de manera decisiva en la productividad de la mencionada actividad. En este sentido se puede diseñar el lenguaje del robot en forma de intérprete o de compilador. En el primer caso el entorno resulta más interactivo y es más sencillo tanto la ampliación como la depuración, sin embargo los lenguajes compilados son más eficientes a la hora de la ejecución.

Se puede clasificar la evolución de los lenguajes de nivel robot en tres generaciones:

- La **primera** solo contempla la programación por guiado, por lo que el programa del robot se limita a reproducir una secuencia de movimientos previamente establecida.

- Con la **segunda** se obliga al programador a tener en cuenta las localizaciones para definir movimientos del brazo robot, este debe asegurarse de definir de forma correcta las localizaciones y sus relaciones de forma que la tarea se lleve a cabo con éxito.

- En la **tercera** generación se utilizan operaciones referidas a los objetos del entorno, liberando al programador de la responsabilidad de tener que diseñar el programa en función de las localizaciones.

5.2.4 Lenguaje V+

En este apartado se van a definir las operaciones e instrucciones más comunes incluidas en los lenguajes de nivel de robot de segunda generación, que son los más comunes en el ámbito de los robots industriales. Vamos a utilizar para ello el lenguaje V+ comúnmente utilizado en brazos manipuladores.

Variables y posiciones del robot

En los lenguajes de nivel robot se define un a grupo menor de tipos de variables y constantes con respecto a los lenguajes convencionales. A diferencia de estos, no se declaran si no que se les asigna un valor y un tipo determinado a la hora de crearlos.

Al igual que en la programación convencional se usan variables numéricas enteras o reales, cadenas de caracteres y vectores.

Se referencian mediante un identificador alfanumérico que comienza siempre por una letra o por el símbolo $ en el caso de las cadenas de caracteres.

Sobre las variables numéricas y los componentes de los vectores se pueden ejecutar operaciones aritméticas, relacionales y binarias igual que en los lenguajes de programación, sin embargo para las cadenas de caracteres existe un conjunto de funciones específicas

Los lenguajes de nivel robot cuentan con una clase especial de variable llamada posiciones del robot. Esta se usa tanto para definir las posiciones que tiene que alcanzar el efector final como para las ubicaciones de los objetos del entorno con los cuales va a interaccionar. Existen 2 formas de precisar estos dos tipos de variables: las localizaciones y los puntos de precisión.

Las l**ocalizaciones** indican la posición y la orientación de un nuevo sistema de coordenadas con respeto al utilizado de referencia que suele ser el sistema {0}, asociado a la base del robot. Se suelen crear con la función *TRANS*, que posee seis argumentos numéricos de entrada, los tres primeros indican el vector de posición en milímetros del origen del nuevo sistema de referencias y los tres últimos su orientación en una representación de ángulos ZYZ. Además tienen asociadas una serie de funciones, entre las que destacan el operador *(:),* cuyo significado es el análogo a la multiplicación de matrices homogéneas, lo que permite la especificación de sistemas de referencias locales y el establecimiento de relaciones geométricas entre ellos.

No se puede acceder directamente a los elementos de una localización como si se tratasen de los componentes de un vector. Es necesario el uso de ciertas funciones especializadas para ello. V+ dispone de DX, DY y DZ que devuelven cada una de las coordenadas cartesianas del vector de posición de una localización o bien de forma general, DECOMPOSE, que retorna, como vector, los seis parámetros que la definen.

Terminando con las variables de tipo localización, comentar que existen funciones diseñadas específicamente para las operaciones de escalado y de traslación.

Los **puntos de precisión** sirven para definir una postura del robot mediante la especificación de su vector articular, y se caracterizan porque su identificador comienza por el carácter *(#),* para crear una variable de este tipo, se usa la función #PPOINT a la que hay que proporcionar seis argumentos de entrada numéricos, que representan una postura del robot.

Este tipo de variables se encuentran ya definidas en el ámbito articular, por lo que el sistema no tiene que aplicar el modelo cinemático inverso para usarlas, evitando así errores inherentes a este tipo de cálculos. A causa de esto mismo, también podemos decir que no posee la ambigüedad de las transformaciones, pero no se pueden

componer como estas últimas y resultan muy dependientes del brazo robot. De las funciones descritas para las localizaciones, sólo admiten el uso de **DECOMPOSE**.

Por último destacar que todas las variables creadas y utilizadas dentro de un programa poseen un ámbito global, es decir, se pueden invocar y modificar desde cualquier otra rutina que se encuentre ubicada en la memoria de la unidad de control del robot.

Se puede anular esta característica para un conjunto concreto de variables, utilizando la orden **LOCAL** acompañada de la lista de variables que se desea que queden para uso exclusivo en la rutina que las creó.

Movimiento del brazo robot y actuación de la pinza.

En este apartado englobamos las ordenes que hacen que el brazo interaccione con su entorno de trabajo y la actuación de su efector final. Estas órdenes de movimientos contemplan la generación de trayectorias articulares y cartesianas. En lo que respecta al efector final, se refieren a la apertura y cierre de la pinza, lo que se realiza con las sentencias, **OPENI Y CLOSEI**.

Las funciones básicas de movimiento son **MOVE y MOVES**, a las que se les asigna como único argumento de entrada una variable de tipo localización (T) que indica donde debe situarse el efector final:

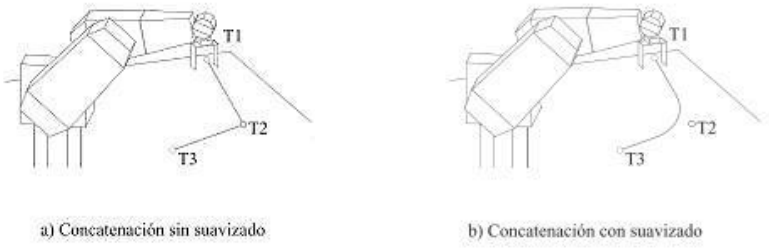
MOVE mueve el brazo mediante una trayectoria articular mientras que **MOVES** utiliza una trayectoria cartesiana.

MOVE T: Movimiento articular hacia T

MOVES T: Movimiento cartesiano hacia T

Por regla general, el lenguaje robot cuando se encuentra más de una orden de movimiento consecutiva en un programa, no efectúa las paradas intermedias, sino que realiza la ejecución de la secuencia como si se tratara de un solo movimiento. Como resultado de esto la trayectoria global construida interpola las localizaciones inicial y final, y aproxima las intermedias. Si se desea que el sistema no concatene el tramo actual con el siguiente se usa la sentencia **BREAK** a continuación de la orden de movimiento.

EJEMPLO: La siguiente imagen muestra dos ejemplos de una trayectoria cartesiana que parte de T1, pasa por T2 y termina en T3. T1 representa la localización inicial de la pinza, T2 y T3 se ha definido con la función TRANS. En la imagen de la izquierda (a) Concatenación sin suavizado) se ilustra el efecto de la sentencia BREAK de forma que el movimiento de la pinza dibuja 2 segmentos rectilíneos



a) Concatenación sin suavizado

b) Concatenación con suavizado

Efecto de la sentencia BREAK en la concatenación de trayectorias

El código que generaría esta trayectoria sería:

MOVES T2
BREAK
MOVES T3

Si eliminamos la sentencia **BREAK**, tendríamos una trayectoria con suavizado, en este caso el brazo concatena los dos segmentos anteriores, y para efectuar la transición entre ambos, suaviza su punto común de unión en T2, pasa por esta localización si hacer parada en ella.

El código que generaría esta trayectoria sería:

MOVES T2
MOVES T3

La velocidad de desplazamiento se define mediante la sentencia **SPEED** con la que se indica al manipulador el porcentaje de la velocidad, considerada como normal, con la que se ejecutarán las órdenes de movimiento.

EJEMPLO: Para hacer que el manipulador se desplace al 20% de la velocidad considerada normal durante el próximo movimiento, la sentencia sería:

SPEED 20

Si queremos que esta velocidad se mantenga durante toda la ejecución del programa, deberíamos escribir:

SPEED 20 ALWAYS

Otro punto importante en los desplazamientos cartesianos es la especificación del tipo de solución del brazo que se tomará para resolver la cinemática inversa, a la que se denomina *control de la configuración del brazo.* Para esto se usan las sentencias:

LEFTY y RIGHTY para elegir entre la solución brazo izquierdo o brazo derecho.

ABOVE o BELOW para indicar si partimos de codo arriba o codo abajo.

FLIP o NOFLIP para indicar si la muñeca se girará o no.

Hay que tener en cuenta que estas órdenes de control de la configuración, sólo determinan la manera en la que el brazo realizará los desplazamientos, y en el caso de que una localización no pueda alcanzarse con la configuración indicada, se generará un error y el sistema no intentará otra alternativa.

Existen también unas órdenes auxiliares de movimiento que facilitan la tarea de manipulación. Estas se utilizan para acercarse hasta o alejarse desde un punto próximo de una localización tomada como referencia para la manipulación de un objeto.

APPRO T, D

Permite al robot aproximar su efector final a una distancia D, expresada en milímetros, de la localización T con un movimiento articular.

APPROS T, D

Permite al robot aproximar su efector final a una distancia D, expresada en milímetros, de la localización T con un movimiento cartesiano.

Por el contrario las órdenes **DEPART** y **DEPARTS** alejan el efector final de su localización actual con un movimiento sobre su eje Z negativo la distancia expresada en milímetros que figura como único argumento de entrada. Tomando como argumento D. las sentencias quedarían como se detalla a continuación:

DEPART D

Aleja al efector final a una distancia D mediante un movimiento articular.

DEPARTS D

Aleja el efector final a una distancia D mediante un movimiento cartesiano.

La orden **READY** es una instrucción especial, que se encarga de llevar al brazo robot a su posición de descanso, definida en grados sexagesimales, mediante un desplazamiento articular.

CODIFICACIÓN TRASLADO CAJAS

Disponemos de un brazo manipulador RX90 al lado del cual tenemos cinco cajas, numeradas del 1 al 5. Las cajas 4 y 5 actúan a modo de plataforma donde en su parte superior se pueden colocar las cajas 1,2 y 3, se pretende trasladas las cajas 1,2 y 3 desde la caja 4 donde están posadas a la caja 5.

Se va a detallar el programa en lenguaje V+ que codifica la tarea descrita donde se puede ver como se declaran las variables, las órdenes de movimiento y de gestión de la pinza del robot.

.PROGRAM PIEZAS

;Definicion de variables de la tarea

SETTBP=TRANS(0,400,400,0,180,90);

SETTBD=TRANS(400,150,200,0,180,0);

LD=150; Separacion en destino

LP=200; Separacion en origen

SD=200; Aproximacion en destino

SP=100; Aproximacion en origen

;Velocidad y control de la configuracion

SPEED 50 ALWAYS; Velocidad

LEFTY ; Brazo izquierdo

ABOVE ; Codo arriba

NOFLIP ; Sin giro

;Bucle principal del programa

FOR I=1 TO 3

;ETAPA 1: Agarre de una pieza

APPRO TBP,SP ;Aproximación al agarre

BREAK

```
MOVES TBP  ;Ir al agarre

CLOSEI;

DEPARTS SP   ;Alejarse del agarre

;ETAPA 2: Traslado de la pieza

APPRO TBD,SD ;Aproximación a destino

BREAK

MOVES TBP ; Ir a destino

OPENI;

DEPARTS S ;   Alejarse del destino

;ETAPA 3:  Origen y destino de la próxima pieza

SET TBP= SHIFT(TBP BY LP,0,0)

SETTBD=SHIFT(TBP BY 0,-LD,0);

END

READY

.END
```

Todos los programas en V+ empiezan con la sentencia .PROGRAM seguida del nombre del programa y termina con .END

Concurrencia y gestión de señales asíncronas

La concurrencia se refiere a la capacidad de un sistema informático para simultanear varias acciones. Desde el enfoque de la programación es importante la concurrencia de alto nivel que permite la ejecución simultánea de varios programas de usuario y la gestión de forma asíncrona de señales recogidas por los sensores externos. Al añadir la capacidad de generar señales de salida, el robot podrá comunicarse y sincronizarse de forma eficiente con el resto de los elementos del entorno.

El lenguaje V+ permite la concurrencia de un programa de control del robot con un conjunto de programas de control de procesos. Las señales digitales de entrada y salida se gestionan con dos sentencias básicas: **SIG** para leer el estado de una señal de entrada o salida y **SIGNAL** para producir una salida.

Clasificación de las señales:

Señales de entrada desde el exterior: Comprendidas en el intervalo [1……64], su estado se lee con la función **SIG**

Señales de salida externa: Comprendidas en el intervalo [1001……1064], su estado puede leerse con **SIG** o bien cambiarlo por **SIGNAL**

Señales internas de solo lectura: se usan igual que las de entrada desde el exterior, su intervalo [3001…..3032]

Señales internas de lectura y escritura: Se manejan de la misma forma que las de salida externa, son la incluidas en el intervalo [2001……2032]

La función SIG devuelve el valor cierto o falso resultado de realizar la operación AND sobre todas la señales binarias cuyos identificadores aparecen como argumentos de entrada.

La sentencia **SIGNAL** se utiliza para cambiar el estado de las señales de salida externas o internas de lectura y escritura, que figuran en su lista de parámetros de entrada. Si un identificador va seguido de un signo negativo indica la desactivación de la señal referenciada, si es positiva su activación.

Con el objeto de que el robot se sincronice con eventos externos, existen tres sentencias básicas: **DELAY, WAIT y REACTI.**

DELAY detiene el movimiento del robot durante los segundos que aparecen el su único parámetro de entrada.

WAIT seguida de una expresión lógica, introduce el programa en un bucle de espera hasta que el resultado de la evaluación de su argumento de entrada sea cierto.

REACTI se utiliza para monitorizar las señales de entrada, tanto externas como internas. Sus argumentos de entrada son el identificador de señal (puede llevar un signo negativo) y el nombre de la rutina que se desea ejecutar.

EJEMPLO

La escena consta de un brazo robot, de una cinta transportadora y una plataforma de paletizado, el sistema emplea también dos sensores externos y digitales de presencia, uno de barrera, situado en un extremo de la cinta transportadora y otro de proximetría situado en la base de la pinza del robot. La tarea consiste en el traslado de las tres piezas, que avanzan sobre la cinta transportadora en dirección al sensor de barrera, a la plataforma de paletizado.

La tarea comienza en el momento que el robot pone en funcionamiento la cinta transportadora mediante el uso de las señales de salida 1 y 2.

SIGNAL 1, 2

Con la primera activa su motor y con la segunda se establece el sentido de la marcha. Esta acción produce el movimiento de las piezas, que se detendrán cuando la primera de ellas sea detectada por el sensor de barrera asociado a la señal de salida 1002.

```
WAIT SIG(1002)
SIGNAL -1
```

Para localizar y coger la pieza, el robot, primero desplaza la pinza a la localización denominada BARRIDO.

```
SET BARRIDO=TRANS (300, 170, 280, 0. 180, 0)
```

Y ahora se realiza un movimiento articular de aproximación sobre la vertical, en este caso se ha tomado 120 mm como distancia de aproximación segura

```
APRO BARRIDO, 120
MOVES BARRIDO
```

En este punto la pinza y el sensor de proximetría apuntan hacia abajo y perpendicular a la cinta. El robot comienza un movimiento rectilíneo con el objeto de detectar la presencia de una caja mediante su sensor de proximetría, cuando esto ocurre se detecta un flanco de subida, que indica que la pinza se encuentra encima de la caja y se activa la función AGARRE

```
REACTI 1001, AGARRE
MOVES BARRIDO: TRANS (-200,0,0,0,0,0)
```

Ahora solo resta cerrar la pinza y subir la caja 150 mm sobre la cinta transportadora.

```
.PROGRAM AGARRE
        MOVES SHIFT (HERE BY 40,0,0)
        APPROS HERE, -80
        CLOSEI
        DEPARTS 150
.END
```

La localización de destino de la caja se define:

```
SET DESTINO=TRANS (0, 400, 0, 180, 90)
```

Para trasladar la caja que porta el robot en su pinza, basta con realizar una secuencia de aproximación a DESTINO, abrir la pinza y alejarse.

```
APPRO DESTINO, 100
MOVES DESTINO
OPENI
DETARTS 100
```

CODIFICACIÓN PALETIZADO

A continuación se detalla el programa PALETIZADO que realiza la tarea completa. Comentar que al finalizar el bucle principal para el traslado de las cajas, se retorna el brazo a su posición de descanso y se desactivan todas las señales de salida mediante la orden RESET.

```
.PROGRAM PALETIZADO
SET BARRIDO=TRANS(300,170,280,0,180,0)
SET DESTINO=TRANS(0,400,400,0,180,90)
SPEED 50 ALWAYS
LEFTY
ABOVE
FLIP
SIGNAL 1, 2
FOR I= 1 TO 3
WAIT SIG(1002)
SIGNAL -1
APPRO BARRIDO, 120
MOVES BARRIDO
REACTI 1, AGARRE
MOVES BARRIDO:TRANS(-200,0,0,0,0,0)
APPRO DESTINO, 100
MOVES DESTINO
OPENI
DEPARTS 100
DESTINO=SHIFT(DESTINO,200,0,0)
SIGNAL 1
END
READY
RESET
.END
```

El programa se ha completado con la especificación de la velocidad del robot y las órdenes para el control de la configuración.

6 SIMULACION CINEMÁTICA

6.1 Sistema de simulación gráfica

La simulación gráfica en tres dimensiones de la cinemática del movimiento de un robot manipulador permite un análisis previo de la ejecución de una tarea antes de programarla en el robot real. Mediante este análisis podemos sacar conclusiones sobre el tipo de trayectoria o identificar posibles colisiones.

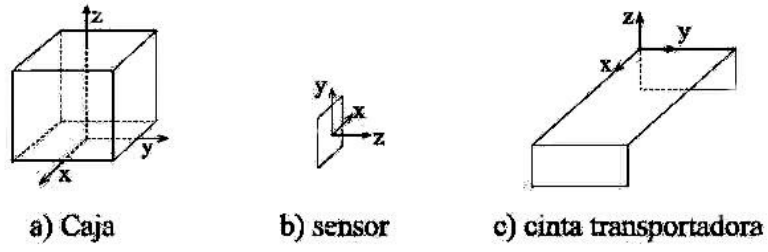
La principal ventaja que aporta este tipo de simulación es que para ciertos pasos del desarrollo de una aplicación no es necesario el uso del robot, evitando así un desgaste innecesario del mismo.

En este capítulo vamos a desarrollar un simulador cinemático del brazo manipulador RX90 construido a partir de las funciones desarrolladas en MATLAB en este trabajo. Con este se pretende tanto ilustrar los aspectos de la programación a nivel robot, como integrar todos los conceptos expuestos sobre la cinemática de los robots manipuladores.

6.2 Funciones del simulador

Este simulador cinemático se basa en una serie de funciones desarrolladas en MATLAB y que se ejecutan dentro de este paquete software, bien desde la ventana de comandos o en el interior de un programa que codifica la tarea del robot. Por otro lado, requiere que se especifique la geometría de los distintos cuerpos que componen el entorno de trabajo del robot. El resultado de la simulación lo constituye la animación cinemática, dentro de una ventana gráfica del robot en interacción con los objetos de su entorno.

Contamos con tres tipos de objetos, caja, sensor y cinta transportadora, que pueden forman parte de la escena del robot.



Tipos de objetos que forman parte del entorno del robot

Todos ellos cuentan con un sistema de coordenadas asociado, de forma que la posición cartesiana de este último con respecto al sistema {0} de la base del robot, se

usa para ubicarlos en el entorno del trabajo. En cuanto a la orientación de las cajas y las cintas transportadoras sólo, permiten un giro sobre los ejes z de sus sistemas solidarios, de manera que los respectivos ejes x e y siempre permanecerán paralelos a los homónimos del sistema {0}

El papel que juegan estos tres elementos en la simulación se describen a continuación:

Caja: Elemento que puede coger, soltar o trasladar el robot por medio de su pinza. El tamaño se define mediante la longitud de sus tres dimensiones, y su localización en la escena se establece mediante la posición cartesiana y el giro sobre el eje z de su sistema asociado.

Sensor: Simula el comportamiento de un sensor digital de proximetría para la detección de objetos de tipo caja. El alcance para la detección se establece en milímetros y con dirección y sentido del eje z del sistema asociado. Su estado es 0 cuando no detecta y 1 cuando si detecta.

Cinta transportadora: Utilizada para mover en la dirección de su eje x todos los elementos de tipo caja posados sobre ella. Trabaja en función de dos señales de salida, una señal para activarla y la otra para decidir el sentido de la marcha: positivo o negativo del eje x. Su tamaño se define mediante sus dimensiones en las tres coordenadas.

Todos los elementos que forman la escena, junto con otros parámetros del simulador, se recogen en un fichero de texto usado para inicializar el simulador.

Este se compone de dos secciones, una obligatoria y otra optativa. En la primera de definen los parámetros del simulador, y en la segunda los elementos de la escena.

A continuación, describimos la primera parte del fichero utilizado para especificar la escena donde se establece el punto de vista del observador, el rango de detección del sensor de proximetría que porta el robot en su pinza, la velocidad de desplazamiento del robot y la sensibilidad de la pinza.

Los parámetros que deben sustituirse por valores numéricos aparecen encerrados entre símbolos menor que (<) y mayor que (>), su significado se describen en la siguiente tabla.

```
Observador (giro, elevación, apertura):
<GR><EL><AP>
Sensor del robot (rango en mm.):
 <SR>
Velocidad del robot (de 0 a 1):
<VR>
Sensibilidad de la pinza (distancia en mm.):
<SP>
```

Fichero de configuración (1º parte)

Parámetro	Descripción
GR, EL, AP	Giro, elevación y apertura de la lente, de forma respectiva.
SR	Establece el alcance, en milímetros del sensor de proximetría que posee el manipulador coincidente con el sistema de la garra.
VR	Establece el intervalo con el cual se extraen las muestras de las funciones de interpolación cartesianas y articulares. Se define como un número real entre cero y uno. Cuanto más cerca de uno, menos posiciones intermedias del robot se dibujarán en la pantalla, y por tanto, se tendrá más sensación de rapidez en los movimientos.
SP	Establece la sensibilidad de la pinza en milímetros. Esta indica la distancia máxima del sistema a la que se puede encontrar el sistema de referencia asociado a una caja, para que al cerrar la pinza esta se considere cogida por el manipulador.

Descripción de los parámetros que forman parte del fichero de configuración (1º parte)

La parte opcional del fichero de configuración se detalla inmediatamente después de la primera parte (como se comentó anteriormente). Su misión se define como la de detallar todos los elementos que conformarán el espacio de trabajo del robot. Consta de 4 partes, en el primero se definen unas plantillas de tamaño y color; y los tres siguientes para especificar las cajas, los sensores y cintas transportadoras. Quedando la segunda parte del fichero de la siguiente forma:

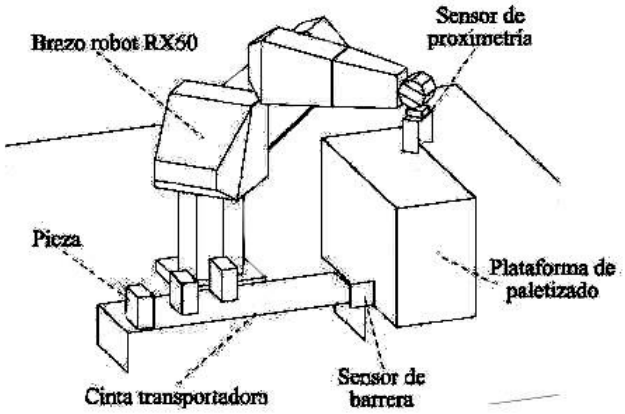
```
Tipos de cajas (lx, ly,lz, color): <NT>
<LX><LY><LZ><CL>
Cajas (posx, posy, posz, giro, tipo): <NC>
<PCX><PCY><PCZ><GC><TC>
Sensores (posx, posy, posz, giro, rango):<NS>
<PSX><PSY><PSZ><GS><TS>
Cinta trans. (posx, posy, posz, giro,tipo, velocidad): <NCT>
<PCTX><PCTY><PCTZ><GCT><TCT><VCT>
```

Fichero de configuración (2º parte)

Parámetro	Descripción
NT	Número de plantillas especificadas para definir el tamaño y el color de las cajas y de las cintas transportadoras.
LX LY LZ	Longitud de los ejes X, Y y Z
CL	Color de la caja
NC	Número de cajas que aparecen en la escena
PCX PCY PCZ	Establecen la posición cartesiana. el cuarto, y el quinto, su alcance en milímetros.
GC	Orientación medida como un giro sobre el eje Y del sensor
TC	Número de orden de la plantilla de tamaño y color especificada
NS	Número de sensores ubicados en el entorno de trabajo sin contar el sensor que tiene el robot en la pinza
PSX PSY PSZ	Establecen la posición cartesiana
GS	Orientación medida como un giro sobre el eje Y del sensor
RS	Alcance del sensor en milímetros
NCT	Número de cintas transportadoras que aparecen en la escena
PCTX PCTY PCTZ GCT	Parámetros para localizar la cinta en el entorno de trabajo, mediante su posición cartesiana y orientación sobre su eje Z.
TCT	Hace referencia a la plantilla de tamaño y color
VCT	Velocidad de movimiento

Descripción de los parámetros que forman parte del fichero de configuración (2º parte)

EJEMPLO: Dada la siguiente escena



El fichero de configuración quedaría de la siguiente forma:

Primera parte
Observador (giro, elevacion, apertura):
60 20 6
Sensor del robot (rango en mm.):
150
Velocidad del robot (de 0 a 1):
0.05
Sensibilidad de la pinza (distancia en mm.):
100

Segunda parte
Tipos de cajas (lx, ly,lz, color): 3
500 200 300 9
600 200 100 11
40 80 80 44
Cajas (posx, posy, posz, giro, tipo): 4
200 400 1 0 1
350 -100 101 90 3
400 -220 101 90 3
450 -350 101 90 3
Sensores (posx, posy, posz, giro, rango): 1
510 190 120 -90 200
Cinta trans. (posx, posy, posz, giro,tipo, velocidad): 1
500 -400 100 90 2 5

En la **primera parte** aparecen los parámetros que hay que detallar al simulador obligatoriamente. En nuestro caso se establece:

El observador con un ángulo de giro de sesenta grados.
Elevación de veinte grados sobre la horizontal.
Apertura de lente de 6 unidades.
Rango de alcance del sensor de la garra de 50 mm.
Sensibilidad de la pinza fijada en cien milímetros.

La **segunda parte** del fichero de configuración se utiliza para definir la presencia de la cinta transportadora, de las tres cajas posadas sobre ella, además de la caja grande y del sensor de barrera.

En la primera sección Tipos de cajas, se definen tres plantillas de tamaño y de color; la primera para definir la caja grande de la escena, la segunda para la cinta transportadora y la tercera para los elementos que se encuentran sobre esta.

A continuación en la sección Cajas se define la posición, orientación, dimensiones y color de las cuatro cajas que entran a formar parte de la escena. Por último, el sensor de barrera se detalla en la sección Sensores y la cinta transportadora en Cintas Trans.

La función **PROGRAM** es la primera orden que debe contener el fichero que codifica la tarea del robot.

Se utiliza para establecer los valores iniciales de las variables internas del simulador con los datos especificados en un fichero de configuración. En el caso de que dichos valores estuviera almacenados en un fichero llamado entorno.dat, la sentencia sería:

PROGRAM (´entorno.dat´)

El conjunto de funciones implantadas en el simulador, se definen como un subconjunto de las instrucciones de los lenguajes a nivel robot detalladas anteriormente. A continuación se describen los prototipos de cada una de las funciones, desarrolladas en MATLAB, donde por motivos de claridad, se ha utilizado el nombre de las ordenes equivalentes de V+ vistas en el capítulo anterior.

6.3 Posiciones del robot.

Para esto usamos las variables propias de MATLAB, básicamente los escalares y las matrices. Con estas últimas se representan las transformadas homogéneas de las localizaciones del robot. Las funciones que tratan este último tipo de datos son:

Tr= TRANS (x, y, z, alfa, beta, gamma) : Realiza una transformada homogénea a partir de la posición cartesiana x,y,z expresada en milímetros y la representación de la orientación ZYZ expresada en grados.

- **x,y,z:** Posición cartesiana en milímetros.
- **Alfa, beta, gamma**: Representación de la orientación en ángulos ZYZ y en grados.
- **Tr:** Transformación homogénea resultante.

Tf=SHIFT (T,x,y,z): Traslada la transformación T.[el] vector "x,y,z", ambos expresados como parámetros de entrada.

- **T**: Transformación homogénea.
- **x, y, z:** Vector de traslación en milímetros.
- **Tf:** Trasformación resultante de la operación de traslación.

T=HERE: Devuelve la localización actual del efector final del robot en coordenadas homogéneas.

- **T**: Localización actual del efector final.

6.4 Configuración del robot y gestión de la pinza.

Este conjunto de funciones no poseen ni parámetros de entrada ni de salida, se encargan exclusivamente de modificar el estado interno del robot para tenerlo en cuenta a la hora de realizar el próximo movimiento.

- **ABOVE**: Establece la configuración “codo arriba” para seleccionar la solución del modelo cinemático inverso.
- **BELOW**: Configuración “codo abajo” para el mismo propósito.
- **LEFTY**: Selección de la solución ”brazo izquierdo”.
- **RIGHTY**: Establece la configuración ”brazo derecho”.
- **FLIP**: Se selecciona la solución ”muñeca girada”.
- **NOFLIP**: Configuración ”muñeca sin girar”
- **OPENI**: Simula la apertura de la pinza del manipulador, si posee un cuerpo cogido, en cuyo caso lo soltaría.
- **CLOSEI**: Simula el cierre de la pinza, de manera que si un cuerpo de tipo caja se encuentra a una distancia menor de la establecida en el parámetro de sensibilidad, este se dará por cogido. En el próximo movimiento el manipulador lo trasladara sujeto por la pinza.

6.5 Funciones de movimiento.

Por las limitaciones de MATLAB, no se contempla la concatenación de las ordenes de movimiento con suavizado. Es decir, la implantación es similar a emplear una sentencia del tipo BREAK al final de cada desplazamiento de la pinza.

- **MOVE (Tf)**: Simula el movimiento articular del brazo manipulador, desde la postura actual hasta el destino especificado en el argumento de entrada. Durante el movimiento traslada al objeto que porta en la pinza y comprueba el estado de las señales de entrada para lanzar la reacción correspondiente.

- **Tf:** Matriz homogénea o vector articular de seis componentes que especifica el destino del movimiento.

- **MOVES (Tf)**: Similar a la función anterior, sólo que realiza un movimiento cartesiano rectilíneo.

  - **Tf**: Matriz homogénea que especifica el destino del efector final.

- **DEPART (d)**: Aleja el efector final con un movimiento articular de la localización actual del mismo. La localización de destino se encuentra a una distancia ”d” sobre el eje Z negativo de la actual.

  - **d**: Distancia en milímetros que se alejara el efector final sobre su eje Z a partir de la posición actual.

- **DEPARTS (d)**: Similar a la función anterior, sólo que el movimiento se realiza en el espacio cartesiano.

- **APPRO (T,d)**: Posiciona el efector final con un movimiento articular en una localización calculada a partir de una de referencia T que se traslada sobre su eje Z una distancia d negativa expresada en milímetros.

  - **T**: Transformación homogénea de referencia.

  - **d**: Distancia en milímetros que se alejar el efector final sobre el eje Z de T.

- **APPROS(T,d)**: Similar a la función anterior, excepto que el movimiento de aproximación se realiza en el espacio cartesiano.

- **READY**: Mueve, de forma articular, el robot a su posición de descanso, especificada por el vector articular [0,-90,90,0,0,0] en grados.

6.6 Tratamiento de señales asíncronas

Este grupo de funciones posee ciertas particularidades que las diferencian de las correspondientes presentadas en V+. En primer lugar, el número de señales de entrada coincide con el de sensores digitales utilizados en la escena.

De esta manera, la señal uno corresponde al sensor que porta en la garra el manipulador, y se van asignando sucesivamente y en orden, una a una, a cada uno de los sensores declarados en el fichero de configuración.

En nuestro ejemplo al único sensor especificado, se le asigna la señal número dos. En cuanto a las señales de salida, se asignan dos de ellas, en orden, a cada cinta transportadora presente en la escena. A la cinta transportadora especificada, se le asignan las señales de salida uno y dos.

- **SIGNAL (sn)**: Activa o desactiva la señal indicada en el parámetro de entradas n, en función del signo de este último (si es positivo activa la señal, y en caso negativo la desactiva). Estas señales se utilizan para cambiar el estado de movimiento de los objetos de tipo *cinta transportadora,* de forma que la cinta i-ésima tiene asociadas las señales i e i+1. La primera de ellas activa el movimiento, y la segunda especifica el sentido del mismo, si i+1 > 0 el movimiento se produce en sentido positivo del eje X de la cinta; en caso contrario si i+1<0 el movimiento es en sentido negativo sobre el dicho eje.

    - **sn:** Cuando es positiva activa la señal especificada, si es negativa la desactiva.

- **s=SIG(sn):** Devuelve el estado de una señal de entrada. El sistema admite la creación de objetos de tipo sensor, que simulan sensores de proximetría digitales. Esta función devuelve 1 si el sensor ha detectado un objeto, en caso contrario devuelve un 0.

    - **sn:** Número del sensor que se desea comprobar.
    - **s:** Estado del sensor: 0 no detecta, 1 detecta.

- **REACTI (sn,fun):** Establece la función *fun* de reacción que se ejecutará cuando se detecte el cambio del estado de la señal *sn* de entrada especificada. Una vez ejecutada esta orden, se sondea la señal durante toda la ejecución de la tarea del robot, de forma que cuando se detecta el cambio de estado, se interrumpe el movimiento del robot y se pasa a ejecutar la función MATLAB especificada en su argumento de entrada.

Una vez acabada, se continúa con la ejecución del programa principal por la instrucción siguiente de donde se produjo la interrupción.

- **sn**: Número de señal de entrada que se sondeara. Si es negativo este parámetro, se lanzara la reacción en un flanco de bajada. Si es positivo en el flanco de subida.
- **fun**: Cadena de caracteres que indica la función MATLAB que se quiere ejecutar.

- **WAIT (sn):** Detiene el programa del robot hasta que la señal especificada cambia de flanco. Mientras el robot permanece parado, se actualiza la posición de todos los objetos sobre las cintas transportadoras y el estado de todas las señales de entrada. En este estado de espera, se atienden las señales y reacciones especificadas con REACTI

  - **sn:** Número de la señal que se espera que cambie de estado. Si es negativo el parámetro, el movimiento del robot queda suspendido hasta que se produzca un flanco de bajada. En el caso de que sea positivo, se detiene hasta un flanco de subida. Si le valor de entrada es nulo, se espera hasta que se pulse una tecla.

EJEMPLO TRASLADO CAJAS

A continuación describimos como se codifican las tareas del robot con el simulador en lenguaje MATLAB, para ello nos basamos en mismo ejemplo que planteamos en el capítulo anterior: ***"Disponemos de un brazo manipulador RX90 al lado del cual tenemos cinco cajas, numeradas del 1 al 5. Las cajas 4 y 5 actúan a modo de plataforma donde en su parte superior se pueden colocar las cajas 1,2 y 3. Trasladar las cajas 1,2 y 3 desde la caja 4 a la 5"***

Las principales diferencias entre la versión de V+ y la que se detalla a continuación, son la desaparición de las sentencias BREAK y SPEED.

PROGRAM (´entorno.dat´)

% Definición de las variables de la tarea

```
PIEZA=TRANS(0,400,400,0,180,90);
DESTINO=TRANS(400,150,200,0,180,0);
LD=150;     %  Separación en destino
LP=200;     % Separación en origen
SD=200;     % Aproximación en destino
SP=100;     % Aproximación en origen
```

% Control de la configuración

```
LEFTY;    % Brazo izquierdo
ABOVE;   % Codo arriba
FLIP;       % Con giro
```

% Bucle principal del programa

```
for i=1:3,
```

% ETAPA 1: Agarre de una pieza

```
APPRO (PIEZA, SP);     % Aproximación al agarre
MOVES (PIEZA);          % Ir al agarre
CLOSEI;
DEPARTS(SP);             % Alejarse del agarre
```

%ETAPA 2: Traslado de una pieza

```
APPRO (DESTINO, SD);  % Aproximación al destino
MOVES (DESTINO);       % Ir al destino
OPENI;
DEPARTS (SD);             % Alejarse del destino
```

%ETAPA 3: Origen y destino de la próxima pieza

```
PIEZA=SHIFT (PIEZA,LP,0,0);
DESTINO=SHIFT (DESTINO,0,-LD,0);
end

READY;
```

Fichero de configuración´entorno.dat´ sería:

```
Observador (giro, elevacion, apertura):
6020 5
Sensor del robot (rango en mm.):
150
Velocidad del robot (de 0 a 1):
0.05
Sensibilidad de la pinza (distancia en mm.):
100
Tipos de cajas (lx, ly, lz, color):3
500   200   300    9
200   400   100  11
 40     80    80   44
Cajas (posx, posy, posz, giro, tipo):5
200 400      1   0  1
400     0      1   0  2
   0 400   301  0  3
200 400   301  0  3
400 400   301  0  3
```

EJEMPLO PALETIZADO

En este ejemplo sirve para explicar la gestión de las señales asíncronas. En este caso, la instrucción WAIT del simulador es sensiblemente diferente de la empleada en V+. Por otro lado las sentencias SIGNAL y SIG del simulador sólo contemplan un número de señal como argumento de entrada.

La escena consta de un brazo robot, de una cinta transportadora y una plataforma de paletizado, el sistema emplea también dos sensores externos y digitales de presencia, uno de barrera, situado en un extremo de la cinta transportadora y otro de proximetría situado en la base de la pinza del robot. La tarea consiste en el traslado de las tres piezas, que avanzan sobre la cinta transportadora en dirección al sensor de barrera, a la plataforma de paletizado.

```
PROGRAM (´entorno3.dat)

BARRIDO=TRANS(300, 170, 280, 0, 180,0);
DESTINO=TRANS(0,400, 400, 0, 180, 90);
LEFTY;
ABOVE;
FLIP;

SIGNAL(2);

for i=1:3,

        SIGNAL(1);
        WAIT (2);
        SIGNAL (-1);
        APPRO(BARRIDO, 120);
        MOVES(BARRIDO);

        REACTI(1, ´AGARRE´);
        MOVES (BARRIDO*TRANS(-200, 0, 0, 0, 0, 0));
        APPRO (DESTINO, 100);
        MOVES (DESTINO);
        OPENI;
        DEPARTS(100);
        DESTINO=SHIFT (DESTINO, 200,0,0);
end
READY;
```

En lo que respecta a la función **AGARRE,** la cual se ejecuta cuando se detecte un flanco de subida del sensor que porta el robot en la pinza, el código sería el siguiente

```
function AGARRE()
MOVES (SHIFT(HERE, 40,0,0);
APROS (HERE, -80);
CLOSEI;
DEPARTS (150);
```

Fichero de configuración ´entorno3.dat´

```
Observador (giro, elevacion, apertura):
60 20 6
Sensor del robot (rango en mm.):
150
Velocidad del robot (de 0 a 1):
0.05
Sensibilidad de la pinza (distancia en mm.):
100
Tipos de cajas (lx, ly, lz, color): 3
500 200 300  9
600 200 100 11
 40  80  80 44
Cajas (posx, posy, posz, giro, tipo): 4
200  400   1   0 1
350 -100 101  90 3
400 -220 101  90 3
450 -350 101  90 3
Sensores (posx, posy, posz, giro, rango): 1
510 190 120 -90 200
Cintas transportadoras (posx, posy, posz, giro, tipo, velocidad): 1
500 -400 100 90 2 5
```

CONCLUSIONES

La elaboración de este trabajo fin de grado ha resultado muy interesante, he tenido la oportunidad de aprender conceptos importantes en el ámbito de la robótica y por otro lado, me ha permitido aplicar contenidos vistos durante el curso pasado en la asignatura de Sistemas Inteligentes.

Analizados los distintos tipos de programación concluimos que en la programación por guiado, las acciones que se pueden programar son limitadas, el robot sufre un desgaste innecesario y no realiza ningún trabajo útil mientras dura el proceso de programación. Por el contrario, en la programación textual, se basa en especificar la tarea del robot mediante sentencias, con lo cual, la flexibilidad de codificación es mayor, se pueden utilizar todas las prestaciones de los lenguajes de programación convencionales; sin embargo, se exige un operador especializado. Dado esto, como solución se opta por la programación a nivel robot, donde el lenguaje es imperativo con instrucciones especializadas para el movimiento del brazo, la gestión de la pinza o para el tratamiento de sensores.

Nótese, que los lenguajes de nivel robot y el lenguaje V+ en concreto, poseen muchas más instrucciones y prestaciones que las descritas en esta memoria. En este trabajo se han descrito e ilustrado las características que distinguen a este tipo de programación mediante ejemplos sencillos.

La simulación gráfica en tres dimensiones de la cinemática del movimiento de un brazo manipulador permite el análisis previo de la ejecución de una tarea, lo cual ayuda a detectar errores de programación (como por ejemplo colisiones del robot manipulador con otro robot u objeto del entorno), dando la oportunidad de corregirlos antes ser programado en el robot real, con el consiguiente ahorro de tiempo y esfuerzo.

De cara a la docencia, la capacidad del simulador para trabajar sin estar conectado al brazo robot real favorece el aprendizaje de los alumnos, ayudándoles a entender con mayor facilidad los principios teóricos básicos para el control y posicionamiento del brazo manipulador. Al mismo tiempo que les permitirá interactuar y practicar con un software cuyas funciones son similares a las que encontrarán en un software profesional para el control de robots industriales.

Este trabajo se ha centrado en las características más relevantes de los lenguajes a nivel robot, en un futuro se podrían desarrollar más funciones con el objeto de que el brazo manipulador sea capaz de simular tareas más específicas.

Bibliografía

Robótica: Control de robots manipuladores (2011)
Fernando Reyes Cortés

Fundamentos de la Robótica. McGraw Hill 2º Edición (2007)
Barrientos A., Peñín L.F:, Balaguer C., Aracil Santoja R.

Robótica. Prentice Hall. 3º Edición. (2006)
Craig John J.

Robótica: Manipuladores y Robot Móviles. Marcombo Boixareu Editores (2001)
Ollero A.

Introduction to Robotics. Addison-Wesley (1991)
McKerrow P. J.

Robótica Industrial. Marcombo (1.988)
Ferraté G., Amat J., Ayza J., Basañez L., Ferrer L., Huber R., Torres C.

Robótica: Control, Detección, Visión e Inteligencia. McGraw Hill (1988)
Fu K.S., González R. C., Lee C.S.G.

Industrial Robotics: Technology, Programming and Applications. McGraw Hill (1986)
Groover M. P., Weiss M., Nagel R.N, Odrey M. G.

The MathWorks Inc. (1994-2012) Using Matlab Version R2012b

ANEXOS

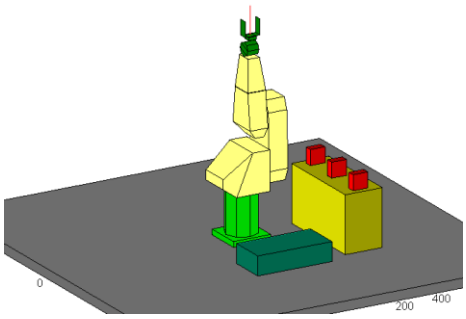
PRÁCTICA 0

En esta práctica disponemos de un brazo manipulador situado junto a un conjunto de cinco cajas numeradas como 1,2,3,4 y 5 las cajas 4 y 5 actúan como plataformas.

OBJETIVO: Trasladar a la caja 5, las cajas 1, 2 y 3 posadas en la caja 4.

Se invoca al fichero de configuración donde se establecen los valores iniciales de las variables internas del simulador, entorno0.txt.

```
PROGRAM('entorno0.txt')
```



La variable PIEZA construye una transformada homogénea a partir de la posición cartesiana (0,400,500) expresada en milímetros, y la representación de la orientación (0,180,90) expresada en grados, igualmente para la variable DESTINO, realiza una transformada homogénea a partir de la posición cartesiana (400,150,400) y la representación de la orientación en grados (0,180,0)

```
PIEZA=TRANS(0,400,500,0,180,90);
DESTINO=TRANS(400,150,400,0,180,0);
```

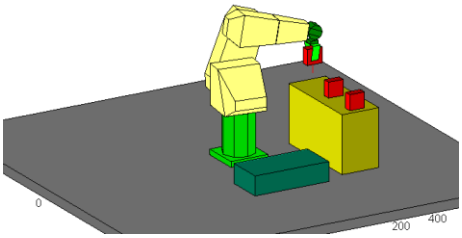
Establece el manipulador como "brazo izquierdo" y "codo arriba”

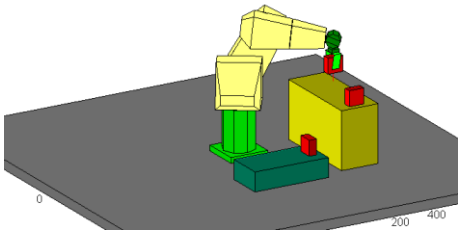
```
LEFTY;
ABOVE;
```

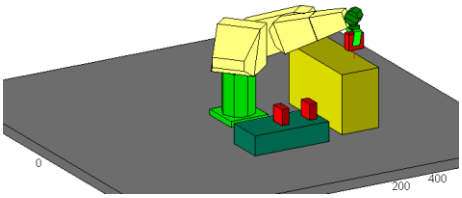
Implementamos un bucle para trasladar las cajas consecutivamente Coge una pieza Posiciona la pinza con un movimiento articular a una localización calculada tomando como referencia la caja en el espacio cartesiano y cierra la pinza para coger la caja y mueve la pinza.

```
for i=1:3,

   MOVE(PIEZA);
   APPROS(PIEZA,-100);
   CLOSEI;
   DEPARTS(100);
```

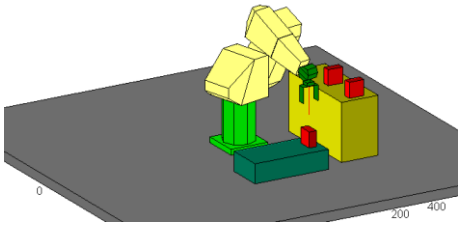


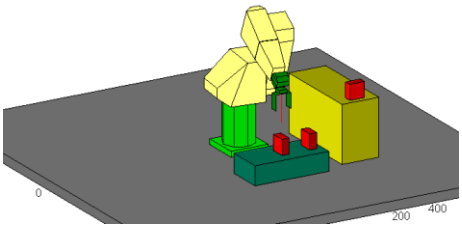


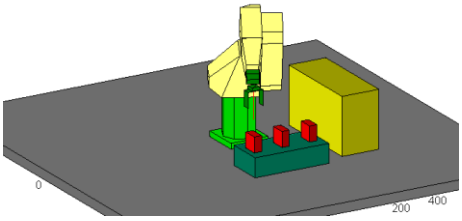


Traslada el objeto que porta la pinza, desde la postura actual hasta el destino especificado en el argumento de entrada y comprueba el estado de las señales de entrada para lanzar la reacción correspondiente. Posiciona la pinza en una localización calculada tomando como referencia de destino y abre la pinza para soltar la caja. Por último aleja la pinza.

```
    MOVES(DESTINO);
        APPROS(DESTINO,-200);
        OPENI;
        DEPARTS(200);
        PIEZA=SHIFT(PIEZA,200,0,0);
        DESTINO=SHIFT(DESTINO,0,-150,0);
```








```
end
READY;
```

FICHERO DE CONFIGURACIÓN PRÁCTICA 0

Observador (giro, elevacion, apertura):
60 20 5

Sensor del robot (rango en mm.):
150

Velocidad del robot (de 0 a 1):
0.05

Sensibilidad de la pinza (distancia en mm.):
100

Tipos de cajas (lx, ly, lz, color): 3
500 200 300 9
200 400 100 11
 40 80 80 44

Cajas (posx, posy, posz, giro, tipo): 5
200 400   1 0 1
400   0   1 0 2
  0 400 301 0 3
200 400 301 0 3
400 400 301 0 3

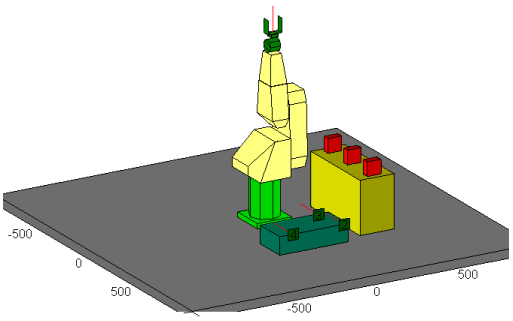
PRÁCTICA 1

En esta práctica disponemos de un brazo manipulador situado junto a un conjunto de cinco cajas numeradas como 1,2,3,4 y 5 las cajas 4 y 5 actúan como plataformas.

OBJETIVO: Trasladar a la caja 5, las cajas 1, 2 y 3 posadas en la caja 4,es el mismo ejemplo que la práctica 0 pero con la salvedad de que aquí entran en juego 3 sensores, el brazo manipulador suelta las cajas donde detecta los sensores. Si observamos las sentencias son las mismas en la práctica 0 y en la práctica 1, la diferencia entre los dos estriba en el fichero de configuración, donde se detallan los sensores.

La función PROGRAM invoca al fichero de configuración donde se establecen los valores iniciales de las variables internas del simulador, entorno1.txt

```
PROGRAM('entorno1.txt')
```



La variable pieza construye una transformada homogénea a partir de la posición cartesiana (0,400,500) expresada en milímetros, y la representación de la orientación (0,180,90) expresada en grados, igualmente para la variable destino realiza una transformada homogénea a partir de la posición cartesiana (400,150,400) y la representación de la orientación en grados (0,180,0)

```
PIEZA=TRANS(0,400,500,0,180,90);
DESTINO=TRANS(400,150,400,0,180,0);
```

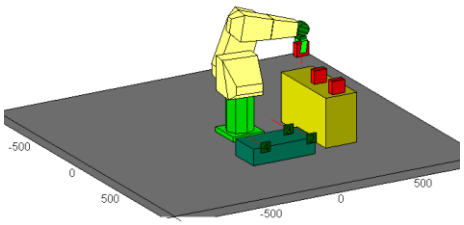
Establece el manipulador como "brazo izquierdo" y "codo arriba”

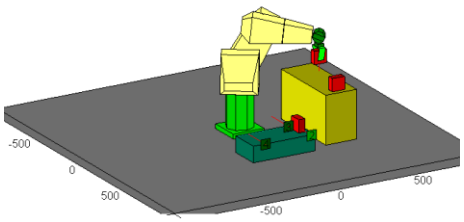
```
LEFTY;
ABOVE;
```

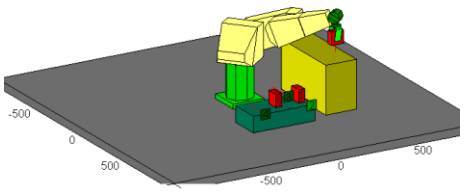
Implementamos un bucle para trasladar las cajas consecutivamente Coge una pieza Posiciona la pinza con un movimiento articular a una localización calculada tomando como referencia la caja en el espacio cartesiano y cierra la pinza para coger la caja y mueve la pinza.

```
for i=1:3,

        MOVE(PIEZA);
        APPROS(PIEZA,-100);
        CLOSEI;
        DEPARTS(100);
```





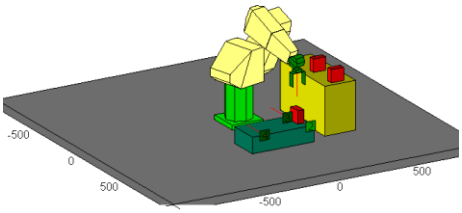


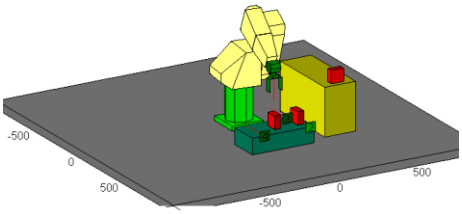
Traslada el objeto que porta la pinza, desde la postura actual hasta el destino especificado, en este caso donde se localiza el sensor, en el argumento de entrada y

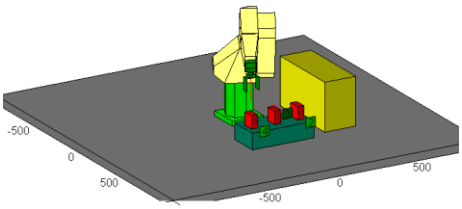
comprueba el estado de las señales de entrada para lanzar la reacción correspondiente. Posiciona la pinza en una localización calculada tomando como referencia de destino y abre la pinza para soltar la caja. Por último aleja la pinza.

```
        MOVES(DESTINO);
        APPROS(DESTINO,-200);
        OPENI;
        DEPARTS(200);


        PIEZA=SHIFT(PIEZA,200,0,0);
        DESTINO=SHIFT(DESTINO,0,-150,0);
```








```
end
READY;
```

FICHERO DE CONFIGURACIÓN PRÁCTICA 1

```
Observador (giro, elevacion, apertura):
60 20 6
Sensor del robot (rango en mm.):
150
Velocidad del robot (de 0 a 1):
0.05
Sensibilidad de la pinza (distancia en mm.):
100
Tipos de cajas (lx, ly, lz, color): 3
500 200 300  9
200 400 100 11
 40  80  80 44
Cajas (posx, posy, posz, giro, tipo): 5
200 400   1 0 1
400   0   1 0 2
  0 400 301 0 3
200 400 301 0 3
400 400 301 0 3
Sensores (posx, posy, posz, giro, rango): 3
550  150 120 -90 200
550    0 120 -90 200
550 -150 120 -90 200
```

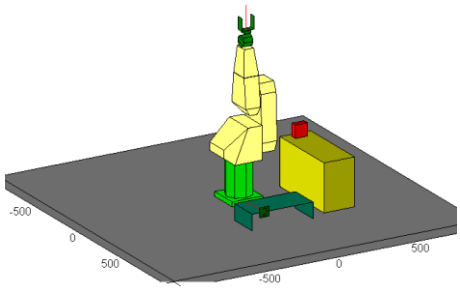
PRÁCTICA 2

En esta práctica disponemos de un brazo manipulador situado junto a un conjunto de cinco cajas numeradas como 1,2,3,4 y 5 las cajas 4 y 5 actúan como plataformas.

OBJETIVO: Trasladar una pieza desde la caja 4 a la caja 5, esta última consta de una cinta transportadora con un sensor de barrera que detiene la cinta cuando detecta la pieza, está orientada para el paletizado de piezas.

La función PROGRAM invoca al fichero de configuración donde se establecen los valores iniciales de las variables internas del simulador, entorno2.txt

```
PROGRAM('entorno2.txt')
```



La variable pieza construye una transformada homogénea a partir de la posición cartesiana (0,400,500) expresada en milímetros, y la representación de la orientación (0,180,90) expresada en grados, igualmente para la variable destino realiza una transformada homogénea a partir de la posición cartesiana (400,150,400) y la representación de la orientación en grados (0,180,0)

```
PIEZA=TRANS(0,400,500,0,180,90);
DESTINO=TRANS(400,150,400,0,180,0);
```

Establece el manipulador como "brazo izquierdo" y "codo arriba”

```
LEFTY;
ABOVE;
```

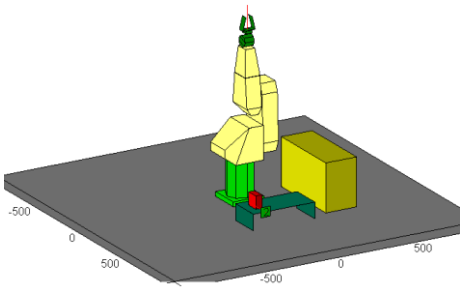
La pinza se mueve hasta la posición donde se encuentra la caja, la agarra y se aleja de la posición inicial.

```
MOVE(PIEZA);
APPROS(PIEZA,-100);
CLOSEI;
DEPARTS(100);
```

La pinza traslada la caja, la lleva hasta la posición de destino y la suelta en la cinta transportadora.

```
MOVES(DESTINO);
APPROS(DESTINO,-200);
OPENI;

SIGNAL(1);
SIGNAL(-2);
DEPARTS(200);
WAIT(2);
SIGNAL(-1)
CLOSEI;
READY;
```



FICHERO DE CONFIGURACIÓN PRÁCTICA 2

```
Observador (giro, elevacion, apertura):
60 20 6
Sensor del robot (rango en mm.):
150
Velocidad del robot (de 0 a 1):
0.05
Sensibilidad de la pinza (distancia en mm.):
100
Tipos de cajas (lx, ly, lz, color): 3
500 200 300  9
400 200 100 11
 40  80  80 44
Cajas (posx, posy, posz, giro, tipo): 2
200 400   1  0 1
  0 400 301  0 3
Sensores (posx, posy, posz, giro, rango): 1
550 -150 120 -90 200
Cintas transportadoras (posx, posy, posz, giro, tipo, velocidad): 1
500 -200 100 90 2 5
```

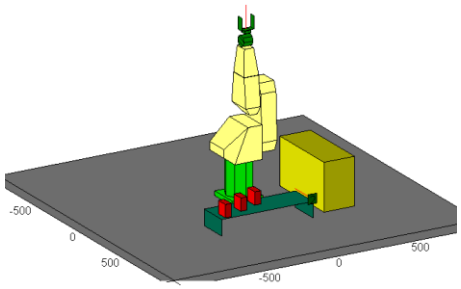
PRÁCTICA 3

Esta práctica presenta un sistema robótico empleado para el paletizado de piezas. Tenemos un brazo robot, una cinta transportadora y una plataforma de paletizado. Asimismo el sistema dispone de dos sensores externos y digitales de presencia, uno de barrera, montado en un extremo de la cinta transportadora y otro de proximetría situado en la base de la pinza del manipulador.

OBJETIVO: Trasladar tres cajas, que avanzan en la cinta transportadora, caja 5, en dirección al sensor de barrera a la plataforma de paletizado, caja 4.

Se carga el fichero de configuración donde se establecen los valores iniciales de las variables internas del simulador, en este caso entorno3.txt

```
PROGRAM('entorno3.txt')
```



La variable BARRIDO construye una transformada homogénea a partir de la posición cartesiana (300,170,280) expresada en milímetros, y la representación de la orientación (0,180,0) expresada en grados, definiendo la posición de la pinza en el momento de agarrar la caja igualmente para la variable DESTINO, realiza una transformada homogénea a partir de la posición cartesiana (0,400,400) y la representación de la orientación en grados (0,180,90), definiendo esta donde la pinza debe posar la caja y soltarla.

```
BARRIDO=TRANS(300,170,280,0,180,0);
DESTINO=TRANS(0,400,400,0,180,90);
```

Establece el manipulador como "brazo izquierdo" y "codo arriba" y "muñeca girada"

```
LEFTY;
ABOVE;
FLIP;
```

Se activa la cinta transportadora y el sentido de la marcha, la caja se traslada sobre la cinta transportadora hasta que el sensor de barrera la detecta y para esta, la pinza se traslada hasta la posición declarada y agarra la caja, que traslada hasta la localización definida en destino, el bucle toma el siguiente valor hasta que trasladar todas las cajas.

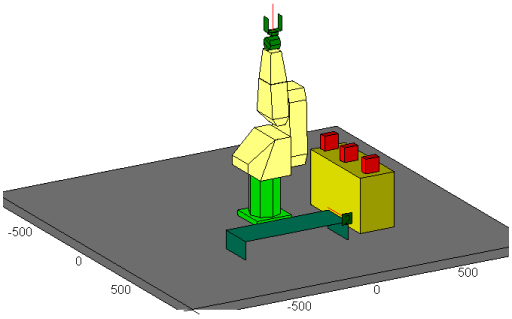
```
SIGNAL(2);

for i=1:3,

        SIGNAL(1);
        WAIT(2);
        SIGNAL(-1);
        APPRO(BARRIDO,120);
        MOVES(BARRIDO);

        REACTI(1,'AGARRE');
        MOVES(BARRIDO*TRANS(-200,0,0,0,0,0));

        APPRO(DESTINO,100);
        MOVES(DESTINO);
        OPENI;
        DEPARTS(100);
        DESTINO=SHIFT(DESTINO,200,0,0);
end
READY;
```



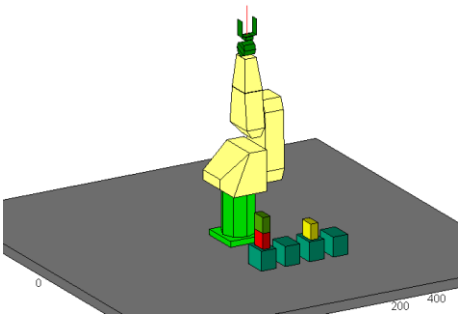
FICHERO DE CONFIGURACIÓN PRÁCTICA 3

Observador (giro, elevacion, apertura):
60 20 6
Sensor del robot (rango en mm.):
150
Velocidad del robot (de 0 a 1):
0.05
Sensibilidad de la pinza (distancia en mm.):
100
Tipos de cajas (lx, ly, lz, color): 3
500 200 300 9
600 200 100 11
 40 80 80 44
Cajas (posx, posy, posz, giro, tipo): 4
200 400   1   0 1
350 -100 101 90 3
400 -220 101 90 3
450 -350 101 90 3
Sensores (posx, posy, posz, giro, rango): 1
510 190 120 -90 200
Cintas transportadoras (posx, posy, posz, giro, tipo, velocidad): 1
500 -400 100 90 2 5

PRUEBA DE PLANIFICACIÓN

En esta prueba se ha querido representar mediante el simulador el mundo de bloques, ejemplo clásico de planificación en Inteligencia Artificial, donde disponemos de un brazo manipulador, cuatro cajas que determinan las posiciones y 3 cajas de colores que tenemos distribuidas en una posición original sobre las anteriores y queremos que lleguen a una distribución objetivo. Los movimientos permitidos son mover una caja de una posición a otra o apilarla sobre otra de las cajas de colores. Para conseguir llegar a la distribución objetivo se ha planteado el problema de planificación como un problema de búsqueda y se ha utilizado el algoritmo A*.

```
PROGRAM('entornobasico.txt');
```



Posición de referencia del primer lugar

```
POS0=[400,-100,100];
```

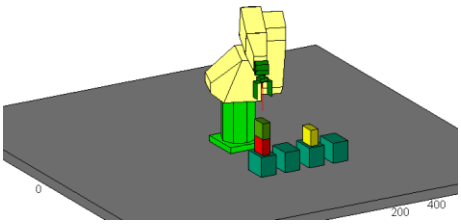
Incremento en coordenadas Y y Z y establecimiento del manipulador como "brazo izquierdo" y "codo arriba”

```
INCYZ=[130,80];
LEFTY;
ABOVE;
```

S: Registro que contiene el nodo de comienzo

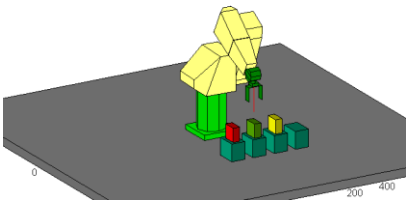
H: Registro que contiene el nodo objetivo

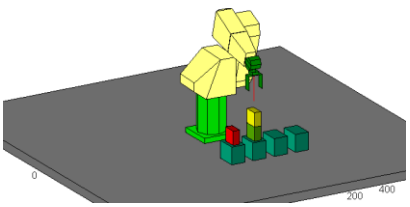
```
S=INICIOENTORNO([3,0,0],[0,2,0,1],POS0,INCYZ);
H=FINAL([0,1,2],[0,0,3,0]);
```

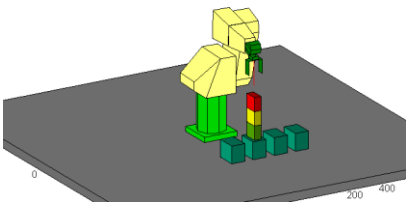


busqueda(): Función que realiza la búsqueda en grafo, en este caso se ha implementado el algoritmo A* que es un algoritmo de búsqueda en grafos que encuentra el camino óptimo, siempre y cuando la heurística proporcionada sea admisible, desde un nodo origen a un nodo destino. Para nuestro caso concreto el camino de menor coste para llegar desde el nodo S al nodo H

```
V=busqueda(S,H);
NACCIONES=length(V);
```







Comprobamos si ha encontrado el camino (planificación)

```
if ~isempty(V),
```

Se inicializa un bucle para ir recuperando los movimientos de la planificación, es decir, las acciones que tiene que ejecutar el robot.

```
        for i=NACCIONES-1:-1:1,

               fprintf('%s \n',V(i).a);
         [ORIGEN,DESTINO]=ORIGENYDESTINO(V(i+1),V(i),POS0,INCYZ);
```

Conjunto de maniobras del robot para coger la caja

```
         APPRO(ORIGEN,200);
         MOVES(ORIGEN);
         CLOSEI
         DEPARTS(200);
```

Conjunto de maniobras del robot para soltar la caja

```
         APPRO(DESTINO,200);
         MOVES(DESTINO);
         OPENI
         DEPARTS(200);

    end
```

Vuelve la pinza a su posición de descanso

```
    READY;

else
```

Si no encuentra ningún camino (planificación) devuelve por pantalla "No existe solución"

```
        fprintf('No existe solucion \n');

end
```