Escola Universitària Politècnica
de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MINIPROYECTO ROBÓTICA

TÍTULO:	**EL LENGUAJE V+**
AUTORES:	**JAUME YEBRA PÉREZ** **NÚRIA LAGOS FERNÁNDEZ**
TITULACIÓN:	**INGENIERÍA TÉCNICA INDUSTRIAL ELECTRÓNICA**
DIRECTOR:	**PERE PONSA ASENSIO**
DEPARTAMENTO:	**INGENIERÍA DE SISTEMAS, AUTOMÁTICA E INFORMÁTICA INDUSTRIAL**
FECHA:	**DICIEMBRE - 2002**

MINIPROYECTO DE ROBÓTICA

RESUMEN (máximo 50 líneas)

Mediante este miniproyecto, se pretende, en primer lugar, ampliar los conocimientos adquiridos en la asignatura de ROBÓTICA de la titulación de Ingeniería Técnica Industrial especialidad Electrónica Industrial. Así mismo, el tema escogido trata sobre los lenguajes de programación de robots industriales, mercado actual, posibilidades, historia, comparativa, etc. Concretamente se ha tratado de ampliar el lenguaje V+.

Por otro lado, se han explicado las características principales del V+, código fuente, programas ejemplo, aplicaciones, etc.

Otra parte del miniproyecto, ha sido la difícil tarea de reunir información técnica suficiente para que tenga lugar este trabajo. Así mismo, se han adjuntado extensos manuales de VAL II y V+ (VAL3), en modo PDF, estos últimos (manuales de V+ de Adept) han sido realmente difíciles de conseguir, y la información que ofrecen es muy completa.

Por último, se ha realizado un entorno multimedia, en formato WEB, para ordenar toda la información procesada.

Palabras clave (máximo 10):

ROBÓTICA	LENGUAJES PROGRAMACIÓN	VAL	VAL II
V+	ADEPT	STÄUBLI	

ÍNDICE

1. INTRODUCCIÓN A LA ROBÓTICA

1.1. Los orígenes de la palabra robot y su supuesto significado.

El término robot es reciente, pero el concepto es muy antiguo. Antiguamente aparecía expresado en sinónimos como autómata, que significaba que pensaba por sí mismo.

La palabra robot aparece por el sueño de los seres humanos en poder librarse de tareas indeseables, peligrosas o demasiado pesadas.

La primera vez que se hablo de estos seres, utilizando el término por el que es hoy mundialmente conocido, robot, fue en 1923 por el escritor Karel Capek en su comedia R.U.R..("Rossum`s Universal Robots"), palabra que proviene del término checo robotnik que significa “siervo”.

Hoy la palabra robot tiene diferentes significados:

- El significado de la palabra **Robot** por el Institute of América: "un manipulador multifuncional y reprogramable, diseñado para mover materiales, piezas, herramientas o dispositivos especiales, mediante movimientos programables y variables que permitan llevar a cabo diversas tareas".

- El significado de la palabra **Robot** por Oxford English dictionary: "un aparato mecánico que se parece y hace el trabajo de un ser humano".

- El concepto de **Robot inteligente** incluye la capacidad de recibir instrucciones de alto nivel expresadas como "comandos" para realizar una tarea general, trasladando dichas instrucciones a un conjunto de acciones que deben ejecutarse para llevar a cabo dicha tarea. Será consciente de su entorno y capaz de tomar decisiones acerca de sus acciones basadas, en parte, en la interpretación de dicho entorno.

1.2. Etapas históricas sobre los robots:

En este cuadro cronológico mostramos las diversas etapas históricas sobre los robots:

FECHA	DESARROLLO
SigloXVIII	Vaucanson construyó varias muñecas mecánicas de tamaño humano que ejecutaban piezas de música.
1801	J. Jaquard invento su telar, que era una máquina programable para la urdimbre.
1805	H. Maillardet construyó una muñeca mecánica capaz de hacer dibujos.
1946	El inventor americano G.C Devol desarrolló un dispositivo controlador que podía registrar señales eléctricas por medios magnéticos y reproducirlas para accionar un máquina mecánica. La patente estadounidense se emitió en 1952.
1951	Trabajo de desarrollo con teleoperadores (manipuladores de control remoto) para manejar materiales radiactivos. Patente de Estados Unidos emitidas para Goertz (1954) y Bergsland (1958).
1952	Una máquina prototipo de control numérico fue objetivo de demostración en el Instituto Tecnológico de Massachussets, después de varios años de desarrollo. Un lenguaje de programación de piezas denominado APT (Automatically Programmed Tooling) se desarrolló posteriormente y se publicó en 1961.
1954	C. W. Kenward solicitó su patente para diseño de robots. Patente británica emitida en 1957.
1954	G.C. Devol desarrolla diseños para Transferencia de artículos programados. Patente emitida en Estados Unidos para el diseño en 1961.
1959	Se introdujo el primer robot comercial por Planet Corporation. estaba controlado por interruptores de fin de carrera.
1960	Se introdujo el primer robot 'Unimate'', basado en la transferencia de articulaciones programada de Devol. Utilizan los principios de control numérico para el control del manipulador (robot de transmisión hidraulica).
1961	Un robot Unimate se instaló en la Ford Motors Company para atender una máquina de fundición de troquel.
1966	Trallfa, (firma noruega) construyó e instaló un robot de pintura por pulverización.
1968	Un robot móvil llamado 'Shakey'' se desarrollo en SRI (standford Research Institute). Estaba provisto de una diversidad de sensores así como una cámara de visión y sensores táctiles y podía desplazarse por el suelo.
1971	El 'Standford Arm'', un pequeño brazo de robot de accionamiento eléctrico (Standford University.)
1973	Se desarrolló en SRI, el primer lenguaje de programación de robots del tipo de computadora para la investigación con la denominación WAVE. Fue seguido por el lenguaje AL en 1974. Los dos lenguajes se desarrollaron posteriormente en el lenguaje VAL comercial, para Unimation, por Víctor Scheinman y Bruce Simano.
1974	ASEA introdujo el robot Irb6 de accionamiento completamente eléctrico.
1974	Kawasaki, bajo licencia de Unimation, instaló un robot para soldadura por arco para estructuras de motocicletas.
1974	Cincinnati Milacron introdujo el robot T3 con control por computadora.
1975	El robot 'Sigma'' de Olivetti se utilizó en operaciones de montaje, una de las primitivas aplicaciones de la robótica al montaje.
1976	Un dispositivo de Remopte Center Compliance (RCC) para la inserción de piezas en la línea de montaje se desarrolló en los laboratorios Charles Stark Draper Labs en estados Unidos.
1978	El robot T3 (con lenguaje propio T3 gestual punto a punto) de Cincinnati Milacron se adaptó y programó para realizar operaciones de taladro y circulación de materiales en componentes de aviones, bajo el patrocinio de Air Force ICAM (Integrated Computer- Aided Manufacturing).
1978	Se introdujo el robot PUMA (Programmable Universal Machine for Assambly) para tareas de montaje por Unimation, basándose en diseños obtenidos en un estudio de la General Motors.
1979	Desarrollo del robot tipo SCARA (Selective Compliance Arm for Robotic Assambly) en la Universidad de Yamanashi en Japón para montaje. Varios robots SCARA comerciales se introdujeron hacia 1981.
1980	Un sistema robótico de captación de recipientes fue objeto de demostración en la Universidad de Rhode Island. Con el empleo de visión de máquina el sistema era capaz de captar piezas en orientaciones aleatorias y posiciones fuera de un recipiente.
1981	Se desarrolló en la Universidad de Carnegie- Mellon un robot de impulsión directa, el cual utilizaba motores eléctricos situados en las articulaciones del manipulador, sin las transmisiones mecánicas habituales
1982	IBM introdujo el robot RS-1 para montaje, basado en varios años de desarrollo interno. Se trata de un robot de estructura de caja que utiliza un brazo constituido por tres dispositivos de deslizamiento ortogonales. El lenguaje del robot AML, desarrollado por IBM, se introdujo también para programar el robot SR-1.
1983	Informe emitido por la investigación en Westinghouse Corp. bajo el patrocinio de National Science Foundation sobre un sistema de montaje programable adaptable (APAS), un proyecto piloto para una línea de montaje automatizada flexible con el empleo de robots.
1984	Robots 8. La operación típica de estos sistemas permitía que se desarrollaran programas de robots utilizando gráficos interactivos en una computadora personal y luego se cargaban en el robot.

1.3. Clasificación de los robots:

En este cuado mostramos los criterios de clasificación de los robots:

CRITERIO	CLASIFICACIÓN
Geometría	Se basa en la forma del área de trabajo producida por el brazo del robot: rectangular, cilíndrica o esférica.
Configuraciones	La configuración polar utiliza coordenadas polares para especificar cualquier posición en términos de una rotación sobre su base, un ángulo de elevación y una extensión lineal del brazo.
	La configuración cilíndrica sustituye un movimiento lineal por uno rotacional sobre su base, con los que se obtiene un medio de trabajo en forma de cilindro.
	La configuración de coordenadas cartesianas posee tres movimientos lineales, y su nombre proviene de las coordenadas cartesianas, las cuales son más adecuadas para describir la posición y movimiento del brazo. Los robots cartesianos a veces reciben el nombre de XYZ, donde las letras representan a los tres ejes del movimiento.
Grados de libertad	Consiste en contar el número de grados de libertad que tengan. Se considera un grado de libertad cada eje a lo largo del cual se puede mover el brazo de un robot.
Área de aplicación	Ensamblaje
	No ensamblaje: soldar, pintar, revestir, manejo de materiales y carga y descarga de maquinaria.
Técnica de control	Lazo cerrado: se monitorea continuamente la posición del brazo del robot mediante un sensor de posición, y se modifica la energía que se manda al actuador de tal forma que el movimiento del brazo se obedece al camino deseado, tanto en dirección como en velocidad. Éste control se puede usar cuando la tarea que se ha de llevar a cabo está dirigida mediante un camino definido por la misma pieza, tal como sería soldar, revestir y ensamblar.
	En un sistema de lazo abierto, el controlador no conoce la posición de la herramienta mientras el brazo se mueve de un punto a otro. Éste tipo de control es muy usado cuando el movimiento que debe seguir el brazo se encuentra determinado previamente, al ser grabado con anterioridad y reproducido sin cambio alguno, lo cual es útil cuando todas las piezas a ser tratadas son exactamente iguales.
Fuente de energía	De energía hidráulica: En los actuadores hidráulicos fluye un líquido, comúnmente aceite. Tienen como ventaja que son pequeños comparados con la energía que proporcionan, y como desventajas que son propensos a fugas, el líquido puede incendiarse y que se requiere numeroso equipo adicional, lo cual incrementa los costos de mantenimiento del robot. Los sistemas hidráulicos están asociados a un mayor nivel de ruido.
	De energía neumática: En los actuadores neumáticos se transfiere gas bajo presión. Generalmente sólo tienen dos posiciones: retraídos y extendidos, si posibilidad de utilizar retroalimentación para usar un control proporcional. La energía neumática tiene las siguientes ventajas: está disponible en la mayoría de las áreas de manufactura, no es cara y no contamina el área de trabajo. La desventaja es que no se puede utilizar retroalimentación ni múltiples pasos.
	De energía eléctrica: Los actuadores eléctricos incluyen una fuente de poder y un motor eléctricos. La mayoría de las aplicaciones utilizan servomotores, el cual generalmente utiliza corriente directa. Las ventajas de esta fuente de energía son que no se requiere transformar la energía eléctrica en otras formas de energía como la hidráulica o neumática, no se contamina el espacio de trabajo y el nivel de ruido se mantiene bajo. La desventaja es la baja potencia que se consigue en comparación con su contraparte hidráulica.

1.4. Propiedades características de los robots

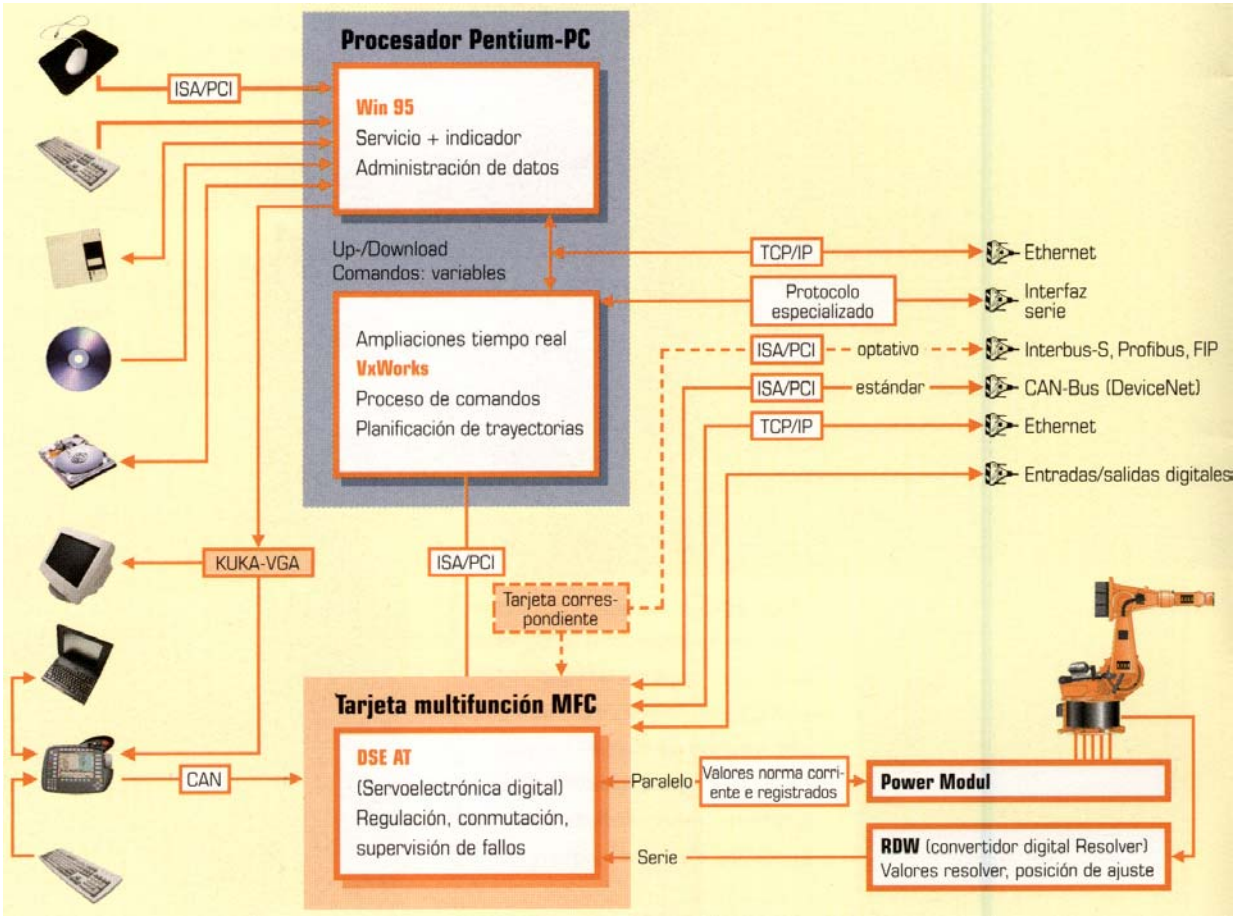
Las características más importantes de los robots son:

- **Versatilidad**: es la posibilidad de ejecutar diversas tareas, o la misma tarea, de formas diversas. Estos robots deben tener una estructura mecánica de geometría variable.

- **Autoadaptabilidad:** de los robots es la posibilidad, de estos, por alcanzar el objetivo que se le ha fijado (ejecutar su tarea), a pesar de las perturbaciones imprevistas del entorno a lo largo de la ejecución de su tarea. Esto supone que el robot sea consciente de su entorno y, por lo tanto, posea sentidos artificiales (sensores).

1.5. Comunicación con los robots.

La comunicación con los robots no sólo se basa en la programación de éste, también hay comunicación entre los distintos periféricos del robot industrial, y cada vez estos periféricos son mayores en cuanto a numero. Los protocolos de comunicación utilizados por los sistemas automatizados suelen ser numerosos para así ajustarse a las posibles necesidades del usuario. En la ilustración inferior, se puede observar la multitud de periféricos instalables en un sistema de KUKA, y las posibles comunicaciones contempladas.



El lenguaje es el medio que utiliza el hombre para gobernar las máquinas controladas por sistemas informáticos, de manera, que su correcta adaptación con la tarea a realizar y sencillez de manejo, son factores determinantes del rendimiento obtenido en los robots industriales.

Hay tres maneras generales de comunicarse con un robot que son: **reconocimiento de voz, enseñanza y repetición y lenguajes de programación de alto nivel.**

Los sistemas de reconocimiento de voz con la tecnología actual, son bastante primitivos. Estos sistemas pueden reconocer un conjunto de palabras concretas de un vocabulario limitado y, en general, exigen al usuario una pausa entre las palabras, aunque en la actualidad es posible reconocer las palabras separadas en tiempo real debido a que cada vez los PC's son más rápidos y eficientes. La utilidad del reconocimiento de palabras separadas para describir la tarea de un robot es bastante limitada. Así mismo, pueden confundirse los ruidos ambientales con la voz.

La enseñanza y repetición, también conocido como **guiado**, es la solución más utilizada en la actualidad en los robots industriales. Este método consiste en guiar al robot (enseñar), dirigiéndole los movimientos que el usuario desea que realice.

La enseñanza y repetición se lleva a cabo, normalmente, con los siguientes pasos:

1) Dirigiendo al robot con un movimiento lento, utilizando el control manual, para realizar la tarea completa y grabando los ángulos del movimiento del robot, en los lugares adecuados, para que vuelva a repetir el movimiento.
2) Reproduciendo y repitiendo el movimiento enseñado.
3) Si el movimiento enseñado es correcto, entonces se hace funcionar al robot a la velocidad correcta, en modo repetitivo.

Podemos guiar al robot en movimientos lentos de varias maneras: usando un joystick, un conjunto de botones (uno para cada movimiento) o un sistema de manipulación maestro-esclavo.

Los lenguajes de programación de alto nivel suministran una solución más general para resolver el problema de comunicación hombre-robot.

En la década anterior, los robots fueron utilizados con éxito en áreas tales como soldadura por arco voltaico o pintura con spray utilizando el guiado. Estas tareas no requieren interacción entre el robot y su entorno y pueden ser programadas fácilmente por guiado. Sin embargo, la utilización de robots, para llevar a cabo las tareas, requieren técnicas de programación en lenguajes de alto nivel, ya que el robot de la línea de producción suele confiar en la realimentación de los sensores. Este tipo de interacción sólo puede ser mantenida por métodos de programación que contengan condiciones.

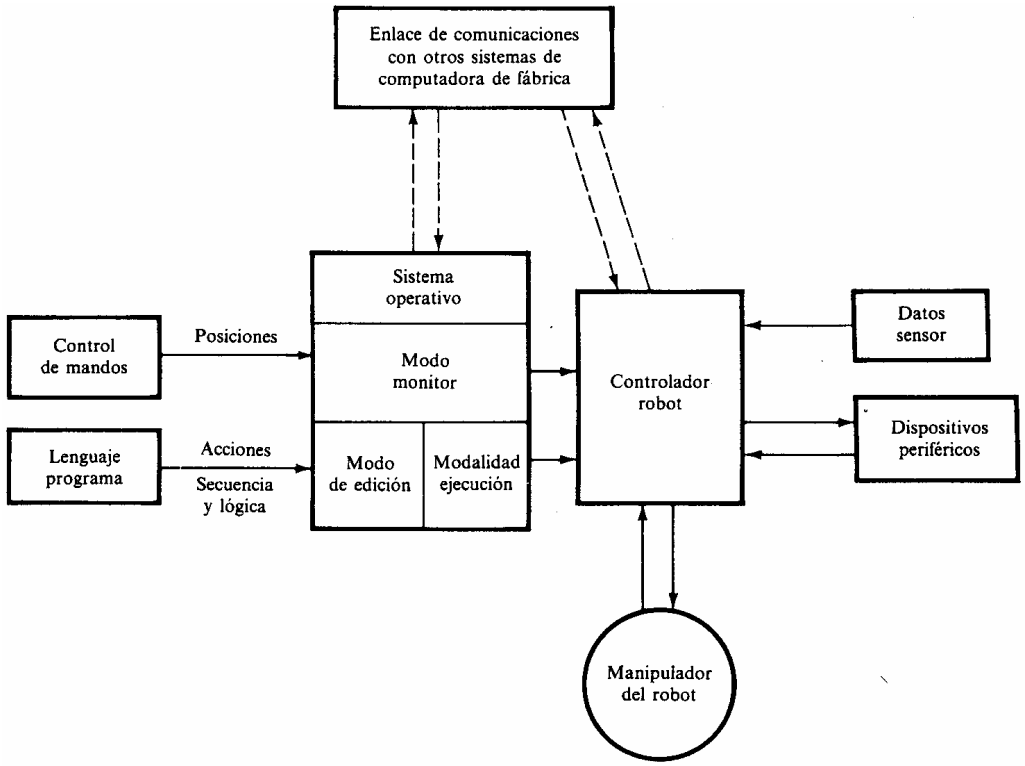
Los lenguajes clásicos empleados en informática, como el FORTRAN, BASIC, PASCAL, C, etc., no disponen de las instrucciones y comandos específicos que necesitan los robots, para aproximarse a su configuración y a los trabajos que han de realizar. Esta circunstancia, ha obligado a los constructores de robots e investigadores a diseñar lenguajes propios de robótica.

El problema, es que los lenguajes desarrollados hasta el momento, se han dirigido a un determinado modelo de controladora de robot, lo que ha impedido la aparición de lenguajes estándar y transportables entre máquinas, es decir, de carácter universal.

La estructura del sistema informático del robot varía notablemente, según el nivel y complejidad del lenguaje y de la base de datos requerida.

1.6. Lenguajes de programación

En el diagrama siguiente, se puede observar los diferentes componentes de un posible sistema automatizado, que deberán ser coordinados por medio del lenguaje de programación.



A continuación se realiza una descripción de los lenguajes de programación más usados en robótica.

1.6.1. Lenguaje de programación Gestual Punto A Punto

Estos lenguajes se utilizan con el robot "in situ", recordando las normas de funcionamiento de un video doméstico, ya que disponen de unas instrucciones similares: PLAY (reproducir), RECORD (grabar), FF (adelantar), FR (atrasar), PAUSE, STOP, etc. Además, puede disponer de instrucciones auxiliares, como INSERT (insertar un punto o una operación de trabajo) y DELETE (borrar). Este manipulador en línea funciona como un digitalizador de posiciones.

Los lenguajes más conocidos en programación gestual punto a punto son el FUNKY (creado por IBM), y el T3 (creado por Cincinnati Milacrom).

En el lenguaje FUNKY se utiliza un mando (joystick), que dispone de un comando especial para centrar la pinza sobre el objeto y controlar los movimientos, mientras que el T3 dispone de un dispositivo de enseñanza ("teach pendant").

El procesador usado en T3 es el AMD 2900 ("bit slice"), mientras que en el FUNKY está constituido por el IBM SYSTEM-7.

1.6.2. Lenguaje de programación A nivel de movimientos elementales.

Algunos de los lenguajes, mas importantes, que tratan los movimientos punto a punto son: **ANORAD, EMILY, RCL, RPL, SIGLA, VAL, MAL**.

Estos lenguajes mantienen las características de los movimientos primitivos, ya sea en coordenadas articulares o cartesianas. Tienen como ventajas destacables, los saltos condicionales y subrutinas, además de un aumento de las operaciones con sensores.

Estos lenguajes son de tipo intérprete, con excepción del RPL, que tiene un compilador. La mayoría dispone de comandos de tratamiento a sensores básicos: tacto, fuerza, movimiento, proximidad y presencia. El RPL dispone de un sistema complejo de visión, capaz de seleccionar una pintura y reconocer objetos presentes en su base de datos. Los lenguajes EMILY y SIGLA son transportables y admiten procesos simples, en paralelo.

Otros datos interesantes de estos lenguajes son:

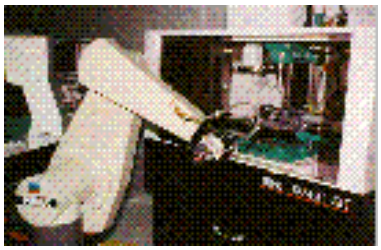
ANORAD

Este lenguajes es una transformación de un lenguaje de control numérico de la casa ANORAD CORPORATION, utilizado para el robot ANOMATIC. Utiliza como procesador el microprocesador 68000 de Motorola de 16/32 bits.

VAL

Fue diseñado por UNIMATION INC para sus robots UNIMATE y PUMA. Utiliza como CPU un LSI-II, que se comunica con procesadores individuales que regulan el servocontrol de cada articulación. Las instrucciones (en inglés) son sencillas e intuitivas como podemos apreciar en el siguiente ejemplo de programa:

```
.PROGRAM PICKUP
1. APRO PART, 25.0
2. MOVES PART
3. CLOSE, 0.0.0
4. APRO PART, -50.0
5. APRO DROP, 100.0
6. MOVES DROP
7. OPEN, 0.0.0
8. APRO DROP, -100.0
.END
```



RPL

Esta dotado con un LSI-II como procesador central que se aplica a los robots PUMA. Fue diseñado por SRI INTERNATIONAL.

EMILY

Es un lenguaje creado por IBM para el control de uno de sus robots. Usa el procesador IBM 370/145 SYSTEM 7 y está escrito en Ensamblador.

SIGLA

Fue desarrollado por OLIVETTI para su robot SUPER SIGMA. Emplea un mini-ordenador con 8K de memoria. Escrito en Ensamblador, es de tipo intérprete.

MAL

Fue creado en el Politécnico de Milán para el robot SIGMA, con un Mini-multiprocesador. Es un lenguaje del tipo intérprete, escrito en FORTRAN.

RCL

Aplicado al robot PACS y desarrollado por RPI, emplea como CPU, un PDP 11/03. Es del tipo intérprete y está escrito en Ensamblador.

1.6.3. Lenguajes Estructurados De Programación Explícita

Algunos de los lenguajes más importantes de programación explícita son: **AL, HELP, MAPLE, PAL, MCL, MAL EXTENDIDO.**

Todos los lenguajes que tratan de la programación explícita, (con excepción de HELP), tienen estructuras de datos de tipo complejo.

Así, el AL utiliza vectores, posiciones y transformaciones; el PAL usa transformaciones y el MAPLE permite la definición de puntos, líneas, planos y posiciones. Sólo el PAL, y el HELP carecen de capacidad de adaptación sensorial.

Los lenguajes AL, MAPLE y MCL tienen comandos para el control de la sensibilidad del tacto de los dedos (fuerza, movimiento, proximidad, etc.). Además, el MCL tiene comandos de visión para poder identificar e inspeccionar objetos.

Exponemos las características más representativas de los lenguajes de programación explícita:

AL

Proporciona las definiciones necesarias acerca de los movimientos relacionados con los elementos sobre los que trabaja el brazo. AL fue diseñado por el laboratorio de Inteligencia Artificial de la Universidad de Stanford, con estructuras de bloques y de control similares al ALGOL (lenguaje en el que fue escrito). Está dedicado al manipulador de Stanford, utilizando como procesadores centrales, a un PDP 11/45 y un PDP KL-10.

HELP

Fue creado por GENERAL ELECTRIC, para su robot ALLEGRO y está escrito en PASCAL/FORTRAN. Este lenguaje permite el movimiento simultáneo de varios brazos y dispone de un conjunto especial de subrutinas para la ejecución de cualquier tarea. Utiliza como CPU un PDP 11.

MAPLE

Fue escrito por IBM, como intérprete en lenguaje PL-1, para el robot de la misma empresa. El cual, tiene capacidad para soportar informaciones de sensores externos. Utiliza como CPU un IBM 370/145 SYSTEM 7.

PAL

Fue desarrollado por la Universidad de Purdure para el manipulador de Stanford. Esta escrito en FORTRAN y ensamblador, y és un intérprete, capaz de aceptar sensores de fuerza y de visión. Cada una de sus instrucciones para mover el brazo del robot en coordenadas cartesianas, es procesada para que satisfaga la ecuación del procesamiento. Como CPU utiliza un PDP 11/70.

MCL

Fue creado por la compañía MC DONALL DOUGLAS, como ampliación de su lenguaje de control numérico APT. Es un lenguaje compilable, apto para la programación de robots "off-line".

MAL EXTENDIDO

Procede del Politécnico de Milán, al igual que el MAL. Incorpora elementos de programación estructurada y se utiliza en el robot SIGMA.

1.6.4. Lenguajes de programación Especificativa a nivel objeto.

De los lenguajes de programación explícita a nivel de objeto destacamos los tres más interesantes: **RAPT, AUTOPASS y LAMA.**

RAPT

El lenguaje RAPT fue creado en la Universidad de Edimburgo, departamento de Inteligencia Artificial; está orientado, en especial, al ensamblaje de piezas. Destinado al robot FREDY, utiliza, como procesador central, a un PDP 10.

Es un intérprete y está escrito en lenguaje APT.

Este lenguaje se basa en definir una serie de planos, cilindros y esferas, que dan lugar a otros cuerpo. Para modelar un cuerpo, se confecciona una biblioteca con sus rasgos más representativos. Seguidamente, se define los movimientos que ligan a los cuerpos a ensamblar (alinear planos, encajar cilindros, etc.).

Un ejemplo:

Si se desea definir un cuerpo C1, se comienza definiendo sus puntos más importantes, por ejemplo:

```
P1 = < x, 0, 0 >
P2 = < 0, y, 0 >
P3 = < x/2, y, 0 >
P4 = < 0, 0, z >
```

Si, en el cuerpo, existen círculos de interés, se especifican seguidamente:

```
C1 = CIRCLE/P2, R;
C2 = CIRCLE/P4, R;
```

A continuación, se determinan sus aristas:

```
L1 = L/P1, P2;
L2 = L/P3, P4;
```

Si, análogamente al cuerpo C1, se define otro, como el C2, una acción entre ambos podría consistir en colocar la cara inferior de C1 alineada con la superior de C2. Esto se escribiría. `AGAINST / BOT / OF C1, TOP / OF C2;`

AUTOPASS

Fue creado por IBM para el ensamblaje de piezas. Utiliza instrucciones, muy comunes, en inglés. Precisa de un ordenador de varios Megabytes de capacidad de memoria.

Además de indicar, como el RAPT, puntos específicos, prevé, también, colisiones y genera acciones a partir de las situaciones reales.

El AUTOPASS realiza todos sus cálculos sobre una base de datos, que define a los objetos como poliedros de un máximo de 20,000 caras. Está escrito en PL/1 y es intérprete y compilable.

Un pequeño ejemplo, que puede proporcionar una idea de la facilidad de relacionar objetos, es el siguiente programa, que coloca la parte inferior del cuerpo C1 alineada con la parte superior del cuerpo C2. Asimismo, alinea los orificios A1 y A2 de C1, con los correspondientes de C2.

```
PLACE C1
SUCH THAT C1 BOT CONTACTS C2TOP
AND B1 A1 IS ALIGNED WITH C2A1
AND B1 A2 IS ALIGNED WITH C2A2
```

LAMA

Fue creado por el laboratorio de Inteligencia Artificial del MIT, para el robot SILVER, en el que se orientaron hacia el ajuste de conjuntos mecánicos. Aporta más inteligencia que el AUTOPASS y permite una buena adaptación al entorno.

La operatividad del LAMA se basa en tres funciones principales:

- Creación de la función de trabajo. Operación inteligente.
- Generación de la función de manipulación.
- Interpretación y desarrollo, de forma interactiva, de una estrategia de realimentación para la adaptación al entorno de trabajo.

1.6.5. Lenguajes de programación En función de los objetivos.

Los lenguajes en función de los objetivos, consisten en definir la situación final del producto a fabricar, a partir de la cual se generan los planes de acción que tienden a conseguirlo, obteniéndose, finalmente, el programa de trabajo. Estos lenguajes, de tipo natural, suponen una potenciación extraordinaria de la Inteligencia Artificial, que descargan al usuario de las labores de programación.

Estos lenguajes pueden realizar una comunicación hombre-máquina, a través de la voz.

De los lenguajes de programación en función de los objetivos destacamos: **STRIPS** y **HILAIRE.**

STRIPS

Fue diseñado, en la Universidad de Stanford, para el robot móvil SHAKEY. Est6á basado en un modelo del universo ligado a un conjunto de planteamientos aritmético-lógicos que se encargan de obtener las subrutinas que conforman el programa final. Es intérprete y compilable, y utiliza como procesadores a un PDP-10 y un PDP-15.

HILAIRE

Fue creado por el laboratorio de Automática y Análisis de Sistemas (LAAS) de Toulouse. Esta escrito en lenguaje LISP. Es uno de los lenguajes naturales más interesantes, por sus posibilidades de ampliación e investigación.

1.6.6. Tabla resumen de los lenguajes más importantes de programación de robots industriales:

LENGUAJE	UNIVERSIDAD /FABRICANTE	ORIGEN Y CARACTERÍSTICAS	APLICACIONES
WAVE (1973)	Stanford	- Nivel actuador - Compilador asociado al robot. - Sintaxis simple: MOVE, SEARCH, CENTER. - Objetos tratados: constantes, variables, vectores, contadores	- Manipulación de piezas mecánicas. - Montaje de bomba de agua. - Ensamblaje.
AL (Assembly Language) (1974)	Stanford	- Basado en PASCAL. - Nivel actuador. - Transformador de coordenadas. - Sintaxis compleja: MOVE TO, MOVE VIA, etc., SEARCH, ACROSS SEARCH, WITHOUT..., CENTER ON ... - Objetos tratados: los de Wave más planos, traslaciones, rotaciones. - Estructura Algol.	- Manipulación de piezas mecánicas. - Sistema de desarrollo de programas Pointy. - Robots Unimation.
RAPT (1978)	Sistemas de Toulouse (Universidad de Edimburgo)	- Interprete escrito en APT. - Nivel objeto.	- Montaje
VAL I (1979) VAL II (1983)	Unimation	- Se partió de AL, pero utilizando BASIC. - Nivel actuador. - Transformador de coordenadas. - Instrucciones de movimiento y de control (MOVE-, DEPART-). - instrucciones Aritméticas. - Estructura Algol.	- Robot Puma de Unimation
MCL	Cincinnati Milacron	- Desarrollo por ICAM y realizado en Fortram	
AML (1979) FUNKY MAPLE AUTOPASS (1977)	IBM	- Alto nivel interactivo estructurado. - Nivel objeto. - Compilador al nivel. - Sintaxis evolucionada, como colocar X sobre Y alineando Z y T.	- Corresponde al nivel de descripción de las gamas de montaje actuales.
LRP	ACMA	- Desarrollado en colaboración con la Universidad de Montpellier.	
V+ (1989)	ADEPT	- Lenguaje textual de alto nivel. - Ejecución de varios programas al mismo tiempo (multitarea). - Proceso asíncrono o ejecución de rutinas de reacción ante determinados eventos.	- Usado en Adept y Staübli
RAPID (1994)	ABB	- Lenguaje textual de alto nivel altamente estructurado. - Estructura modular del programa. - Uso de estructuras predefinidas para especificar la configuración del robot y las características de la herramienta.	
KAREL	FANUC	- Basado en PASCAL. - Soporta la multitarea. - Estructura de datos asociados modificable.	
KRL	KUKA	- Basado en el estándar IRL, descrito en la norma DIN 66312, lo que aporta transportabilidad.	

1.7. CARACTERÍSTICAS DE UN LENGUAJE IDEAL PARA LA ROBÓTICA

Las seis características básicas de un lenguaje ideal, expuestas por Pratt, son:

1. Claridad y sencillez.
2. Claridad de la estructura del programa.
3. Sencillez de aplicación.
4. Facilidad de ampliación.
5. Facilidad de corrección y mantenimiento.
6. Eficacia.

Estas características son insuficientes para la creación de un lenguaje "universal" de programación en la robótica, por lo que es preciso añadir las siguientes:

- Transportabilidad sobre cualquier equipo mecánico o informático.
- Adaptabilidad a sensores (tacto, visión, etc.).
- Posibilidad de descripción de todo tipo de herramientas acoplables al manipulador.
- Interacción con otros sistemas.

En el aspecto de claridad y sencillez, la programación gestual es la más eficaz, pero impide la confección de programas propiamente dichos. Los lenguajes a nivel de movimientos elementales, como el VAL, disponen de bastantes comandos para definir acciones muy parecidas que fueron surgiendo según las necesidades y que, en gran medida, oscurecen su comprensión y conocimiento.
Aunque, inicialmente, las técnicas de programación estructurada son más difíciles de dominar, facilitan, extraordinariamente, la comprensión y corrección de los programas.

Respecto a la sencillez de aplicación, hay algunos lenguajes (como el MCL) dedicados a las máquinas herramienta (APT), que pueden ser valorados, positivamente, por los usuarios conocedores de este campo. El PAL, estructurado sobre la matemática matricial, sólo es adecuado para quienes están familiarizados con el empleo de este tipo de transformaciones.

Uno de los lenguajes más fáciles de utilizar es el AUTOPASS, que posee un juego de comandos con una sintaxis similar a la del ingles corriente.

Es imprescindible que los lenguajes para los robots sean fácilmente ampliables, por lo que se les debe dotar de una estructura modular, con inclusión de subrutinas definidas por el mismo usuario.

La adaptabilidad a sensores externos implica la posibilidad de una toma de decisiones, algo muy interesante en las labores de ensamblaje. Esta facultad precisa de un modelo dinámico del entorno, así como de una buena dosis de Inteligencia Artificial, como es el caso del AUTOPASS.

Aunque los intérpretes son más lentos que los compiladores, a la hora de la ejecución de un programa, resultan más adecuados para las aplicaciones de la robótica. Las razones son las siguientes:

1) El intérprete ejecuta el código como lo encuentra, mientras que el compilador recorre el programa varias veces, antes de generar el código ejecutable.
2) Los intérpretes permiten una ejecución parcial del programa.
3) La modificación de alguna instrucción es más rápida con intérpretes, ya que un cambio en una de ellas no supone la compilación de las demás.

Finalmente, el camino para la superación de los problemas propios de los lenguajes actuales ha de pesar, necesariamente, por la potenciación de los modelos dinámicos del entorno que rodea al robot, acompañado de un aumento sustancial de la Inteligencia Artificial.

1.8. CLASIFICACIÓN DE LA PROGRAMACIÓN USADA EN ROBÓTICA

La programación empleada en Robótica puede tener un **carácter explícito**, en el que el operador es el responsable de las acciones de control y de las instrucciones adecuadas que las implementan, o estar **basada en la modelación del mundo exterior**, cuando se describe la tarea y el entorno y el propio sistema toma las decisiones.

La programación explícita es la más utilizada en las aplicaciones industriales y consta de dos técnicas fundamentales:

1. **Programación Gestual o Directa (Guiado)**: consiste en guiar el brazo del robot directamente a lo largo de la trayectoria que debe seguir. Los puntos del camino se graban en memoria y luego se repiten. Este tipo de programación, exige el empleo del manipulador en la fase de enseñanza, o sea, trabaja "on-line".

   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 la programación "on-line".

   Los lenguajes de programación gestual, además de necesitar al propio robot en la confección del programa, carecen de adaptabilidad en tiempo real con el entorno y no pueden tratar, con facilidad, interacciones de emergencia. La programación gestual se subdivide en dos clases:

   - **Programación por aprendizaje directo**. El punto final del brazo se traslada con ayuda de un dispositivo especial colocado en su muñeca, o utilizando un brazo maestro o maniquí, sobre el que se efectúan los desplazamientos que, tras ser memorizados, serán repetidos por el manipulador.

     La técnica de aprendizaje directo se utiliza, extensamente, 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 aprendizaje 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.

   - **Programación mediante un dispositivo de enseñanza (botonera)**. 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.

     El dispositivo de enseñanza suele estar constituido por botones, teclas, pulsadores, luces indicadoras, ejes giratorios o "joystick". 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.
- Generación de retardos.
- Señalización del estado de los sensores.
- Borrado y modificación de los puntos de trabajo.
- Funciones especiales.

Al igual que con la programación directa, 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.

2. **Programación Textual**: En la programación textual, las acciones que ha de realizar el brazo se especifican mediante el programa, que consta de un texto de instrucciones o sentencias (en un lenguaje determinado), cuya confección no requiere de la intervención del robot; es decir, se efectúan "off-line". Así mismo, con este tipo de programación, el operador no define, prácticamente, las acciones del brazo manipulado, sino que se calculan, en el programa, mediante el empleo de las instrucciones textuales adecuadas.

   Las trayectorias del manipulador se calculan matemáticamente con gran precisión y se evita el posicionamiento a ojo, muy corriente en la programación gestual. En esta labor no participa la máquina (off-line).

   Los lenguajes de programación textual se encuadran en varios niveles, según se realice la descripción del trabajo del robot. Estos son los lenguajes de programación por orden creciente de complejidad:
   - Lenguajes elementales, que controlan directamente el movimiento de las articulaciones del manipulador
   - Lenguajes dirigidos a posicionar el elemento terminal del manipulador.
   - Lenguajes orientados hacia el objeto sobre el que opera el sistema.
   - Lenguajes enfocados a la tarea que realiza el robot.

   En una aplicación tal como el ensamblaje de piezas, en la que se requiere una gran precisión, los posicionamientos seleccionados mediante la programación gestual 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 debe intervenir, sólo, en la puesta a punto final.

   Según las características del lenguaje, pueden confeccionarse programas de trabajo complejos, 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, etc.

   Dentro de la programación textual, existen dos grandes grupos, de características netamente diferentes:

1. **Programación textual explícita:** el programa consta de una secuencia de órdenes o instrucciones concretas, que van definiendo con rigor las operaciones necesarias para llevar a cabo la aplicación. Se puede decir que la programación explícita engloba a los lenguajes que definen los movimientos punto por punto, similares a los de la programación gestual, pero bajo la forma de un lenguaje formal. Con este tipo de programación, la labor del tratamiento de las situaciones anormales, colisiones, etc., queda a cargo del programador.

   Dentro de la programación explícita, hay dos niveles:

   a) Nivel de movimiento elemental: comprende los lenguajes dirigidos a controlar los movimientos del brazo manipulador. Existen dos tipos:

   - **Articular**, cuando el lenguaje se dirige al control de los movimientos de las diversas articulaciones del brazo. **Los lenguajes del tipo articular** indican los incrementos angulares de las articulaciones. Aunque esta acción es bastante simple para motores de paso a paso y corriente continua, al no tener una referencia general de la posición de las articulaciones con relación al entorno, es difícil relacionar al sistema con piezas móviles, obstáculos, cámaras de TV, etc.

     Los lenguajes correspondientes al nivel de movimientos elementales aventaja, principalmente, a los de punto a punto, en la posibilidad de realizar bifurcaciones simples y saltos a subrutinas, así como de tratar informaciones sensoriales.
   - **Cartesiano**, cuando el lenguaje define los movimientos relacionados con el sistema de manufactura, es decir, los del punto final del trabajo (Tool Center Point). **Los lenguajes del tipo cartesiano** utilizan transformaciones homogéneas. Este hecho confiere "popularidad" al programa, independizando a la programación del modelo particular del robot, puesto que un programa confeccionado para uno, en coordenadas cartesianas, puede utilizarse en otro, con diferentes coordenadas, mediante el sistema de transformación correspondiente. Son lenguajes que se parecen al BASIC, sin poseer una unidad formal y careciendo de estructuras a nivel de datos y de control.

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

   Se puede decir que los lenguajes correspondientes a este tipo de programación adoptan la filosofía del PASCAL, LABVIEW, o cualquier programa parecido. 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, se utilizan mucho las transformaciones de coordenadas, que exigen un cierto nivel de conocimientos. Por este motivo dichos lenguajes no son populares hoy en día.

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 se ha de disponer del modelo de donde se encuentra el robot. Este modelo será, normalmente, una base de datos más o menos compleja, requiriendo siempre computadoras potentes para el procesado de una abundante información.

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

   Dentro de la programación textual especificativa, hay dos clases, según la orientación a la que se refiera el modelo, objeto o objetivos:

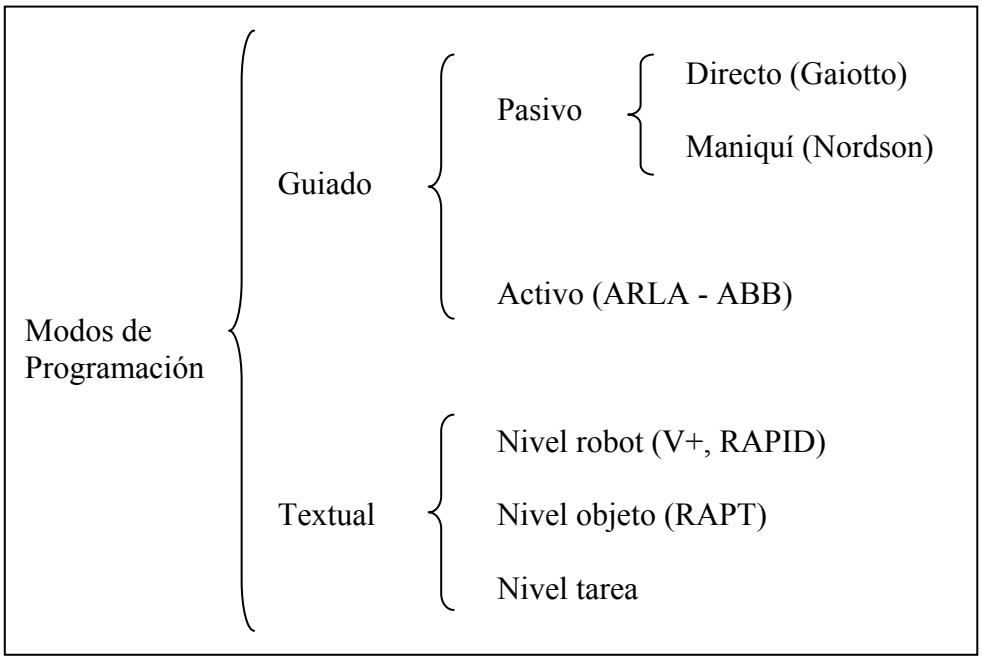
- Si el modelo se orienta al **nivel de los objetos**, el lenguaje trabaja con ellos y establece las relaciones entre ellos. La programación se realiza "off-line" y la conexión CAM es posible.

  Dada la inevitable 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.
- Cuando el **modelo se orienta hacia los objetivos**, se define el producto final.

  La creación de lenguajes de muy alto nivel transferirá una gran parte del trabajo de programación, desde el usuario hasta el sistema informático; éste resolverá la mayoría de los problemas, combinando la Automática y la Inteligencia Artificial.

Otra posible clasificación de los modos de programación que hemos encontrado es la que muestra el cuadro siguiente:



2. EL LENGUAJE V+

El lenguaje V+ aparece como mejora al lenguaje VAL II, el cual, a su vez, reemplazó al lenguaje VAL. Así mismo, el V+ entra en la categoría de los denominados segunda generación de lenguajes de programación para robots industriales.

2.1 EL LENGUAJE DE ROBÓTICA INDUSTRIAL VAL

El VAL (Victor's Assembly Language, después Vistor Scheiman), fue el primer lenguaje para robots comercialmente disponible, que tiene parte de los conceptos del WAVE y del AL .VAL fue introducido en 1979 por Unimation, Inc. Para su serie de robots PUMA. Este lenguaje forma parte de la primera generación de lenguajes de programación de robots.

El VAL (y los lenguajes de la primera generación) tiene limitaciones como la incapacidad para realizar cálculos aritméticos complejos para su uso durante la ejecución del programa, la incapacidad para hacer uso de sensores complejos y de los datos de los sensores y la capacidad limitada para comunicarse con otras computadoras. Tampoco pueden ampliarse para posibles mejoras futuras.

2.2 EL LENGUAJE DE ROBÓTICA INDUSTRIAL VAL II

El VAL II, reemplazó al VAL y fue lanzado al mercado en 1984.

Este lenguaje, forma parte de la segunda generación de lenguajes de programación de robots. VAL II (y los lenguajes de la segunda generación), salvan algunas limitaciones de los lenguajes de la primera generación y las añaden a estas capacidades mediante la incorporación de características que hacen que el robot parezca más inteligente. Activan el robot para que realice tareas más complejas.

A estos lenguajes de la segunda generación, se les ha llamado lenguajes de programación estructurada, porque poseen las construcciones de control estructuradas utilizadas en los lenguajes de programación de la computadora.

Este lenguaje tiene las siguientes características y capacidades:

1. Control de movimiento. Esta característica es básicamente la misma que para los lenguajes de la primera generación, pero a veces, va más allá por su inclusión de problemas geométricos más complejos que la interpolación de la línea recta.
2. Capacidad de sensor avanzadas. Entre las mejoras de estos lenguajes de la segunda generación, se incluye la capacidad para tratar con más que señales binarias simples (todo o nada) y la capacidad para controlar los dispositivos por medio de los datos del sensor.
3. Inteligencia limitada. Esta capacidad utiliza la información recibida sobre el entorno de trabajo para modificar el comportamiento del sistema de una manera programada.
4. Comunicaciones y procesamiento de datos. Los lenguajes de esta segunda generación suelen tener los medios adecuados para interactuar con computadoras y bases de datos de computadora con el propósito de almacenar registros, generar informes y controlar las actividades de la célula de trabajo.

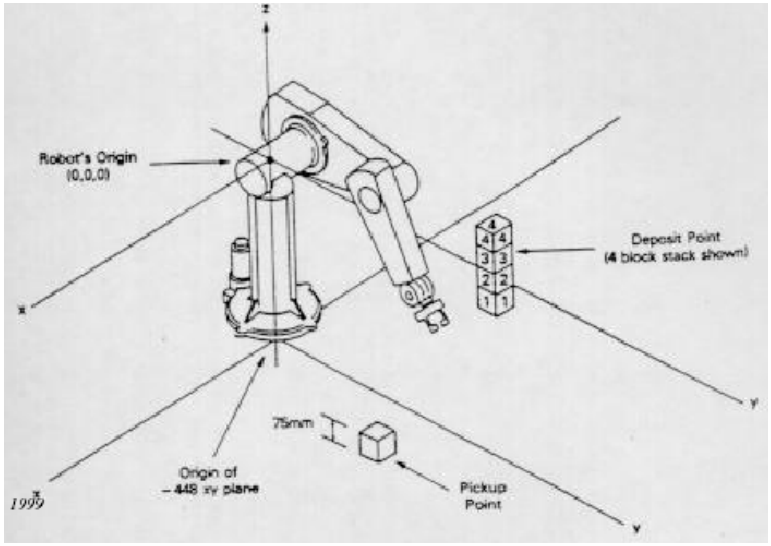
Las aportaciones más relevantes de VAL II son:

- Al concepto de programa de control del robot añade el de programa de control de proceso.
- Prioridad relativa entre procesos asíncronos.
- Admite las cinco estructuras clásicas de control.
- Comunicación con un ordenador central (*host*).

La comunicación con un "host", aumenta enormemente las posibilidades del sistema ya que pone al alcance del robot toda la potencia de cálculo y de gestión de información procedente de sensores representada por el "host".

Se puede así empezar a pensar para los robots de Unimation en tareas coordinadas con sistemas de visión, en aplicación de técnicas de I.A. para el control de la actuación de manipuladores, etc. Se abre así un nuevo campo de posibilidades en tareas de robótica inteligente. [Ferraté et al., 86].

En resumen, VAL II es un lenguaje y sistema de control basado en computadora diseñado para los robots industriales Unimation. Proporciona la capacidad para facilitar la definición de la tarea que va a realizar un robot, puesto que las tareas se definen mediante los programas escritos por el usuario. Incluyéndose, también, la capacidad para responder a la información a partir de sensores tales como la visión de máquina, el perfeccionamiento en la ejecución del movimiento del brazo y el trabajo en las situaciones imprevisibles o utilizando sistemas de referencia.



Ejemplo de programa para apilar bloques VAL II

```
1. REMARK Este programa
… REMARK recoge piezas
6. REMARK y las apila.
7. OPENI
8. SET B = DEPOSIT
9. SETI COUNT = 0
10. 10 APPROS PICKUP, 200.00
11. MOVES PICKUP
12. CLOSEI
13. DEPARTS 200.00
14. APPRO B, 200.00
15. MOVES B
16. OPENI
17. DEPARTS 200.00
18. SETI COUNT = COUNT + 1
19. TYPEI COUNT
20. REMARK COUNT indica el nº tot de apilamientos
21. IF COUNT EQ 4 THEN 20
22. REMARK mover a B incrementando z en 75.00 mm
23. SHIFT B BY 0.00, 0.00, 75.00
24. GOTO 10
25. 20 SPEED 50.00 ALWAYS
26. READY
27. TYPE
*** FIN DE PROGRAMA APILAR ***
```

Significado de las líneas de programa anteriores:

1 a 6 son comentarios
7 da la orden de abrir la pinza del elemento terminal
8 asigna a la variable B el valor de la localización donde se desean depositar los objetos
9 inicializa la variable entera COUNT
10 mueve el elemento terminal en línea recta desde la posición actual hasta una que dista 200 mm sobre el eje Z de la herramienta en el punto donde se desea recoger el objeto
11 realiza un movimiento en línea recta que sitúa el elemento terminal en el punto PICKUP
12 cierra la pinza
13 se aleja de la posición anterior 200 mm en línea recta sobre el eje Z de la herramienta
14 realiza un movimiento de aproximación hasta 200 mm del punto B
15 mueve el elemento terminal en línea recta hasta B
16 abre la pinza
17 se aleja en línea recta de B
18 aumenta una unidad el valor del contador COUNT
19 muestra el valor actual del contador COUNT
21 expresión condicional para comprobar si ha terminado el traslado de los cuatro objetos
23 modifica la localización de B en 75 mm según la coordenada Z
24 vuelve a la línea con el indicador 10
25 cambia la velocidad (50 % de la nominal)
26 lleva al manipulador a la posición de reposo
27 escribe el mensaje de final de programa

2.3 EL LENGUAJE DE ROBÓTICA INDUSTRIAL V+

El lenguaje V+ es un lenguaje de programación textual de alto nivel, desarrollo en 1989 por Adept Technology. Es una evolución del VAL II, y parece ser que se denomina indistintamente V+ y VAL III (ver fotografía).

Actualmente, se utiliza principalmente en las marcas de robots industriales Adept y en algunos robots Stäubli, como por ejemplo en el Rx 90 (ver fotografía).

El lenguaje V+ proporciona una gran interacción entre el hombre y el robot. Presenta las siguientes aportaciones más relevantes:

• Inteligibilidad: Nos puede proporcionar una buena documentación, además de un diseño ordenado y coherente del programa.

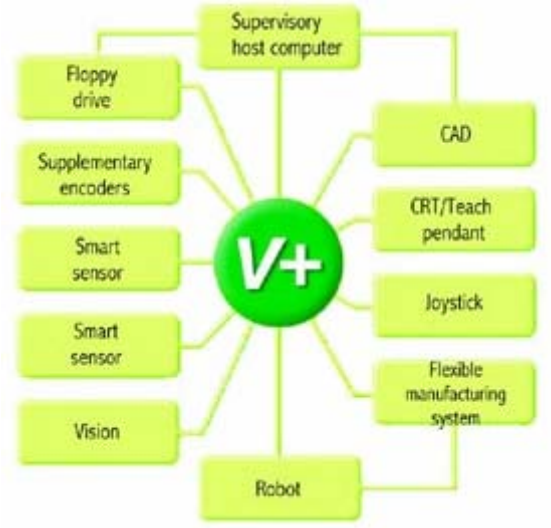
• Fiabilidad: Sobre todo en sistemas que deben responder a situaciones imprevistas.

• Adaptabilidad: Se pueden mejorar, ampliar y modificar los programas con poco esfuerzo.

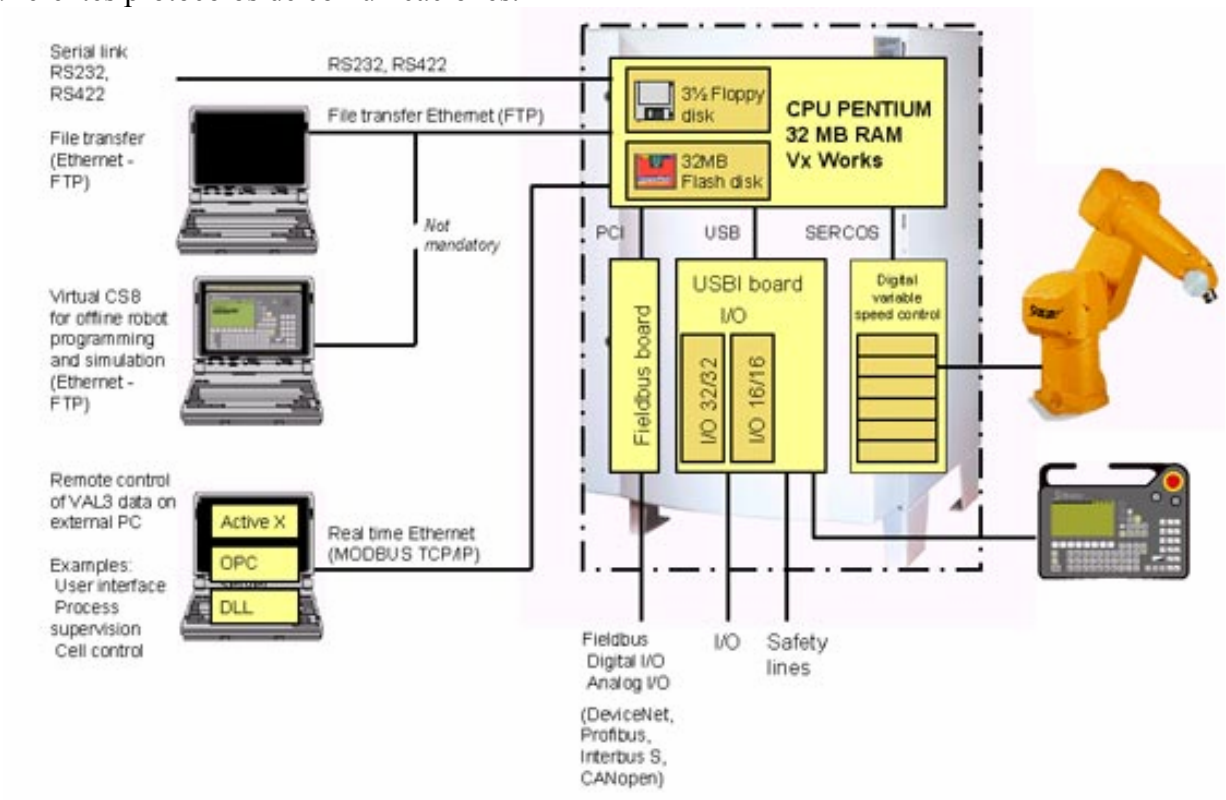
• Transportabilidad: Los programas se pueden desarrollar en computadoras (ej: un PC) para poder introducirlos, posteriormente, en el controlador del sistema a través de un disquete.

Así mismo, V+ es muy versátil, contando con las siguientes características:

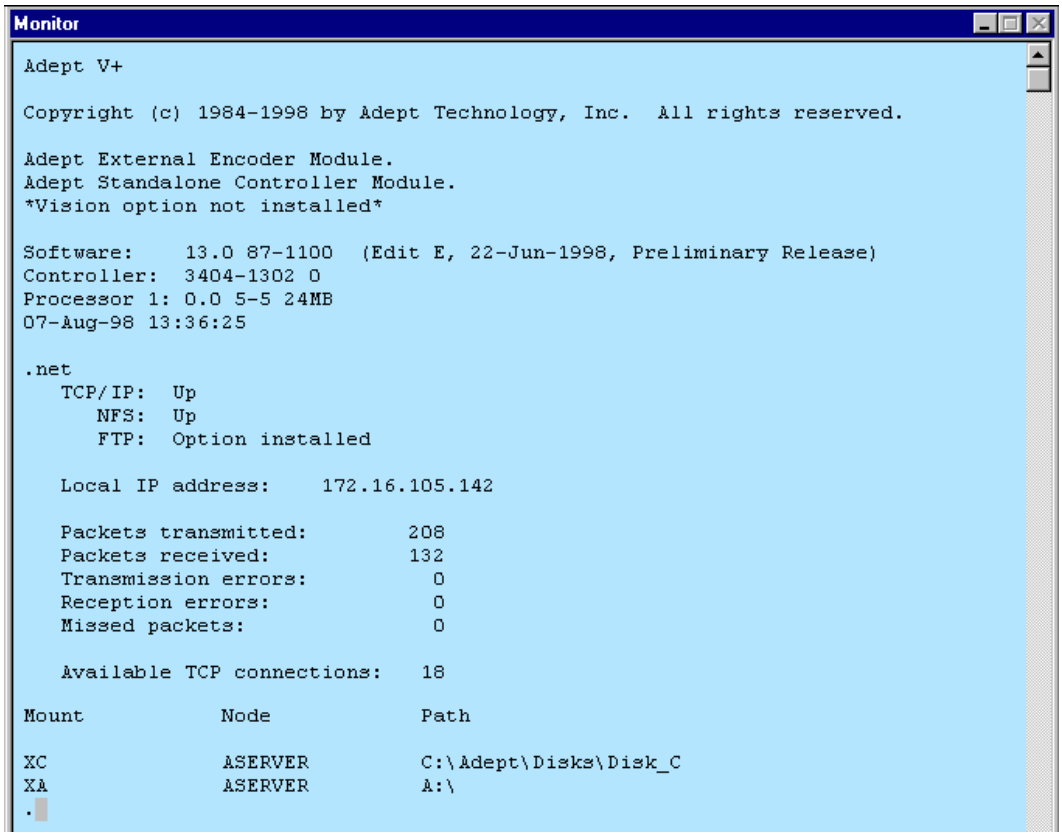
- Rápido y alto nivel de interpretación del lenguaje en las aplicaciones.
- Multitarea del sistema a tiempo real.
- Sistema de vision integrada en el proceso en el mismo programa V+
- Arquitectura abierta de diseño, pudiendo añadirse fácilmente nuevos periféricos, redes, etc.



V+ usa un protocolo KERMIT y el programa de aplicación XMODEM que permite la transferencia de programas entre el controlador Adept y otro ordenador a través de la línea serie. Así mismo el sistema es abierto, y permite comunicación entre diferentes elementos y mediante diferentes protocolos de comunicaciones.

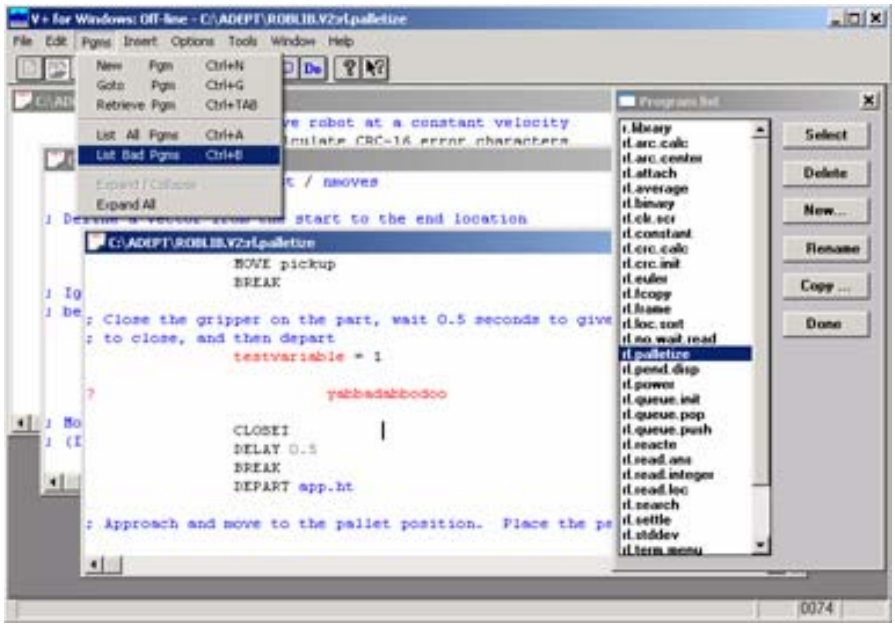


A continuación, se puede ver una pantalla capturada del V+ donde se muestra la configuración ETHERNET TCP/IP y el estado de la comunicación de la controladora del robot:



Para una programación más elaborada, el V+ lenguaje interpretado y estructurado de alto nivel, ofrece al programador entre otras, las siguientes posibilidades:

- Programas de aplicación accesibles al operador mediante mouse, ventanas y gráficos en color.
- Modificación de la trayectoria en tiempo real, permitiendo adaptar la trayectoria en función de los sensores externos, tales como la visión, sensor de esfuerzo, cinta transportadora.
- Acceso a comunicación con el BUS VME.
- Cálculos booleanos, funciones matemáticas, funciones geométricas, tales como transformadas compuestas, "frame", corrección de herramientas, etc..
- Funciones gráficas para crear ventanas, iconos, gestión de eventos,...
- Construcción estructurada.
- Movimientos continuos.
- 7 tareas paralelas (con posible extensión a 28 tareas).
  Acceso a *256* entradas/ *256* salidas digitales externas, a 28 entradas / 16 salidas analógicas, 4 (ó 5) puertos de comunicación serie.
- Gestión de ejes suplementarios y de encoders externos
- Matrices de hasta 3 dimensiones.
- Variables numéricas de 32 bits, coma flotante, simple y doble precisión.
  Manipulación de cadenas de caracteres. Herramientas de detección de errores.
- Los programas pueden ser escritos off-line en otros ordenadores y transferidos mediante disquetes en formato PC.
- Subrutinas recursivas y reentrantes con paso de parámetros



Los programas de V+ consisten en conjuntos de instrucciones. Cada línea se interpreta como una instrucción del programa. El formato típico de cada línea:

Número _ línea *etiqueta* *instrucción* *;comentario*

Cada línea de programa tiene asignado un número de línea, que se ajusta automáticamente cuando se insertan o borran líneas. Para identificar a una línea de puede colocar una etiqueta (un número entre 0 y 65535) en el principio de línea. Después de la etiqueta se puede colocar las instrucciones (opcional).

Se pueden incluir comentarios después del carácter ";".

El editor de V+, comprueba automáticamente la sintaxis, si es correcta, cada vez que introducimos una línea.

La primera instrucción que aparecerá en nuestro programas será ".PROGRAM", seguida del nombre del programa. Tendrá el siguiente formato:

.PROGRAM nombre (lista de parámetros)

El *nombre* del programa debe ponerse, mientras que la *lista de parámetros* son opcionales.
El final del programa se indica con la instrucción ".END."

El lenguaje V+ se pueden usar diferentes variables, dependiendo de la forma en la que se acceden a ellas. Se pueden clasificar de tres maneras:

• **Globales**: Todos los programas en memoria tienen acceso a estas variables. Su mal uso puede provocar que el valor que tiene la variable sea modificada por una subrutina u otro programa. Para que no sucedan estas modificaciones indeseables, se utilizan variables locales y automáticas.

• **Locales**: Las variables pueden ser locales con el uso de la instrucción LOCAL. Esta variable se crea cuando le ponemos la instrucción LOCAL, y mantiene su valor entre llamadas de programas (mantiene su valor aunque salgamos fuera del programa, hasta la siguiente vez que entramos en el programa). Para que no ocurran conflictos al usar diferentes tareas con esas variables se crean las variables automáticas. La desventaja de estas variables es que ocupan espacio en memoria aunque no se ejecute el programa.

• Programas de comandos del monitor (**Automáticas**): Estas variables se crean dentro de los programas con la instrucción AUTO. Se parecen a las variables locales porque solo se puede acceder a ellas dentro del programa donde residen. Cuando se entra dentro del programa la variable automática, se crea una copia separada de cada variable automática, y el valor de esta copia se pierde cuando se sale del programa. Si la variable automática se utiliza en diversas tareas a la vez, sus valores son independientes en cada momento, con lo que no provocan conflictos. La ventaja de esta variable es que cuando se sale del programa se libera la memoria.

Los argumentos de las subrutinas son un caso especial de variable automática. El alcance de las subrutinas es el mismo que el de las variables automáticas.

El lenguaje V+ permite especificar el contexto de las variables locales y automáticas con los comandos HERE, POINT, TEACH, et. Tienen el siguiente formato:

Comando @tarea:programa parámetros

El *comando* representa el nombre de un comando, y *parámetros* representa los parámetros normales del comando.

El elemento *@tarea:programa* (opcional) especifica el contexto para las variables referenciadas en los parámetros del comando.

El elemento *tarea* es un entero y representa una de las tareas del sistema.

El elemento *programa* representa el nombre de un programa en la memoria del sistema.

Existen tres tipos diferentes de programas que podemos realizar con el V+, dependiendo de los comandos o tipos de instrucciones que utilizamos, y acciones que llevamos a cabo. Los tipos de programas son los siguientes

- Programas de **control** del robot: Estos programas controlan directamente el robot y pueden contener cualquier instrucción, además de las instrucciones de movimiento del robot. Se suelen ejecutar con la tarea principal (numero 0), pero también pueden ejecutarse con cualquier tares del sistema.

- Programas de control de propósito **general**: Estos programas no son controlados por el robot, y pueden ejecutarse al mismo tiempo. Pueden controlar procesos externos a través delineas binarias externas y comunicarse con el programa de control mediante variables compartidas y señales de software.
  Estos programas no pueden ejecutar instrucciones que afecten al movimiento del robot, directamente. Aunque, estos programas pueden cambiar de clasificación durante su ejecución, pasando de ser un programa de propósito general a un programa de control, o viceversa.

- Programas de comandos del **monitor**: Estos programas se componen de comandos de monitor, más que de instrucciones de programa y, se usan, para realizar secuencias de comandos del monitor.

El lenguaje de programación V+ permite la ejecución de diferentes programas al mismo tiempo. Un ejemplo de esto, es la posibilidad de ejecutarse a la vez, un programa de control y otros programas adicionales (por ejemplo controlar el robot y la cinta transportadora simultáneamente). El sistema de V+ ejecuta cada programa como una tarea independiente y cuenta con siete tareas disponibles. Dependiendo del tipo de programa a realizar o de la tarea que se desee ejecutar, ser ejecutará de diversas maneras. La tarea con más alta prioridad y más significativa es la número 0, ya que es la que utilizamos, generalmente, para la ejecución del programa de control del robot.

Las instrucciones se suelen ejecutar de manera secuencial. También existen instrucciones que pueden variar el flujo de la ejecución del programa como GOT, CALL, IF...GOTO, WAIT, STOP, etc.

Una característica particular del V+ es el proceso asíncrono, el cual consiste en la capacidad del sistema de responder a eventos cuando estos ocurren. Cuando aparecen los eventos (si esta activada la opción) se ejecuta un programa específico, llamado rutina de reacción (como una instrucción CALL).

Se le llama Proceso Asíncrono porque la ejecución no está sincronizada con el flujo normal del programa. Estos procesos asíncronos se habilitan con las instrucciones REACT, REACTIVE y REACTI. Estas instrucciones son usadas para cada una de las tareas, para preparar el procesado independiente de los eventos.

Una rutina de reacción se ejecuta, solamente, si la prioridad del programa principal es menor que la de reacción.

El programa principal se ejecuta, normalmente con prioridad 0, mientras que la menor prioridad que puede tener una reacción es uno. De esta manera, una reacción puede interrumpir el programa principal, normalmente.

Cuando ejecutamos una rutina de reacción, la prioridad del programa se ajusta automáticamente a la prioridad de la reacción para evitar que sea interrumpido por reacciones de igual o menor prioridad.

Al salir de la reacción, el programa principal recupera la prioridad que tenia.

2.4. Uso del entorno de programación V+

El robot SCARA Adept One de Inser Robótica, S.A. posee un lenguaje propio, el lenguaje V/V+, basado en posiciones del espacio tridimensional.

Inicio del sistema

La sesión se inicia poniendo en marcha el controlador. Para ello, se debe situar la palanca de encendido de la derecha en la posición **ON** y a continuación el interruptor verde, *SYSTEM POWER*, que suministra energía al sistema, en la posición **1**.

Una vez iniciado el sistema, en el terminal se visualizará el *prompt* que consiste en un punto. El primer comando a introducir debe ser:

.ENABLE POWER

para activar los circuitos del brazo robótico. Este comando también se utiliza para volver a reactivar el robot cuando pulsamos la seta de emergencia o se detiene debido a algún tipo de problema, como por ejemplo una colisión.

Una vez activado el robot, se debe calibrar el mismo utilizando para ello el comando:

.CALIBRATE

El sistema pide confirmación de la operación de calibrado, ya que ésta tarda un tiempo en llevarse a cabo. La pregunta que hace es: *"ARE YOU SURE (Y/N)?"*, a la que se debe responder con **Y**. Es obligatorio realizar esta operación sólo una vez al encender
la máquina.

Estos dos comandos también se pueden sustituir por sus abreviaturas:

.EN POW
.CAL

Comandos para ficheros

Cada uno de los ficheros almacenados en el disco duro contiene un conjunto de datos y de programas.

Cada vez que se carga un fichero del disco, se produce un volcado de su contenido a la memoria activa del sistema, de tal manera que se pueden ejecutar los programas que contiene.

Los siguientes comandos permiten manejar ficheros:

.FDIRECTORY C:
Este comando muestra el contenido del disco C:

.LOAD C:<FICHERO>.V2
Carga en memoria el contenido del fichero especificado siempre y cuando se encuentre en C:

.FLIST C:<FICHERO>
Lista el contenido del fichero especificado. En el caso de que queramos interrumpir el listado, lo podremos hacer pulsando Ctrl+C.

.STORE C:<FICHERO>.V2

Almacena en el fichero especificado todos los programas y datos activos que se encuentren en memoria.

.FRENAME C:<NONBRE_ANTIGUO>.V2=<NONBRE_NUEVO>.v2

Permite cambiar el nombre a un fichero almacenado en disco.

Actualización de ficheros en disco

No se puede actualizar un fichero ya existente en disco. Para ello es necesario realizar tres pasos: grabar la memoria actual en otro fichero, borrar el anterior y cambiar el nombre del nuevo.

Por ejemplo, si queremos guardar nuestro trabajo en un fichero que se llama *ANTIGUO.V2* que ya existe en el disco, deberemos hacer lo siguiente:

.STORE C:NUEVO.V2
.FDELETE C:ANTIGUO.V2
.FRENAME C:ANTIGUO.V2=NUEVO.V2

Comandos para programas

.DIRECTORY
Muestra todos los programas que hay en la memoria activa.

.SEE <PROGRAMA>

Es el editor de textos. Con él podremos editar y crear programas. En el caso de que *<PROGRAMA>* no exista, nos preguntará si queremos crearlo.

Teclas importantes del *SEE*:

- *I:* Permite insertar líneas. Es el *INSERT MODE*.
- *F4:* Sale del editor.

El editor *SEE*, además de permitir la edición de programas, realiza una labor de interprete el código a la vez que éste se va escribiendo. Cualquier error en una línea del programa se indica con una interrogación, **?**. Una vez creado un programa con *SEE*, si no contiene errores, no es necesario compilarlo.

El programa que se haya creado nuevo se mantiene en la memoria del sistema.

.EXECUTE <PROGRAMA>

Ejecuta el programa que le pasemos como *<PROGRAMA>*.

Otros comandos

.HERE <NOMBRE_DE_VARIABLE>

Guarda en la variable las coordenadas de la posición actual del robot.

.WHERE

Muestra las coordenadas de la posición actual de robot.

.DISABLE POWER

Desactiva el robot (*DIS POW*).

.STATUS 0

Muestra en pantalla el estado actual del manipulador.

Formato de las posiciones. Sentencia *SET-TRANS*

Una posición en V/V+ consta de una tupla de seis parámetros *(X, Y, Z, y, p, r)*, donde **X**, **Y**, **Z** representan las coordenadas cartesianas absolutas en **milímetros**; **y** (yaw) representa un deslizamiento o rotación de la garra en el eje X (siempre vale 0º); **p** (pitch) representa el cabeceo o rotación de la garra en el eje Y (siempre vale 180º); y **r** (roll) representa el balanceo o rotación de la garra, en grados, en el eje Z. Este último parámetro puede tomar cualquier valor, ya que se corresponde con uno de los grados de libertad del robot.

La sentencia *SET-TRANS* se utiliza para definir una variable, dentro de la memoria activa del sistema, en la que se almacenará las coordenadas absolutas del espacio de operaciones del brazo robótico.

Por ejemplo, una posición a la que llamaremos *garra1* se define en un programa de la siguiente manera:

SET garra1 = TRANS (123.789, -488.511, 755, 0, 180, -61.87)

Sentencias e instrucciones de V/V+

A continuación se comenta el conjunto de sentencias básicas del lenguaje de programación V/V+. Cada una de estas instrucciones debe ir en una línea del programa.
Si se quiere incluir *comentarios*, se deben poner detrás del carácter punto y coma, ";".

SPEED <VALOR> [ALWAYS]

Establece la velocidad a la que queremos que se mueva el robot. *<VALOR>* indica un porcentaje de la velocidad máxima.

Si se especifica el término *ALWAYS*, entonces mantendrá la velocidad hasta el final del programa o hasta que encuentre otra sentencia *SPEED*. Generalmente, cuando se hacen pruebas, se suele establecer la velocidad siempre al 25%: *SPEED 25 ALWAYS*

RIGHTY

Hace que el robot se comporte como un brazo derecho.

LEFTY

Hace que el robot se comporte como un brazo izquierdo. Generlmente se utiliza **siempre esta disposición del brazo** para evitar colisiones.

MOVE <VARIABLE_DE_POSICIÓN>

Desplaza la garra del robot hasta la posición indicada por la variable de posición.
Para realizar este movimiento utiliza una interpolación que implique el **menor movimiento de ejes**, es decir, que para llegar al punto de destino no se mueve en línea recta, sino que se desplaza de forma que las rotaciones de las articulaciones del brazo se muevan lo mínimo.

MOVES <VARIABLE_DE_POSICIÓN>

Desplaza la garra del robot hasta la posición indicada por la variable de posición.

Para realizar este movimiento utiliza una interpolación **lineal**, es decir, que para llegar al punto de destino va a obligar a la garra a desplazarse en línea recta. Esto es muy útil para sacar las garras de su plataforma.

APPRO <VARIABLE_DE_POSICIÓN>, <VALOR>

Mueve el robot a la posición indicada por la variable de posición, pero aumentando o disminuyendo la coordenada Z con el valor especificado. El movimiento se realiza mediante interpolación **lineal**.

DEPARTS <VALOR>

Incrementa o decrementa la coordenada Z con el valor especificado.

OPENI

Abre los enganches para coger o dejar la garra.

CLOSEI

Cierra los enganches para coger la garra, si no la tiene cogida, o cierra la garra, si la tiene cogida.

RELAXI

Abre la garra que debe tener cogida.

DELAY(<VALOR)

Establece un retardo de tiempo medio en segundos.

CALL <NOMBRE_DE_PROGRAMA>

Permite desde un programa llamar a otro, <*NOMBRE_DE_PROGRAMA*>, de modo que se desvía el flujo de ejecución al programa indicado, retornando cuando haya finalizado.

Comando *DO*

Éste es un comando especial que nos permite utilizar las instrucciones del lenguaje de programación anteriores desde fuera de un programa, como si fuesen comandos.

Por ejemplo, si queremos desde el *prompt* definir una posición llamada *SAFE* y mover el brazo del terminal hasta ella, se escribiría

.***DO*** *SET SAFE=TRANS(560, -560, 855, 0, 180, -124)*
.***DO*** *MOVE SAFE*

2.5. Tablas resumen de Instrucciones V+.

Tabla de operaciones de control de programas:

Palabra Clave	Tipo	Función	Estructura
ABORT	Instrucción de Programa	Terminal de ejecución de programa	GOTO label
CALL	Instrucción de Programa	Suspende la ejecución del programa actual y continua la ejecución con un nuevo programa (es decir, un subprograma).	CALL program (arg_list) Ej: CALL check_data(locx, locy, length)
CALLS	Instrucción de Programa	Suspende la ejecución del programa actual y continua la ejecución con un nuevo programa (es decir, un subprograma) especificado con un valor de la secuencia.	$program_name = $program_list[program_select] Ej: CALLS $program_name(length, width)
CASE	Instrucción de Programa	Proceso iniciado con una estructura del CASE definiendo el valor de interés.	Ej: CASE target OF VALUE list_of_values: code block (executed when target is in list_of_values) VALUE list_of_values: code block (executed when target is in list_of_values) ... ANY code block (executed when target not in any list_of_values) END
CLEAR.EVENT	Instrucción de Programa	Despeja un acontecimiento asociado a la tarea especificada.	
CYCLE.END	Instrucción de Programa	Termina el programa de control especificado la próxima vez que ejecuta una instrucción del programa de PARADA (o su equivalente). Suspenda el proceso de un programa en uso o del programa del comando hasta que un programa termina la ejecución.	
DO	Instrucción de Programa	Introduce una estructura DO.	EJ: DO. code block . UNTIL expression
EXECUTE	Instrucción de Programa	Empieza la ejecución de un programa de control.	
EXECUTE	Comando de Monitor	Empieza la ejecución de un programa de control.	
EXIT	Instrucción de Programa	Sale de una estructura de control FOR, DO ...WHILE.	
FOR	Instrucción de Programa	Ejecuta un grupo de instrucciones de programa un cierto número de veces.	Ej: FOR index = start_val TO end_val STEP incr . code block . END
GET.EVENT	Función de valor Real	Devuelve acontecimientos que se fijan para la tarea especificada.	
GOTO	Instrucción de Programa	Realiza un parte incondicional del programa identificado por la etiqueta dada.	
HALT	Instrucción de Programa	Para la ejecución de programa y no permite que el programa sea reasumido.	
IF...GOTO	Instrucción de Programa	Mira si el valor de la etiqueta especificada de una expresión lógica es TRUE (distinto a cero).	IF logical_expression GOTO 100
IF...THEN	Instrucción de Programa	Ejecuta un grupo de instrucciones condicionales (o uno de dos grupos) dependiendo del resultado de una expresión lógica.	Ej: IF expression THEN code block (executed when expression is true) ELSE code block (executed when expression is false) END

INT.EVENT	Instrucción de Programa	Envía una instrucción de SET.EVENT a la tarea actual si una interrupción ocurre en un vector especificado del bus de VME.	
LOCK	Instrucción de Programa	Fija la prioridad del cierre de la reacción del programa al valor dado.	
MCS	Instrucción de Programa	Invoca un comando de control del monitor	
NEXT	Instrucción de Programa	Rompe una estructura FOR, DO, o WHILE y comienza la iteración siguiente de la estructura del control.	
PAUSE	Instrucción de Programa	Para la ejecución de programa y no permite que el programa sea reasumido.	
PRIORITY	Función de valor Real	Devuelve la prioridad actual del cierre de la reacción hacia el programa.	
REACT / REACTI	Instrucción de Programa	Inicia el control continuo de una señal numérica especifica y acciona automáticamente una llamada de subprograma si las transiciones de señal son correctas.	REACT signal_number, program, priority Ej: 40 REACT 1001, alarm, 10
REACTE	Instrucción de Programa	Inicia la supervisión de los errores que ocurren durante la ejecución de la tarea del programa actual.	
REALASE	Instrucción de Programa	Permite la tarea siguiente disponible del programa para funcionar.	
RETRY	Instrucción de Programa	Controla si la PROGRAM START causa un resumens de programa .	
RETURN	Instrucción de Programa	Termina la ejecución de la ejecución actual del subprograma y de la ultima ejecución del programa en el siguiente paso de la instruccion CALL o CALLS que causa la subrutina invocada.	
RETURNE	Instrucción de Programa	Termina la ejecución de la ejecución actual del subprograma y de la ultima ejecución del programa en el siguiente paso de la instrucción que causa la subrutina invocada.	
RUNSIG	Instrucción de Programa	Enciende o apaga la señal digital mientras la ejecución de la tarea invocada continua.	
SET.EVENT	Instrucción de Programa	Fija un acontecimiento asociado a la tarea especificada.	
STOP	Instrucción de Programa	Termina la ejecución del ciclo del programa	
WAIT	Instrucción de Programa	Pone en el programa un lazo de espera hasta que la condición es TRUE.	Ej: WAIT SIG(1032, 1028)
WAIT.EVENT	Instrucción de Programa	Suspende la ejecución de un programa hasta que ha ocurrido un acontecimiento especificado, o hasta que ha transcurrido una cantidad de tiempo especificada.	Ej: WAIT SIG(1032, 1028)
WHILE	Interruptor de sistema	Inicia un proceso con la estructura WHILE si la condición es TRUE o salta la estructura del WHILE si la condición es inicialmente FALSA.	Ej: WHILE expression DO code block END

Tabla de funciones de tipo string.

Palabra clave	Función
ASC	Devuelve un solo valor de carácter dentro de una secuencia.
$CHR	Devuelve una secuencia de un carácter que tiene un valor dado.
DBLB	Devuelve el valor de ocho octetos de una secuencia interpretada como número de coma flotante de precisión doble de IEEE.
$DBLB	Devuelve una secuencia de 8-byte que contiene la representación binaria de un valor verdadero en el formato de coma flotante con la precisión doble de IEEE.
$DECODE	Extrae la parte de una secuencia según lo delimitado por los caracteres de rotura dados.
$ENCODE	Devuelve una secuencia creada por especificaciones de salida. La secuencia producida es similar a la salida de una instrucción de TYPE.
FLTB	Devuelve el valor de cuatro octetos de una secuencia interpretada como número de coma flotante de IEEE.
$FLTB	Devuelve una secuencia de 4-byte que contiene la representación binaria de un valor verdadero en el formato floating-point de IEEE en la forma de coma flotante.
$INTB	Devuelve una secuencia de 2-byte que contiene la representación binaria de un número entero de16-bit.
LEN	Devuelve el número de caracteres de la secuencia dada.
LNGB	Devuelve el valor de cuatro octetos de una secuencia interpretada como número entero binario de 32-bit enteros.
$LNGB	Devuelve una secuencia de 4-byte que contiene la representación binaria de un número entero 32-bit.
$MID	Devuelve una subsecuencia de la secuencia especificada.
PACK	Substituye una subsecuencia dentro de una matriz de (128-character) o dentro de la variable (nonarray) de la secuencia de string variables.
POS	Retorna la posición del carácter que comienza en una subsecuencia en una secuencia.
$TRANSB	Retorna una secuencia de 48-byte que contiene la representación binaria de un valor de la transformación.
$TRUNCATE	Devuelve todos los caracteres en la secuencia de la entrada hasta un ASCII NUL (o el extremo de la secuencia) encontrada.
$UNPACK	Retorna una subsecuencia de una matriz de 128 variables de caracteres.
VAL	Devuelve el valor real representado por los caracteres en la secuencia de la entrada.

Tabla de funciones lógicas:

Palabra clave	Función
FALSE	Retorna el valor usado por V+ para representar un resultado lógico falso.
ON	Retorna el valor usado por V+ para representar un resultado lógico falso.
OFF	Retorna el valor usado por V+ para representar un resultado lógico verdadero.
TRUE	Retorna el valor usado por V+ para representar un resultado lógico verdadero.

Tabla de funciones para valores numéricos

Palabra Clave	Función
ABS	Retorna un valor absoluto
ATAN2	Devuelve el tamaño del ángulo (grados) que tiene la tangente trigonometrica igual a value_1/value_2.
BCD	Convierte un valor real a formato de Código Decimal binario (BCD).
COS	Retorna el cosinos trigonométrico del ángulo dado.
DCB	Convierte digitos BCD a un número entero equivalente.
FRACT	Retorna la parte fraccionada de un argumento.
INT	Retorna la parte entera de un número.
INTB	Retorna el valor de dos bytes de un string interpretado
MAX	Devuelve el valor máximo contenido en la lista de valores.
MIN	Devuelve el valor mínimo contenido en la lista de valores.
OUTSIDE	Comprueba un valor para ver si es exterior al rango especificado.
PI	Devuelve el valor de la constante matemática pi (3,141593).
RANDOM	Devuelve un número pseudoaleatorio.
SIGN	Devuelve el valor 1 con la señal del parámetro del valor.
SIN	Devuelve el seno trigonométrico de un ángulo dado.
SQR	Devuelve el cuadrado del parámetro.
SQRT	Devuelve la raíz cuadrada del parámetro.

Tabla de funciones de sistemas de control:

Palabra clave	Función
DEFINED	Determina si se ha definido una variable.
ERROR	Devuelve el número del error de un error reciente que causó una parada mientras se ejecutaba el programa o causó una reacción de REACTE.
$ERROR	Retorna el mensaje de error asociado con el código de error.
FREE	Devuelve la cantidad de espacio de almacenaje libre inutilizado de la memoria.
GET.EVENT	Retorna acontecimientos que se fijan para la tarea especificada.
ID	Devuelve los que identifican la configuración del sistema actual.
$ID	Devuelve la fecha de creación del sistema y la información de edit/revision.
LAST	Devuelve el índice más alto usado para una matriz (dimensión).
PARAMETER	Devuelve el ajuste actual del parámetro nombrado del sistema.
PRIORITY	Devuelve la prioridad actual del cierre de la reacción para el programa.
SELECT	Devuelve el número de la unidad que es seleccionado actualmente por la tarea actual para el dispositivo nombrado.
STATUS	Retorna la información de estado para una aplicación de programa.
SWITCH	Devuelve una indicación del ajuste de un interruptor del sistema.
TAS	Devuelve el valor actual de una variable de valor real y le asigna un nuevo valor. Las dos acciones se hacen indivisiblemente, así que, ninguna otra tarea del programa puede modificar la variable en el mismo tiempo.
TASK	Retorna información sobre una tarea de la ejecución de programa.
TIME	Devuelve un valor del número entero que representa la fecha o la hora especificada en el parámetro dado, de la secuencia.
$TIME	Devuelve un valor de la secuencia que contiene la fecha actual del sistema y hora o la fecha y la hora especificadas.
TIMER	Devuelve el valor del tiempo actual (en segundos) del contador de tiempo especificado del sistema.
TPS	Devuelve el número de las señales del reloj del sistema que ocurren por segundo (señales por segundo).

Señales especiales

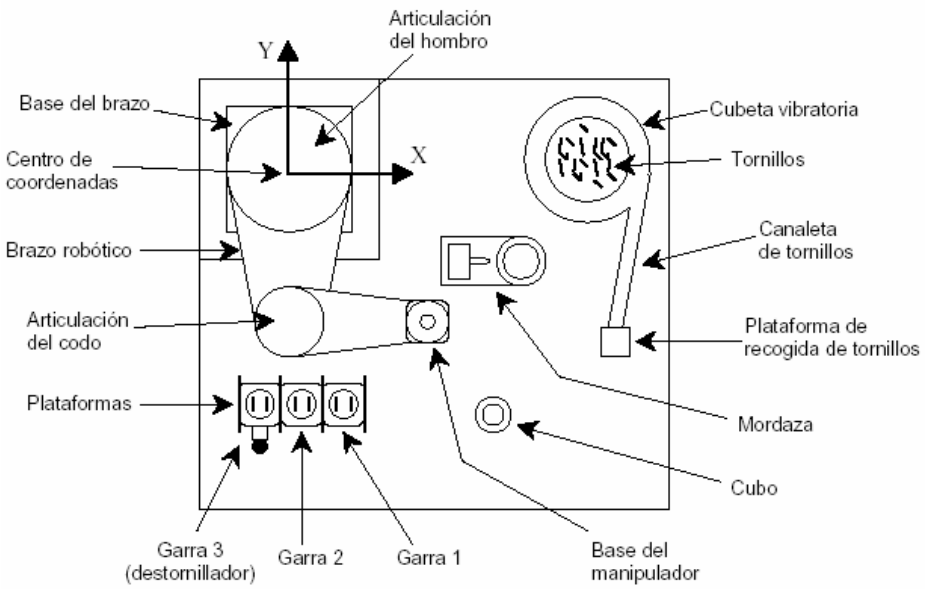
Existen algunas señales que permiten activar los elementos relacionados con los tornillos. Se manejan con la instrucción *SIGNAL(<VALOR>)* desde dentro de un programa. Dependiendo del valor dado, activa o desactiva ciertos elementos del entorno de trabajo. Unos ejemplos podrían ser:

– *Signal **(1)***: Activa la cubeta para la subida de los tornillos.
– *Signal **(-1)***: Desactiva la cubeta para la subida de los tornillos.
– *Signal **(2)***: Activa la plataforma de recogida de los tornillos.
– *Signal **(-2)***: Desactiva la plataforma de recogida de los tornillos.
– *Signal **(3)***: Activa el cierre de la mordaza.
– *Signal **(-3)***: Desactiva el cierre de la mordaza.
– *Signal **(4)***: Sube la plataforma para la inserción de tornillos en el destornillador.
– *Signal **(-4)***: Baja la plataforma para la inserción de tornillos en el destornillador.

Apagado del sistema

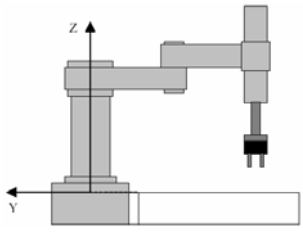
Una vez terminada la sesión, se debe proceder a apagar el controlador en el orden inverso al encendido, es decir, primero el interruptor verde, *SYSTEM POWER* debe ponerse en la posición **0** y después se debe situar la palanca de encendido en la posición **OFF**.

Una posible ejemplo de aplicación:



Posiciones de los elementos

Posición de seguridad: SET SAFE = TRANS (560, -560, 855, 0, 180, -124)
Posición justo encima de la garra 1: SET MM1 = TRANS (123.789, -488.511, 830, 0, 180, - 61.87)
Posición de la garra 1: SET GARRA1 = TRANS (123.789, -488.511, 755, 0, 180, -61.87)
Diferencia de altura entre mm1 y garra1: -75
Posición de salida de la garra 1: SET SALIR1 = TRANS (123.789, -629.947, 755, 0, 180, -61.87)
Posición justo encima de la garra 2: SET MM2 = TRANS (-16.867, -488.511, 830, 0, 180, -61.87)
Posición de la garra 2: SET GARRA2 = TRANS (-16.867, -488.511, 755, 0, 180, -61.87)
Diferencia de altura entre mm2 y garra2: -75
Posición de salida de la garra 2: SET SALIR2 = TRANS (-16.867, -629.947, 755, 0, 180, -61.87)
Posición del cubo en la practica 2: SET CUBO = TRANS (314.880, -685.921, 851.294, 0, 180, -62.071)
Altura desde esta posición a la posición adecuada para coger el cubo: -140
Posición final del cubo en la práctica 2: SET POSFINAL = TRANS (581.330, -471.566, 716.413, 0, 180, -61.848)
Altura desde esta posición a la posición adecuada para dejar el cubo: -5
Posición del destornillador: SET MM3 = TRANS (-156.304, -488.511, 830, 0, 180, -61.87)
Posición del destornillador: SET GARRA3 = TRANS (-156.304, -488.511, 755, 0, 180, -61.87)
Diferencia de altura entre mm3 y garra3: -75
Posición de salida del destornillador: SET SALIR3 = TRANS (-156.304, -629.947, 755, 0, 180, -61.87)
Posición sobre el tornillo para sercog ido por el destornillador: SET TORNILLO = TRANS (715.240, -286.026, 842.133, 0, 180, -147.883)
Altura desde esta posición a la posición adecuada para recoger el tornillo: -10
Altura a la que tiene que subir el destornillador una vez cogido el tornillo: 36
Posición del primer agujero del cubilete: SET AGUJERO1 = TRANS (581.164, -248.053, 867.959, 0, 180, -147.872)
Altura desde esta posición a la necesaria para atornillar: -65.408
Posición del segundo agujero del cubilete: SET AGUJERO2 = TRANS (527.615, -289.625, 867.959, 0, 180, -56.016).



Procedimiento para coger garras

1. Se debe situar la base del manipulador justo encima de la garra que se va a coger.

2. Se baja la base del manipulador a la posición en la que los enganches se introduzcan dentro de los agujeros de la garra.

3. Se cierran los enganches de la base del manipulador (*CLOSEI*).

4. Se saca la garra de la plataforma. Para ello, se debe salir en horizontal con un movimiento rectilíneo variando solamente la coordenada Y.

5. Ya se puede operar libremente con la garra cogida.

Algoritmo:

```
PROGRAM cogergarra(garra, salir)
LEFTY ; comportamiento de brazo iquierdo
APPRO garra, 100 ; mueve la base a 100mm encima de la garra
DELAY(0.5) ; retardo de 0.5 segundos
OPENI ; abre los enganches de la base
DELAY(0.5)
MOVES garra ; mueve la base hasta entrar en la garra
DELAY(0.5)
CLOSEI ; cierra los enganches de la garra
DELAY(0.5)
MOVES salir ; saca la garra de su plataforma
DELAY(0.5)
DEPARTS 100 ; sube la garra 100mm
```

Procedimiento para dejar garras

1. Se debe situar la base del manipulador justo encima de la posición de salida de la garra que se va a dejar.

2. Se baja la base del manipulador a la posición de salida de la garra.

3. Se introduce la garra en la plataforma en horizontal con un movimiento rectilíneo variando solamente la coordenada Y.

4. Se abren los enganches de la base del manipulador (*OPENI*) para liberar la garra.

5. Se levanta la base del manipulador.

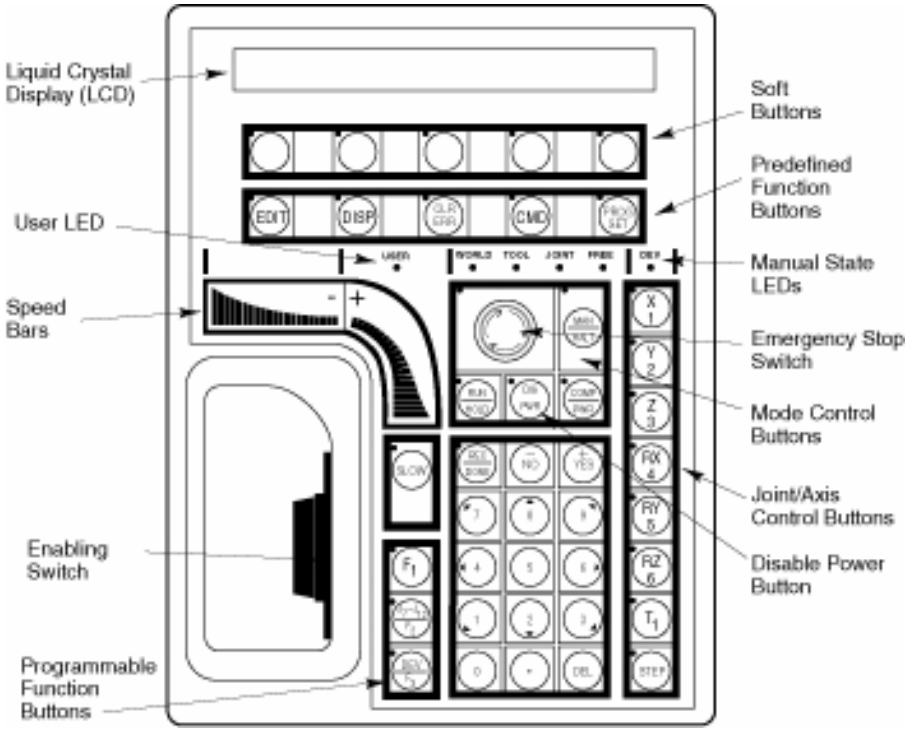
Algoritmo:

```
PROGRAM dejargarra(garra, salir)
LEFTY ; comportamiento de brazo iquierdo
APPRO salir, 100 ; mueve la garra a 100mm encima de salir
DELAY(0.5)
MOVES salir ; mueve la garra a la posición salir
DELAY(0.5)
MOVES garra ; introduce la garra en la plataforma
DELAY(0.5)
OPENI ; abre los enganches de la base
DELAY(0.5)
DEPARTS 100 ; levanta la base a 100mm
```

2.6. Botones de la paleta (Botonera).

La Paleta de control (Botonera) permite controlar el robot.

Botones de funciones

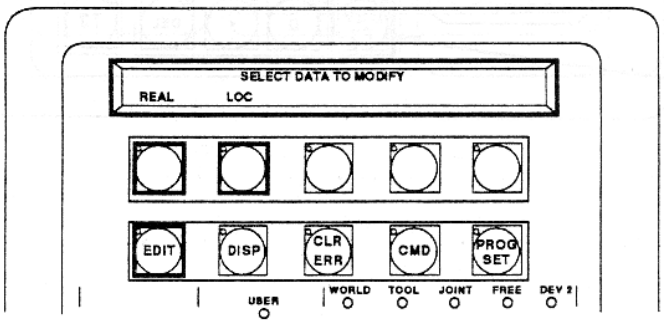


Elementos básicos para el control del robot industrial (Botonera, Armario de control o controladora y panel de control):

Botones de funciones Predefinidas

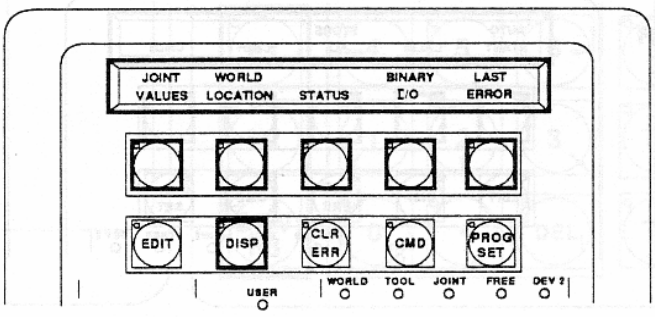
Botón EDIT

El botón EDIT permite editar las variables de posición (LOCATION) y reales (REAL) eventualmente, permite las modificaciones de estas.



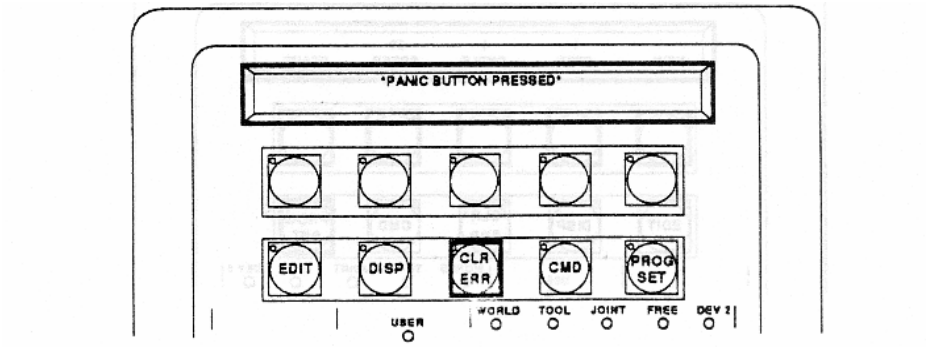
Boton DISP

El botón DISP permite fijar las coordenadas de articulación del robot.



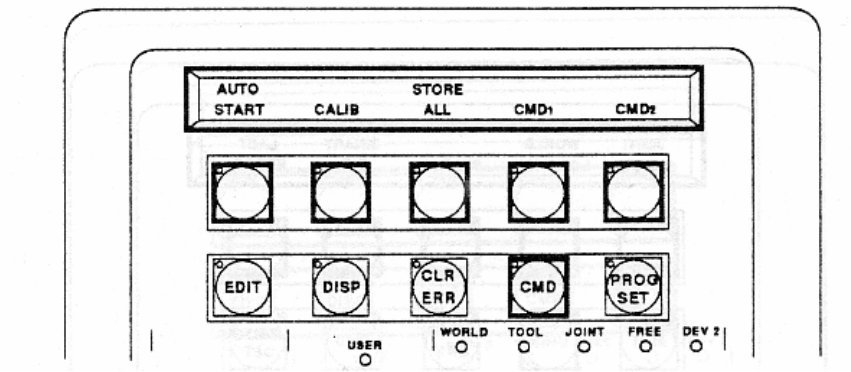
Botón CLR ERR

El botón CLR ERR permite seguir la ejecución de un programa, después de haberse provocado un error. Al apretar el botón volvemos a al momento antes de producirse el error.



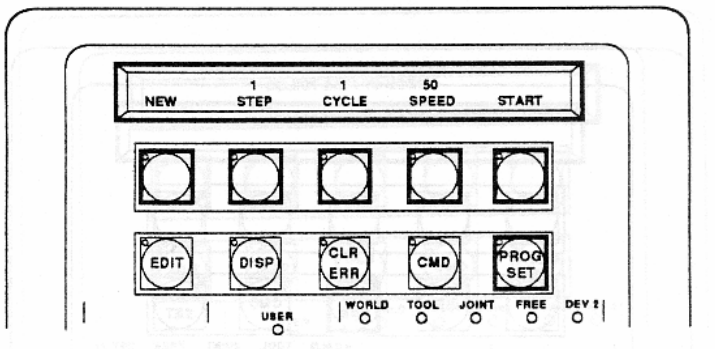
Botón CMD

El botón CMD permite acivar los comandos de los programas en la pantalla CMD1 o CMD2.



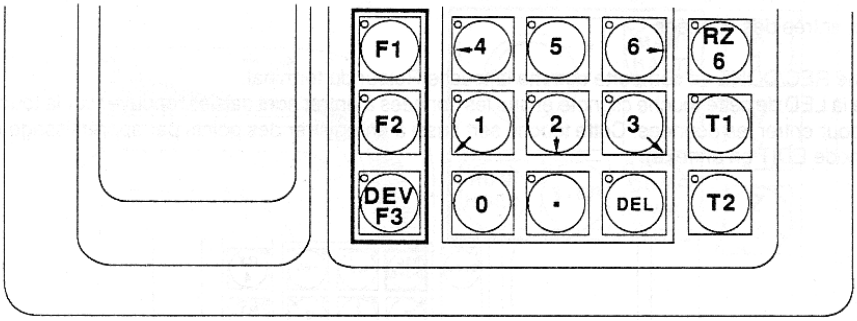
Botón PROG SET

El botón PROG SET permite elegir un nuevo programa a ejecutar que resida en memoria.



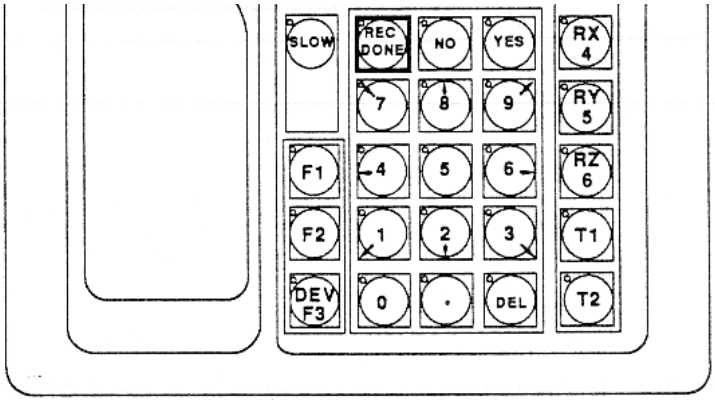
Botones de función programables : F1, F2, DEV/F3

Los botones de funciones programables libres (F1, F2 et DEV/F3) pueden ser utilizados por qualquier aplicación de programa, para responder a diferentes necesidades especificas.



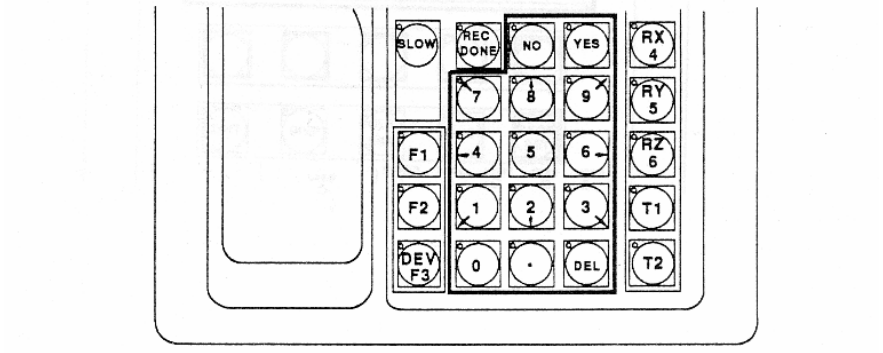
Boton REC / DONE

El botón REC/DONE permite memorizar los distintos puntos en el modo TEACH .



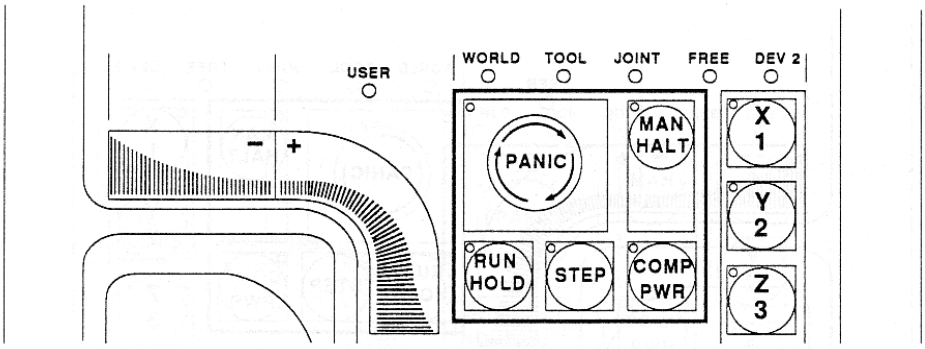
Botones de entrada de Datos

Estos botones de entrada de Datos, permiten responder las solicitudes de ficheros auxiliares mostrandolos por el LCD.



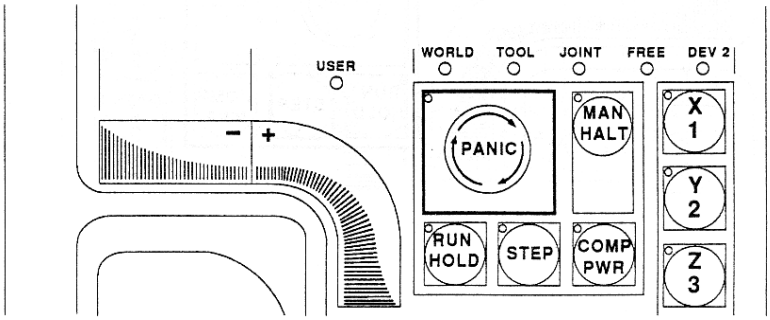
Botones de seleccion de modo

Estos botenes son de color rojo, para poderse identificar bien. Sirven para canviar el modo de operar con el robot.



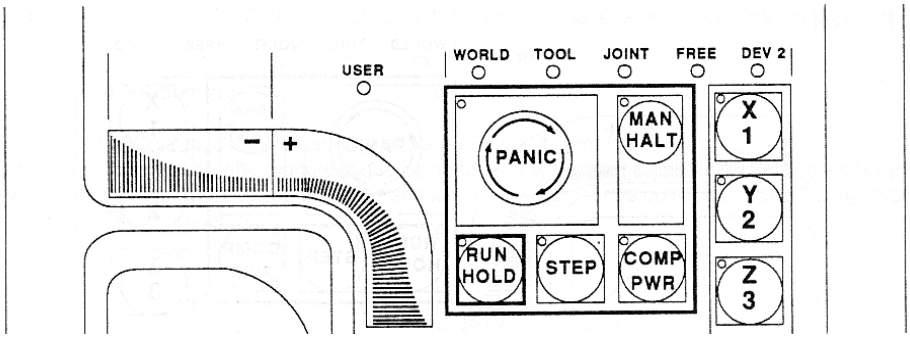
Botón PANIC

El botón PANIC para el programa en ejecución y desactiva la alimentación de los brazos del robot y activa los frenos



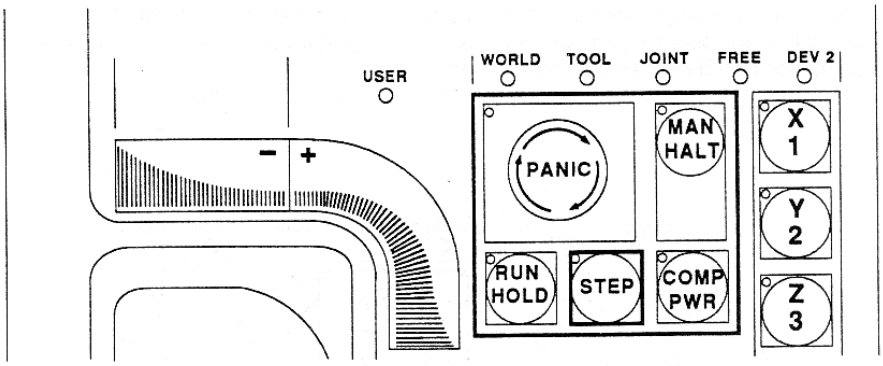
Botón RUN/HOLD

El botón RUN / HOLD pone el programa en Marcha, saliendo el panel AV



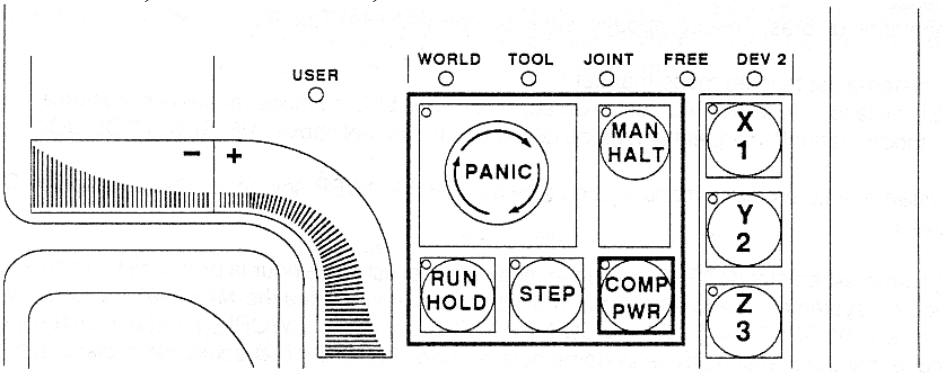
Botón STEP

El botón STEP pone el programa en Marcha, saliendo el panel AV



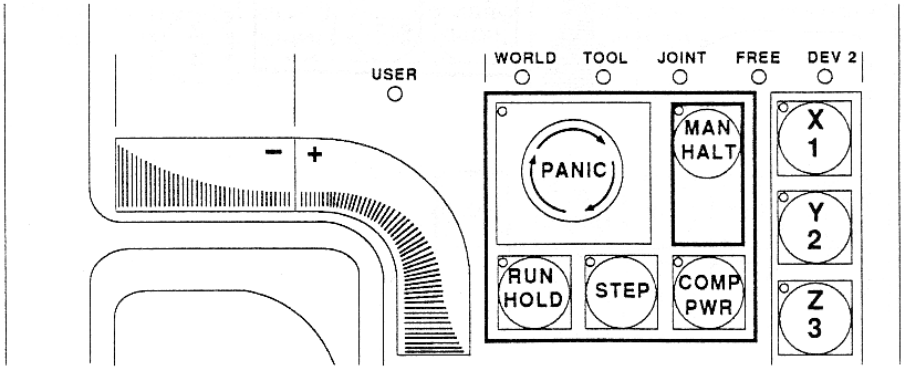
Botón COMP / PWR

El botón COMP / PWR permite reactivar la alimentación del brazo cuando el brazo del robot esta parado por una condición de error, o esta desactivado, etc…



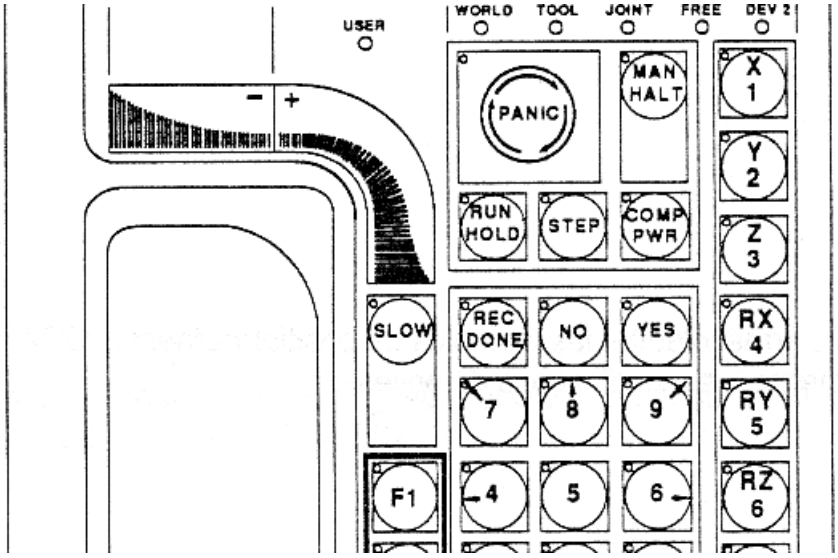
Botón MAN / HALT

Se utiliza para la ejecución de un programa
La botón MAN / HALT puede igualmente parar una ejecución del programa al final del desplazamiento de carrera del robot.



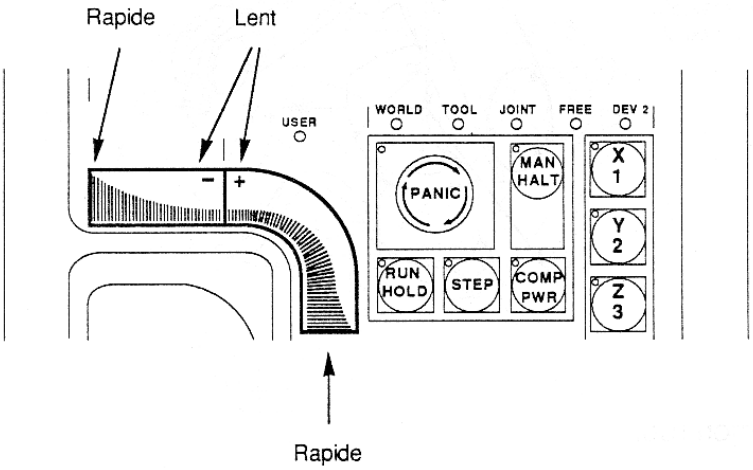
Botones de comandos Manuales

Los botones en modo manual, permiten elegir los ejes de desplazamiento del robot : X/1, Y/2, Z/3, RX/4, RY/5, RZ/6.



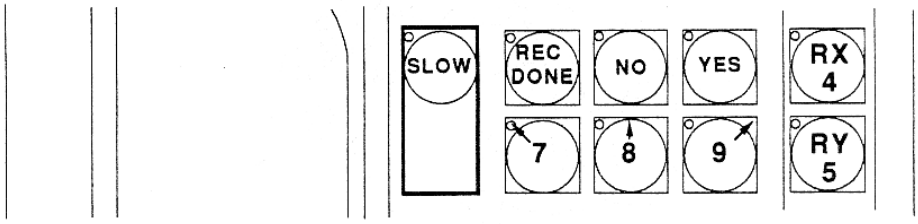
Barras de velocidad

Les barras de velocidad, permiten elegir la velocidad en los desplazamientos manuales. Con estas barras se puede elegir una velocidad más rápida o lenta según pulse (+/-).



Slow (lento)

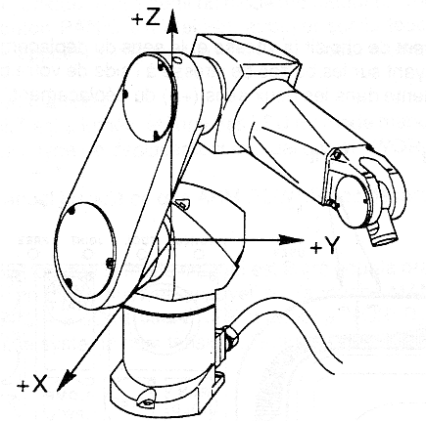
El botón SLOW permite elegir dos tipos de desplazamiento, el desplazamiento normal o lento.



Estados del Robot

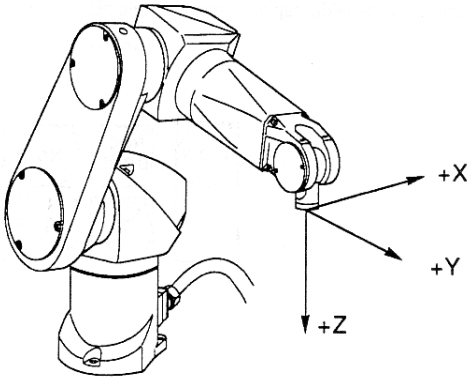
Botón WORLD

El botón WORLD permite desplazar libremente los ejes, paralelamente a las coordenadas de la base.



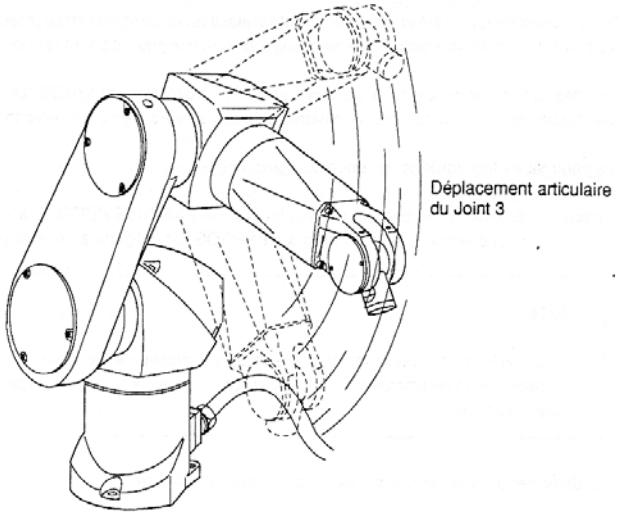
Botón TOOL

El botón TOOL permite desplazar libremente los ejes paralelamente a las coordenadas utiles.



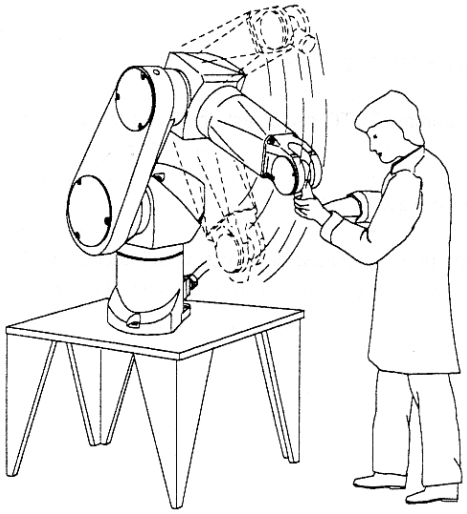
Botón JOINT

El botón JOINT permite desplazar angularmente los distintos ejes.

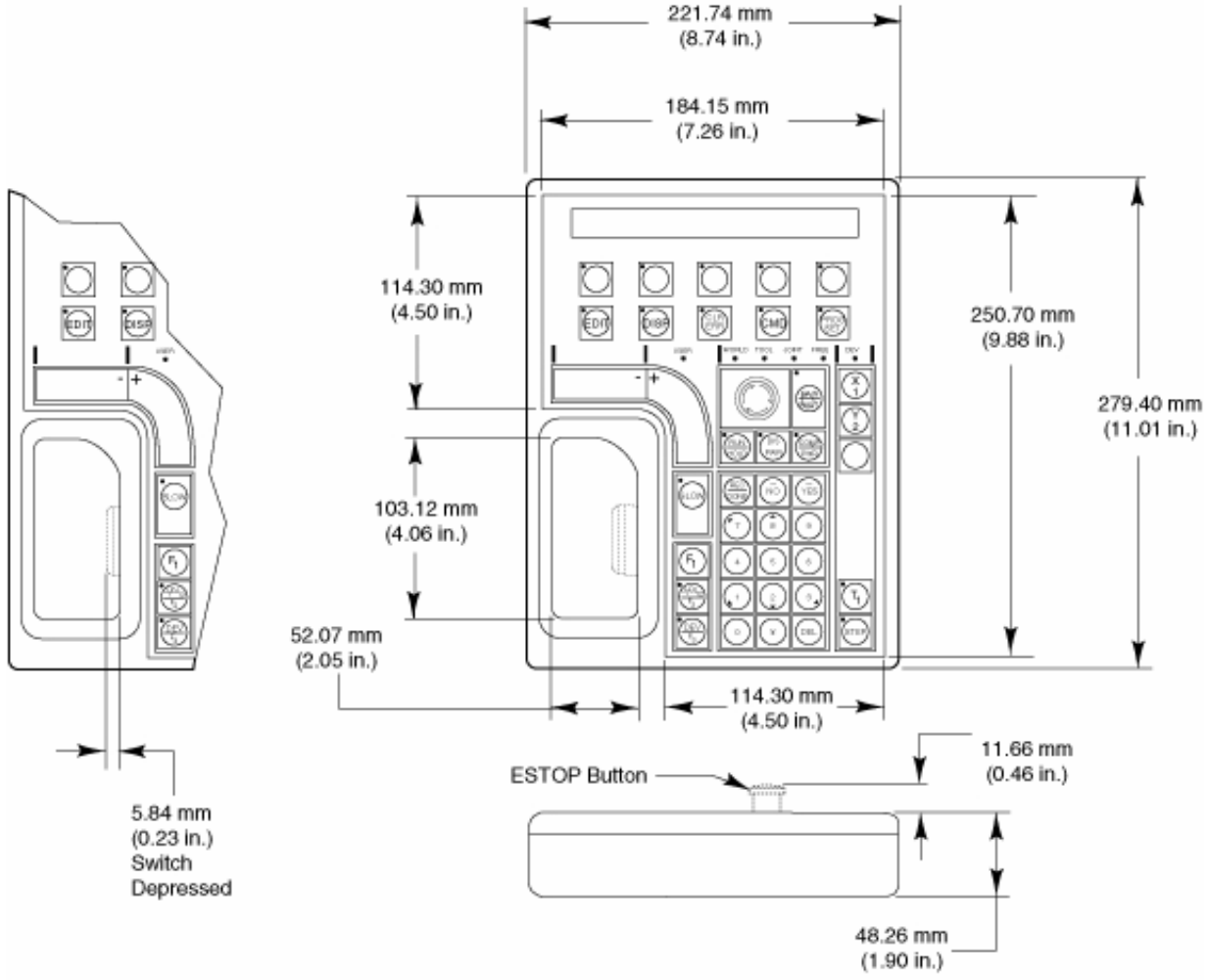


Botón FREE

El botón FREE permite liberar cada eje del robot.



Dimensiones



Ejemplo de botonera (terminal de enseñanza, paleta de control, telemando)

En la actualidad, y debido a la gran competencia entre distintas marcas de fabricación de robots industriales, se intenta hacer más agradable el entorno de la botonera. Así mismo, aquí se puede observar como la pantalla de LCD de dos líneas se ha transformado en una pantalla mucho mayor para hacer la visualización del programa más sencilla y menos engorrosa.

3. APLICACIONES

3. APLICACIONES

Para implantar un robot industrial, en un determinado proceso, exige un detallado estudio previo del proceso en cuestión, examinando las ventajas e inconvenientes que conlleva la introducción del robot.

También habrá que considerar aspectos como el área de acción, velocidad de carga, capacidad de control, coste, etc.

3.1. Aplicaciones en Fundición.

La fundición de materiales por inyección fue el primer proceso robotizado en 1960. En este proceso el material usado, en estado liquido, es inyectado a presión en el molde. El molde está formado por dos mitades que se mantienen unidas durante la inyección del metal mediante la presión ejercida por dos cilindros. La pieza solidificada se extrae del molde y se enfría. El molde, una vez limpio de residuos de metal y adecuadamente lubricado, puede ser usado de nuevo.

Los robots, en estos procesos son utilizados para el transporte de las piezas a un lugar de enfriado y posteriormente a otro proceso (desbardado, corte, etc.).

Las cargas manejadas por los robots en estas tareas suelen ser medias o altas (del orden de decenas de kilogramos), no se necesita una gran precisión y su campo de acción ha de ser grande. Su estructura más frecuente es la polar y la articular, su sistema de control es por lo general sencillo.

3.2. Aplicaciones de Soldadura.

Las utilización de los robots para soldadura ha sido impulsada por la industria automovilística.

La tarea más robotizada dentro de la fabricación de automóviles es la soldadura de carrocerías. En este proceso, dos piezas metálicas se unen en un punto para la fusión conjunta de ambas partes, denominándose a este tipo de soldadura, por puntos.

Para realizar este tipo de soldaduras, se hace pasar una corriente eléctrica elevada y baja tensión a través de dos electrodos enfrentados entre los que se sitúan las piezas a unir. Los electrodos instalados en una pinza de soldadura, deben sujetar las piezas con una presión determinada (de lo que depende la precisión de la soldadura). Además deben de ser controlados los niveles de tensión e intensidad necesarios, así como el tiempo de aplicación. Todo ello exige el empleo de un sistema de control del proceso de soldadura.

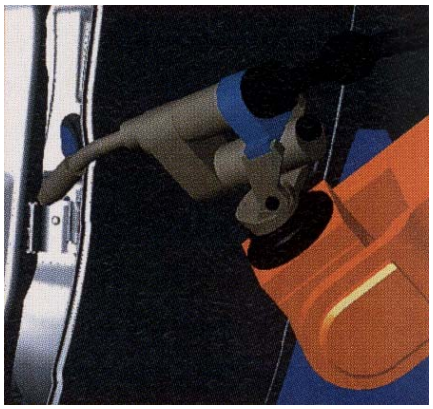
Los robots de soldadura por puntos precisan capacidad de cargas del orden de los 50-100 Kg. y estructura articular, con suficientes grados de libertad (5 o 6) como para posicionar y orientar la pinza de soldadura (o pieza según el caso) en lugares de difícil acceso.

Las tareas de soldadura por puntos admiten dos soluciones, dependiendo del tamaño, peso y manejabilidad de las piezas:

1) El robot transporta la pieza presentando esta a los electrodos que están fijos
2) El robot transporta la pinza de soldadura posicionando los electrodos en el punto exacto de la pieza en la que se desea realizar la soldadura.

Un ejemplo de soldadura:

En las grandes líneas de soldadura de carrocerías de automóviles, estas pasan secuencialmente por varios robots dispuestos formando un pasillo, de una manera coordinada y posicionan las piezas de soldadura realizando varios puntos consecutivamente.

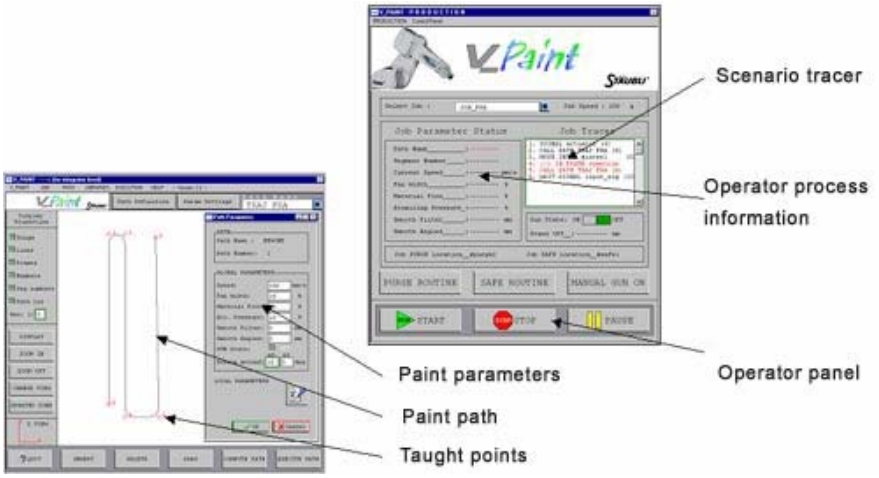


3.3. Aplicación de Pintura, Esmalte, Partículas de metal, etc.

La utilización de los robots para el recubrimiento de un cierto material (pintura, esmalte, partículas de metal, etc.) con fines decorativos o de protección, es una parte critica en muchos procesos de fabricación.

La tarea más significativa de aplicación de materiales es la pintura (sobre todo para los automóviles, electrodomésticos, muebles, etc.), ya que es una tarea desagradable, peligrosa (provoca una atmósfera toxica, un alto nivel de ruido y riesgo de incendio).

La tarea de pintura requiere cubrir una superficie con una mezcla de aire y material pulverizada mediante una pistola, hasta conseguir una perfecta homogeneidad en el reparto de la pintura. Para ello, se realiza un control de la viscosidad, de la distancia entre piezas y la pistola, velocidad de movimiento de esta, numero de pasadas etc.



Los robots de pintura suelen ser robots articulares, ligeros, con 6 o más grados de libertad que les permiten proyectar pintura en todos los huecos de la pieza. Cuentan con protecciones especiales para defenderse de las partículas en suspensión dentro de la cabina de pintura y sus posibles consecuencias (explosiones, incendio, deterioro mecánico). También tienen una accionamiento hidráulico por el riesgo de incendio.

Lo más importante de estos robots es la programación: Cuentan con un control de trayectoria continua, para especificar trayectoria, punto final e inicial. El método normal de programación es el de aprendizaje con un muestreo continuo de la trayectoria.

El operario realiza una vez el proceso de pintura con el propio robot. La unidad de programación registra continuamente gran cantidad de puntos y luego los repite.

Ejemplos de Robots de la casa Stäubli para la Pintura.

MODELO	CAPACIDAD CARGA MÁXIMA	ALCANCE MÁXIMO	REPETIBILIDAD
 RXPaint60	4,5 kg	665 mm	± 0.02 mm
 RXPaint60 L	2,5 kg	865 mm	± 0.033 mm
 RXPaint90	12 kg	985 mm	± 0.02 mm
 RXPaint90 L	9 kg	1185 mm	± 0.025 mm
 RXPaint130	25 kg	1360 mm	± 0.03 mm
 RXPaint130 L	12 kg	1600 mm	± 0.035 mm
 RXPaint130 XL	8 kg	2185 mm	± 0.05 mm

3.4. Alimentación de maquinas.

La utilización de robots para alimentar máquinas aparece por la peligrosidad y monotonía de las operaciones de carga y descarga de maquinas como prensas, estampadoras, hornos, etc.

Los robots usados en estas tareas son de baja complejidad, precisión media, numero reducido de grados de libertad y un control sencillo, basado, en ocasiones, con manipuladores secuenciales y con un campo de acción grande. Se pueden necesitar robots con capacidad de carga de pocos kilogramos, hasta algunos cientos (existen robots capaces de manipular hasta tonelada y media).

Las estructuras mas frecuentemente utilizadas son la cilíndrica, esférica y articular. También la cartesiana puede aportar solución.

En este apartado aparecerían los robots aplicados en células flexibles de mecanizado. Estos emplean centros de mecanizado o varias maquinas de control numérico para conseguir complejos y distintos mecanizados sobre una pieza para dar a esta la forma programada. La capacidad de programación de estas maquinas permite una producción flexible de piezas.

Estas maquinas emplean diferentes herramientas que se acoplan a un cabezal común de manera automática cuando el proceso de mecanizado lo precisa. Las herramientas a usar en el proceso concreto son almacenadas en tambores automáticos que permiten un rápido intercambio de la herramienta.

3.5. Corte.

El corte de materiales mediante el robot es una aplicación reciente, gracias a la capacidad de reprogramación del robot y su integración en un sistema que hace que sea el elemento ideal para transportar la herramienta de corte sobre la pieza, realizando con precisión un programa de corte desde un sistema de diseño asistido por computador (CAD).

Los métodos de corte no mecánico mas empleados son oxicorte, plasma, láser y chorro de agua, dependiendo de la naturaleza del material a cortar. En todos ellos el robot transporta la boquilla por la que se emite el material de corte, proyectando este sobre la pieza al tiempo que sigue una trayectoria determinada.

De todos los métodos de corte, el que cuenta con mejores ventajas seria el corte por chorro de agua por los motivos siguientes:

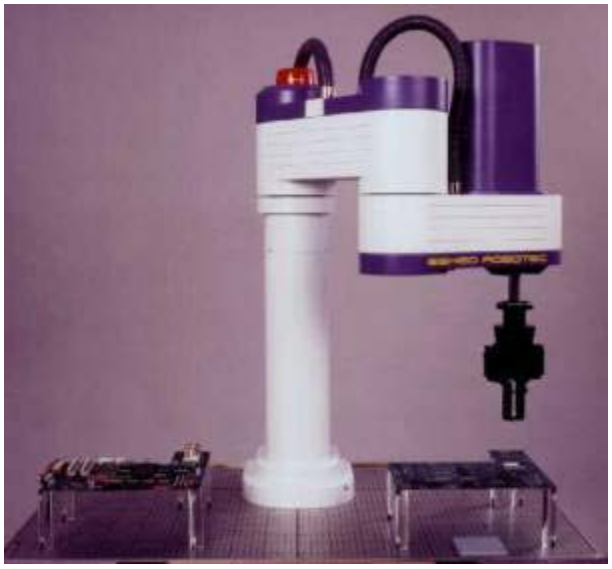
-No provoca aumento de temperatura en el material.
-No es contaminante.
-No provoca cambios de color.
-No altera las propiedades de los materiales.
-Coste de mantenimiento bajo.

Los robots empleados para corte, precisan control de trayectoria continua y elevada precisión. Su campo de acción varia con el tamaño de las piezas a cortar. Puede ser de una envergadura media de 1 a 3 metros de radio, por esto, con mucha frecuencia se dispone al robot suspendido boca abajo sobre la pieza.

3.6. Montaje / Ensamblaje.

Las operaciones de montaje, por la gran precisión y habilidad que normalmente exigen, presentan grandes dificultades para su automatización flexible.

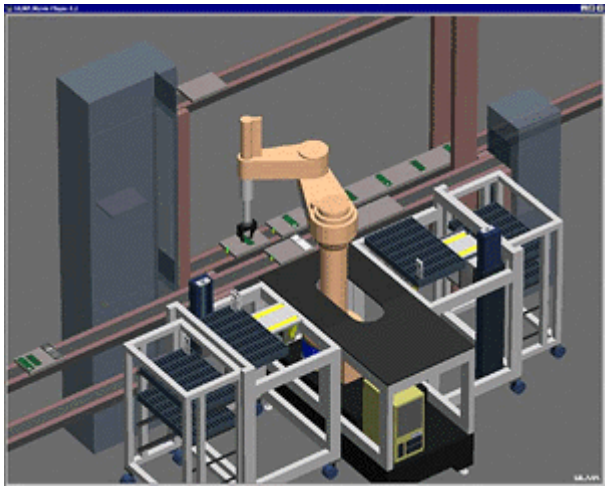
Muchos procesos de ensamblado se han automatizado empleando maquinas especiales que funcionan con gran precisión y rapidez. Sin embargo, el mercado actual precisa de sistemas muy flexibles, que permitan introducir frecuentes modificaciones en los productos con unos costes mínimos. Por este motivo el robot industrial se ha convertido en muchos casos en la solución ideal para la automatización del ensamblaje.



En particular, el robot resuelve correctamente muchas aplicaciones de ensamblado de piezas pequeñas en conjuntos mecánicos o eléctricos. Para ello el robot precisa una serie de elementos auxiliares cuyo coste es similar o superior al del propio robot. Entre estos cabe destacar a los alimentadores (tambores vibradores, por ejemplo), posicionadores y los posibles sensores que usa el robot para ayudarse en su tarea (esfuerzos, visión, tacto, etc.). Estos sensores son indispensables en muchos casos debido a las estrechas tolerancias con que se trabaja en el ensamblaje y a los inevitables errores, aunque sean muy pequeños, en el posicionamiento de las piezas que entran a tomar parte de él.

Los robots empleados en el ensamblaje requieren, en cualquier caso, una gran precisión y repetibilidad, no siendo preciso que manejen grandes cargas.

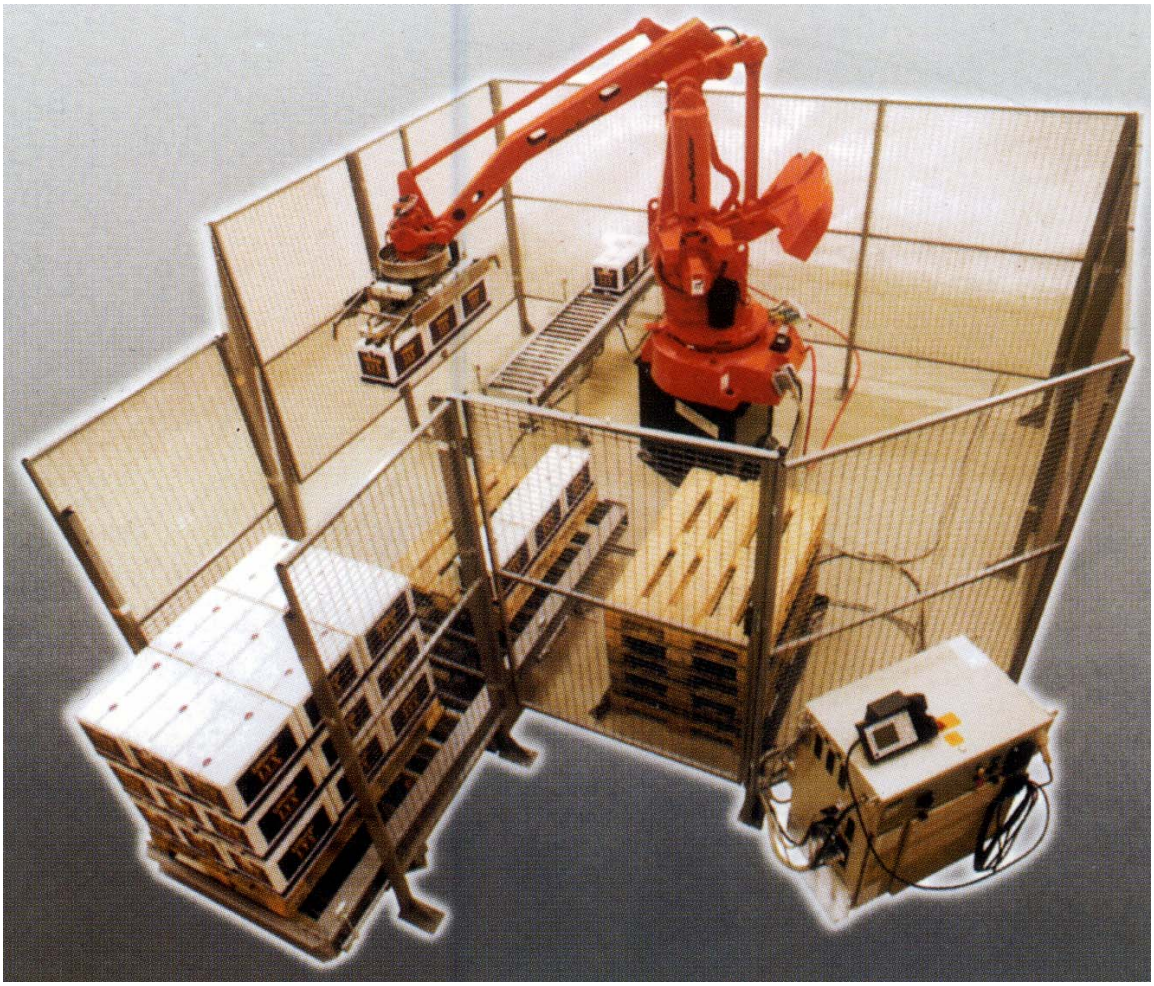
El tipo SCARA ha alcanzado gran popularidad en este tipo de tareas por su bajo coste y buenas características. Estas se consiguen por su adaptabilidad selectiva, presentando facilidad para desviarse, por una fuerza externa, en el plano horizontal y una gran rigidez para hacerlo en el eje vertical. También se usan con frecuencia robots cartesianos por su elevada precisión y, en general, los robots articulares que pueden resolver muchas de estas aplicaciones con suficiente efectividad. La dificultad inherente de este tipo de tareas obliga, en casi todos los casos, a facilitarlas con un adecuado rediseño de las partes que componen el conjunto a ensamblar. De este modo, conjuntos cuyo ensamblaje automatizado seria inabordable con su diseño inicial, pueden ser montados de una manera competitiva mediante el empleo de robots.



3.7. Paletización.

La paletización es un proceso básicamente de manipulación, consistente en disponer de piezas sobre una plataforma o bandeja (palet). Las piezas en un palet ocupan normalmente posiciones predeterminadas, procurando asegurar la estabilidad, facilitar su manipulación y optimizar su extensión. Los palets son transportados por diferentes sistemas (cintas transportadoras, carretillas, etc.) llevando su carga de piezas, bien a lo largo del proceso de fabricación, bien hasta el almacén o punto de expedición.

Dependiendo de la aplicación concreta, un palet puede transportar piezas idénticas (para almacenamiento por lotes por ejemplo), conjuntos de piezas diferentes, pero siempre los mismos subconjuntos procedentes de ensamblados) o cargas de piezas diferentes y de composición aleatoria (formación de pedidos en un almacén de distribución).



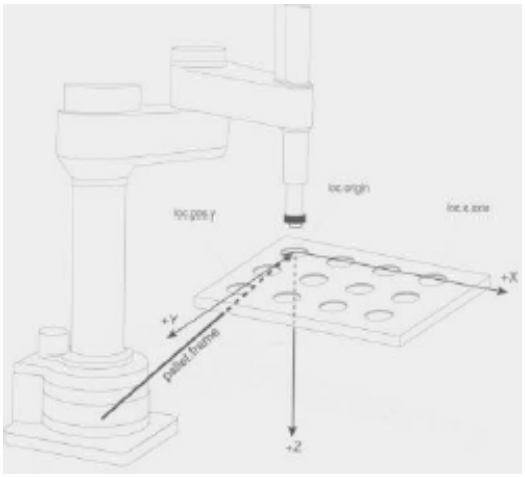
Existen diferentes tipos de maquinas especificas para realizar operaciones de paletizado. Estas frente al robot, presentan ventajas en cuanto a velocidad y coste, sin embargo, son rígido en cuanto a su funcionamiento, siendo incapaces de modificar su tarea de carga y descarga. Así mismo, actualmente la diferencia de coste es aproximadamente igual, y los robots realizan con ventaja aplicaciones de paletización en las que la forma, numero o características generales de los productos a manipular, cambian con relativa frecuencia. En estos casos, un programa de control adecuado permite resolver la operación de carga y descarga, optimizando los movimientos del robot, aprovechando la capacidad del palet o atendiendo a cualquier otro imperativo. Por otro lado, el robot industrial siempre se puede reutilizar para cualquier operación industrial debido a su flexibilidad.

Generalmente, las tareas de paletización implican el manejo de grandes cargas, de peso y dimensiones elevadas. Por este motivo, los robots empleados en este tipo de aplicaciones acostumbran a ser robots de gran tamaño, con una capacidad de carga de 10 a 150 kg aproximadamente. No obstante, se

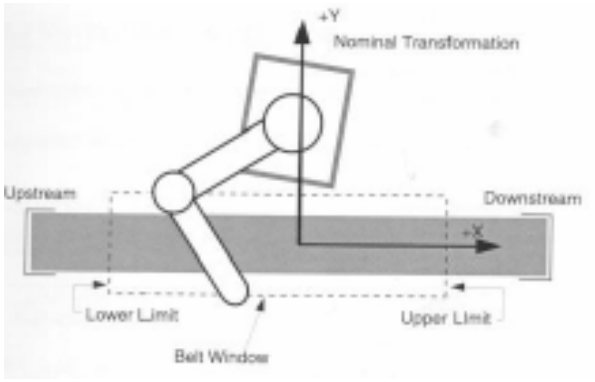
pueden encontrar aplicaciones de paletizacion de pequeñas piezas, en las que un robot con una capacidad de carga de 5Kg. es suficiente.

Las denominadas tareas de Pick and place, aunque en general con características diferentes al paletizado, guardan estrecha relación con este.

El lenguaje V+ posee diferentes facilidades para modelar determinadas tareas, como la definición de palets (pallet.frame) a partir de tres localizaciones especificadas, o para el seguimiento y manipulación de objetos en cintas transportadoras, que se mueven a velocidad constante (definiendo unas ventanas de trabajo sobre la propia cinta).



Definición de palets.



Seguimiento de cintas

```
.PROGRAM move.parts()
;DESCRIPCION:     Este programa coge cajas en la localización "pick"
; y las deposita en "place", incrementando la z de place, y así apilando
; las cajas en el palet.
      parts = 6                     ; nº de cajas a apilar
      height1 = 300                 ; altura de "approach/depart" en "pick"
      height2 = 500                 ; altura de "approach/depart" en "place"
      parameter HAND.TIME = 0.16    ; movimiento del brazo lento
      OPEN                          ; apertura de pinza
      RIGHTY                        ; seleccionamos configuración derecha
      MOVE start                    ; mover a la localización segura de inicio
      FOR i = 1 TO parts            ; iniciar el apilado de cajas
            APPRO pick, height1     ; ir a "pick-up"
            MOVES pick              ; mover hacia la caja
            CLOSEI                  ; cerrar la pinza
            DEPARTS height1         ; volver a la posición anterior
            APPRO place, height2    ; ir a "put-down"
            MOVES place             ; mover a la localización de destino
            OPENI                   ; abrir la pinza
            DEPARTS height2         ; volver a la posición anterior
            SHIFT heicht2 BY 0.00, 0.00, 300
      END
      TYPE "Fin de tarea. ", /IO, parts, " cajas apliadas."
      RETURN                  ; fin del programa
.END
```

3.8. Pick and place.

La misión de un robot trabajando en un proceso de pick and place consiste en recoger piezas de un lugar y depositarlas en otro. La complejidad de este proceso puede ser muy variable, desde el caso más sencillo en el que el robot recoge y deja las piezas en una posición prefijada, hasta aquellas aplicaciones en las que el robot precise de sensores externos, como visión artificial o tacto, para determinar la posición de recogida y colocación de las piezas.

Al contrario que en las operaciones de paletizado, las tareas de picking suelen realizarse con piezas pequeñas (peso inferior a 5Kg) necesitándose velocidad y precisión.

Un ejemplo típico de aplicación de robot al paletizado seria la formación de palets de cajas de productos alimenticios procedentes de una línea de empaquetado. En estos casos, cajas de diferentes productos llegan aleatorianmente al campo de acción del robot. Ahí son identificadas bien por una célula de carga, por alguna de sus dimensiones, o por un código de barras. Conocida la identidad de la caja, el robot procede a recogerla y a colocarla en uno de los diferentes palets que, de manera simultanea, se están formando.

El propio robot gestiona las líneas de alimentación de las cajas y de palets, a la vez que toma las decisiones necesarias para situar la caja en el palet con la posición y orientación adecuadas de una manera flexible.

El robot podrá ir equipado con una serie de ventosas de vació y su capacidad de carga estaría entorno a los 50 kg aproximadamente.

Como curiosidad, el robot industrial KUKA, modelo KR 305/1 de la ilustración, la capacidad de carga asciende a 350 kg.

Ejemplo de aplicación Pick and Place

```
.PROGRAM sub.menu()

; DESCRIPCIÓN: Este programa ofrece al operario un menú con las
; opciones que puede seleccionar, desde su terminal de usuario
; (botonera/ PC).
; Después de introducir una entrada desde el teclado, el programa
; ejecutará la operación seleccionada.
; El menú incluye la ejecución del programa de pick and place, mostrar
; los puntos, y volver al menu principal.
; SUPONEMOS: Las funciones pick.place() y teach() definidas.

AUTO opcion, salir, $frase

salir = FALSE

      DO
            TYPE /C2, "MENU DE OPERACIÓN PICK AND PLACE"
            TYPE /C1, " 1 => Inicializar pick and place"
            TYPE /C1, " 2 => Mostrar localización puntos"
            TYPE /C1, " 3 => Volver al menu principal", /C1

            PROMPT "Selecciona una opcion y presiona INTRO: ", $frase

            opcion = VAL($frase)    ;Convierte un string en un numero

            CASE opcion OF          ;entramos en el menu...
                  VALUE 1:          ;...seleccion 1
                  TYPE /C2, "Iniciando operacion..."
                  CALL move.parts()
                  VALUE 2:          ;...seleccion 2
                  CALL teach()
                  VALUE 3:          ;...seleccion 3
                  salir = TRUE
                  ANY         ;...si se presiona cualquier otra tecla
                  TYPE /B, /C1, "** ENTRADA NO VALIDA E R R O R **"
            END                     ;Fin del CASE
      UNTIL salir                   ;Fin del DO
.END
```

```
.PROGRAM move.parts()
;DESCRIPCION:     Este programa coge piezas en la localización "pick"
; y las deposita en "place"
      parts = 100                   ; nº de piezas a procesar
      height1 = 25.4                ; altura de "approach/depart" en "pick"
      height2 = 50.8                ; altura de "approach/depart" en "place"
      parameter HAND.TIME = 0.16    ; configuración para brazo lento
      OPEN                          ; apertura de pinza
      RIGHTY                        ; seleccionamos configuración derecha
      MOVE start                    ; mover a la localización segura de inicio
      FOR i = 1 TO parts            ; procesar las piezas
            APPRO pick, height1     ; ir a "pick-up"
            MOVES pick              ; mover a la pieza
            CLOSEI                  ; cerrar la pinza
            DEPARTS height1         ; volver a la posición anterior
            APPRO place, height2    ; ir a "put-down"
            MOVES place             ; mover a la localización de destino
            OPENI                   ; abrir la pinza
            DEPARTS height2         ; volver a la posición anterior
      END
      TYPE "Fin de tarea. ", /IO, parts, " piezas procesadas."
      RETURN                        ; fin del programa
.END
```

Este programa permite al operario grabar las posiciones de pick, place y start, con la paleta de control manual. Las dos líneas del cristal líquido (LCD) de la paleta de control, son usadas en esta. El programa graba automáticamente las posiciones después de presionar una tecla.

```
.PROGRAM teach(pick, place, start)

; ABSTRACT: Este programa se usa para grabar las posiciones
;"pick", "place", y "start" del "move.parts" programa.
; PARÁMETROS DE ENTRADA: Ninguno
; PARÁMETROS DE SALIDA: pick, place, y empezar
; EFECTOS SECUNDARIOS: El robot está inactivo mientras la rutina está
; activada.

AUTO $borrar.display    ;variable

$borrar.display = $CHR(12)+$CHR(7)

ATTACH (1)  ;Desactiva el control automatico del robot
DETACH (0)  ;Realiza el control manual del robot

; Salida en el display del control manual.

WRITE (1) $borrar.display, "Mueve el robot a la posicion 'START' & pulsa
RECORD"
WRITE (1) /X17, "RECORD", $CHR(5), /S
WRITE (1) $CHR(30), $CHR(3), /S     ;Parpadea el led de la paleta de control

WAIT PENDANT(3)   ;Esperar que una tecla sea pulsada

HERE start  ;Graba la posición "start"

; Aviso, en el display de la segunda localización

WRITE (1) $borrar.display, "Mueve el robot a la posicion  'PICK' y pulsa
RECORD"
WRITE (1) /X17, "RECORD", $CHR(5), /S

WAIT PENDANT(3)   ; Esperar que una tecla sea pulsada

HERE pick   ;Graba la posición "pick"

; Aviso, en el display de la tercera localización

WRITE (1) $borrar.display, "Mueve el robot a la posición  'PLACE' y pulsa
RECORD"
WRITE (1) /X17, "RECORD", $CHR(5), /S

WAIT PENDANT(3)   ; Esperar que una tecla sea pulsada

HERE place  ;Graba la posicion "place"

ATTACH (0)   ;Activa el control automatico del robot
DETACH (1)  ;Anula el control manual del robot

RETURN      ;Vuelve a llamar al programa
.END        ; Finaliza el programa
```

A continuación se presenta un programa de ejemplo en el que el robot ha de realizar operaciones de manipulación. En el programa se introduce un ejemplo del concepto de ejecución concurrente (ejecución de varios programas en paralelo) y de la habilitación de un proceso asíncrono para la gestión de errores. El robot ha de realizar una tarea de *pick and place* entre posiciones variables calculadas en un ordenador que se encarga de recibir y transmitir mensajes al robot. Los dos programas, el de comunicaciones con la estación y el de control del robot se ejecutan en paralelo.

El programa de comunicación tendrá el siguiente aspecto:

```
.PROGRAM comunica()
;PROGRAMA DE COMUNICACION CON LA ESTACION
        lu_est  = 10            ; asigna a la unidad lógica de la estación el valor 10
        hay_cod_fun = FALSE     ; indica que no hay código de función disponible
        $rnens_ttir =""         ; el mensaje a transmitir es una cadena vacía
        hay_syst_err = FALSE    ;indica que no hay error del sistema
;CODIGOS DE ERROR
er_rob.no.ok = 256

;INICIALIZA EL CONTROL DEL ROBOT
SPEED 20                        ;se asigna la velocidad del robot
READY                           ;prepara al robot posicionándole en la posición de espera
DETACH (0)                      ;se libera al robot de la tarea #0
EXECUTE 1 robot()               ;se ejecuta el programa de control del robot con la tarea #1
                                ;a partir de aquí los dos programas se ejecutan en paralelo
;ESTABLECIMIENTO DE LA CONEXIÓN ROBOT-ORDENADOR
10      TYPE "Esperando establecer la conexión..."      ;mensaje por pantalla
        ATTACH (lu_est,0)                               ;se asigna la línea de comunicación al programa
        ;se establece un bucle que controla si la comunicación se realiza con éxito
        IF IOSTAT (lu_est,0) <=0 THEN
            TYPE "Error al intentar establecer la comunicación !!!!"
            DETACH (lu_est)                             ;libera la unidad lógica de la tarea
            GOTO 10 ELSE
        ELSE
            TYPE" Robot conectado a la estación"
        END

;RECEPCION Y DECODIFICACION DE ORDENES
;se establece un bucle que comprueba el estado del canal de comunicación y lee el mensaje mandado por el
;ordenador. Si hay un error del sistema se envía un mensaje través del canal lógico y se lee el nuevo mensaje.
        20 DO
            IF(hay_syst_err) THEN
                WRITE (lu_est) $mens_ttir, /S
            END
            READ (lu_est,,1) $mens_rbdo
        UNTIL (IOSTAT(lu_est,0) <>-256)

         ;se comprueba el estado del canal de comunicación comprobando que la recepción es correcta
        IF IOSTAT(lu_est,0) <=0 THEN
            TYPE "iostat error!!!", IOSTAT(lu_est,0)
            TYPE "Recepción incorrecta de orden"
            DETACH (lu_est)                 ;se libera la unidad lógica de la tarea asignada
            GOTO 10
        END
        ;se asigna el código de la función recibida
         cod_fun = ASC($MID($mens_rbdo,l,1))
```

```
;EJECUCION DE ORDENES
;se comprueba si el robot está listo. Si lo está' se da paso al programa de control, si no, se envía un mensaje diciendo
;que el robot no está preparado
        IF ((STATE(1) <>2) OR (codJun == 4)) THEN
                hay cod_fun = TRUE          ;da paso al programa del robot que estaba en espera activa.
                WAIT (NOT hay_con_fun)      ;espera a que se haya ejecutado la orden
        ELSE
                $mens_ttir = $CHR(50+cod_fun)+$INTB(er_rob.no.ok)
        END

;TRANSMISION DE MENSAJES
        WRITE (lu_est) $mens_ttir, /S       ;se envía el mensaje a través de la unidad lógica
        ;se comprueba si hay error en la transmisión. En ese caso se restablece la conexión
        IF IOSTAT(lu_est) <=0 THEN
                TYPE "ERROR en la transmisión de"
                TYPE ASC($MID($mens_ttir, 1,1))
                DETACH (lu_est)             ;se libera el canal de comunicación
                GOTO 10                     ;se restablece la conexión
        END
                Hay_syst_err = FALSE        ;indica que no ha habido error

        GOTO 20                             ;espera a una nueva orden
    .END                                    ;fin del programa de comunicación
```

El programa de control del robot se encarga de ejecutar los códigos de función pasados por el ordenador en $cod_fun y devuelve el mensaje $mens_ttir. Según el código de función recibido, el robot realizar tres tareas : inicialización, tarea de *pick & place* y parada.

El presente ejemplo sólo se muestra la programación de la tarea de pick & place, pudiendo el realizar el ejercicio de completar las otras dos tareas. El programa de control, podría tener el siguiente aspecto:

```
PROGRAM robotO

;INICIAIZACION DE LAS VARIABLES
altura = 20                     ;indica la altura de aproximación y salida en milímetros
rapido = 150                    ;indica la velocidad rápida
lento = 30                      ;indica la velocidad lenta

;PUNTO DE REENTRADA TRAS SYST_ERR
TYPE "El programa de control ha sido relanzado"
REACTE errores                  ;habilita el proceso asíncrono de tratamiento de errores
20 WAIT(hay_cod_fun)            ;espera hasta que esté disponible un código de función
TYPE "Código recibido", cod_fun ;indica a través del monitor el código que se ha recibido
CASE cod_fun OF
        VALUE 1:    ;INICIALIZACION
                ;aquí se incluiría el código de inicialización
        VALUE2:     ;PARADA
                ;aquí se incluiría el código correspondiente a la parada
        VALUE 3:    PICK & PLACE
        TYPE "Recibida la orden de PICK & PLACE"
        ATTACH (0)  ;Asigna al programa la tarea número O

        SET pos_in = TRANS(431,610,523,0,180,45)        ;indica el punto de recogida de las piezas
        SET pos_dej = TRANS(227,-548,7 12,0, 180,45)    ;indica el punto de dejada de las piezas
        ;comienza la operación de picking
        SPEED rapida ALWAYS                             ;selecciona la velocidad
```

```
            ACCEL 90, 75                              ;selecciona la aceleración
            APPRO pos_in, altura          ;se aproxima al punto de recogida a una distancia "altura"
            SPEED lenta                   reduce la velocidad
            MOVE pos_in                   ;se mueve al punto de recogida
            CLOSEI                        ;cierra la pinza
            DEPARTS altura                ;se separa hasta una distancia "altura"

            ;operación de dejada de la pieza
            SPEED rapida ALWAYS           ;selecciona la velocidad
            ACCEL 90,75                   ;selecciona la aceleración
            APPRO pos_dej, altura         ;se aproxima al punto de dejada a una distancia "altura"
            SPEED lenta                   ;reduce la velocidad
            MOVE pos_dej                  ;se mueve al punto de dejada
            OPENI                         ;abre la pinza
            DEPARTS altura                ;se separa hasta una distancia "altura"
.END        ;fin de la instrucción CASE

hay_cod_fun = FALSE           ;indica que ya no hay código de función
GOTO 20                       ;regresa al estado de espera
.END                          ;fin del programa de control
```

Por último, se muestra un ejemplo de cómo sería el programa de tratamiento de errores.

```
.PROGRAM errores()

    TYPE "RUTINA DE TRATAMIENTO DE ERRORES DEL SISTEMA V+"
    TYPE "Error número: ",ERROR(-1)
    TYPE "Mensaje: ",$ERROR(ERROR(-1))

    error = ABS(ERROR(-1))        ;asigna el error que se ha producido
    $cod_er = $INTB(error)        ;comprueba cuál es el error
    hay_Sist._err = TRUE          ;indica que se ha producido un error
    hay_cod_fun = FALSE           ;indica que no hay código de función
    EXECUTE 2 err_robot()         ;ejecuta el programa err_robot asignándole la tarea 2
    CYCLE.END 2
    RETURN
    .END                          ;fin del programa de tratamiento de errores


.PROGRAM err_robotO
    ABORT 1                       ;aborta la ejecución del programa de control del robot
    CYCLE.END 1
    EXECUTE 1 robotO              ;ejecuta el programa de control del robot ;asignándole la tarea #1


    ABORT2
    .END                          ;fin del programa err_robotO
```

4. BIBLIOGRAFÍA

LIBROS

Robótica: Control, Detección, Visión e Inteligencia
K.S. FU, R.C González, C.S.G. LEE
Ed. McGraw Hill

Robótica Practica Tecnología y Aplicaciones
José Ma. Angulo
Ed. Paraninfo

Robótica Industrial: Fundamentos y aplicaciones
Arantxa Rentería, Maria Rivas
Ed. McGraw Hill

Fundamentos de robótica,
Antonio Barrientos, Luis Felipe Peñín, Carlos Balaguer i Rafael Aracil
Ed. McGraw Hill

MANUALES

Curso nº 1 de VAL II
Curso nº 2 de VAL II
Curso de V+ (Francés)

WEB'S

www.staubli.com

www.adept.com