Escuela Politécnica Superior de Linares

Grado en Ingeniería Mecánica

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

PUESTA EN MARCHA DE BRAZO ROBÓTICO Y DESARROLLO DE APLICACIONES

Alumno: **Axel José Córdoba López**

Tutor: Prof. D. Luis Felipe Sesé
Depto.: Ingeniería Mecánica y Minera

Septiembre, 2016

Escuela Politécnica Superior de Linares

Grado en Ingeniería Mecánica

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

PUESTA EN MARCHA DE BRAZO ROBÓTICO Y DESARROLLO DE APLICACIONES

Alumno: **Axel José Córdoba López**

Tutor: Prof. D. Luis Felipe Sesé
Depto.: Ingeniería Mecánica y Minera

CORDOBA LOPEZ AXEL JOSE - 26253246B

Firmado digitalmente por CORDOBA LOPEZ AXEL JOSE - 26253246B
Fecha: 2016.09.01 12:31:57 +02'00'

FELIPE SESE LUIS ANTONIO - 26242833V

Firmado digitalmente por FELIPE SESE LUIS ANTONIO - 26242833V
Nombre de reconocimiento (DN): c=ES, serialNumber=26242833V, sn=FELIPE SESE, givenName=LUIS ANTONIO, cn=FELIPE SESE LUIS ANTONIO - 26242833V
Fecha: 2016.09.01 12:55:42 +02'00'

Septiembre, 2016

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 RESUMEN

Este trabajo consiste en la puesta en marcha y desarrollo de aplicaciones para un brazo robótico IRB120 de la marca ABB, el cual constituye el primer equipo de robótica en la Escuela Politécnica Superior de Linares.

Para desarrollar dicho trabajo, ha sido imprescindible realizar un estudio de las propiedades geométricas, mecánicas y cinemáticas del robot con el propósito de definir su funcionamiento.

Como parte de la puesta en marcha del robot en el laboratorio, se ha diseñado una célula de trabajo en la que interactuar y se le han proporcionado los medios necesarios para un correcto funcionamiento.

Con objeto de acercar la robótica al alumnado, se ha diseñado una serie de prácticas en el laboratorio con las que se pretende enseñar el funcionamiento de un brazo robótico de la marca ABB, explicando la programación necesaria para su uso de una forma sencilla, progresiva y visual.

1 ABSTRACT

This project consists of the implementation and application development for a robotic arm IRB120 of the ABB brand, which is the first robotic equipment at the Polytechnic School of Linares.

To develop this work, making a study of geometric, mechanical and kinematic of the robot has been essential in order to define its operation properties.

As part of the commissioning of the robot in the laboratory, it has designed a work space to interact and has been provided with the necessary means for a proper operation.

In order to bring robotics to students, a serie of practices of laboratory has been designed to teach the operation of a robotic arm of the ABB brand, explaining the programming necessary to use in a manner simple, progressive and visual.

2 OBJETIVO Y MOTIVACIÓN

En este capítulo se presenta el objeto de realizar dicho trabajo, la motivación y los objetivos que se quieren cumplir.

2.1 Objeto

La titulación del Grado en Ingeniería Mecánica, al igual que el resto de titulaciones impartidas en la Escuela Politécnica Superior de Linares, debe concluir según la normativa del plan vigente con la elaboración y la defensa por parte del estudiante de un Trabajo Fin de Grado. Este trabajo debe realizarse en la fase final del plan de estudios y debe estar orientado a la evaluación de competencias asociadas al título.

2.2 Motivación

Como puede verse en la industria, cada vez es más usual la utilización de robots para cualquier tipo de servicio, desde grandes cadenas de montaje a pequeñas células de trabajo. Se puede decir, por tanto, que la robótica en general está en auge exigiendo que los futuros ingenieros lleguen a la sociedad con una buena base de conocimientos sobre manejo de robots.

La reciente incorporación del primer robot industrial dedicado a la docencia en la Escuela Politécnica Superior de Linares ha generado la necesidad de una puesta en marcha del equipo, que permita acercar la robótica al alumnado, estudiándose el robot con detenimiento desde puntos teóricos y creando prácticas que expliquen el funcionamiento de los robots industriales.

2.3 Objetivos

En este trabajo se pretenden alcanzar los siguientes objetivos:

- Realizar un estudio del brazo robótico en distintas categorías.
- Llevar a cabo un análisis cinemático del robot.
- Crear una interfaz gráfica en el software MATLAB para el estudio de la cinemática del robot.
- Dotar a la célula robótica de unas herramientas y lugar de trabajo adecuados para su funcionamiento.
- Elaborar un protocolo de utilización adecuado del robot en el ámbito docente.
- Diseñar una serie de prácticas para la asignatura “Mecánica de robots” del Grado en Ingeniería Mecánica.

3 INTRODUCCIÓN

La robótica posee un carácter interdisciplinar ya que aborda temas referentes a la teoría de control, la mecánica, la electrónica, el álgebra y la informática, entre otras. El robot de este estudio es un equipo de nueva dotación del área de Ingeniería Mecánica del departamento de Ingeniería Mecánica y Minera de la Escuela Politécnica Superior de Linares. Su objetivo es acercar la robótica al alumnado del Grado en Ingeniería Mecánica, en especial para la formación de cinemática en tres dimensiones en la asignatura de Mecánica de Robots, de una forma práctica y visual.

La robótica puede ser explicada de dos formas diferentes, una es centrándose en el diseño del robot, basándose en la geometría y las matrices de movimiento que necesita para su funcionamiento y otra es centrándose en el uso del mismo robot por el usuario, explicando el manejo del robot y su programación de una forma práctica.

Es interesante observar que en nuestra época los ciudadanos tienen diferentes ideas respecto al uso y funcionalidad del concepto “robot”. Los robots se han dado a conocer en tiempos pasados mediante la literatura en obras de ficción, representándose como androides autómatas con forma humana. Liberados de convencionalismos sobre el termino, los avances tecnológicos en las últimas décadas nos han acercado a una interpretación más común de lo que significa el concepto de robot, no ya industrial sino incluso doméstico, se trata de un automatismo que nos libera de trabajo pudiendo realizar tareas antes reservadas solo a los humanos.

Por lo tanto, debido a sus múltiples usos, la definición de un robot debe estar acompañada de un adjetivo que describa su función: robot aéreo, robot submarino, robot de limpieza, robot de cocina, robot industrial, etc.

Figura 3.1. Robot aéreo, Robot de limpieza y Robot submarino. [22,23,24]

En este trabajo se tiene el concepto de robot como un manipulador industrial multifuncional y reprogramable, centrándose el autor en el aspecto cinemático y de programación. Este tipo de robot industrial es el más utilizado hasta la fecha, destinado a la fabricación flexible en líneas de producción.

Sin embargo, la robótica en sí empezó mucho antes de que se creara el primer manipulador industrial.

3.1 Antecedentes históricos

El ser humano siempre ha sentido fascinación por máquinas y dispositivos capaces de imitar sus funciones y movimientos, por lo que la realización de mecanismos animados con palancas, engranajes, etc. ha sido una constante desde la antigüedad.

El autómata más antiguo que se conserva en la actualidad es el Gallo de Estrasburgo [1] (Fig 3.2.a.), de 1350, que formaba parte del reloj de la torre de la catedral, y al dar las horas movía las alas, el pico y cacareaba. Sin embargo, se tiene constancia de que existían autómatas de este estilo mucho tiempo atrás. Por ejemplo, se tienen datos de que Dédalo [2] construyó estatuas que se movían solas, y Herón de Alejandría [2] (Siglo I A.C) describe dispositivos en forma de aves que vuelan, gorjean y beben (Fig 3.2.b).

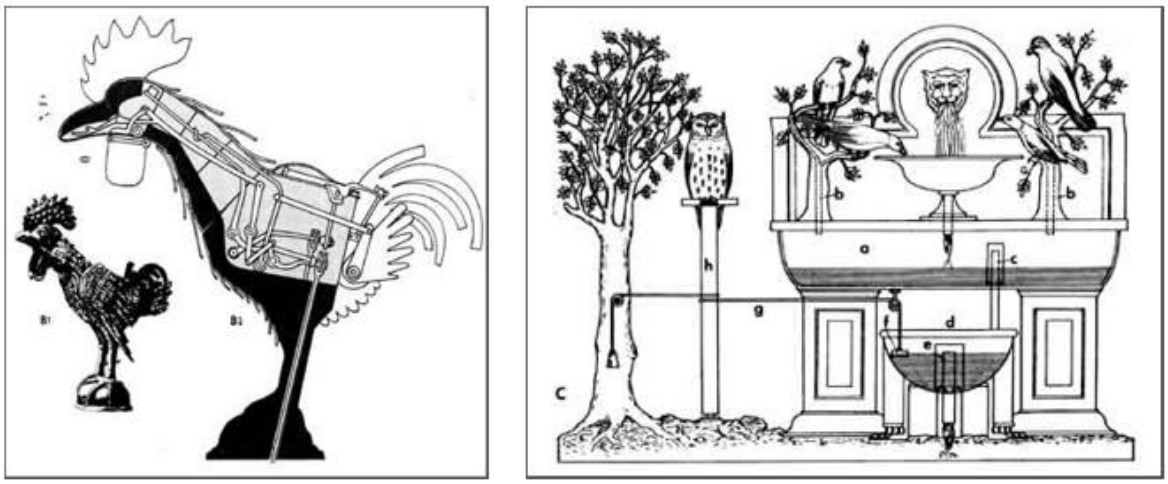


Figura 3.2. a) Gallo de Estrasburgo [1] y b) Fuente de pájaros Cantores de Herón [2].

Más adelante serían los árabes, grandes herederos de la cultura griega, quienes le darían una aplicación práctica construyendo dispensadores automáticos de agua mediante el uso de levas, resortes y el tornillo sin fin, dispositivos ya inventados por Arquímedes (287-212 A.C) [17].

Hasta la revolución industrial se desarrollaron multitud de mecanismos basados en engranajes y similares, en la mayoría de casos realizados por artesanos relojeros y sin utilidad más allá del puro entretenimiento.

La evolución de los manipuladores industriales no ha sido tan espectacular como la de los robots, la cual empezó para manejar elementos radioactivos sin riesgo para el ser humano alrededor de 1948.

Más adelante, gracias a un ingeniero llamado G.C. Devol [2], se diseñó el sistema de un ordenador que controlaba los movimientos de un brazo mecánico. Esta máquina se denominó “robot industrial”, dando así comienzo a la robótica dedicada a la industria. Las primeras aplicaciones industriales datan de 1967 en las que se incorporaron robots a las cadenas de producción de carrocerías de automóviles. Estos robots cumplían diferentes funciones como pueden ser: pintura, montaje, empaque, soldadura, etc.

3.2 Definición de robot industrial

Según la Asociación Internacional de Estándares (ISO) [18] se define Robot manipulador industrial como: *Manipulador de 3 o más ejes, con control automático reprogramable, multiaplicación, móvil o no, destinado a ser utilizado en aplicaciones de automatización industrial. Incluye al manipulador (sistema mecánico y accionadores) y al sistema de control (software y hardware de control y potencia).*

Figura 3.3. Robots industriales a) IRB 120 de ABB [12], b) KR 6 R700 fivve (KR AGILUS) de la marca KUKA [19]

Con esta descripción y las muchas otras, con ligeras modificaciones, adoptadas por diferentes organizaciones como pueden ser la *Organización Internacional de Estándares (ISO),* la *Asociación Francesa de Normalización (AFNOR*) y por último la *Federación Internacional de Robótica (IFR*), recogidas en la referencia [1], se puede

obtener una idea de a qué se refiere el concepto de robot industrial. Dichas definiciones dejan claro la aceptación de que se trata de un brazo mecánico con uno o varios grados de libertad, capaz de manipular materiales y herramientas con cierto control sobre sus movimientos.

3.3 Clasificación del robot industrial

Según el apartado 1.4 de la referencia [1], un robot puede clasificarse atendiendo a diferentes criterios o características.

3.3.1 Clasificación atendiendo a la Generación

La generación de un robot hace referencia al momento tecnológico en que este aparece. De este modo, se puede considerar que se pasa de una generación a la siguiente cuando da un hito que supone un avance significativo en las capacidades de los robots.

- Robots de primera generación: Repite la tarea programada secuencialmente. No toma en cuenta las posibles alteraciones de su entorno.
- Robots de segunda generación: Adquiere información limitada de su entorno y actúa en consecuencia. Puede localizar, clasificar (visión) y detectar esfuerzos y adaptar sus movimientos en consecuencia.
- Robots de tercera generación: Su programación se realiza mediante el empleo de un lenguaje natural. Posee capacidad para planificación automática de tareas.

3.3.2 Clasificación atendiendo al Área de Aplicación

Desde el punto de vista del uso que se da al robot es posible clasificarlos bien en base al sector económico en el que se encuentran trabajando o bien en base al tipo de aplicación o tarea que desarrollan, independientemente de en qué sector económico trabajen. La IFR realiza una clasificación exhaustiva en las referencias [4] y [20] en las que se clasifican según el sector, la categoría y la aplicación que realizan (Tabla 3.1).

000	Sin especificar.
110	Manipulación en fundición.
130	Manipulación en moldeo de
140	Manipulación en tratamientos
150	Manipulación en la forja y
160	Soldadura.
170	Aplicación de materiales.
180	Mecanización.
190	Otros procesos.
200	Montaje.
210	Paletización y empaquetado.
220	Medición, inspección, control de
230	Manipulación de materiales.
240	Formación, enseñanza e
900	Otros.

Tabla 3.1. Clasificación de las aplicaciones de robots industriales manipuladores, según IFR [20]

Los robots de servicio se clasifican entre ellos según el servicio al que está dirigido y la interacción que necesitan con los humanos. Aunque la clasificación más práctica se hace en base a la aplicación (Tabla 3.2) dividiéndose estos en Robots personales y domésticos, Robots de servicios profesionales, y Robots dedicados a aplicaciones en I+D.

Sección I	**Robots personales y domésticos**
1-5	Robots para tareas domesticas
6-0	Robots de entretenimiento
11-14	Asistencias , ayuda a discapacidades
15	Transporte Personal
16	Seguridad y vigilancia de la vivienda
17	Otros usos personales y domésticos
Sección II	**Robots de servicios profesionales**
18-23	Robots de exteriores
24-28	Limpieza profesional
29-31	Sistemas de inspección
32-36	Construcción y demolición
37-40	Sistemas logísticos
41-44	Medicina
45-50	Defensa, rescate y seguridad
51	Submarinos
52	Plataformas móviles de uso general
53-55	Robots de laboratorio
56-59	Relaciones publicas
60-61	Propósito especial
62	Humanoides
63	Robots a medida
64	Otros no especificados
Sección III	**I+D en robótica**
64	Percepción
65-67	Actuación
68	Micro y nano robots
69	Arquitecturas e integración
70	Navegación y control
71	Interfaces con usuario y otras
72	Otras actividades de I+D no especificadas
73	Investigación básica

Tabla 3.2. Clasificación de los robots de servicio por Áreas de aplicación según IFR [20]

3.3.3 Clasificación atendiendo al tipo de Actuadores

Dependiendo de cuál sea el tipo de energía utilizada por los ejes principales del robot, este puede ser clasificado como:

- Robot Neumático.
- Robot Hidráulico.
- Robot Eléctrico.

3.3.4 Clasificación atendiendo al Tipo de Control.

El tipo de control utilizado también se emplea para clasificar a los robots en:

- Robot secuencial: Los movimientos son generados eje por eje por un sistema de control. Al finalizar el movimiento de un eje comienza el siguiente.
- Robot controlado por trayectoria: Los ejes del robot se mueven de forma simultánea controlados por un sistema de control que les marca la trayectoria a seguir por el extremo.
- Robot adaptativo: Este tipo de control se ayuda de sensores que definen los movimientos a realizar.
- Robot teleoperado: Este robot se dirige remotamente por un operador humano.

3.3.5 Clasificación atendiendo al Número de Ejes.

Esta clasificación es solo aplicable a robots compuestos de eslabones unidos en una cadena cinemática. Según la definición ISO, el manipulador industrial debe tener al menos 3 ejes, sin embargo, para posicionar y orientar en cualquier posición el extremo de un brazo robótico es necesario que disponga al menos de 6 grados de libertad, 6 parámetros para definir el punto, tres de posición y tres de orientación. Es por esto que en la práctica la mayor parte de robots tienen 6 ejes, seguidos por los de 4.

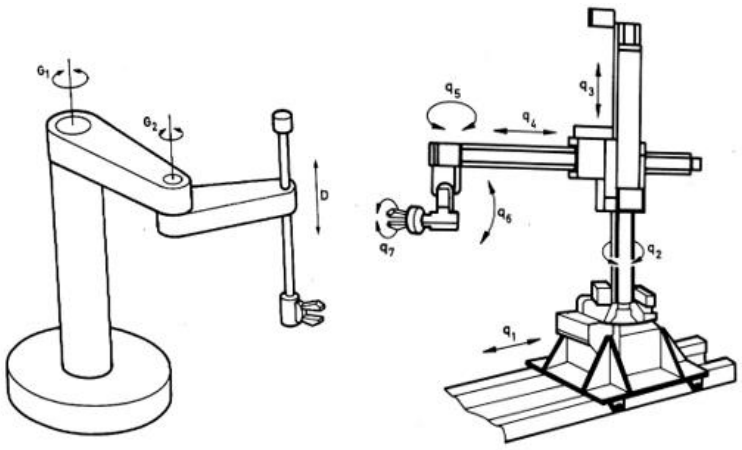


Figura 3.4. Ejemplo de configuración SCARA (3 GDL) [2] y robot dotado de movilidad traslacional (6GDL) [2].

Los robots con más de 6 ejes se llaman robots redundantes y suelen ser poco frecuentes, suelen estar dedicados a manipulación en lugares de difícil acceso o para añadir el robot a una guía móvil en la que desplazarse.

3.3.6 Clasificación atendiendo a la Configuración

La mayoría de robots industriales están formados por cadenas cinemáticas de varios eslabones, los cuales se unen entre ellos con diferentes configuraciones y articulaciones. El empleo de diferentes combinaciones de estas nos proporciona un abanico de posibilidades a la hora de clasificar un robot siendo las más habituales la configuración cartesiana, cilíndrica, esférica, angular y SCARA (Fig 3.5).

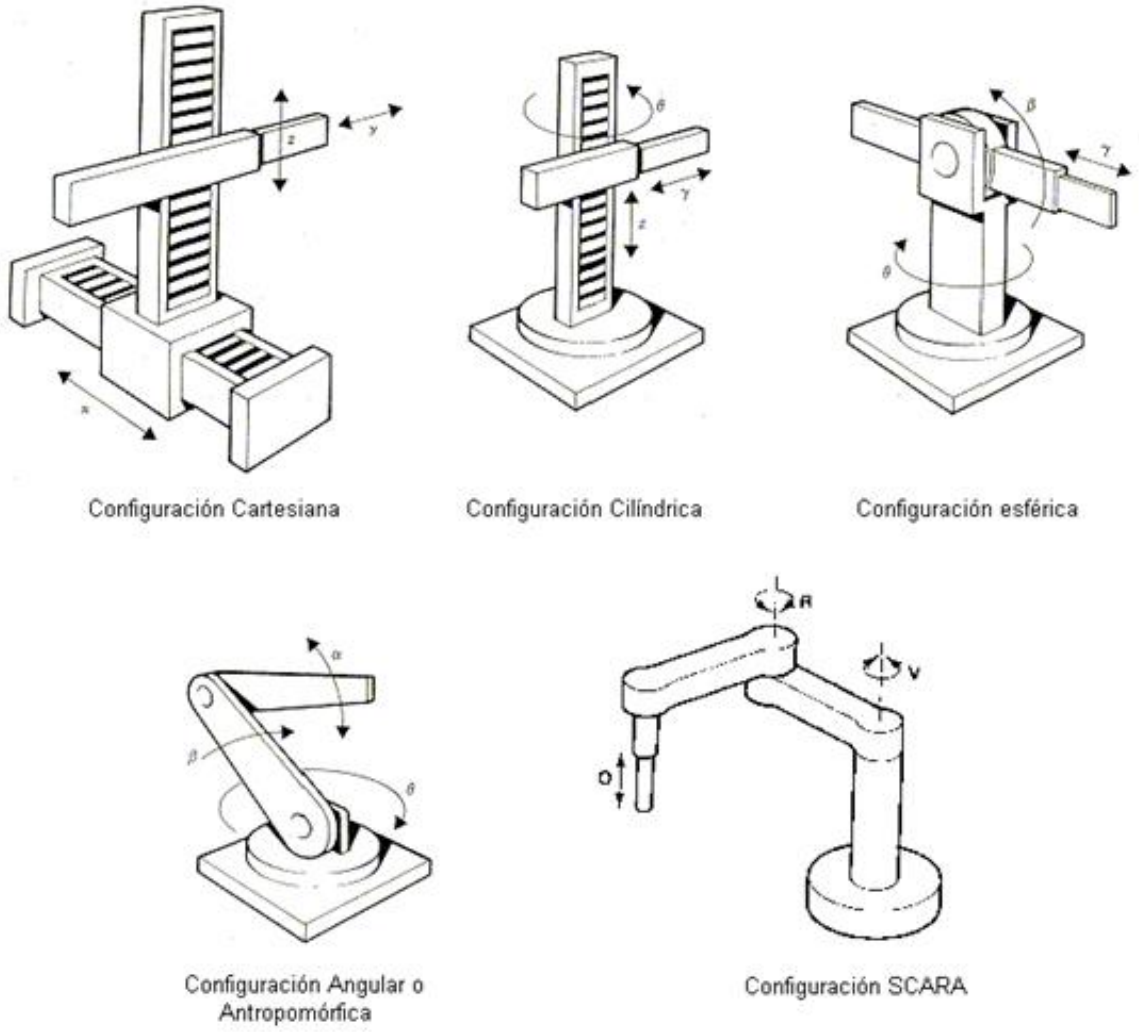


Figura 3.5. Tipos de configuraciones de robots más frecuentes [1].

Estas configuraciones hacen uso de distintas combinaciones de articulaciones (Fig 3.6), siendo las más utilizadas las que disponen de un solo grado de libertad como son la prismática y de rotación.

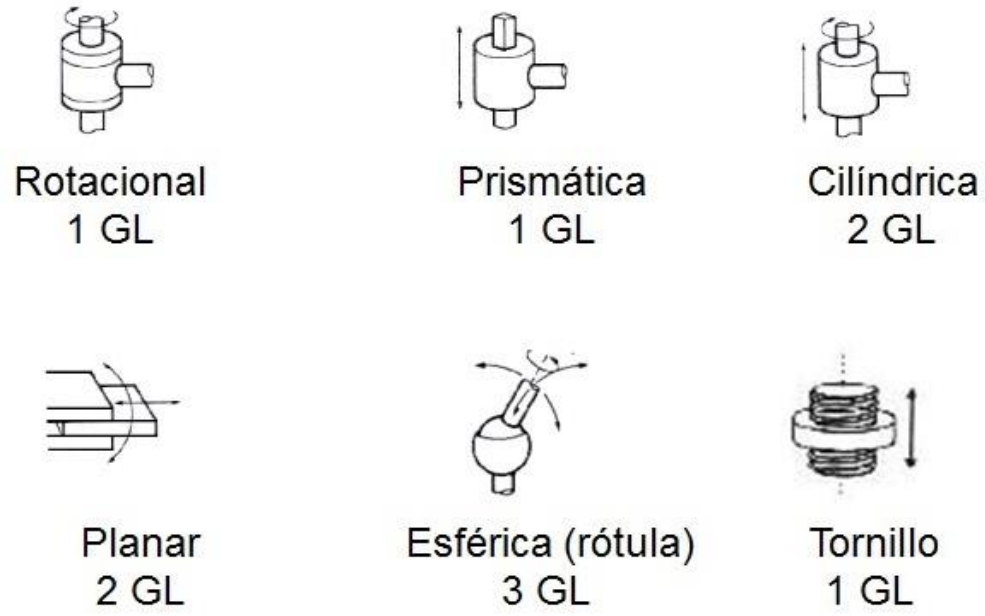


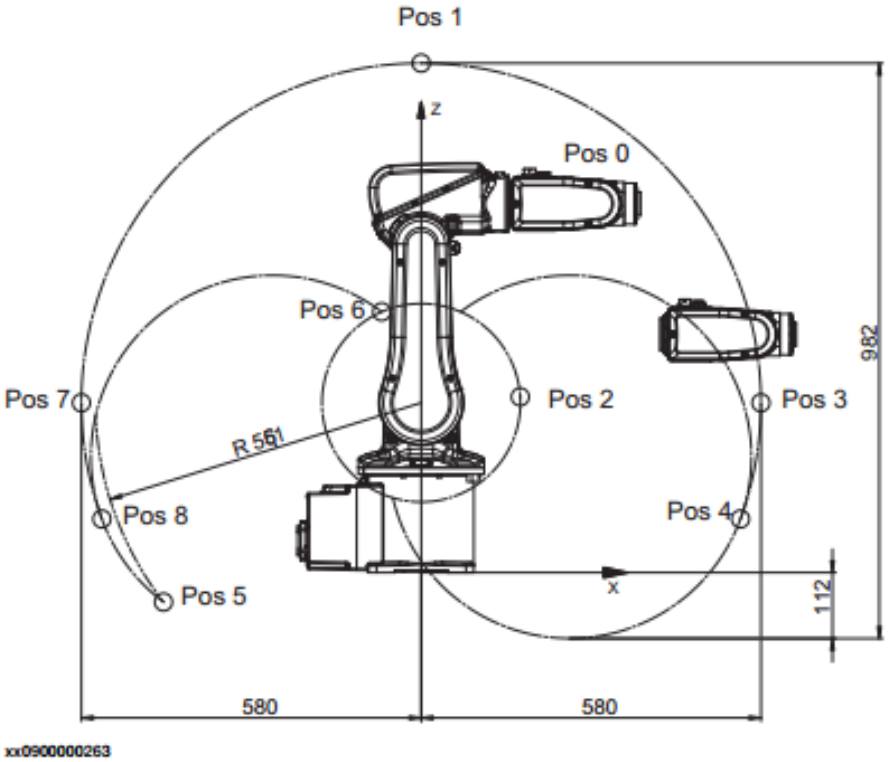
Figura 3.6. Tipos de articulaciones más frecuentes [1].

Dependiendo de estas configuraciones, el robot dispondrá de un área de trabajo diferente (Fig 3.5). El área de trabajo o campo de acción es el volumen espacial al que puede llegar el extremo del robot. Este volumen está determinado por el tamaño, forma y tipo de los eslabones que integran el robot, así como por las limitaciones de movimiento impuestas por el sistema de control.

Nunca deberá utilizarse el efector colocado en la muñeca para la obtención del espacio de trabajo, ya que se trata de un elemento añadido al robot, y en el caso de variar el efector el área de trabajo se tendría que calcular de nuevo.

En los catálogos suministrados por los fabricantes (Fig 3.7) se suele indicar el área de trabajo mediante un dibujo acotado. Cuando la información es de tipo numérico, el área de trabajo se indica mediante el rango de recorrido de cada articulación.

El robot debe elegirse de modo que su área de trabajo (o campo de acción) le permita llegar a todos los puntos necesarios para llevar a cabo su tarea.



Posición	Posición en el centro de la muñeca (mm)		Ángulo (grados)	
	X	Z	Eje 2	Eje 3
A	302 mm	630 mm	0°	0°
B	0 mm	870 mm	0°	-77°
C	169 mm	300 mm	0°	+70°
D	580 mm	270 mm	+90°	-77°
E	545 mm	91 mm	+110°	-77°
F	-440 mm	-50 mm	-110°	-110°
G	-67 mm	445 mm	-110°	+70°
H	-580 mm	270 mm	-90°	-77°
J	-545 mm	91 mm	-110°	-77°

Figura 3.7. Área de trabajo de robot IRB 120 de la marca ABB [9]

El que el robot pueda acceder a todo el espacio de trabajo no significa que lo pueda hacer con cualquier orientación. Existirán un conjunto de puntos, los más alejados y los más cercanos, que únicamente se podrán acceder con unas orientaciones determinadas, mientras que otros puntos admitirán cualquier orientación.

3.4 Morfología

Como se ha visto en el apartado anterior, un robot está formado por varios eslabones los cuales se unen mediante articulaciones con distintos grados de libertad. Según la disposición de dichos eslabones y del tipo de articulación se obtiene una configuración diferente para cada robot.

Los componentes de un robot industrial según su función, son los siguientes:

- Brazo: Realizar los grandes movimientos.
- Muñeca: adopta la orientación deseada.
- Elemento terminal: Mano, pinza o herramientas.

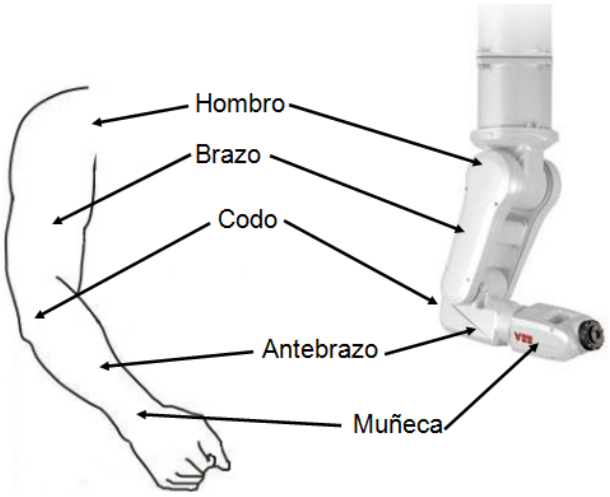


Figura 3.8. Morfología de brazo robótico industrial, similitud con brazo humano.

3.5 Elementos terminales

Los elementos terminales, ubicados en el extremo del mecanismo del robot, realizan la tarea concreta que se pretende: pegar, pintar, soldar, coger, etc.

Se dividen según la siguiente clasificación [2]:

3.5.1 Elementos de sujeción

Los elementos de sujeción más comunes son las llamadas “pinzas”. Estas pueden ser de accionamiento eléctrico o neumático. El uso de diferentes modelos de pinzas puede deberse a múltiples factores: la necesidad de una fuerza de agarre precisa mediante accionamiento neumático, la precisión en la distancia de agarre de una pinza con accionamiento eléctrico, agarres angulares o rectos de los dedos de la pinza, el número y la morfología de los dedos.

Normalmente para aplicaciones en las que la pinza debe agarrar un solo objeto, esta debe estar diseñada para maximizar la superficie de agarre y facilitar el movimiento.

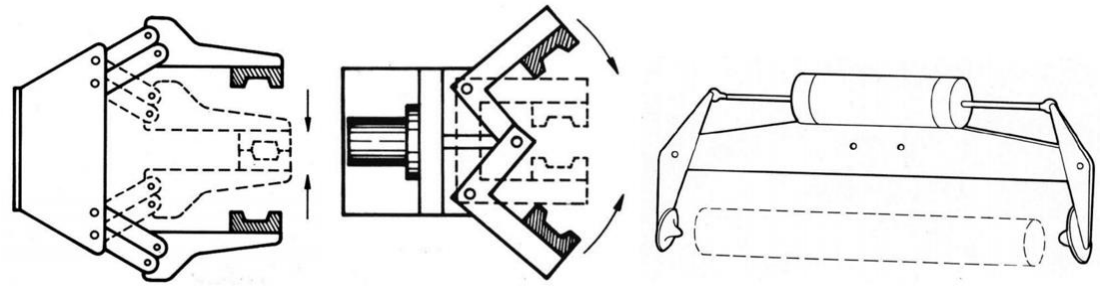


Figura 3.9. Ejemplo de distintos tipos de pinzas [2].

También es muy frecuente el uso de ventosas, las cuales usan el efecto Venturi o Coanda para crear el vacío entre la pinza y la ventosa (Fig 3.10).

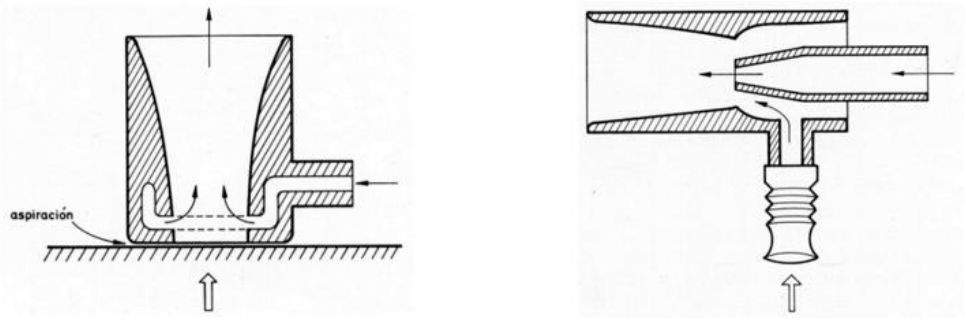


Figura 3.10. Modos de crear vacío [2]: a) Efecto Coanda, b) Efecto Venturi

Para piezas metálicas es frecuente ver también como herramienta de sujeción el uso de electroimanes. Esto se suele usar normalmente para piezas de pequeño tamaño las cuales no pueden ser agarradas mediante pinzas.

3.5.2 Herramientas terminales

Los robots industriales se utilizan habitualmente para operaciones de soldadura, pintura o mecanizado. Para estos casos es necesario que el robot disponga como elemento terminal la herramienta adecuada para la operación, la cual se suele unir de forma rígida a la muñeca del robot. Existen multitud de herramientas según la operación a realizar. En las referencias [2] y [5] puede encontrarse un resumen de las distintas herramientas y pinzas para robots.

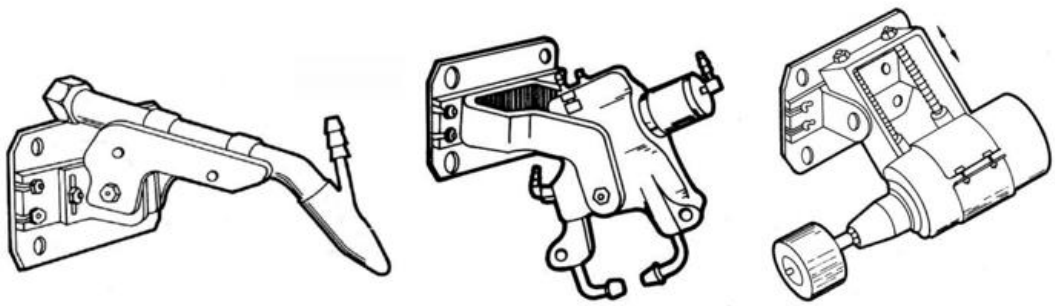


Figura 3.11. Herramienta para soldadura por arcos, por puntos y mecanizado [2].

4 FUNDAMENTOS TEÓRICOS

En este capítulo se presentan los fundamentos teóricos utilizados para la realización del trabajo, centrándose en las ecuaciones y métodos cinemáticos utilizados, además de un resumen de la teoría y tipos de programación de robots industriales.

4.1 Cinemática de robots

En este apartado se explican los conocimientos necesarios para definir la cinemática de un robot, esto se consigue referenciando las velocidades y posiciones del robot respecto a un sistema de coordenadas. Se empezará explicando qué es un sistema de coordenadas y cómo se relaciona con las distintas posiciones del robot. Después se realizará un estudio de la formulación para conseguir referenciar los distintos sistemas de coordenadas. Más tarde se estudia la relación de velocidades de las articulaciones con las velocidades del extremo mediante formulación matricial.

4.1.1 Sistema de coordenadas

Un sistema de coordenadas define un sistema coordenado bidimensional o tridimensional, partiendo de un punto fijo conocido como origen. Los objetivos y las posiciones del robot se localizan mediante medidas a lo largo de los ejes de los sistemas de coordenadas.

Los robots utilizan varios sistemas de coordenadas, cada uno de ellos adecuado para tipos concretos de movimientos o programaciones.

- Sistema de coordenadas mundo: Tiene su punto cero en un punto fijo de la célula de trabajo y puede ser utilizado para varios robots a la vez.
- Sistema de coordenadas de la base: El sistema de coordenadas de la base tiene su punto cero en la base del robot, lo que resulta útil a la hora de mover el robot de posición. Suele coincidir con el sistema de coordenadas mundo cuando solo existe un robot en la célula de trabajo y este se mantiene inmóvil.
- Sistema de coordenadas del objeto de trabajo: A la hora de programar un robot, suele ser más adecuado este sistema de referencia ya que corresponde al plano de trabajo o pieza en el que el robot desempeña su función. Un robot puede tener varios sistemas de objeto diferentes dado que puede trabajar con distintas piezas y en distintos planos.

  Al reposicionar el objeto de trabajo en la estación de trabajo, solo es necesario cambiar la posición del sistema de coordenadas para que las trayectorias se actualicen a la vez.

- TCP(Sistema de coordenadas de la herramienta): Tiene su origen en el punto central de la herramienta seleccionada, definiendo la posición y la orientación de la misma.

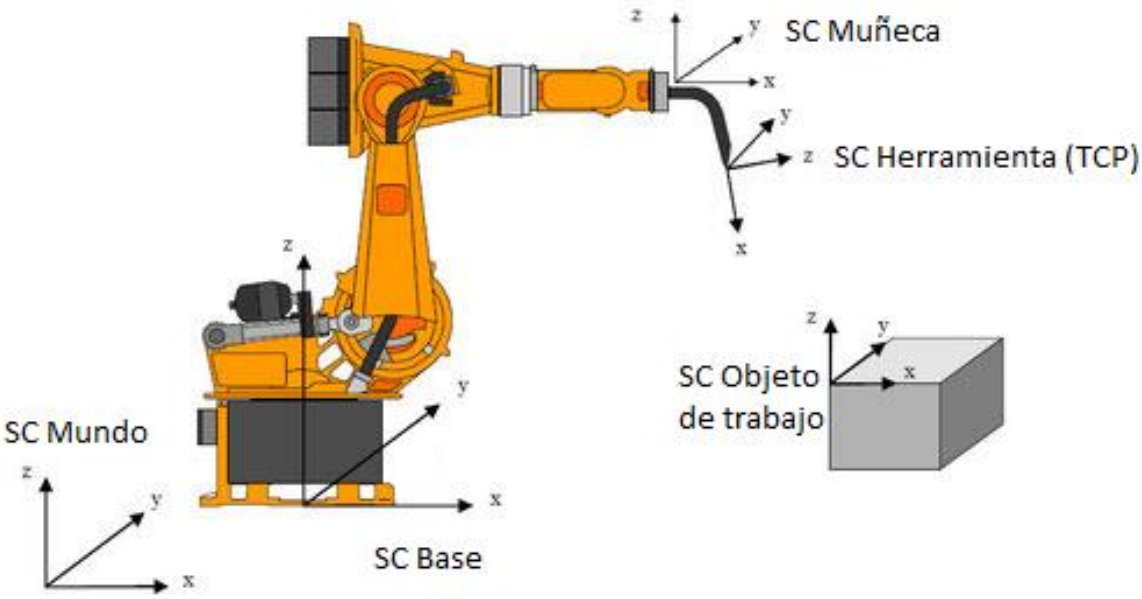


Figura 4.1. Sistemas de coordenadas de Robot Industrial.

4.1.2 Matrices de transformación homogénea

Se han visto los distintos sistemas de coordenadas y su ubicación y función para el robot, sin embargo para la definición de un sistema de coordenadas es necesario la utilización de matrices homogéneas de posición y rotación.

Se define matriz de transformación homogénea **T** a una matriz que expresa la posición y orientación de un sistema de coordenadas respecto a otro. Esta matriz está compuesta por 4 submatrices de distinto tamaño que representan la rotación, translación, perspectiva y escalado.

$$\mathbf{T} = \begin{bmatrix} \text{Rotación} & \text{Translación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix} = \begin{bmatrix} \mathbf{R_{3x3}} & \mathbf{p_{3x1}} \\ \mathbf{f_{1x3}} & w_{1x1} \end{bmatrix} \qquad \text{(Ec 4.1)}$$

- $\mathbf{R_{3x3}}$: Matriz de rotación.
- $\mathbf{P_{3x1}}$: Vector de translación.
- $\mathbf{f_{1x3}}$: Vector de perspectiva (se emplea en visión artificial). Para robótica **f**=(0,0,0).
- w_{1x1}: Factor de escala. Si no se indica lo contrario, siempre se tomará *w* =1.

4.1.3 Problema cinemático directo

El problema cinemático directo consiste en la representación de la orientación y posición del extremo final del robot conociendo las variables que definen sus articulaciones. En un caso de robot angular serían los ángulos girados por cada uno de sus ejes. Para ello se puede utilizar las matrices de transformación homogéneas (Ec.4.1).

Aunque para describir la relación que existe entre dos elementos contiguos se puede hacer uso de cualquier sistema de referencia ligado a cada elemento, la forma habitual que se suele utilizar en robótica es la representación de Denavit Hartenberg (D-H). Denavit y Hartenberg [6] propusieron en 1955 un método matricial que establece la localización que debe tomar cada sistema de coordenadas {$\mathbf{S_i}$} ligado a cada eslabón **i** de una cadena articulada, para poder sistematizar la obtención de las ecuaciones cinemáticas de la cadena completa.

Escogiendo los sistemas de coordenadas asociados a cada eslabón según la representación propuesta por D-H [6] será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Hay que hacer notar que si bien una matriz de transformación homogénea queda definida por 6 grados de libertad, el método de Denavit-Hartenberg, permite, en eslabones rígidos, reducir este a 4 grados de libertad con la correcta elección de los sistemas de coordenadas.

Estas 4 transformaciones básicas consisten en una sucesión de rotaciones y translaciones que permiten relacionar el sistema de referencia del elemento *i-1* con el sistema del elemento *i*. Las transformaciones en cuestión son las siguientes (es importante recordar que el paso del sistema {$\mathbf{S_{i-1}}$} al {$\mathbf{S_i}$} mediante estas 4 transformaciones está garantizado solo si los sistemas {$\mathbf{S_{i-1}}$} y {$\mathbf{S_i}$ } han sido definidos de acuerdo a unas normas determinadas (Fig 4.2)):

1. Rotación alrededor del eje $\mathbf{Z_{i-1}}$ un ángulo $\boldsymbol{\theta_i}$.
2. Traslación a lo largo de $\mathbf{Z_{i-1}}$ una distancia d_i; vector $\mathbf{d_i}$ (0, 0, d_i).
3. Traslación a lo largo de $\mathbf{X_i}$ una distancia a_i; vector $\mathbf{a_i}$ (0, 0, a_i).
4. Rotación alrededor del eje $\mathbf{X_i}$ un ángulo α_i.

Dado que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene que:

$$^{\mathrm{i}-\mathbf{1}}\mathbf{A_i} = \mathbf{Rotz}(\theta_i)\ \mathbf{T}(0,0,d_i)\ \mathbf{T}(0,0,\mathrm{a_i})\ \mathbf{Rotx}(\alpha_i) \qquad \text{(Ec 4.2)}$$

Y realizando el producto entre matrices se obtiene que:

$$\mathbf{^{i-1}A_i} = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \mathrm{a}_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & \mathrm{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(Ec 4.3)}$$

$$= \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & \mathrm{a}_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & \mathrm{a}_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & \mathrm{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde θ_i, d_i ,a_i ,α_i son los parámetros D-H del eslabón *i*. De este modo, basta con identificar los parámetros θ_i, d_i ,a_i ,α_i para obtener las matrices $^{i-1}\mathbf{A_i}$ y relacionar así todos y cada uno los eslabones del robot.

Como se ha indicado, para que la matriz $^{i-1}\mathbf{A_i}$ relacione los sistemas {$\mathbf{S_{i-1}}$} y {$\mathbf{S_i}$} es necesario que los sistemas se hayan escogido de acuerdo a unas determinadas normas. Estas normas junto con la definición de los cuatro parámetros de Denavit Hartenberg, conforman el siguiente el algoritmo para la resolución del problema cinemático directo que se comenta a continuación:

1) Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con *n* (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.
2) Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en *n*.
3) Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
4) Para *i* de 0 a *n-1* situar el eje $\mathbf{z_i}$ sobre el eje de la articulación *i+1*.
5) Situar el origen del sistema de la base {$\mathbf{S_0}$} en cualquier punto del eje $\mathbf{z_0}$. Los ejes $\mathbf{x_0}$ e $\mathbf{y_0}$ se situarán de modo que formen un sistema dextrógiro con $\mathbf{z_0}$.
6) Para *i* de 1 a *n-1*, situar el sistema {$\mathbf{S_i}$} (solidario al eslabón *i*) en la intersección del eje $\mathbf{z_i}$ con la línea normal común a $\mathbf{z_{i-1}}$ y $\mathbf{z_i}$. Si ambos ejes se cortasen se situaría {$\mathbf{S_i}$} en el punto de corte. Si fuesen paralelos {$\mathbf{S_i}$} se situaría en la articulación *i+1*.

7) Para *i* de 1 a *n-1*, situar $\mathbf{x_i}$ en la línea normal común a $\mathbf{z_{i-1}}$ y $\mathbf{z_i}$.
8) Para *i* de 1 a *n-1*, situar $\mathbf{y_i}$ de modo que forme un sistema dextrógiro con $\mathbf{x_i}$ y $\mathbf{z_i}$.
9) Situar el sistema $\{\mathbf{S_n}\}$ en el extremo del robot de modo que $\mathbf{z_n}$ coincida con la dirección de $\mathbf{z_{n-1}}$ y $\mathbf{x_n}$ sea normal a $\mathbf{z_{n-1}}$ y $\mathbf{z_n}$.
10) Obtener θ_i como el ángulo que hay que girar en torno a $\mathbf{z_{i-1}}$ para que $\mathbf{x_{i-1}}$ y $\mathbf{x_i}$ queden paralelos.
11) Obtener d_i como la distancia, medida a lo largo de $\mathbf{z_{i-1}}$, que habría que desplazar $\{\mathbf{S_{i-1}}\}$ para que $\mathbf{x_i}$ y $\mathbf{x_{i-1}}$ quedasen alineados.
12) Obtener a_i como la distancia medida a lo largo de $\mathbf{x_i}$, que ahora coincidiría con $\mathbf{x_{i-1}}$, que habría que desplazar el nuevo $\{\mathbf{S_{i-1}}\}$ para que su origen coincidiese con $\{\mathbf{S_i}\}$.
13) Obtener α_i como el ángulo que habría que girar en torno a $\mathbf{x_i}$, que ahora coincidiría con $\mathbf{x_{i-1}}$, para que el nuevo $\{\mathbf{S_{i-1}}\}$ coincidiese totalmente con $\{\mathbf{S_i}\}$.
14) Obtener las matrices de transformación $^{i-1}\mathbf{A_i}$.
15) Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $\mathbf{T} = {}^{0}\mathbf{A_1} \cdot {}^{1}\mathbf{A_2} \cdot {}^{n-1}\mathbf{A_n}$.
16) La matriz **T** define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las *n* coordenadas articulares.

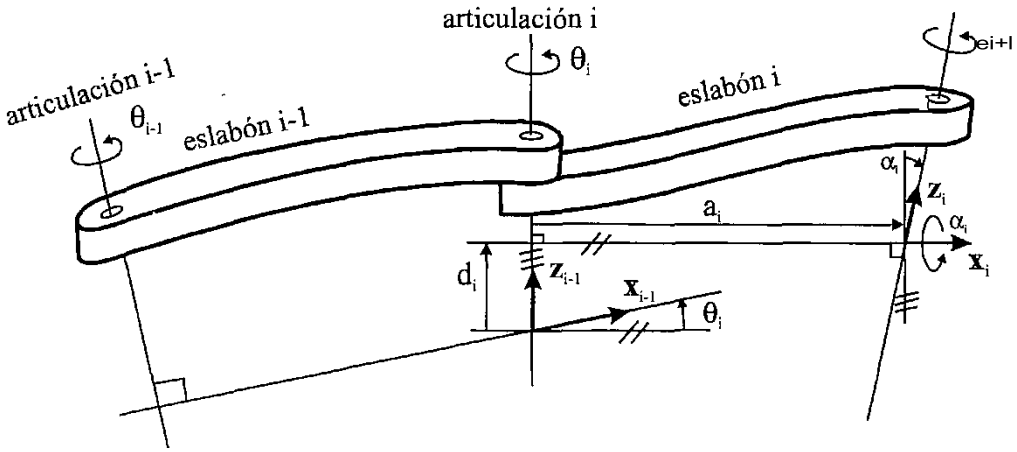


Figura 4.2. Parámetros de D-H para un eslabón giratorio [1].

Los cuatro parámetros de D-H (θ_i, d_i ,a_i ,α_i) dependen únicamente de las características de geométricas de cada eslabón y de las articulaciones que las unen con el anterior y el siguiente.

- θ_i **:** Es el ángulo que forman los ejes $\mathbf{X_{i-1}}$ y $\mathbf{X_i}$ medido en un plano perpendicular al eje $\mathbf{Z_{i-1}}$, utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.
- d_i **:** Es la distancia a lo largo del eje $\mathbf{Z_{i-1}}$ desde el origen del sistema de coordenadas (i-1)ésimo hasta la intersección del eje $\mathbf{Z_{i-1}}$ con el eje $\mathbf{X_i}$. Se trata de un parámetro variable en articulaciones prismáticas.
- a_i **:** Es la distancia a lo largo del eje $\mathbf{X_i}$ que va desde la intersección del eje $\mathbf{Z_{i-1}}$ con el eje $\mathbf{X_i}$ hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes $\mathbf{Z_{i-1}}$ y $\mathbf{Z_i}$.
- α_i **:** Es el ángulo de separación del eje $\mathbf{Z_{i-1}}$ y el eje $\mathbf{Z_i}$, medido en un plano perpendicular al eje $\mathbf{X_i}$, utilizando la regla de la mano derecha.

Una vez obtenidos los parámetros D-H, el cálculo de las relaciones entre los eslabones consecutivos de robot es inmediato, ya que viene dado por las matrices $\mathbf{^{i-1}A_i}$ **,** que se calculan según la expresión general (Ec 4.3). Las relaciones entre varios eslabones consecutivos dos a dos vienen dadas por las matrices **T**, que se obtienen como producto de un conjunto de matrices **A**.

Obtenida la matriz **T**, ésta expresará la orientación y posición del extremo del robot en función de sus coordenadas articulares, con lo que quedará resuelto el problema cinemático directo.

4.1.4 Cuaternios

Un cuaternio es un vector, el cual se utiliza como otro método para definir la orientación de un sistema de coordenadas [1].

Un cuaternio **Q** está constituído por cuatro componentes: **Q**= (q_1 q_2 q_3 q_4) que representan las coordenadas del cuaternio en una base {**e,i,j,k**}.

Frecuentemente se denomina parte escalar *s* del cuaternio a la componente según *e*, es decir, q_1, y parte vectorial $\vec{v}$ al resto de componentes de manera que un cuaternio se puede representar como:

$$\mathbf{Q} = [q_1, q_2, q_3, q_4] = [s, \vec{\mathbf{v}}] \qquad \text{(Ec 4.4.)}$$

Una matriz rotacional describe la dirección de los ejes del sistema de coordenadas respecto a un sistema de referencia.

Esto significa que el componente *x* del vector *X* en el sistema de coordenadas de referencia será x_1, el componente *y* será x_2, etc.

X=(x_1,x_2,x_3)

Y=(y_1,y_2,y_3)

Z=(z_1,z_2,z_3)

Estos tres vectores pueden ser puestos juntos en una matriz rotacional, donde cada uno de los vectores forma una columna.

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

Un cuaternio es solo una forma más concisa de referirse a esta matriz de rotación. Los cuaternios se calculan partiendo de los elementos de la matriz de rotación.

$$q_1 = \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2}$$

$$q_2 = \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2} \quad signo\ q_2 = signo(y_3 - z_2)$$

$$q_3 = \frac{\sqrt{-x_1 + y_2 - z_3 + 1}}{2} \quad signo\ q_3 = signo(z_1 - x_3) \qquad \text{(Ec 4.5)}$$

$$q_4 = \frac{\sqrt{-x_1 - y_2 + z_3 + 1}}{2} \quad signo\ q_4 = signo(x_2 - y_1)$$

4.1.5 Problema cinemático inverso

El problema cinemático inverso consiste en encontrar los valores que deben tener las coordenadas articulares (variables de nudo) para que su extremo se posicione y oriente según una determinada localización espacial. Este planteamiento es mucho más dificultoso que el problema cinemático directo debido a que puede que no haya una solución para todos los puntos del espacio, para ello el punto debe estar dentro del área de trabajo. Además, si hubiera solución puede que esta no fuera única, ya que como puede verse en la figura 4.4 se puede llegar a una misma posición del extremo mediante distintas configuraciones de las articulaciones.

Si se conoce la matriz de transformación homogénea del brazo robot **T**, que a partir de las coordenadas articulares proporciona la posición del extremo, es lógico pensar que podría manipularse la matriz para obtener la relación inversa. En la práctica, esta operación es compleja sobre todo para robots con determinados GDL. En un robot de 6 ejes, manipular la matriz implica trabajar con 12 ecuaciones, pero en realidad tenemos solo 6 incógnitas, por lo que habrá que tener cuidado al seleccionar las ecuaciones y que no sean redundantes.

Figura 4.3. Diagrama de relación entre cinemática directa e inversa.

Existen tres métodos para la resolución del problema cinemático inverso:

- Método geométrico:

  Este método es muy apropiado para robots con pocos grados de libertad (hasta 3) debido a que hace uso de relaciones geométricas y trigonométricas entre los eslabones del robot. Sin embargo, esto resulta más complicado cuantos más grados de libertad tenga el robot.

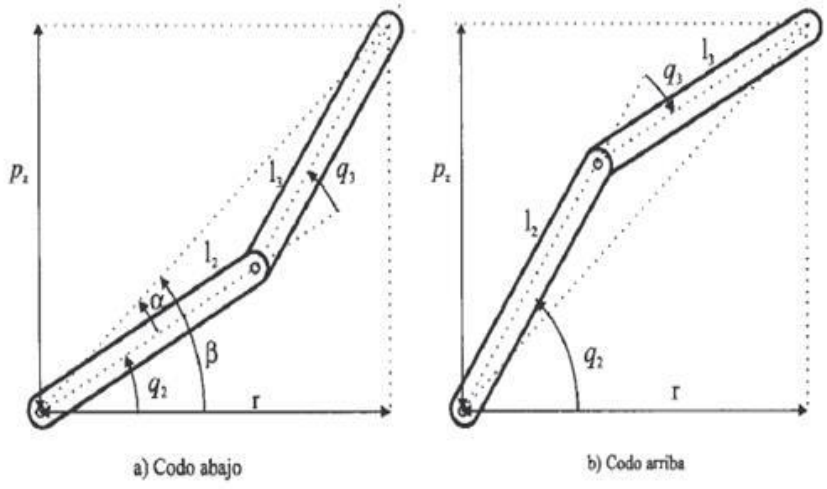


Figura 4.4. Dos configuraciones para una misma posición final [1].

- Método basado en la matriz de transformación homogénea:

  Conociéndose la matriz de transformaciones homogénea, la cual ofrece la posición y orientación del extremo final según las coordenadas articulares, se

puede pensar que de una manera inversa la misma matriz puede resolver el caso contrario; sin embargo, este método implica la resolución de sistemas con más ecuaciones que incógnitas, por eso hay que tener cuidado a la hora de no trabajar con ecuaciones redundantes. Este método, al igual que el anterior, es sencillo de utilizar para robots con pocos grados de libertad.

- Desacoplo cinemático :

  El desacoplo cinemático es un método que nos permite separar el problema cinemático inverso en dos problemas, uno consistirá en obtener la posición de la muñeca mediante el método geométrico para más tarde obtener la orientación con el método basado en la matriz de transformación homogénea.

4.1.6 Matriz Jacobiana geométrica

La matriz Jacobiana de un robot es una matriz diferencial que relaciona el vector de velocidades articulares con otro vector de velocidades expresado en un espacio distinto [1].

Se puede clasificar entre Jacobiana analítica, la cual expresa las velocidades de las articulaciones con los ángulos de Euler, y Jacobiana geométrica, que relaciona las velocidades de las articulaciones con velocidades angulares según el sistema de coordenadas de la base del robot ,que será la que se use en este trabajo.

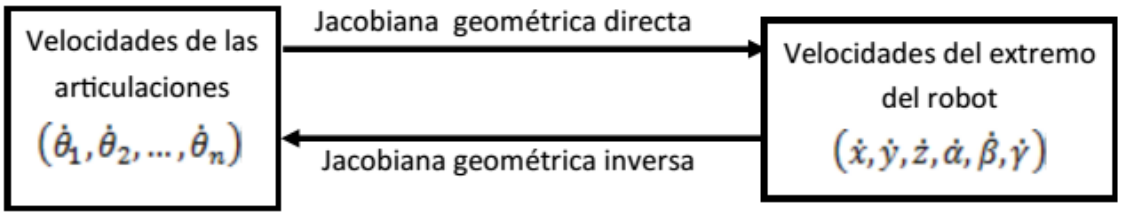


Figura 4.5. Jacobiana geométrica directa e inversa.

El método más directo para obtener la relación entre estas velocidades es haciendo la derivación de las ecuaciones de posición. Sin embargo, no disponemos de una ecuación que exprese los ángulos de giro según el sistema de coordenadas de la base [7].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = J_g \cdot \begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} con\ J_g = \begin{bmatrix} \frac{\partial f_x}{\partial \theta_1} & \cdots & \frac{\partial f_x}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_\gamma}{\partial \theta_1} & \cdots & \frac{\partial f_\gamma}{\partial \theta_n} \end{bmatrix} \quad \text{(Ec 4.6)}$$

Debido a esto se procede al cálculo de la jacobiana geométrica mediante obtención numérica, la cual usa la información disponible en las matrices de transformación homogénea del problema cinemático directo.

Procedemos a calcular los siguientes vectores, especificados en el método seguido según [1]:

- Se denomina $^{0}\mathbf{z}_{i}$ al vector unitario orientado según el eje de la articulación *i+1*, definido en el sistema de coordenadas de la base del robot {$\mathbf{S}_0$}. De modo que $^{0}\mathbf{z}_{i}$ estará definido por los tres primeros elementos de la tercera columna de $^{0}\mathbf{A}_{i}$. (Al ser $^{0}\mathbf{A}_0$ la matriz identidad $^{0}\mathbf{z}_0$ será el vector (0,0,1)).

$$^{0}\boldsymbol{z_i} = {}^{0}\boldsymbol{A_i}\ (1{:}3,3) \qquad \text{(Ec 4.7)}$$

- $^{0}\boldsymbol{z_i} = {}^{0}\boldsymbol{A_i}\ (1{:}3,3)$Se denominará $^{i}\mathbf{p}_{n}$ al vector que va desde el origen del sistema {$\mathbf{S}_i$} hasta el extremo del robot {$\mathbf{S}_n$} expresado en el sistema de la base del robot {$\mathbf{S}_0$}. Puesto que la cuarta columna de $^{0}\mathbf{A}_{n}$ contiene las coordenadas del extremo del robot en el sistema {$\mathbf{S}_0$} y la cuarta columna del $^{0}\mathbf{A}_{i}$ contiene las coordenadas del origen del sistema {$\mathbf{S}_i$} en el sistema {$\mathbf{S}_0$}, $^{i}\mathbf{p}_{n}$ se obtendrá restando las cuartas columnas de $^{0}\mathbf{A}_{n}$ y $^{0}\mathbf{A}_{i}$:

$$^{i}\boldsymbol{p_n} = {}^{0}\boldsymbol{A_n}\ (1{:}3,4) - {}^{0}\boldsymbol{A_i}\ (1{:}3,4) \qquad \text{(Ec 4.8)}$$

Definidos los vectores $^{0}\mathbf{z}_{i}$ y $^{i}\mathbf{p}_{n}$, la matriz jacobiana que relaciona las velocidades articulares con las velocidades de traslación y rotación del extremo del robot, expresadas en el sistema de coordenadas de la base, se puede obtener mediante una matriz 6 x *n* (*n*: número de grados de libertad) expresada por columnas como:

$$J = [J_1|J_2|\dots|J_n] \qquad \text{(Ec 4.9)}$$

Donde:

$$J_i = \begin{cases} \begin{bmatrix} {}^{0}z_{i-1} \times {}^{i-1}p_n \\ {}^{0}z_{i-1} \end{bmatrix} & Si\ el\ eslabón\ i\ es\ de\ rotación \\ \begin{bmatrix} {}^{0}z_{i-1} \\ 0 \end{bmatrix} & Si\ el\ eslabón\ i\ es\ de\ translación \end{cases}$$

4.1.7 *Jacobiana inversa*

Del mismo modo que se obtiene la relación directa de velocidades, se puede obtener la relación inversa. Este caso es mucho más sencillo que para la posición ya que, simplemente, conocida la matriz Jacobiana, se puede obtener la relación inversa invirtiendo la matriz.

$$\begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} = {J_g}^{-1} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} con\ {J_g}^{-1} = \begin{bmatrix} \frac{\partial f_{\theta_1}}{\partial x} & \cdots & \frac{\partial f_{\theta_1}}{\partial \gamma} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{\theta_n}}{\partial x} & \cdots & \frac{\partial f_{\theta_n}}{\partial \theta \gamma_n} \end{bmatrix} \quad \text{(Ec 4.10)}$$

4.1.8 *Configuraciones singulares*

Se definen como configuraciones singulares los puntos del área de trabajo del robot en los cuales el robot no puede acceder según cualquier orientación. Estos puntos suelen encontrarse en los límites del área de trabajo o en puntos donde se alienan dos o más ejes.

Para calcular estos puntos deben calcularse las configuraciones en las que el determinante de la matriz Jacobiana sea nulo, por este motivo no existiría su matriz inversa [1].

4.2 Dinámica de robots

En este apartado se revisan los conceptos teóricos referentes a la dinámica de un robot, haciendo una explicación de la obtención de los modelos.

4.2.1 *Introducción a la dinámica de robots.*

Se conoce como modelo dinámico de un robot [1] a aquel modelo que permite conocer la relación entre el movimiento del robot y las fuerzas implicadas. Se trata por tanto de establecer relaciones matemáticas entre:

- La localización del robot, definida mediante sus variables articulares o mediante la ubicación y orientación del extremo final y sus derivadas: velocidad y aceleración. En definitiva, las variables cinemáticas del robot.
- Los esfuerzos aplicados en las articulaciones o en el extremo del robot.
- Longitudes, masas e inercias de los eslabones, es decir, los parámetros dimensionales del robot.

La obtención del modelo no es demasiado compleja en robots de 1 o 2 GDL, pero se complica bastante en robots de más ejes. De hecho, no siempre es posible obtener un modelo dinámico en forma cerrada, es decir, una serie de ecuaciones diferenciales que al integrarlas pueda caracterizarse un movimiento a partir de unas fuerzas conocidas, o averiguar las fuerzas necesarias para conseguir un movimiento determinado. En estos casos, la mayoría en robots industriales, el modelo dinámico se resuelve mediante algún método numérico iterativo, como los que se mostrarán a continuación.

A pesar de resultar complejo, el modelo dinámico es necesario para:

- Realizar simulaciones fiables del movimiento del robot.
- Dimensionar los actuadores.
- Diseñar el control dinámico del robot, del que dependen precisión y velocidad en los movimientos.

Para que el modelo sea preciso, debe incluir no solo información sobre los eslabones, sino también sobre los sistemas de transmisión, actuadores y electrónica de control, que aportan rozamientos e inercias. En la mayor parte de las aplicaciones reales se consideran los eslabones como solidos rígidos, lo cual simplifica el modelo pero, en algunos casos, como robots espaciales o de grandes dimensiones, es necesario considerar la deformación de los eslabones y tratarlos por tanto como sólidos deformables.

4.2.2 Modelo dinámico de la estructura mecánica

La obtención del modelo dinámico de un robot se basa fundamentalmente en el planteamiento del equilibrio de fuerzas establecido por la segunda ley de Newton, o su equivalente en movimientos de rotación establecido por la ley de Euler:

$$\sum \vec{F} = m \cdot \vec{v} \quad \text{(Ec} \tag{Ec 4.11}$$

$$\sum \vec{T} = \vec{I} \cdot \dot{\omega} + \omega \times (\vec{I}\omega) \tag{Ec 4.12}$$

Para el caso más sencillo del robot monoarticular de la Figura 4.6, el equilibrio de momentos supone:

$$T = \vec{I} \cdot \frac{d^2\theta}{dt^2} + mgLCos\ \theta = mL^2\ddot{\theta} + mgLCos\ \theta \tag{Ec 4.13}$$

Donde se supone toda la masa concentrada en el CDG del eslabón, no existe rozamiento y no se manipula carga alguna (fuerza en el extremo).

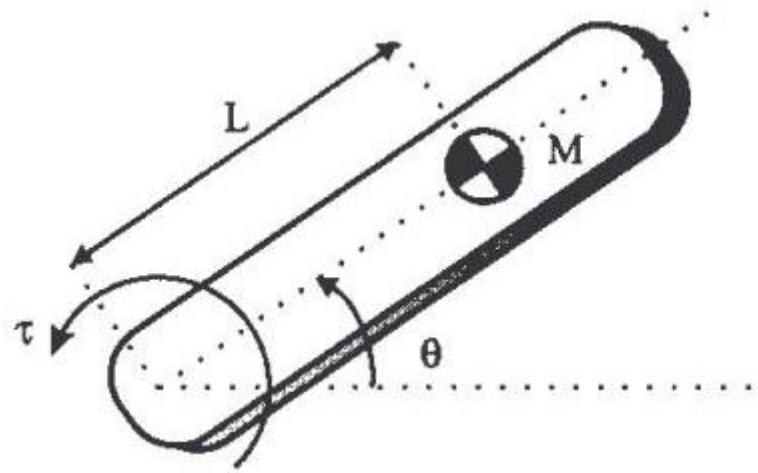


Figura 4.6. Modelo de eslabón con masa concentrada [1].

Así conocidos los parámetros dimensionales del robot (*m,L*) (Fig 4.6), para un motor *T* determinado, integrando la ecuación anterior (Ec 4.13) se obtienen las variables cinemáticas de posición $\theta(t)$, velocidad $\dot{\theta}(t)$, y aceleración $\ddot{\theta}(t)$. Por otro lado, si se pretende obtener un determinado movimiento $\theta(t)$, puede emplearse la ecuación para obtener el par motor necesario $T(t)$.

Se distinguen así dos modelos dinámicos:

- **Modelo dinámico directo:** El que expresa la evolución temporal de las coordenadas articulares del robot en función de las fuerzas y pares que intervienen.
- **Modelo dinámico inverso:** Expresa las fuerzas y pares que intervienen en función de la evolución de las coordenadas articulares y sus derivadas.

El planteamiento del equilibrio de fuerzas en un robot real de 5 o 6 GDL es mucho más complejo, no solo por disponer de más eslabones, sino porque aparecen fuerzas de Coriolis y fuerzas centrípetas que dependen de la configuración instantánea del robot.

4.3 Programación de robots

La programación de un robots se puede definir [1] como el proceso mediante el cual se le indica a éste la secuencia de acciones que deberá llevar a cabo durante la realización de su tarea. Estas acciones consisten en su mayor parte en moverse a puntos predefinidos y manipular objetos del entorno.

Durante la ejecución de un programa se interacciona con la memoria del sistema, leyendo y actualizando el contenido de las variables utilizadas en el programa, y con los sistemas cinemático y dinámico del robot, encargados de dar la señal de mando a los accionamientos de las máquinas y elementos que componen su entorno.

En España no existe una norma de obligado cumplimiento en cuanto a los sistemas de programación. La norma UNE EN ISO 8373:1998 [3] establecía algunas definiciones y métodos de programación, pero dicha norma ha sido anulada en 2012 y no ha sido publicada una que la sustituya. Cada fabricante ha desarrollado sus propios métodos (incluso lenguajes), si bien en la gran mayoría se dan una serie de características comunes.

En el siguiente subapartado se examinan los distintos métodos existentes para la programación de robots, realizándose una clasificación de los mismos. A continuación se analizan las características propias de los sistemas de programación y sus requerimientos de funcionamiento.

4.3.1 Métodos de programación. Clasificación.

Existen diversos criterios para la clasificación de los métodos de programación de robots [1]. El criterio más extendido hace referencia al sistema empleado para indicar la secuencia de acciones a realizar por el robot, según el cual, un robot puede ser programado de una forma guiada del elemento terminal o con un procedimiento textual. Muchos robots actuales implementan ambos métodos. Una última opción es la programación off-line o simulación, que permite al usuario optimizar el estudio de viabilidad y reducir plazos de entrega. También es cierto que requiere de personal cualificado y una puesta a punto del sistema para eliminar diferencias entre el modelo teórico y el real.

4.3.1.1 Programación por guiado.

Consiste en acompañar o guiar al elemento terminal (manualmente o mediante algún elemento de la interfaz) a lo largo de todo el ciclo de movimientos, al tiempo que se registran las posiciones y configuraciones adoptadas para su posterior repetición de manera automática [1].

Para guiar el movimiento del robot existen dos métodos:

- Guiado pasivo: Los actuadores del robot están desconectados, de manera que es el programador el que debe aportar la energía manualmente para mover el robot. Para evitar el gran esfuerzo físico necesario para mover el robot suelen usarse maniquís del mismo robot pero más ligero.

- Guiado activo: Emplea el propio sistema de accionamiento del robot, controlado desde una botonera, un joystick o similar.

Dentro del guiado pasivo, según los datos almacenados se puede distinguir en:

- Guiado básico: El robot sigue la trayectoria programada de forma secuencial sin que sea posible incluir ningún tipo de estructura de control.
- Guiado extendido: Permite especificar, además de la trayectoria, datos relativos a velocidad, precisión deseada en determinados puntos, etc.

En general, la programación por guiado presenta importantes ventajas pues es sencilla de aprender y requiere poco espacio en memoria, resultando muy útil e incluso imprescindible en determinadas circunstancias y ocasiones.

También presenta inconvenientes. En primer lugar, se necesita al robot para realizar un programa, de manera que se le invalida temporalmente en el proceso productivo. Además, el programa queda sin documentar, lo que dificulta posteriores modificaciones y, por tanto, la realización de una buena puesta a punto de los programas (revisión, adición de puntos de control, etc.).

4.3.1.2 Programación textual.

Se indica la tarea al robot mediante el uso de un lenguaje de programación específico, formado por una serie de órdenes que son creadas, editadas y posteriormente ejecutadas. Existen tres niveles de programación textual:

- Nivel robot: Las órdenes se refieren a los movimientos a realizar por el robot. Es necesario especificar cada uno de los movimientos, así como la velocidad, precisión, apertura y cierre de pinzas, etc. Este nivel se encuentra completamente implementado en los robots industriales del mercado.
- Nivel objeto: Las órdenes se refieren al estado en que deben quedar los objetos a mover o con los que se interactúa, de manera que un planificador de tareas se encarga de consultar una base de datos y generar instrucciones a nivel robot. Este nivel se encuentra en pleno desarrollo y algunos fabricantes ya lo implementan parcialmente.
- Nivel tarea: Las órdenes se refieren al objetivo a conseguir. El programa se reduce a sentencias globales en las que se indica qué debe conseguir el robot, en lugar de cómo conseguirlo (P.ej: ensamblar A en B). En fase de investigación.

Como resumen, la Figura 4.7 recoge los diferentes métodos de programación de robots existentes comentados anteriormente.

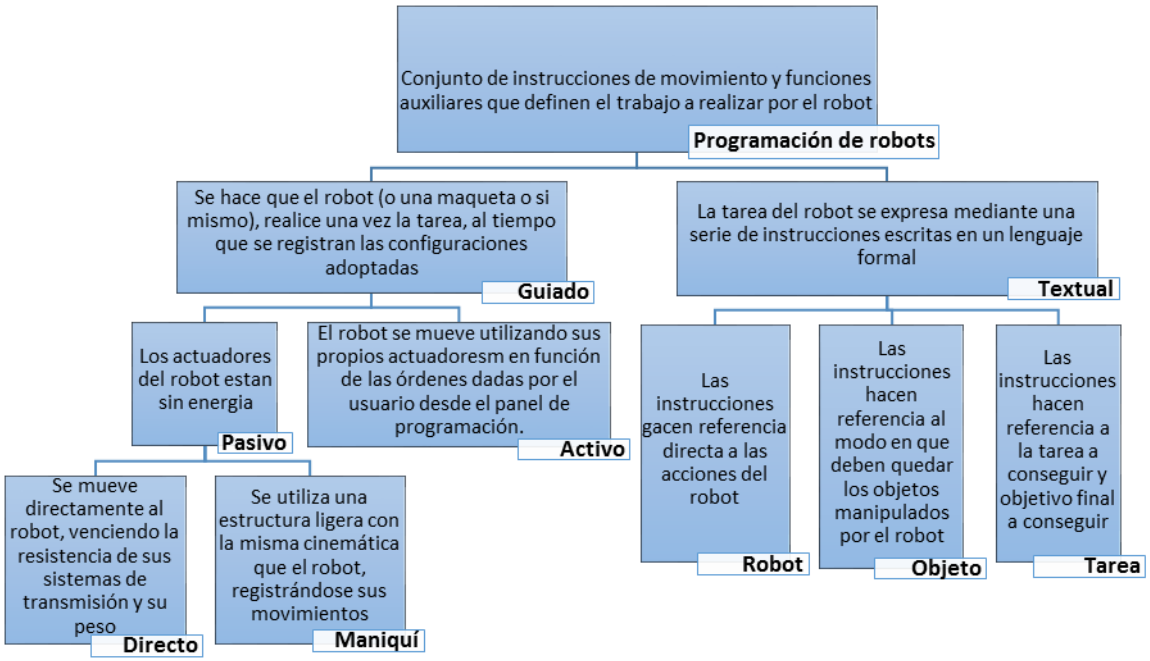


Figura 4.7. Modos de programación de Robots Manipuladores [1].

4.3.2 Requerimientos de un sistema de programación de robots

A pesar de la falta de normalización de los métodos de programación, las necesidades más comunes para el usuario han originado cierto paralelismo entre casi todos ellos, que se traducen en unos requerimientos generales [1] para cualquier sistema de programación de robots:

- Entorno de programación: Es importante que el sistema de programación presente una buena capacidad de depuración y de ejecución paso a paso, pues se trata de un proceso continuo de prueba y error, con especial consideración en la interacción entre los distintos equipos de la célula con el controlador del robot.
- Modelado del entorno: Es la representación que tiene el robot de los objetos con los que interacciona. Normalmente se limita a características geométricas (dimensiones, posición y orientación), extendiéndose según necesidades a otras características (forma, peso, etc.).
- Tipos de datos: Además de los convencionales (enteros, reales, etc.) debe poder manejar datos específicamente destinados a definir las operaciones de interacción con el entorno, como son los que especifican posición y orientación de puntos a los que debe acceder el robot, bien mediante

coordenadas articulares (se necesitan elementos de orden n, n-uplas, siendo n el nº GDL), o bien mediante coordenadas cartesianas (ángulos, cuaternios y matrices pueden ser requeridos).

- Manejo de entradas / salidas: La comunicación del robot con otras máquinas se consigue mediante señales binarias E/S. A mayor nivel se requiere de redes locales de comunicación.
- Control del movimiento del robot: Un método de programación debe incluir la posibilidad de especificar el movimiento del robot (punto a punto, trayectoria continua,…) además de velocidad en el recorrido, precisión, puntos singulares, etc. También hay que considerar la influencia en el movimiento de las señales captadas por los sensores.
- Control del flujo de ejecución del programa: Es vital poder controlar el orden de ejecución de las diferentes tareas, así como las paradas y el poder establecer prioridades entre órdenes. Para ello se emplean las estructuras habituales de bucles *for*, *if, while, repeat*, etc.

5 MATERIALES Y MÉTODOS

En este capítulo se hará un recorrido por los materiales y métodos utilizados para la realización de este TFG, donde se explicará la maquinaria utilizada, su metodología de utilización y puesta en marcha.

5.1 Material utilizado

En este subapartado nos centramos en especificar las características del material utilizado por el autor de este trabajo.

5.1.1 Célula didáctica IRB 120

Se dispone de una célula didáctica de la marca ABB la cual se encuentra en el laboratorio de mecánica y medios continuos de la Escuela Politécnica Superior de Linares. Esta célula viene con el paquete educativo ofertado por ABB para futuros ingenieros, con el propósito de acercar la robótica y su producto de una forma docente a las universidades y centros de enseñanza [21].

Figura 5.1. Célula robótica IRB 120 de la marca ABB [21]

Está compuesto por un Robot IRB 120 de ABB, una estructura de montaje con superficie de trabajo, un controlador compacto IRC5 [11], 13 opciones de Robotware y 50 Licencias de RobotStudio.

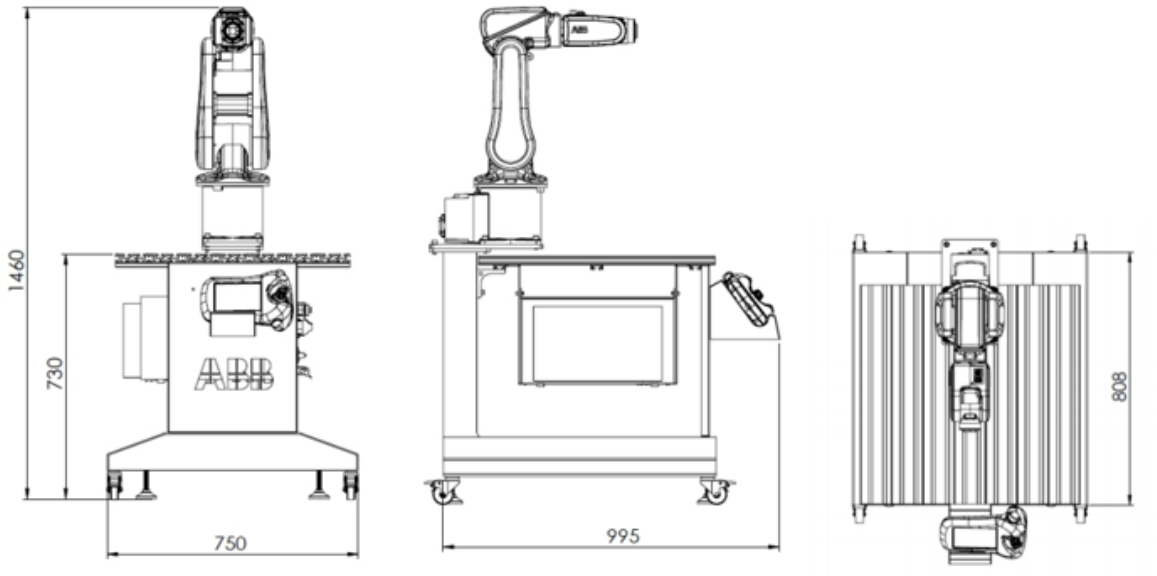


Figura 5.2. Célula didáctica IRB-120. Medidas generales [21].

Las 13 opciones de RobotWare incluidas son:

- Multitasking: Ejecutar hasta 14 programas de RAPID simultáneamente. Utilizarlos para la supervisión de un equipo externo, el operador o cálculos avanzados.
- World Zones: Define acciones cuando un robot entra en una zona definida del espacio de trabajo. Las zonas se pueden utilizar para parar el robot de entrar en una zona, ya sea de forma permanente o sólo cuando otro robot está trabajando en la zona.
- PC Interface: PC interfaz proporciona la interfaz de comunicaciones entre el robot y un PC de red.

  Esto es útil cuando se desea:

  - Utilizar una interfaz de servidor OPC para la integración SCADA.
  - Uso de RobotStudio para interactuar con el controlador a través de una conexión de red.
- FlexPendant Interface: Permite a los usuarios ejecutar sus propias aplicaciones en el FlexPendant, por ejemplo un panel de operador. Las aplicaciones se desarrollan en Microsoft Visual Studio.net.
- FTP and NFS client: El cliente FTP / NFS hace que sea posible leer la información en un disco duro remoto directamente desde el controlador.

- Collision Detection: Protege a los equipos y el robot de daños graves. Se detiene el robot si se exceden los valores de par de movimiento
- Path Offset: Un seguimiento de la trayectoria del robot programado a una distancia de desplazamiento dado. El robot puede alternar siguiendo el camino y hacer un desplazamiento, en función de las aportaciones de una IA / DI o canal serie.
- Advanced Shape Tuning: Ofrece la posibilidad de compensar los efectos de fricción que pueden aparecer en velocidades bajas (10-100 mm / s).
- Path Recovery: Almacena todos los datos del sistema, cuando se produce una interrupción (mensaje de fallo o de otro tipo) y los restaura después de que se han tomado las medidas necesarias. Útil para interrupciones de servicio.
- Sensor Interface: Caja de herramientas para integrar sensores basados en la comunicación en serie.
- Production Manager: La pantalla de producción es una interfaz gráfica fácil de usar basada para el FlexPendant. El software se basa en widgets, elementos gráficos que se utilizan para ejecutar funciones de su elección.
- Soft Move: En aplicaciones en las que los materiales o herramientas no se pueden posicionar con precisión, el robot puede ser ajustado en el modo Soft Servo, permitiendo que el robot pueda actuar como un resorte mecánico cuando se enfrentan a la resistencia del entorno
- Independent Axis: Permite utilizar un eje adicional de forma independiente a los propios del sistema.

5.1.2 Brazo robótico IRB 120 ABB

El robot a estudio de este trabajo es un Robot IRB 120 de la marca ABB. Se trata de un robot industrial de 6 ejes de movimiento, con una carga máxima de 3 kg y un alcance máximo de 580 mm. Es miembro de la última generación de robots de la marca ABB, siendo el más pequeño de esta, pesando solo 25 kg.

Según la clasificación de robots comentada anteriormente, se puede definir este robot de la siguiente manera:

- Según su área de aplicación: Debido a su tamaño y su gran agilidad este robot es usado principalmente para el manejo de materiales de pequeño tamaño, diseñado principalmente para la industria farmacéutica y electrónica en tareas de empaquetamiento y distribución. En este caso, al estar empleado con fines docentes en la universidad, podemos definirlo como un robot dedicado a la “Formación, enseñanza e investigación”.

- Según tipo de actuadores: Dispone de 6 motores eléctricos paso a paso que ejecutan los movimientos con una repetibilidad de 10 micras [9].
- Tipo de control: El robot está equipado con el controlador IRC5 Compact y el software de control de robots RobotWare. RobotWare admite todos los aspectos del sistema de robot, como el control del movimiento, el desarrollo y la ejecución de programas, la comunicación, etc. [9].
- Numero de ejes: Dispone de 6 ejes de movimiento rotacionales, lo que le permite ubicar su extremo en cualquier posición de su área de trabajo.

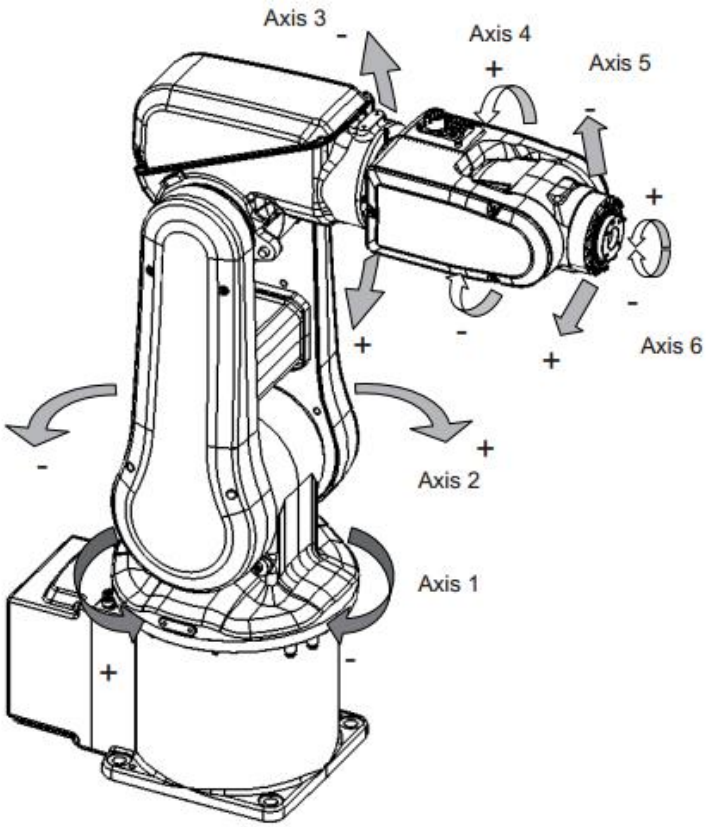


Figura 5.3. Número de ejes, manual de especificaciones de producto IRB120 [9]

- Configuración: Como puede verse en la figura 5.3, el robot dispone de una configuración angular o antropomórfica lo que le proporciona zona amplia de trabajo.

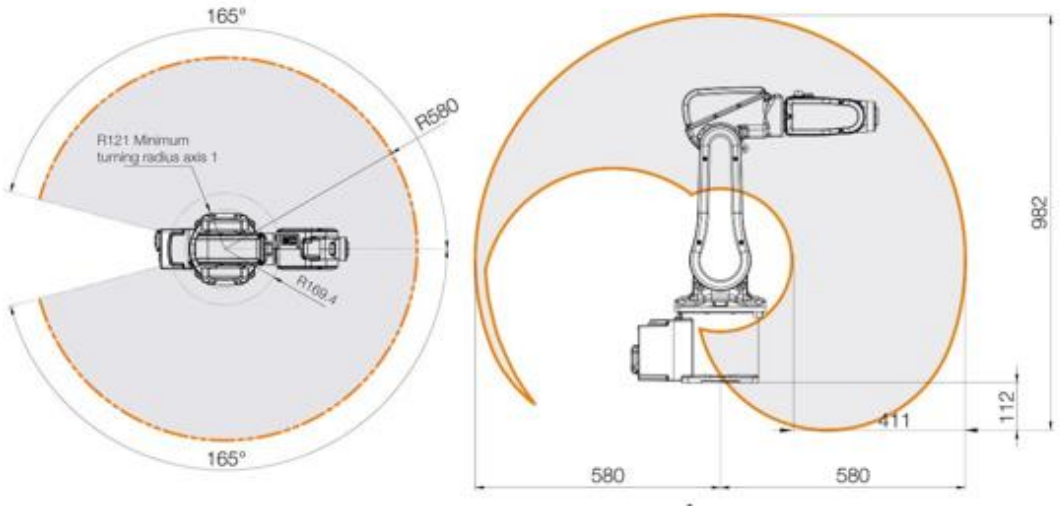


Figura 5.4. Área de trabajo del centro de la muñeca (eje 5) [9]

- Capacidad de carga: El robot puede manipular hasta 3 kg (carga máxima genérica) pero no puede realizarlo en todo el área de trabajo, dependiendo de la ubicación y orientación del extremo esta capacidad puede verse disminuida.

  Para conocer la carga máxima en cada punto es necesario utilizar los diagramas de carga del producto [10], los cuales expresan cuánto peso puede levantar el extremo del robot dependiendo de la altura a levantar y la separación de la base.

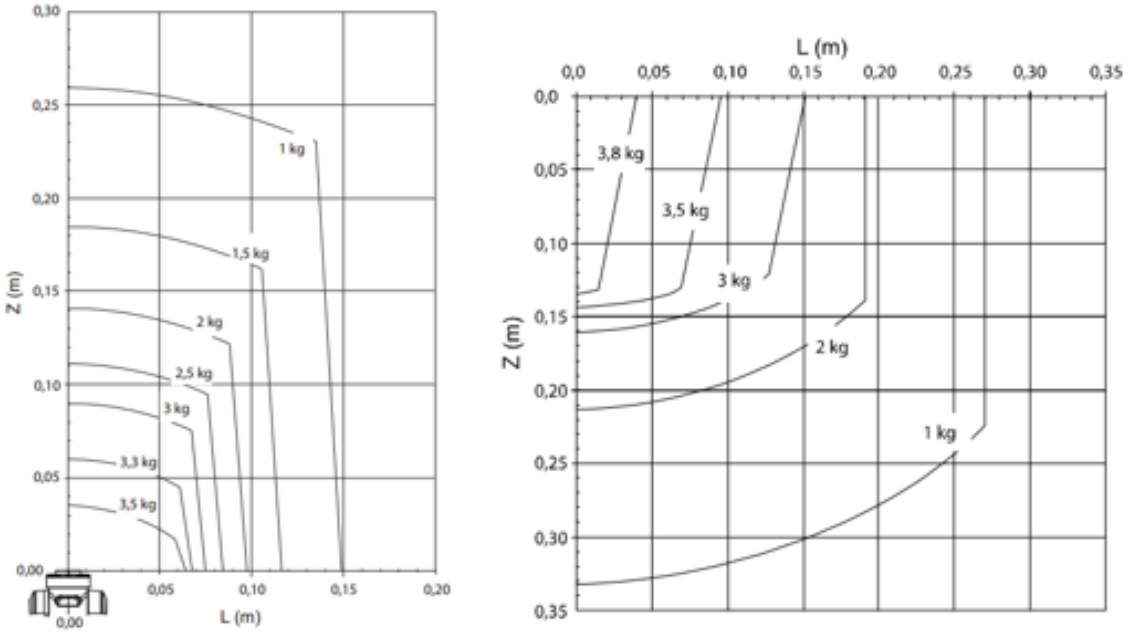


Figura 5.5. Diagrama de carga normal y con la muñeca vertical [9]

5.1.3 Controlador IRC5 Compacto

El controlador IRC5 (Fig 5.6) contiene todas las funciones necesarias para mover y controlar el robot.

Figura 5.6. Controlador IRC5 Compacto [11].

El módulo de control contiene todos los elementos electrónicos de control como el ordenador principal, las tarjetas de E/S y la unidad de almacenamiento (Fig 5.7). El módulo de accionamiento contiene todos los elementos electrónicos de alimentación que proporcionan la alimentación a los motores del robot. Un módulo de accionamiento IRC5 puede contener los accionamientos de los seis ejes del robot y además dos o tres accionamientos para los ejes externos en función del modelo de robot. En el controlador IRC5 Compacto, el módulo de control y el de accionamientos están integrados en un solo armario y se utiliza con robots pequeños.

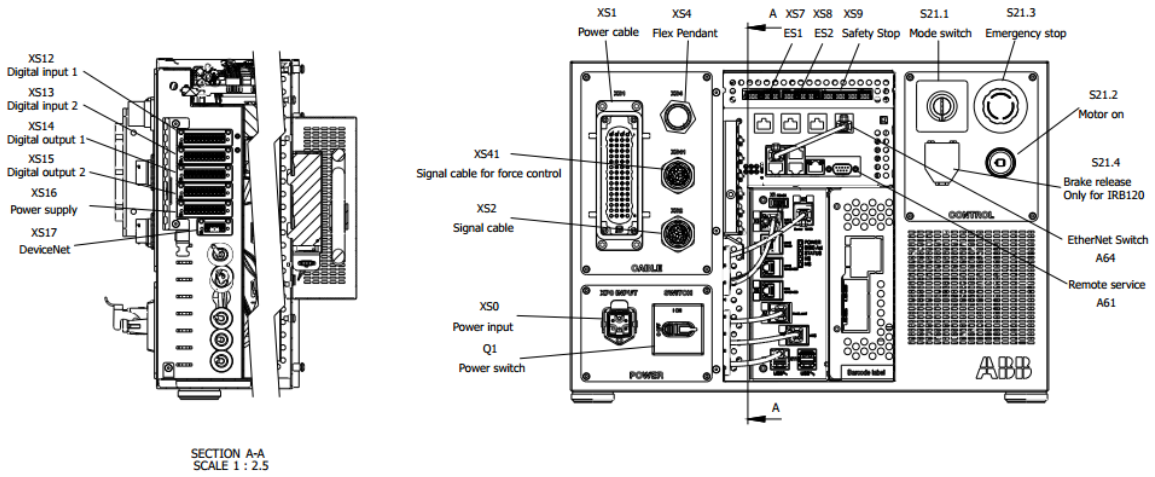


Figura 5.7. Esquemas eléctricos IRC5 [11].

La parte delantera del módulo de control también cuenta con un puerto de servicio. Está situado debajo de los pulsadores y oculto tras una cubierta protectora. El puerto de servicio puede usarse para conectarse a un PC (Conexión Ethernet).

Con el controlador IRC5 se dispone de dos modos de funcionamiento: Modo automático y modo manual (Tabla 5.1).

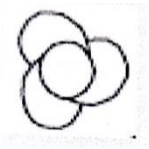
Modo automático 	• Modo de producción. • Velocidad de ejecución programada, sin limitar.
Modo manual 	• Modo de programación y test. • Velocidad de ejecución limitada a 250 mm/s. • Supervisión manual del movimiento.

Tabla 5.1. Modos de funcionamiento [8]

Este controlador está adaptado con la integración del software RobotStudio y la utilización de una unidad de programación o FlexPendant.

5.1.4 Unidad de programación.

La unidad de Programación es un dispositivo que maneja muchas de las funciones relacionadas con el uso del sistema del robot, como ejecutar programas, mover el manipulador, crear y editar programas de aplicación, etc.

Figura 5.8. Unidad de programación de la marca ABB o FlexPendant [12]

Se compone de una pantalla táctil, con botones y un joystick. Este aparato está conectado al módulo de control a través de un cable con conector integrado.

Figura 5.9. Distribución de botones de FlexPendant [8]

5.1.5 RobotStudio

El RobotStudio es un software que se ejecuta en un PC. Debe estar conectado al puerto de servicio del controlador o la conexión WAN, permite la programación y edición del programa así como la carga del sistema operativo del robot.

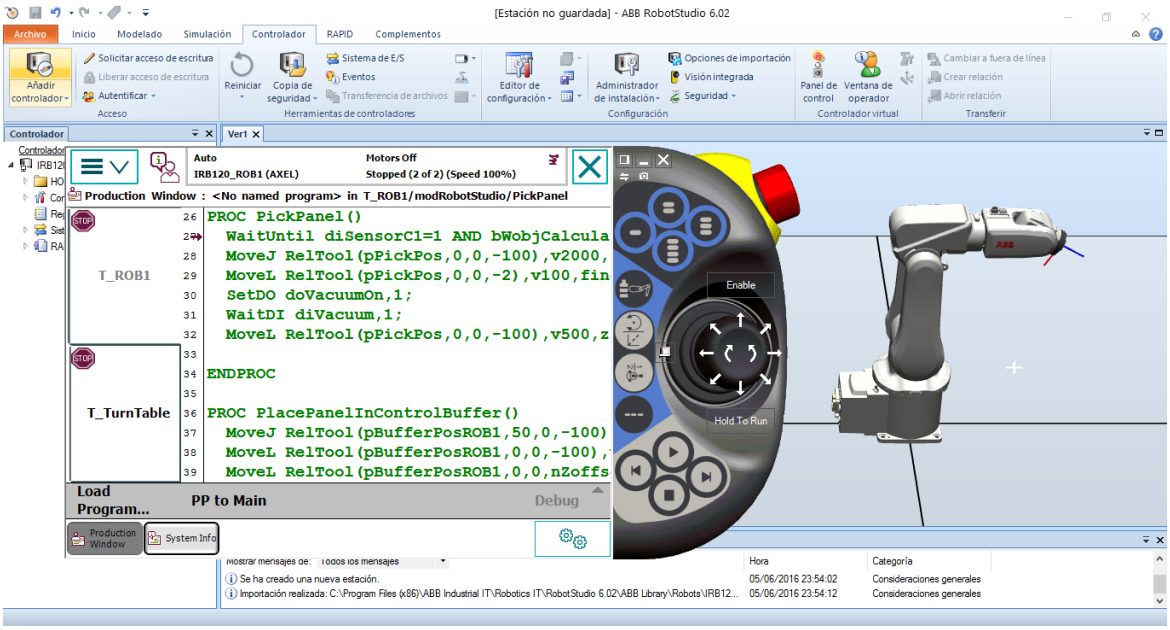


Figura 5.10. RobotStudio 6.02 con FlexPendant virtual.

Con RobotStudio se puede diseñar una célula robótica idéntica a la que opera el robot y realizar la programación y simulaciones sobre ella sin necesidad del robot. Permite también generar una simulación de la unidad de programación lo que permite mover y programar el robot exactamente igual que si se tuviera físicamente.

5.1.6 Pinza eléctrica SMC

La herramienta terminal que dispone el brazo robótico es una pinza eléctrica de la marca SMC modelo LEHZ25K2-14-R16P1.

El número del modelo especifica algunas características de la pinza (Tabla 5.2), obtenidas a partir de la página web del fabricante, catalogo y manual de especificaciones técnicas del producto [16].

Tamaño del cuerpo	25
Modelo de motor	Estándar
Carrera	14 mm
Opciones de dedo	Modelo básico
Entrada del cable del motor	Básico, entrada en el lado izquierdo
Tipo de cable del actuador	R [Cable robótica (cable flexible)]
Tipo de controlador	6P (Con controlador PNP)
Longitud del cable E/S	1 (1.5 m)
Montaje del controlador	Montaggio con viti

Tabla 5.2. Características y especificaciones de pinza de la marca SMC [16]

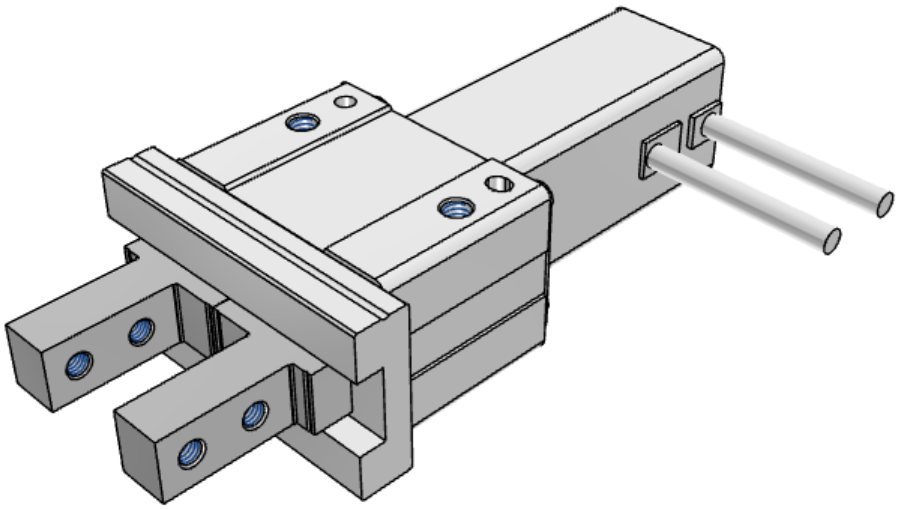


Figura 5.11. Modelo 3D de pinza eléctrica de la marca SMC [12]

Estas pinzas disponen de actuadores eléctricos que tienen como ventaja sobre las garras mecánicas la posibilidad de regular el recorrido de los dedos controlando el movimiento de los motores. Estos dedos se encuentran en posición paralela y se mueven de forma síncrona para el agarre de las piezas (Fig 5.11).

Dispone de 14 mm de desplazamiento entre los dedos y una fuerza de agarre variable de 16 a 40 N. La forma plana de los dedos de agarre proporciona una superficie dificultosa para el agarre de piezas, por ello dispone de dos taladros pasantes M5 en la dirección de apertura/cierre para añadir una pieza de agarre para la pinza que adecue la superficie de contacto con la pieza a agarrar.

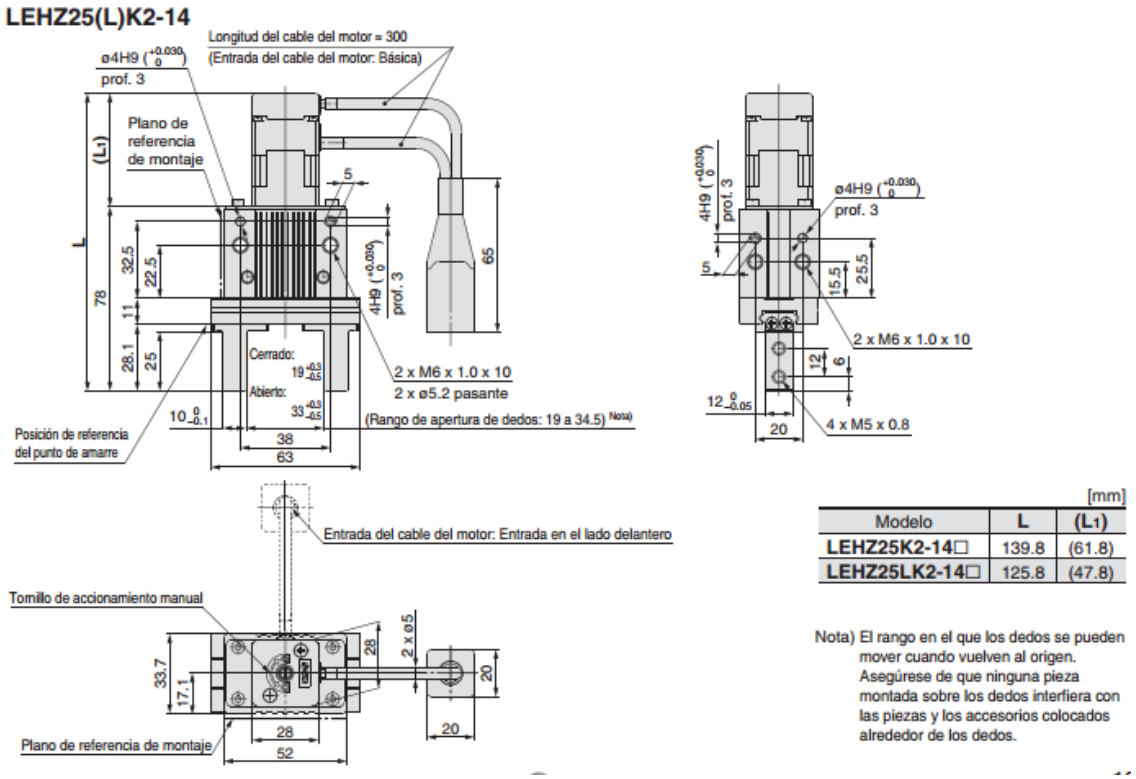


Figura 5.12. Plano de la pinza del catálogo de productos de SMC [13]

Para este robot, se dispone de unas mordazas de agarre genérico de superficie plana con dos aberturas triangulares, que se añaden a los dedos mediante dos tornillos para aumentar el rango de piezas a manipular.

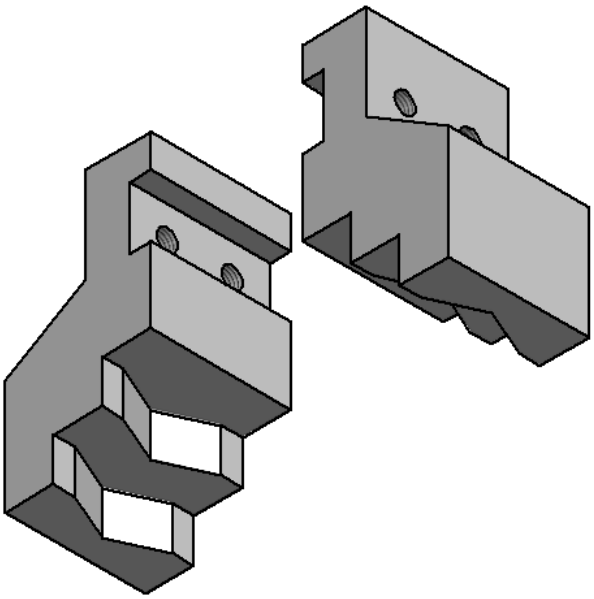


Figura 5.13. Mordazas de sujeción. Dibujadas con SolidEdge.

Para aumentar el agarre de la superficie, se le proporcionó un baño de material plástico mediante inmersión que hiciera la superficie más flexible y a la vez más rugosa para que las piezas no se deslizaran al agarrarlas. Por otra parte, esta capa de plástico protege la pinza y piezas de golpes y arañazos.

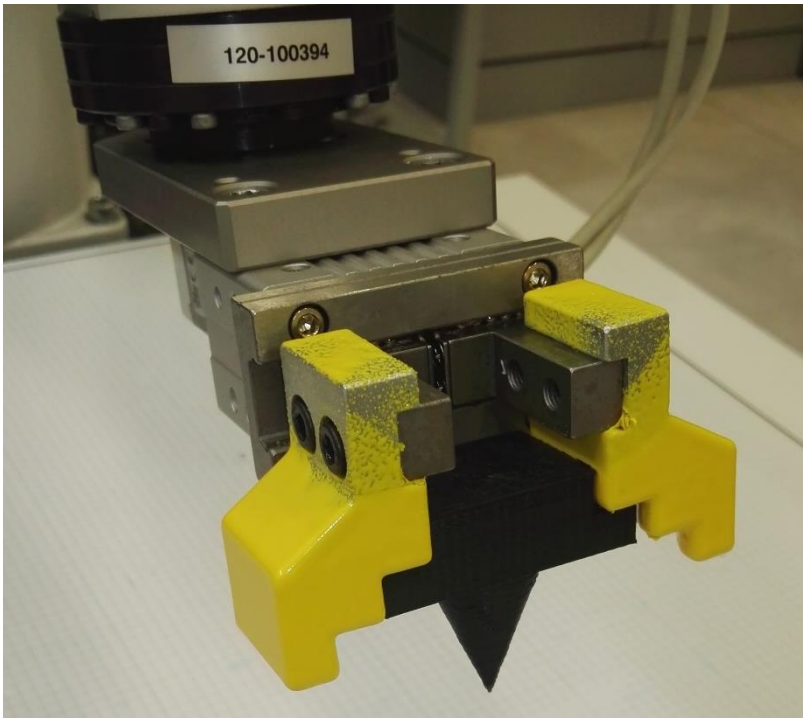


Figura 5.14. Mordazas con el recubrimiento plástico montadas en la pinza

El modo de integración de la pinza con el robot se realiza mediante el uso del ordenador conectado al controlador utilizando el software “ACT Controller”. Su funcionamiento se explicará más adelante en el apartado de instrucciones.

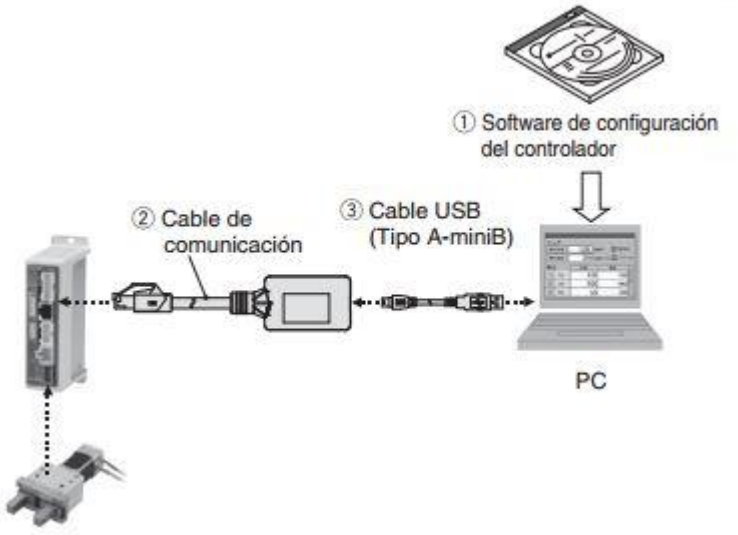


Figura 5.15. Conexión pinza – PC [15].

5.2 Metodología para la puesta en marcha

En este apartado se va a exponer el procedimiento para poner en marcha el equipo. Con puesta en marcha, el autor de este trabajo quiere referirse al procedimiento y acciones que se han llevado a cabo para hacer de la célula robótica un equipo funcional y útil para su uso, dotándolo de las herramientas y programas necesarios para su utilización, aparte de la creación de unas instrucciones de uso para cualquier usuario.

5.2.1 Elemento terminal

Todas las herramientas deben tener definido un TCP (punto central de la herramienta), que es el punto respecto del cual se definen todas las posiciones del robot. El punto central de la herramienta se define respecto de la posición de la brida de sujeción del manipulador, donde se encuentra definida la herramienta “tool0”. Este punto también constituye el origen del sistema de coordenadas de la herramienta. El sistema del robot puede manejar varias herramientas a la vez, pero solo puede estar activa una de ellas en cada momento.

Al utilizar las garras eléctricas no podemos definir un punto final exacto como punto central de la herramienta ni utilizar la misma para definir otros parámetros como el plano de trabajo.

Por tanto, se procede al diseño de un útil, el cual se pueda agarrar con la pinza eléctrica del robot y nos sirva como punto de referencia para definir tanto la herramienta como el plano de trabajo (Fig 5.16).

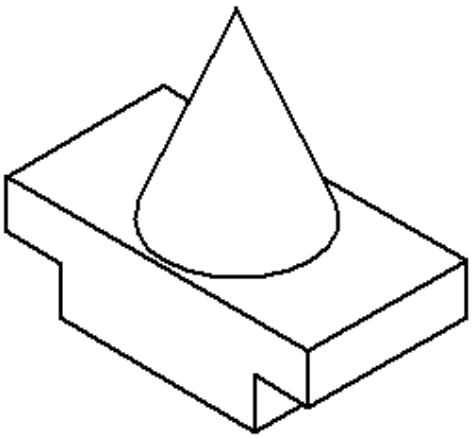


Figura 5.16. Imagen 3D de la herramienta cono, dibujada con SolidEdge

A esta herramienta se le llamará durante el trabajo “cono”. Se trata de una pieza diseñada mediante el software SolidEdge e imprimida con una impresora 3D. Esta pieza dispone de una parte rectangular adaptada a la geometría de la pinza para facilitar su agarre y una parte cónica que nos ofrece un punto final preciso para marcarlo como TCP.

5.2.2 Definición de herramienta

Paso 1: Colocación cono

Lo primero que se debe realizar es la colocación del cono en la pinza, para ello se abre la pinza al completo, se coloca la pieza en la posición correspondiente (Fig 5.17) y se cierra la pinza usando el código cerrar_pinza_linea2. Esta línea accede a la distancia preestablecida (2,20 mm) en la línea 2 en el programa de gestión de la pinza.

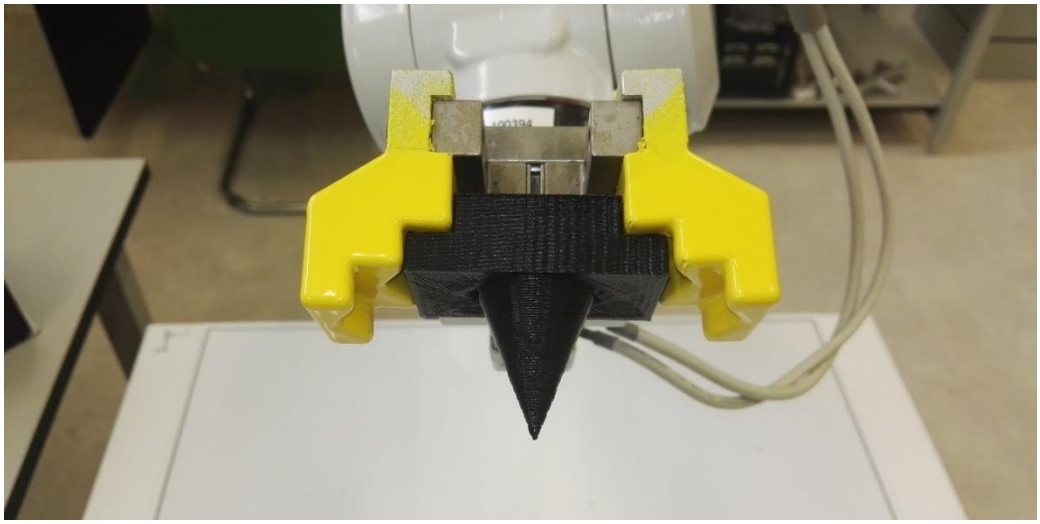


Figura 5.17. Herramienta cono ubicada en su posición en la pinza.

Paso 2: Creación de nuevo TCP o herramienta cono.

Una vez el cono quede sujeto a la pinza procederemos a la creación del nuevo TCP.

Un sistema de coordenadas de la herramienta puede ser definido manualmente o utilizando el robot como elemento de medida. Las definiciones manuales se utilizan cuando se disponen de datos precisos sobre las dimensiones de la herramienta. Como en nuestro caso no disponemos de la medida exacta de la pinza usaremos el robot como elemento de medida.

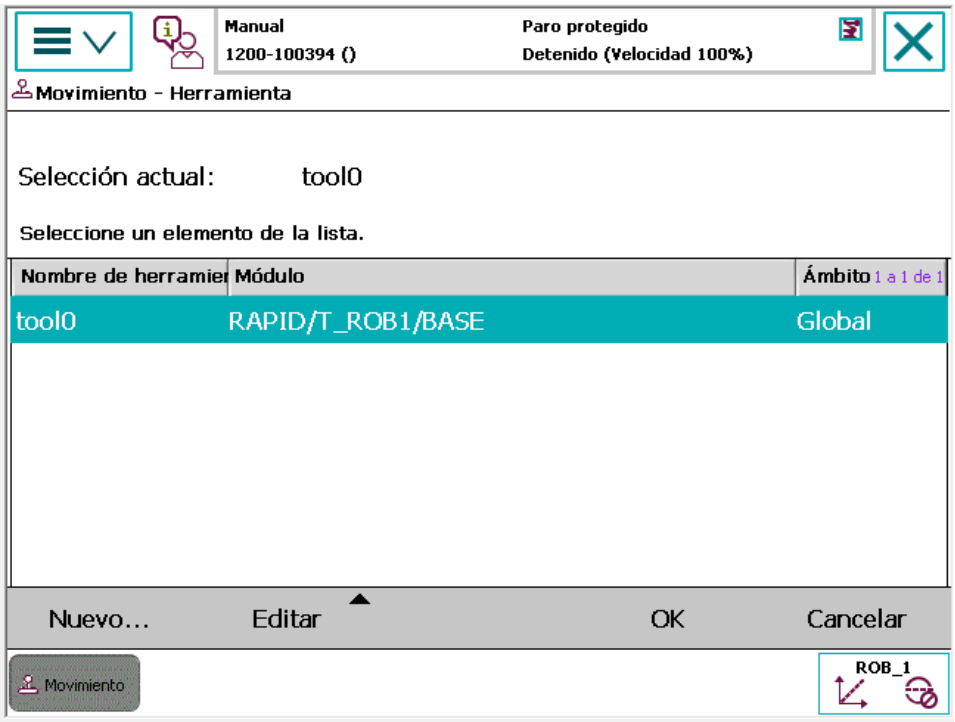


Figura 5.18. Menú de herramientas disponibles en FlexPendant.Tool0.

Las características de la herramienta, como posición y orientación del TCP, y las características físicas de la carga se describen en el tipo de dato tooldata.

Estos datos están definidos a partir de la herramienta tool0 la cual define el sistema de coordenadas de la muñeca y tiene como punto de origen el centro de la brida de montaje.

Por tanto, el primer paso a realizar es seleccionar la herramienta tool0 como la herramienta que se va a utilizar para la definición de la nueva herramienta.

Para ello se empieza con el manejo de la de unidad de programación.

Se accede al menú principal y se abre la ventana de movimientos, donde elegimos:

- Herramienta: tool0
- Objeto de trabajo: wobj0

Como se ha comentado anteriormente, se usa la herramienta y objeto preestablecidos en el robot para definir herramientas y objetos nuevos.

El siguiente paso a realizar es la creación de una nueva herramienta.

- Se selecciona *Herramienta* para ver la lista de herramientas disponibles.
- Se selecciona *Nuevo*… para crear una nueva herramienta.

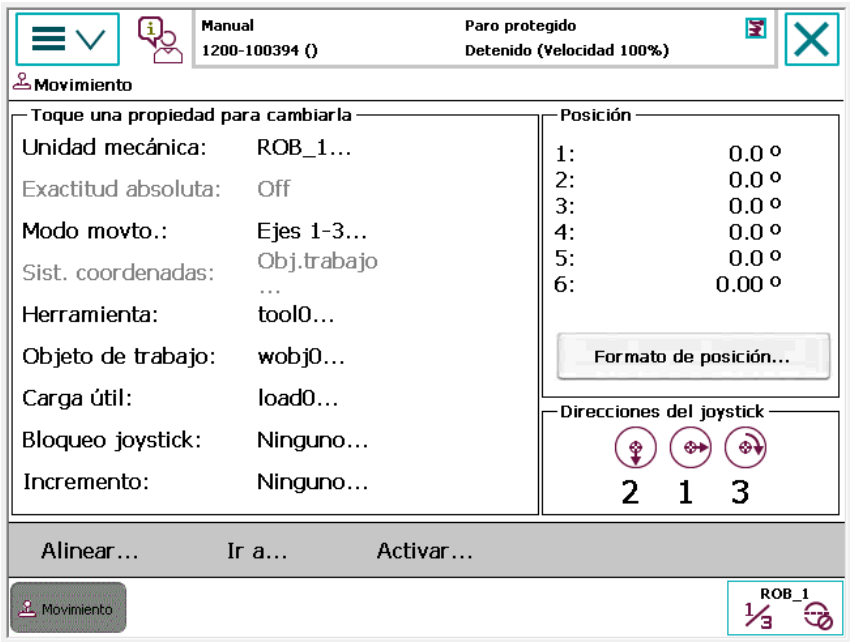


Figura 5.19. Ventana de movimientos en el FlexPendant. Creación de herramienta.

Para la creación de la herramienta se usarán los siguientes datos (Fig 5.20):

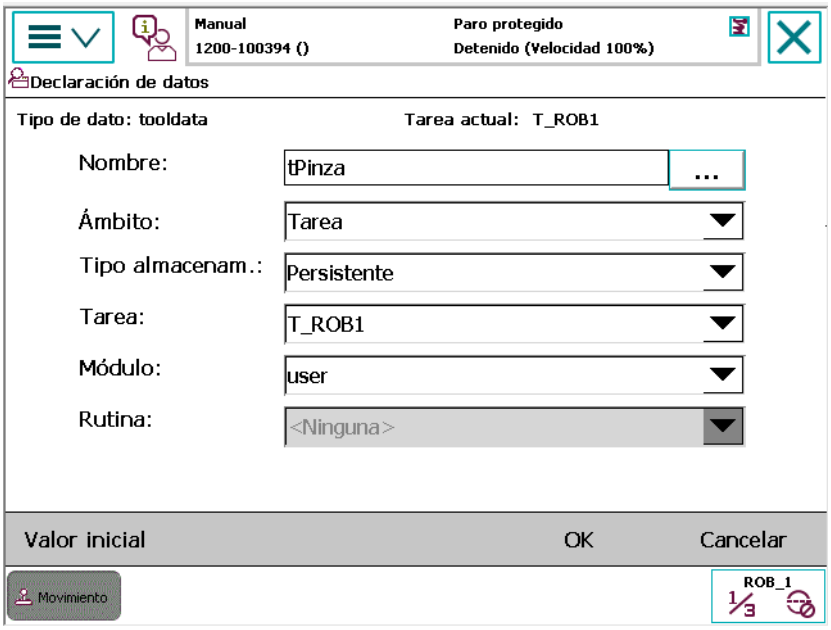


Figura 5.20. Ventana de nueva herramienta en el FlexPendant.

Paso 3: Definición de características de herramienta cono.

La herramienta creada no puede usarse hasta definir sus características (coordenadas del TCP, peso, etc.) debido a que al crearla lo único que se define es su nombre y cómo debe guardarse en el sistema.

Para definir el sistema de coordenadas de la herramienta, se necesita en primer lugar un punto de referencia fijo (en nuestro caso usaremos un pedestal con una varilla acabada en punta que dispone de una buena fijación) situado dentro del área de trabajo del robot.

Figura 5.21. Punto de referencia para declaración de coordenadas de herramienta.

Seguidamente se acerca la punta de la herramienta al punto de referencia fijo anterior con 4 orientaciones distintas del robot, mediante las cuales se calculan las coordenadas del TCP.

Para ello se realizan los siguientes pasos:

- En el menú principal, se selecciona *Datos de programa*.
- Se selecciona *tooldata* y *Mostrar datos* para ver la lista de herramientas disponibles.
- Se selecciona la herramienta *cono* y se selecciona *Editar*.
- En el menú, se selecciona *Definir* y aparece la ventana de definición del sistema de coordenadas.
- Se selecciona el método que se prefiera en el menú emergente *Método.*
- En el caso de este trabajo, el autor sitúa el TCP en un punto alejado de tool0 manteniendo la orientación de los ejes de coordenadas, por lo tanto se elige la opción de *TCP* (orient.predet).

- Se selecciona el número de puntos de aproximación en el menú emergente *Nº de puntos.*

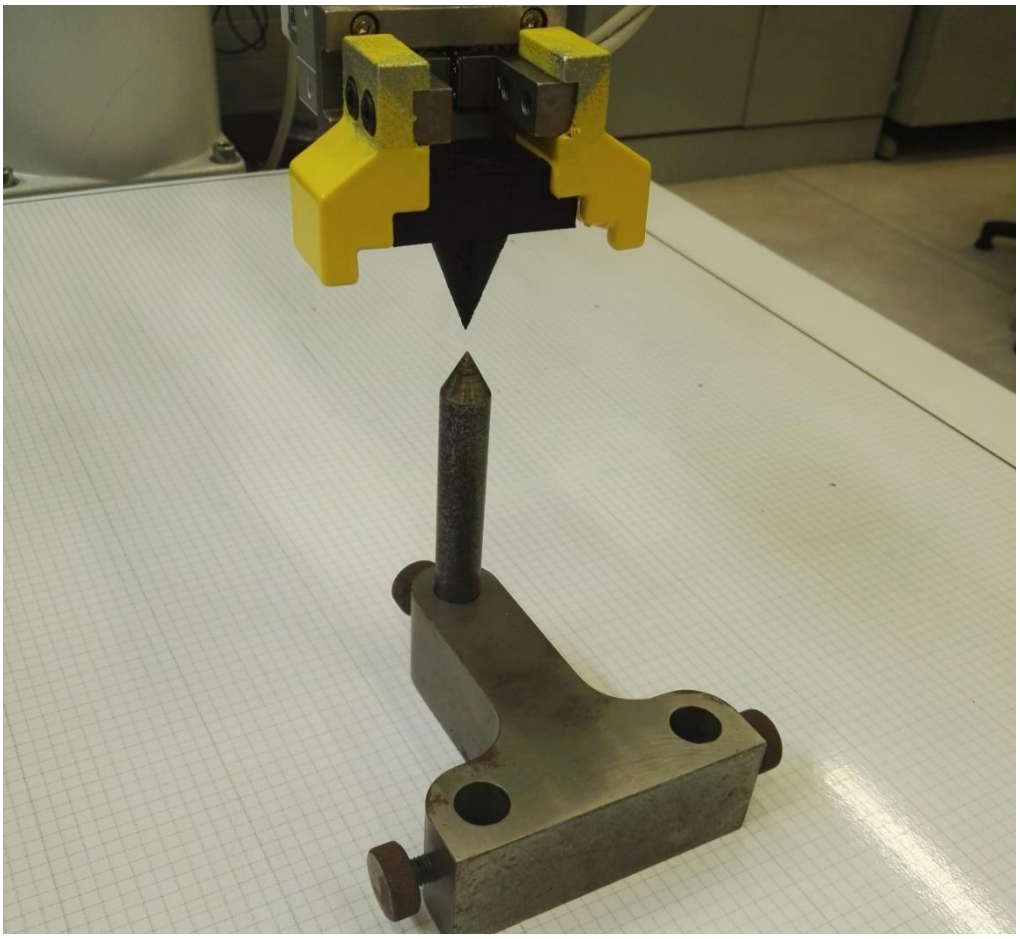


Figura 5.22. Procedimiento para marcado de puntos significativos de herramienta

Para la toma de puntos se procede de la siguiente forma:

- Se mueve el robot hasta una posición adecuada para el punto de aproximación.
- Se selecciona el punto que desea definir y se pulsa *modificar posición*.
- Se repiten los pasos anteriores con los demás puntos de aproximación.
- Cuando se termina de definir los puntos se pulsa OK. No es necesario que estos puntos se guarden en la memoria del robot.

Por último la herramienta no podrá ser utilizada si no se le define una masa. Para ello se realizan los siguientes pasos:

- En el menú principal, se selecciona *Datos de programa*.
- Se selecciona *tooldata* y *Mostrar datos* para ver la lista de herramientas disponibles.
- Se selecciona la herramienta *cono* y se selecciona *Editar*.

- En el menú, se selecciona *cambiar valor* y aparecerá la lista de valores adjudicados a la herramienta a definir.
- Se busca el dato “mass” referente a la masa de la herramienta y se introduce su masa en kilogramos.

Con este paso se completa la definición de la herramienta “cono”, y ya se podrá utilizar en el movimiento del robot y en la definición de un objeto o plano de trabajo.

Antes de pasar a definir el objeto de trabajo se puede definir la pinza del robot en función de los parámetros geométricos de la herramienta “cono”. Para ello se procede de la siguiente forma:

- En el menú principal, se selecciona *Datos de programa*.
- Se selecciona *tooldata* y *Mostrar datos* para ver la lista de herramientas disponibles.
- Se selecciona la herramienta *t_cono* y se selecciona *Copiar*.
- Se abrirá una ventada donde pedirá el nombre de la nueva herramienta, a la cual se llamara en este trabajo “tpinza”.

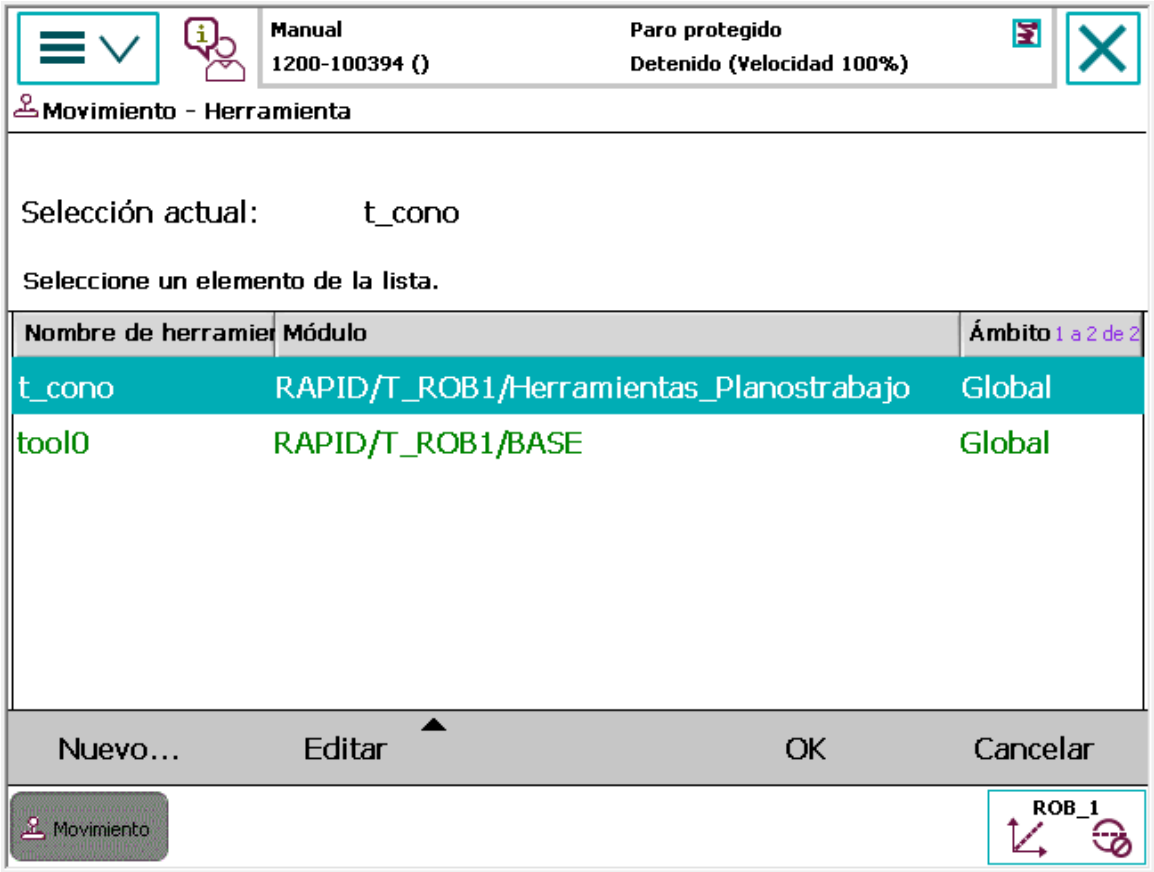


Figura 5.23. Menú de herramientas disponibles en FlexPendant. T_cono.

Esta nueva herramienta tiene los mismos parámetros que la herramienta “cono”.

Paso 4: Creación y definición de herramienta pinza

Para declarar el TCP como un punto central en el agarre de la pinza lo único que se debe realizar es restarle la altura de la herramienta "cono". Esto se realiza de la siguiente forma:

- En el menú principal, se selecciona *Datos de programa*.
- Se selecciona *tooldata* y Mostrar datos para ver la lista de herramientas disponibles.
- Se selecciona la herramienta *tpinza* y se selecciona *Editar*.
- En el menú, se selecciona *cambiar valor* y aparecerá la lista de valores adjudicados a la herramienta a definir.
- Se busca el dato "z" referente a la medida en la dirección "z" de la herramienta y se resta a esta medida 30 mm referentes a la altura del cono.

Este nuevo TCP se encuentra por tanto entre los dos posibles agarres de la pinza, un punto intermedio con el que trabajar con la herramienta.

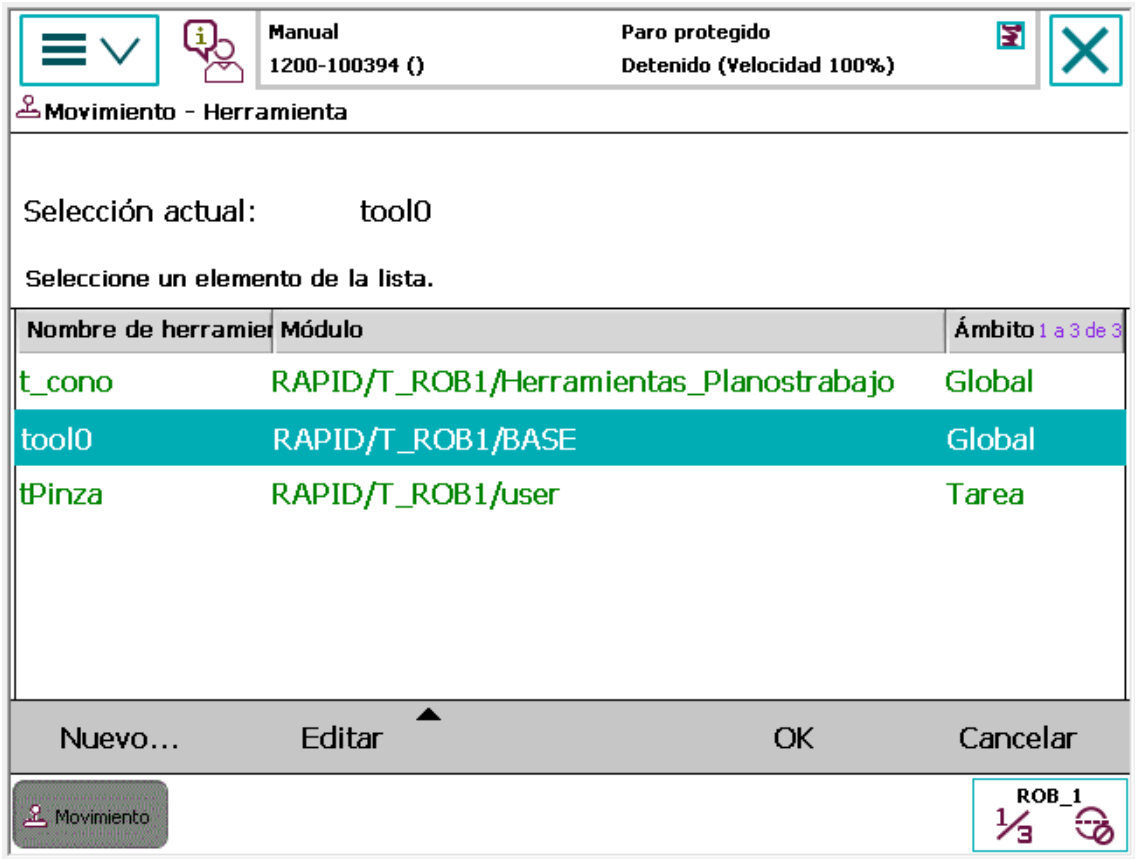


Figura 5.24. Menú de herramientas disponibles en FlexPendant. tPinza

Una vez definidas las herramientas que se van a utilizar se procede a definir el plano de trabajo del robot.

5.2.3 Plano de trabajo

Un plano u objeto de trabajo es un sistema de coordenadas que se utiliza principalmente para simplificar la programación durante la edición de programas debido a los desplazamientos asociados a tareas, objetos, procesos y otros elementos concretos.

La disposición del robot del laboratorio simula la disposición de un robot en el centro de la célula de trabajo. En esta disposición el robot se sitúa de modo que quede rodeado por el resto de elementos que intervienen en la célula. Se trata de una disposición típica para robots de estructura articular, polar, cilíndrica o SCARA, en la que se puede aprovechar al máximo su campo de acción, que presenta una forma básica de esfera.

La disposición del robot en el centro se usa frecuentemente en aquellas aplicaciones en las que el robot debe alcanzar diversos puntos fijos dentro de su área de trabajo.

Paso 1: Adecuación de plano de trabajo

En el caso del robot del laboratorio se define el plano de trabajo como la superficie plana de la que dispone el brazo robótico para operar. Esta mesa de trabajo inicialmente disponía de una bandeja metálica con ranuras para atornillar otros elementos (Fig 5.25).

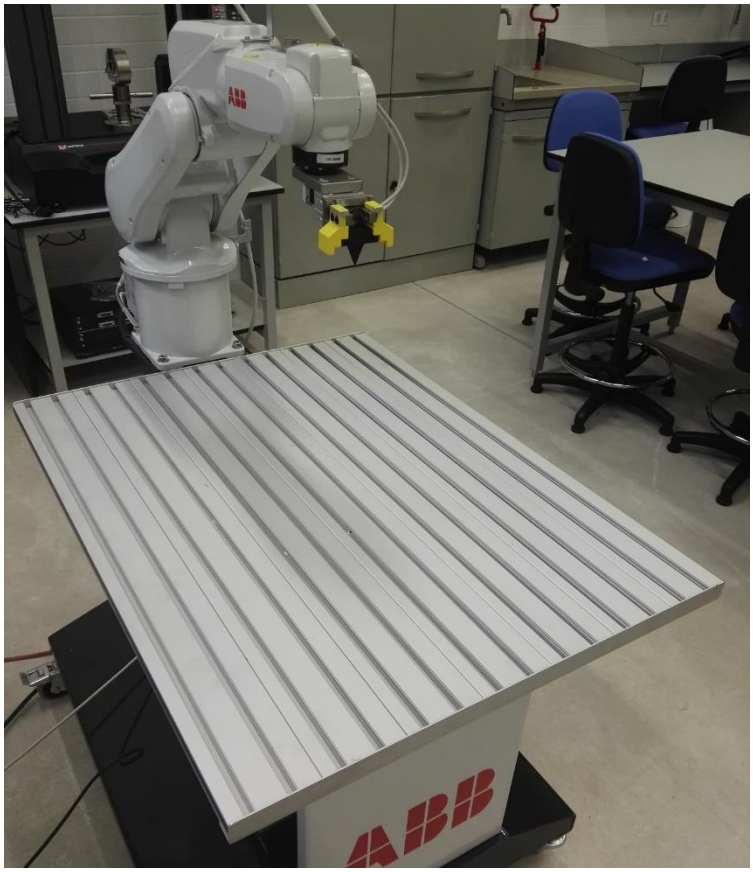


Figura 5.25. Célula robótica IRB120 de la marca ABB

En un primer momento se decide utilizar un tablero de madera contrachapado de 3 mm de espesor para poder disponer de una superficie plana sin ranuras donde poder marcar mejor nuestro plano de trabajo (Figura 5.26). Se hicieron varias pruebas de funcionamiento y se crearon programas en base a dicho plano de trabajo, pero este no estaba a medida, necesitaba estar agarrado mediante gatos y debido a su pequeño espesor no proporcionaba una superficie del todo plana.

Figura 5.26. Cambio de superficie de trabajo por tablero fino.

Para hacer más accesible el uso de la mesa se compró una madera a medida de 7 mm de espesor para poder disponer de una superficie plana y sin agarres (Fig 5.27).

Figura 5.27. Tablero de madera DM a medida.

Más tarde con el objetivo de facilitar la toma de medidas en la mesa y marcar posiciones ortogonales más fácilmente se dispuso un vinilo con cuadricula en la madera (Fig 5.28). La cuadricula dispone de separaciones de 5 mm lo cual permite marcar posiciones de forma más eficiente.

Figura 5.28. Superficie cuadriculada

Paso 2: Creación de plano u objeto de trabajo

Una vez colocado el tablero en su posición final debe ser definido como un objeto de trabajo, el cual será usado para la creación de distintos programas más adelante.

El sistema de coordenadas del objeto de trabajo se corresponde con el plano de trabajo: define el posicionamiento del sistema de coordenadas del plano de trabajo respecto al sistema de coordenadas mundo.

Es en estos sistemas de coordenadas de objetos de trabajo donde se crean los objetivos y trayectorias durante la programación del robot. Con ello se consiguen numerosas ventajas ya que, al reposicionar el objeto de trabajo en la estación, solo es necesario cambiar la posición del sistema de coordenadas del objeto de trabajo para que todas las trayectorias se actualicen a la vez.

Para definir un objeto de trabajo debe procederse de la siguiente manera:

El primer paso a realizar es seleccionar la herramienta “cono” (Fig 5.29) como la herramienta que se va a utilizar para la definición del objeto de trabajo, ya que dispone de una punta final que facilita el marcado de puntos en el plano.

También debe de escogerse “wobj0” (Fig 5.29) como objeto de trabajo, ya que se definirá el nuevo sistema de coordenadas respecto al sistema de coordenadas mundo.

Para ello se empieza con el manejo de la unidad de programación.

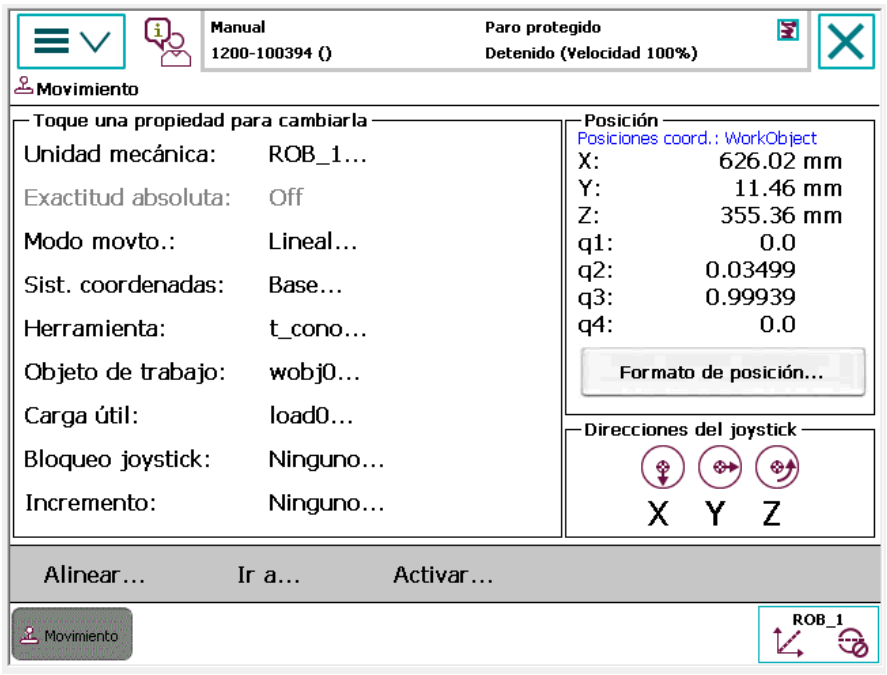


Figura 5.29. Ventana de movimientos en FlexPendant. Creación objeto de trabajo.

El siguiente paso a realizar es la creación de un nuevo objeto de trabajo:

- Seleccione *Objeto de trabajo* para ver la lista de objetos de trabajo disponibles.
- Seleccione *Nuevo…* para crear un nuevo objeto de trabajo.

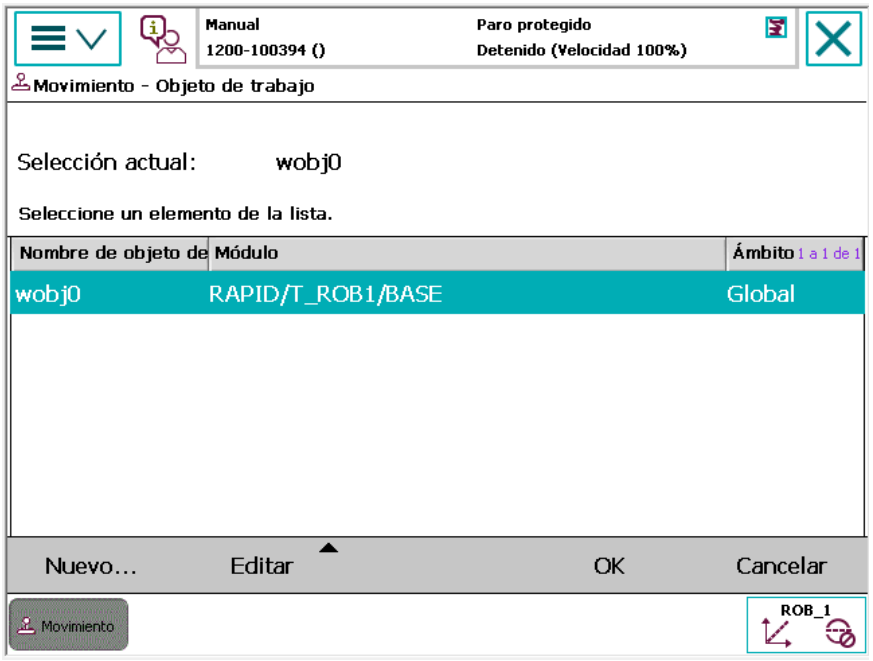


Figura 5.30. Menú de objetos de trabajo disponibles en FlexPendant

Para la definición del nuevo objeto de trabajo, el cual se llamará en este trabajo “wobj_cuadricula”, se usan los siguientes datos:

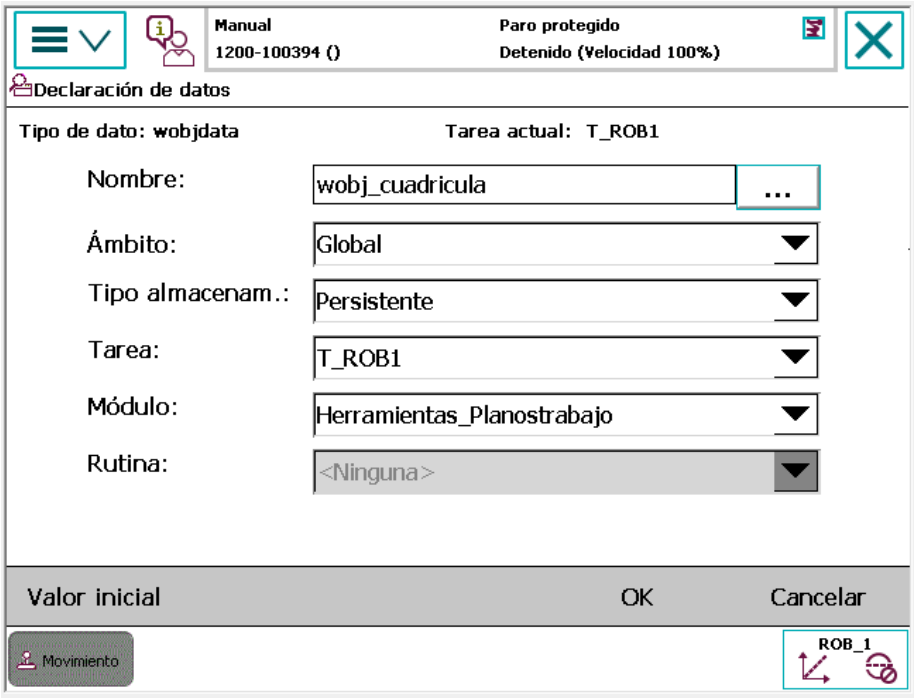


Figura 5.31. Ventana de nuevo objeto de trabajo en FlexPendant

El objeto de trabajo creado no puede usarse hasta definir sus características (centro de sistema de coordenadas y ubicación de sus ejes) debido a que al crearlo lo único que se define es su nombre y cómo debe guardarse en el sistema.

Para ello se realizan los siguientes pasos:

- En el menú principal, se selecciona *Datos de programa*.
- Se selecciona *wobjdata* y *Mostrar datos* para ver la lista de objetos de trabajo disponibles.
- Se selecciona el objeto de trabajo *wobj_cuadricula* y se selecciona *Editar*.
- En el menú, se selecciona *Definir* y aparece la ventana de definición del sistema de coordenadas.
- Se definirá el objeto de trabajo como un sistema de usuario definiéndolo con tres puntos.
- Se mueve el robot con la herramienta *cono* hasta el primer punto X1, el cual marca el centro de coordenadas del nuevo sistema de referencia.
- Se selecciona el punto en la lista y se pulsa *Modificar posición*.
- Se repiten estos pasos con los puntos X2 e Y1.
- Cuando se terminan de definir los puntos se pulsa *OK*. No es necesario que estos puntos se guarden en la memoria del robot.

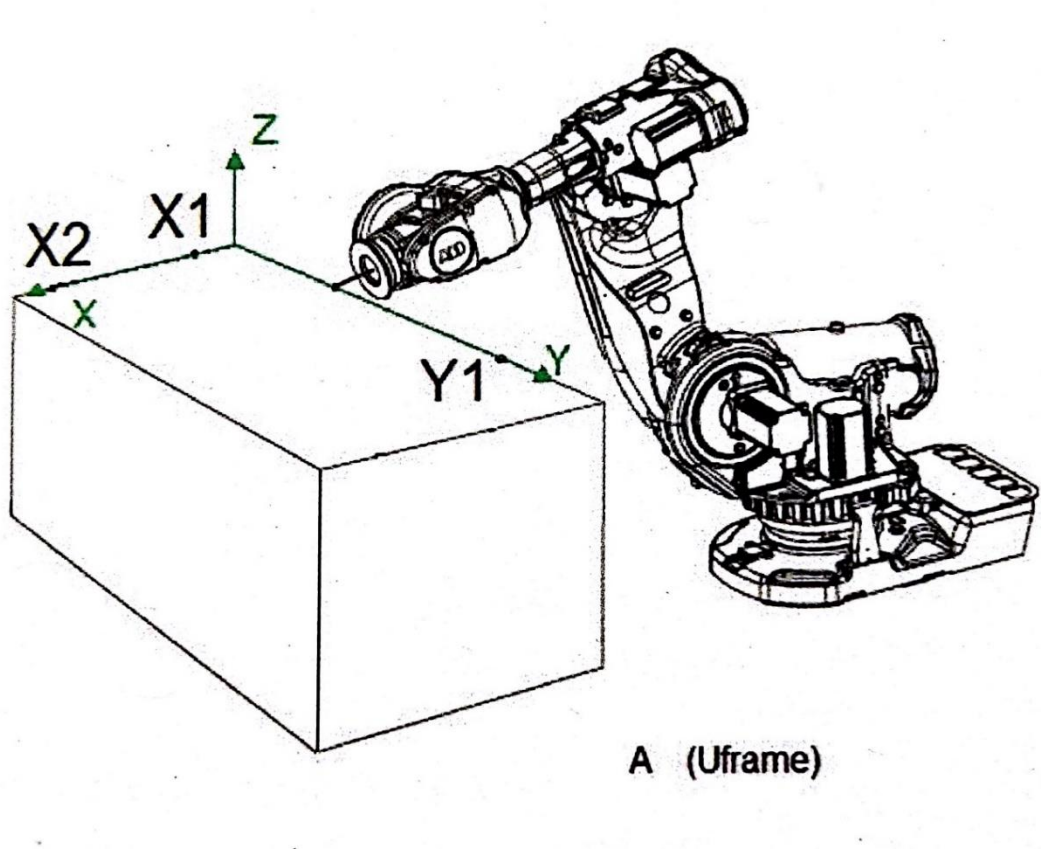


Figura 5.32. Sistema de coordenadas del objeto de trabajo [8]

Con este paso se completa la definición del objeto de trabajo “wobj_cuadricula” y ya se podrá utilizar en el movimiento del robot y se podrá usar como sistema de coordenadas para los movimientos del robot.

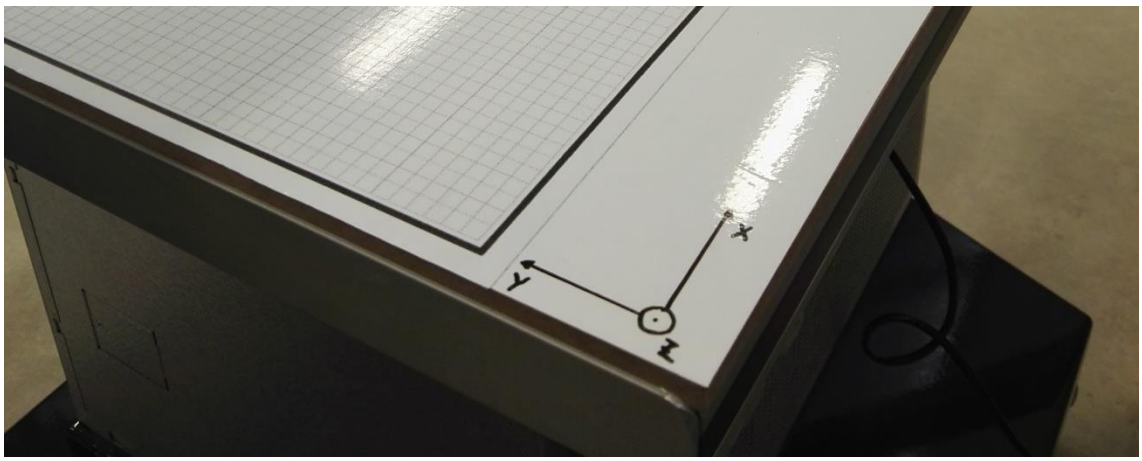


Figura 5.33. Sistema de coordenadas del objeto o superficie de trabajo

5.3 Instrucciones de uso

En este apartado se dan unas instrucciones para el uso diario de la célula robótica, explicándose cómo debe ser encendida, manejo y resolución de errores de la misma. También se explican el método y los sistemas de programación que el autor de este trabajo ha diseñado para facilitar el uso del robot a futuros usuarios. Este apartado es muy necesario debido a que es la primera vez que se dispone de una célula robótica en la Escuela Politécnica Superior de Linares, y no existe una base de conocimientos de robótica firme para su utilización. Este apartado servirá para el uso de la célula a próximos usuarios debido a que no pueden obtener estos conocimientos en ninguna otra parte dentro de la escuela.

5.3.1 Encendido del sistema

En este subapartado se explica cómo debe ser encendida la célula y los instrumentos necesarios para su uso.

- Enchufar robot y ordenador a la corriente.
- Encender robot usando la palanca Power.

Figura 5.34. Palanca Power en Controlador IRC5

- Restaurar copia de seguridad en robot: como parte de este trabajo se ha creado una serie de puntos de restauración predefinidos para el usuario:
    - Back up original: Este es el Backup inicial en el que solo está definido el uso de la pinza.
    - Back up cero: Restaurando a este punto se obtiene una interfaz limpia de RAPID en la que solo se han introducido los datos de herramienta y objeto de trabajo
    - Prácticas: Restaurando a este punto se obtiene el listado de programas para su utilización en las sesiones de prácticas.

- Prácticas 15/16: Restaurando a este punto se obtienen una interfaz con las prácticas realizadas por los alumnos de la asignatura “Mecánica de robots” en el curso académico 2015/16.

Para explicar el uso del programa usaremos la copia de seguridad “Prácticas”.

- Encender ordenador: Encender ordenador de forma normal, pulsando F2 para continuar.
- Abrir RobotStudio: Para acceder a los programas definidos en la memoria del robot y poder modificarlos se debe proceder de la siguiente manera:
  - Controlador.
  - Añadir Controlador > Añade los controladores disponibles en la red.
  - Nombre del sistema >1200>Aceptar.
  - Puerto de servicio>Rapid>T_ROB1>Axel>Practicas.
  - Solicitar acceso de escritura.
  - Aceptar en el FlexPendant.
- Si el FlexPendant muestra error en la pinza, debe procederse de la siguiente manera:
  - Abrir en el ordenador el programa “ACT Controller”
  - Easy Mode
  - Reset

5.3.2 Instrucciones de pinza SMC

La distancia entre los dedos de la pinza está controlada por el software “ACT Controller”, el cual se encuentra instalado en el ordenador. Para acceder a él simplemente debemos pinchar en el acceso directo del escritorio y abrirlo en “Easy Mode”.

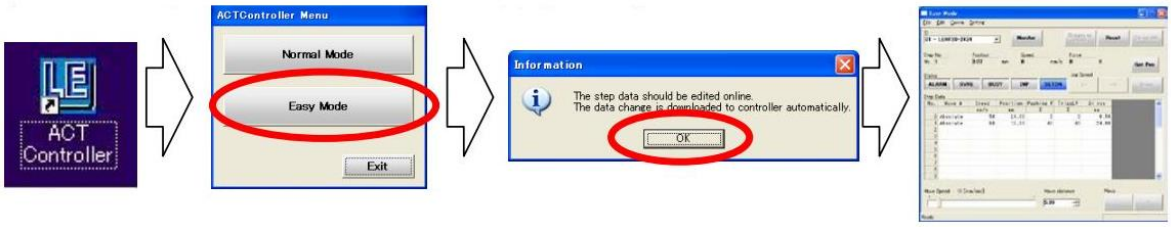


Figura 5.35. Modo de arranque software ACT Controller [14]

Una vez hecho esto se abre la ventana de gestión de movimientos de la pinza.

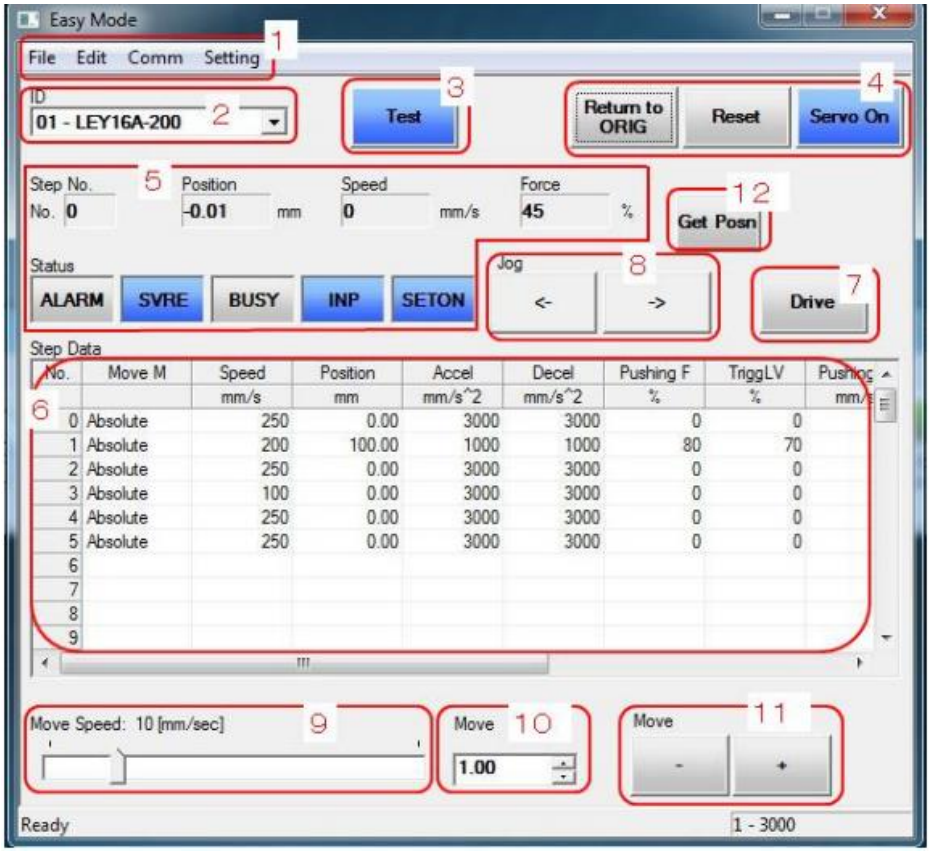


Figura 5.36. Menu Easy Mode del software ACT Controller [15]

1. Menú: Se utiliza para guardar los datos de configuración o modificar las E/S de la pinza. Cuando el programa no detecta la pinza hay que buscarla en *comm*.
2. Indicador de conexión: Hace referencia al eje conectado que se está manipulando ya que pueden existir varios ejes de movimiento conectados a la vez.
3. Botón de cambio de modo: Este es un botón para cambiar entre modo monitor y modo test. No pulsar este botón mientras el actuador está activado.

Modo monitor	Modo test
	
Este modo sirve para comprobar el estado de la pinza, denotando si se encuentra en condición de alarma o en qué posición se encuentra.	Este método sirve para cambiar las posiciones configuradas de apertura.

4. Botones de funciones:
    a. **Return to ORIG:** Sirve para volver a una posición de la pinza después de un desplazamiento de los dedos.
    b. **Servo On/Servo Off:** Sirve para encender o apagar el servo.
    c. **Reset:** Debe pulsarse cuando la pinza se encuentra en modo Alarm.

Figura 5.37. Estado de alarma en ACT Controller [15]

En el caso de que la alarma no desaparezca pulsando el botón Reset, debe cortarse la alimentación del controlador de la pinza y volver a conectar.

Al apagar/encender el controlador robot, o realizar un reinicio desde la unidad de programación, se ejecutará la rutina evento "RCalibracion_Pinza". Esta acción asegurará el retorno de la pinza a la posición de origen (pinza cerrada). **Será necesario extraer cualquier objeto que la pinza sostenga antes de apagar/encender o reiniciar el controlador robot.**

5. Estado: Muestra el estado del controlador mostrando la posición en la que se encuentra la pinza. También aparecen los botones de estado:
    a. **ALARM (Alarma)**: Se vuelve rojo cuando hay algún error.
    b. **SVRE (Servo):** Se vuelve azul cuando el servo está conectado.
    c. **BUSY (Motor):** Se vuelve azul cuando el motor está funcionando.
    d. **INP (En posición):** Se vuelve azul cuando se completa el desplazamiento.
    e. **SET ON (Encender):** Se vuelve azul cuando se completa la acción *Return to the origin.*
6. Datos de posiciones: Se muestran las diferentes posiciones de la pinza guardadas, mostrando la distancia desplazada y la velocidad para llegar a la posición.

7. Botón DRIVE: Sirve para llevar la pinza a una de las posiciones definidas. Para ello clicamos en la posición deseada y posteriormente al botón DRIVE.
8. Jog transfer: Estos dos botones sirven para abrir y cerrar la pinza a una velocidad constante mientras se está pulsando uno de los botones.
9. Velocidad: En este apartado se marca la velocidad de movimiento para los botones “Jog transfer”.
10. Distancia de paso: Es la distancia desplazada cada vez que se pulsan los botones de movimiento por pasos.
11. Botones de movimiento por paso: Al pulsar estos botones la pinza se desplazara la distancia marcada anteriormente cada vez que se pinche sobre el botón.
12. Marcar posición: Con este botón se guarda la distancia de los dedos que se encuentra en ese momento en la posición que se desee en la lista.

La integración del movimiento de la pinza en los programas de RAPID se realiza mediante el siguiente conjunto de códigos:

Código de calibración de pinza:

```
PROC RCalibracion_Pinza()
    WaitTime 0.2;
    Reset DO_SVON;
    !Orden Ejecución Posición de Origen Pinza
    PulseDO\PLength:=1, DO_RESET_PINZA;
    WaitTime 0.2;
    Set DO_SVON;
    WaitTime 1;
    Set DO_SETUP;
    WaitTime 2;
    Reset DO_SETUP;
    WaitTime 0.2;
    Reset DO_SVON;
      ENDPROC
```

Código para abrir la pinza al máximo: dicho código genera un movimiento de la pinza a la posición 1 en la lista de posiciones guardadas en el software, la cual está configurada como 16 mm.

```
PROC Abrir_Pinza()
        Set DO_SVON;
        WaitTime 0.2;
    !Mueve Pinza a la Posición 1 (Apertura)
        SetGO GO_PINZA, 1;
        WaitTime 0.2;
    !Orden Ejecución Posición
        PulseDO\PLength:=0.2, DO_DRIVE;
    TPWrite "";
    TPWrite "";
    TPWrite "          Abriendo Pinza";
        WaitTime 3;
        Reset DO_SVON;
    TPErase;
      ENDPROC
```

Código para cerrar a pinza por completo: dicho código genera un movimiento de la pinza a la posición 0 en la lista de posiciones guardadas en el software, la cual está configurada como 0mm.

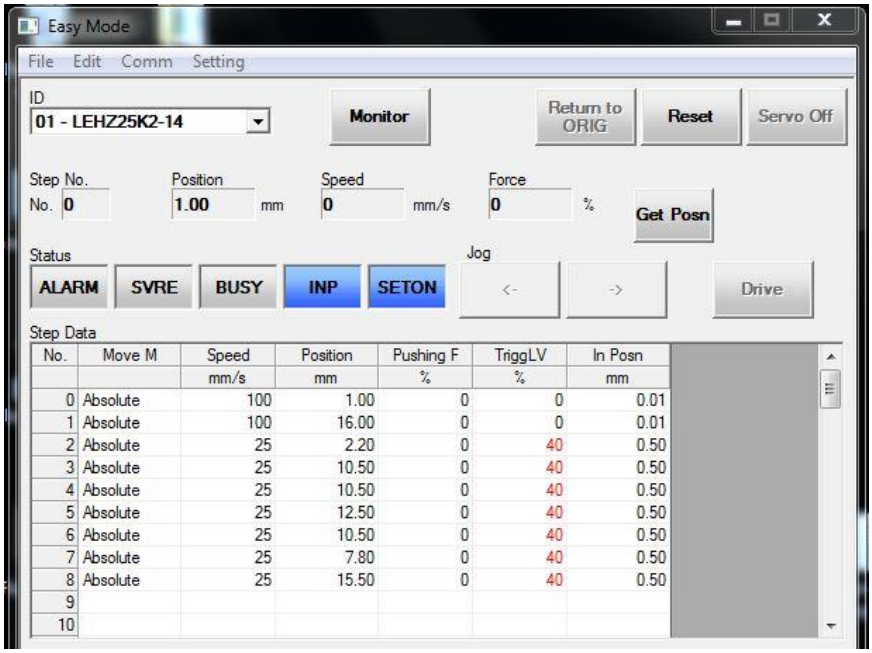
```
PROC Cerrar_Pinza()
        Set DO_SVON;
        WaitTime 0.2;
    !Mueve Pinza a la Posición 0 (Cierre)
        SetGO GO_PINZA, 0;
        WaitTime 0.2;
    !Orden Ejecución Posición
        PulseDO\PLength:=0.2, DO_DRIVE;
    TPWrite "";
    TPWrite "";
    TPWrite "         Cerrando Pinza";
        WaitTime 3;
        Reset DO_SVON;
    TPErase;
      ENDPROC
```

Como se puede ver lo único que cambia el código es la posición a la que accede del software, lo cual se modifica en la línea:

SetGO GO_PINZA, **Posición predefinida**;

Para los demás movimientos de la pinza se ha configurado un módulo en el sistema RAPID, que dispone de todos los códigos necesarios para mover la pinza a las diferentes medidas propuestas en el trabajo. Para ello se ha introducido el código anterior, modificando las posiciones y el nombre del código llamándolo según la línea de referencia en el software de control.

Estas posiciones están definidas para los programas de la práctica y no deben ser cambiadas. Si se necesitan guardar nuevos puntos hay infinitas líneas en el software para ser modificadas.



No.	Move M	Speed	Position	Pushing F	TriggLV	In Posn
		mm/s	mm	%	%	mm
0	Absolute	100	1.00	0	0	0.01
1	Absolute	100	16.00	0	0	0.01
2	Absolute	25	2.20	0	40	0.50
3	Absolute	25	10.50	0	40	0.50
4	Absolute	25	10.50	0	40	0.50
5	Absolute	25	12.50	0	40	0.50
6	Absolute	25	10.50	0	40	0.50
7	Absolute	25	7.80	0	40	0.50
8	Absolute	25	15.50	0	40	0.50
9						
10						

Figura 5.38. Medidas preconfiguradas para la pinza en ACT Controller

Para hacer más fácil el manejo de la pinza durante la programación se han configurado los cuatro botones disponibles en el FlexPendant para abrir y cerrar la pinza simplemente pulsándolos. Esta configuración esta guardada en todos los sistemas creados por el autor de este trabajo.

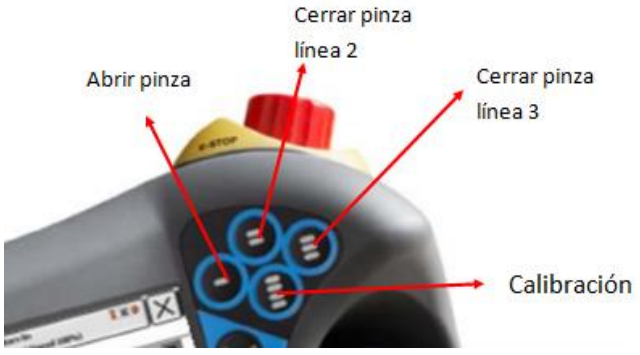


Figura 5.39. Teclas de acceso rápido programables

5.3.3 Programación mediante RAPID

Programar un robot es indicar paso a paso las diferentes acciones que tiene que realizar durante su funcionamiento automático. Para ello la marca ABB hace uso del RAPID, el cual es un lenguaje de programación de alto nivel.

Una aplicación RAPID está compuesta por diferentes programas y módulos de sistema. Un programa básicamente está formado por las siguientes partes:

- Rutina Principal Main: Es la rutina que ejecuta el robot, siguiendo paso a paso las funciones que la componen.
- Subrutinas: Dentro de la rutina principal, se puede hacer referencia a sub-rutinas. Esto sirve para disminuir el tamaño de la rutina y programar de una forma organizada. Además permite la utilización de las subrutinas en diferentes rutinas. Las prácticas diseñadas en este trabajo están realizadas como sub-rutinas de procedimientos en un módulo de sistema diferente (Axel). Para hacer que una de estas subrutinas funcione solo se debe escribir su nombre en la rutina principal Main.
- Datos de programa: Estos datos son necesarios en todos los programas. Sirven para definir las posiciones, sistemas de coordenadas, herramientas, etc. que se necesitan para la ejecución. Pueden estar definidos tanto en la Rutina principal, como en las sub-rutinas, pero siempre deben estar definidos todos los datos antes de las líneas de funciones.

  Estos datos pueden ser definidos según su variabilidad:
  - Constantes (CONS): Valor fijo.
  - Variables (VAR): Puede variar durante la ejecución del programa.
  - Persistentes (PERS): Al cambiar su valor también se cambia su inicialización. En el momento inicial es necesario asignarle un valor.

  También pueden clasificarse según el tipo de dato:
  - Bool: Valores lógicos.
  - Clock: Medición de tiempo.
  - Confdata: Datos de configuración.
  - Jointtarget: Datos de posición de los ejes.
  - Loaddata: Datos de carga.
  - Num: Valores numéricos.
  - Orient: Orientación.
  - Robtarget: Datos de posición.
  - Speeddata: Datos de velocidad.

- String: Cadena de Caracteres.
- Tooldata: Datos de herramienta.
- Wobjdata: Datos del objeto de trabajo.
- Zonedata: Datos de la zona.

- Instrucciones

  A continuación se muestran los distintos comandos para programar en RAPID:

  - Instrucciones de movimiento:
    - **MoveJ** Punto, Velocidad, Zona, Herramienta: El movimiento se realiza de punto a punto sin necesidad de seguir una trayectoria.
    - **MoveL** Punto, Velocidad, Zona, Herramienta: El robot mueve su extremo de forma lineal.
    - **MoveC** Punto Intermedio, Punto Destino, Velocidad, Zona, Herramienta: El robot genera un arco de circunferencia pasando por un punto intermedio hasta el punto de destino, para ello el ángulo del arco debe ser ≤180 º.
    - **Offs** (Punto inicial,Desp x,Desp y,Desp z): Sirve para añadir un desplazamiento a una posición predefinida.
  - Control de flujo:
    - IF

```
IF <condición> THEN
        !Instrucciones;
ELSE
        !Instrucciones;
ENDIF
```

    - WHILE

```
WHILE <condición> DO
        !Instrucciones;
ENDWHILE
```

    - TEST

```
TEST <dato>
  CASE valor1, valor2,..., valor(n-1): rutina1;
  CASE valorn: rutinax;
  DEFAULT
        !Instrucciones;
ENDTEST
```

- Juego total de instrucciones

:=	Asignar un valor
Abs()	Obtener un valor absoluto
AInput()	Leer el valor de una señal de entrada analógica
AccSet	Reducir la aceleración
Add	Sumar un valor numérico
Clear	Borrar un valor
ClkStart	Iniciar un reloj para la toma de tiempos
ClkStop	Parar un reloj para la toma de tiempos
Comment	Comentarios
CompactIF	Si se cumple una condición, entonces… (una instrucción)
ConfJ	Controlar la configuración durante un movimiento articular
ConfL	Monitorizar la configuración del robot durante movimiento
Decr	Decrementar en 1
EXIT	Terminar la ejecución del programa
FOR	Repetir un número de veces
GetTime()	Leer el valor de la hora actual como valor numérico
GOTO	Ir a una nueva instrucción
GripLoad	Definir la carga del robot
HoldMove	Interrumpir el movimiento del robot
IF	Si se cumple una condición, entonces…;de otra manera…
Incr	Incrementar en 1
InvertDO	Invertir el valor de una salida digital
Label	Nombre de una línea
LimConfl	Definir la desviación permitida en la configuración del robot
MoveC	Mover el robot en movimiento circular
MoveJ	Movimiento articular del robot
MoveL	Movimiento del robot en línea recta
Offs()	Desplazamiento de la posición del robot
Open	Apertura de un fichero o de un canal serie
Present()	Comprobar que se utiliza un parámetro opcional
ProcCall	Llamada a un nuevo procedimiento
PulseDO	Generar un pulso en una señal digital de salida
RAISE	Llamada a un manejador de errores
RelMove	Continuar con el movimiento del robot
Reset	Reset de una salida digital
Retry	Recomenzar tras un error
Return	Terminar la ejecución de una rutina
Set	Set de una salida digital
SetAO	Cambar el valor de una salida analógica
SetDO	Cambar el valor de una salida digital
SetGO	Cambiar el valor de un grupo de salidas digitales
SingArea	Definición de la interpolación alrededor de puntos singulares
Stop	Parar la ejecución de un programa
TEST	Dependiendo del valor de la expresión…
TPErase	Borrar el texto de la paleta de programación
TPReadFK()	Leer las teclas de función del a paleta de programación
TPWrite	Escribir en la paleta de programación
VelSet	Cambiar la velocidad programada
WaitDI	Esperar hasta el set de una entrada digital
WaitTime	Esperar un tiempo determinado
WaitUntil	Esperar hasta que se cumpla una condición
WHILE	Repetir mientras…
Write	Escribir en un fichero de caracteres o en un canal serie
WriteBin	Escribir en un canal serie binario

Tabla 5.3. Juego total de instrucciones RAPID [8].

A continuación se explicará el proceso para realizar un programa empleando el Sistema FlexPendant.

Lo primero a realizar es entrar en el editor de programas, donde podremos añadir instrucciones y ver el código de programa. Se accede a esta ventana mediante el menú principal.

Una vez se entre en el editor de programas, y para realizar un programa, se comienza añadiendo las instrucciones que se deseen en el programa. Para ello, primero se clica en la línea donde se quiere añadir el código y se pulsa *añadir instrucción*.

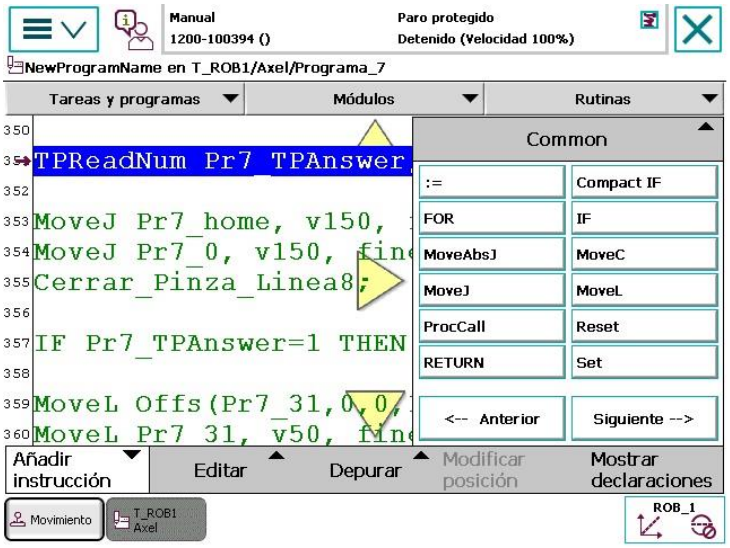


Figura 5.40. Ventana de editor de programas en FlexPendant. Añadir instrucción.

Al pulsar *añadir instrucción* se mostrarán en pantalla las funciones más utilizadas y solo habrá que seguir las instrucciones indicadas para introducir cualquier función.

La manera más fácil de realizar una programación de movimientos es utilizar un guiado activo donde se coloque la herramienta en el punto final del movimiento. Una vez colocado se añade la instrucción del movimiento deseado hasta ese punto, y así sucesivamente.

Una vez creada la línea de código pertinente se puede acceder a ella para modificar sus condiciones, características como la velocidad a la que debe hacer determinado movimiento o el nombre del punto al que se va a mover.

Posteriormente, se siguen añadiendo líneas de código hasta que el programa esté listo para su ejecución, lo cual puede realizarse desde la misma pantalla de editor de programas.

A la hora de ejecutar el programa se debe decir desde dónde se quiere empezar la ejecución. Para ello está el menú Depurar. Al pulsarlo aparece en la ventana un menú con distintas opciones de ejecución.

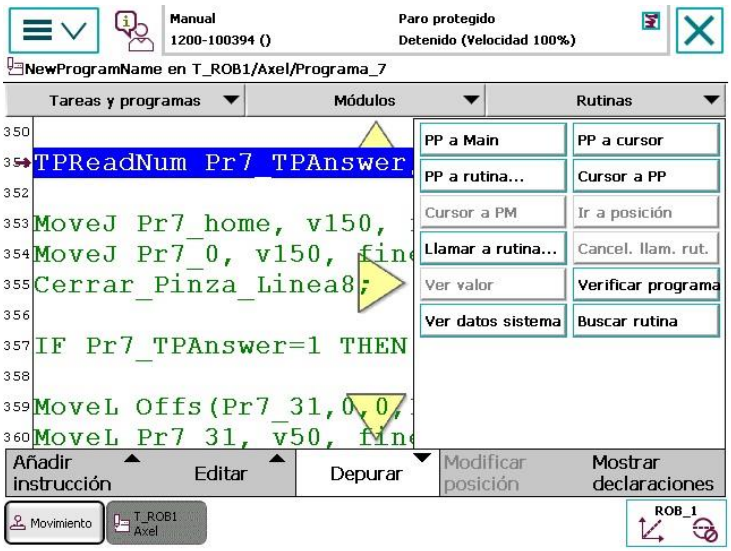


Figura 5.41. Ventana de editor de programas en FlexPendant. Depurar.

Lo más usual es comenzar el programa desde el principio. Esto se realiza pulsando *PP a Main.* Sin embargo, puede que se quiera comenzar la ejecución desde un punto intermedio del programa. Para ello, se debe pinchar en la línea donde se quiere empezar la ejecución y pulsar *PP a cursor.*

5.3.4 Movimiento del robot mediante FlexPendant

Para la ayuda al movimiento del robot, el FlexPendant consta de una ventana llamada “Ventana de movimientos” en la que se puede visualizar y modificar el tipo de movimiento y el sistema de coordenadas de referencia y herramienta usada. También muestra la posición final de la herramienta y los ángulos girados por los ejes.

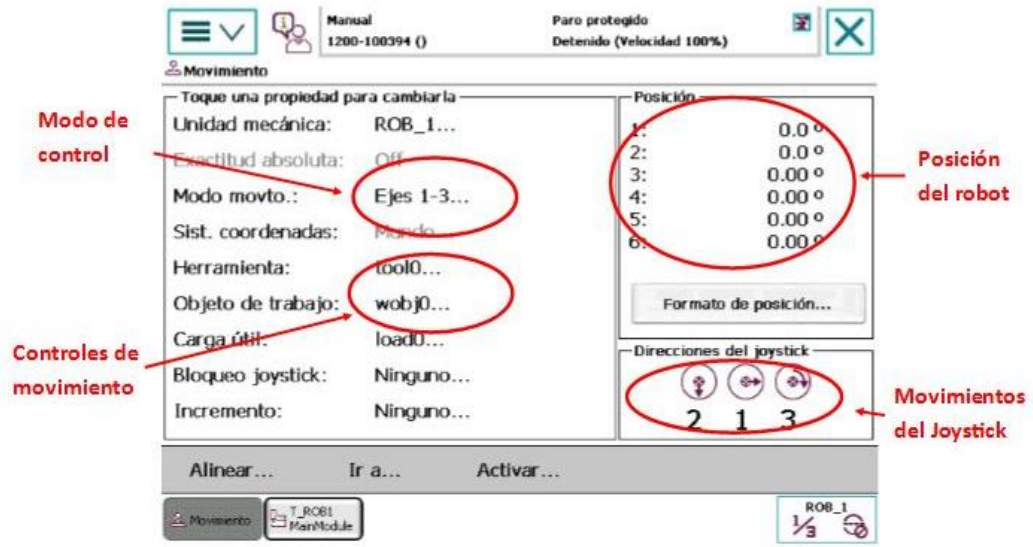


Figura 5.42. Ventana de movimientos en FlexPendant.

6 ESTUDIO CINEMÁTICO Y DINÁMICO DEL BRAZO ROBÓTICO IRB 120

En este apartado se realiza un estudio de la cinemática y dinámica del robot, haciendo uso de los fundamentos teóricos explicados anteriormente. Para hacer más amena la explicación, en ocasiones, se utilizarán simplificaciones en la formulación tales que:

$$Cos(\theta_1) = C\theta_1 = C_1$$
$$Sen(\theta_1) = S\theta_1 = S_1$$
$$Cos(\theta_1 + \theta_2) = C(\theta_1 + \theta_2) = C_{12}$$
$$Sen(\theta_1 + \theta_2) = S(\theta_1 + \theta_2) = S_{12}$$

6.1 Cinemática directa

Como se ha comentado en el capítulo de Fundamentos teóricos, el estudio del problema cinemático directo nos ofrece obtener una relación entre las coordenadas del extremo del robot en función de las coordenadas articulares. Para ello se empieza el análisis estudiando la geometría del brazo robótico de estudio. Las medidas utilizadas del robot se han obtenido de la página web del fabricante [12] y han sido comprobadas con el robot del laboratorio.

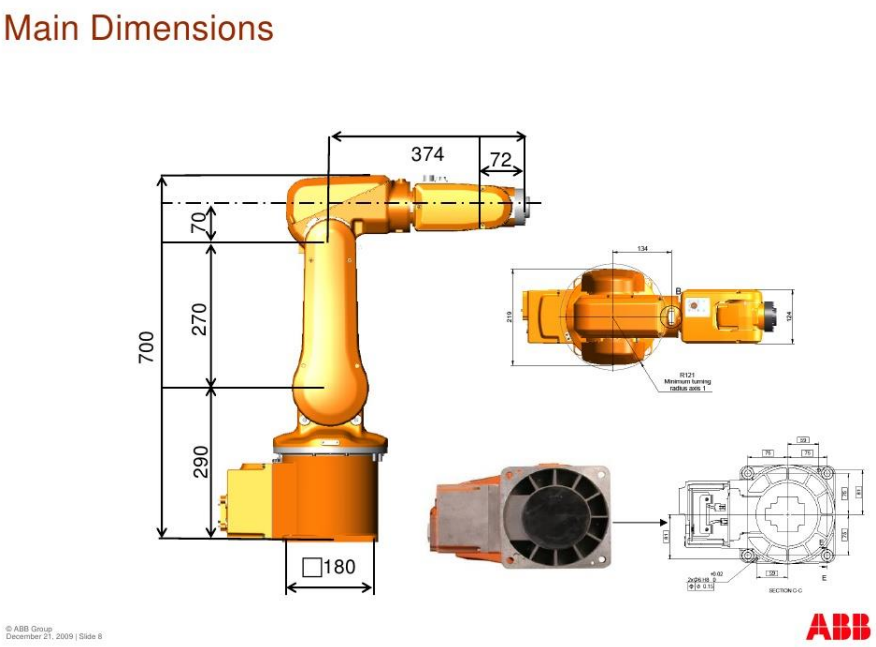


Figura 6.1. Dimensiones robot IRB 120 [12]

Una vez comprobadas las medidas (Fig 6.1) se continúa el análisis identificando y numerando los eslabones del robot. Como se ha comentado anteriormente, el robot IRB 120 consta de 6 GDL y 6 eslabones, los cuales pueden diferenciarse en la figura 6.2.

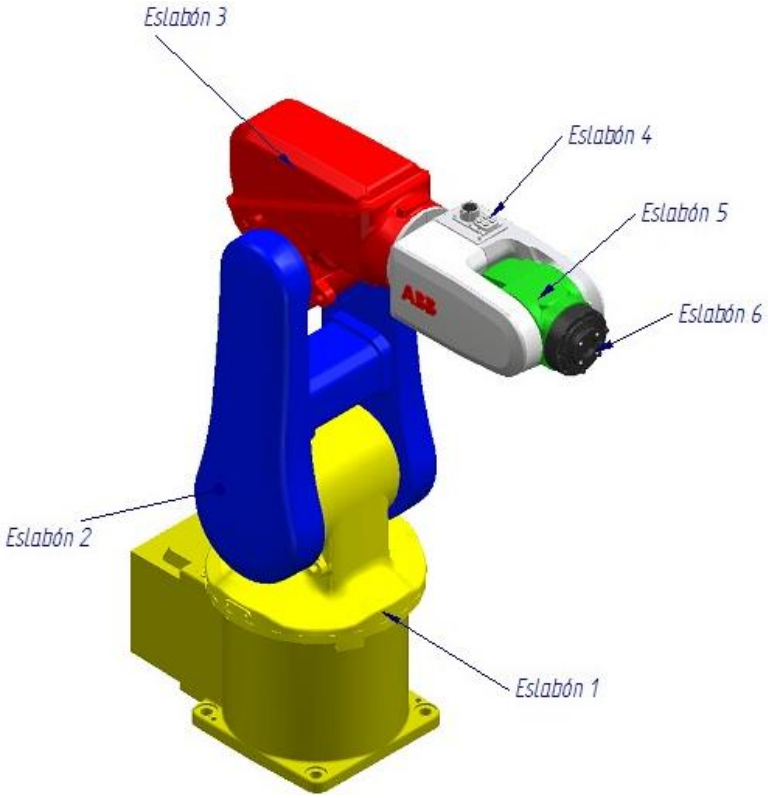


Figura 6.2. Eslabones de robot IRB 120, figura dibujada con SolidEdge.

Todas las articulaciones de este robot son de tipo rotatorio.

El giro del eslabón 1 no contempla la base del robot (parte amarilla oscura), pero el autor de este trabajo lo considera como un único eslabón (Eslabón 1, amarillo oscuro y claro), debido a que ésta es la medida que de la que se dispone y no resulta relevante a la hora del cálculo de posicionamiento del robot.

Una vez identificados los eslabones se procede a localizar los sistemas de referencia de cada una de las articulaciones mediante el método de Denavit-Hartenberg.

Desde su posición inicial, se van colocando los ejes desde la base hasta el extremo del robot, asignando el 0 a la base, y *n-1* para la última articulación, siendo *n* el número de GDL o articulaciones. El sistema de coordenadas *n* se ubicará en el extremo del robot orientando el eje **Z** según el sentido de la aproximación del extremo.

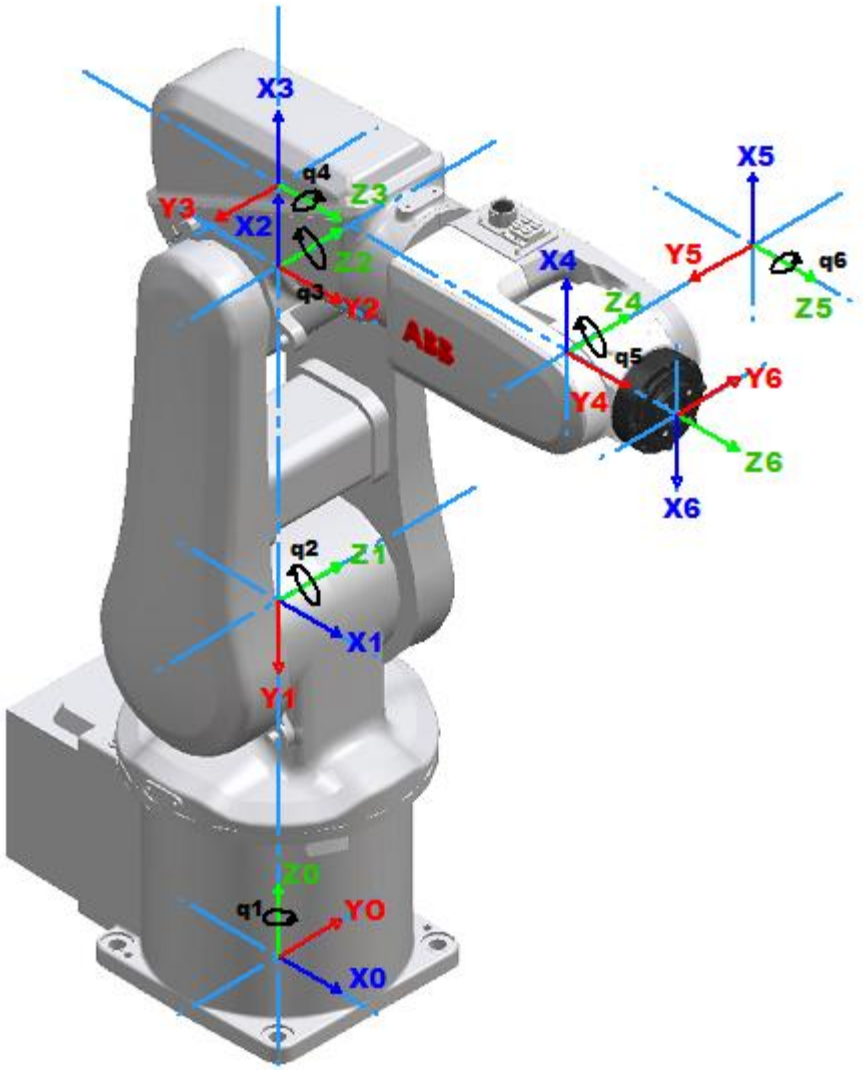


Figura 6.3. Sistemas de coordenadas según Denavit-Hartenberg.

Posteriormente, utilizando las medidas de las articulaciones y la ubicación de los sistemas de coordenadas, se determinan los parámetros de Denavit-Hartenberg del robot, con los que se construye la tabla 6.1: (θ_i, d_i ,a_i ,α_i)

Articulación	θ (Grados)	d (mm)	a (mm)	α (Grados)
1	θ_1	290	0	-90
2	θ_2-90	0	270	0
3	θ_3	0	70	-90
4	θ_4	302	0	90
5	θ_5	0	0	-90
6	θ_6-180	72	0	0

Tabla 6.1. Parámetros según Denavit-Hartenberg de robot IRB 120

Una vez calculados los parámetros de cada eslabón se calculan las matrices $^{i-1}\mathbf{A_i}$, sustituyendo en la ecuación general (Ec 4.3) :

$$^{i-1}\boldsymbol{A_i} = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{0}\boldsymbol{A_1} = \begin{bmatrix} C\theta_1 & 0 & -S\theta_1 & 0 \\ S\theta_1 & 0 & C\theta_1 & 0 \\ 0 & -1 & 0 & 290 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}\boldsymbol{A_2} = \begin{bmatrix} C(\theta_2 - 90) & -S(\theta_2 - 90) & 0 & 270 \cdot C(\theta_2 - 90) \\ S(\theta_2 - 90) & C(\theta_2 - 90) & 0 & 270 \cdot S(\theta_2 - 90) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S(\theta_2) & C(\theta_2) & 0 & 270 \cdot S(\theta_2) \\ -C(\theta_2) & S(\theta_2) & 0 & -270 \cdot C(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}\boldsymbol{A_3} = \begin{bmatrix} C\theta_3 & 0 & -S\theta_3 & 70 \cdot C\theta_3 \\ S\theta_3 & 0 & C\theta_3 & 70 \cdot S\theta_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{3}\boldsymbol{A_4} = \begin{bmatrix} C\theta_4 & 0 & S\theta_4 & 0 \\ S\theta_4 & 0 & -C\theta_4 & 0 \\ 0 & 1 & 0 & 302 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}\boldsymbol{A_5} = \begin{bmatrix} C\theta_5 & 0 & -S\theta_5 & 0 \\ S\theta_5 & 0 & C\theta_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{5}\boldsymbol{A_6} = \begin{bmatrix} C(\theta_6 - 180) & -S(\theta_6 - 180) & 0 & 0 \\ S(\theta_6 - 180) & C(\theta_6 - 180) & 0 & 0 \\ 0 & 0 & 1 & 72 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -C(\theta_6) & S(\theta_6) & 0 & 0 \\ -S(\theta_6) & -C(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & 72 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Así pues, se obtiene la matriz T que indica la localización del sistema final con respecto al sistema de referencia de la base del robot, como producto del conjunto de matrices:

$$\boldsymbol{T} = {}^{\mathbf{0}}\boldsymbol{A_1} \cdot {}^{\mathbf{1}}\boldsymbol{A_2} \cdot {}^{\mathbf{2}}\boldsymbol{A_3} \cdot {}^{\mathbf{3}}\boldsymbol{A_4} \cdot {}^{\mathbf{4}}\boldsymbol{A_5} \cdot {}^{\mathbf{5}}\boldsymbol{A_6} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(Ec 6.1)}$$

A continuación se desarrollan los términos de la matriz:

$$n_x = -C_6(C_5(S_1S_4 + C_4C_1S_{23}) + S_5C_1C_{23}) - S_6(C_4S_1 - S_4C_1S_{23})$$
$$n_y = -C_6(C_5(C_1S_4 - C_4S_1S_{23}) - S_5S_1C_{23}) + S_6(C_4C_1 + S_4S_1S_{23})$$
$$n_z = S_6C_{23}S_4 + C_6(S_{23}S_5 - C_{23}C_4C_{5)}$$

$$o_x = S_6(C_5(S_1S_4 + C_4C_1S_{23}) + S_5C_1C_{23}) - C_6(C_4S_1 - S_4C_1S_{23})$$
$$o_y = -S_6(C_5(C_1S_4 - C_4S_1S_{23}) - S_5S_1C_{23}) + C_6(C_4C_1 + S_4S_1S_{23})$$
$$o_z = C_6C_{23}S_4 - S_6(S_{23}S_5 - C_{23}C_4C_{5)}$$

$$a_x = -S_5(S_1S_4 + C_4C_1S_{23}) + C_5C_1C_{23}$$
$$a_y = -S_5(C_1S_4 - C_4S_1S_{23}) + C_5S_1C_{23}$$
$$a_z = C_5S_{23} - C_4S_5C_{23}$$

$$p_x = 72 \cdot (C_5C_1C_{23} - S_5(S_1S_4 + C_4C_1S_{23})) + C_1(70 \cdot S_{23} + 302 \cdot C_{23} + 270 \cdot S_2)$$
$$p_y = 72 \cdot (S_5(C_1S_4 - C_4S_1S_{23}) + C_5S_1C_{23}) + S_1(270 \cdot S_2 + 70 \cdot S_{23} + 302 \cdot C_{23})$$
$$p_z = -302 \cdot S_{23} + 70 \cdot C_{23} + 270 \cdot C_2 - 72 \cdot S_{23}C_5 + 290$$

Para este cálculo se usará el siguiente código introducido en el programa MATLAB:

Código para el cálculo simbólico de la matriz

```
clear all
close all

%Eslabón 1
syms theta1
d=290;
alpha=-90;
a=0;
theta=theta1;
A1=[cos(theta)  -sin(theta)*cosd(alpha)  sin(theta)*sind(alpha)  a*cos(theta)  ;   sin(theta)
cosd(alpha)*cos(theta)  -cos(theta)*sind(alpha)  a*sin(theta)  ; 0 sind(alpha)  cosd(alpha)  d
; 0 0 0 1]

%Eslabón 2
syms theta2
d=0;
alpha=0;
a=270;
theta=theta2-90;
A2=[cos(theta)  -sin(theta)*cosd(alpha)  sin(theta)*sind(alpha)  a*cos(theta)  ;   sin(theta)
cosd(alpha)*cos(theta)  -cos(theta)*sind(alpha)  a*sin(theta)  ; 0 sind(alpha)  cosd(alpha)  d
; 0 0 0 1]

%Eslabón 3
syms theta3
d=0;
alpha=-90;
a=70;
theta=theta3;
A3=[cos(theta)  -sin(theta)*cosd(alpha)  sin(theta)*sind(alpha)  a*cos(theta)  ;   sin(theta)
cosd(alpha)*cos(theta)  -cos(theta)*sind(alpha)  a*sin(theta)  ; 0 sind(alpha)  cosd(alpha)  d
; 0 0 0 1]

%Eslabón 4
syms theta4
d=302;
alpha=90;
a=0;
theta=theta4;
A4=[cos(theta)  -sin(theta)*cosd(alpha)  sin(theta)*sind(alpha)  a*cos(theta)  ;   sin(theta)
cosd(alpha)*cos(theta)  -cos(theta)*sind(alpha)  a*sin(theta)  ; 0 sind(alpha)  cosd(alpha)  d
; 0 0 0 1]

%Eslabón 5
syms theta5
d=0;
alpha= -90;
a=0;
theta=theta5;
A5=[cos(theta)  -sin(theta)*cosd(alpha)  sin(theta)*sind(alpha)  a*cos(theta)  ;   sin(theta)
cosd(alpha)*cos(theta)  -cos(theta)*sind(alpha)  a*sin(theta)  ; 0 sind(alpha)  cosd(alpha)  d
; 0 0 0 1]

%Eslabón 6
syms theta6
d=72;
alpha=(0);
a=0;
theta=theta6-180;
A6=[cos(theta)  -sin(theta)*cosd(alpha)  sin(theta)*sind(alpha)  a*cos(theta)  ;   sin(theta)
cosd(alpha)*cos(theta)  -cos(theta)*sind(alpha)  a*sin(theta)  ; 0 sind(alpha)  cosd(alpha)  d
; 0 0 0 1]

%Matriz de Conversión

T=A1*A2*A3*A4*A5*A6
```

Esta matriz es genérica, aparecen en ella los ángulos θ sin sustituir. Para el cálculo de una matriz exacta se debe utilizar el siguiente código, introduciendo los ángulos en grados:

Código para el cálculo de la matriz exacta

```
clear all
close all
%Primero definimos los valores de los ángulos theta(grados)
theta1=0; theta2=0; theta3=0; theta4=0; theta5=0; theta6=0;


 %Eslabón 1
d=290;
alpha=-90;
a=0;
theta=theta1;
A1=[cosd(theta)    -sind(theta)*cosd(alpha)    sind(theta)*sind(alpha)    a*cosd(theta)    ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 2
d=0;
alpha=0;
a=270;
theta=theta2-90;
A2=[cosd(theta)    -sind(theta)*cosd(alpha)    sind(theta)*sind(alpha)    a*cosd(theta)    ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 3
d=0;
alpha=-90;
a=70;
theta=theta3;
A3=[cosd(theta)    -sind(theta)*cosd(alpha)    sind(theta)*sind(alpha)    a*cosd(theta)    ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 4
d=302;
alpha=90;
a=0;
theta=theta4;
A4=[cosd(theta)    -sind(theta)*cosd(alpha)    sind(theta)*sind(alpha)    a*cosd(theta)    ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 5
d=0;
alpha= -90;
a=0;
theta=theta5;
A5=[cosd(theta)    -sind(theta)*cosd(alpha)    sind(theta)*sind(alpha)    a*cosd(theta)    ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 6
d=72;
alpha=(0);
a=0;
theta=theta6-180;
A6=[cosd(theta)    -sind(theta)*cosd(alpha)    sind(theta)*sind(alpha)    a*cosd(theta)    ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Matriz de Conversión
T=A1*A2*A3*A4*A5*A6
```

Se ha introducido como ejemplo la posición original (ángulos=0) para demostrar el funcionamiento de la matriz.

El resultado de la matriz de conversión del brazo robot es el siguiente:

$$T = \begin{bmatrix} 0 & 0 & 1 & 374 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 630 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Esta matriz está dividida en dos partes, la primera matriz (3x3) expresa el giro del sistema de coordenadas del extremo con el de la base, y la segunda matriz (3x1) que indica la ubicación del sistema de coordenadas del extremo respecto a la base.

Como se puede comprobar en las figuras 6.2 y 6.3 estos datos son correctos.

La marca ABB usa los cuaternios para definir la orientación de la herramienta y por ello se procede a su cálculo usando los datos de la matriz de transformación. Para ello, se introducirán en la rutina desarrollada en Matlab las ecuaciones vistas en el apartado de cuaternios (Ec 4.5), en las que se introducirán los valores de la matriz de transformación homogénea del robot.

Siendo:

$$T = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot {}^3A_4 \cdot {}^4A_5 \cdot {}^5A_6 = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Se obtienen los valores de los cuaternios:

$$q_1 = \frac{\sqrt{n_x + o_y + a_z + 1}}{2}$$

$$q_2 = \frac{\sqrt{n_x - o_y - a_z + 1}}{2} \qquad signo\ q_2 = signo(o_z - a_y)$$

$$q_3 = \frac{\sqrt{-n_x + o_y - a_z + 1}}{2} \qquad signo\ q_3 = signo(a_x - n_z)$$

$$q_4 = \frac{\sqrt{-n_x - o_y + a_z + 1}}{2} \qquad signo\ q_4 = signo(n_y - o_x)$$

Usando el siguiente código se calcula dicho vector:

```
%Columna de coordenadas cartesianas
Coordenadas_TCP=T(:,4);
%Cuaternios
q1=0.5*sqrt(T(1,1)+T(2,2)+T(3,3)+1);
q2=sign(T(3,2)-T(2,3))*0.5*sqrt(T(1,1)-T(2,2)-T(3,3)+1);
q3=sign(T(1,3)-T(3,1))*0.5*sqrt(-T(1,1)+T(2,2)-T(3,3)+1);
q4=sign(T(2,1)-T(1,2))*0.5*sqrt(-T(1,1)-T(2,2)+T(3,3)+1);
 cuaternios_vector=[q1;q2;q3;q4];
%Posición y orientación TCP
Pos_orien=[Coordenadas_TCP;cuaternios_vector]
```

6.2 Cinemática inversa

El problema de la cinemática inversa ha sido muy complejo en este trabajo, debido a la poca información en forma de ejemplos de la bibliografía existente. Además, el modelo cinemático inverso explicado normalmente en la bibliografía [1,2,7] consultada utiliza modelos de robot en los que el sistema de coordenadas {$\mathbf{S_2}$} y {$\mathbf{S_3}$} se encuentran en el mismo punto, lo que facilita los cálculos debido a que elimina parámetros que en el problema del robot IRB 120, al estar separados, complica notablemente el cálculo.

Sin embargo, el autor de este trabajo ha desarrollado una ingeniosa solución, de las diferentes posibles, gracias al método de desacoplo cinemático [1], con el que primero se calculan los tres primeros grados de libertad usando métodos geométricos y los tres últimos grados haciendo uso de las matrices de rotación de los eslabones.

Por lo tanto se deben iniciar estos cálculos expresando la posición de la muñeca (**pm**) en función del extremo final (**p**). Puesto que la dirección del eje $\mathbf{z_6}$ coincide con $\mathbf{z_5}$, y la distancia entre sus ejes es conocida, se pueden relacionar con la siguiente expresión:

$$\overrightarrow{\boldsymbol{pm}} = \vec{\boldsymbol{p}} - d_6 \cdot \vec{\boldsymbol{z}}_{\boldsymbol{6}}$$

$$\begin{bmatrix} pm_x \\ pm_y \\ pm_z \end{bmatrix} = \begin{bmatrix} p_x - d_6 \cdot a_x \\ p_y - d_6 \cdot a_y \\ p_z - d_6 \cdot a_z \end{bmatrix}$$

Para empezar, debe obtenerse la matriz homogénea de posición y orientación del extremo del robot. Esta matriz **T** no puede ser calculada por el método usado en la cinemática directa debido a que no se conocen los parámetros articulares. Para calcularla se hace uso de las coordenadas de la herramienta y de los cuaternios, los cuales expresan los parámetros de orientación de la misma. La representación de la matriz de transformación en función de los componentes de un cuaternio y de su posición viene dada por la siguiente matriz [1]:

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = 2 \begin{bmatrix} {q_1}^2 + {q_2}^2 - \frac{1}{2} & q_2 q_3 - q_4 q_1 & q_2 q_4 + q_3 q_1 & \frac{p_x}{2} \\ q_2 q_3 + q_4 q_1 & {q_1}^2 + {q_3}^2 - \frac{1}{2} & q_3 q_4 + q_2 q_1 & \frac{p_y}{2} \\ q_2 q_4 - q_3 q_1 & q_3 q_4 + q_2 q_1 & {q_1}^2 + {q_4}^2 - \frac{1}{2} & \frac{p_z}{2} \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

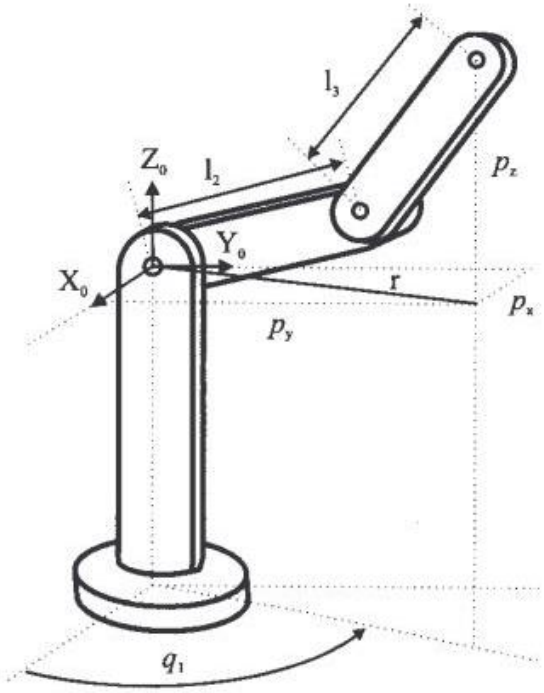


Figura 6.4. Simplificación de base de robot articular [1]

La primera coordenada articular se calcula de forma sencilla. En la figura 6.4 se puede observar que el valor de θ_1 se obtiene inmediatamente como:

$$\theta_1 = arctg\left(\frac{pm_y}{pm_x}\right)$$

Obtenida la primera incógnita, se consideran ahora únicamente los eslabones 2 y 3, que quedan contenidos en el plano de la figura 6.5.

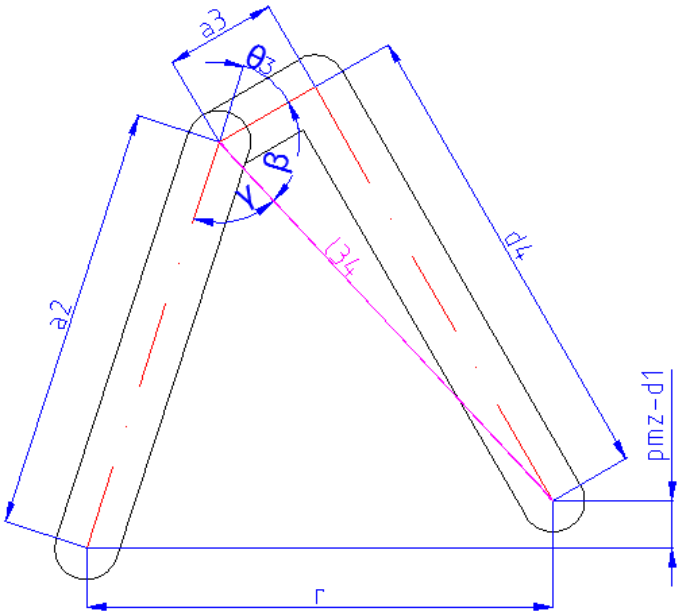


Figura 6.5. Esquema gráfico para la determinación de θ3.

Se calculan primero los valores geométricos directos de la figura 6.5:

$$\beta = arctg\left(\frac{d_4}{a_3}\right)$$

$$l_{34} = \sqrt{{a_3}^2 + {d_4}^2}$$

$$r = \sqrt{{pm_x}^2 + {pm_y}^2}$$

En este caso, el sistema de referencia tomado para los cálculos se encuentra en la primera articulación, por lo tanto habrá que tener en cuenta la altura de dicha articulación en los cálculos. Para ello se tomará como altura de la muñeca *(pm_z-d_1).*

Mediante el teorema del coseno se obtiene:

$$r^2 + (pm_z - d_1)^2 = {a_2}^2 + {l_{34}}^2 - 2a_2 l_{34}\cos\gamma$$

Despejando y sustituyendo *r* se obtiene:

$$\gamma = arcos\ \frac{{pm_x}^2 + {pm_y}^2 + (pm_z - d_1)^2 - {a_2}^2 - {l_{34}}^2}{-2a_2 l_{34}}$$

El cálculo de θ_3 se hace a partir de la diferencia de ángulos:

$$\theta_3 = 180 - \gamma - \beta$$

Para el cálculo de θ_2 se hace uso de razones trigonométricas para hallar los ángulos adyacentes (Fig 6.6).

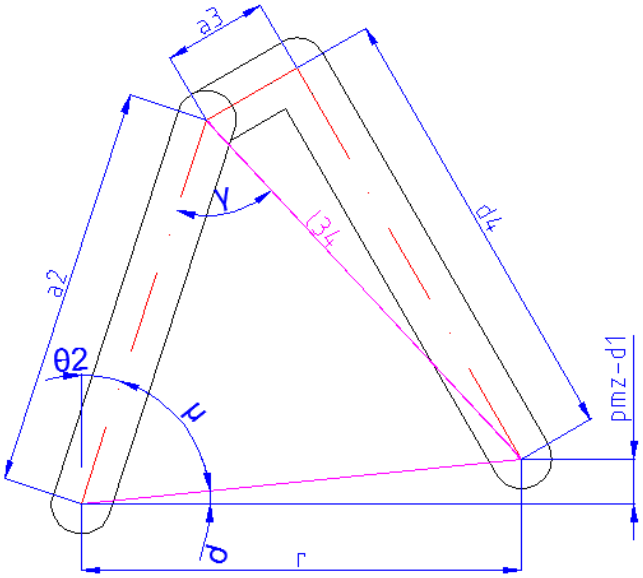


Figura 6.6. Esquema gráfico para la determinación de θ_2

El cálculo de ρ es sencillo siendo:

$$\rho = arctg \frac{pm_z - d_1}{\sqrt{{pm_x}^2 + {pm_y}^2}}$$

Para hallar el valor de μ se hace uso del teorema del seno:

$$\frac{l_{34}}{sen\,\mu} = \frac{\sqrt{{pm_x}^2 + {pm_y}^2 + (pm_z - d_1)^2}}{sen\,\gamma}$$

$$\mu = arcsen\left(\frac{l_{34} \cdot Sen\,\gamma}{\sqrt{{pm_x}^2 + {pm_y}^2 + (pm_z - d_1)^2}}\right)$$

Igual que antes, el valor de θ_2 se consigue haciendo la diferencia de ángulos:

$$\theta_2 = 90 - \mu - \rho$$

Una vez cálculados los tres primeros grados de libertad, se procede a calcular los tres últimos por medio de las matrices de rotación [1] implícitas en las matrices de transformación homogénea del robot. Como ya se conoce la posición de la muñeca se puede separar la matriz de rotación del robot en dos partes y, conociéndose la matriz del movimiento completo, sacar la matriz de rotación que falta, la cual contiene la información correspondiente a los tres últimos grados de libertad.

Para ello se debe considerar que las orientaciones de los sistemas **{S₆}** y **{S₃}** vienen relacionadas por:

$$\boldsymbol{{}^0R_6} = \boldsymbol{{}^0R_3} \cdot \boldsymbol{{}^3R_6} = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}$$

$$\boldsymbol{{}^3R_6} = \left(\boldsymbol{{}^0R_3}\right)^{-1} \cdot \boldsymbol{{}^0R_6} = \left(\boldsymbol{{}^0R_3}\right)^{T} \cdot \boldsymbol{{}^0R_6} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Siendo la matriz $\boldsymbol{{}^0R_6}$ conocida por ser la submatriz de rotación de la matriz **T**.

La matriz $^0\mathbf{R_3}$ puede obtenerse a partir de las matrices $^0\mathbf{R_1}$,$^1\mathbf{R_2}$,$^2\mathbf{R_3}$, submatrices de rotación de $^0\mathbf{A_1}$,$^1\mathbf{A_2}$,$^2\mathbf{A_3}$, tomando los valores:

$$^0\boldsymbol{R_1} = \begin{bmatrix} C\theta_1 & 0 & -S\theta_1 \\ S\theta_1 & 0 & C\theta_1 \\ 0 & -1 & 0 \end{bmatrix};\ ^1\boldsymbol{R_2} = \begin{bmatrix} S(\theta_2) & C(\theta_2) & 0 \\ -C(\theta_2) & S(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix};\ ^2\boldsymbol{R_3} = \begin{bmatrix} C\theta_3 & 0 & -S\theta_3 \\ S\theta_3 & 0 & C\theta_3 \\ 0 & -1 & 0 \end{bmatrix}$$

Resultando:

$$^0\boldsymbol{R_3} = {}^0\boldsymbol{R_1}\ {}^1\boldsymbol{R_2}\ {}^2\boldsymbol{R_3} = \begin{bmatrix} C_1S_{23} & S_1 & C_1C_{23} \\ S_1S_{23} & -C_1 & S_1C_{23} \\ C_{23} & 0 & -S_{23} \end{bmatrix}$$

Con lo que:

$$^3\boldsymbol{R_0} = \left(\,^0\boldsymbol{R_3}\right)^{-1} = \left(\,^0\boldsymbol{R_3}\right)^{T} = \begin{bmatrix} C_1S_{23} & S_1S_{23} & C_{23} \\ S_1 & -C_1 & 0 \\ C_1C_{23} & S_1C_{23} & -S_{23} \end{bmatrix}$$

Por último, la matriz $^3\mathbf{R_6}$ será función de $\theta_4, \theta_5, \theta_6$ y se obtiene a partir de los parámetros de Denavit-Hartenberg por:

$$^3\boldsymbol{R_6} = {}^3\boldsymbol{R_4}\ {}^4\boldsymbol{R_5}\ {}^5\boldsymbol{R_6}$$

Siendo:

$$^3\boldsymbol{R_4} = \begin{bmatrix} C\theta_4 & 0 & S\theta_4 \\ S\theta_4 & 0 & -C\theta_4 \\ 0 & 1 & 0 \end{bmatrix};\ ^4\boldsymbol{R_5} = \begin{bmatrix} C\theta_5 & 0 & -S\theta_5 \\ S\theta_5 & 0 & C\theta_5 \\ 0 & -1 & 0 \end{bmatrix};\ ^5\boldsymbol{R_6} = \begin{bmatrix} -C(\theta_6) & S(\theta_6) & 0 \\ -S(\theta_6) & -C(\theta_6) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$^3\boldsymbol{R_6} = {}^3\boldsymbol{R_4}\ {}^4\boldsymbol{R_5}\ {}^5\boldsymbol{R_6} = \begin{bmatrix} -C_4C_5C_6 + S_4S_6 & C_4C_5S_6 + S_4C_6 & -C_4S_5 \\ -S_4C_5C_6 - C_4S_6 & S_4C_5S_6 + C_4C_6 & -S_4S_5 \\ -S_5C_6 & S_5S_6 & C_5 \end{bmatrix}$$

Obtenidas por tanto las expresiones de $^0\mathbf{R_3}^T$ y $^0\mathbf{R_6}$, como datos conocidos y la de $^3\mathbf{R_6}$ en función de $\theta_4, \theta_5, \theta_6$, se tendrá que la expresión $^3\boldsymbol{R_6} = \left(\,^0\boldsymbol{R_3}\right)^T \cdot {}^0\boldsymbol{R_6}$ proporciona 9 ecuaciones con 3 incógnitas.

$$\begin{bmatrix} -C_4C_5C_6 + S_4S_6 & C_4C_5S_6 + S_4C_6 & -C_4S_5 \\ -S_4C_5C_6 - C_4S_6 & S_4C_5S_6 + C_4C_6 & -S_4S_5 \\ -S_5C_6 & S_5S_6 & C_5 \end{bmatrix} = \begin{bmatrix} C_1S_{23} & S_1S_{23} & C_{23} \\ S_1 & -C_1 & 0 \\ C_1C_{23} & S_1C_{23} & -S_{23} \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}$$

Utilizando las ecuaciones correspondientes a la última columna se obtiene:

$$-C_4S_5 = C_1S_{23}a_x + S_1S_{23}a_y + C_{23}a_z$$
$$-S_4S_5 = S_1a_x - C_1a_y$$
$$C_5 = C_1C_{23}a_x + S_1C_{23}a_y - S_{23}a_z$$

De la tercera es posible obtener θ_5, que usando la expresión del arcocoseno toma como valor:

$$\theta_5 = arcos(\ C_1C_{23}a_x + S_1C_{23}a_y - S_{23}a_z)$$

Posteriormente se divide la segunda ecuación entre la primera con la que se consigue:

$$\theta_4 = arctg\left(\frac{S_1a_x - C_1a_y}{C_1S_{23}a_x + S_1S_{23}a_y + C_{23}a_z}\right)$$

Se toman por ultimo las ecuaciones correspondientes a la última fila:

$$-S_5C_6 = C_1C_{23}n_x + S_1C_{23}n_y - S_{23}n_z$$
$$S_5S_6 = C_1C_{23}o_x + S_1C_{23}o_y - S_{23}o_z$$

Dividendo la segunda entre la primera se obtiene:

$$\theta_6 = arctg\left(-\frac{C_1C_{23}o_x + S_1C_{23}o_y - S_{23}o_z}{C_1C_{23}n_x + S_1C_{23}n_y - S_{23}n_z}\right)$$

Es importante considerar que estos valores obtenidos para θ_4 y θ_6 sólo son validos si $\theta_5 \neq 0$. En caso contrario, se encuentra que en la matriz $^3\mathbf{R_6}$ los valores de θ_4 y θ_6 aparecen siempre sumados y por lo tanto solo será posible obtener el valor de la suma de los ángulos. Es decir, habrá infinitas soluciones correspondientes a que $\theta_4 + \theta_6$ tomen un valor concreto. Esta situación $(\theta_5 = 0)$ corresponde a un punto singular y en él desaparece un grado de libertad pues θ_4 y θ_6 consiguen el mismo efecto.

6.3 Creación de Interfaz Gráfica de la cinemática del robot en MATLAB

Se ha creado, como parte de este trabajo, una Interfaz Gráfica de Usuario (GUI) para la resolución del problema cinemático directo e inverso del robot IRB 120 en la que se encuentran incorporadas todas las relaciones matemáticas desarrolladas en el apartado anterior.

Para ello se ha diseñado una ventana de introducción de datos en la que el usuario podrá obtener los datos de la cinemática del robot de una forma sencilla y automática.

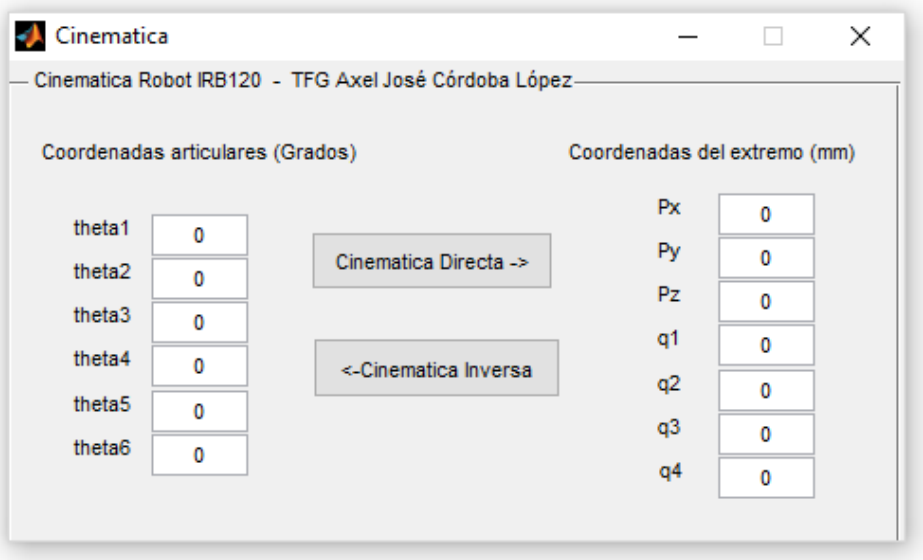


Figura 6.7. Interfaz Gráfica en Matlab de la cinemática del robot IRB 120.

La interfaz funciona de una forma sencilla, simplemente deben rellenarse los datos de una de las columnas y al pulsar el botón del problema cinemático correspondiente, la interfaz rellenará la otra columna con la resolución del problema.

6.4 Modelo diferencial. Matriz Jacobiana

El modelo diferencial de la cinemática del robot pretende encontrar una relación entre las velocidades de las coordenadas articulares y las de posición y orientación del extremo.

6.4.1 Jacobiana directa

Una vez calculadas las relaciones entre las posiciones, se puede proceder a calcular la relación entre las velocidades gracias a la matriz Jacobiana. Para calcularla usaremos el método numérico explicado en el capítulo de fundamentos teóricos. Estas ecuaciones resueltas en el software Matlab nos determinan la matriz de forma sencilla y rápida.

Con el siguiente código se obtiene la matriz de la Jacobiana con los parámetros, para que se vea la formulación final.

```
clear all
close all
  %Calculo de matriz simbólica
%Eslabón 1
syms theta1
d=290;
alpha=-90;
a=0;
theta=theta1;
A01=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha)
a*cos(theta) ;  sin(theta) cosd(alpha)*cos(theta) -
cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d ; 0 0
0 1] ;

%Eslabón 2
syms theta2
d=0;
alpha=0;
a=270;
theta=theta2-pi;
A12=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha)
a*cos(theta) ;  sin(theta) cosd(alpha)*cos(theta) -
cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d ; 0 0
0 1] ;


%Eslabón 3
syms theta3
d=0;
alpha=-90;
a=70;
theta=theta3;
A23=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha)
a*cos(theta) ;  sin(theta) cosd(alpha)*cos(theta) -
cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d ; 0 0
0 1] ;

%Eslabón 4
syms theta4
d=302;
alpha=90;
a=0;
theta=theta4;
A34=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha)
a*cos(theta) ;  sin(theta) cosd(alpha)*cos(theta) -
cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d ; 0 0
0 1] ;

%Eslabón 5
syms theta5
d=0;
alpha= -90;
a=0;
theta=theta5;
```

```
A45=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha)
a*cos(theta) ;  sin(theta) cosd(alpha)*cos(theta) -
cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d ; 0 0
0 1] ;

%Eslabón 6
syms theta6
d=72;
alpha=0;
a=0;
theta=theta6+pi;
A56=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha)
a*cos(theta) ;  sin(theta) cosd(alpha)*cos(theta) -
cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d ; 0 0
0 1] ;

%Matriz de Conversión
A02=simplify(A01*A12);
A03=simplify(A01*A12*A23);
A04=simplify(A01*A12*A23*A34);
A05=simplify(A01*A12*A23*A34*A45);
A06=simplify(A01*A12*A23*A34*A45*A56);

%Vector z
Z00=[0;0;0];
Z01=A01(1:3,3);
Z02=A02(1:3,3);
Z03=A03(1:3,3);
Z04=A04(1:3,3);
Z05=A05(1:3,3);
Z06=A06(1:3,3);

%Vector p
P06=A06(1:3,4);
P16=A06(1:3,4)-A01(1:3,4);
P26=A06(1:3,4)-A02(1:3,4);
P36=A06(1:3,4)-A03(1:3,4);
P46=A06(1:3,4)-A04(1:3,4);
P56=A06(1:3,4)-A05(1:3,4);

% J
J1=simple([cross(Z00,P06);Z00]);
J2=simple([cross(Z01,P16);Z01]);
J3=simple([cross(Z02,P26);Z02]);
J4=simple([cross(Z03,P36);Z03]);
J5=simple([cross(Z04,P46);Z04]);
J6=simple([cross(Z05,P56);Z05]);

J=simplify([J1,J2,J3,J4,J5,J6])
```

Como resultado de este código se obtiene la matriz genérica (Ec 4.6) para cualquier orientación del robot.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \dot{\alpha}\\ \dot{\beta}\\ \dot{\gamma}\end{bmatrix} = J_g \cdot \begin{bmatrix}\dot{\theta}_1\\ \vdots\\ \dot{\theta}_n\end{bmatrix} con\, J_g = \begin{bmatrix}\frac{\partial f_x}{\partial \theta_1} & \cdots & \frac{\partial f_x}{\partial \theta_n}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_\gamma}{\partial \theta_1} & \cdots & \frac{\partial f_\gamma}{\partial \theta_n}\end{bmatrix} \quad \text{(Ec 4.6)}$$

A continuación se desarrollan los términos de la matriz:

$$\frac{\partial f_x}{\partial \theta_1} = 0$$

$$\frac{\partial f_x}{\partial \theta_2} = -C_1 \cdot (302 \cdot S_{23} - 70 \cdot C_{23} - 270 \cdot C_2 + 72 \cdot S_{23}C_5 + 72 \cdot C_{23}C_4S_5)$$

$$\frac{\partial f_x}{\partial \theta_3} = -C_1 \cdot (302 \cdot S_{23} - 70 \cdot C_{23} + 72 \cdot S_{23}C_5 + 72 \cdot C_{23}C_4S_5)$$

$$\frac{\partial f_x}{\partial \theta_4} = 72 \cdot S_5 \cdot (C_1S_{23}S_4 - C_4S_1)$$

$$\frac{\partial f_x}{\partial \theta_5} = -72 \cdot C_1C_{23}S_5 - 72 \cdot C_5S_1S_4 - 72 \cdot C_1C_4C_5S_{23}$$

$$\frac{\partial f_x}{\partial \theta_6} = 0$$

$$\frac{\partial f_y}{\partial \theta_1} = 0$$

$$\frac{\partial f_y}{\partial \theta_2} = -S_1 \cdot (302 \cdot S_{23} - 70 \cdot C_{23} - 270 \cdot C_2 + 72 \cdot S_{23}C_5 + 72 \cdot C_{23}C_4S_5)$$

$$\frac{\partial f_x}{\partial \theta_3} = -S_1 \cdot (302 \cdot S_{23} - 70 \cdot C_{23} + 72 \cdot S_{23}C_5 + 72 \cdot C_{23}C_4S_5)$$

$$\frac{\partial f_x}{\partial \theta_4} = 72 \cdot S_5 \cdot (S_1S_{23}S_4 + C_4C_1)$$

$$\frac{\partial f_x}{\partial \theta_5} = -72 \cdot S_1C_{23}S_5 + 72 \cdot C_5C_1S_4 - 72 \cdot S_1C_4C_5S_{23}$$

$$\frac{\partial f_y}{\partial \theta_6} = 0$$

$$\frac{\partial f_z}{\partial \theta_1} = 0$$

$$\frac{\partial f_z}{\partial \theta_2} = -302 \cdot C_{23} - 70 \cdot S_{23} - 270 \cdot S_2 + 72 \cdot C_{23}C_5 + 72 \cdot S_{23}C_4S_5$$

$$\frac{\partial f_z}{\partial \theta_3} = -302 \cdot C_{23} - 70 \cdot S_{23} + 72 \cdot C_{23}C_5 + 72 \cdot S_{23}C_4S_5$$

$$\frac{\partial f_z}{\partial \theta_4} = 72 \cdot C_{23} S_4 S_5$$

$$\frac{\partial f_z}{\partial \theta_5} = 72 \cdot S_{23} S_5 - 72 \cdot C_{23} C_4 C_5$$

$$\frac{\partial f_z}{\partial \theta_6} = 0$$

$$\frac{\partial f_\alpha}{\partial \theta_1} = 0$$

$$\frac{\partial f_\alpha}{\partial \theta_2} = -S_1$$

$$\frac{\partial f_\alpha}{\partial \theta_3} = -S_1$$

$$\frac{\partial f_\alpha}{\partial \theta_4} = C_1 C_{23}$$

$$\frac{\partial f_\alpha}{\partial \theta_5} = C_1 S_{23} S_4 - C_4 S_1$$

$$\frac{\partial f_\alpha}{\partial \theta_6} = C_1 C_{23} C_5 - S_5 (S_1 S_4 + C_1 C_4 S_{23})$$

$$\frac{\partial f_\beta}{\partial \theta_1} = 0$$

$$\frac{\partial f_\beta}{\partial \theta_2} = C_1$$

$$\frac{\partial f_\beta}{\partial \theta_3} = C_1$$

$$\frac{\partial f_\beta}{\partial \theta_4} = S_1 C_{23}$$

$$\frac{\partial f_\beta}{\partial \theta_5} = S_1 S_{23} S_4 - C_4 C_1$$

$$\frac{\partial f_\beta}{\partial \theta_6} = S_1 C_{23} C_5 - S_5 (C_1 S_4 + S_1 C_4 S_{23})$$

$$\frac{\partial f_\gamma}{\partial \theta_1} = 0$$

$$\frac{\partial f_\gamma}{\partial \theta_2} = 0$$

$$\frac{\partial f_\gamma}{\partial \theta_3} = 0$$

$$\frac{\partial f_\gamma}{\partial \theta_4} = -S_{23}$$

$$\frac{\partial f_\gamma}{\partial \theta_5} = C_{23}S_4$$

$$\frac{\partial f_\gamma}{\partial \theta_6} = S_{23}C_5 - C_{23}C_4S_5$$

Para el cálculo de la Matriz Jacobiana exacta, es decir, sustituyendo los valores de las articulaciones, se usara el siguiente código:

```
%Calculo de coordenadas de TCP
clear all
close all
%Primero definimos los valores de los ángulos theta(grados)
theta1=5.28;
theta2=78.51;
theta3=-67.29;
theta4=1.01;
theta5=96.86;
theta6=9.25;

%Eslabón 1
d=290;
alpha=-90;
a=0;
theta=theta1;
A01=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
a*cosd(theta) ;  sind(theta) cosd(alpha)*cosd(theta) -
cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha) cosd(alpha) d ; 0
0 0 1]

%Eslabón 2
d=0;
alpha=0;
a=270;
theta=theta2-90;
A12=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
a*cosd(theta) ;  sind(theta) cosd(alpha)*cosd(theta) -
cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha) cosd(alpha) d ; 0
0 0 1]


%Eslabón 3
d=0;
alpha=-90;
a=70;
theta=theta3;
A23=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
a*cosd(theta) ;  sind(theta) cosd(alpha)*cosd(theta) -
cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha) cosd(alpha) d ; 0
0 0 1]

%Eslabón 4
d=302;
```

```
alpha=90;
a=0;
theta=theta4;
A34=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
a*cosd(theta) ;  sind(theta) cosd(alpha)*cosd(theta) -
cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha) cosd(alpha) d ; 0
0 0 1]

%Eslabón 5
d=0;
alpha= -90;
a=0;
theta=theta5;
A45=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
a*cosd(theta) ;  sind(theta) cosd(alpha)*cosd(theta) -
cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha) cosd(alpha) d ; 0
0 0 1]

%Eslabón 6
d=72;
alpha=(0);
a=0;
theta=theta6+180;
A56=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
a*cosd(theta) ;  sind(theta) cosd(alpha)*cosd(theta) -
cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha) cosd(alpha) d ; 0
0 0 1]

%Matriz de Conversión
T=A01*A12*A23*A34*A45*A56

%Matriz de Conversión
A02=A01*A12;
A03=A01*A12*A23;
A04=(A01*A12*A23*A34);
A05=(A01*A12*A23*A34*A45);
A06=(A01*A12*A23*A34*A45*A56);

%Vector z
Z00=[0;0;0];
Z01=A01(1:3,3);
Z02=A02(1:3,3);
Z03=A03(1:3,3);
Z04=A04(1:3,3);
Z05=A05(1:3,3);
Z06=A06(1:3,3);

%Vector p
P06=A06(1:3,4);
P16=A06(1:3,4)-A01(1:3,4);
P26=A06(1:3,4)-A02(1:3,4);
P36=A06(1:3,4)-A03(1:3,4);
P46=A06(1:3,4)-A04(1:3,4);
P56=A06(1:3,4)-A05(1:3,4);

% J
J1=([cross(Z00,P06);Z00]);
```

```
J2=([cross(Z01,P16);Z01]);
J3=([cross(Z02,P26);Z02]);
J4=([cross(Z03,P36);Z03]);
J5=([cross(Z04,P46);Z04]);
J6=([cross(Z05,P56);Z05]);

J=([J1,J2,J3,J4,J5,J6])
```

6.5 Dinámica del robot

Como se ha visto en el capítulo de Fundamentos teóricos, para un estudio avanzado de la dinámica de un robot se necesita incorporar a la formulación datos dinámicos del robot IRB 120: masas, centros de gravedad, momentos de inercia, etc. Para obtener estos datos de forma fiable, hay que recurrir al fabricante ABB y, este, no los suministra por motivos de confidencialidad del producto. Conseguirlos de forma teórica o experimental sobre el robot real resulta imposible para el alcance de este trabajo.

No obstante, aunque no se ha podido llevar a cabo el estudio dinámico del robot en este trabajo, debido a su dificultad, se propone una solución al problema de obtención de datos. La propuesta consiste en obtener los datos mediante el programa SolidEdge, en función del modelo 3d que el fabricante ABB suministra en su página web [12].

7 PRÁCTICAS “MECÁNICA DE ROBOTS”

Uno de los principales objetivos de este proyecto es el de realizar el diseño de diferentes prácticas que puedan ser interesantes para el alumnado de ingeniería de la Escuela Politécnica Superior de Linares. Se han realizado una serie de prácticas en el laboratorio para la asignatura de Mecánica de Robot del grado de Ingeniería Mecánica, con la intención de acercar la teoría de la asignatura a la visión práctica de la misma.

Estas prácticas están configuradas como clases magistrales en las que el alumno aprende los conceptos del manejo del robot mediante código RAPID de forma progresiva en dificultad con problemas prácticos, en los cuales el alumno debe de poder utilizar los conocimientos que se expliquen para la realización de la práctica y poder usarlos para avanzar en otros aspectos de programación del robot.

A continuación se explica pormenorizadamente cada práctica.

7.1 Practica 1. Dibujar en plano

La primera práctica realizada en el laboratorio está diseñada para una introducción al código de programación RAPID, aprendiendo en esta las funciones más básicas de movimiento con programación textual.

Estas funciones serán realizadas en un solo plano de trabajo para no complicar el aprendizaje de las mismas.

La práctica consiste en programar al robot para que dibuje en un papel las iniciales del alumno, aprendiendo con ello el uso de las funciones MOVEL, MOVEC, MOVEJ, la definición de los puntos en el entorno de trabajo y las relaciones de dichos puntos con los sistemas de coordenadas definidos en la unidad de programación.

Como se ha comentado en el capítulo de materiales y métodos, el robot del laboratorio dispone de unas pinzas eléctricas de desplazamiento lineal y agarre plano. El desarrollo de esta práctica consiste en que el robot manipule un rotulador cilíndrico para realizar los dibujos sobre papel. Para ello se ha diseñado un adaptador para las pinzas que se adaptara a la forma circular del rotulador (Figura 7.2). Además, para facilitar el desplazamiento del rotulador por el folio sin dañarse, se ha utilizado un adaptador prismático para el rotulador (Figura 7.1), haciendo que este varíe su posición vertical al ser sometido a presión.

La manera de proceder para el diseño comenzó por introducir el rotulador, de marca BIC para pizarras con una medida de 130 mm, en el interior del adaptador prismático, haciendo que este se introdujera por completo para poder obtener una medida más fiable de invarianza al cambiar de rotuladores en un futuro.

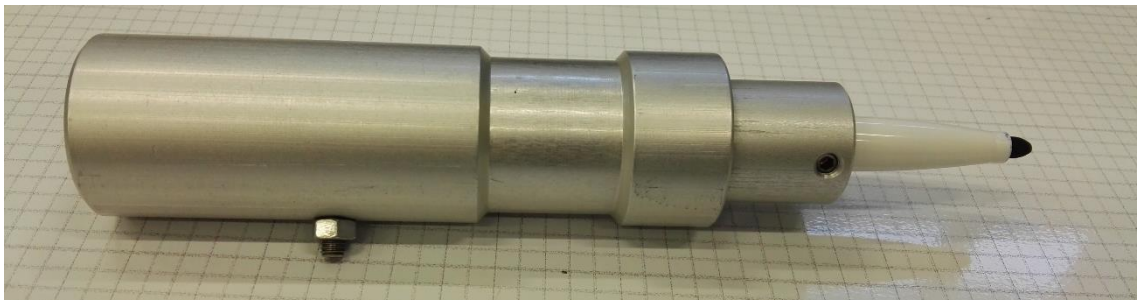


Figura 7.1. Adaptador prismático para rotulador.

Una vez introducido el rotulador se tomaron las medidas necesarias para diseñar un adaptador entre el rotulador y la pinza. Este diseño se ha realizado con el software de modelado en 3D SolidEdge y se ha imprimido con una impresora 3D.

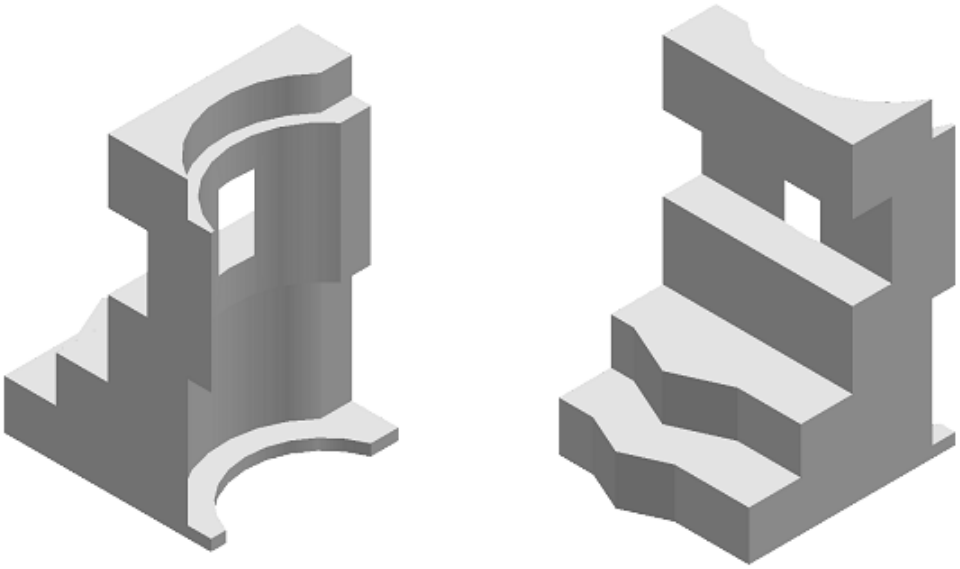


Figura 7.2. Adaptador mordazas-rotulador

A continuación se procede a colocar el rotulador con su adaptador en la pinza del robot como se muestra en figura 7.3. Para ello se abre la pinza al máximo, se introduce todo el conjunto de sujeción y se cierra la pinza con la función *cerrar_pinza_linea8* la cual cierra la pinza con una medida de 15.50 mm. De esta manera el rotulador queda fijado a la pinza del robot y no se moverá al realizar las operaciones de la práctica.

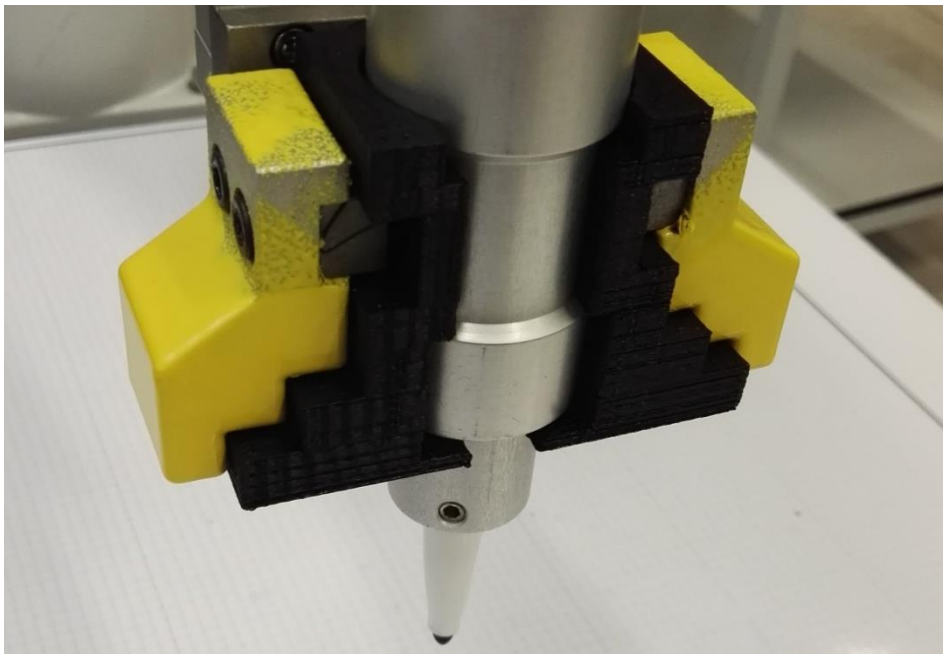


Figura 7.3. Rotulador montado en la pinza

El desarrollo de la práctica consiste, como ya se ha explicado anteriormente, en programar movimientos del robot que generen un dibujo sobre un plano. Los dibujos en este caso se realizarán sobre un folio, el cual se fija al plano de trabajo poniendole peso encima para evitar movimientos y colocándose según indica la figura 7.4.

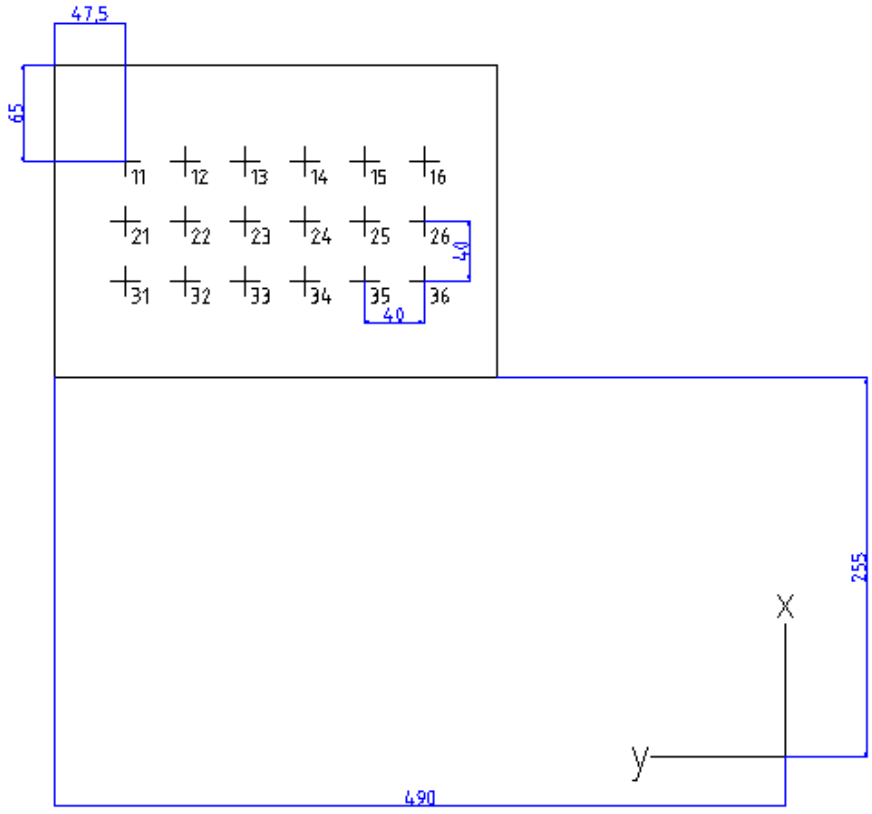


Figura 7.4. Colocación de folio en el plano de trabajo. Malla de puntos en folio

Para facilitar la práctica se realiza la definición de una cuadricula de puntos sobre el folio, como se muestra en la figura 7.4, los cuales puedan ser utilizados por los alumnos para programar los movimientos del robot. Estos han sido programados y ubicados por el autor de este trabajo como se indica a continuación.

Para ello, en primer lugar se define un punto de la cuadricula, en este caso el punto 31 para así colocar el bolígrafo perpendicular al plano de dibujo y con la altura necesaria para no dañar el papel ni el rotulador y poder realizar de una manera correcta el dibujo.

A partir de este punto se crean los demás puntos de la cuadricula con la función OFFSET de RAPID, con la cual se pueden definir puntos nuevos a partir de desplazamientos de un punto base. Este concepto se le explica en la práctica al alumno para que pueda definir puntos nuevos si lo necesita en su dibujo, sin la necesidad de mover físicamente el brazo robótico al punto deseado. Esta manera de definir los puntos de la cuadricula a partir de uno inicial sirve de gran utilidad por diferentes motivos, uno de ellos es que se define con precisión la distancia entre los puntos y se asegura que en todos ellos se mantenga una altura constante del rotulador. Por otra parte, si las condiciones de la base del dibujo varían, se modifican todos los parámetros del programa simplemente cambiando el punto inicial 31.

Por último, antes de que el alumno se disponga a programar, se le expone un ejemplo de lo que debe realizar. Utilizando como base el código de dibujo de las iniciales del autor de este trabajo, se les explica línea por línea lo que el robot realiza en cada momento y cómo funcionan y se declaran las funciones MOVEL, MOVEJ y MOVEC de RAPID. En dicho código se detallan línea por línea los movimientos programados para que después de la explicación del profesor el alumno mantenga este código y pueda usarlo de base para realizar su propio dibujo.

Una vez realizados los programas por el alumno, los cuales han sido escritos mediante programación textual en un archivo .txt, son copiados en el módulo de operaciones del robot del laboratorio para su comprobación.

Para ello se introducirán en Programa_7 (Anexo 2.7), realizado por el autor de este trabajo, en el cual se encuentran definidos los puntos de la matriz y un programa que pregunta al usuario qué dibujo realizar.

Al iniciarse el Programa_7 se ejecuta la función TPReadNum, la cual pregunta al usuario qué dibujo desea realizar. Al elegir un número este queda guardado como

respuesta a la pregunta. Dependiendo del número, el programa ejecutará el código de dibujo correspondiente.

Para que esto funcione cada código del alumno debe ser introducido en un bucle IF que se realice si el número elegido por el controlador del robot es el del alumno.

Una vez elegido el número de dibujo que se quiere realizar, el robot se sitúa encima del papel, luego a un punto próximo al comienzo de dibujo y por ultimo procede a ejecutar el código del alumno. Una vez terminado vuelve a su posición original y pregunta de nuevo qué dibujo se desea realizar.

7.2 Práctica 2. Reorientar pieza

La segunda práctica realizada en el laboratorio está diseñada para que el alumno amplíe sus conocimientos y práctica con las funciones de movimiento del robot, sin embargo, en esta práctica los movimientos no se realizarán en un mismo plano, con lo que se pretende que se comprenda con mejor detalle la función MOVEJ, la cual permite realizar cambios de orientación de la herramienta. También se presentará una especial atención al cambio de velocidades en los movimientos del robot y se explicará el uso de la pinza eléctrica.

Esta segunda práctica consiste en el giro y reubicación de una pieza de aluminio mediante el uso del robot con las funciones de movimiento y el uso de la pinza.

Para ello el alumno recibe como material de apoyo e instructivo el código del movimiento de la pieza de una posición y orientación *A* a una posición *C* pasando por un punto intermedio *B* como se muestra en la figura 7.5.

Este código, generado por el autor de este trabajo en el Programa_1 (Anexo 2.1), detalla línea por línea lo que realiza el robot en cada momento. Mientras se ejecuta el programa, el alumno puede ver qué línea de código está ejecutando el robot para así entenderlo de una forma más intuitiva.

Una vez hecha la demostración y explicado el código, la práctica que deberá realizar el alumno, con el código del Programa_1 como base, consistirá en la realización inversa de los movimientos anteriores para dejar la pieza como estaba en un origen, es decir, el código generado por el alumno deberá permitir que el robot coja la pieza en la posición C y la oriente y posicione en la posición A pasando antes por el punto intermedio B.

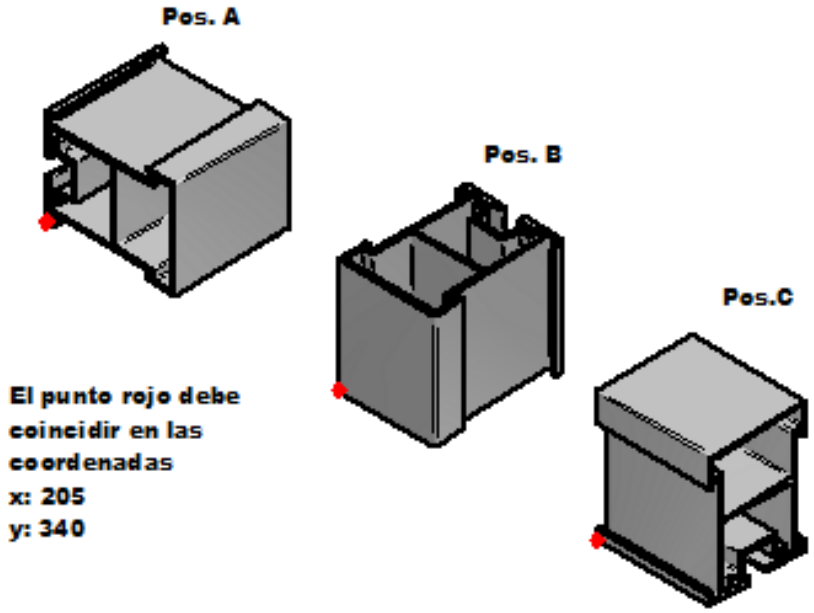


Figura 7.5. Coordenadas de posicionamiento de pieza en práctica 2

En esta práctica el alumno debe estar muy atento de modificar las velocidades de aproximación del robot a la pieza y, con los conocimientos aprendidos de las funciones, debe ser capaz de modificar el código entre MOVEL Y MOVEJ en los momentos que se necesite un cambio de orientación.

Los puntos que usa el robot para realizar los movimientos están declarados fijos en el programa y no se necesitará crear ningún punto adicional, solo se necesitará reordenar y modificar las funciones de movimiento hacia esos puntos.

El profesor dispone del código de la práctica realizado por el autor de este trabajo en el Programa_2 (Anexo 2.2) para poder orientar así al alumno y comprobar que esté bien programado antes de ejecutarlo.

7.3 Práctica 3. Dispensador. Aprendizaje de funciones de programación

La práctica 3 está diseñada como una práctica de aprendizaje en la que el alumno no deberá realizar ningún código. Esta servirá para explicar el uso de funciones de programación de RAPID como por ejemplo IF, WHILE, TPReadNum y TPReadFK.

Estas funciones serán utilizadas para la realización de la práctica 4 y por eso es necesario que el alumno preste especial atención a los conceptos clave de la programación y su funcionamiento.

La práctica consiste en una explicación de dichas funciones con la ayuda del Programa_4 (Anexo 2.4), el cual ejecuta el Programa_3 (Anexo 2.3) en base a las funciones que el alumno debe aprender.

El Programa_3 genera un movimiento de agarre de una pieza cilíndrica de un dispensador de piezas usando la función *cerrar_pinza_linea7* la cual cierra la pinza con una distancia de 7,8 mm, y la deposita en lo alto de dicho dispensador.

Para poder realizar dicha práctica, el autor de este trabajo ha debido diseñar el dispensador de cilindros, cuyo diseño ha sido realizado mediante el software SolidEdge e imprimido con una impresora 3D.

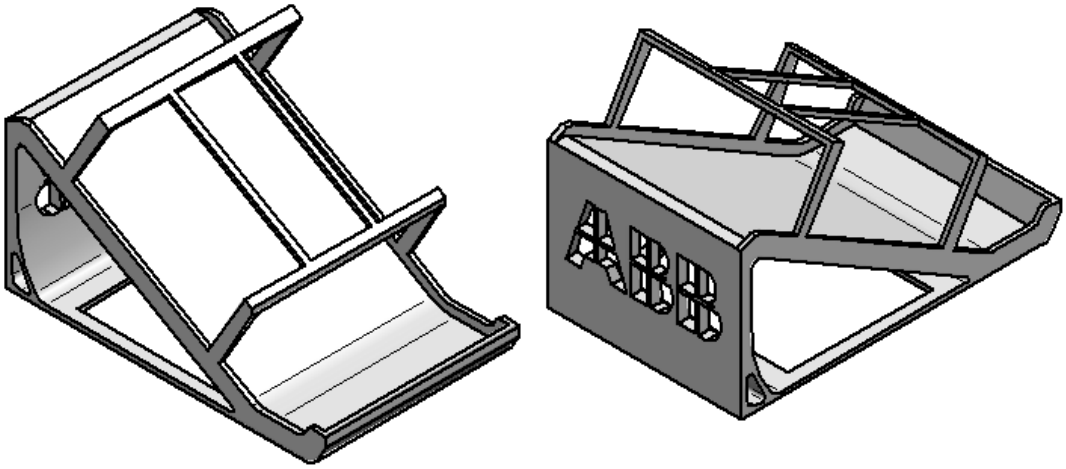


Figura 7.6. Dispensador de cilindros para práctica 3. Dibujado con SolidEdge

Se utilizaron cuatro cilindros de cartón cortados a medida a una longitud de 75 mm para que pudieran ser agarrados por la pinza.

Para la realización del Programa_4 se debía diferenciar uno de los cilindros de los demás, por lo tanto se les dio una imprimación para diferenciarlos y para que el cartón no se deteriorara con la humedad.

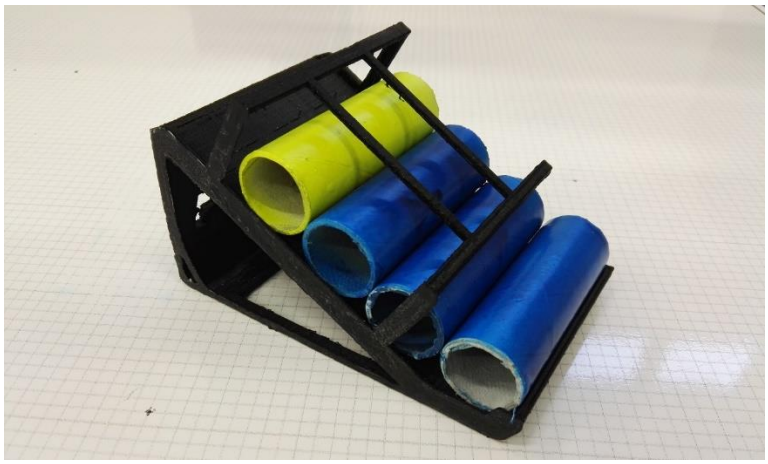


Figura 7.7. Dispensador con cilindros diferenciados

Para el agarre del cilindro explicado en el Programa_3 el dispensador debe ser ubicado en un sitio concreto del plano y con la orientación correcta. Para ello se colocará la esquina inferior izquierda del dispensador en las coordenadas x:320mm, y:340mm colocándose ortogonalmente y coincidiendo con la cuadrícula.

El propósito del Programa_4 es hacer que el cilindro amarillo acabe siempre en la primera posición del dispensador. Para ello lo primero que el programa realiza es una pregunta al usuario para que informe de en qué posición se encuentra. Una vez que el robot sabe en qué posición se encuentra el cilindro amarillo ejecuta el Programa_3 tantas veces como sea necesario hasta que llegue al amarillo.

Con este programa por tanto se pueden explicar las funciones TPReadNum y TPReadFK , las cuales sirven para pedir datos al operario como números o pulsaciones de tecla de una forma práctica, ya que estas son utilizadas para preguntar en qué posición se encuentra el cilindro amarillo.

Una vez elegido, el Programa_4 ejecutará el Programa_3, hasta que este quede en la posición correcta, usando la función WHILE.

La función IF entra en este código como condición en el caso de que el cilindro ya se encuentre en la posición primera. Cuando esto ocurre debe realizarse igualmente el código moviendo todos los cilindros hasta que se vuelva a encontrar en la misma posición.

Aparte de las funciones ya nombradas también se explicarán las funciones INCR y DECR en el uso de contadores dentro del WHILE.

7.4 Práctica 4. Apilado de piezas

Usando todos los conceptos y funciones aprendidos anteriormente se puede realizar la práctica 4, que consiste en un movimiento de pieza condicionado a preguntas a realizar al usuario, usando puntos variables, puntos fijos y funciones de repetición.

En la práctica 4 se dispondrá de 4 piezas similares a las de la práctica 2, ubicadas a 100 mm de separación entre ellas en línea recta.

Con el uso del Programa_5 (Anexo 2.5) el robot apilará en una columna central las 4 piezas anteriormente nombradas en el orden que especifique el operario que lo esté manejando. La misión del alumno será realizar el movimiento contrario, es decir, estando las 4 piezas apiladas en una columna central, el robot debe cogerlas una a una y ubicarlas en la posición que se especifique.

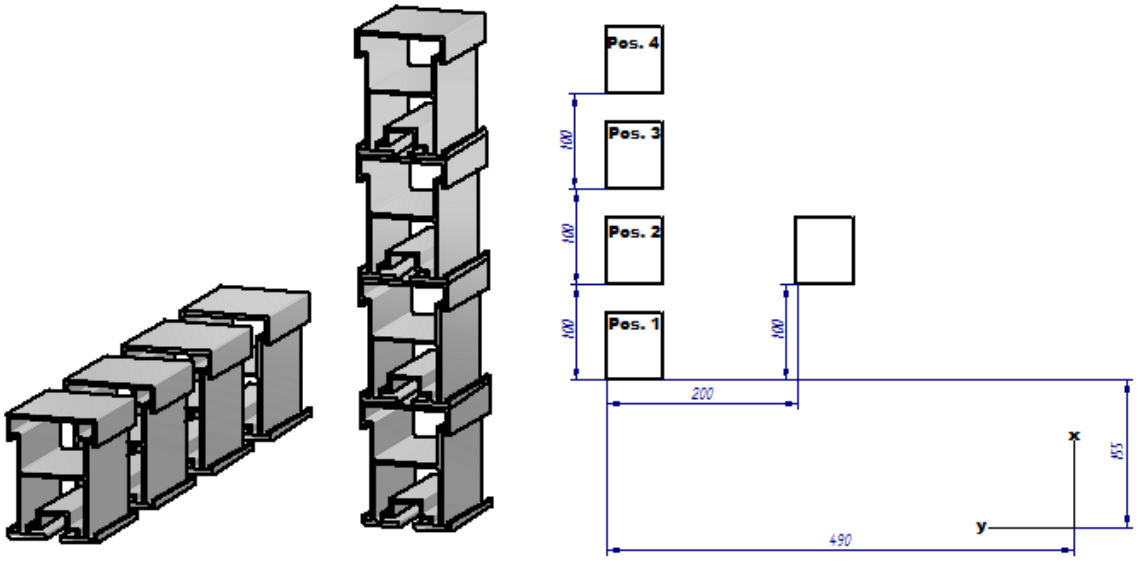


Figura 7.8. Plano de posiciones para práctica 4

Para que el alumno pueda realizar dicho código debe entender cómo se ha realizado la programación del Programa_5 ya que este será la base para realizar la práctica 4. El código del Programa_5, comentado línea a línea, será entregado al alumno para que vaya siguiendo el código mientras recibe la explicación del mismo.

Para la programación del Programa_5 lo primero que se realiza es la declaración del punto de agarre de la primera pieza. Esta declaración se hace mediante el joystick para obtener más precisión en el agarre como se explicó en la práctica 2.

Una vez delimitado este punto los demás serán definidos a partir de este mediante la función Offset como se vio en la práctica 1.

Estos puntos deben ser definidos como variables dependientes al orden de agarre que se vaya a realizar. Para saber el orden de agarre, debe ser preguntado al operario como se explica en la práctica 3.Una vez definidos todos los puntos, el robot generará el movimiento de agarrar y soltar la pieza en la posición deseada con las funciones de movimiento aprendidas en la práctica 2.

Por último, este proceso debe ser repetido hasta colocar las 4 piezas en la columna central. Esto se consigue con las funciones de repetibilidad aprendidas en la práctica 3.

De esta forma se da un repaso a todas las funciones y conceptos aprendidos en las sesiones de prácticas para poder realizar en esta el programa inverso al que se ha explicado.

Este programa inverso está realizado por el autor de este trabajo en el Programa_6 como guía para el profesor para ayudar y corregir al alumno.

7.5 Encuesta sobre prácticas

Debido a ser el primer año de realización de estas prácticas y a la organización de las mismas dentro del horario de impartición de la asignatura, hubo falta de tiempo disponible para prácticas y no se pudo realizar la práctica 3. No obstante, los fundamentos teóricos de dicha práctica se explicaron en la sesión de la práctica 4.

Una vez realizadas las prácticas se pasó una encuesta al alumnado para que valorara su experiencia.

La encuesta es la siguiente:

- Preguntas generales:
    - Los contenidos vistos en prácticas coinciden con los teóricos de la asignatura.
    - Las prácticas de la asignatura sirven para afianzar los contenidos teóricos.
    - La dificultad de las prácticas es adecuada.
    - La disponibilidad de un brazo robot en las prácticas de la asignatura resulta interesante.
    - La programación del brazo robot en prácticas resulta interesante.
    - El aprendizaje del lenguaje de programación RAPID resulta interesante.
- Preguntas para cada práctica:
    - Disponía de antemano de los conocimientos necesarios para su realización.
    - Realizar la práctica me ha servido para afianzar los conceptos tratados en ella.
    - Considero que estos conceptos son útiles de cara a mi futuro laboral.
    - Considero que la dificultad de la práctica ha sido adecuada.
    - Considero que la práctica es interesante.
- Sugerencias del alumnado. Por ultimo, en la encuesta existía un apartado opcional para rellenar por el alumno, donde poner su opinión sobre las prácticas y comentar posibles modificaciones.

8 RESULTADOS

En este capítulo se muestran los resultados obtenidos con la realización de este trabajo fin de grado. Se exponen la comprobación de los datos cinemáticos con los reales y los resultados de satisfacción de las prácticas realizadas por el alumnado.

8.1 Resultados cinemáticos

Una vez resuelta la cinemática del robot de forma teórica, se comprueban los datos obtenidos con el robot del laboratorio para corroborar que las funciones obtenidas son correctas.

Para ello se coloca el robot en una serie de posiciones, tomando nota de las coordenadas articulares y las coordenadas de posición y orientación de la herramienta, datos que se son mostrados por el robot a través del FlexPendant.

Posteriormente se introducen estos datos en la Interfaz gráfica, diseñada por el autor de este trabajo, dedicada a la cinemática directa e inversa del robot IRB 120 del laboratorio. Se introducirán los datos realizando las operaciones de cinemática directa e inversa y se comprobarán con los datos recogidos anteriormente.

8.1.1 Posición 1. Posición de calibración

En este caso se procede ubicando el robot en la posición de calibración, en la que todas las coordenadas articulares son 0 tal y como se muestra en la figura 6.3.

Una vez está el robot ubicado en la posición, se procede a tomar los datos de posición y orientación.

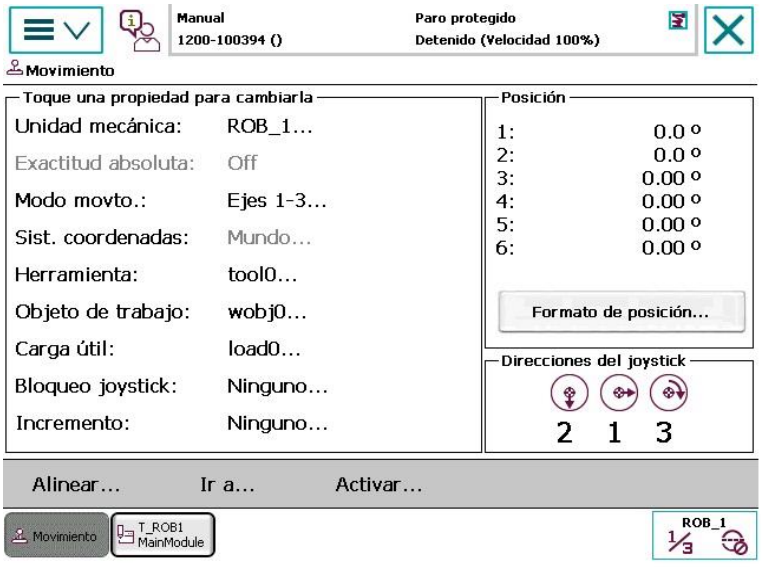


Figura 8.1. Coordenadas articulares de la posición 1

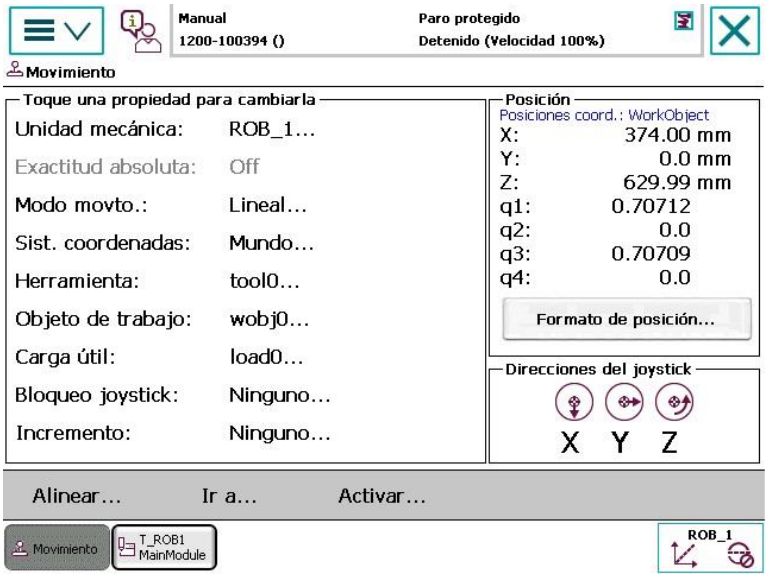


Figura 8.2. Coordenadas de posición y orientación de extremo en posición 1

Se procede a continuación a introducir los datos de las coordenadas articulares (Fig 8.1) en la interfaz gráfica y pulsar el botón de cinemática inversa para obtener los datos de posición y orientación teóricos de la herramienta.

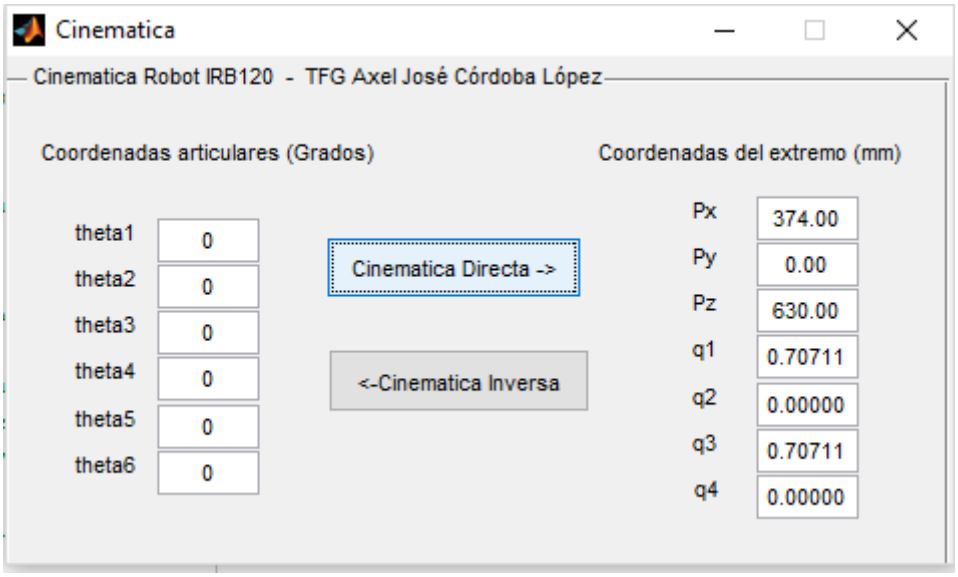


Figura 8.3. Cinemática Directa posición 1. Interfaz gráfica de MATLAB

Como puede comprobarse las coordenadas del extremo teóricas que pueden verse en la figura 8.3 coinciden con los datos ofrecidos por el robot real del laboratorio en la figura 8.2.

Por último, se comprueba que la cinemática inversa también es correcta introduciendo los datos de posición y orientación de la herramienta (Fig 8.2) en la interfaz gráfica y pulsando el botón de cinemática inversa, lo que da como resultados los encontrados en la figura 8.4.

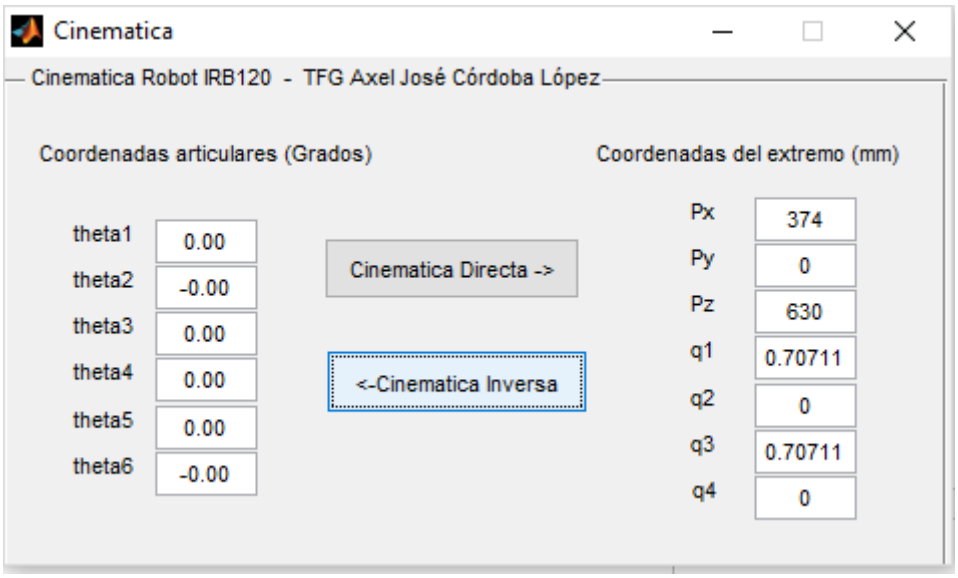


Figura 8.4. Cinemática inversa posición 1. Interfaz gráfica de MATLAB.

De igual manera, comparando los datos de coordenadas articulares obtenidos teóricamente en la figura 8.4 con los datos ofrecidos por el robot en la figura 8.1, se puede decir que los resultados son idénticos y que la formulación teórica de la cinemática del robot es correcta y coincide con los datos reales.

8.1.2 Posición 2.

Se hace una segunda comprobación con una posición distinta. Esta vez la posición del robot será una posición aleatoria, un punto cualquiera dentro del área de trabajo.

Al igual que se ha realizado para la posición 1, se toman los datos ofrecidos por el FlexPendant de la posición 2 del robot.

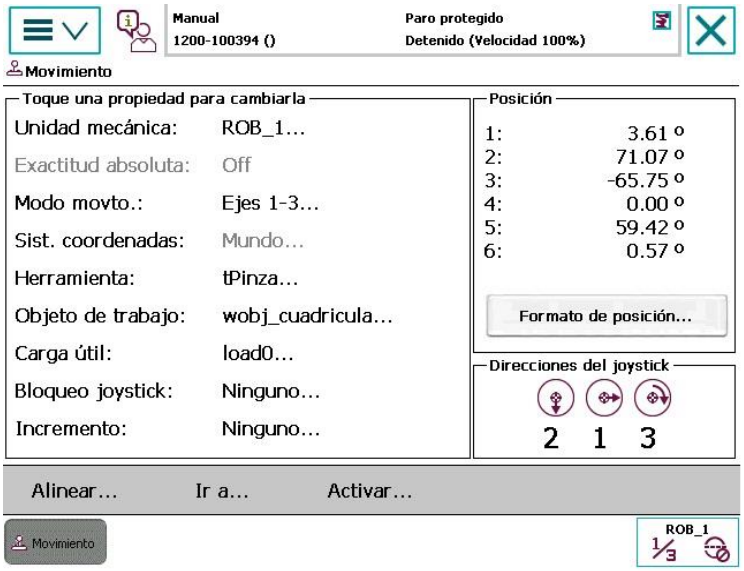


Figura 8.5. Coordenadas articulares de la posición 2

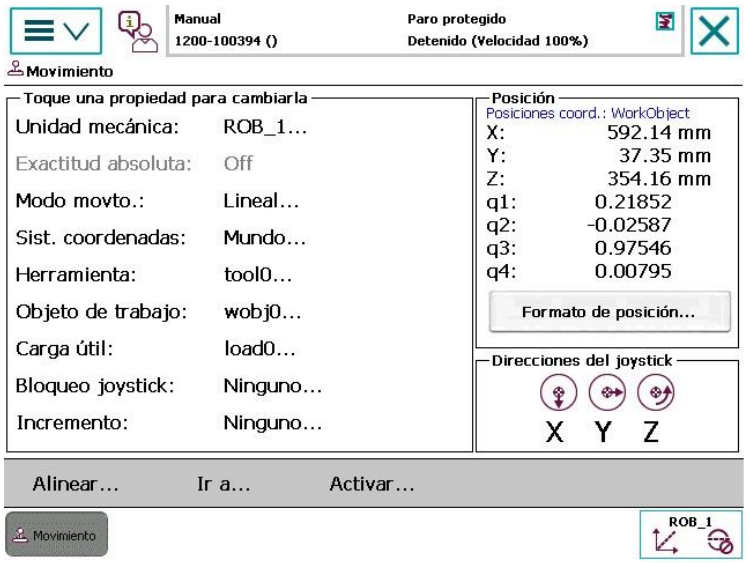


Figura 8.6. Coordenadas de posición y orientación de extremo en posición 2

Se procede ahora a introducir los datos de las coordenadas articulares en la interfaz gráfica para obtener los valores de posición de la herramienta con el botón de cinemática directa.

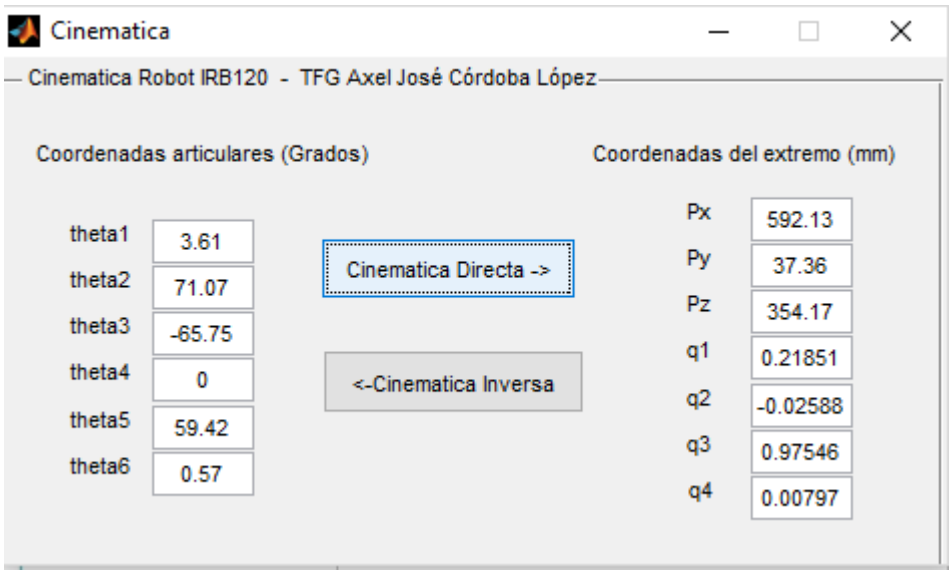


Figura 8.7. Cinemática Directa posición 2. Interfaz gráfica de MATLAB

Se puede comprobar de nuevo en la figura 8.7 que los datos son idénticos a los datos ofrecidos por el FlexPendant en la figura 8.6.

Por último, se vuelven a introducir los datos de coordenadas del extremo y se pulsa el botón de cinemática inversa para obtener las coordenadas articulares.

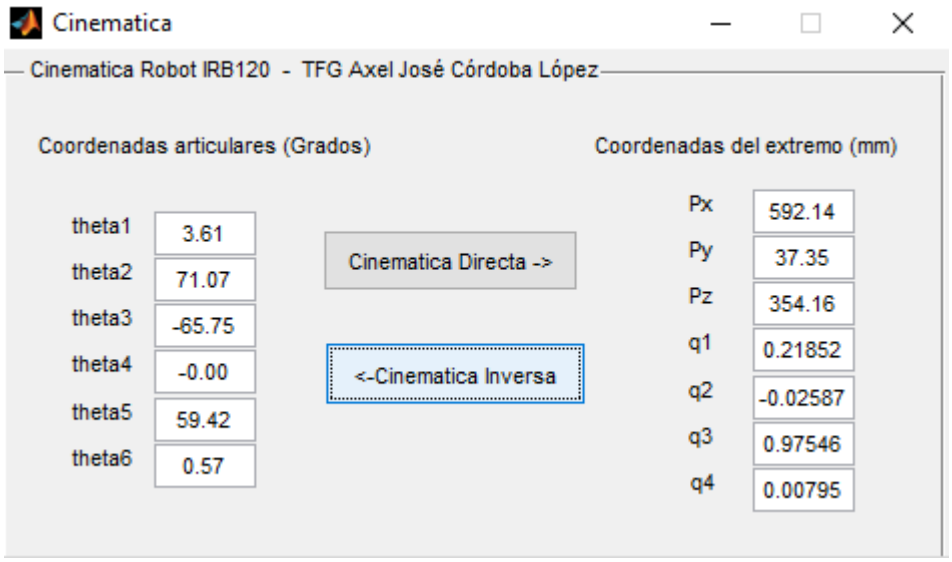


Figura 8.8. Cinemática Inversa posición 2. Interfaz gráfica de MATLAB.

Del mismo modo, se comparan los datos obtenidos en la columna de coordenadas articulares en la figura 8.8 con los datos ofrecidos por el robot en la figura 8.5. Al ser estos idénticos, se corrobora que las funciones teóricas de la cinemática del robot son correctas y coinciden con el robot real.

8.2 Resultados de encuesta de satisfacción de prácticas.

En este apartado se muestran los resultados de la encuesta de satisfacción de las prácticas. Se presenta la media obtenida de las respuestas de los 22 alumnos de la asignatura “Mecanica de Robots” del curso académico 2015/16.

Las preguntas debían ser valoradas de 1 a 5 según el grado de concordancia siendo 1 “nada de acuerdo” y 5 “muy de acuerdo”.

- Preguntas generales

Los contenidos vistos en prácticas coinciden con los teóricos de la asignatura.	4,14.
Las prácticas de la asignatura sirven para afianzar los contenidos teóricos.	3,91
La dificultad de las prácticas es adecuada.	4,05
La disponibilidad de un brazo robot en las prácticas de la asignatura resulta interesante.	4,95
La programación del brazo robot en prácticas resulta interesante.	4,64
El aprendizaje del lenguaje de programación RAPID resulta interesante.	4,36

- Preguntas para cada práctica

	Práctica 1	Práctica 2	Práctica 4
Disponía de antemano de los conocimientos necesarios para su realización.	3,23	3,32	3,50
Realizar la práctica me ha servido para afianzar los conceptos tratados en ella.	4,23	4,50	4,45
Considero que estos conceptos son útiles de cara a mi futuro laboral.	4,14	4,14	4,18
Considero que la dificultad de la práctica era adecuada.	3,95	3,86	3,91
Considero que la práctica es interesante.	4,50	4,68	4,50

- Sugerencias del alumnado.

  A continuación se exponen todos los comentarios voluntarios realizados por los alumnos en el apartado de sugerencias:

  - Las prácticas han sido muy interesantes, he aprendido mucho y el único comentario que tengo que añadir es mi enhorabuena al compañero Axel José por el trabajo realizado en estas prácticas.
  - Las prácticas han sido muy interesantes, poder realizarlas con el robot ha sido una gran experiencia. ¡Por fin algo que no sea mera teoría! El

compañero Axel ha realizado una gran labor y los programas que nos facilitaba han sido divertidos de ver y de intentar programar.

- Dar rápidamente la parte teórica, profundizando solo en la parte necesaria para la realización de las prácticas que hemos realizado. Una vez hecho esto enseñar la programación del robot y pasar más tiempo con éste, programando cada uno pequeños programas, con el fin de aprovechar mucho más el brazo robot ya que se tiene.
- Añadiría más tiempo al horario dedicado a las prácticas, aprendiendo así más funciones de RAPID y usando más el robot.
- Se debería de haber dado más despacio el tema de la programación del robot y/o utilizado más ejemplos.
- Más ejercicios prácticos.
- Solo añadiría más ejemplos resueltos de la programación del robot junto a su funcionamiento, para aclarar más las ideas a la hora de realizar nosotros el ejercicio.
- Las prácticas en general han estado bien, aunque me hubiese gustado estudiar en mayor profundidad la programación del brazo robot.
- De las prácticas, no cambiaría nada, de la asignatura en general, creo que se debería enfocar más como una asignatura práctica, es decir, dar por encima la teoría y centrarse en el manejo del robot industrial, así como en entender la programación del mismo, que es lo más importante de cara a nuestro futuro laboral.
- Se debería dedicar alguna clase anterior a la práctica para explicarla y que los alumnos tengan tiempo para prepararla. O al menos dedicar tiempo de teoría para explicar bien el lenguaje de programacion y hacer ejemplos parecidos.
- Uso del simulador virtual del brazo robótico.
- Creo que si se dedicara una clase de teória previa a cada práctica explicando cómo programar en Rapid, o los diferentes comandos, las prácticas se facilitarían bastante. La última fue demasiado compleja para el nivel que teníamos.
- Las prácticas son muy buenas, interesantes, me ha encantado la asignatura.

9 DISCUSIÓN Y CONCLUSIONES

Este trabajo ha supuesto un punto de inflexión en el estudio de la robótica industrial en la Escuela Politécnica Superior de Linares gracias a la puesta en marcha de la primera célula robótica en el campus, un brazo robótico dedicado a la docencia con el cual el alumnado puede interaccionar y ver de forma práctica los conocimientos aprendidos en asignaturas de robótica.

En el apartado referente al estudio teórico de la cinemática del robot se explican las grandes dificultades encontradas para hallar un método de resolución de la cinemática inversa para este robot en concreto, que debido a la distribución y morfología de sus articulaciones no puede ser resuelta mediante los métodos matriciales tradicionales. Haciendo una extensa revisión bibliográfica no se ha conseguido un método eficaz para la resolución del problema, pero sí ha proporcionado los fundamentos teóricos cinemáticos necesarios para su resolución mediante el método de desacoplo cinemático del problema desarrollando la primera parte de este con métodos geométricos, lo que hace que el cálculo de las variables sea más complicado, pero efectivo.

Como resultados de este TFG cabe destacar:

- Estudio cinemático de un robot real disponible en el laboratorio, comprobándose que los datos concuerdan con la realidad, ayudando así a su explicación de forma didáctica para el alumno.
- Creación de una interfaz gráfica, fácil de usar, que muestra los datos referentes a la cinemática del robot de laboratorio.
- Puesta en marcha de célula robótica, diseñándose y creándose herramientas y planos donde trabajar y ser funcional.
- Diseño de las primeras prácticas de la asignatura “Mecánica de robot” impartida en la Universidad. En la Escuela Politécnica Superior de Linares es la primera vez que se dispone de un robot y unas prácticas para esta asignatura.
- La satisfacción de los estudiantes respecto a las prácticas impartidas es palpable en los datos obtenidos en la encuesta.
- Se han creado también unas instrucciones de uso para la célula robótica que permitirán a futuros usuarios poder manejarla sin complicación. Además, explican la resolución de algunos problemas que le han surgido al autor de este trabajo y que han ido sido subsanadas a base de prueba y error y mucha investigación de forma autónoma.

10 TRABAJOS FUTUROS

Gracias a este trabajo, el cual, como se ha comentado anteriormente, ha sido el primero en el ámbito de la robótica realizado por la EPSL, se crea un punto de partida para trabajos de esta índole y se abre una rama de investigación y desarrollo referente a esta. De esta forma existen innumerables vertientes donde seguir trabajando en esta temática y avanzar en este trabajo, que debido a su alcance no ha permitido poder desarrollar por completo todas las posibilidades que ofrece.

En este capítulo se proponen distintas ramas de continuación para trabajos futuros.

10.1 Ampliación de prácticas

Según los datos obtenidos en la encuesta de prácticas, los estudiantes han quedado muy satisfechos con las prácticas realizadas durante este curso y añaden que les hubiera gustado dedicar más tiempo de la asignatura a manejar el robot y realizar más prácticas.

Por lo tanto, un trabajo futuro podría ser el diseño de más prácticas donde se puedan ampliar los conocimientos aprendidos y el alumno pueda estar más tiempo manejando el robot y aprendiendo a usar más detalladamente el FlexPendant.

También podría diseñarse alguna práctica que simulara las operaciones de pintura y soldadura, las cuales son muy utilizadas en la industria.

10.2 Elementos terminales

En este trabajo se ha diseñado un adaptador para la pinza que le permitiera agarrar un rotulador para las prácticas.

Un futuro trabajo podría ser diseñar nuevos adaptadores o herramientas terminales que permitieran realizar más funciones al robot de la célula.

Por ultimo podría introducirse una herramienta de sujeción por medio de ventosas. Gracias a las instalaciones de aire comprimido instaladas en el laboratorio, no sería difícil adaptar este sistema al robot.

10.3 Dinámica de robot

El cálculo de la dinámica del robot es un tema complicado. Debido al alcance de este trabajo no se ha podido desarrollar un estudio dinámico completo, pero se ha descrito el método a llevar a cabo y una propuesta de cómo obtener los datos necesarios para su realización.

Gracias al uso de acelerómetros, se puede comprobar si los datos obtenidos en este análisis corresponden con la realidad.

10.4 RobotStudio

Por último, en un futuro debería ahondarse más en el software RobotStudio, el cual permite crear la misma célula de trabajo virtualmente, lo que ofrecería la posibilidad de que todos los alumnos pudieran manejar un robot de forma virtual en su ordenador.

11 BIBLIOGRAFÍA

[1] BARRIENTOS, A. *Fundamentos de robótica*. Madrid: McGraw-Hill, Interamericana de España, 2007.

[2] FERRATÉ, G. et al. *Robótica industrial*. Barcelona: Marcombo, 1986.

[3] RENTERÍA, A. y RIVAS, M. *Robótica industrial*. Madrid: McGraw-Hill, Interamericana de España, 2000.

[4] TORRALVA, C. Nueva clasificación de la IFR para los robots. *Automatización industrial y revista de robótica*, Abril 1992, 67.

[5] GROOVER, M. *Robótica Industrial*. Madrid: McGraw-Hill, Interamericana de España, 1989.

[6] DENAVIT, J. y HARTENBERG, R.S. A Kinematic Notation far Lower-Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics*, junio 1995.

[7] CRAIG, J.J. *Robótica, Tercera Edición*. México: Pearson Prentice Hall, 2006.

[8] ABB Group. *Curso Básico Robot IRC5*, 2015. (Solicitado a compañía ABB Group. Disponible en Laboratorio L-037, EPSL).

[9] ABB Group. *Especificaciones de producto IRB 120.* ID: 3HAC035960-005. Suiza: Revisión L, 2014. (Solicitado a compañía ABB Group. Disponible en Laboratorio L-037, EPSL).

[10] ABB Group. *Manual del producto IRB 120.* ID: 3HAC035728-005. Suiza: Revisión K, 2015. (Solicitado a compañía ABB Group .Disponible en Laboratorio L-037, EPSL).

[11] ABB Group. *Datos técnicos controlador IRC5*. Suecia: 2016. Disponible en: https://library.e.abb.com/public/bedd1769ea1e4bb9c1257da10037e215/IRC5_IndustrialRobotController_ROB0295EN.pdf

[12] ABB en España, 2016. www.*New.abb.com/es* [Consultado 13/05/2016].

[13] *Catálogo pinzas eléctricas CAT.EUS100-77E-ES*. España: SMC Corporation (Europe), 2016. [Consultado 13/05/2016]. Disponible en: https://content2.smcetech.com/pdf/LEH-E_ES.pdf

[14] SMC`s Corporate. *Manual de instalación LEC-OM00701*. (Disponible en Laboratorio L-037, EPSL).

[15] SMC Corporation. *Operation manual: Electric gripper LEH Series.* LEHZ-OM00216. Tokyo: 2016. [Consultado 13/05/2016]. Disponible en: https://www.smc.eu/smc/Net/EMC_DDBB/op_manual/data/attachments/OM_LEH_OM00216EN.pdf

[16] SMC Corporation, 2016. www.*smces.es* [Consultado 13/05/2016].

[17] DOUGHERTY, F. C., MACARI, J. y OKAMOTO, C. *Pulleys*. Society of Women Engineers, 2007.

[18] Asociación Española de Normalización y Certifcación. *Robots. Manipuladores industriales. Vocabulario.* UNE EN ISO 8373:1998. Madrid: AENOR, 1998.

[19] KUKA Robots industriales, 2016. www.*Kuka-robotics.com* [Consultado 13/05/2016].

[20] International Federation of Robotics and United Nations. *Statistics, Market Analysis, Forecast, Case Studies and rofitability of Robot Investment, United Nations Publications*. World Robotics, 2004.

[21] ABB Group. *Celula didáctica IRB-120*. 2015. (Disponible en Laboratorio L-037, EPSL).

[22] KEE, E. *Aquabotix HydroView beams back high definition video*. Übergizmo, 2012. [Consultado 13/05/2016]. Disponible en: http://www.ubergizmo.com/2012/05/aquabotix-hydroview-beams-back-high-definition-video/

[23] *Parrot AR.Drone 2.0 Quadricopter Review*, 2014. *Dronepug*. [Consultado 13/05/2016]. Disponible en: http://bfpug.com/parrot-ar-drone-2-0-quadricopter-review/

[24*] iRobot Roomba® 651 - iRobotEMEA*. Tienda.irobot.es. 2016. [Consultado 13/05/2016]. Disponible en: http://tienda.irobot.es/es/irobot-roomba-651/R651040.html?cgid=es&lang=es_ES

12 ANEXOS

En este capítulo se detallan las funciones, programas y planos utilizados para el desarrollo de este trabajo.

12.1 Funciones y código en Matlab

A continuación se muestran los códigos utilizados en Matlab para la resolución de la cinematica del robot.

12.1.1 Cinemática directa simbólica

```
clear all
close all
  %Cálculo de matriz simbólica
%Eslabón 1
syms theta1
d=290;
alpha=-90;
a=0;
theta=theta1;
A01=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1]

%Eslabón 2
syms theta2
d=0;
alpha=0;
a=270;
theta=theta2-90;
A12=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1]


%Eslabón 3
syms theta3
d=0;
alpha=-90;
a=70;
theta=theta3;
A23=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1]

%Eslabón 4
syms theta4
d=302;
alpha=90;
a=0;
theta=theta4;
A34=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1]

%Eslabón 5
syms theta5
d=0;
alpha= -90;
a=0;
theta=theta5;
A45=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1]

%Eslabón 6
syms theta6
```

```
d=72;
alpha=(0);
a=0;
theta=theta6-180;
A56=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1]

%Matriz de Conversión

T=simplify(A01*A12*A23*A34*A45*A56)
```

12.1.2 Función matriz homogénea

Se realiza esta función para facilitar el cálculo de la matriz homogénea. Esta función será utilizada en funciones posteriores .Para su funcionamiento se debe introducir el número de eslabón y la matriz de parámetros de Denavit-Hartenberg del robot.

```
%Matriz homogenea
function [MH]=Matriz_homogenea(Eslabon,IRB120_DH)


theta=IRB120_DH(Eslabon,1);
d=IRB120_DH(Eslabon,2);
a=IRB120_DH(Eslabon,3);
alpha=IRB120_DH(Eslabon,4);

MH=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1] ;
end
```

12.1.3 Función para cinemática directa

Con esta función se obtienen los datos de posición y orientación del extremo del robot en función de las coordenadas articulares introducidas.

```
function
[px,py,pz,q1,q2,q3,q4]=Cinematica_directa(theta1,theta2,theta3,theta4,theta5,theta6)
%Constantes geométricas del Robot IRB 120
    %Eslabon1
    d1=290;
    a1=0;
    alpha1=-90;
    %Eslabon2
    d2=0;
    a2=270;
    alpha2=0;
    theta2=theta2-90;
    %Eslabon3
    d3=0;
    a3=70;
    alpha3=-90;
    %Eslabon4
    d4=302;
    a4=0;
    alpha4=90;
    %Eslabon5
    d5=0;
    a5=0;
    alpha5=-90;
```

```
    %Eslabon6
    d6=72;
    a6=0;
    alpha6=0;
    theta6=theta6-180;

    IRB120_DH=[theta1 d1 a1 alpha1;theta2 d2 a2 alpha2;theta3 d3 a3 alpha3;theta4 d4 a4
alpha4;theta5 d5 a5 alpha5;theta6 d6 a6 alpha6];
%Matrices geométricamente homogéneas

A01=Matriz_homogenea(1,IRB120_DH);
A12=Matriz_homogenea(2,IRB120_DH);
A23=Matriz_homogenea(3,IRB120_DH);
A34=Matriz_homogenea(4,IRB120_DH);
A45=Matriz_homogenea(5,IRB120_DH);
A56=Matriz_homogenea(6,IRB120_DH);

A06=A01*A12*A23*A34*A45*A56;

%Coordenadas del extremo
px=A06(1,4);
py=A06(2,4);
pz=A06(3,4);

%Cuaternios
q1=0.5*sqrt(A06(1,1)+A06(2,2)+A06(3,3)+1);
q2=sign(A06(3,2)-A06(2,3))*0.5*sqrt(A06(1,1)-A06(2,2)-A06(3,3)+1);
q3=sign(A06(1,3)-A06(3,1))*0.5*sqrt(-A06(1,1)+A06(2,2)-A06(3,3)+1);
q4=sign(A06(2,1)-A06(1,2))*0.5*sqrt(-A06(1,1)-A06(2,2)+A06(3,3)+1);

Coordenadas_extremo=[px,py,pz,q1,q2,q3,q4];
end
```

12.1.4 Función para cinemática inversa

Con esta función se calculan las coordenadas articulares según los datos de posición y orientación del extremo del robot.

```
%Cinemática inversa
function
[theta1,theta2,theta3,theta4,theta5,theta6]=Cinematica_inversa(px,py,pz,q1,q2,q3,q4)
%Constantes geométricas del Robot IRB 120
    %Eslabon1
    d1=290;
    a1=0;
    %Eslabon2
    d2=0;
    a2=270;
    %Eslabon3
    d3=0;
    a3=70;
    %Eslabon4
    d4=302;
    a4=0;
    %Eslabon5
    d5=0;
    a5=0;
    %Eslabon6
    d6=72;
    a6=0;

%Cálculo de matriz de transformación homogénea a partir de componentes de
%cuaternio

    nx= 2*(q1^2+q2^2-1/2);
    ox= 2*(q2*q3-q4*q1);
    ax= 2*(q2*q4+q3*q1);
    ny= 2*(q2*q3+q4*q1);
```

```
    oy= 2*(q1^2+q3^2-1/2);
    ay= 2*(q3*q4-q1*q2);
    nz= 2*(q2*q4-q1*q3);
    oz= 2*(q3*q4+q2*q1);
    az= 2*(q1^2+q4^2-1/2);


    T=[nx ox ax px;ny oy ay py; nz oz az pz;0 0 0 1];

%Coordenadas de la muñeca
    pmx= px -d6*ax;
    pmy= py -d6*ay;
    pmz= pz -d6*az;

%Cálculo de las tres primeras coordenadas articulares mediante métodos
%geométricos
    %theta1
        theta1= atand(pmy/pmx);
    %theta3
        l34=(a3^2+d4^2)^(1/2);
        Beta=atand(d4/a3);
        Gamma=acosd((pmx^2+pmy^2+(pmz-d1)^2-a2^2-l34^2)/(-2*a2*l34));
        theta3=180-Beta-Gamma;
    %theta2
        Mu=asind((sind(Gamma)*l34)/(pmx^2+pmy^2+(pmz-d1)^2)^0.5);
        Rho=atand((pmz-d1)/((pmx^2+pmy^2))^(1/2));
        theta2=90-Mu-Rho;

%Cálculo de las tres coordenadas articulares restantes mediante las
%matrices de rotación

%Matrices homogéneas

%Eslabón 1
d=290;
alpha=-90;
a=0;
theta=theta1;
A01=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1] ;
%Eslabón 2
d=0;
alpha=0;
a=270;
theta=theta2-90;
A12=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1] ;
%Eslabón 3
d=0;
alpha=-90;
a=70;
theta=theta3;
A23=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1] ;

%Matrices de rotación

R01=A01(1:3,1:3);
R12=A12(1:3,1:3);
R23=A23(1:3,1:3);

R03=R01*R12*R23;
R06=T(1:3,1:3);

R36=inv(R03)*R06;

theta4=atand(R36(2,3)/R36(1,3));
theta5=acosd(R36(3,3));
theta6=asind(R36(3,2)/-R36(3,1));
end
```

12.1.5 GUI (Interfaz gráfica de usuario) de Cinemática IRB120

Este es el código utilizado para el diseño de la interfaz gráfica creada para la cinemática del robot IRB120. Muestra los datos de la cinemática inversa o directa pulsando un botón.

```
function varargout = Cinematica(varargin)
% CINEMATICA MATLAB code for Cinematica.fig
%      CINEMATICA, by itself, creates a new CINEMATICA or raises the existing
%      singleton*.
%
%      H = CINEMATICA returns the handle to a new CINEMATICA or the handle to
%      the existing singleton*.
%
%      CINEMATICA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CINEMATICA.M with the given input arguments.
%
%      CINEMATICA('Property','Value',...) creates a new CINEMATICA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Cinematica_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Cinematica_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Cinematica

% Last Modified by GUIDE v2.5 22-Jun-2016 23:14:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Cinematica_OpeningFcn, ...
                   'gui_OutputFcn',  @Cinematica_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Cinematica is made visible.
```

```
function Cinematica_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Cinematica (see VARARGIN)

% Choose default command line output for Cinematica
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Cinematica wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Cinematica_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in BotonCD.
function BotonCD_Callback(hObject, eventdata, handles)
% hObject    handle to BotonCD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
theta1=str2double(get(handles.txttheta1,'string'));
theta2=str2double(get(handles.txttheta2,'string'));
theta3=str2double(get(handles.txttheta3,'string'));
theta4=str2double(get(handles.txttheta4,'string'));
theta5=str2double(get(handles.txttheta5,'string'));
theta6=str2double(get(handles.txttheta6,'string'));

[px,py,pz,q1,q2,q3,q4]=Cinematica_directa(theta1,theta2,theta3,theta4,theta5,theta6);
set(handles.txtpx,'string',num2str(sprintf('%.2f',px)));
set(handles.txtpy,'string',num2str(sprintf('%.2f',py)));
set(handles.txtpz,'string',num2str(sprintf('%.2f',pz)));
set(handles.txtq1,'string',num2str(sprintf('%.5f',q1)));
set(handles.txtq2,'string',num2str(sprintf('%.5f',q2)));
set(handles.txtq3,'string',num2str(sprintf('%.5f',q3)));
set(handles.txtq4,'string',num2str(sprintf('%.5f',q4)));

% --- Executes on button press in BotonCI.
function BotonCI_Callback(hObject, eventdata, handles)
% hObject    handle to BotonCI (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
px=str2double(get(handles.txtpx,'string'));
py=str2double(get(handles.txtpy,'string'));
pz=str2double(get(handles.txtpz,'string'));
q1=str2double(get(handles.txtq1,'string'));
q2=str2double(get(handles.txtq2,'string'));
q3=str2double(get(handles.txtq3,'string'));
q4=str2double(get(handles.txtq4,'string'));
[theta1,theta2,theta3,theta4,theta5,theta6]=Cinematica_inversa(px,py,pz,q1,q2,q3,q4);

set(handles.txttheta1,'string',num2str(sprintf('%.2f',theta1)));
set(handles.txttheta2,'string',num2str(sprintf('%.2f',theta2)));
set(handles.txttheta3,'string',num2str(sprintf('%.2f',theta3)));
set(handles.txttheta4,'string',num2str(sprintf('%.2f',theta4)));
set(handles.txttheta5,'string',num2str(sprintf('%.2f',theta5)));
set(handles.txttheta6,'string',num2str(sprintf('%.2f',theta6)));
```

12.1.6 Jacobiana geométrica simbólica

Con este código se obtiene la matriz Jacobiana en función de las coordenadas articulares.

```
clear all
close all
  %Calculo de matriz simbólica
%Eslabón 1
syms theta1
d=290;
alpha=-90;
a=0;
theta=theta1;
A01=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1] ;

%Eslabón 2
syms theta2
d=0;
alpha=0;
a=270;
theta=theta2-pi;
A12=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1] ;


%Eslabón 3
syms theta3
d=0;
alpha=-90;
a=70;
theta=theta3;
A23=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1] ;

%Eslabón 4
syms theta4
d=302;
alpha=90;
a=0;
theta=theta4;
A34=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
```

```
; 0 0 0 1] ;

%Eslabón 5
syms theta5
d=0;
alpha= -90;
a=0;
theta=theta5;
A45=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1] ;

%Eslabón 6
syms theta6
d=72;
alpha=0;
a=0;
theta=theta6+pi;
A56=[cos(theta) -sin(theta)*cosd(alpha) sin(theta)*sind(alpha) a*cos(theta) ;  sin(theta)
cosd(alpha)*cos(theta) -cos(theta)*sind(alpha) a*sin(theta) ; 0 sind(alpha) cosd(alpha) d
; 0 0 0 1] ;

%Matriz de Conversión

A02=simplify(A01*A12);
A03=simplify(A01*A12*A23);
A04=simplify(A01*A12*A23*A34);
A05=simplify(A01*A12*A23*A34*A45);
A06=simplify(A01*A12*A23*A34*A45*A56);

%Vector z
Z00=[0;0;0];
Z01=A01(1:3,3);
Z02=A02(1:3,3);
Z03=A03(1:3,3);
Z04=A04(1:3,3);
Z05=A05(1:3,3);
Z06=A06(1:3,3);

%Vector p
P06=A06(1:3,4);
P16=A06(1:3,4)-A01(1:3,4);
P26=A06(1:3,4)-A02(1:3,4);
P36=A06(1:3,4)-A03(1:3,4);
P46=A06(1:3,4)-A04(1:3,4);
P56=A06(1:3,4)-A05(1:3,4);

% J

J1=simple([cross(Z00,P06);Z00]);
J2=simple([cross(Z01,P16);Z01]);
J3=simple([cross(Z02,P26);Z02]);
J4=simple([cross(Z03,P36);Z03]);
J5=simple([cross(Z04,P46);Z04]);
J6=simple([cross(Z05,P56);Z05]);

J=simplify([J1,J2,J3,J4,J5,J6])
```

12.1.7 Jacobiana geométrica exacta

Con este código se obtiene la matriz Jacobiana exacta, introduciendo los datos de las coordenadas articulares.

```
%Calculo de coordenadas de TCP
clear all
close all
%Primero definimos los valores de los ángulos theta(grados)
```

```
theta1=5.28;
theta2=78.51;
theta3=-67.29;
theta4=1.01;
theta5=96.86;
theta6=9.25;

%Eslabón 1
d=290;
alpha=-90;
a=0;
theta=theta1;
A01=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 2
d=0;
alpha=0;
a=270;
theta=theta2-90;
A12=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]


%Eslabón 3
d=0;
alpha=-90;
a=70;
theta=theta3;
A23=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 4
d=302;
alpha=90;
a=0;
theta=theta4;
A34=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 5
d=0;
alpha= -90;
a=0;
theta=theta5;
A45=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Eslabón 6
d=72;
alpha=(0);
a=0;
theta=theta6+180;
A56=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta) ;
sind(theta) cosd(alpha)*cosd(theta) -cosd(theta)*sind(alpha) a*sind(theta) ; 0 sind(alpha)
cosd(alpha) d ; 0 0 0 1]

%Matriz de Conversión

T=A01*A12*A23*A34*A45*A56

%Matriz de Conversión

A02=A01*A12;
A03=A01*A12*A23;
A04=(A01*A12*A23*A34);
A05=(A01*A12*A23*A34*A45);
A06=(A01*A12*A23*A34*A45*A56);
```

```
%Vector z
Z00=[0;0;0];
Z01=A01(1:3,3);
Z02=A02(1:3,3);
Z03=A03(1:3,3);
Z04=A04(1:3,3);
Z05=A05(1:3,3);
Z06=A06(1:3,3);

%Vector p
P06=A06(1:3,4);
P16=A06(1:3,4)-A01(1:3,4);
P26=A06(1:3,4)-A02(1:3,4);
P36=A06(1:3,4)-A03(1:3,4);
P46=A06(1:3,4)-A04(1:3,4);
P56=A06(1:3,4)-A05(1:3,4);

% J

J1=([cross(Z00,P06);Z00]);
J2=([cross(Z01,P16);Z01]);
J3=([cross(Z02,P26);Z02]);
J4=([cross(Z03,P36);Z03]);
J5=([cross(Z04,P46);Z04]);
J6=([cross(Z05,P56);Z05]);

J=([J1,J2,J3,J4,J5,J6])
```

12.2 Programas

Programas realizados en RAPID para las prácticas de la asignatura “Mecánica de Robots”.

12.2.1 Programa 1. Enderezar pieza

```
PROC Programa_1()
   CONST robtarget Pr1_1:=[[217.68,295.06,385.78],[0.000611983,0.0249673,0.999683,0.00323775],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_2:=[[217.69,295.06,113.61],[0.000657061,0.0249639,0.999683,0.00323605],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_3:=[[216.47,298.98,68.80],[0.000659602,0.00550069,0.999979,0.00322002],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_4:=[[216.45,298.98,81.11],[0.000602029,0.00546519,0.99998,0.00321967],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_5:=[[216.45,310.55,557.51],[0.0005491,0.00542186,0.99998,0.00321355],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_6:=[[244.08,271.25,548.32],[0.496272,-0.504473,-0.500589,-0.49863],[-1,1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_7:=[[244.03,271.26,95.69],[0.496345,-0.504476,-0.500571,-0.498572],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_8:=[[244.02,271.27,56.01],[0.49637,-0.504486,-0.500547,-0.498561],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_9:=[[243.94,235.22,141.35],[0.496518,-0.504517,-0.50046,-0.49847],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_10:=[[204.89,310.38,402.14],[0.00344962,0.704474,-0.00341789,0.709713],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_11:=[[196.58,310.38,101.81],[0.00343199,0.704574,-0.00337068,0.709614],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_12:=[[206.81,310.38,51.98],[0.0034531,0.704583,-0.0033857,0.709605],[0,0,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_13:=[[206.78,310.37,96.96],[0.003474,0.704645,-0.00335265,0.709544],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_14:=[[206.77,310.37,473.00],[0.00347011,0.70469,-0.00330426,0.7095],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_15:=[[235.49,324.36,486.77],[0.0017233,-0.705643,-0.70856,-0.0027759],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_16:=[[235.49,324.37,106.88],[0.0017083,-0.705655,-0.708548,-0.00280686],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_17:=[[235.48,324.38,80.62],[0.00169583,-0.705678,-0.708525,-0.00282601],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_18:=[[235.47,324.38,127.31],[0.00170497,-0.705665,-0.708538,-0.00280591],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_19:=[[235.47,324.37,369.55],[0.00171459,-0.705654,-0.708549,-0.00278303],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


 MoveAbsJ calib_pos, v200, z50, tPinza;                !Movimiento a posición ejes 0 grados
Abrir_Pinza;
 MoveJ Pr1_1, v300, z50, tPinza\WObj:=wobj_cuadricula;    !Colocar posición horizontal
 MoveL Pr1_2, v200, z50, tPinza\WObj:=wobj_cuadricula;    !Acercamiento a la pieza rápido
 MoveL Pr1_3, v20, fine, tPinza\WObj:=wobj_cuadricula;    !Aproximación lenta a la pieza
Cerrar_Pinza_Linea5;
 MoveL Pr1_4, v20, fine, tPinza\WObj:=wobj_cuadricula;    !Levantamiento de la pieza delicado
 MoveL Pr1_5, v50, fine, tPinza\WObj:=wobj_cuadricula;    !Levantamiento de la pieza rápido
 MoveJ Pr1_6, v20, fine, tPinza\WObj:=wobj_cuadricula;    !Giro para colocar pieza en posición vertical
 MoveL Pr1_7, v50, fine, tPinza\WObj:=wobj_cuadricula;    !Aproximación de pieza a superficie rápida
 MoveL Pr1_8, v20, fine, tPinza\WObj:=wobj_cuadricula;    !Aproximación de pieza a superficie lenta
```

```
Abrir_Pinza;
MoveL Pr1_9, v20, fine, tPinza\WObj:=wobj_cuadricula;      !Alejamiento de la pieza delicado
MoveJ Pr1_10, v50, fine, tPinza\WObj:=wobj_cuadricula;     !Giro rápido para colocar pinza vertical
MoveL Pr1_11, v100, fine, tPinza\WObj:=wobj_cuadricula;    !Acercamiento a la pieza rápido
MoveL Pr1_12, v20, fine, tPinza\WObj:=wobj_cuadricula;     !Aproximación lenta a la pieza
Cerrar_Pinza_Linea6;
MoveL Pr1_13, v20, fine, tPinza\WObj:=wobj_cuadricula;     !Levantamiento de la pieza delicado
MoveL Pr1_14, v50, fine, tPinza\WObj:=wobj_cuadricula;     !Levantamiento de la pieza rápido
MoveJ Pr1_15, v20, fine, tPinza\WObj:=wobj_cuadricula;     !Giro de pieza
MoveL Pr1_16, v100, fine, tPinza\WObj:=wobj_cuadricula;    !Aproximación de pieza a superficie rápida
MoveL Pr1_17, v20, fine, tPinza\WObj:=wobj_cuadricula;     !Aproximación de pieza a superficie lenta
Abrir_Pinza;
MoveL Pr1_18, v20, fine, tPinza\WObj:=wobj_cuadricula;     !Alejamiento de la pieza delicado
MoveL Pr1_19, v200, fine, tPinza\WObj:=wobj_cuadricula;    !Alejamiento de la pieza rápido
MoveAbsJ calib_pos, v200, z50, tPinza;                     !Movimiento a posición ejes 0 grados
  ENDPROC !Enderezar pieza
```

12.2.2 Programa 2. Invertir programa 1

```
PROC Programa_2()
        CONST robtarget Pr1_1:=[[217.68,295.06,385.78],[0.000611983,0.0249673,0.999683,0.00323775],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_2:=[[217.69,295.06,113.61],[0.000657061,0.0249639,0.999683,0.00323605],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_3:=[[216.47,298.98,68.80],[0.000659602,0.00550069,0.999979,0.00322002],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_4:=[[216.45,298.98,81.11],[0.000602029,0.00546519,0.99998,0.00321967],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_5:=[[216.45,310.55,557.51],[0.0005491,0.00542186,0.99998,0.00321355],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_6:=[[244.08,271.25,548.32],[0.496272,-0.504473,-0.500589,-0.49863],[-1,1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_7:=[[244.03,271.26,95.69],[0.496345,-0.504476,-0.500571,-0.498572],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_8:=[[244.02,271.27,56.01],[0.49637,-0.504486,-0.500547,-0.498561],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_9:=[[243.94,235.22,141.35],[0.496518,-0.504517,-0.50046,-0.49847],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_10:=[[204.89,310.38,402.14],[0.00344962,0.704474,-0.00341789,0.709713],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_11:=[[196.58,310.38,101.81],[0.00343199,0.704574,-0.00337068,0.709614],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_12:=[[206.81,310.38,51.98],[0.0034531,0.704583,-0.0033857,0.709605],[0,0,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_13:=[[206.78,310.37,96.96],[0.003474,0.704645,-0.00335265,0.709544],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_14:=[[206.77,310.37,473.00],[0.00347011,0.70469,-0.00330426,0.7095],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_15:=[[235.49,324.36,486.77],[0.0017233,-0.705643,-0.70856,-0.0027759],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_16:=[[235.49,324.37,106.88],[0.0017083,-0.705655,-0.708548,-0.00280686],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_17:=[[235.48,324.38,80.62],[0.00169583,-0.705678,-0.708525,-0.00282601],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Pr1_18:=[[235.47,324.38,127.31],[0.00170497,-0.705665,-0.708538,-.00280591],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
MoveAbsJ calib_pos, v200, z50, tPinza;
Abrir_Pinza;
MoveJ Pr1_18, v300, z50, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_17, v20, z50, tPinza\WObj:=wobj_cuadricula;
Cerrar_Pinza_Linea6;
MoveL Pr1_16, v20, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_15, v50, fine, tPinza\WObj:=wobj_cuadricula;
MoveJ Pr1_14, v50, fine, tPinza\WObj:=wobj_cuadricula;
MoveJ Pr1_13, v50, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_12, v20, fine, tPinza\WObj:=wobj_cuadricula;
Abrir_Pinza;
MoveL Pr1_11, v20, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_10, v100, fine, tPinza\WObj:=wobj_cuadricula;
MoveJ Pr1_9, v100, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_8, v20, fine, tPinza\WObj:=wobj_cuadricula;
 Cerrar_Pinza_Linea5;
MoveL Pr1_7, v20, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_6, v50, fine, tPinza\WObj:=wobj_cuadricula;
MoveJ Pr1_5, v50, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_4, v50, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_3, v20, fine, tPinza\WObj:=wobj_cuadricula;
Abrir_Pinza;
MoveL Pr1_2, v20, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr1_1, v100, fine, tPinza\WObj:=wobj_cuadricula;

MoveAbsJ calib_pos, v200, z50, tPinza;
  ENDPROC!Invertir Programa_1
```

12.2.3 Programa 3. Dispensador.

```
PROC Programa_3()
  CONST robtarget Pr3_1:=[[304.80,297.20,111.52],[0.000633942,-0.000830955,0.999994,0.00320443],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget Pr3_2:=[[304.81,297.21,58.23],[0.000661058,-0.00080982,0.999994,0.00320436],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget Pr3_3:=[[307.77,297.21,40.85],[0.000666498,-0.000808217,0.999994,0.00320463],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget Pr3_4:=[[307.73,297.21,126.98],[0.000576858,-0.000845506,0.999994,0.00319677],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget Pr3_5:=[[406.46,297.20,139.27],[0.000531011,-0.000879934,0.999994,0.00318908],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget Pr3_6:=[[406.44,297.19,97.73],[0.000479334,-0.000848308,0.999994,0.00318813],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget Pr3_7:=[[406.42,297.19,138.73],[0.000409888,-0.000872951,0.999994,0.00318473],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
 MoveJ Pr3_1, v400, z80, tPinza\WObj:=wobj_cuadricula;   !Movimiento a posición cercana al dispensador
 MoveL Pr3_2, v400, fine, tPinza\WObj:=wobj_cuadricula;  !Acercamiento rápido para coger
 MoveL Pr3_3, v100, fine, tPinza\WObj:=wobj_cuadricula;  !Acercamiento delicado a cilindro
Cerrar_Pinza_Linea7;
 MoveL Pr3_4, v100, z80, tPinza\WObj:=wobj_cuadricula;   !Alejamiento del dispensador verticalmente
 MoveJ Pr3_5, v400, z80, tPinza\WObj:=wobj_cuadricula;   !Movimiento a posición para soltar
 MoveL Pr3_6, v100, fine, tPinza\WObj:=wobj_cuadricula;  !Acercamiento a posición para soltar cilindro
Abrir_Pinza_Rapido;
 MoveL Pr3_7, v400, z80, tPinza\WObj:=wobj_cuadricula;   !Alejamiento del dispensador verticalmente
   ENDPROC !Dispensador
```

12.2.4 Programa 4. Selección de dispensador

```
PROC Programa_4()!Selección dispensador

  !Definición de variables para pregunta
   VAR num Pr4_TPAnswer:=0;
         CONST string Pr4_TPText:="Posición del cilindro";
         CONST string Pr4_TPFK1:="Pos. 1";
         CONST string Pr4_TPFK2:="Pos. 2";
         CONST string Pr4_TPFK3:="Pos. 3";
         CONST string Pr4_TPFK4:="Pos. 4";

TPReadFK Pr4_TPAnswer, Pr4_TPText, Pr4_TPFK1, Pr4_TPFK2, Pr4_TPFK3, Pr4_TPFK4, stEmpty; !Código de pregunta
 IF  Pr4_TPAnswer=1 THEN  !Si el dispensador se encuentra en primera posición, mover hasta dejar igual
     Pr4_TPAnswer:=5;
 ENDIF
WHILE Pr4_TPAnswer>1 DO  !Mientras la posición sea mayor que 1 realizara el programa 3
   Programa_3;
    Pr4_TPAnswer:=Pr4_TPAnswer-1;
ENDWHILE

ENDPROC !Selección dispensador
```

12.2.5 Programa 5. Amontonar piezas.

```
PROC Programa_5()   !Amontonar piezas

   CONST robtarget Pr5_Pcoger:=[[174.30,459.87,79.62],[0.000807884,0.00141511,0.999994,0.00320774],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
         CONST string Pr5_TPText:="Pieza para colocar";
         CONST string Pr5_TPFK1:="Pos. 1";
         CONST string Pr5_TPFK2:="Pos. 2";
         CONST string Pr5_TPFK3:="Pos. 3";
         CONST string Pr5_TPFK4:="Pos. 4";

   VAR robtarget Pr5_POrigen;                          !Punto de origen donde empezar el movimiento selección de piezas

   VAR robtarget Pr5_Pcoger_posicion;                  !Punto de agarre de pieza
   VAR robtarget Pr5_Pcoger_aprox_posicion;            !Punto de aproximación a la pieza a la hora de coger
   VAR robtarget Pr5_Pcoger_alejar_posicion;           !Punto de separación de la pieza para evitar choques

   VAR robtarget Pr5_Pdejar;                           !Punto de depositar pieza en superficie
   VAR robtarget Pr5_Pdejar_altura;                    !Punto de depositar pieza en altura especifica
   VAR robtarget Pr5_Pdejar_aprox;                     !Punto de aproximación a depositar pieza a altura especifica
   VAR robtarget Pr5_Pdejar_alejar;                    !Punto de separación de la pieza para evitar choques según la altura
   VAR num Pr5_posicion:=0;                            !Posición de la pieza a coger
   VAR num Pr5_altura:=0;                              !Altura de dejar pieza



MoveAbsJ calib_pos, v200, z50, tPinza;                 !Movimiento de ejes a 0

WHILE Pr5_altura<4 DO                                  !Bucle que ejecuta el programa hasta que las cuatro piezas están apiladas

TPReadFK Pr5_posicion, Pr5_TPText, Pr5_TPFK1, Pr5_TPFK2, Pr5_TPFK3, Pr5_TPFK4, stEmpty;  !Pregunta de selección de pieza


 Pr5_POrigen:=Offs(Pr5_Pcoger,0,0,100);                          !Se define el punto como un desplazamiento del punto de cogida de la primera pieza
 Pr5_Pcoger_posicion:=Offs(Pr5_Pcoger,(Pr5_posicion-1)*100,0,0); !Se define este punto dependiendo de la posición de la pieza a coger, separadas entre sí 100 mm
 Pr5_Pcoger_aprox_posicion:=Offs(Pr5_Pcoger_posicion,0,0,15);    !Se define este punto como una aproximación a 15 mm de la pieza para hacer el movimiento de
aproximación con delicadeza
 Pr5_Pcoger_alejar_posicion:=Offs(Pr5_Pcoger_posicion,0,0,(Pr5_altura*80)+100);  !Se define la altura a la que debe subir la pieza después de ser cogida para evitar
choques
 Pr5_Pdejar:=Offs(Pr5_Pcoger,100,-200,0);                        !Se define el punto como un desplazamiento del punto de cogida de la primera pieza
```

```
 Pr5_Pdejar_altura:=Offs(Pr5_Pdejar,0,0,Pr5_altura*80);              !Se define la altura a la que se deposita el objeto a partir del punto de depositar pieza en superficie
 Pr5_Pdejar_aprox:=Offs(Pr5_Pdejar_altura,0,0,15);                   !Se define este punto como una aproximación a 15 mm de la pieza para hacer el movimiento de
aproximacion con delicadeza
 Pr5_Pdejar_alejar:=Offs(Pr5_Pdejar_altura,0,0,100);                !Se define la altura a la que debe subir la pieza después de ser cogida para evitar choques
    Abrir_Pinza;
    MoveJ Pr5_POrigen, v200, z50, tPinza\WObj:=wobj_cuadricula;                    !Posición origen de selección de pieza
    MoveL Pr5_Pcoger_aprox_posicion, v100, z10, tPinza\WObj:=wobj_cuadricula;      !Aproximación rápida a pieza elegida
    MoveL Pr5_Pcoger_posicion, v20, fine, tPinza\WObj:=wobj_cuadricula;            !Aproximación lenta a pieza elegida
    Cerrar_Pinza_Linea6;
    MoveL Pr5_Pcoger_aprox_posicion, v20, fine, tPinza\WObj:=wobj_cuadricula;      !Alejamiento de la pieza delicado
    MoveL Pr5_Pcoger_alejar_posicion, v100, fine, tPinza\WObj:=wobj_cuadricula;    !Alejamiento de la pieza rápido

    MoveJ Pr5_Pdejar_alejar, v100, z50, tPinza\WObj:=wobj_cuadricula;          !Movimiento a posición de dejar pieza pero a una altura adecuada para evitar choques
    MoveL Pr5_Pdejar_aprox, v100, z10, tPinza\WObj:=wobj_cuadricula;           !Aproximación rápida a la posición para depositar
    MoveL Pr5_Pdejar_altura, v20, fine, tPinza\WObj:=wobj_cuadricula;          !Aproximación delicada de la pieza a superficie para depositar
    Abrir_Pinza;
    WaitTime 1;                                                                !Tiempo de espera de 1 segundo para evitar movimientos bruscos al abrir pinza
    MoveL Pr5_Pdejar_aprox, v20, fine, tPinza\WObj:=wobj_cuadricula;                !Alejamiento de la pieza delicado
    MoveL Pr5_Pdejar_alejar, v100, z100, tPinza\WObj:=wobj_cuadricula;              !Alejamiento de la pieza rápido
    MoveL Offs(Pr5_Pdejar_alejar,0,200,0),v100, z100, tPinza\WObj:=wobj_cuadricula;  !Movimiento horizontal para colocar pieza encima de la fila de selección de pieza

    MoveJ Pr5_POrigen, v200, z50, tPinza\WObj:=wobj_cuadricula;                    !Posición origen de selección de pieza
     INCR Pr5_altura;                                                              !Incremento del valor de la altura para apilar piezas
   ENDWHILE

   MoveAbsJ calib_pos, v200, z50, tPinza;                                          !Al terminar de apilar posición de ejes 0

ENDPROC  !Amontonar piezas
```

12.2.6 Programa 6. Invertir programa 5.

```
PROC Programa_6() !Invertir Programa_5
      CONST robtarget Pr5_Pcoger:=[[174.30,459.87,79.62],[0.000807884,0.00141511,0.999994,0.00320774],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST string Pr6_TPText:="Pieza para colocar";
          CONST string Pr6_TPFK1:="Pos. 1";
          CONST string Pr6_TPFK2:="Pos. 2";
          CONST string Pr6_TPFK3:="Pos. 3";
          CONST string Pr6_TPFK4:="Pos. 4";
```

```
    VAR robtarget Pr5_Pcoger_aprox;
    VAR robtarget Pr5_POrigen;
    VAR robtarget Pr5_Pcoger_posicion;
    VAR robtarget Pr5_Pcoger_aprox_posicion;
    VAR robtarget Pr5_Pcoger_alejar_posicion;
    VAR robtarget Pr5_Pdejar;
    VAR robtarget Pr5_Pdejar_altura;
    VAR robtarget Pr5_Pdejar_aprox;
    VAR robtarget Pr5_Pdejar_aprox_altura;
    VAR robtarget Pr5_Pdejar_alejar;
    VAR num Pr5_posicion:=0;
    VAR num Pr5_altura:=3;

     MoveAbsJ calib_pos, v200, z50, tPinza;

    WHILE Pr5_altura>-1 DO
    TPReadFK Pr5_posicion, Pr6_TPText, Pr6_TPFK1, Pr6_TPFK2, Pr6_TPFK3, Pr6_TPFK4, stEmpty;

Pr5_Pcoger_aprox:=Offs(Pr5_Pcoger,0,0,15);
Pr5_POrigen:=Offs(Pr5_Pcoger,0,0,100);
Pr5_Pcoger_posicion:=Offs(Pr5_Pcoger,(Pr5_posicion-1)*100,0,0);
Pr5_Pcoger_aprox_posicion:=Offs(Pr5_Pcoger_posicion,0,0,15);
Pr5_Pcoger_alejar_posicion:=Offs(Pr5_Pcoger_posicion,0,0,(Pr5_altura*80)+100);
Pr5_Pdejar:=Offs(Pr5_Pcoger,100,-200,0);
Pr5_Pdejar_altura:=Offs(Pr5_Pdejar,0,0,Pr5_altura*80);
Pr5_Pdejar_aprox:=Offs(Pr5_Pdejar_altura,0,0,15);
Pr5_Pdejar_alejar:=Offs(Pr5_Pdejar_altura,0,0,100);

MoveJ Pr5_Pdejar_alejar, v100, z100, tPinza\WObj:=wobj_cuadricula;
MoveL Pr5_Pdejar_aprox, v100, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr5_Pdejar_altura, v20, fine, tPinza\WObj:=wobj_cuadricula;
Cerrar_Pinza_Linea6;
MoveL Pr5_Pdejar_aprox, v100, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr5_Pdejar_alejar, v100, z100, tPinza\WObj:=wobj_cuadricula;

MoveJ Pr5_Pcoger_alejar_posicion, v100, fine, tPinza\WObj:=wobj_cuadricula;
MoveL Pr5_Pcoger_aprox_posicion, v100, z10, tPinza\WObj:=wobj_cuadricula;
MoveL Pr5_Pcoger_posicion, v20, fine, tPinza\WObj:=wobj_cuadricula;
Abrir_Pinza;
MoveL Pr5_Pcoger_aprox_posicion, v100, z10, tPinza\WObj:=wobj_cuadricula;
```

```
 MoveL Pr5_Pcoger_alejar_posicion, v100, fine, tPinza\WObj:=wobj_cuadricula;

 MoveL Pr5_Pdejar_alejar, v100, z100, tPinza\WObj:=wobj_cuadricula;
 Decr Pr5_altura;
   ENDWHILE
ENDPROC
```

12.2.7 Programa 7. Rotulador.

```
PROC Programa_7()!Rotulador

! Lo primero que se debe realizar en cada programa es la declaración de las variables del programa (variables  numéricas, datos de posición, valores lógicos, etc.)
   ! Definición de punto Pr7_home, punto encima de folio con orientación de rotulador.
   CONST robtarget Pr7_home:=[[315.95,326.80,302.96],[0.00095107,-0.01701,0.99985,0.00319417],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
         ! Definición de punto Pr7_home, punto de aproximación al papel.
   CONST robtarget Pr7_0:=[[345.67,479.47,122.30],[0.000937462,-0.0170353,0.999849,0.00318498],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
         ! Definición de punto Pr7_31, punto inicial para definir los demás.
   CONST robtarget Pr7_31:=[[308.89,434.54,80.82],[0.000897264,-0.0170202,0.99985,0.00318057],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   ! Definición de los puntos de la plantilla como puntos variables ya que dependen del punto Pr7_31
   VAR robtarget Pr7_32;
   VAR robtarget Pr7_33;
   VAR robtarget Pr7_34;
   VAR robtarget Pr7_35;
   VAR robtarget Pr7_36;
   VAR robtarget Pr7_11;
   VAR robtarget Pr7_12;
   VAR robtarget Pr7_13;
   VAR robtarget Pr7_14;
   VAR robtarget Pr7_15;
   VAR robtarget Pr7_16;
   VAR robtarget Pr7_21;
   VAR robtarget Pr7_22;
   VAR robtarget Pr7_23;
   VAR robtarget Pr7_24;
   VAR robtarget Pr7_25;
   VAR robtarget Pr7_26;

   ! Definición de parámetros para pregunta
   CONST string Pr7_TPText:="Numero de dibujo";
```

```
    VAR num Pr7_TPAnswer;

    ! Creación de la matriz en base a desplazamientos del punto Pr7_31 mediante la función Offs
    Pr7_32:=Offs(Pr7_31,0,-40,0);
    Pr7_33:=Offs(Pr7_31,0,-80,0);
    Pr7_34:=Offs(Pr7_31,0,-120,0);
    Pr7_35:=Offs(Pr7_31,0,-160,0);
    Pr7_36:=Offs(Pr7_31,0,-200,0);
    Pr7_21:=Offs(Pr7_31,40,0,0);
    Pr7_22:=Offs(Pr7_31,40,-40,0);
    Pr7_23:=Offs(Pr7_31,40,-80,0);
    Pr7_24:=Offs(Pr7_31,40,-120,0);
    Pr7_25:=Offs(Pr7_31,40,-160,0);
    Pr7_26:=Offs(Pr7_31,40,-200,0);
    Pr7_11:=Offs(Pr7_31,80,0,0);
    Pr7_12:=Offs(Pr7_31,80,-40,0);
    Pr7_13:=Offs(Pr7_31,80,-80,0);
    Pr7_14:=Offs(Pr7_31,80,-120,0);
    Pr7_15:=Offs(Pr7_31,80,-160,0);
    Pr7_16:=Offs(Pr7_31,80,-200,0);
    Pr7_TPAnswer:=0;

! Una vez definidos las variables del programa se procede a realizar la ejecución de las funciones;
    ! La siguiente función sirve para mostrar la pregunta "Numero de dibujo" en pantalla y asigna a Pr7_TPAnswer la respuesta.
    TPReadNum Pr7_TPAnswer, Pr7_TPText;

! Una vez elegido el dibujo el robot genera unos movimientos comunes a todos los dibujos para orientarse y posicionarse.
    MoveJ Pr7_home, v150, fine, tPinza\WObj:=wobj_cuadricula;    !Movimiento y orientación a punto encima del folio.
    MoveJ Pr7_0, v150, fine, tPinza\WObj:=wobj_cuadricula;       !Aproximación a folio manteniendo orientación.
    Cerrar_Pinza_Linea8;                                         !Cierre de la pinza para asegurar que el rotulador está bien sujeto.

! Dependiendo del dibujo elegido el robot entrara en el bloque IF correspondiente.
    IF Pr7_TPAnswer=1 THEN! Como ejemplo se proporciona el dibujo de las iniciales ACL
        !A
        MoveL Offs(Pr7_31,0,0,10), v100, z20, tPinza\WObj:=wobj_cuadricula; !Aproximación a 10mm del punto 31 de la cuadricula.
        MoveL Pr7_31, v50, fine, tPinza\WObj:=wobj_cuadricula;           !Baja al punto 31, donde comienza el dibujo de la letra A.
        MoveL Pr7_11, v50, fine, tPinza\WObj:=wobj_cuadricula;           !Movimiento en línea recta de 31 a 11.
        MoveL Pr7_12, v50, fine, tPinza\WObj:=wobj_cuadricula;           !Movimiento en línea recta de 11 a 12.
        MoveL Pr7_22, v50, fine, tPinza\WObj:=wobj_cuadricula;           !Movimiento en línea recta de 12 a 22.
        MoveL Pr7_21, v50, fine, tPinza\WObj:=wobj_cuadricula;           !Movimiento en línea recta de 22 a 21.
        MoveL Pr7_22, v50, fine, tPinza\WObj:=wobj_cuadricula;           !Movimiento en línea recta de 11 a 22.
```

```
    MoveL Pr7_32, v50, fine, tPinza\WObj:=wobj_cuadricula;          !Movimiento en línea recta de 22 a 32.
    MoveL Offs(Pr7_32,0,0,10), v100, z20, tPinza\WObj:=wobj_cuadricula; !En el punto 32 se levanta el rotulador el papel a 10mm para pasar a la siguiente letra.

    !C
    MoveL Offs(Pr7_34,0,0,10), v100, z20, tPinza\WObj:=wobj_cuadricula; !Aproximación a 10mm del punto 34 de la cuadricula.
    MoveL Pr7_34, v50, fine, tPinza\WObj:=wobj_cuadricula;          !Baja al punto 34, donde comienza el dibujo de la letra C.
    MoveC Pr7_23,Pr7_14,v50, fine, tPinza\WObj:=wobj_cuadricula;      !Movimiento circular desde 34 a 14 pasando por 23.
    MoveL Offs(Pr7_14,0,0,10), v100, z20, tPinza\WObj:=wobj_cuadricula; !En el punto 33 se levanta el rotulador el papel a 10mm para pasar a la siguiente letra.

    !L
    MoveL Offs(Pr7_15,0,0,10), v100, z20, tPinza\WObj:=wobj_cuadricula; !Aproximación a 10mm del punto 15 de la cuadricula
    MoveL Pr7_15, v50, fine, tPinza\WObj:=wobj_cuadricula;          !Baja al punto 15, donde comienza el dibujo de la letra L.
    MoveL Pr7_35, v50, fine, tPinza\WObj:=wobj_cuadricula;          !Movimiento en línea recta de 15 a 35.
    MoveL Pr7_36, v50, fine, tPinza\WObj:=wobj_cuadricula;          !Movimiento en línea recta de 35 a 36.
    MoveL Offs(Pr7_36,0,0,10), v100, z20, tPinza\WObj:=wobj_cuadricula; !En el punto 36 se levanta el rotulador el papel a 10mm.

ENDIF! 1. ACL
```

12.3 Planos

A continuación se muestran los planos generados para la realización de este TFG.

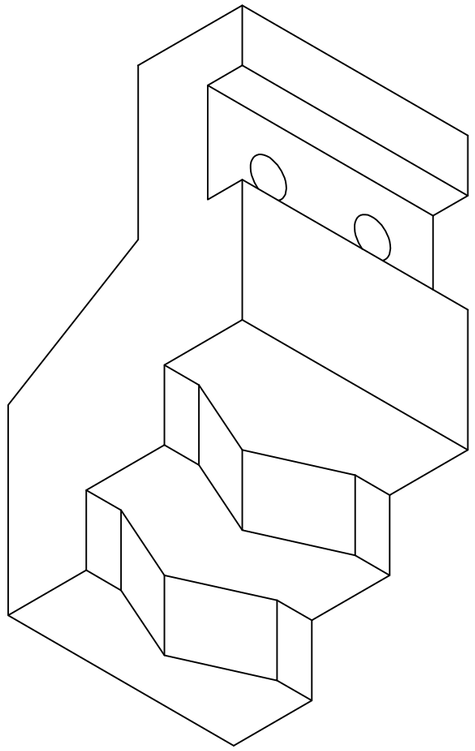
12.3.1 Mordazas

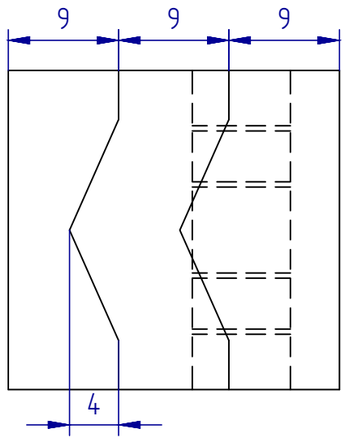
12.3.2 Adaptador retráctil para rotulador

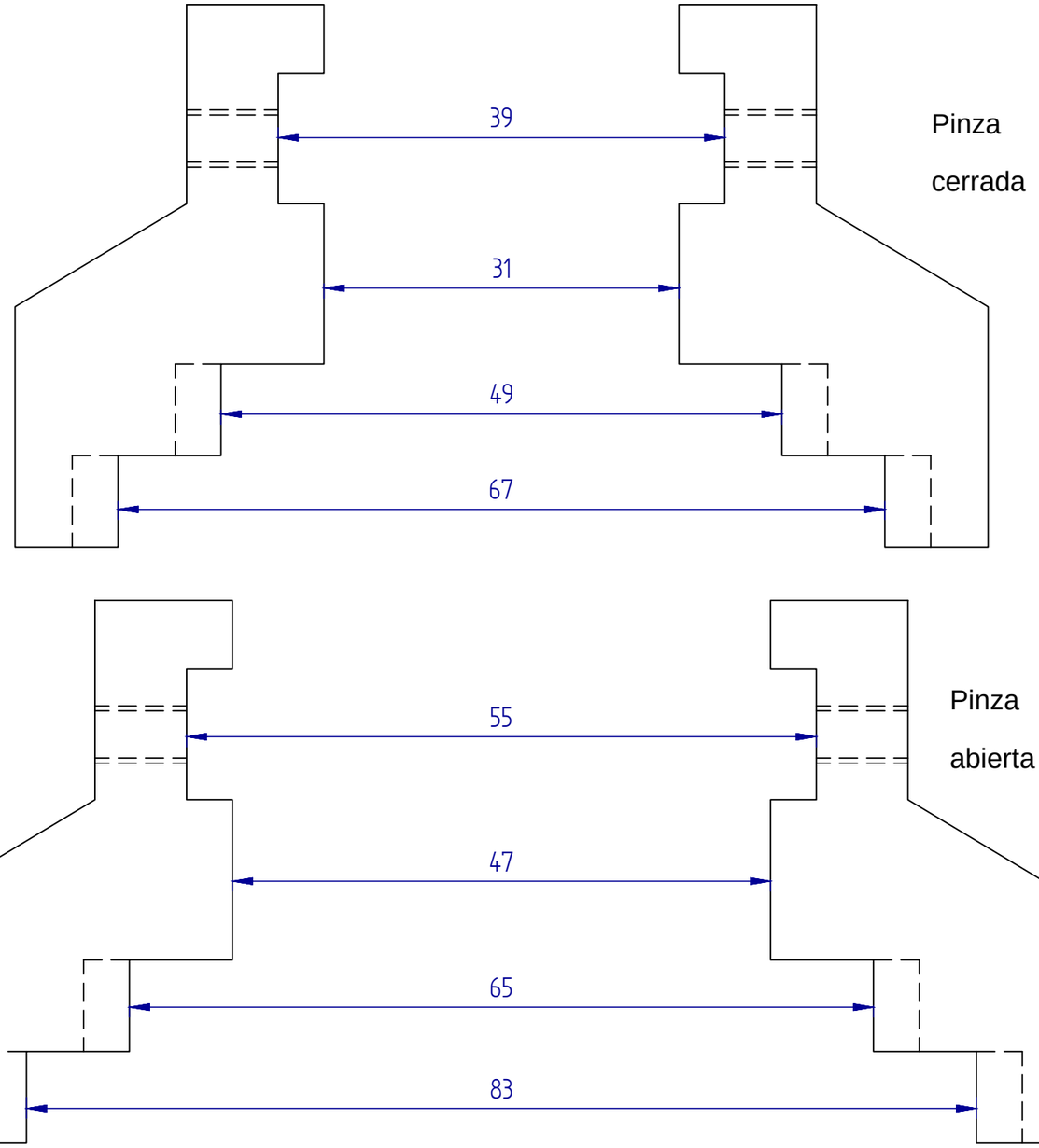
12.3.3 Adaptador rotulador-mordazas

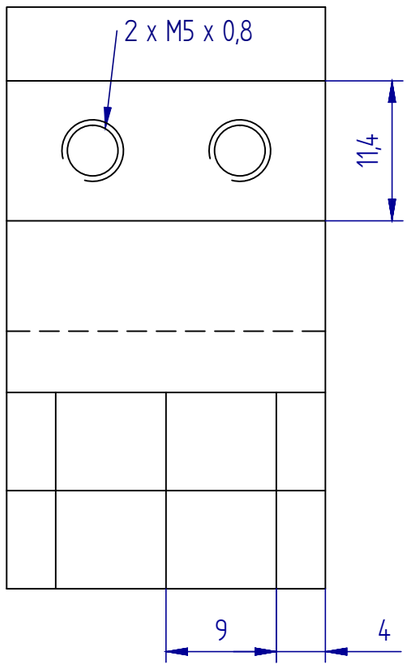
12.3.4 Pieza de prácticas

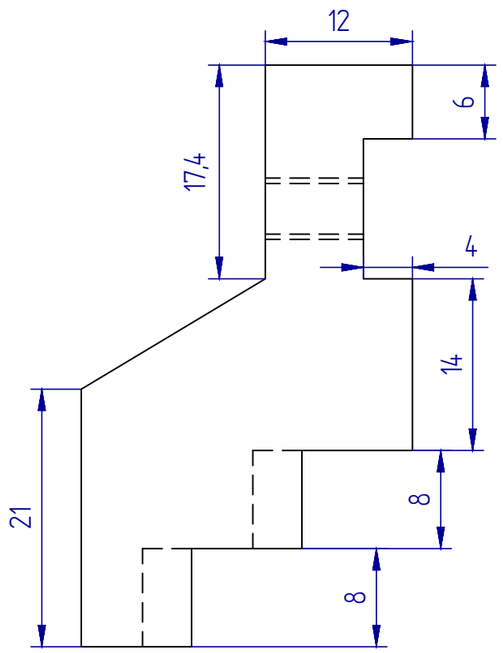
12.3.5 Dispensador









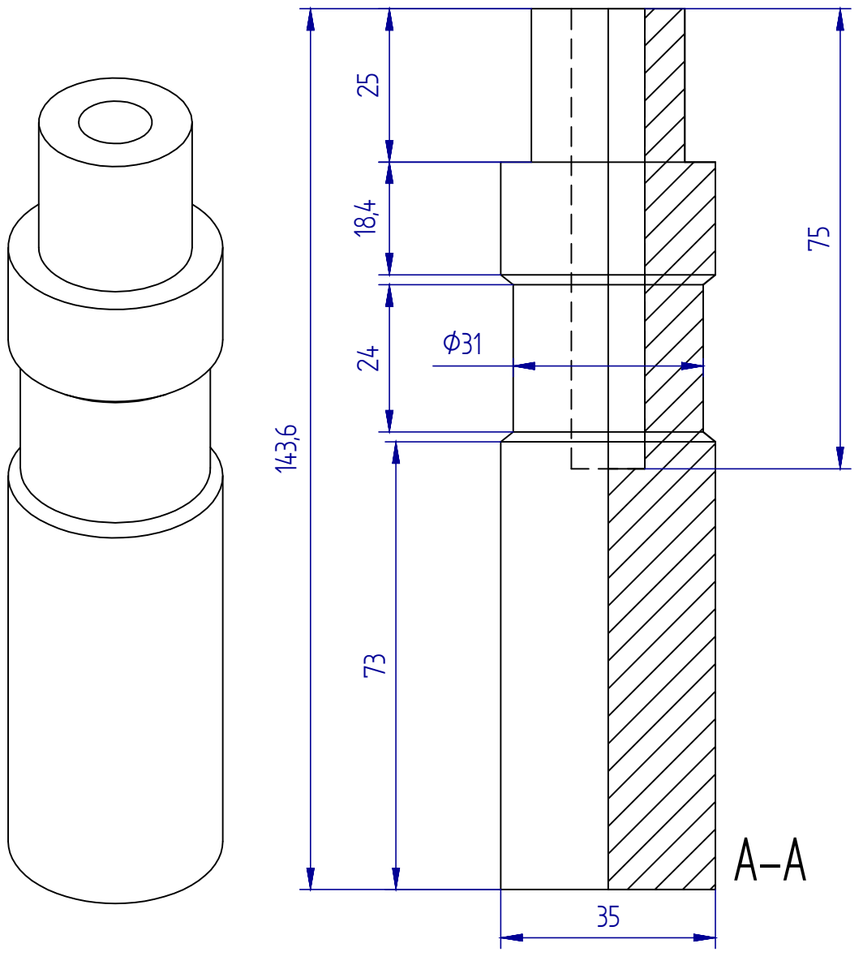


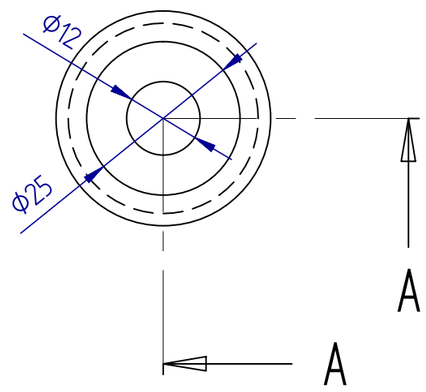
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO		**A.Córdoba**		
COMPROBADO				

ESCALA: **2:1**	Título del TFG: **Puesta en marcha de brazo robótico y desarrollo de aplicaciones** Título del Plano: **Anexo 3.1: Mordazas**	Nº PLANO **1/5** SUSTITUYE A: SUSTITUIDO POR:

PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK

PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK



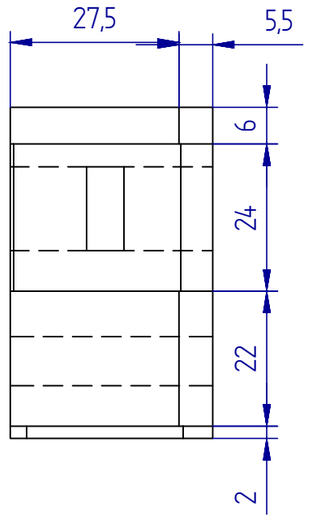


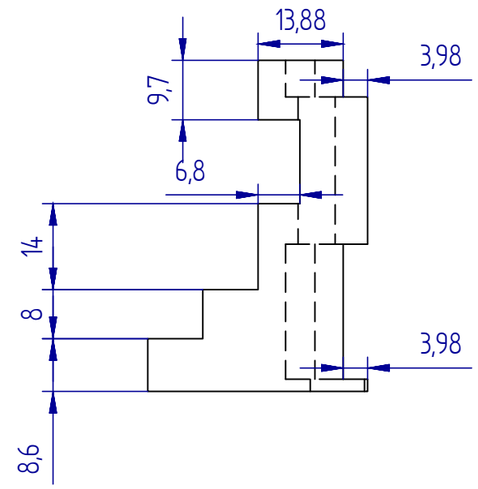
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO		**A.Córdoba**		
COMPROBADO				

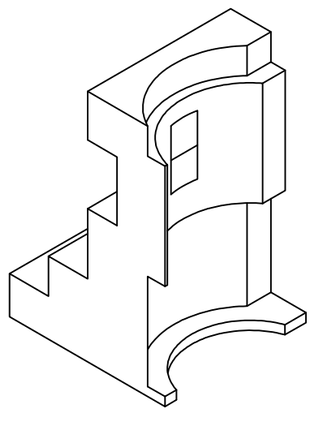
ESCALA: **1:1**	Título del TFG: **Puesta en marcha de brazo robótico y desarrollo de aplicaciones** Título del Plano: **Anexo 3.2: Adaptador retráctil para rotulador**	Nº PLANO **2/5** SUSTITUYE A: SUSTITUIDO POR:

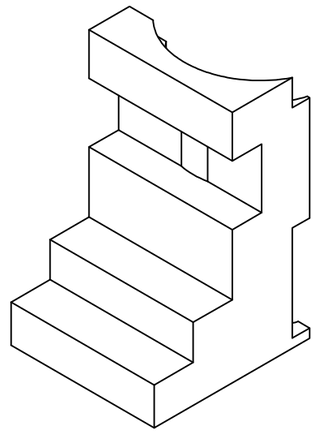
PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK

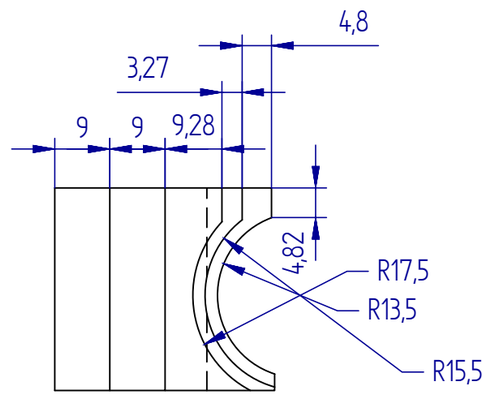
PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK









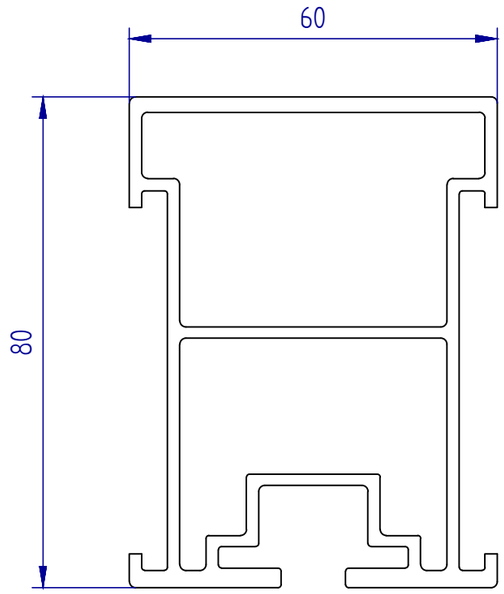


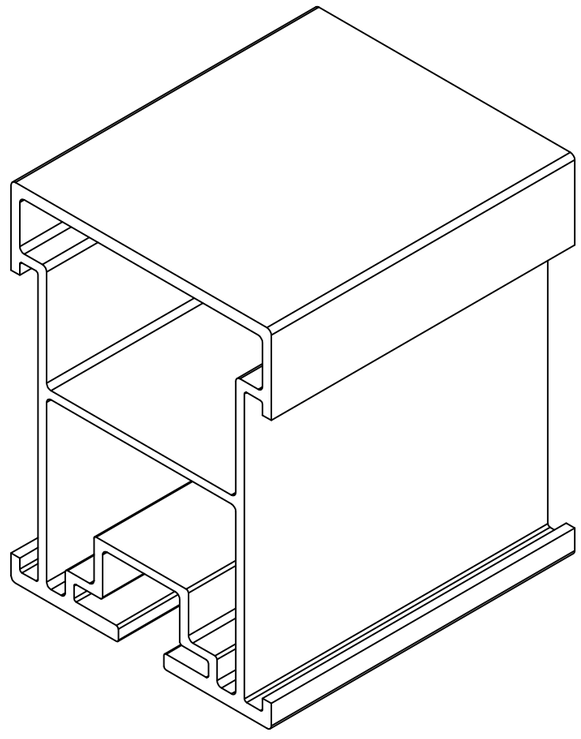
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO		**A.Córdoba**		
COMPROBADO				

ESCALA: **1:1**	Título del TFG: **Puesta en marcha de brazo robótico y desarrollo de aplicaciones** Título del Plano: **Anexo 3.3: Adaptador rotulador-mordazas**	Nº PLANO **3/5** SUSTITUYE A: SUSTITUIDO POR:

PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK

PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK

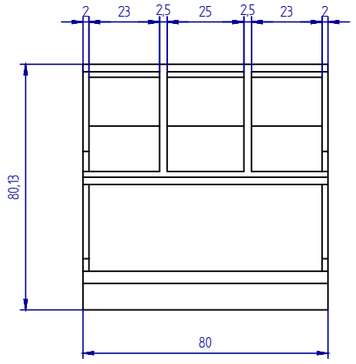


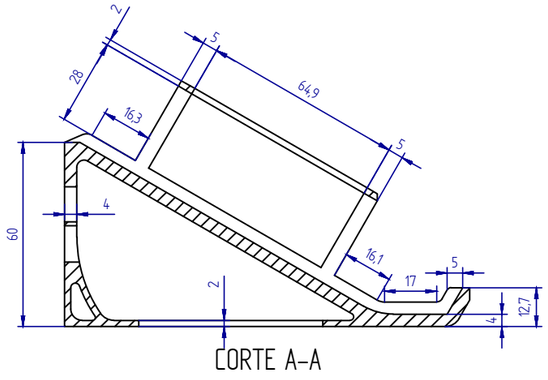


	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO		**A.Córdoba**		
COMPROBADO				

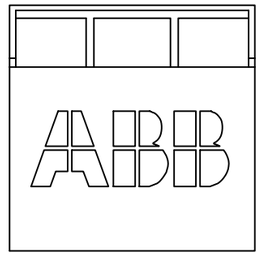
ESCALA: **1:1**	Título del TFG: **Puesta en marcha de brazo robótico y desarrollo de aplicaciones** Título del Plano: **Anexo 3.4: Pieza de prácticas**	Nº PLANO **4/5** SUSTITUYE A: SUSTITUIDO POR:

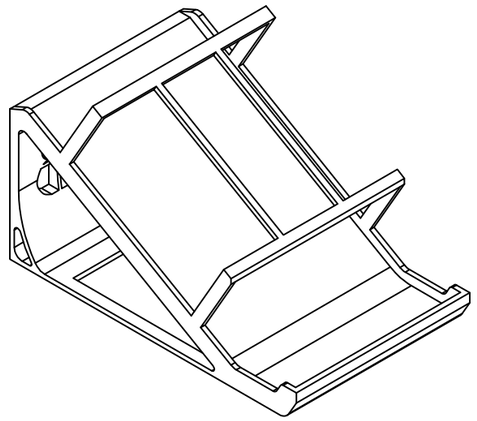
PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK

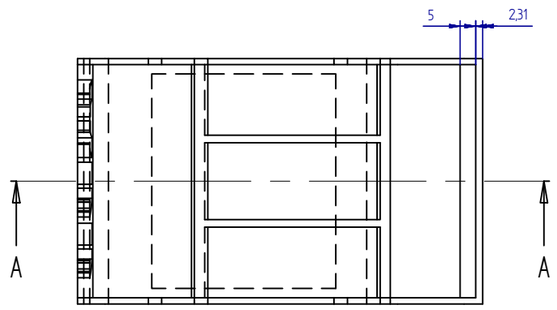


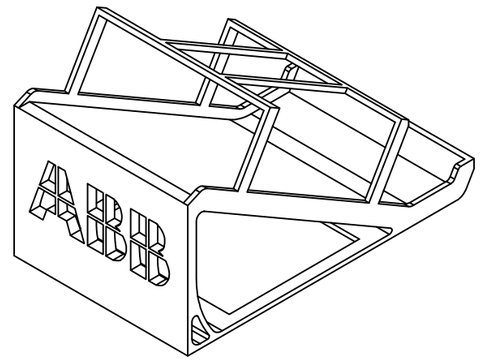


CORTE A-A









	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES	
DIBUJADO		**A.Córdoba**			
COMPROBADO					
ESCALA: **1:2**	Título del TFG: **Puesta en marcha de brazo robótico y desarrollo de aplicaciones**				Nº PLANO **5/5**
	Título del Plano: **Anexo 3.5: Dispensador**				SUSTITUYE A:
					SUSTITUIDO POR:

PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK

PRODUCIDO POR UN PRODUCTO EDUCATIVO DE AUTODESK