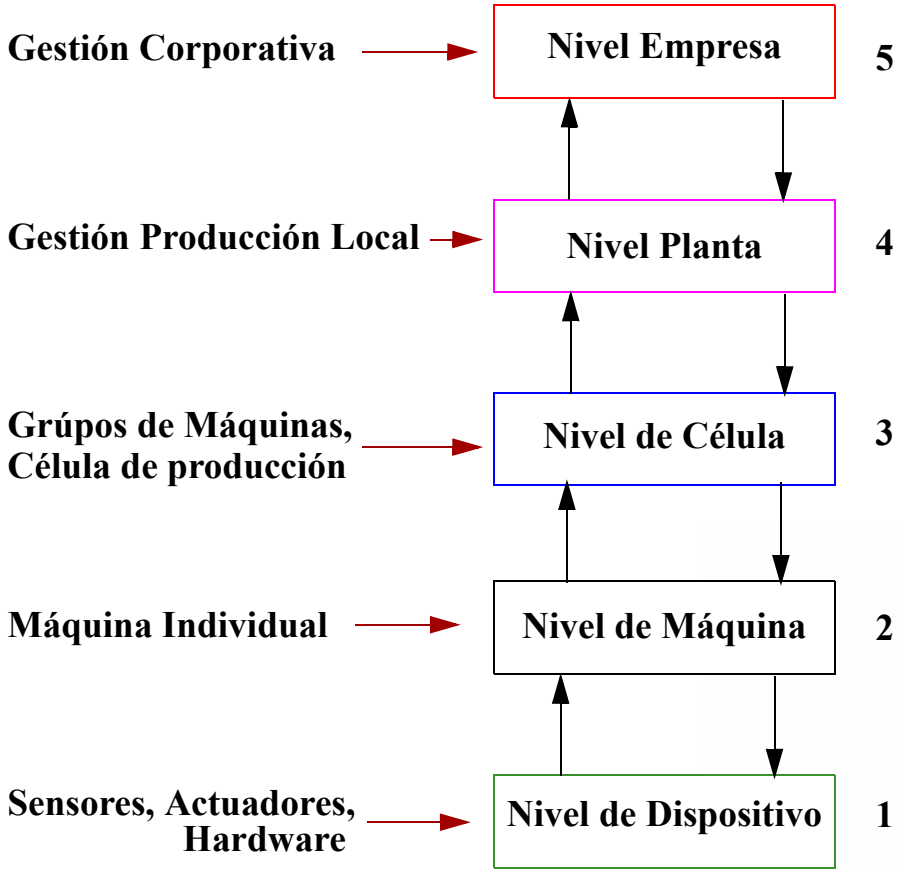
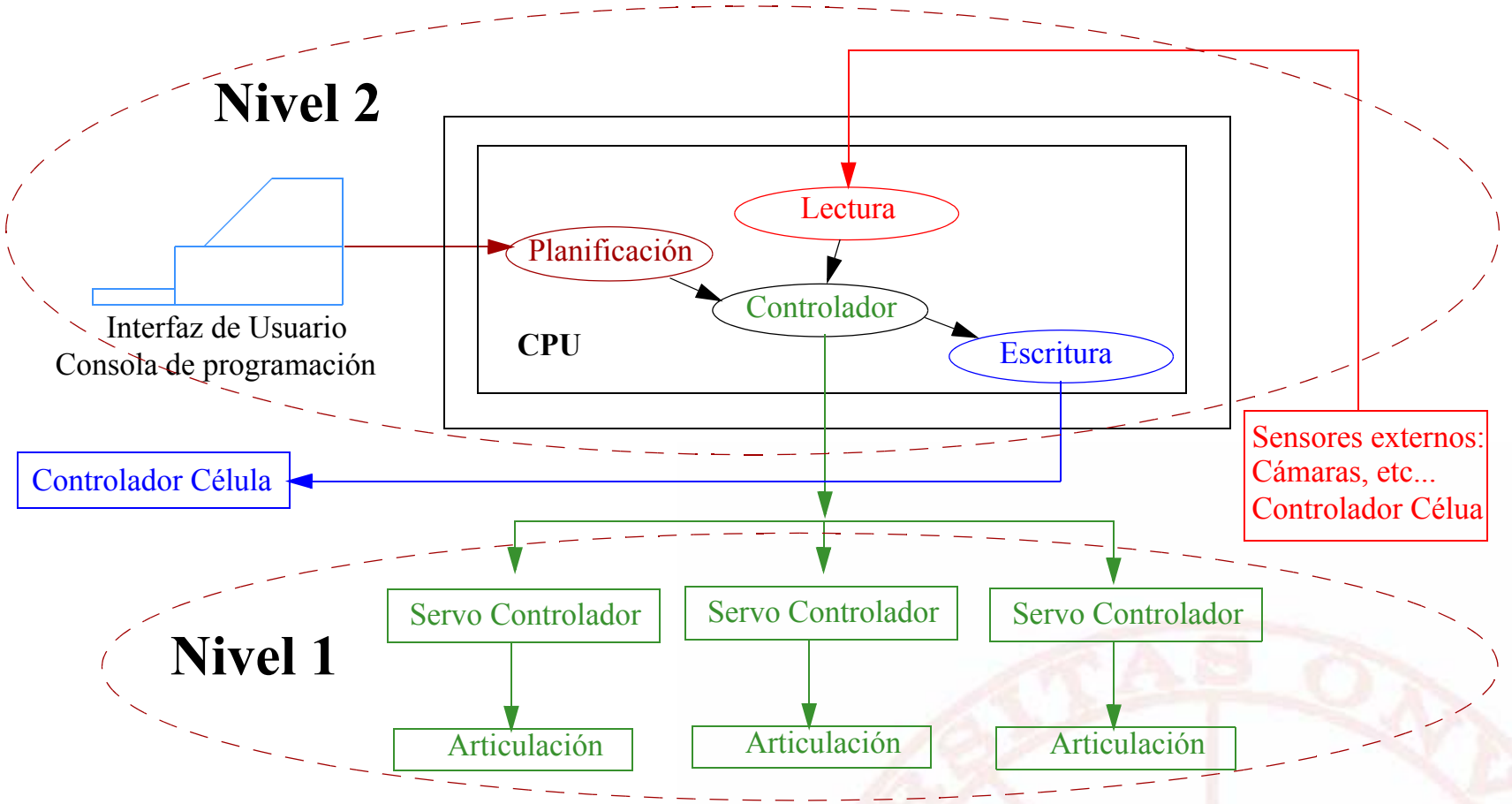
TEMA 3.5 PROGRAMACIÓN D MANIPULADORE

3.5.1 Niveles de Automátización y Células de Produscción
3.5.2 El manipulador dentro de la Célula de Producción.
3.5.3 Programación de Robots Industriales

Niveles de Automatización y Células de Producción

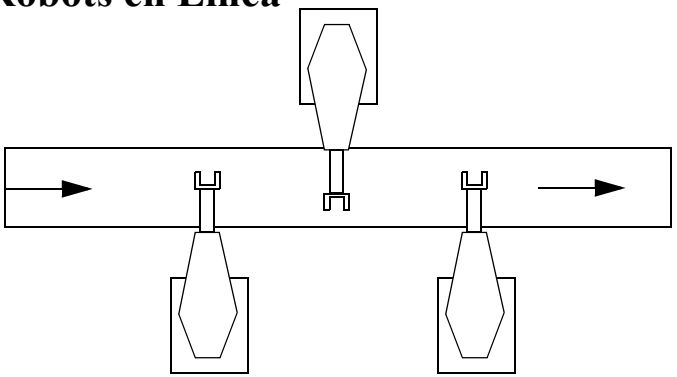


Arquitectura Manipulador Industrial: Niveles 2 y 1)

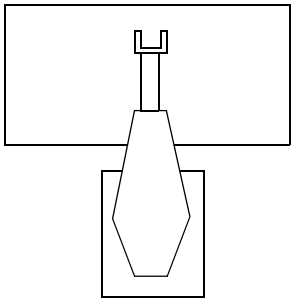


El Manipulador dentro de la Célula de Producción: Nivel 3

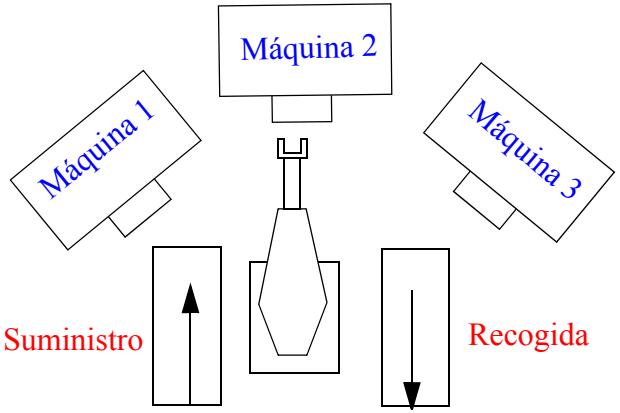
Robots en Línea



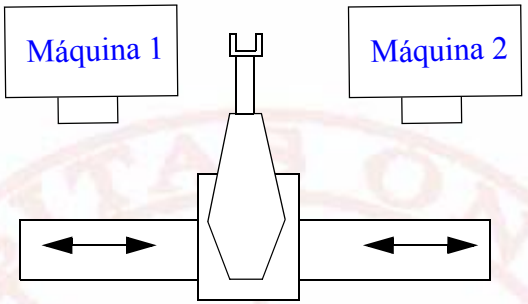
Robot Servido



Robot en Centro



Robots Base Móvil



Programación de Robots Industriales

Es posible hablar de diferentes niveles de programación.

Programación a bajo nivel: El programa maneja directamente los controladores de las articulaciones accediendo a los valores de los sensores y actuadores de cada articulación.

Programación a nivel articular: El control del movimiento está delegado en elementos incorporados a la arquitectura de control. reposando en el programa la responsabilidad de planificar las trayectorias en el espacio de trabajo o en el articular

Programación de nivel superior: El sistema de control provee de funciones que permiten que el programa se dedique a especificar la tarea a realizar de una forma más o menos abstracta.

Programación articular y de bajo nivel

Es habitual diferenciar entre el control de las articulaciones y la definición de la trayectoria articular.

Es necesario establecer cuáles son los objetivos de manipulación definidos por la tarea y traducir dichos objetivos al espacio articular, aplicando las técnicas de cálculo del problema cinemático inverso o del inverso del jacobiano.

Con esta información, el sistema de control establecerá los acciones necesarias para que los actuadores hagan que las articulaciones adopten las configuraciones necesarias.

Lenguajes de programación de programación de dispositivos electrónicos basados en microprocesadores, tales como ensamblador, C, Java etc.

Programación de alto nivel

La programación de alto nivel en robot industriales consiste en indicar paso por paso las diferentes acciones (moverse a un punto, adoptar una configuración, abrir o cerrar la pinza, etc.) que éste deberá realizar durante su funcionamiento.

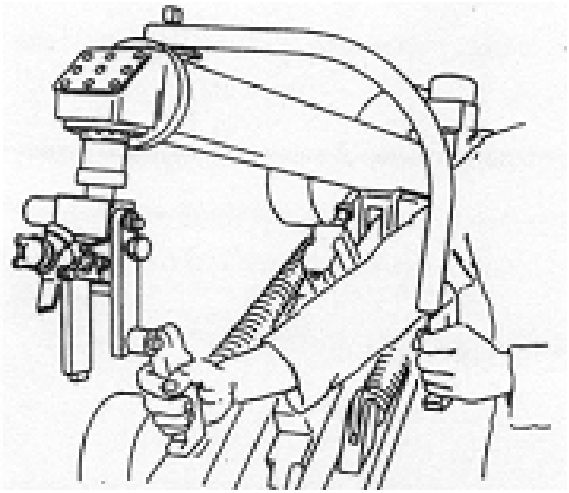
Actualmente no existe normalización en relación a los procedimientos de programación de robots, cada fabricante desarrolla su método particular, el cual es válido solamente para sus propios robots.

Existen varios criterios para clasificar los métodos de programación. Según el sistema utilizado para indicar la secuencia de acciones:

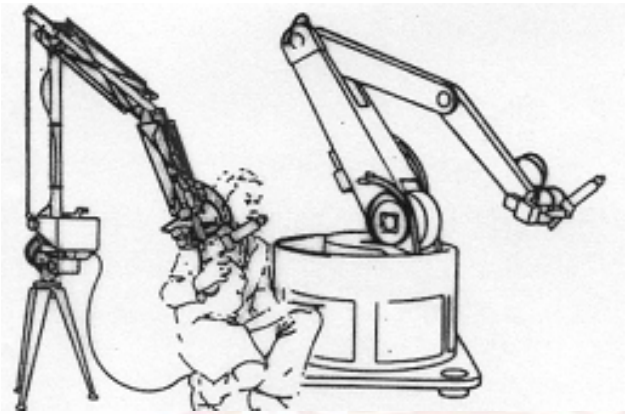
* **Programación por guiado**.
* **Programación Textual**.

Programación por Guiado

La programación por guiado o aprendizaje consiste en hacer realizar al robot, o a una maqueta del mismo, la tarea, registrando las configuraciones adoptadas para su posterior repetición en forma automática.



Pasivo Directo



Pasivo Indirecto

Programación por Guiado

Guiado activo: Esta posibilidad permite emplear el propio sistema de accionamiento del robot, controlado desde una botonera joystick para que sea éste el que mueva sus articulaciones.

Existen dos formas básicas de registro de los movimientos:

Registro en puntos de paso: El robot es guiado por los puntos por los cuales se desea que pase durante la fase de ejecución automática del programa. Durante el guiado, se registran dichoso puntos y la unidad de control establecen las trayectorias que interpolan dichos untos.

Registro continuo: Se registran con un frecuencia de muestreo fija los movimientos de guiado.

Programación textual

Este método de programación permite indicar la tarea al robot a través de un lenguaje de programación específico. Un programa se entiende como una serie de órdenes que son editadas y posteriormente ejecutadas, por lo tanto, existe un texto para el programa.

La programación textual se puede clasificar en tres niveles:

Nivel robot: las órdenes se refieren a los movimientos a realizar por el robot.

Nivel objeto: las órdenes se refieren al estado en que deben ir quedando los objetos.

Nivel tarea: las órdenes se refieren al objetivo a conseguir.

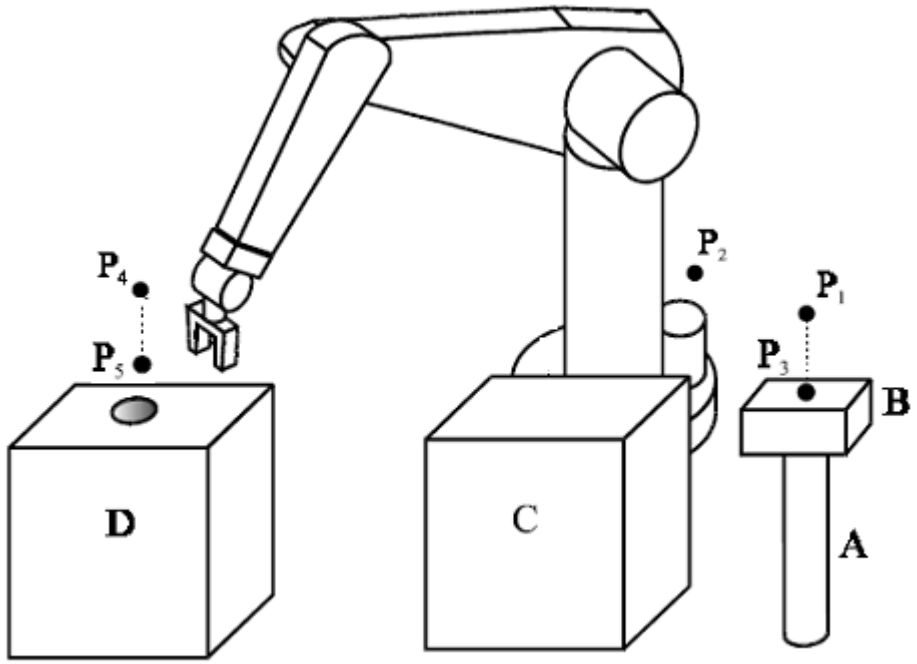
Actualmente, la mayoría de los lenguajes de programación son de nivel robot, entre los que destacan por orden cronológico los siguientes:

- AL (Universidad de Stanford - 1974)
- AML (IBM - 1979)
- LM (Universidad de Grenoble - 1981)
- VAL II (Unimation-1984)
- V+(ADEPT - 1989)
- RAPID (ABB - 1994)

A nivel destacan los siguientes ejemplos:

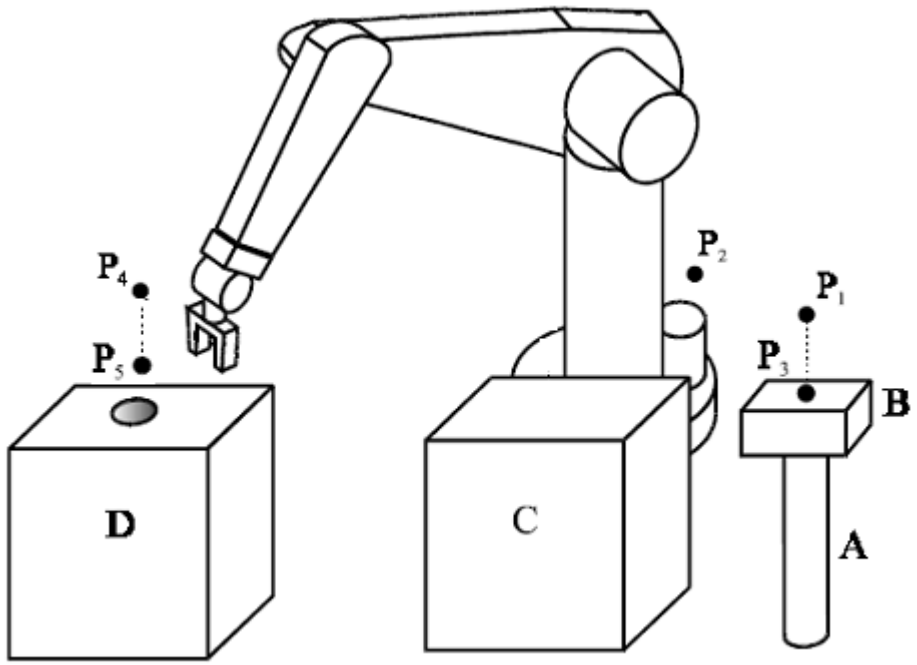
- LAMA (MIT - 1976)
- AUTOPASS (IBM - 1977)
- RAPT (Universidad de Edimburgo- 1978)

Programación textual, una primera aproximación:



Nivel Tarea: se especifica qué es lo que debe hacer el robot en lugar de cómo debe hacerlo.

Ensamblar A con D



Nivel Objeto: Las instrucciones se dan en función de los objetos a manejar:

Situar B sobre D haciendo coincidir
LADO_B1 con LADO_D1;
Situar A dentro D haciendo coincidir
EJE_A con EJE_HUECO_ y BASE_A con BASE_D ;

Nivel Robot: Se debe especificar cada uno de los movimientos que ha de realizar el robot, como velocidad, direcciones de aproximación y salida, apertura y cierre de la pinza, etc.

```
Mover_a P1 via P2; Situarse en un punto sobre la pieza B
Vel = 0.2 * VELMAX; Reducir la velocidad
Pinza = ABRIR; Abrir la pinza
Prec = ALTA            ; Aumentar la precisión
Mover_recta_a P3; Descender verticalmente en línea recta
Pinza = CERRAR ; Cerrar la pinza para coger la pieza B
Espera= 0.5; Esperar para garantizar cierre de pinza
Mover_recta_a P1; Ascender verticalmente en longa recta
Prec = MEDIA; Decrementar la precisión
Vel = VELMAX; Aumentar la velocidad
mover_a P4 via P2; Situarse sobre la pieza C
Prec = ALTA; Aumentar la precisión
Vel = 0.2 * VELMAX ; Reducir velocidad
Mover_recta_a P5; Descender verticalmente en línea recta
Pinza = ABRIR; Abrir pinza
```

Requisitos para los sistemas de programación de robots

Tradicionalmente los requerimientos generales que se han establecido para un sistema de programación de robots son los siguientes:

- Entorno de Programación
- Modelado del Entorno
- Tipo de Datos
- Manejo de Entradas/Salidas (digital y análoga)
- Control del Movimiento del Robot
- Control del flujo de ejecución del programa

Programación en VAL II y V+.

El lenguaje VAL (Victor's Assembly Language), fue el primer lenguaje para robots comercialmente disponible.

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.

VAL II es un lenguaje y sistema de control basado en computador diseñado para los robots industriales **Unimation**. VAL II, reemplazó a en 1984.

Este lenguaje, forma parte de la segunda generación de lenguajes de programación de robots.

Programación en VAL II y V+.

Los lenguajes de la segunda generación salvan algunas limitaciones de los lenguajes de la primera generación y las añaden a estas nuevas capacidades.

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

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 a veces se denomina indistintamente V+ y VAL III.

Programación en VAL II y V+.

Un programa en estos lenguajes consiste en un conjunto de instrucciones secuenciadas en líneas de programa. El formato general de cada línea es:

número_linea <etiqueta> <instrucción> <;comentario>

La etiqueta es un número entero que permite identificar un lugar en el código para poder ser direccionado con sentencias GOTO.

La primera línea de un programa es la instrucción ".PROGRAM" seguida del nombre del programa y de los parámetros que deba recibir o devolver.

.PROGRAM nombre <(lista de parámetros)>

El final del programa se indica con una línea que contiene la instrucción **".END"**

VARIABLES

Respecto a su ámbito, es posible definir variables **globales**, **locales** o **automáticas**.

Variables globales, son aquella cuyo valor que pueden se accedida por cualquier de los programas que se encuentren en memoria.

Variables locales se definen mediante la **instrucción LOCAL**. Estas variables solo pueden ser utilizadas en el ámbito del programa o rutina donde han sido definidas. Mantienen su valor entre llamadas al programa.

Variables automáticas. Son parecidas a las variables locales, pero cada vez que se entra al programa, se crea una copia separada de las existentes hasta ese momento. El valor de cada copia se pierde cada vez que se sale del programa. Estas últimas, se crean con la **instrucción AUTO**.

Respecto de uso, existen dos tipos de variables: las **variables de punto**, y las **variables reales**.

Dentro de las variables de punto podemos distinguir dos tipos: **puntos de precisión** y **localizaciones en el espacio**.

Un **punto de precisión** es un punto en el espacio articular del robot. Se declara añadiendo # delante del nombre del punto. Se puede definen utilizando la **función PPOINT**:

#lugar = PPOINT (a , b, c, d, e, f)

También es posible establecer un punto de precisión con el **comando Here**:

HERE #lugar

Un **punto de localización** en el espacio 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 donde y (yaw), p (pitch) y r (roll) representa los ángulos de Euler ZYX.

La sentencia SET-TRANS se utiliza para definir una variable, punto de localización.

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)

Se recomienda la consulta de la bibliografía Ollero, (2001), Capítulo 11, para conocer más detalles sobre la definición de transformaciones compuestas y especificación de localizaciones.

En cuanto a las **variables reales**, su definición se realiza mediante una **asignación a un nombre cualquiera de un valor entero o real**.

Escribir A=12 declara y asigna la variable A con el valor 12.

En el caso de VAL II, el valor debe estar comprendido obligatoriamente entre 6.0E-39 y 2.854E+38 para valores positivos y hasta -6.0E-39 para valores negativos.
Operaciones matemáticas con variables reales:

+,-,*,/.
ABS(expr)
SIGN(expr) da el signo de la expresión.
FRACT(expr) Da la parte fraccionaria de la expresión.
INT(expr) Da la parte entera de la expresión.
etc...

SENTENCIAS DE CONFIGURACIÓN Y MOVIMIENTO

MOVE <VARIABLE_DE_PUNTO/>: Mueve al robot a la configuración descrita por la variable por medio de una interpolación articulada.

MOVES <VARIABLE_DE_PUNTO/>: Mueve al robot a la configuración descrita por la variable. La mano es movida en línea recta. La "S" indica "straight line".

DELAY: Causa que el movimiento del robot pare por un periodo de tiempo especifico.

OPENI o **CLOSEI**: Abre o cierra inmediatamente la pinza, asume un estado abierto o cerrado respectivamente.

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.

DEPART <VALOR>: Movimiento de decremento de la coordenada Z con el valor especificado.

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.

SENTENCIAS DE GESTIÓN DE SEÑALES EXTERNAS

SIG <CANAL>: Devuelve true si la señal del canal está a ON.

WAIT SIG <CANAL>: Para la ejecución del programa hasta que la señal del canal de entrada especificado esté en ON.

SIGNAL <CANAL>: Activa o desactiva las señales externas de salida especificas.

RESET: Apaga todas las señales de salida.

SENTENCIAS DE CONTROL DE FLUJO

GOTO <ETIQUETA>: Realiza un parte incondicional del programa identificado por ETIQUETA.

IF ..... GOTO: Salto condicional a la marca especificada.

CALL: Llama a una subrutina.

RETURN: Final de una Subrutina.

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.

PROMPT "<mensaje>",<variable>,<variable>,..: Esta instrucción permite, a partir del teclado, entrar los valores que se asignan a las variables.

CASE: Proceso iniciado con una estructura del CASE definiendo el valor de interés.

TYPE {/carácter de control},{variable},{etc.} Esta instrucción permite hacer salir por pantalla un mensaje durante la ejecución de un programa y/o hacer salir en pantalla el contenido de una o varias variables declaradas.

WHILE: Inicia un proceso con la estructura WHILE si la condición es TRUE o salta la estructura del WHILE si la condición es inicialmente FALSA

EJEMPLOS DE PROGRAMAS

Estrategias para afrontar el diseño de un programa

1º Diseñar el flujo del programa

2º Diseñar los movimientos

3º Cálculo de las configuracones que se quieren alcanzar

4º Escribir el código

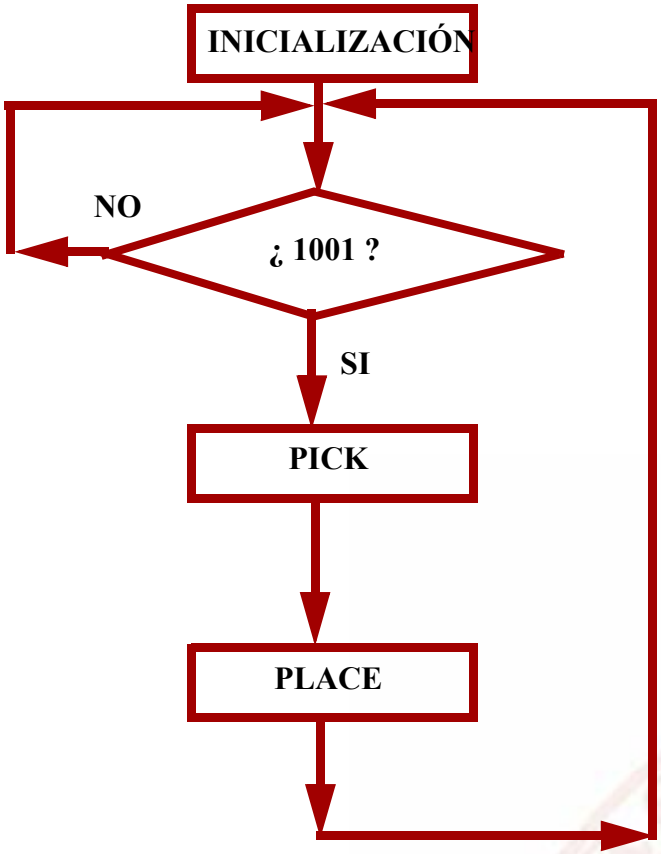
5º Comprobar el funcionamiento

EJEMPLOS DE PROGRAMAS

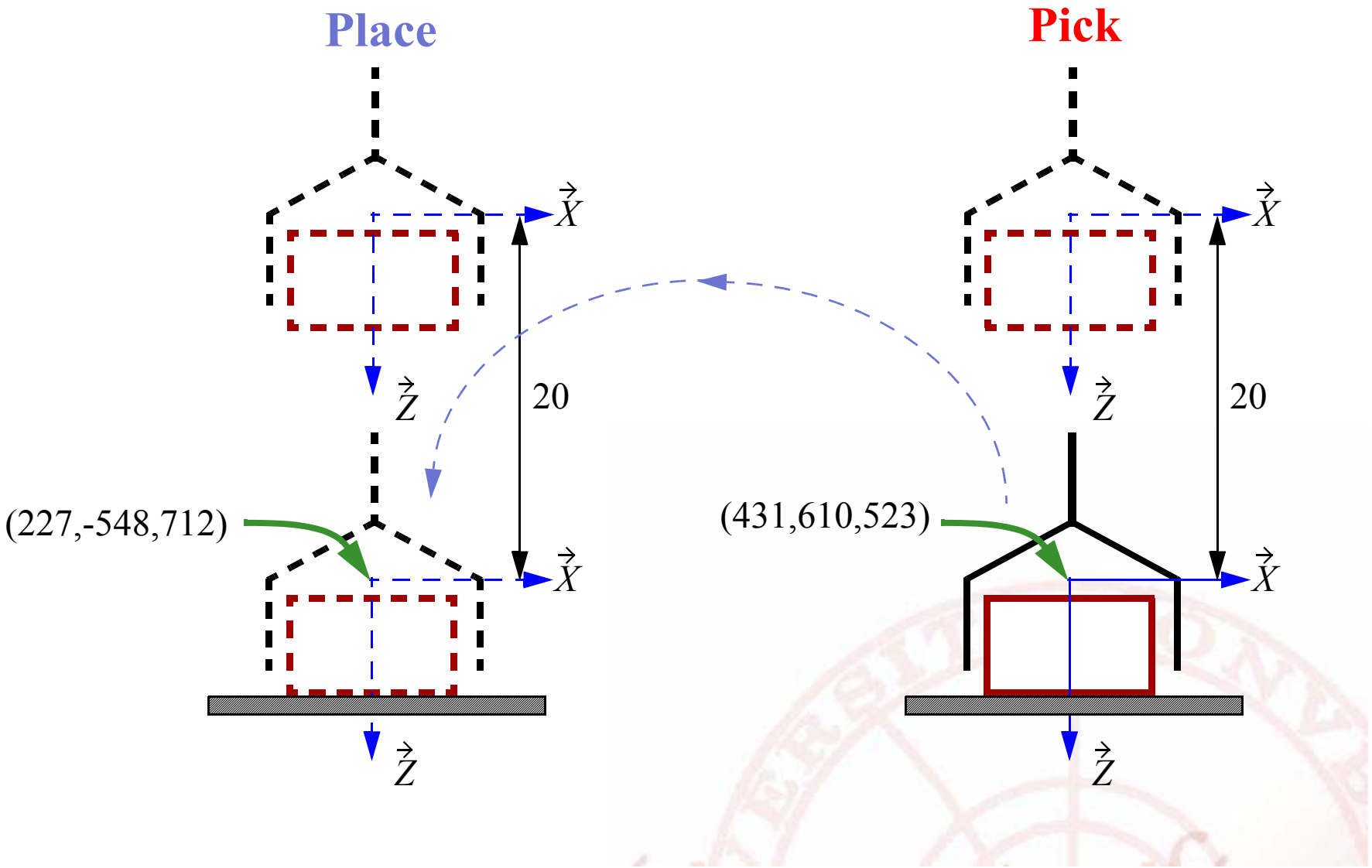
Ejemplo 1:

Realizar un programa para realizar una tarea de pick and place de una pieaz que será detectada mediante un sensor conectado al canal 1001. EL manipulador debe coger la pieza situando la pinza en el punto (431,610,523) con unos ángulos de Euler ZXY (0,180,45) y debe depositar la pieza en el punto (227,-548,712) con unos ángulos de Euler ZXY (0,180,45).

PLANIFICANDO EL FLUJO DEL PROGRAMA



PLANIFICANDO MOVIMIENTOS



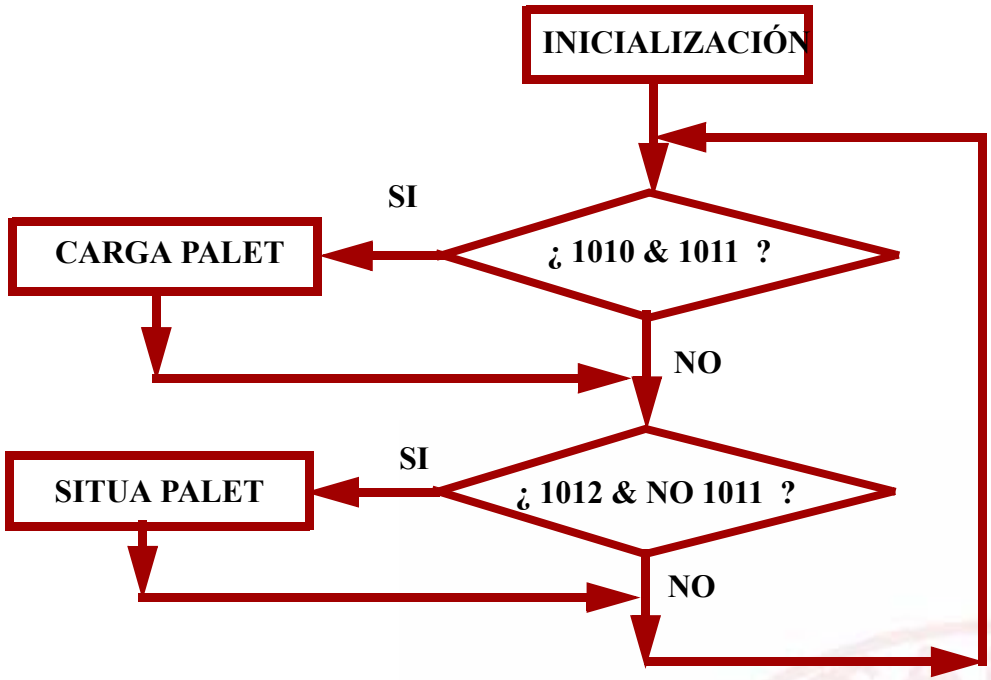
DISEÑO DEL CÓDIGO

```
.PROGRAM robot()
;INICIALIZACION DE LAS VARIABLES
altura = 20 ;indica la altura de aproximación y salida en milímetros
rapido = 100 ;indica la velocidad rápida
lento = 30 ;indica la velocidad lenta

TYPE "El programa de control ha sido relanzado"

20 WAIT SIG(1001) ;espera detección de la pieza

    TYPE "Recibida la orden de PICK & PLACE"
    SET pos_in = TRANS(431,610,523,0,180,45) ; punto de recogida
    SET pos_dej = TRANS(227,-548,712,0,180,45); punto de dejada

    ;///////
    ;comienza la operación de picking
    ;/////

    SPEED rapida  ;selecciona la velocidad
    APPROS pos_in, altura ;se aproxima al punto de recogida
    SPEED lenta, ALWAYS; reduce la velocidad
    MOVES pos_in ;se mueve al punto de recogida
    CLOSEI ;cierra la pinza
    DEPARTS altura ;se separa hasta una distancia "altura"

    ;///////
    ;comienza la operación de place
    ;/////

    SPEED rapida ;selecciona la velocidad
    APPROS pos_dej, altura ;se aproxima al punto dejada
    MOVES pos_dej ;se mueve al punto de dejada
    OPENI ;abre la pinza
    DEPARTS altura ;se separa hasta una distancia "altura"


    GOTO 20 ;regresa al estado de espera

.END ;fin del programa de control
```

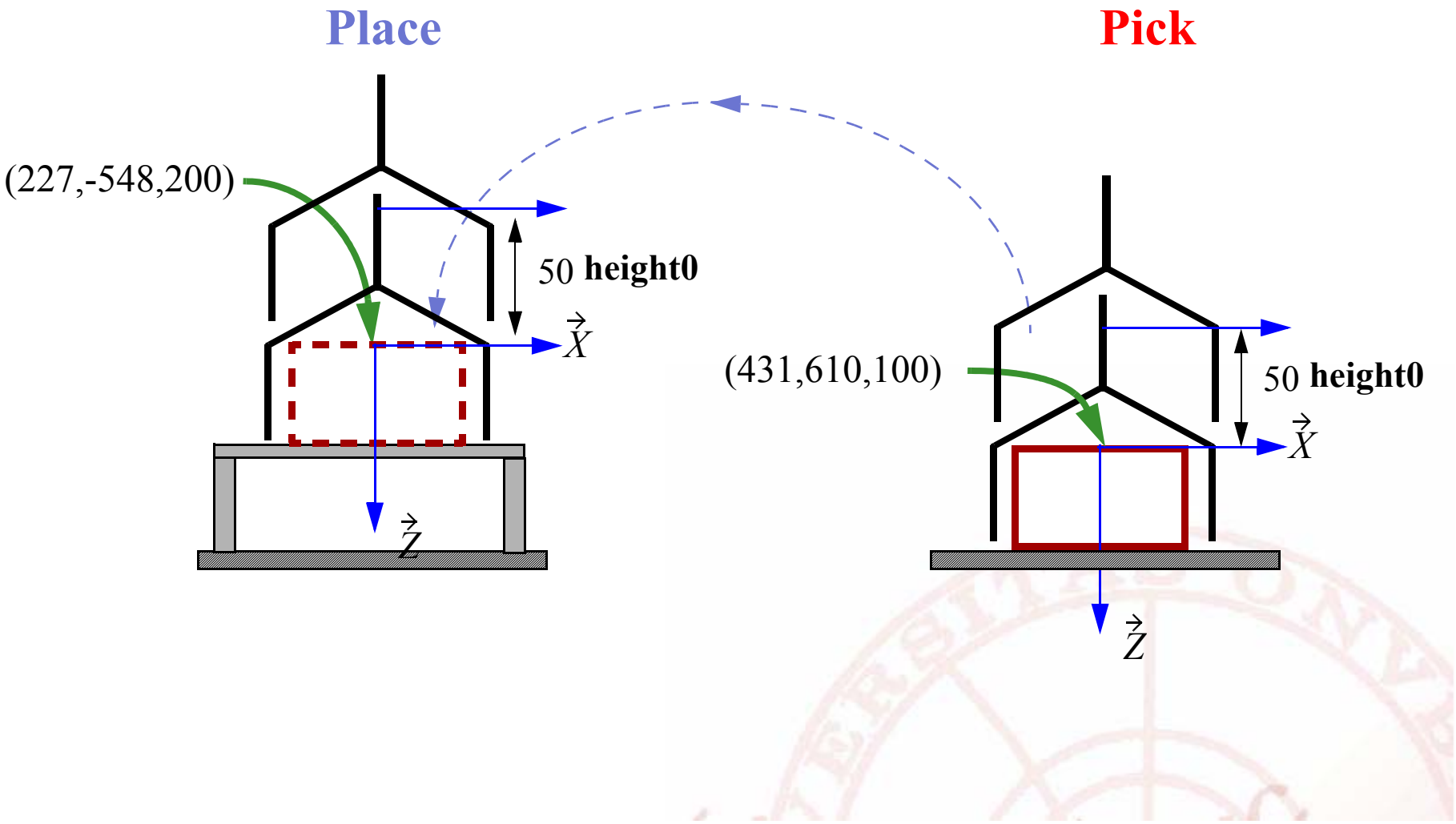
Ejemplo 2:

Programar un manipulador para que cargue un palet con cajas que le son servidas. El palet estará cargado con un total de 3 cajas. Cada caja mide 100 mm de alto al igual que el palet sobre el que irán cargadas. **Cuando haya una caja** preparada para ser cargada el **canal 1010 estará a nivel alto**. **Cuando haya un palet colocado** para ser cargado el **canal 1011 estará a nivel alto**. Igualmente si no hay palet preparado para ser cargado, el manipulador deberá coger un palet de la una zona donde debe haber un palet de reserva. **Si hay palet de reserva** estará activado el **canal 1012**. Una vez el palet ha sido cargado con las tres cajas, debe **encenderse una luz** que se activará con el **canal de salida 10** y el manipulador quedar a la espera de que el palet sea retirado para volver a iniciar de nuevo el ciclo de carga. **Las caja se recogerán en** la posición (431,610,100) **y el palet estará situado** en (227,-548,202). **El palet vacio estará ubicado** en (431,610,523,0,180,45). Las cajas y el palet se recogerán con la muñeca invertida.

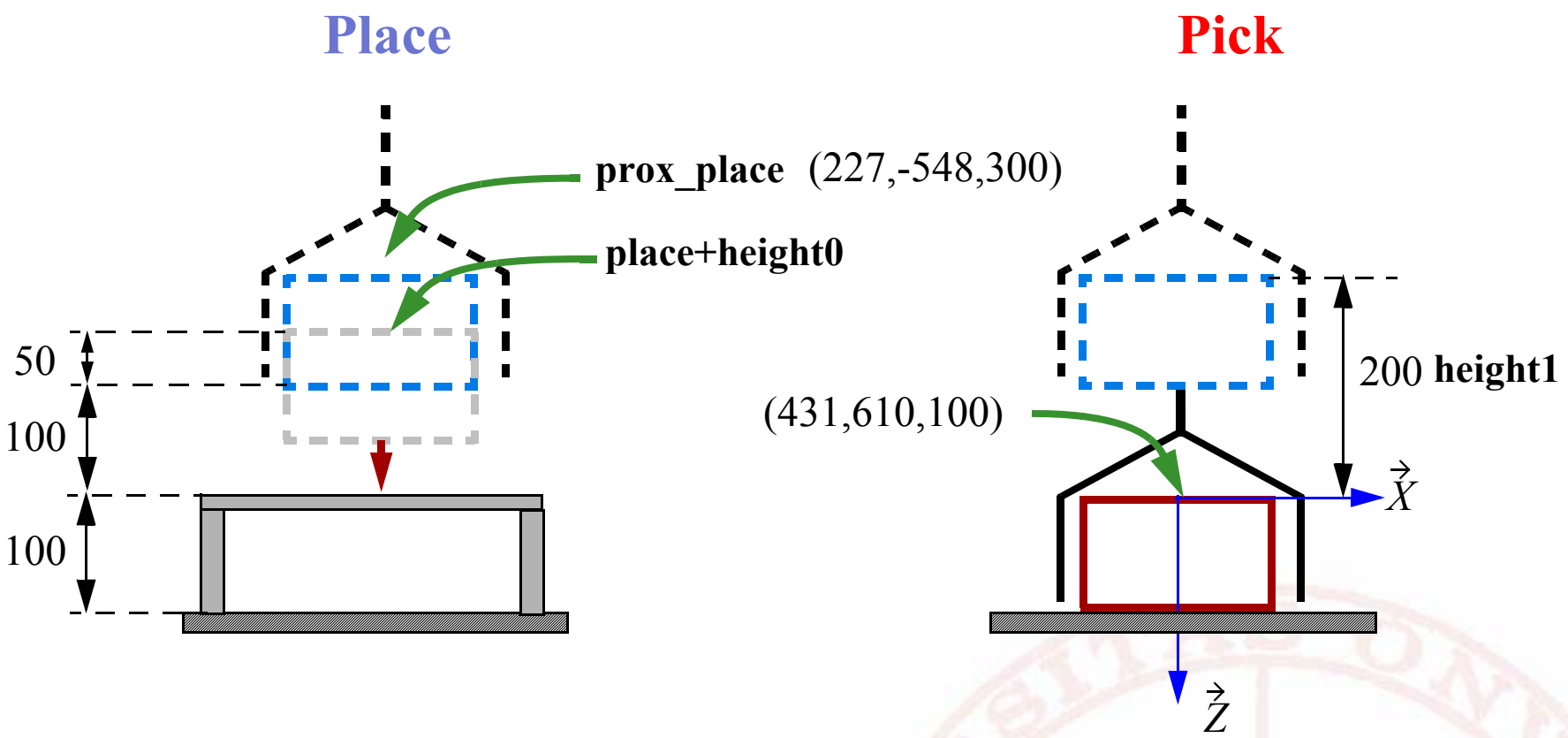
PLANIFICANDO EL FLUJO DEL PROGRAMA



PLANIFICANDO MOVIMIENTOS



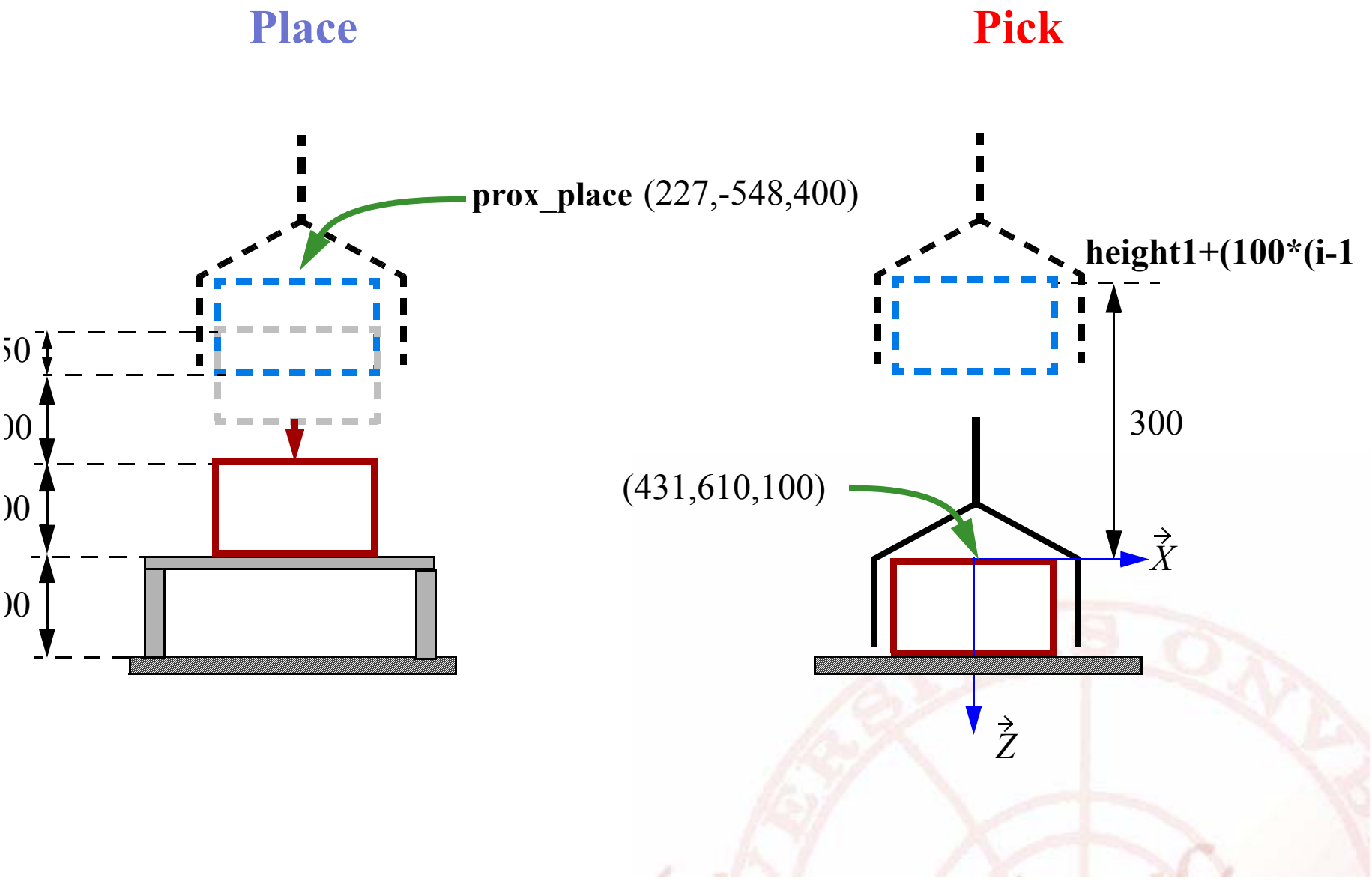
PLANIFICANDO MOVIMIENTOS: 1ª CAJA



PLANIFICANDO MOVIMIENTOS: 2ª CAJA

DISEÑO DEL CÓDIGO

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
.PROGRAM principal();
;DESCRIPCION: Este programa queda a la espera de leer la activación del canal 1010 que detecta
que ha sido colocada una caja, del canal 1011 si hay un palet dispuesto a ser cargado, y del canal 1012
que detecta que hay un palet libre para se ubicado en la zona de carga.

SET #start=PPOINT(0,0,0,0,0,0)

      WHILE(TRUE) DO

            MOVE start ; mover a la localización segura de inicio

            IF SIG (1010, 1011) THEN ; si hay caja y palet para cargar
            CALL carga.palet
            END

            IF SIG (1012,-1011) THEN
            CALL situa.palet
            END
```


Place
Pick
prox_place (227,-548,400)
50
00
00
00
height1+(100*(i-1
300
(431,610,100)
X
Z

```
      END

.END
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

DISEÑO DEL CÓDIGO

DISEÑO DEL CÓDIGO

```
.PROGRAM carga.palet()
;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 = 3 ; nº de cajas a apilar
    height0 = 50 ; altura de "appros" en "pick" y en place
    height1 = 200 ; altura de "appro/departs" en "pick"

    fast = 100 ;indica la velocidad rápida
    slow = 30 ;indica la velocidad lenta

    SET pick = TRANS(431,610,100,0,180,0) ; punto de recogi-
    da
    SET place = TRANS(227,-548,202,0,180,0); punto de dejada
    SET prox_place = TRANS(227,-548,300,0,180,0); punto de
    aproximación

    OPEN ; apertura de pinza

    FOR i = 1 TO parts ; iniciar el apilado de cajas
        SPEED rapida ALWAYS; auenta velocidad
        APPROS pick, height1 ; aproximar a "pick-up"
        WAIT SIG(1010); espera a que haya caja.
        APPROS pick, height0 ; aproximar a "pick-up"
        SPEED slow ;reduce la velocidad
        MOVES pick ; mover hacia la caja
        CLOSEI ; cerrar la pinza
        DEPARTS (height1+100*(i-1)) ; volver a la pos-
        ición anterior
        MOVES prox_place  ;movimiento en linea recta
        para situar sobre palet
        APPROS place, height0 ; ir a "put-down"
        SPEED slow; ALWAYS reduce la velocidad
        MOVES place ; mover en línea recta a la localiza-
        ción de destino
        OPENI; abrir la pinza
        DEPARTS height0 ; volver a la posición anterior-
        SPEED rapida ALWAYS; auenta velocidad
        MOVES prox_place
        SHIFT place BY 0.00, 0.00, 100
        SHIFT prox_place BY 0.00, 0.00, 100
    END

    TYPE "Fin de tarea. ", /IO, parts, " cajas apliadas."
    SINAL 10 ;Activa la luz de retirado de palet
    READY
    WAIT SIG(-1011) ; espera a que sea retirado el palet
.END
```

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
.PROGRAM situa.palet ()
;DESCRIPCION: Este programa coloca  un palet libre en la zona de carga

        SET pick_palet_vacio = TRANS(431,610,523,0,180,45) ; punto de recogida
        SET place palet_vacio = TRANS(227,-548,102,0,180,45); punto de dejada
        OPEN; apertura de pinza
        SPEED 100 ; velocidad alta
        APPRO pick_palet_vacio, 100 ; ir a "pick-up"
        SPEED 20 ; velocidad baja
        MOVES pick_palet_vacio ; mover hacia el palet vacio
        CLOSEI; cerrar la pinza
        SPEED 80 ALWAYS; auenta velocidad
        DEPARTS 100 ; volver a la posición anterior
        APPRO place palet_vacio, 30 ; aproximacion
        SPEED 20 ; velocidad baja
        MOVES place palet_vacio ; coloca en zona de carga
        OPENI ; abrir la pinza
        SPEED 100 ALWAYS; auenta velocidad
        DEPARTS 300 ; volver a la posición anterior
        READY
.END
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

TEMA III: ROBOTS ARTICULADOS