VISTO BUENO Y CALIFICACIÓN DEL PROYECTO

Título proyecto:

Proyectante: _____

_____

Director/es: _____

VºBº director/es del proyecto: Fdo.: Fdo.:

Lugar y fecha: _____

CALIFICACIÓN NUMÉRICA ☐ MATRÍCULA DE HONOR ☐

Méritos justificativos en el caso de conceder Matrícula de Honor:

_____

_____

_____

_____

Conforme presidente: Fdo.:	Conforme secretario: Fdo.:	Conforme vocal: Fdo.:

Lugar y fecha: _____

“Nada es tan fácil como parece,…
ni tan difícil como lo explica el manual”

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

INGENIERÍA INDUSTRIAL

"PUESTA EN FUNCIONAMIENTO Y PROGRAMACIÓN DE UNA CÉLULA DE FABRICACIÓN FLEXIBLE"

PROYECTO FIN DE CARRERA

Septiembre 2009

AUTOR: Joaquín Fernández Gálvez
DIRECTORES: María Asunción Vicente Ripoll
César Fernández Peris

ÍNDICE .

1.- Introducción.

Aplicando los conocimientos adquiridos a lo largo de la carrera en el ámbito de la robotización y la automatización, decidí llevar a cabo un estudio en el cual aplicar y ampliar dichos conocimientos.

En la realización de este estudio acerca de una célula de fabricación flexible, ha sido posible el aprendizaje y manejo de programas informáticos necesarios para la comunicación entre diversos componentes informáticos, la programación de autómatas para llevar a cabo tareas cíclicas, el software necesario para controlar el brazo robot y la adquisición y tratamiento de imágenes para el reconocimiento de objetos. Además del mantenimiento y puesta a punto de cada uno de los componentes que forman dicha célula de fabricación flexible.

La automatización de procesos industriales es un área de la Ingeniería Industrial clave en nuestros días. La competencia, la optimización de recursos, la minimización del tiempo de fabricación y la filosofía empresarial "Just In Time" son suficientes para justificar la realización de este estudio.

Con una gran aceptación, las células de fabricación flexible están extendidas a todas las cadenas de montaje que forman parte de cualquier sistema de producción dentro de una empresa. Son parte fundamental dentro de la Industria y su buena acogida es debida a que:

- Aumentan la productividad de un ciclo productivo al reducir costes y tiempo de producción, además de mejorar la calidad del producto final.
- Mejora la disponibilidad de los productos, al proveer las cantidades solicitadas en el momento justo.
- Reducen puestos de trabajo repetitivos y disminuyen los tiempos de ejecución de los mismos.
- Mejora la seguridad del sistema de producción al ser una maquina la encargada de manipular objetos que resulten dañinos, ya sea por corrosión, temperatura o radiación para el operario.
- Se integra la gestión y producción.

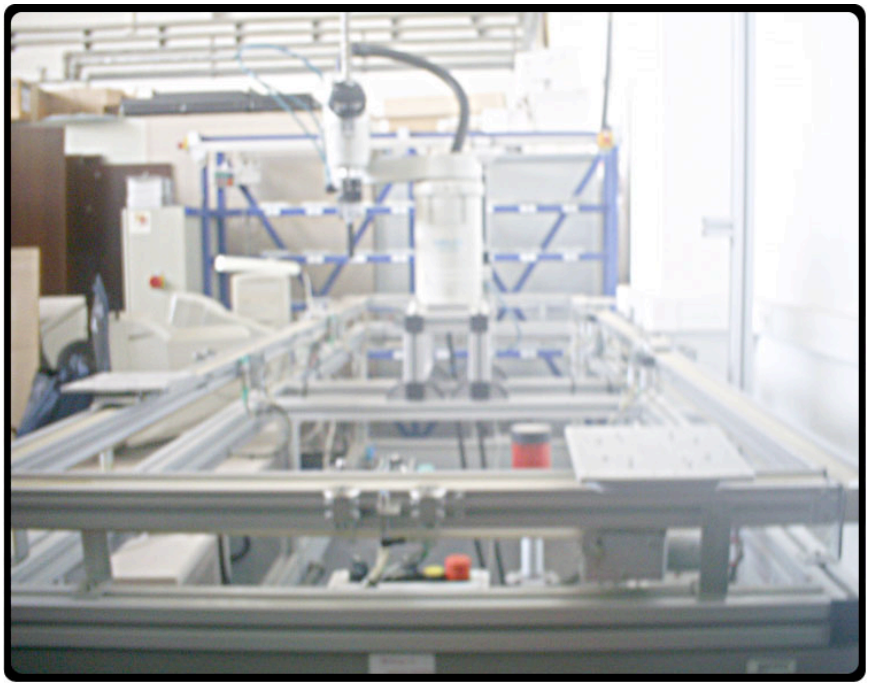
En nuestros días una célula de fabricación flexible es una buena solución para mejorar la productividad de un ciclo productivo. Además de aumentar la calidad del producto final, establece una mejor organización del sistema de producción y crea una estructura ordenada, para la mejor ubicación de diversos puestos de control de calidad.

1.1.- Planteamiento y resolución del problema.

Inicialmente nos encontramos con diferentes sistemas autónomos, con los cuales se puede formar una célula de fabricación flexible. Siendo necesaria antes una puesta a punto para después, establecer una comunicación entre cada uno de estos componentes que permita su correcto funcionamiento.

El conjunto de los elementos que forman esta célula de fabricación flexible pueden agruparse como:

- Cinta transportadora y autómata, encargado de controlar dicha cinta.
- Brazo robot con su controlador.
- Cámara Web encargada de la adquisición de imágenes.
- PC encargado de la comunicación y tratamiento de imágenes.
- Tarjeta de adquisición de datos



El problema se plantea al querer que todos los componentes de dicha célula de fabricación operen sincronizadamente. Con este fin será necesario programar cada uno de los componentes y establecer una comunicación entre ellos para realizar una tarea específica.

La tarea a realizar por esta célula de fabricación flexible será la de transportar, reconocer y manipular diferentes objetos en forma, siempre dentro de un rango de tamaño establecido por las características de los componentes que forman dicha célula de fabricación. Esta operación resulta muy orientativa para posibles futuros usos, tales como: soldadura, aplicación de pintura, control de calidad.

El estudio se va a centrar en la puesta a punto, programación y mantenimiento de una célula de fabricación flexible, con el objetivo del reconocimiento mediante visión artificial y la manipulación de diferentes piezas o productos finales con un brazo robot dentro de un proceso o cadena de montaje.

1.2.- Sistemas de Transporte Automatizado.

Parte fundamental en cualquier cadena de montaje, los sistemas de transporte automatizado tienen la tarea de transportar de una forma segura y eficaz un producto final o intermedio desde una posición inicial a otra final.

Permiten enlazar los sectores de la producción con el almacén, la zona de preparación y de picking e integrar también la zona de carga para que el sistema funcione con la máxima eficacia.

Durante la elaboración de una solución logística, la selección del sistema de transporte es decisiva para el rendimiento de la línea. Para encontrar la solución óptima se determinan con anterioridad las relaciones entre los trayectos, las distancias entre las posiciones objetivo, las tareas de transporte necesarias y los rendimientos exigidos.

Igualmente hay que determinar el espacio disponible y los costos presupuestarios en el momento de seleccionar una variante. Dependiendo del peso de estos criterios se utilizarán transportadores de palets, monocarriles aéreos eléctricos o vehículos de guiado automático.

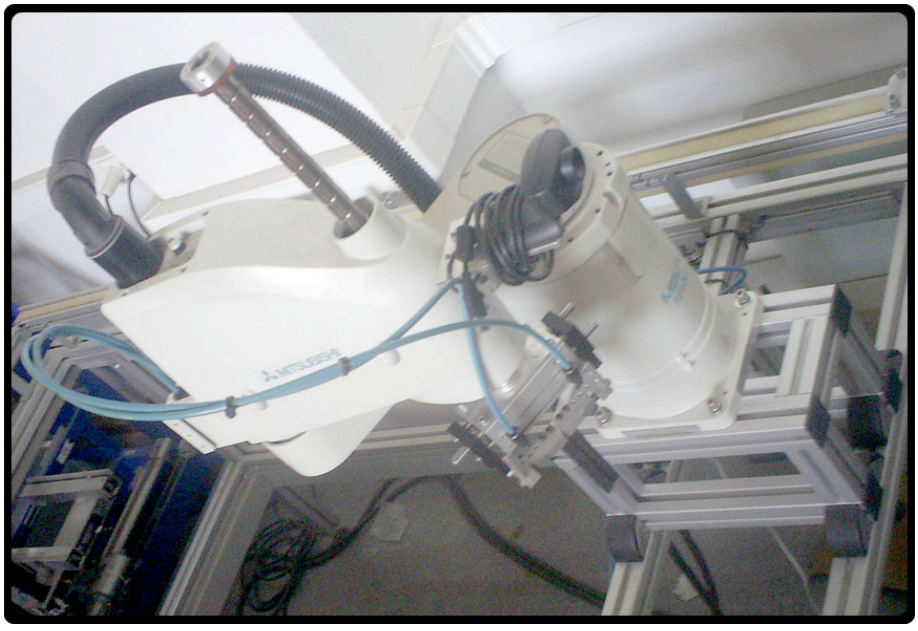
1.3.- Sistemas de Manipulación Automáticos.

Otro campo de la Ingeniería muy importante, es el proceso de manipular objetos a través de un proceso cíclico y automático, como por ejemplo, a través de un brazo robot. La tarea de manipulación es un término muy general con diversas aplicaciones, algunos ejemplos serian desde coger y cambiar de posición objetos, soldar, cortar, pintar, etc.

La manipulación de piezas o productos finales puede resultar una operación que requiere mucho tiempo y trabajo cuando sólo se dispone de recursos manuales. Además, un sistema de manipulación automático permite:

- Reducir el trabajo manual.
- Trabajar en turnos múltiples.
- Trabajar en entornos fríos.
- Mantener el inventario actualizado en tiempo real.
- Operar el sistema en entornos sin luz.
- Mejorar la calidad.
- Manipular los bienes con cuidado.
- Mantener un flujo de trabajo constante.

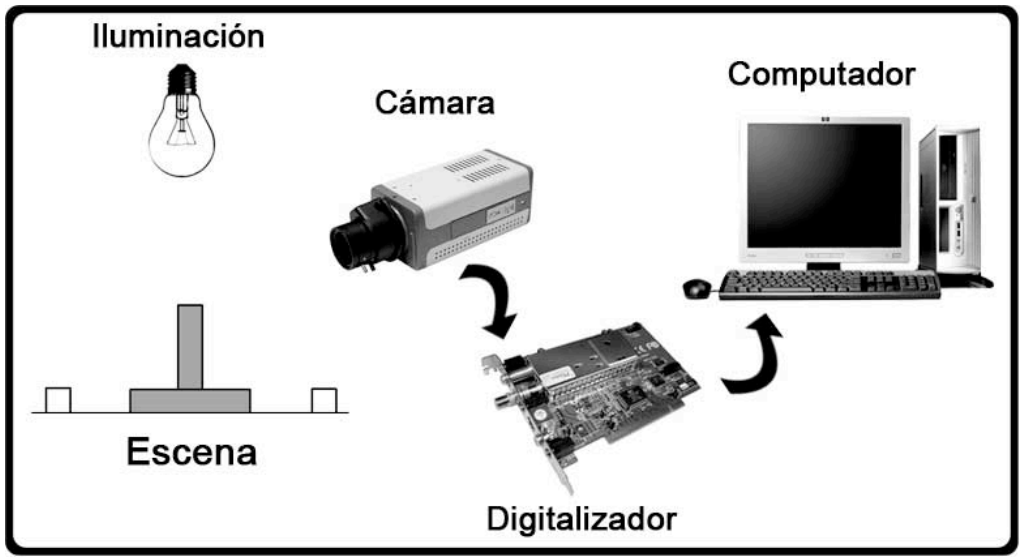
En lo que a este proyecto concierne, la tarea de manipulación consistirá en coger un objeto de forma firme y segura, a partir de la información obtenida del análisis de la imagen adquirida por la cámara Web, para su posterior colocación en un lugar fijado.



1.4.- Sistemas de Visión Artificial.

El objetivo principal de cualquier sistema de visión artificial no es otro que el de obtener información del mundo físico partiendo de una imagen utilizando para ello un computador.

Los componentes que forman un sistema de visión artificial son:



Iluminación.- Encargada de iluminar el objeto a analizar.

Escena.- Situación real de trabajo de la que se pretende extraer información.

Cámara.- Dispositivo electrónico para extraer información de la escena.

Digitalizador.- Realiza la operación de digitalizar la información extraída por la cámara.

Computador.- Analizará y extraerá información en base a los datos digitalizados proporcionados por el digitalizador.

Las ventajas de los sistemas de visión artificiales son numerosas debido a su mejor evaluación de magnitudes físicas y su buen desempeño de tareas rutinarias. Supera con creces las capacidades del ojo humano susceptible a la fatiga y poca objetividad, además de la posibilidad de trabajar en frecuencias, como por ejemplo infrarrojos, las cuales el ojo humano es incapaz de visualizar.

La imagen debe ser adquirida por un dispositivo físico electrónico, el cual captara la información luminosa y la transformará en una imagen digital, la cual podrá ser interpretada y analizada por un computador, al estar formada por una matriz de números donde cada componente representa el nivel de luminosidad captado por un píxel.

Los objetivos de las aplicaciones industriales de la visión por computador en los procesos Industriales son:

- Mejora en la calidad de la inspección.
- Mejora en la cantidad de la inspección.
- Sustitución de los operarios.
- Integración en el entorno automatizado.
- Incremento de la fiabilidad.

Las limitaciones de las aplicaciones industriales de la visión por computador se centran en la poca adaptación a situaciones imprevistas y la utilización de métodos indirectos en la determinación de las características.

Por ello hay que crear un espacio de trabajo donde el tipo y la intensidad de la luz sean constantes optimizando así las condiciones lumínicas, para poder ofrecer así los mejores resultados en el análisis de imágenes mediante visión artificial.

2.- Antecedentes.

En la actualidad, debido a las condiciones de mercado, casi todas las empresas dedicadas a la producción o montaje de cualquier producto con una alta demanda de productos finalizados, excluyendo de este grupo las empresas dedicadas a trabajos artesanos, incluyen en su sistema de producción, tareas automatizadas controladas por una o varias computadoras.

Un ejemplo claro sería la industria de la alimentación, donde grandes demandas de productos acabados como la bollería, o productos primarios como la leche, exigen la automatización de los procesos necesarios para garantizar tanto la higiene como el precio del producto en el mercado.

Este año El Centro Español de Logística (CEL) entregó al Grupo Leche Pascual el premio “CEL Empresa 2009” por el proyecto de innovación logística 'Apolo' (Aranda Plataforma Operativa Logística) en su plataforma de Aranda de Duero (Burgos). La planificación y automatización de los procesos, la minimización de los 'stocks' o la optimización del tiempo de preparación de la carga fueron decisivos en la valoración del proyecto.

El proyecto "Apolo" supuso la ampliación de la capacidad de almacenaje automático de Grupo Leche Pascual en 17.000 paletas, además de la construcción de varias zonas de carga y descarga de camiones y la integración de sistemas de información para alcanzar el objetivo de "cero papel".

Otro ejemplo podría ser la industria del automóvil, donde los costes, la precisión y la reducción de los tiempos de producción son la clave para poder competir en el mercado. Para conseguir estos objetivos la automatización debe de estar presente en los ciclos productivos de las empresas como Seat.

Seat destaca como marca autónoma dentro del mercado automovilístico. Debido a su diseño deportivo la empresa a tenido un gran éxito en los últimos años. En 2006 se produjeron un total de 400.000 vehículos, el 60 por ciento de los cuales fueron exportados fuera de España. Todos los modelos de coches que produce Seat como el Alhambra, el Altea, el Córdoba, el Ibiza, el León o el Toledo, incluyen en su cadena de producción soluciones tecnológicas de automatización y seguridad elaboradas por Sick.

Así, por ejemplo, en los últimos años se han instalado en Martorell centenares de escáneres láser de seguridad PLS y cortinas de luz de seguridad FGS y MSL, ya sea de forma directa por parte de Seat o integrados en las máquinas.

En algunos casos, el tamaño del producto a fabricar impone la automatización de los ciclos productivos. Siendo el tamaño de los componentes que forman el producto, tan pequeños se debe recurrir al empleo de robots para la producción de productos como teléfonos móviles, GPS, Ordenadores portátiles, cámaras digitales.

Una empresa grande en el sector de la telefonía móvil sin duda es Nokia empresa finlandesa que actualmente produce 900.000 teléfonos al día. Ésa es la cifra que producen las nueve fábricas que Nokia tiene actualmente en funcionamiento por todo el mundo. En vez de optar por la subcontratación, la firma finlandesa considera la producción propia como uno de sus mayores activos, y por eso ha refinado sus procesos de fabricación para poder manejar 275 millones de componentes cada día y obtener así, los márgenes más competitivos del sector.

En cada sector de la industria, la automatización está presente en las empresas de diversas formas. Puede formar parte de la célula de almacenamiento destinada a los productos finales o a las materias primas ofreciendo desde sistemas de transporte automatizado hasta completos sistemas de almacenaje automatizado, donde cada producto es reconocido y catalogado haciendo que la dirección y organización de Stock sea más fácil y efectiva.

En los ciclos de producción es donde la automatización toma un papel predominante, en el montaje de componentes ya sea por que son muy pequeños, frágiles o incluso nocivos, los sistemas electrónicos automáticos realizan las tareas más complejas con gran precisión. La soldadura automática, la inyección de resina, la fabricación de piezas de plástico mediante moldes son ejemplos de células de fabricación flexible dentro de un ciclo productivo.

Otro de los campos donde la automatización es aplicable es el control de la calidad. La precisión de los sensores electrónicos y su capacidad de observar propiedades físicas no apreciables para los sentidos humanos hacen más precisa y rápida la tarea de supervisión y control de calidad a lo largo de todo un ciclo productivo.

La visión artificial es una forma de aplicar la capacidad computacional a la tarea de supervisar y formar un punto de control dentro de un ciclo productivo. La detección de fallos en soldaduras, la selección de objetos por color, el reconocimiento de objetos son ejemplos de la aplicación de la visión artificial.

3.- Bases del trabajo.

El objetivo de este estudio es la puesta a punto y la implantación de un programa de control, el cual sea capaz de automatizar la tarea de adquisición y reconocimiento de imágenes para posterior manipulación de piezas a partir de la información obtenida del proceso de imágenes.

Para llevar a cabo esta tarea, los componentes disponibles en la Universidad a utilizar serán:

- Una cinta transportadora de la empresa Festo, la cual será controlada por el autómata.
- Un Autómata Siemens S-7, con una CPU 224.
- Un robot Industrial de la marca MITSUBISHI serie RH, con su controlador.
- Una cámara Web de gama media encargada de la adquisición de imágenes.
- Un PC doméstico de gama media con el sistema operativo Windows XP.
- Una tarjeta de adquisición de datos modelo ACL-8112PG.

En un principio, el autómata utilizado fue un Siemens S7-300 de gama alta, con un módulo de comunicación para una red Profibus. El cual fue sustituido por un autómata de gama media debido a que, al utilizar una cámara para detectar la pieza, no era necesaria la comunicación a través de la red Profibus para la determinación del contenido de las bandejas.

A continuación se describe y comenta, con más profundidad, cada uno de los componentes mencionados anteriormente.

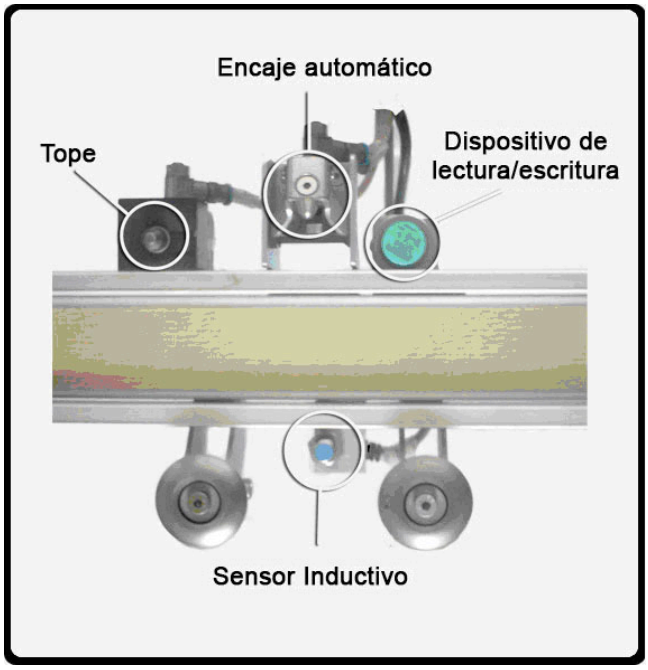
3.1.- Cinta transportadora.

Como parte importante de una célula de fabricación flexible, se dispone de un modulo de transporte de la empresa Festo, con una configuración especial para la universidad.

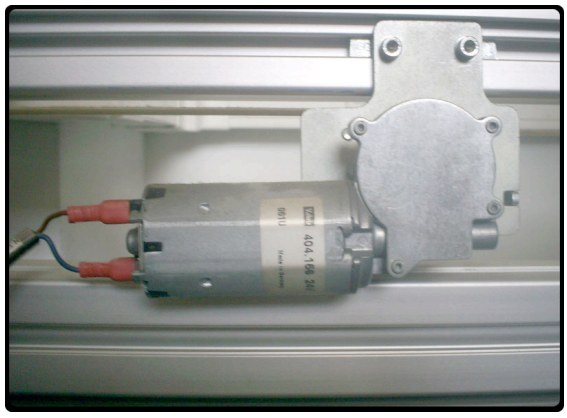
3.1.1.- Características de la cinta transportadora.

La cinta transportadora a utilizar, esta formada por un soporte de acero sobre el cual van superpuestas cuatro cintas de banda formando un circuito cerrado de dimensiones 3.5 x 1 m.

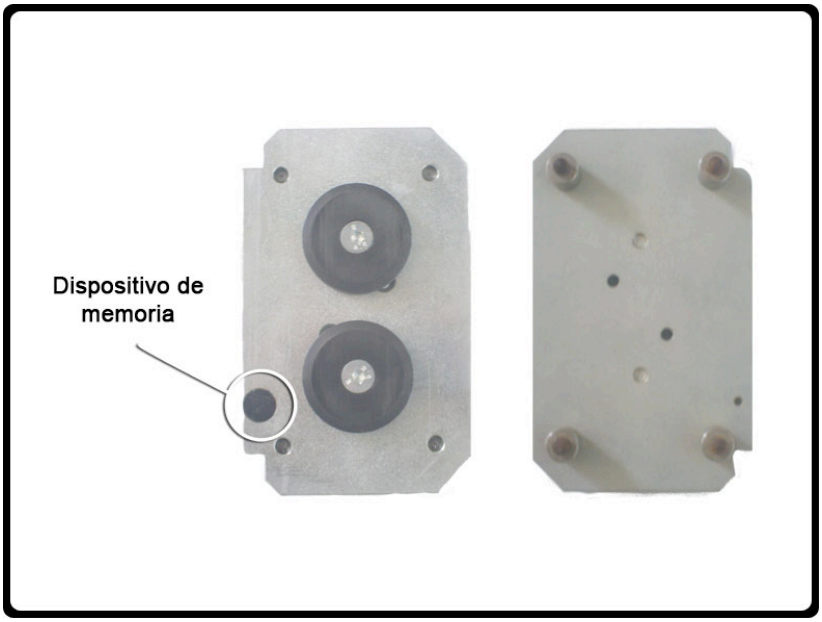
Seis estaciones donde paran las bandejas, cada estación contiene unos topes y unos encajes neumáticos, un sensor inductivo encargado de detectar si hay bandeja encima de la estación y un dispositivo de lectura y escritura (el cual no se utilizará en este proyecto).



Cuatro motores eléctricos de 24 V, cada uno de los cuales es situado al lado de la cinta que pone en movimiento.



Cuatro bandejas que actúan como soporte para transportar las piezas hacia las estaciones a través de las cintas de banda.



Un cuadro de mando, situado sobre el panel de control del módulo, permite su operabilidad mediante la actuación manual sobre unos accionamientos conectados a las entradas del autómata mediante un modulo de conexionado.

En un lateral del soporte de acero, hay una zona habilitada para instalar un panel de control, en el cual se pueden distinguir los siguientes elementos:

-La fuente de alimentación, que alimentara las salidas por relés del autómata y los sensores inductivos de las estaciones de la cinta transportadora.

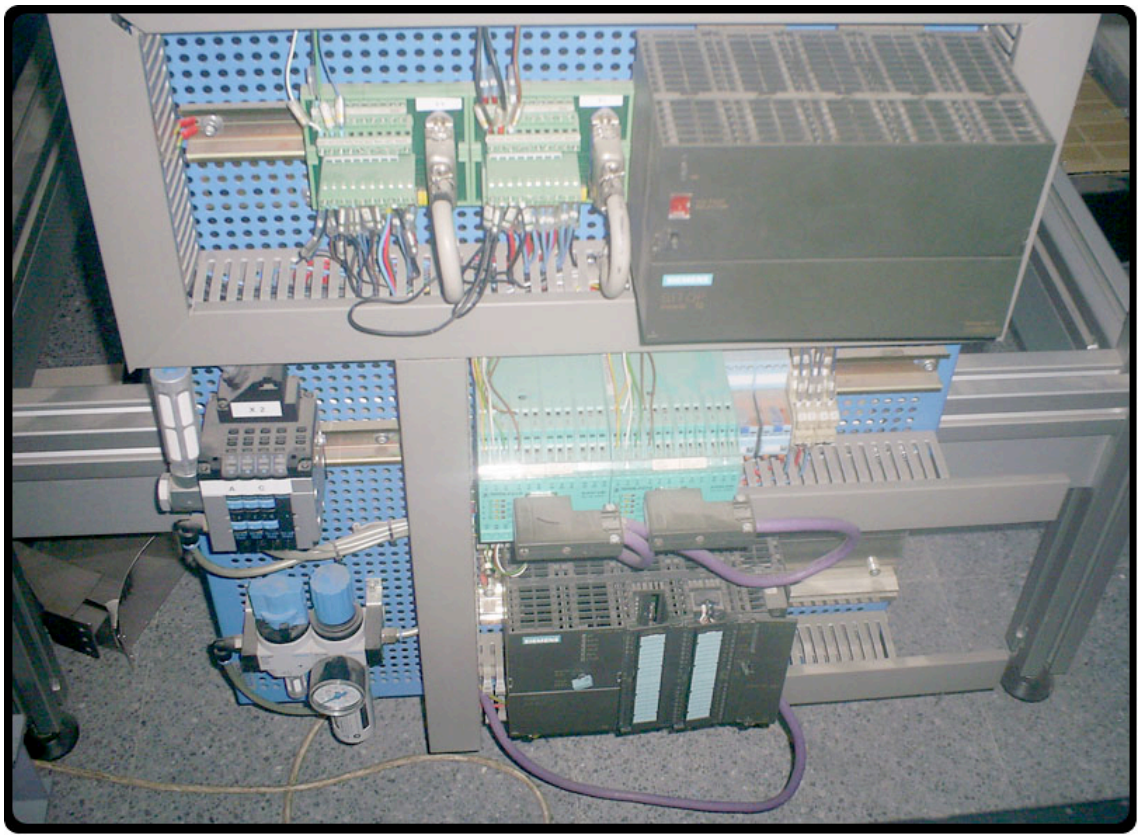
-El autómata S7-224xp con todos sus componentes.

-Los módulos de conexionado de los sensores y actuadores de módulo.

-Dos relés, cada uno de ellos activa dos motores DC de 24 V que accionan las cintas transportadoras.

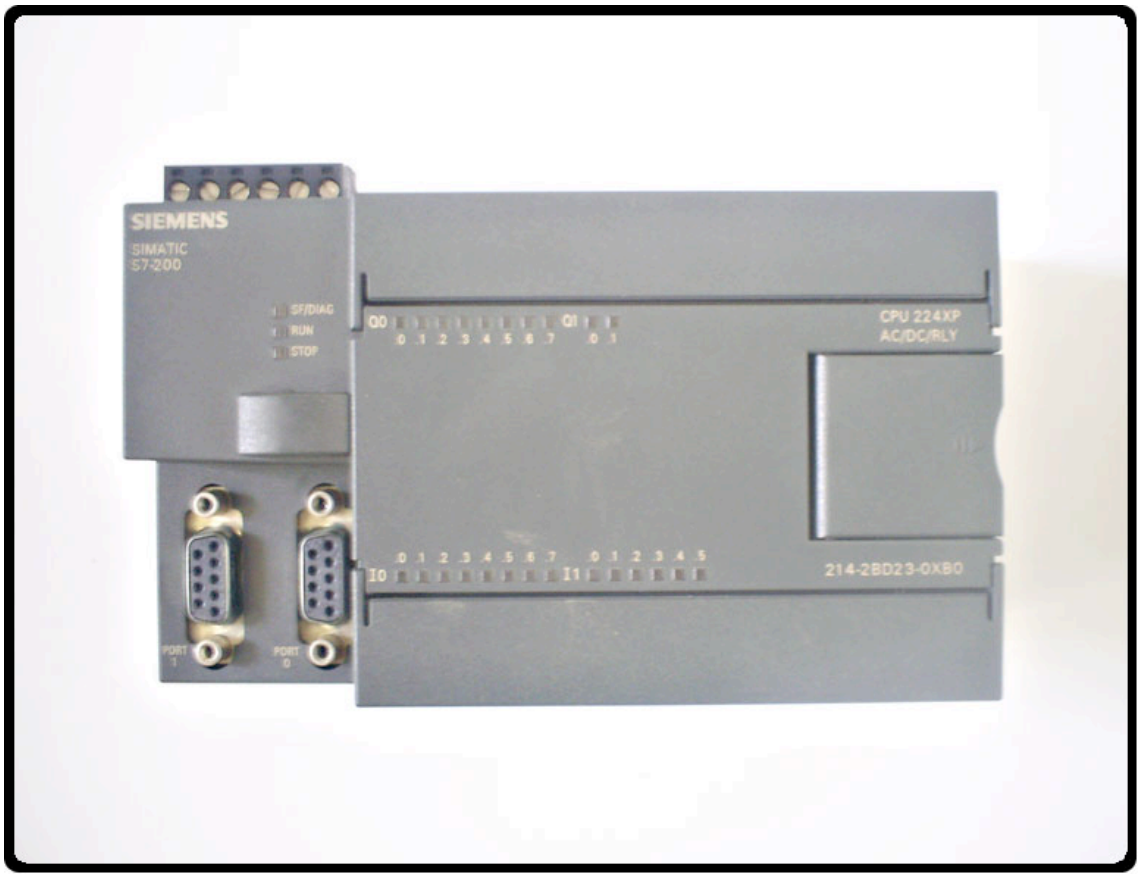
-Los elementos neumáticos de control y seguridad.

-El autómata S7-300 que no se utilizará.



3.2.- Autómata Siemens Simatic-7 CPU224 XP..

Para desarrollar este proyecto se ha utilizado un autómata Siemens s7-224XP, el cual era el que mejor se ajustaba a las necesidades que habíamos establecido. En los siguientes apartados describiremos dicho autómata y sus características.



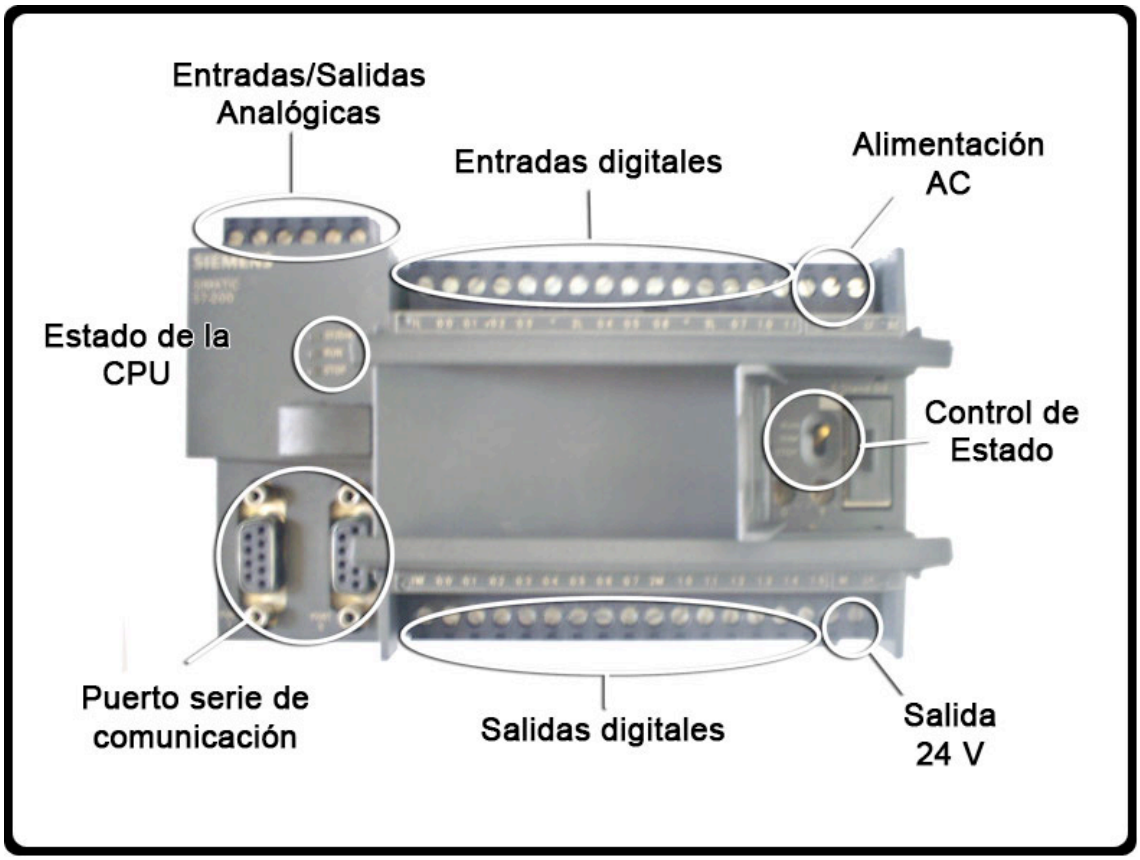
3.2.1.- Características del autómata.

En la siguiente tabla se exponen las características del autómata S7-224XP:

Voltaje de alimentación	120 V AC, 230 V AC
Rango de frecuencias de alimentación	47-63 Hz
Voltaje para las salidas (señales de control)	24 V, con un rango permisible de 5-30 V
Consumo de corriente	20A a 264 V
Tiempo de vida de las baterías	100 Horas
Módulos de memoria	Memoria basada en la tecnología EEPROM. Memoria de Datos: 8 Kilobytes Memoria de Programa: 12 Kilobytes
Tiempo de ciclo de la CPU	0.22µs
Contadores	256 contadores disponibles con un rango que va desde 0 a 32767
Temporizadores	256 temporizadores disponibles con una unidad de tiempo mínima de 1 ms.
Marcas especiales	De la M 0.0 a la M 31.7
Configuración del Hardware	SIMATIC PG/PC estándar PC
Conexión al PC	RS 485 con una velocidad entre 19.2kBit/s a 187.5kBits/s
Lenguajes de programación	LAD, FUP y AWL
Número de entradas digitales • Para la señal “0” • Para la señal “1”	14 0 a 5 V Min. 15 V con 2.5mA
Número de salidas digitales • Voltaje de salida para “1” • Corriente de salida para “1”	10, por relé L+/L1 2 A
Número de entradas analógicas	2; Potenciómetro analógico de 8 bits

3.2.2.- Elementos del autómata S7-224xp.

En la siguiente figura se representan los elementos de comunicación, indicadores y demás partes del autómata S7-224.



Los estados del autómata son tres; RUN, STOP y TERM, se comentan a continuación:

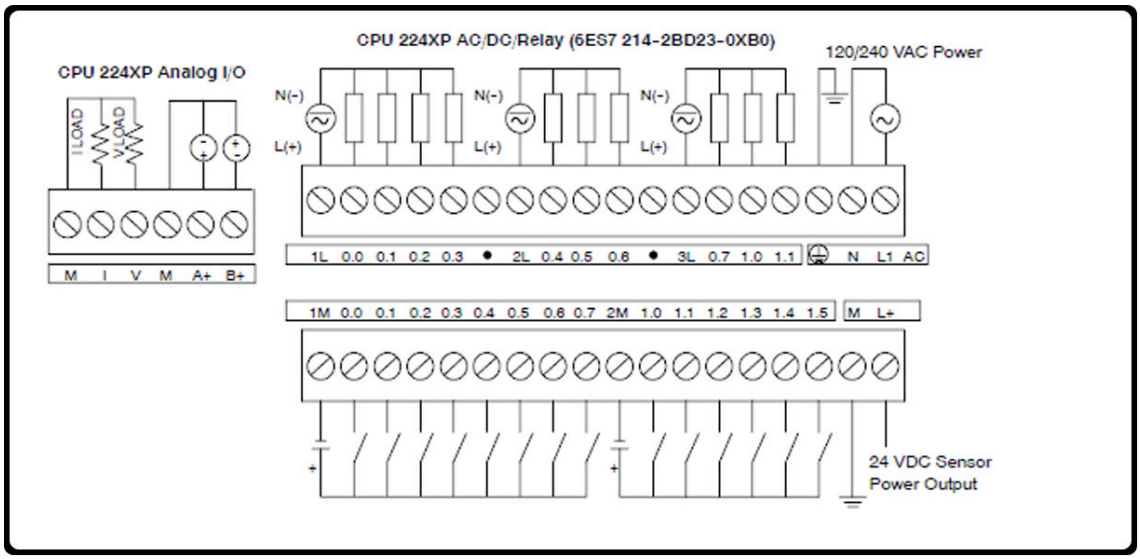
RUN es el estado en el que el autómata ejecuta el programa que se ha guardado en la memoria.

STOP es el estado de parada, es decir, el autómata no ejecuta ningún programa.

TERM (Terminal) es el estado en el cual el autómata es controlado por el ordenador conectado al puerto serie.

3.2.3.- Mapa de conexiones para la alimentación.

En el siguiente esquema se representa la forma correcta de alimentar el autómata S7-224XP:



3.2.4.- Direccionamiento de los módulos de señales.

Para el correcto direccionamiento de los canales de los módulos de señales en el programa de aplicación, es necesario definir los siguientes conceptos:

-Direcciones de los módulos digitales.
La dirección de una entrada o salida de un modulo digital se forma agrupando la dirección de byte y la dirección de bit.

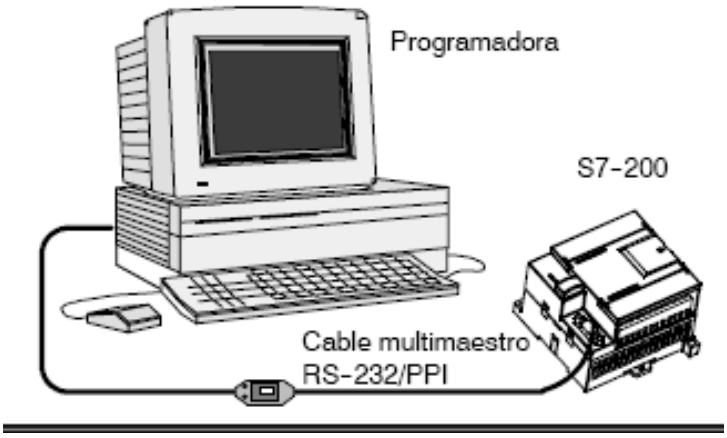
La dirección de byte depende de la dirección inicial del módulo.

La dirección de bit puede leerse en el módulo.

3.2.5.- Interfaz MPI.

La interfaz MPI es el enlace entre el sistema destino (CPU 224XP) y el sistema origen (PC) Para comunicarse a través del programa STEP 7 vía MPI o para la comunicación en una subred MPI.

La velocidad preajustada para la comunicación es de 187,5 Kbaudios.



La CPU es la encargada de enviar por la interfaz MPI de forma automática sus parámetros de bus ajustados. De esta forma cualquier unidad de programación puede comunicarse automáticamente en una subred MPI.

Cada estación de red configurada dentro de una subred se identifica con una dirección MPI única.Para conectar el cable, hay que seguir los siguientes pasos:

1. Una el conector RS—232 (identificado con ”PC”) del cable multimaestro RS--232/PPI al puerto de comunicación de la programadora. (En el presente ejemplo, conectar a COM 1.)

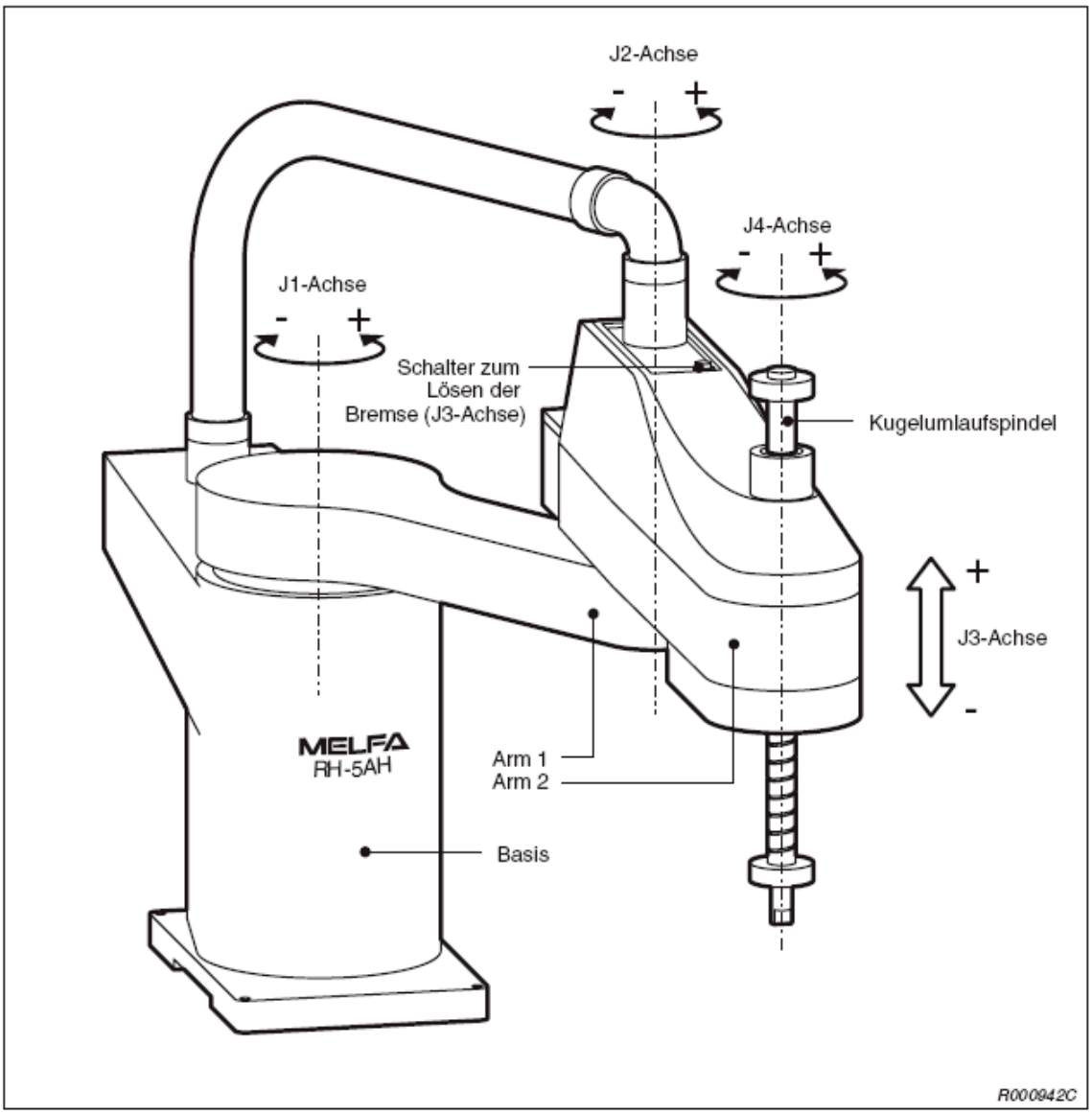
2. Una el conector RS--485 (identificado con ”PPI”) del cable multimaestro RS--232/PPI al puerto 0 ó 1 del S7--200.

3. Vigile que los interruptores DIP del cable multimaestro RS--232/PPI estén configurados como muestra la siguiente figura:

En este proyecto, se utiliza el cable multimaestro RS--232/PPI. El cable multimaestro RS--232/PPI sustituye el cable PC/PPI que se empleaba anteriormente.

3.3.- Robot industrial Mitsubishi.

El robot industrial cedido por la universidad para la realización de este estudio es de la marca Mitsubishi modelo RH-5AH. El RH-5AH es un brazo robot con 4 grados de libertad, gran precisión y velocidad.



3.3.1.- Características del robot.

Longitud máxima del brazo	550mm
Posición de instalación	En el suelo
Grados de libertad	4
Accionado por	Servomotores AC
Métodos de posicionamiento	Encoder Absoluto
Capacidad de los motores	J1=200W J2=100W J3=100W(z) J4=100W(θ axis)
Longitud de los brazos	Para el brazo nº 1 la longitud es 325mm Para el brazo nº 2 la longitud es 225mm La longitud total máxima será 550mm

-Las articulaciones J1, J2, J4 no tienen freno, la articulación J3 si que tiene.

-Rango de operatibilidad. Se define el rango de operatibilidad como el rango de grados o milímetros que una articulación determinada puede girar o desplazarse.

-Para J1= ±127º
-Para J2= ±143º
-Para J3= 200mm
-Para J4= ±360º

-Velocidades de movimiento:

-Para J1 =337,5º/s
-Para J2 =540º/s
-Para J3 =1mm/s
-Para J4 = 1.87º/s

-Velocidad horizontal máxima = 5,360mm/s

-Tiempo de ciclo.

Se define el tiempo de ciclo como el tiempo requerido para realizar un ciclo, el cual es una ejecución sin repetición de un programa de trabajo, (Puede variar según la precisión exigida o la posición de la operación o por el criterio de trabajo).

Para el RH-5AH55 corresponde un tiempo de ciclo de 0,48 segundos.

-Capacidad de carga:

Media = 2kg

Máxima = 5kg

Ambas incluyendo el peso de la pinza.

-Repetibilidad.

- En las direcciones x e y = ±0,002mm
- En la articulación J3 = ±0,01mm
- En la articulación J4 = ±0,03mm

-Temperatura de trabajo.

El rango de temperatura para el correcto funcionamiento del robot RH-5AH55 esta comprendido entre [0ºC a 40ºC]

-Peso del robot.

Igual a 19Kg. aproximadamente

-Especificaciones de protección.

IEC IP54 → Estándar de protección para prevenir cualquier mal funcionamiento ante la caída de agua vertical en el equipo a proteger de 10 litros por minuto a una presión de 80 a 100kPa durante 5 minutos.

3.3.2.- Controlador del robot.

-Tipo CR2A-572.

-CPU de 64 bits RISC y DSP.

-Capacidad de memoria para almacenar 88 programas:

Cada programa 5 pasos.

Cada paso 25 puntos.

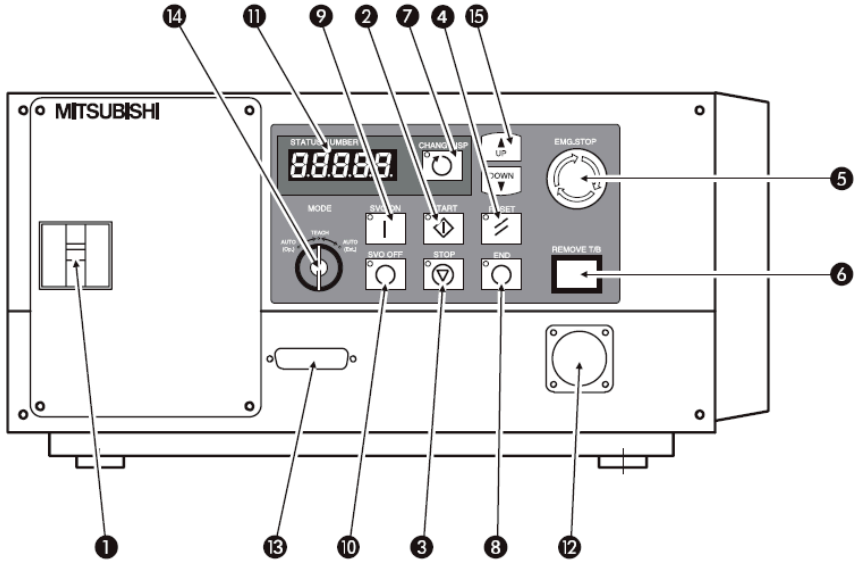
-Interfaces:

- RS-232C → Puerto 1→Para conectar tanto un ordenador portátil como un sensor de visión.
- Ranura para la mano (pinza) →Slot 1→Para una mano neumática.
- Robot input/output→ Canal 1.

-Alimentación de corriente.
→1 fase 180-253V AC.
→Potencia media consumida→0,7 KVA (no incluye picos de corriente al enchufar).

-Dimensiones del controlador:
440x400x200mm

-Peso del controlador → 18kg aproximadamente.

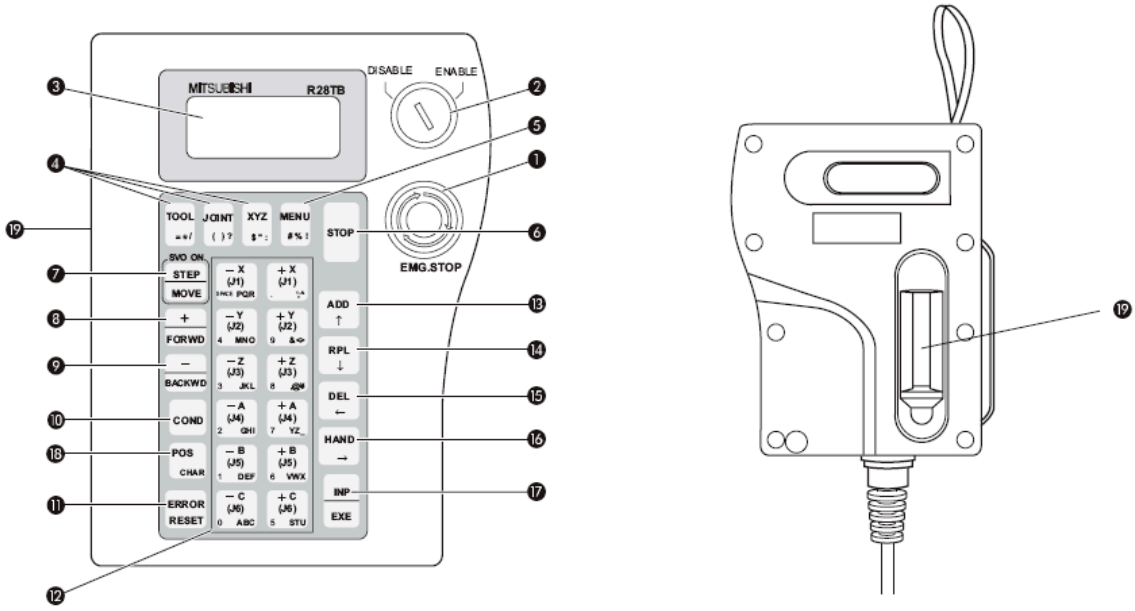


En la siguiente tabla se muestran los elementos de la controladora:

Nº	NOMBRE		FUNCIÓN
1	POWER		Enciende o apaga la controladora.
2	START		Ejecuta el programa. El Led verde se enciende durante una operación.
3	STOP		Detiene el robot inmediatamente.
4	RESET		Resetea el error.
5	EMG.STOP		Detiene el robot cuando existe una emergencia.
6	REMOVE T/B		Conecta o desconecta el T/B.
7	CHANG.DISP		Cambia los detalles mostrados en el display.
8	END		Detiene el programa en ejecución.
9	SVO.ON		Enciende el servo.
10	SVO.OFF		Apaga el servo.
11	STATUS.NUMBER		Muestra el número de programa, error o valor.
12	T/B		Conexión del T/B.
13	RS232 Conexión		Conexión entre la controladora y el PC.
14	MODE	AUTO(Op)	Solo se ejecutan las operaciones desde la controladora.
		TEACH	Solo las operaciones desde el T/B son ejecutadas.
		AUTO(Ext)	Solo las operaciones desde el dispositivo externo son ejecutadas.
15	UP/DOWN		Sube o baja los detalles mostrados en el display.

3.3.3.- Teaching pendant.

El teaching pendant es un dispositivo de entrada usado para programar el robot, es una consola formada por un display, una serie de botones con comandos, el interruptor con llave y el botón de hombre muerto.



1):Paro de emergencia	El servo robot se apaga y la operación para.
2):T/B encendido / apagado	Este botón cambia el estado de la llave T/B.
3):LCD Display Panel	Donde se muestran el estatus y los menús de control.
4):<TOOL,JOINT,XYZ>	Selecciona la forma de mover.
5):<MENU>	Vuelve el display a la pantalla menú.
6):<STOP>	Detiene el programa y frena el robot hasta parar.
7):<STEP/MOVE>	Manteniendo pulsado este botón y apretando otro del panel de movimiento el robot se mueve.
8):<+ FORWD>	Desplazamiento hacia delante.
9):<-BACKWD>	Desplazamiento hacia atrás.
10):<COND>	Este botón fija el programa.
11):<ERROR RESET>	Esto resetea el error y libera el programa.
12):Panel de movimientos	Mueve el robot dependiendo del modo seleccionado.

13):<ADD/ ↑>	Añade una nueva posición al programa y también mueve el cursor arriba.
14):<RPL/↓>	Corrige la posición almacenada y mueve el cursor abajo.
15):<DEL/←>	Borra la posición almacenada y mueve el cursor a la izquierda.
16):HAND/→>	Abre y cierra la mano y mueve el cursor a la derecha.
17):<INP/EXE>	Introduce el programa y ejecuta.
18):<POS CHAR>	Cambia entre números o caracteres.
19):Botón Deadman	Si este botón no esta presionada o esta presionado a tope el servo se apaga y la operación de parada.

-Dimensiones:
95x236x34mm

-Interfaz → RS-422.

-Display → LCD 16 caracteres y 4 líneas.

-Especificaciones de protección → IP65

3.4.- Cámara Web.

La cámara WEB usada en el estudio de la célula de fabricación flexible es una cámara WEB convencional modelo Creative Labs WebCam Go.

Con un sensor CMOS captura imágenes con una resolución VGA de 640x480. La conexión con el PC se realiza a través de un puerto USB.

3.5.- Computador personal.

Para poder realizar este proyecto el ordenador a utilizar, si bien se recomienda que tenga gran velocidad computacional y una gran capacidad de memoria, debe cumplir al menos los siguientes requisitos:

- Puerto serie, para la comunicación con el autómata.
- Puerto serie, para la comunicación con el robot.
- Conexión USB, para conectar la cámara WEB.
- Sistema Operativo XP.

Los programas que se van a utilizar se nombran a continuación:

- MATLAB 7
- STEP 7 Microwin

3.6.- Tarjeta de adquisición de datos.

Para realizar la comunicación entre el computador y el autómata, además del control de la pinza del robot, se usó una tarjeta de adquisición de datos modelo ACL -8112PG la cual se describe y comenta a continuación.

3.6.1.- Características básicas de la tarjeta

- Bus AT (interfaz ISA).
- 16 canales de entrada con masa común o bien 8 diferenciales.
- Ganancia programable de x1, x2, x4, x8 y x16.
- Circuito integrado de muestreo y retención (sample&hold).
- 2 canales de salida analógicos de 12 bits monolíticos multiplexados.
- 16 canales de entrada digital y 16 de salida digital.
- 3 contadores independientes de cuenta atrás de 16 bits.
- Frecuencia de muestreo programable de hasta 100KHz.
- Tres modos de disparo del convertidor A/D: disparo por software, disparo programable mediante temporizador/contador y disparo mediante pulso externo.
- Capacidad para trabajar mediante interrupción: 11 niveles de interrupción (IRQ3 a IRQ15) seleccionables mediante Jumper.
- Conector de 37 pines tipo D.

3.6.2.- Especificaciones detalladas de la tarjeta

- Entrada analógica (A/D):
  - Convertidor A/D: B.B. ADS774 (aproximaciones sucesivas).
  - Canales de entrada: 16 con masa común o bien 8 diferenciales.
  - Resolución: 12 bits.
  - Rango de entrada (controlados por software): Bipolar de ±10V, ±5V, ±2.5±, ±0.625V y ±0.3125V (a menor rango mayor resolución).
  - Tiempo aproximado de conversión: 8μs.
  - Protección de sobretensión: ±35V DC.
  - Precisión: 0.015% de la lectura ±1 bit.
  - Impedancia de entrada: 10MΩ.
  - Modos de transferencia: por software, por interrupción y por DMA.
  - Velocidad de transferencia de datos (frecuencia de muestreo): 100KHZ (máximo).

- Salida analógica (D/A):
  - Canales de salida: 2 salidas analógicas con doble buffer.
  - Resolución: 12 bits.
  - Rango de salida: unipolar de 0 a 5V o 0 a 10V mediante referencia interna y unipolar de +10V o –10V con referencia externa.

- Convertidor D/A: B.B. DAC7541 de multiplexación monolítica.
- Tiempo de conversión: 30μs.
- Linealidad: ±1/2 bit LSB.
- Intensidad máxima: ±5mA.

o Entradas y salidas digitales (DIO):

- Canales: 16 de entrada y 16 de salida compatibles TTL.
- Tensión de entrada: mínimo de 0V y máximo de 0.8V para estado 0 y mínimo de +2.0V para estado 1.
- Carga admisible: +0.5V a –0.2mA máximo para estado 0 y +2.7V a +10mA máximo para estado 1.
- Tensión de salida: mínimo de 0V y máximo de 0.4V para estado 0 y mínimo de +2.4V para estado 1.

o Contador programable:

- Dispositivo: 8254.
- Temporizador para conversión A/D: temporizador de 32 bits (dos contadores de 16 bits en cascada) con base de tiempos de 2MHz.
- Contador: un contador de 16 bits con base de tiempos de 2MHz.
- Frecuencia de salida del temporizador: de 0.00046 Hz a 0.5MHz.

Para realizar la conexión con los dispositivos se utiliza una tarjeta adicional conectada mediante un cable de 37 hilos a la ACL - 8112PG. Dicha tarjeta adicional consiste en una regleta donde se puede acceder fácilmente a cada uno de los canales. La disposición de los distintos canales y señales se muestra en la siguiente tabla:

PIN	DESCRIPCIÓN
1 a 8	Canal 0 a 7 de entrada analógica
9 – 10	Masa de los canales 0 a 7 de entrada analógica
11	Referencia de tensión externa 2
13	Salida de +12 V
14	Masa de los canales de entrada analógica
15	Masa de los canales de salida digital
16	Señal de salida del contador 0
17	Señal de disparo externo
18	No conectada
19	+5V
20 a 27	Canal 8 a 15 de entrada analógica
28 – 29	Masa de los canales 8 a 15 de entrada analógica
30	Canal 0 de salida analógica
31	Referencia de tensión externa 1
32	Canal 1 de salida analógica
33	Entrada de reloj para 8254
34	Entrada para 8254
35 – 36	No conectada
37	Entrada de señal de reloj externo
38	Masa de los canales de salida analógica

4.- Fases de desarrollo del proyecto.

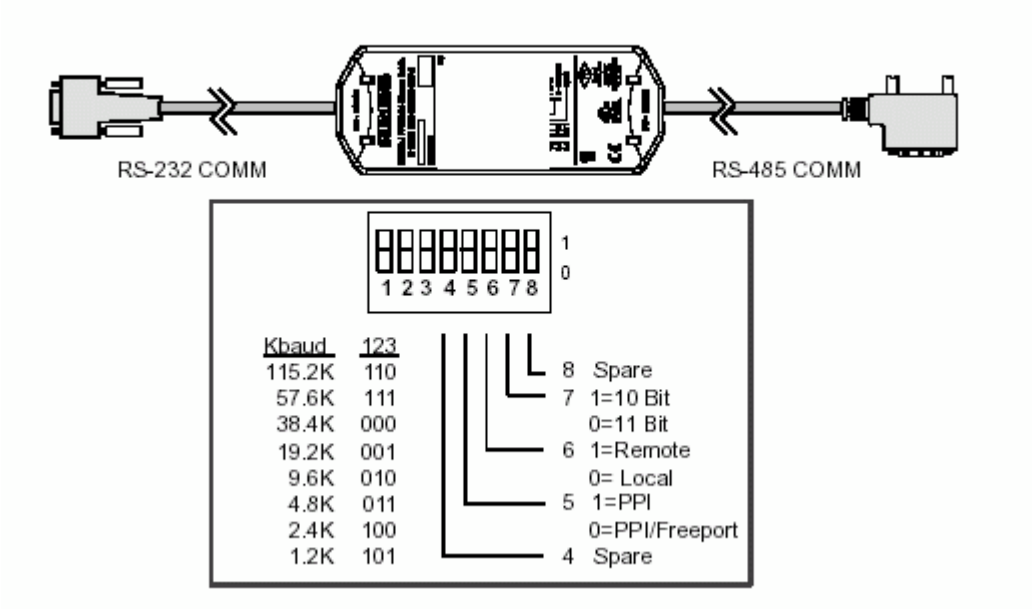
4.1- Programación y conexionado del autómata.

En los siguientes apartados se describe el trabajo realizado para la programación y comunicación del autómata S7-224 con el ordenador. Siendo esta comunicación de tipo maestro/esclavo donde el maestro es el PC y el autómata el esclavo.

4.1.1- Conexionado de comunicación PC/autómata.

Para la comunicación entre el autómata y el PC se utilizará un cable PC/PPI, el cual se deberá configurar de la siguiente forma:

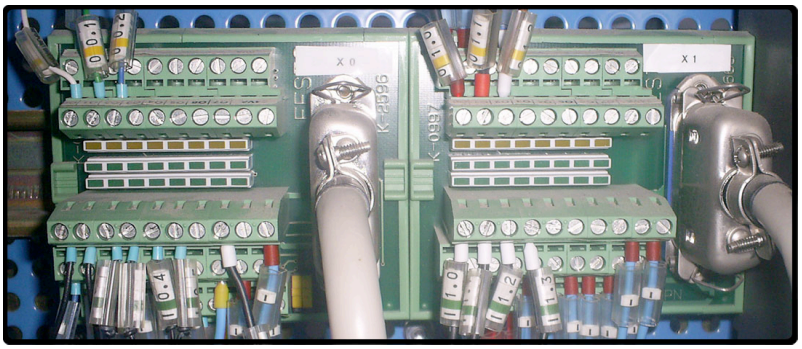
- Coloque el interruptor 5 en la **posición "0"**, para ajustar el **modo PPI Multimaestro y el modo FREEPORT**.
- Configure los interruptores 1-3 en función de la velocidad de transferencia utilizada.
- Se recomienda la configuración de los interruptores 1, 2 y 3 para una velocidad de comunicación de 9.6 K.



Lo siguiente para programar el PLC será; Identificar las entradas y salidas del autómata.

4.1.2.- Conexiones de entrada/salida del autómata.

Las entradas y salidas del autómata se conectaran al panel indicado en la figura, localizado en la cinta transportadora.



Conexiones para las entradas digitales al autómata S7 224xp:

Nombre	S7-224XP	Panel
Marcha	I 1.0	X1//12
IND/INT	I 1.1	X1//13
Reset	I 1.2	X1//14
Manual/Auto	I 1.3	X1//15
Emergencia	I 0.0	X0//9
Detección de la bandeja en la estación 1	I 0.1	X0//2
Detección de la bandeja en la estación 2	I 0.2	X0//3
Detección de la bandeja en la estación 3	I 0.3	X0//4
Detección de la bandeja en la estación 4	I 0.4	X0//5
Detección de la bandeja en la estación 5	I 0.5	X0//6
Detección de la bandeja en la estación 6	I 0.6	X0//7
Entrada del ordenador	I 0.7	Tabla PC

Conexiones para las salidas digitales del autómata S7 224xp:

Nombre	S7-224XP	Panel
Cintas 1 y 2	Q 1.0	X1//32
Cintas 3 y 4	Q 1.1	X1//33
Enclavar	Q 0.0	X0//22
Desenclavar	Q 0.1	X0//23
Topes	Q 0.2	X0//24
Luz de emergencia	Q 0.7	X1//34
Salida al ordenador	Q 0.3	Tabla PC

4.1.3.- Programación del autómata S7-224.

Para programar el sistema de control del autómata primero se marcaran unos objetivos para a partir de ellos desarrollar el Grafcet de control y el posterior programa de control en lenguaje KOP.

4.1.3.1- Objetivos.

El autómata controlará la cinta transportadora cuyo objetivo es el de transportar la pieza o el objeto a la estación deseada para que después el robot pueda manipularla a partir de la información generada por el sistema de visión artificial.

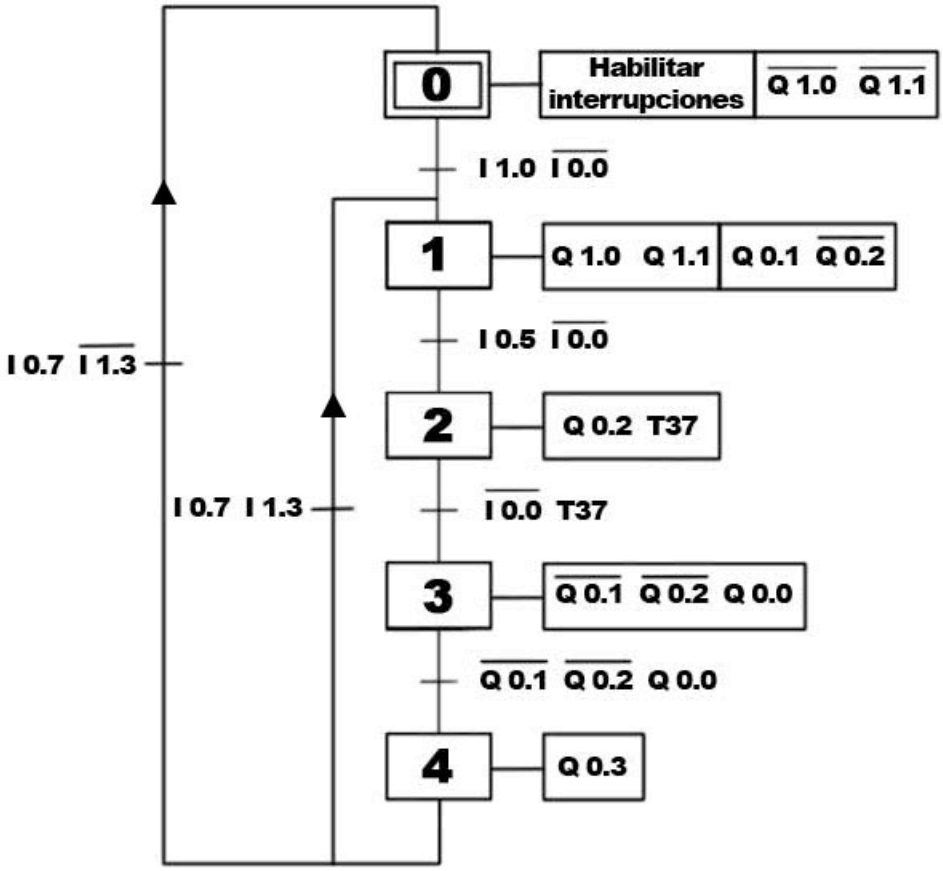
El programa de control del autómata además de controlar la cinta transportadora, establecerá dos tipos de operación, manual y automático.
El programa en modo manual, realizará un ciclo completo y esperará la pulsación de marcha para la ejecución del siguiente ciclo.

En modo automático, el autómata ejecutara ciclos de programa hasta que se pare la célula de fabricación flexible u ocurra una parada de emergencia.

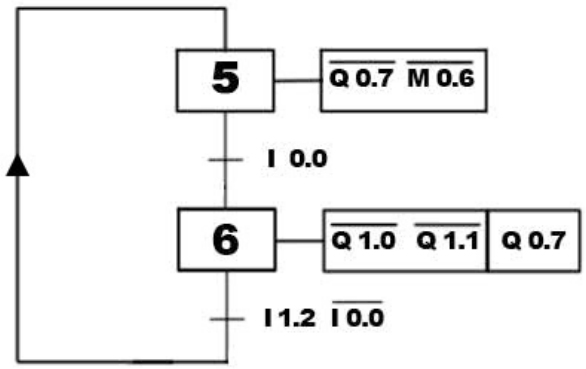
El programa debe contemplar parámetros de seguridad tales como las paradas de emergencia. Esto se conseguirá a través de subrutinas orientadas a la parada total de la cinta de transporte en caso de que se pulsara el botón de parada de emergencia.

4.1.3.2.- Grafcet.

El diagrama Grafcet del programa de control es el que se muestra a continuación.

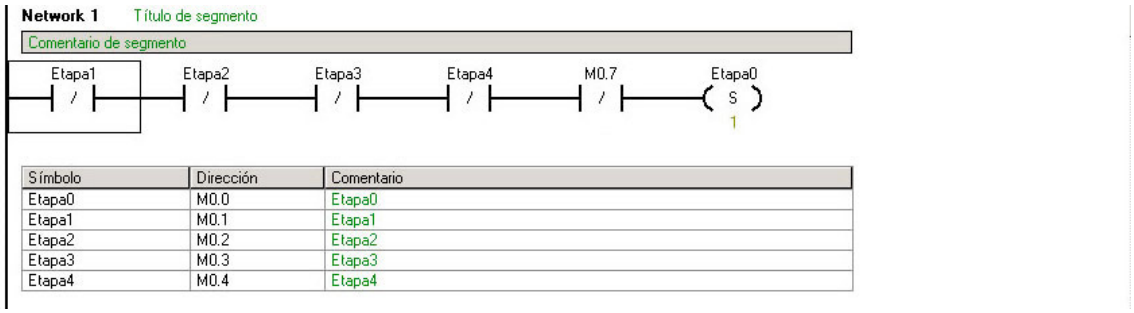


El diagrama de la rutina de interrupción, la cual se activará al detectar un flanco positivo por la entrada I 0.0, es el siguiente:



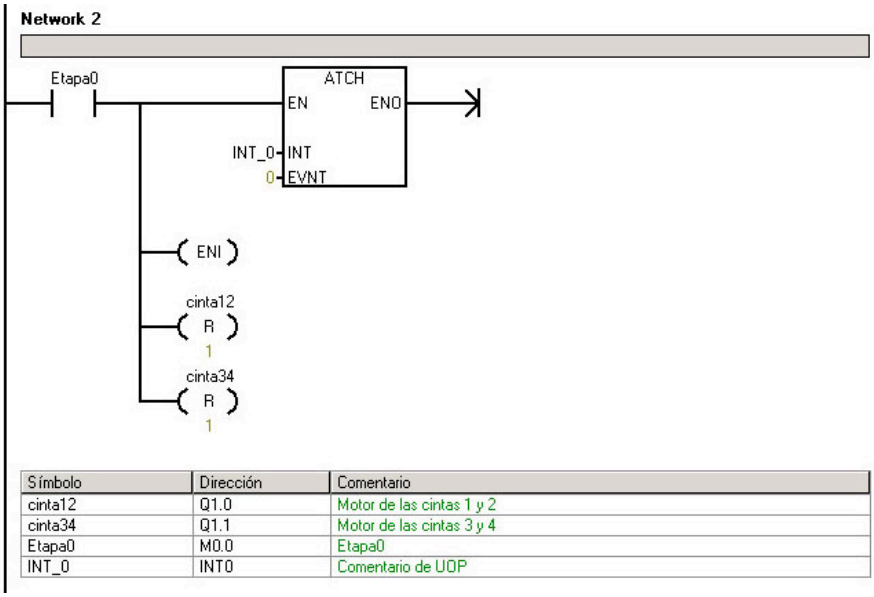
4.1.3.3.- Programa de control.

El programa de control se realizará dentro del programa STEP 7 utilizando el lenguaje de contactos, dicho programa se describe a continuación:



Símbolo	Dirección	Comentario
Etapa0	M0.0	Etapa0
Etapa1	M0.1	Etapa1
Etapa2	M0.2	Etapa2
Etapa3	M0.3	Etapa3
Etapa4	M0.4	Etapa4

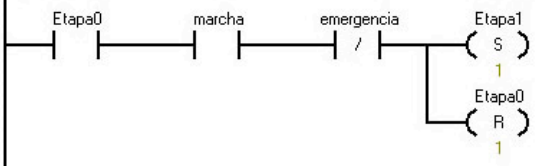
Estando todas las demás etapas y la marca especial 0.7 desactivadas, se activa la etapa0, la cual empieza un ciclo nuevo.



Símbolo	Dirección	Comentario
cinta12	Q1.0	Motor de las cintas 1 y 2
cinta34	Q1.1	Motor de las cintas 3 y 4
Etapa0	M0.0	Etapa0
INT_0	INT0	Comentario de UOP

Al estar activa la etapa 0, se habilitan las interrupciones con el comando (ENI) y además, por medio de la operación ATCH, asociamos el evento de interrupción “0” (REF: El evento de interrupción 0 corresponde con la entrada de un flanco positivo en la entrada I0.0) a una parte del programa indicada por el número de la rutina de interrupción, en este caso el 0. También se asegura que las cintas 1, 2, 3 y 4 estén paradas.

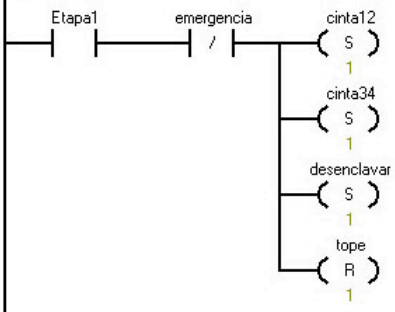
Network 3



Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
Etapa0	M0.0	Etapa0
Etapa1	M0.1	Etapa1
marcha	I1.0	Boton de marcha

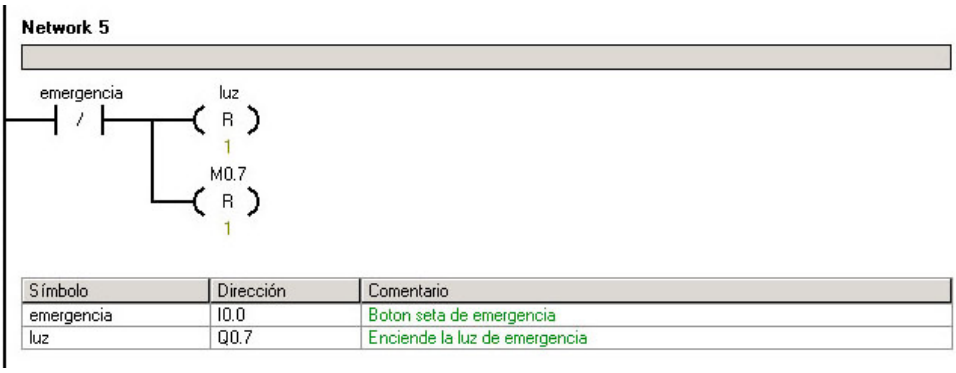
Estando la etapa 0 activada, habiendo pulsado el botón de Marcha y estando el pulsador de emergencia desactivado, se activa la etapa 1 y se desactiva la etapa anterior.

Network 4



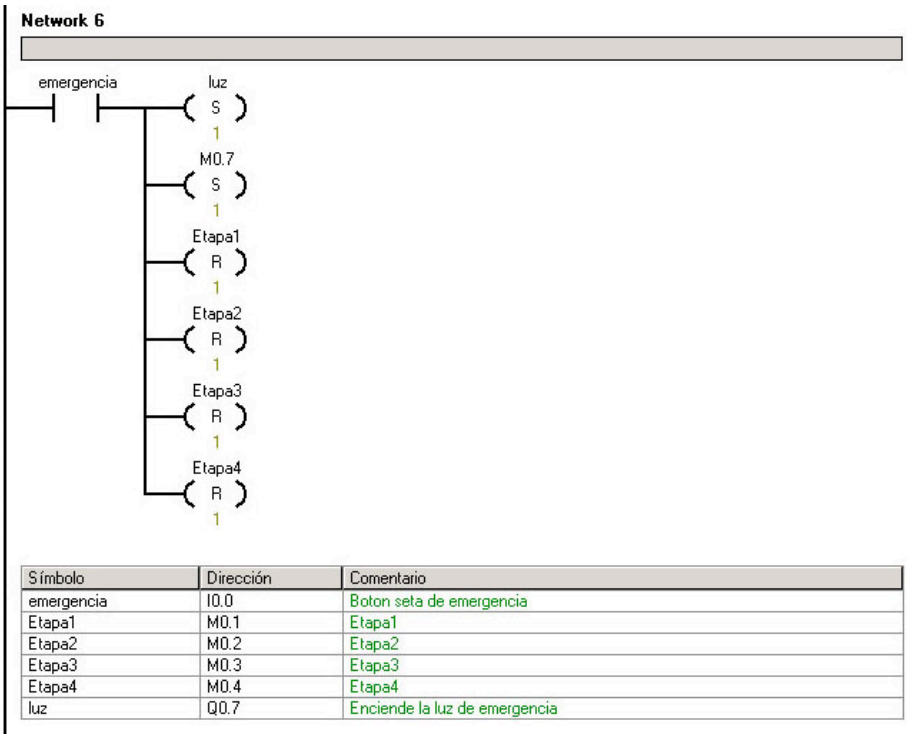
Símbolo	Dirección	Comentario
cinta12	Q1.0	Motor de las cintas 1 y 2
cinta34	Q1.1	Motor de las cintas 3 y 4
desenclavar	Q0.1	Desenclavar los carros
emergencia	I0.0	Boton seta de emergencia
Etapa1	M0.1	Etapa1
tope	Q0.2	Eleva los topes

Al activarse la etapa 1 y estando desactivada la entrada de emergencia, se activan las cintas 1, 2, 3 y 4 se desenclavan las estaciones y se quitan los topes.



Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
luz	Q0.7	Enciende la luz de emergencia

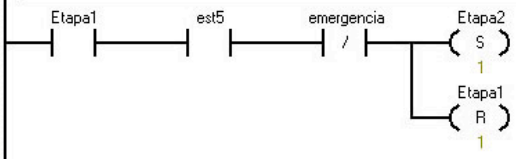
Al desactivar la entrada de emergencia, la luz de emergencia se apaga y también se resetea la marca especial 0.7.



Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
Etapa1	M0.1	Etapa1
Etapa2	M0.2	Etapa2
Etapa3	M0.3	Etapa3
Etapa4	M0.4	Etapa4
luz	Q0.7	Enciende la luz de emergencia

Al activarse la entrada de emergencia, se enciende la luz de emergencia, se activa la marca especial 0.7 y se desactivan todas las etapas.

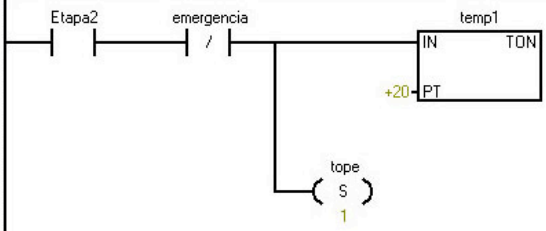
Network 7



Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
est5	I0.5	Deteccion del carro en la estacion 5
Etapa1	M0.1	Etapa1
Etapa2	M0.2	Etapa2

Con la etapa 1 activada, la entrada est5 activada y la entrada de emergencia desactivada, se activa la etapa 2 y se desactiva la uno.

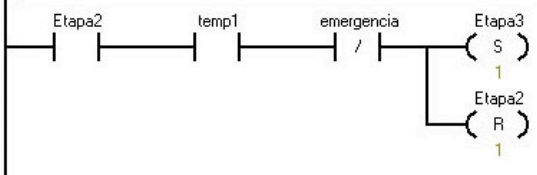
Network 8



Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
Etapa2	M0.2	Etapa2
temp1	T37	Tiempo para asegurar la posicion de la plataforma
tope	Q0.2	Eleva los topes

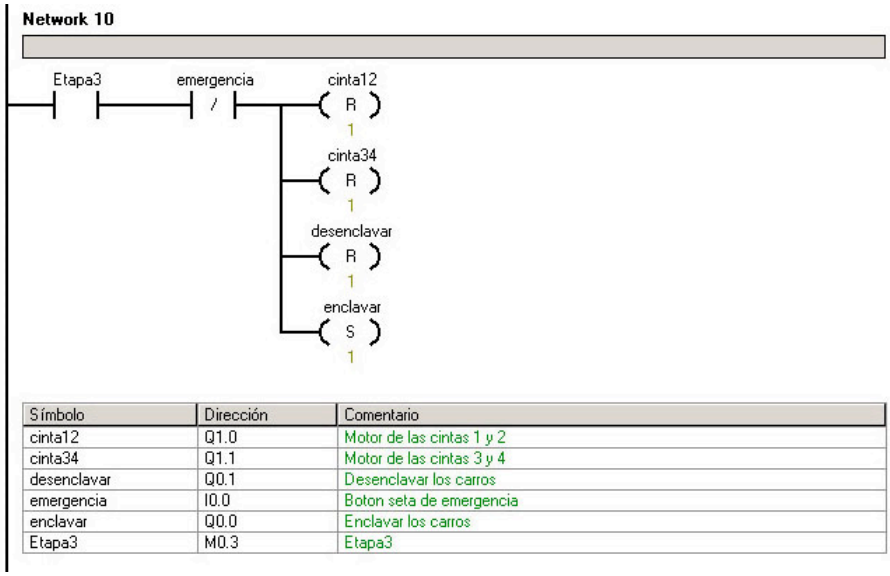
Al activarse la etapa 2 y estar la entrada de emergencia desactivada, se inicializa el temporizador 1 y se suben los topes.

Network 9



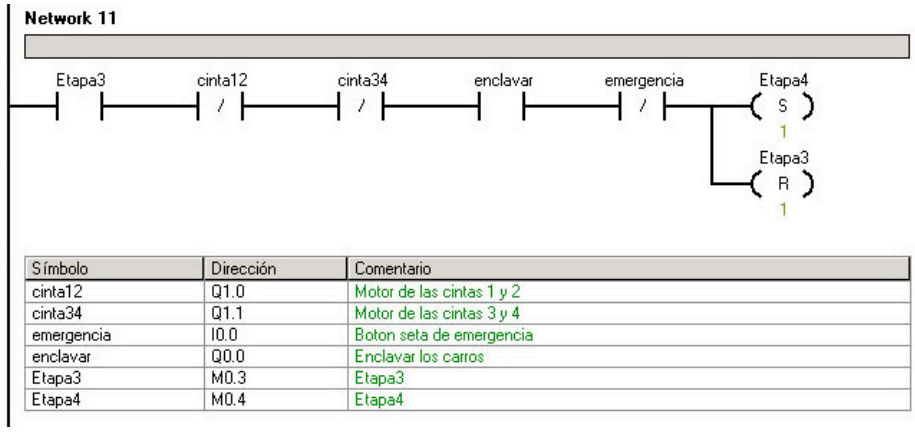
Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
Etapa2	M0.2	Etapa2
Etapa3	M0.3	Etapa3
temp1	T37	Tiempo para asegurar la posicion de la plataforma

Estando activa la etapa 2, la luz de emergencia desactivada y el temporizador indica que a pasado el tiempo programado, se activa la etapa 3 y desactiva la 2.



Símbolo	Dirección	Comentario
cinta12	Q1.0	Motor de las cintas 1 y 2
cinta34	Q1.1	Motor de las cintas 3 y 4
desenclavar	Q0.1	Desenclavar los carros
emergencia	I0.0	Boton seta de emergencia
enclavar	Q0.0	Enclavar los carros
Etapa3	M0.3	Etapa3

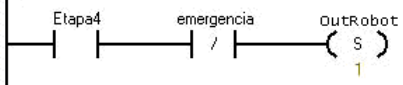
Con la etapa 3 activa y la entrada de emergencia desactivada, se paran las cintas 1, 2, 3 y 4 y se enclavan las estaciones.



Símbolo	Dirección	Comentario
cinta12	Q1.0	Motor de las cintas 1 y 2
cinta34	Q1.1	Motor de las cintas 3 y 4
emergencia	I0.0	Boton seta de emergencia
enclavar	Q0.0	Enclavar los carros
Etapa3	M0.3	Etapa3
Etapa4	M0.4	Etapa4

Cuando se comprueba que la etapa 3 esta activa, las cintas paradas, la entrada de emergencia desactivada y las estaciones están enclavadas, se activa la etapa 4 y desactiva la 3.

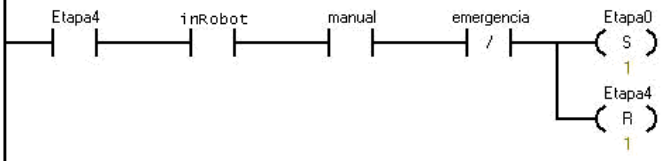
Network 12



Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
Etapa4	M0.4	Etapa4
OutRobot	Q0.3	Tiempo empleado por el robot en realizar la tarea

Al estar activa la etapa 4 y desactivada la entrada de emergencia se activa la señal que indica el comienzo de la tarea del robot.

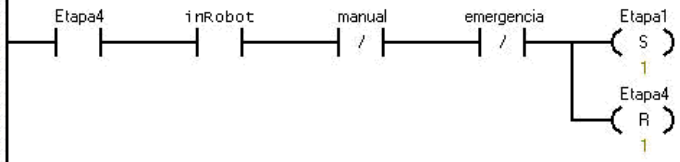
Network 13



Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
Etapa0	M0.0	Etapa0
Etapa4	M0.4	Etapa4
manual	I1.3	Boton seleccion manual automatico
inRobot	I0.7	Fin Tiempo empleado por el robot en realizar la tarea

Cuando se reciba la señal de que el robot ha terminado y la entrada de manual esta activada, se activa la etapa 0 y desactiva la 4.

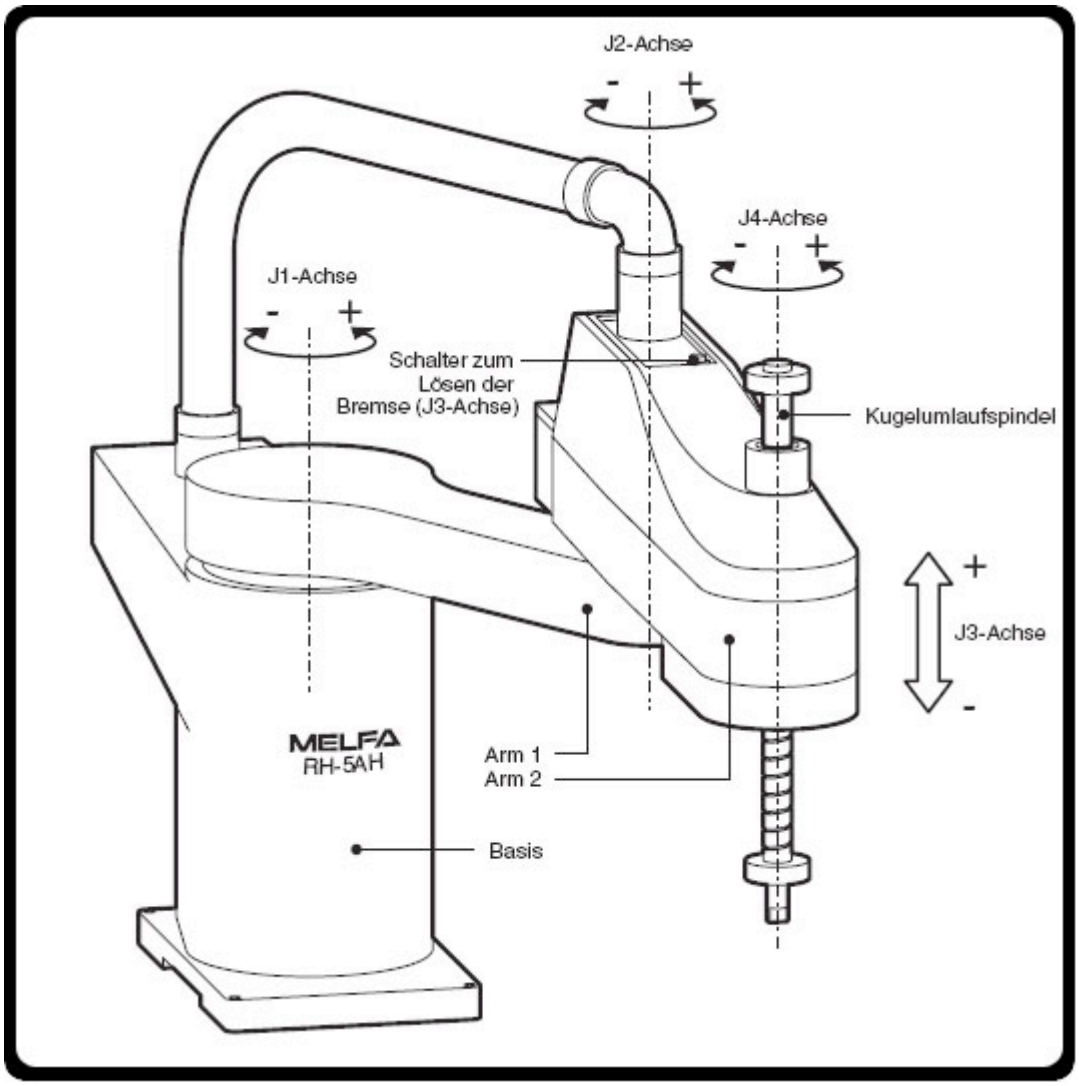
Network 14



Símbolo	Dirección	Comentario
emergencia	I0.0	Boton seta de emergencia
Etapa1	M0.1	Etapa1
Etapa4	M0.4	Etapa4
manual	I1.3	Boton seleccion manual automatico
inRobot	I0.7	Fin Tiempo empleado por el robot en realizar la tarea

Si por el contrario, se activa la señal indicando que el robot terminó su tarea y la entrada en manual esta desactivada (modo Auto) se activa la etapa 1 y desactiva la 4.

4.2.- Programación y puesta en marcha del robot.



Los robots SCARA de Mitsubishi son especialistas en la paletización rápida, en la clasificación y en la instalación de componentes. Disponen de un ciclo breve de hasta 0,5 segundos para un movimiento de elevación vertical de 25mm, de desplazamiento horizontal de 300mm, de descenso de 25mm y de retorno.

4.2.1.- Recambio de las baterías del robot.

Debido a que las baterías del robot estaban en mal estado, el origen de coordenadas y la actual posición de cada una de las articulaciones, eran borrados cada vez que se inicializaba el robot. Por ello se solicitó un recambio de baterías a la empresa Mitsubishi y se sustituyeron las antiguas baterías defectuosas, siguiendo las indicaciones del manual de mantenimiento y puesta a punto del brazo robot:

1. Asegúrese de que el brazo robot y el controlador están conectados con cable.

2. Coloque el botón control power en ON.

3. Presione el botón de emergencia, esto es una medida de seguridad que siempre debe realizarse.

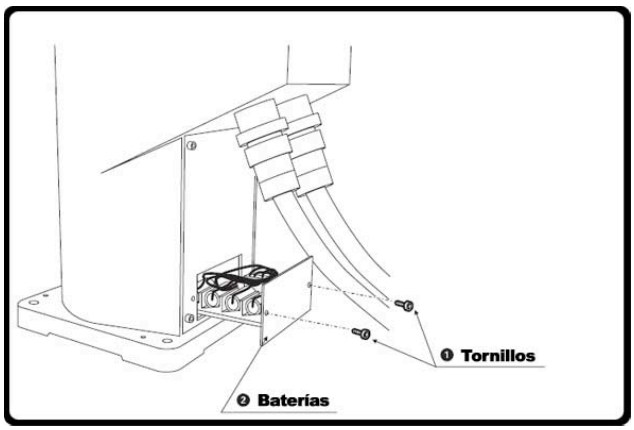
4. Quite los dos tornillos que sujetan la tapa de las baterías y quítela.

5. Retire las viejas baterías desconectándolas del cable.

6. Inserte las nuevas baterías y conecte de nuevo el cable.

7. Lleve a cabo los pasos del 4 al 6 al revés para colocarlo todo.

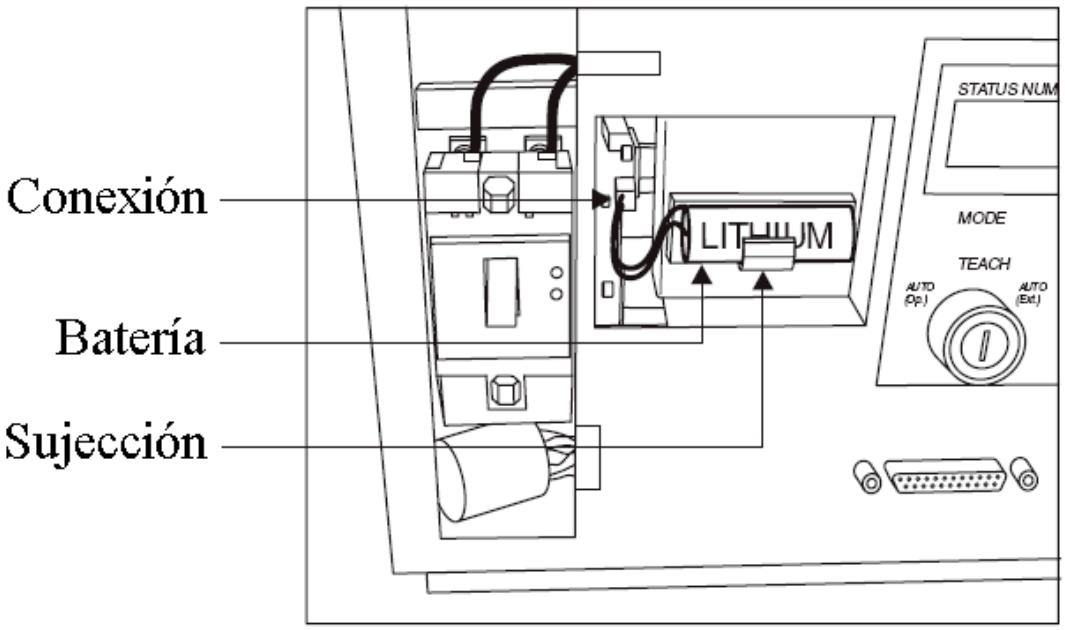
8. Inicialice el tiempo de consumición de las baterías.



Para llevar a cabo la inicialización del tiempo de consumición de las baterías seguiremos las instrucciones del manual de explicaciones detalladas de funciones y operaciones:

Nr.	Display	Teaching pendant
①	<MENU> 1.TEACH 2.RUN 3.FILE 4.MONI 5.MAINT 6.SET	+ C (J6) 5 STU
②	<MAINT> 1.PARAM 2.INIT 3.BRAKE 4.ORIGIN 5.POWER	− A (J4) 2 GHI
③	<INIT> INIT (2) 1.PROGRAM 2.BATT	− A (J4) 2 GHI ⇨ INP EXE
④	<INIT> BATT. OK? (1) 1:EXECUTE	− B (J5) 1 DEF ⇨ INP EXE

1. Presione “5”, seleccionará la pantalla de mantenimiento.
2. Presione “1” y seleccionará la pantalla de iniciación.
3. Presione “2” y después “INP” para seleccionar la pantalla de Inicialización del tiempo de consumición.
4. Cuando “INP” es presionado después de “1” el tiempo de consumición de la batería empezara, después de ejecutarse, aparecerá la pantalla de inicialización.



4.2.2.- Establecimiento el origen del robot.

Para un correcto funcionamiento del brazo robot, era indispensable establecer los puntos de origen del sistema de coordenadas del robot, para ello se siguieron las indicaciones del manual de mantenimiento y puesta a punto del brazo robot, las cuales se describen a continuación:

1º-Usando el T/B:

Nr.	Pantalla	Teaching pendant
①	<MENU> 1.TEACH 2.RUN 3.FILE 4.MONI 5.MAINT 6.SET	+C (J6) 5 STU
②	<MAINT> 1.PARAM 2.INIT 3.BRAKE 4.ORIGIN 5.POWER	−Y (J2) 4 MNO
③	<ORIGIN> 1.DATA 2.MECH 3.TOOL 4.ABS 5.USER	−A (J4) 2 GHI
④	<MECH> SERVO OFF OK? (1) 1:EXECUTE	−B (J5) 1 DEF ⇨ INP / EXE

1. Presione “5” en la pantalla menú y aparecerá la pantalla de mantenimiento.
2. Presione “4” en la pantalla de mantenimiento, accederá a la pantalla de selección.
3. Presione “2” y después presione “1” más la “INP” para parar el robot.

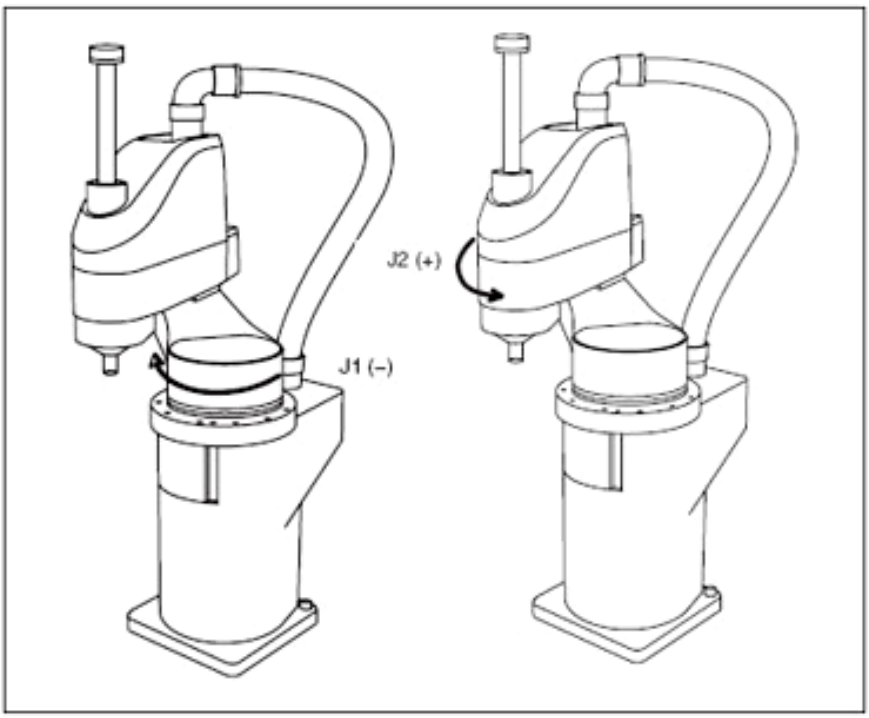
2º-Establecimiento del origen de los ejes J1 y J2:

Nr.	Pantalla	Teaching pendant
①	<MECH> 12345678 BRAKE (00000000) AXIS (00000000) ORIGIN: NOT DEF	
②	<MECH> 12345678 BRAKE (00000000) AXIS (00000000) ORIGIN: NOT DEF	RPL ↓

1. Con ambas manos, mueva despacio el eje J1 en sentido de las agujas del reloj hasta tocar el tope.
2. Presione ”↓” . El cursor se moverá a “SET AXIS”.

③	<MECH> 12345678 BRAKE (10000000) AXIS (10000000) ORIGIN: NOT DEF	−B (J5) 1 DEF → INP EXE
④	<MECH> SET ORIGIN OK? (1) 1:EXECUTE	−B (J5) 1 DEF → INP EXE
⑤	<MECH> 12345678 BRAKE (00000000) AXIS (10000000) ORIGIN: COMPLETED	

3. Introduzca los ejes que van a ser ajustados. Seleccione “1” para el eje J1 y “0” para los demás. El eje “J1” es el indicado por el primer digito por la izquierda, seguido de “J2”, “J3” y “J4”.
4. Presione las teclas “1” y “INP”, la postura de origen será almacenada.
5. El origen de “J1” ha sido establecido.



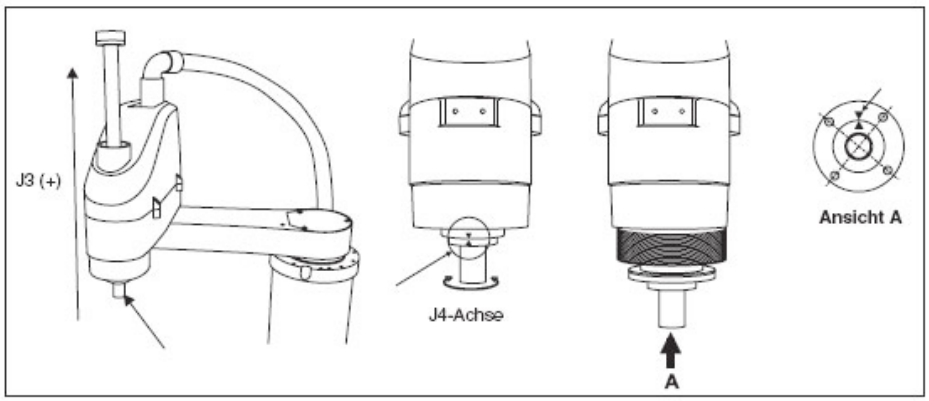
6. Con ambas manos, mueva muy despacio el eje “J2”, en sentido contrario a las agujas del reloj hasta tocar el tope.
7. Repita el proceso para establecer el origen de “J1” pero en este caso seleccionando el eje “J2”, introduciendo un “1” en la segunda posición empezando por la izquierda.
8. El eje “J2” ha sido establecido. Los siguientes ejes, “J3” y ”J4”, deben ser configurados a la vez.

3º-Establecimiento del origen de los ejes J3 y J4:

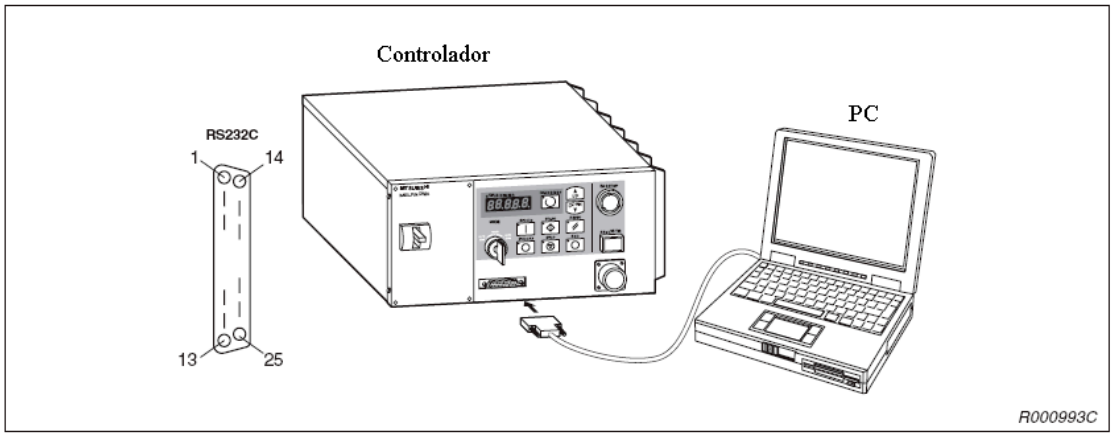
Nr.	Pantalla	Teaching pendant
①	<MECH> 12345678 BRAKE (00100000) AXIS (00000000) ORIGIN: NOT DEF	2 x [−C (J6) 0 ABC] ⇨ [−B (J5) 1 DEF]
②		Totmannschalter + [STEP MOVE] + [+X (J1)]
③	<MECH> 12345678 BRAKE (00100000) AXIS (00000000) ORIGIN: NOT DEF	[RPL ↓]
④	<MECH> 12345678 BRAKE (00100000) AXIS (00110000) ORIGIN: NOT DEF	2 x [−B (J5) 1 DEF] ⇨ [INP EXE]
⑤	<MECH> SET ORIGIN OK? (1) 1:EXECUTE	[−B (J5) 1 DEF] ⇨ [INP EXE]
⑥	<MECH> 12345678 BRAKE (00100000) AXIS (00110000) ORIGIN: COMPLETED	

1. Sitúese en la fila de la selección de frenos “BRAKE”, introduzca dos ceros y un uno, de manera que este seleccionada la tercera posición empezando por la izquierda.
2. Mientras se aprieta el pulsador de hombre muerto, presione al mismo tiempo “MOVE” + ”+X”. El freno se soltara mientras se mantienen pulsadas las teclas.
3. Sitúe el cursor en la fila “axis”.
4. Introduzca “1” en las posiciones de J3 y J4, la tercera y cuarta empezando por la izquierda.
5. Introduzca “1” y presione la tecla “INP”.
6. Se completo el establecimiento del origen.

Posiciones de origen de los ejes J3 y J4:



4.2.3.- Comunicación PC/robot.



Para poder establecer la comunicación entre el brazo robot y el PC se usó el puerto serie del ordenador el cual, usando el programa MATLAB, se configuró sus parámetros para la comunicación con el brazo robot, se introdujo el siguiente código dentro de un fichero .m:

****

```
% Crea un objeto puerto serie y lo configuramos según los parámetros del
robot
% Usaremos el Puerto serie número 1
s1=serial ('COM1');
% Especificamos la velocidad de transmisión
s1.BaudRate = 9600;
% Especificamos el número de bits de datos a transmitir
s1.DataBits = 8;
% Configuramos la paridad de comprobación
s1.Parity = 'even';
% Especificamos el carácter de finalización
s1.Terminator = 'CR';
% Especificamos el número de bits para terminar un byte
s1.StopBits = 2;

% Abrimos el puerto serie que acabamos de crear
fopen (s1);
% Comprobamos que la comunicación entre el robot es correcta, pidiéndole
al dispositivo, en este caso el robot, que se identifique
fprintf(s,'*IDN?');
```

```
% Leemos los datos de identificación del dispositivo
iden = fscanf(s);

% Con la siguiente función podremos cerrar la conexión al finalizar
fclose(s1);
% Para finalizar borraremos el puerto serie creado para liberar memoria
Delete(s);
```

****

En la siguiente tabla se comentan los pines del puerto RS232 y su uso:

Nº PIN	SIMBOLO	Nº PIN	SIMBOLO
1	FG	14	No usado
2	SD (TXD)	15	No usado
3	RD (RXD)	16	No usado
4	RS (RTS)	17	No usado
5	CS (CTS)	18	No usado
6	DR (DSR)	19	No usado
7	SG	20	ER (DTR)
8	No usado	21	No usado
9	No usado	22	No usado
10	No usado	23	No usado
11	No usado	24	No usado
12	No usado	25	No usado
13	No usado		

SÍMBOLO	TIPO	FUNCIÓN
FG		Masa a tierra
SD (TXD)	Salida	Envio de datos desde la controladora al PC
RD (RXD)	Entrada	Recibo de datos desde el PC a la controladora
RS (RTS)	Salida	Recibo de envio al PC
CS (CTS)	Entrada	Fín de palabra del PC
DR (DSR)	Entrada	Señal de preparado del PC
SG		Señal de masa
ER (DTR)	Salida	Señal de preparado de la controladora

4.2.4.- Programación del robot.

El lenguaje de programación en que trabaja el robot es el MELFA BASIC, el cual se describe en el apartado (referencia).

El programa de control que se programo en el brazo robot es el que se comenta a continuación:

****

```
% Definimos la velocidad de transmisión
20 CBAUE11 = 9600
% Definimos el bit de paridad en impar
30 CPRTYE11 = 2
% Definimos el carácter de finalización, en este caso, 0 para CR
40 CSTOPE11 = 0
% Abrimos la comunicación mediante el puerto RS-232
50 OPEN “COM1” AS #1
% Nos desplazamos a la posición P1, almacenada en memoria, donde la
cámara queda por encima de la bandeja
60 MOV P1
%Enviamos el mensaje para que el programa de control capture la imagen
mediante la cámara Web
70 PRINT #1, “PosCamara”
% Obtenemos del canal de comunicación el valor de M1
80 INPUT #1, M1
% Obtenemos del canal de comunicación el valor de M2
90 INPUT #1, M2
% Obtenemos del canal de comunicación el valor de M3
100 INPUT #1, M3
% Una vez obtenidos los parámetros M1, M2 y M3, actualizamos la nueva
posición
110 P_01 = P1
120 P_01.X = P_01.X + M1
130 P_01.Y = P_01.Y + M2
140 P2.A = M3
% Una vez obtenida la posición de agarre, desplazaremos el robot primero
a la posición P_01 y después a P2 que contiene la altura y el ángulo
150 MOV P_01
160 MOV P2
% Enviamos la señal al ordenador de que el robot esta correctamente
posicionado
```

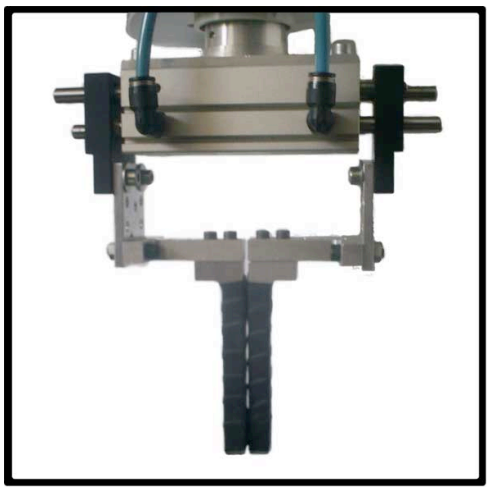
```
170 PRINT #1, "PosAga"
% Esperamos a que el programa de control cierre la pinza
180 INPUT #1, C1$
% Una vez la pinza se haya cerrado, moveremos el robot a la posición
donde posar la pieza, almacenada en P3
190 MOV P3
% Por último descenderemos la pinza hasta la altura para soltar el objeto
200 MOV P4
%Enviamos por el puerto serie el mensaje para soltar la pieza
210 PRINT #1, "PosSu"
%Esperamos a que termine
220 INPUT #1, C2$
% Desplazamos el robot hasta la postura de reposo, almacenada en P3
230 MOV P4
```

****

4.2.5.- Colocación de la pinza en el robot.

La pinza usada para la manipulación de las piezas estaba formada por dos dedos recubiertos de material antideslizante, era impulsada por aire comprimido y era accionada por un relé controlado por el robot.

La pinza se sujetaba al brazo robot por medio de cuatro tornillos Allen.



De la abertura o cierre de la pinza se encargará el ordenador personal a través de la tarjeta de adquisición de datos ACL – 8112PG.

4.3.- Puesta en marcha de la cámara Web.

La cámara Web se colocó encima de las articulaciones tres y cuatro, para poder obtener una imagen bien definida del contenido de la bandeja sin obstaculizar el libre movimiento del robot.

4.3.1.- Descripción de la conexión cámara – PC.

Para la puesta en marcha de la cámara Web se colocó en uno de los puertos USB del ordenador, se instalaron los drivers que controlan la cámara Web y se comprobó que funcionara correctamente.
Los drivers de la cámara Web corresponden al modelo de "Cámara Web Go Plus" y se pueden descargar de la página oficial de creative, siguiendo el siguiente enlace: http://es.europe.creative.com/

Para realizar la conexión con la cámara web utilizaremos el "Image Acquisition Toolbox" de Matlab cuyas características también pueden accederse a través de la interfaz gráfica de usuario imaqtool.

Las instrucciones en Matlab para abrir un canal de video y capturar imágenes se comentan a continuación:

****

```
% Esto crea el canal de video a la cámara web
canalVideo=videoinput('winvideo');
% Abrimos una ventana de previsualización para ver a donde apuntamos
con la cámara
preview(canalVideo);
% Inicializamos el canal de Video
start(canalVideo);
% Tomamos una instantánea con la cámara que se guarda en imgAdq
imgAdq=getsnapshot(canalVideo);
% Mostramos la imagen capturada
imshow(imgAdq);
% Cerramos la ventana de previsualización
closepreview(canalVideo);
% Borramos el canal de video
delete(canalVideo);
```

****

4.4.- Diseño del sistema de visión artificial.

4.4.1.- Introducción.

La visión artificial se podría describir como la deducción automática de las propiedades físicas de un objeto o escena real tridimensional, a partir de una o varias imágenes bidimensionales de dicho objeto o escena. Dichas propiedades físicas las podríamos agrupar en dos categorías:

-Propiedades geométricas; Tales como forma tamaño y localización.
-Propiedades materiales; Como podrían ser el color, la textura y composición.

Para la obtención de dichas propiedades físicas un sistema de visión artificial parte de una imagen obtenida mediante un elemento de adquisición, la cual contiene la información de la escena que se quiere analizar en un lenguaje que el ordenador puede interpretar. Haciendo uso de un programa informático formado por algoritmos, procederemos a extraer de dicha imagen las características o propiedades físicas que nos interesen.
La visión artificial cubre un amplio aspecto de aplicaciones, de las cuales nombraremos las más importantes, como podrían ser: la medición o calibración, la detección de defectos dentro de un ciclo productivo, el reconocimiento de objetos y la localización e identificación de objetos dentro de una escena real.

4.4.2.- La visión artificial en robótica.

Gracias a la visión artificial, los robots industriales pueden presentar características tales como: adaptabilidad, flexibilidad y capacidad de reorganización. Dichas características permiten que un sistema de manipulación automática merezca el calificativo de “inteligente”.

La visión artificial proporciona la descripción de la escena, así como su evolución en el tiempo, información que el sistema de control del robot utiliza en la generación y modificación de sus planes de trabajo, en el monitoreo de la ejecución de tareas y en la detección de errores e imprevistos. El empleo de sistemas de VA en la manipulación controlada sensorialmente permite, por un lado resolver problemas de conocimiento a priori del ambiente, precisión, costo y fiabilidad. Por otro lado, permite a los robots industriales evolucionar en ambientes variables.

4.4.3.- Formación y representación de la imagen.

La formación y representación de la imagen es el conjunto de operaciones que se realizan para transformar la iluminación de una escena en una señal digital que pueda ser interpretada por un computador. Para realizar esta tarea serán necesarios los siguientes elementos:

- Iluminación.
- Cámara.
- Digitalizador.

Como el objetivo de este sistema de visión artificial es el de obtener puntos de agarre óptimos a partir del perímetro del objeto, nos convendría una retroiluminación, es decir, la iluminación del objeto por detrás, situando el objeto entre la pantalla y los focos de iluminación o entre la cámara y la pantalla de fondo, de manera que lo que se ilumina es el fondo de la escena.

Pero debido a inconvenientes físicos y de diseño del material empleado, es imposible esa elección ya que las bandejas utilizadas para el transporte de los objetos son de aluminio lo cual imposibilita este tipo de iluminación.

Debido a esto, el tipo de iluminación a aplicar será el formado por un foco de luz difusa posicionado encima de la escena, con el fin de ofrecer una iluminación frontal. Esta elección se justifica debido a que la iluminación frontal y perpendicular a la escena minimiza el serio inconveniente de las sombras proyectadas por el objeto.

Por el contrario este tipo de iluminación presenta otro gran inconveniente, el inconveniente de los reflejos, los objetos brillantes producen brillos que también perjudica la detección de contornos y detalles del objeto. Para minimizar este inconveniente se usará un foco de iluminación difusa, la cual es la óptima para este caso.

En nuestro caso, la cámara web integra la cámara y el digitalizador ya que da como salida una señal que es capaz de interpretar un computador con los debidos drivers instalados. La imagen obtenida por la cámara web estará formada por una matriz de números, cuyos valores y posición, serán proporcionales al valor de intensidad lumínica que a obtenido el píxel de la cámara en la correspondiente posición.

4.4.4.- Tratamiento de la imagen.

A continuación se describe el preproceso de la imagen que tiene como finalidad la optimización de la imagen para la posterior extracción de la información que contiene la imagen original. Para ello se realizaran las siguientes tareas:

-Ajuste de la intensidad.
-Eliminación del posible ruido.

A partir de este momento, todas las imágenes a tratar serán imágenes en escala de grises, debido a que en nuestro caso, las diferentes tonalidades de color no nos aportan ninguna información útil para nuestra tarea.

Dicha tarea computacional se realizará en el programa MATLAB 7 utilizando la "Image Processing Toolbox" el cual es un paquete de funciones que nos facilitara el trabajo a la hora de trabajar con imágenes.

Algunas de las funciones del "Image Processing Toolbox" utilizadas en este proyecto se describen a continuación:

IMREAD.- Lee una imagen desde un archivo de gráficos.

>> A=IMREAD('nombredearchivo.extensión') lee una imagen desde el archivo especificado en nombredearchivo.extensión y la almacena en la matriz A. Soporta los formatos más comunes como pueden ser jpg, bmp, gif, tiff, png, etc. etc.

RGB2GRAY.- Transforma una imagen de color en una de escala de grises.

>>C=RGB2GRAY(A) Transforma una imagen en color RGB almacenada en la matriz A en una imagen en escala de grises, eliminando la información de brillo y saturación y respetando la iluminancia.

IMSHOW.- Muestra la imagen por pantalla.

>>IMSHOW(I ,N) Nos muestra la imagen de intensidad I con N niveles de gris. Si omitimos el parámetro N, IMSHOW usara un valor de 256 niveles de gris para representar la imagen.

4.4.4.1.- Ajuste de la intensidad.

Antes de explicar los pasos seguidos para ajustar la intensidad de una imagen es necesario definir previamente el concepto de histograma de una imagen.

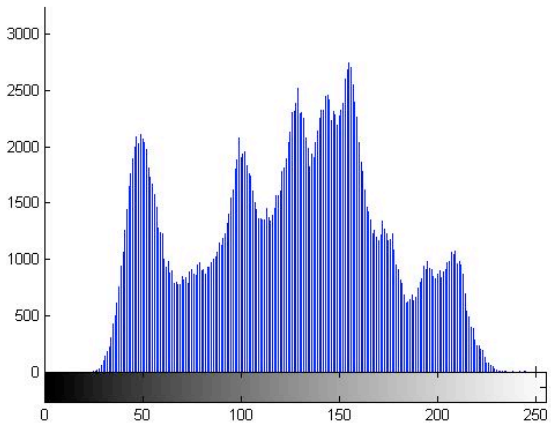
Histograma de una imagen.

El histograma de una imagen es un gráfico que muestra la distribución de intensidades de una imagen en escala de grises. Supongamos que tenemos una imagen en escala de grises, para realizar el histograma de dicha imagen, colocaríamos en el eje de abcisas la escala de niveles de grises de dicha imagen (normalmente de 0 a 255) y en el eje de ordenadas el número de pixeles, que aparecen en la imagen, correspondientes a cada nivel de gris.

Dentro de MATLAB 7, encontramos una función perteneciente al "Image Processing Toolbox" llamada imhist, la cual calcula y nos muestra el histograma de una imagen.

>> IMHIST(A, N) La función IMHIST tiene como entrada una imagen en escala de grises almacenada en la matriz 'A' y un número 'N' el cual especifica la división del eje de ordenadas, el cual se puede omitir. Veamos un ejemplo:

```
>> A=imread('lena.bmp')
>>C=rgb2gray(A)
>>imshow(C)
>>figure, imhist(C)
```

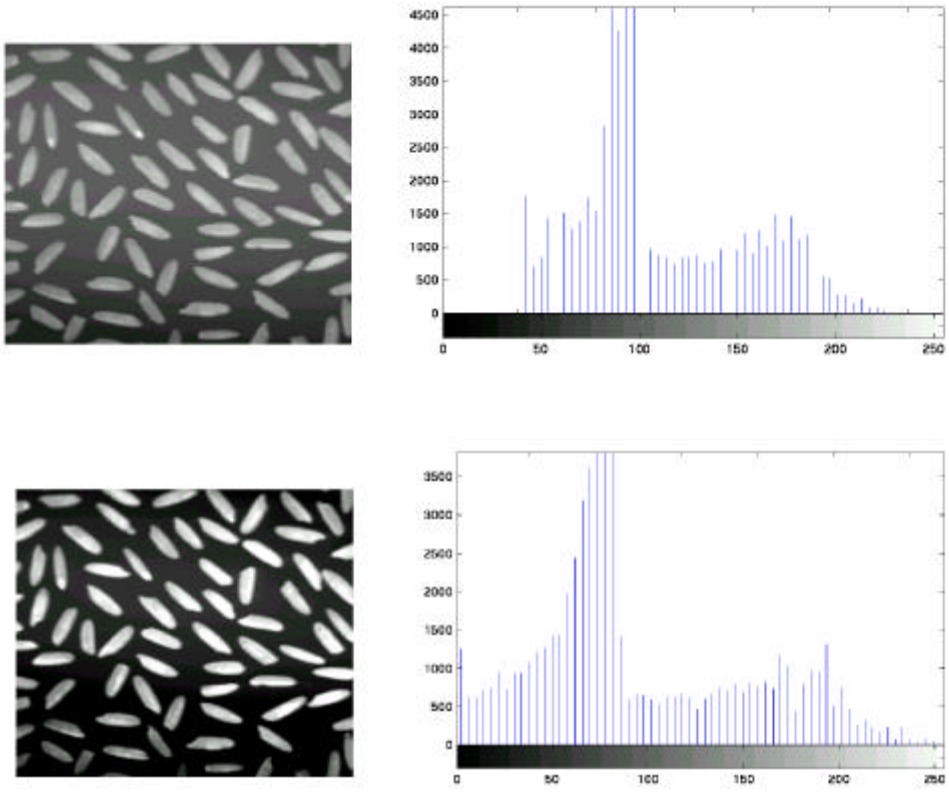


Una vez explicado esto, seguiremos con el ajuste de intensidad.

El ajuste de intensidad es una técnica para mapear los valores de intensidad en un nuevo rango, con esto se consigue mejorar el contraste de la imagen original. Para realizar esto el "Image Processing Toolbox" posee dos funciones que se explican a continuación:

IMADJUST.- Ajusta los valores de intensidad de una imagen.

>> J=IMADJUST(I, [LOW_IN; HIGH_IN], [LOW_OUT; HIGH_OUT]) Mapea los valores de intensidad de la imagen I a unos nuevos valores en J, de tal forma que los valores entre LOW_IN y HIGH_IN se transforman en nuevos valores entre LOW_OUT y HIGH_OUT.



En la imagen de arriba podemos ver como el histograma esta concentrado dentro de un rango que no utiliza todos los tonos de gris posibles. En la imagen de abajo podemos observar como el histograma ha sido ajustado de forma que la misma imagen hace uso de todos los niveles de gris posibles.

HISTEQ.- Ecualización de un histograma.

>> J=HISTEQ(I, HGRAM) Transforma la intensidad de la imagen I de forma que el histograma de la imagen resultante J se aproxima a los valores del histograma almacenado en HGRAM.
Si omitimos este dato, histeq automáticamente escala HGRAM de forma que en la imagen resultante se obtendrá un histograma plano, lo cual da buenos resultados.

En la imagen de la derecha podemos ver el resultado de aplicar histeq a una imagen de escala de grises sin pasarle ningún parámetro en HGRAM, se observa que el contraste de la imagen mejora notablemente.

4.4.4.2.- Eliminación de ruido de la imagen.

Algunas imágenes pueden tener ruido, el cual puede ser introducido de muchas formas, dependiendo de cómo sea creada la imagen.

El "Image Processing Toolbox" posee una serie de herramientas para poder eliminar ruido de una imagen. Métodos distintos producen resultados mejores o peores según sea el tipo de ruido.

Algunos métodos son:

-Filtrado lineal.
-Filtrado de la mediana.

4.4.4.3.- Filtrado lineal.

Algunos filtros lineales como el filtro promedio o el filtro gaussiano son acertados para tal propósito. Por ejemplo, un filtro de promedio puede ser útil para eliminar el ruido granular. Este tipo de filtros son filtros paso bajo ya que eliminan el ruido de alta frecuencia.

Para realizar el filtrado lineal de una imagen usaremos dos funciones del "Image Processing Toolbox".

FSPECIAL.- Crea la matriz filtro.

>> H=FSPECIAL (TIPO) crea un filtro bidimensional basado en el tipo especificado, los posibles valores de TIPO son: 'average', 'gaussian', 'motion', 'sobel', 'laplacian', etc.

FILTER2.-Filtro digital bidimensional.

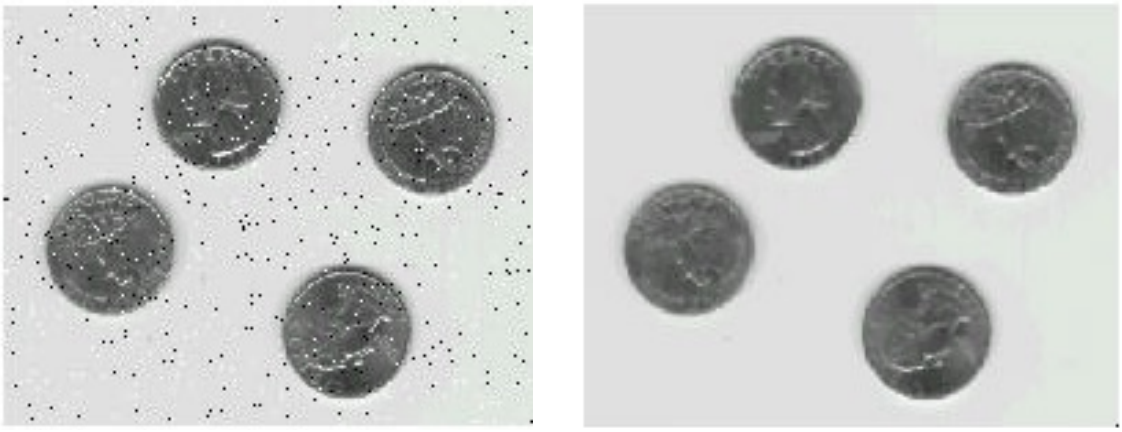
>>Y=FILTER2(B, X, TIPO) Devuelve una imagen Y computada por una correlación 2-D con el tamaño especificado en TIPO.

4.4.4.4.- Filtrado de la mediana.

En el filtrado de la mediana, el valor de un píxel viene determinado por el "valor mediano" de los pixeles de su vecindad. Este valor es más robusto que el valor promedio. Para realizar este filtrado usaremos:

MEDFILT2.- Realiza el filtro de la mediana.

>> B = MEDFILT2(A, [M N]) realiza el filtrado de la mediana de la matriz A en dos dimensiones. Cada píxel de salida contiene el valor mediano de los pixeles de vecindad M por N correspondientes al píxel de la imagen de entrada.



La imagen de la derecha es el resultado de aplicar un filtro de la mediana a la imagen de la izquierda, la cual esta distorsionada debido a un ruido de tipo sal y pimienta.

Se puede observar como este filtro elimina el ruido perfectamente, pero también al aplicar el filtro de la mediana ha disminuido la nitidez de la imagen resultado.

4.4.4.5.- Umbralización de la imagen.

Para poder separar lo que es el objeto del resto de la imagen, llamado normalmente fondo, el primer paso a realizar será el de umbralizar la imagen.

La técnica de umbralizar una imagen consiste en obtener una imagen binaria, formada por unos y ceros, a partir de una imagen en escala de grises determinando, a partir de un cierto nivel de gris, el valor del píxel perteneciente a la imagen resultante, siendo cero si es menor y uno si es mayor o igual. El "Image Processing Toolbox" también posee una función para umbralizar.

IM2BW.- Convierte una imagen a binario por umbralización.

>> B=IM2BW(A, LEVEL) Convierte una imagen almacenada en la matriz A en una imagen binaria, donde los pixeles pertenecientes a la imagen A que tengan menor iluminancia que la determinada por LEVEL, darán como resultado un cero en la imagen B, y 1 para todos los demás pixeles.

El valor de Umbralización que especificamos en LEVEL esta dentro del rango [0, 1], aunque para calcular un nivel de gris apropiado para umbralizar nuestra imagen, podemos utilizar otra función perteneciente al "Image Processing Toolbox".

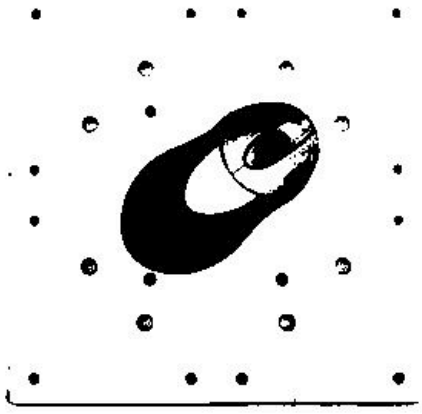
GRAYTHRESH.- Calcula el umbral de una imagen.

>> LEVEL = GRAYTHRESH(I) Calcula el umbral de una imagen I normalizando el valor de intensidad dentro del rango [0, 1], el cual minimiza la interferencia entre el umbral de pixeles negros y el umbral de pixeles blancos.

Veamos un ejemplo:

```
>> A=imread('foto.jpg');
>> B=rgb2gray(A);
>> level=GRAYTHRESH(B);
>> C=im2bw(B, level);
```

Las imágenes ‘foto.jpg’ y la foto obtenida C se muestran a continuación:



Podemos observar como los resultados obtenidos son adecuados para poder extraer lo que es el objeto que se encuentra encima de las bandejas del resto de la imagen.

4.4.4.6.- Función Invertir.

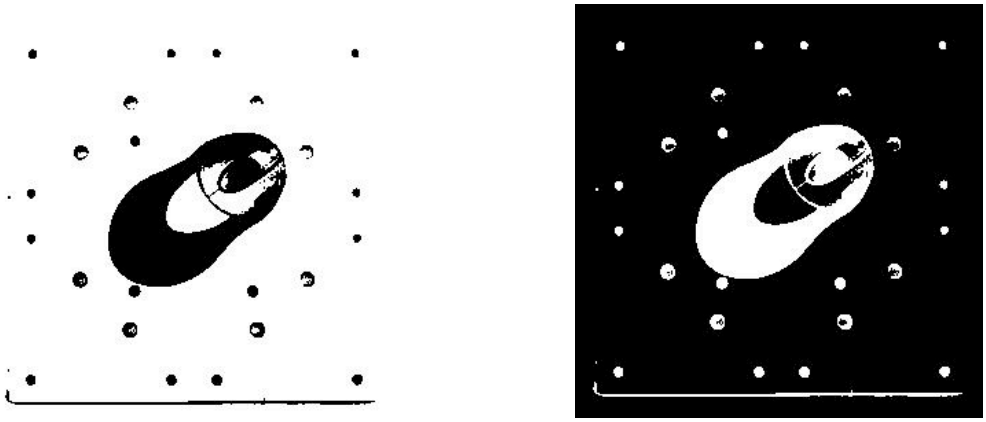
Antes de poder aplicar operaciones morfológicas a nuestra imagen binaria, es necesario realizar una pequeña transformación a la imagen para que los resultados sean los correctos.

Esta transformación se llamará invertir y se almacenará como un archivo M-file. Dado que en nuestra imagen binaria el fondo esta formado por unos y el objeto esta formado por los pixeles de valor cero, la función invertir tendrá como objetivo devolver una nueva imagen binaria donde el valor de los pixeles estará invertido, es decir, los pixeles de la imagen de entrada cuyo valor sea 1, darán como resultado pixeles de valor 0 en la imagen de salida, y viceversa.

****

```
%Función invertir, nos devuelve una imagen binaria inversa a la imagen
binaria de entrada
function C=invertir(A)
%Obtenemos el tamaño de la imagen de entrada
[x y]=size(A);
%Realizamos un bucle que nos recorrerá la imagen y sustituirá los valores
de la imagen de entrada, por los valores invertidos en la imagen de salida
for i=1:x,
   for j=1:y,
      if A(i,j)==1
         C(i,j)=0;
      else
         C(i,j)=1;
      end
   end
end
%Muestra la imagen por pantalla
imshow(C)
```

****



A la derecha se puede observar la imagen resultado de aplicar la función invertir a una imagen binaria de entrada. Ahora la imagen está preparada para las operaciones morfológicas que sean necesarias.

4.4.4.7.- Operaciones morfológicas.

Las operaciones morfológicas son métodos para procesar imágenes binarias basados en formas. Estas operaciones toman una imagen binaria de entrada y devuelven una imagen binaria como salida. El valor de cada píxel de salida esta calculado en base al correspondiente píxel de entrada y sus vecinos. Eligiendo una forma de vecindad acertada podemos construir una operación morfológica sensible a formas específicas de la imagen de entrada.

Dilatación y Erosión.

Las principales funciones que realizan operaciones morfológicas son dilatación y erosión. La dilatación añade pixeles a la frontera de los objetos mientras que la erosión quita pixeles de los bordes de los objetos.

El estado de un píxel en la imagen de salida viene determinado por la aplicación de una regla a la vecindad del píxel correspondiente en la imagen de entrada. La regla usada define si la operación es una dilatación o una erosión:

-Para una dilatación, si algún píxel perteneciente a la vecindad del píxel de la imagen de entrada tiene de valor 1, el píxel de salida valdrá 1. Si no el píxel de salida será 0.

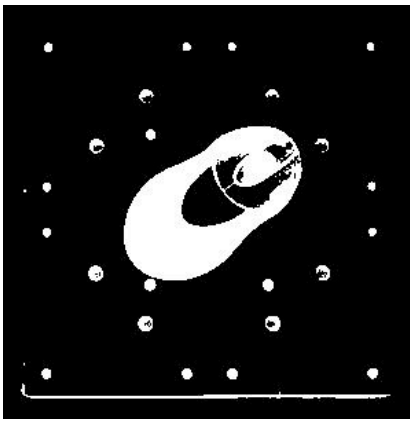
-Para una erosión, Si todos los pixeles de la vecindad del píxel de la imagen de entrada están a 1 el píxel de salida valdrá 1. Si no el píxel de salida será 0.

Las funciones incluidas en "Image Processing Toolbox" para realizar las operaciones de dilatación y erosión se describen a continuación:

IMDILATE.- Dilata una imagen.

>> A=IMDILATE(I, MASCARA) dilata la imagen binaria o a escala de grises contenida en I, devolviendo la imagen dilatada A. El parámetro MASCARA es un elemento estructural que define la máscara a utilizar en la operación morfológica. Veamos un ejemplo:

```
>> mascara=ones(3,3);
>> D=imdilate(C, mascara);
>> imshow(D);
```

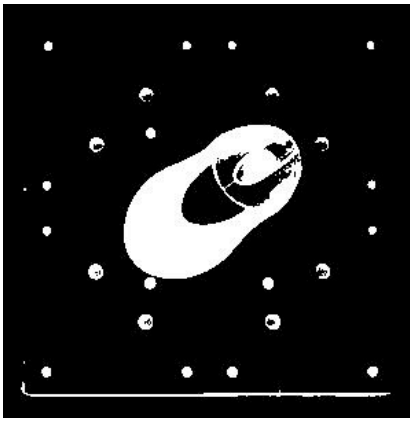


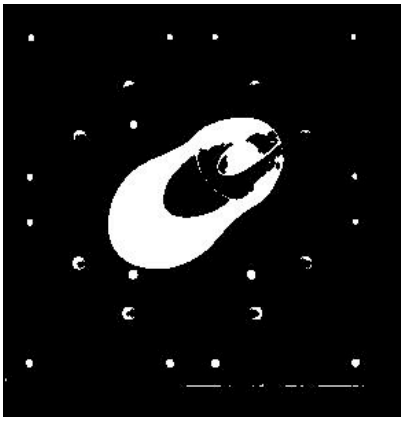
Podemos observar en la imagen de la derecha el resultado de dilatar una imagen usando una mascara de 3x3 formada por unos. Se observa como los bordes y el resto de objetos que aparecen en la imagen de la izquierda se hacen más visibles y grandes en la imagen de la derecha.

IMERODE.- Erosiona una imagen.

>> A=IMERODE(I, MASCARA) Erosiona una imagen binaria o en escala de grises, devolviendo la imagen erosionada. El parámetro MASCARA es un elemento estructural que define la máscara a utilizar en la operación morfológica. Veamos un ejemplo:

```
>> mascara=ones(3,3);
>> E=imerode(C, mascara);
>> imshow(E);
```





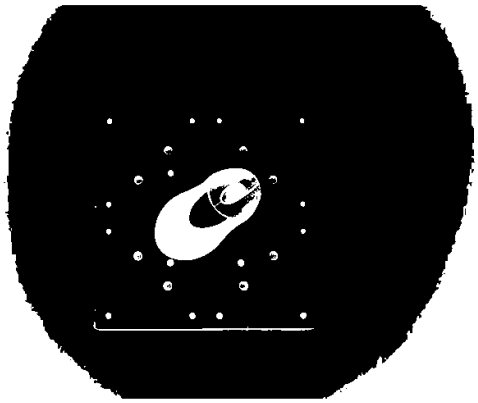
Se observa como la imagen resultado de la derecha esta erosionada, eliminando bordes innecesarios y elementos puntuales.

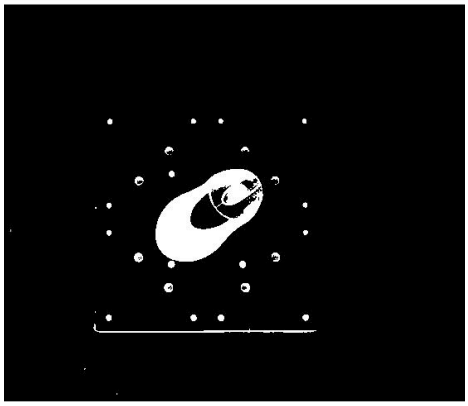
4.4.4.8.- Otras operaciones morfológicas.

Las siguientes funciones pertenecientes a "Image Processing Toolbox" que se explican a continuación, son de gran utilidad y algunas serán de uso en el presente proyecto:

IMCLEARBORDER.- Limpia los bordes de la imagen.

>> B=IMCLEARBORDER(A, CONN) Suprime las estructuras formadas por unos que están conectadas con el borde de la imagen. La imagen de entrada puede ser tanto imagen en escala de grises como binaria. La conectividad por defecto es 8 para 2 dimensiones, siendo posible especificarla mediante el parámetro CONN.

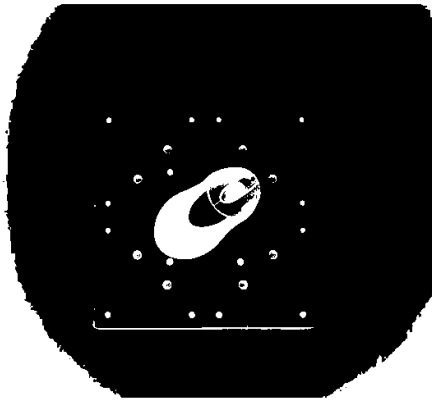


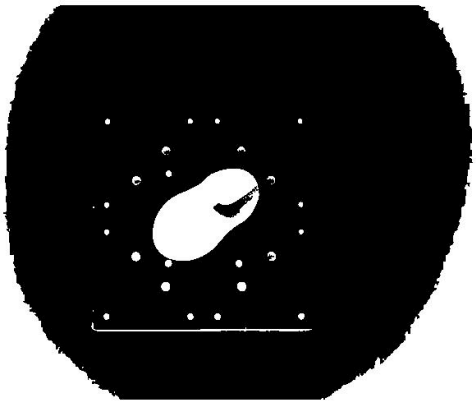


En la imagen de la derecha podemos observar como se han retirado todos los pixeles con valor 1 que estaban conectados al borde de la imagen.

IMFILL.- Rellena regiones de la imagen.

>> B=IMFILL(A, LOCATIONS) realiza la tarea de rellenar regiones de la imagen especificadas en el parámetro LOCATIONS, el cual puede ser tanto un vector que indique el comienzo de la región, como una matriz que defina completamente la región a tratar.

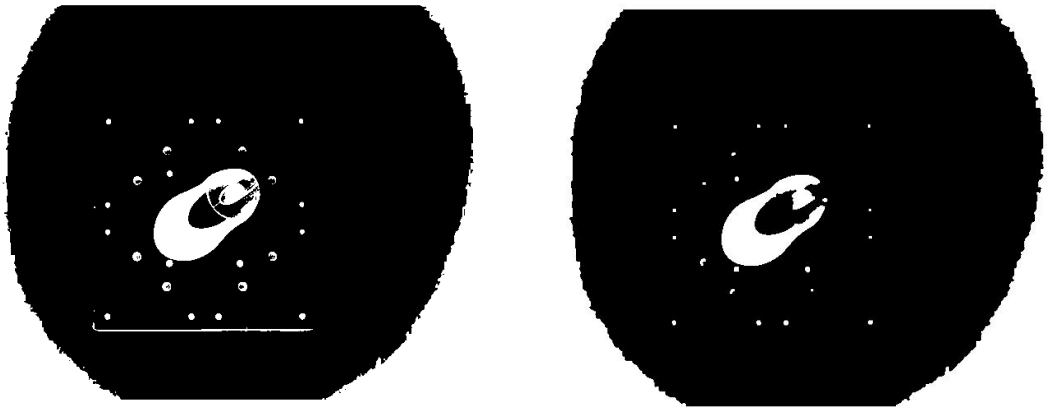




Observamos como en la imagen de la derecha casi todos los agujeros han sido rellenados, en este caso seria necesario realizar otra pasada para obtener buenos resultados.

IMOPEN.- Abre una imagen.

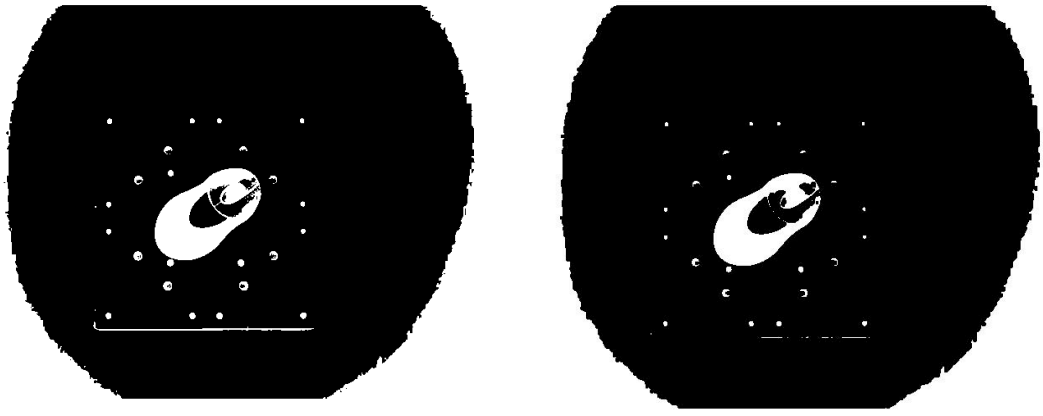
>> B=IMOPEN(A, MASCARA) Realiza la operación morfológica "opening" sobre la imagen de entrada A usando la máscara especificada por el parámetro MASCARA, el cual debe ser una matriz.



Se observa en la imagen de la derecha el resultado de la operación morfológica "opening" que es la aplicación de una erosión más una dilatación, con una mascara de 3x3 formada por unos.

IMCLOSE.- Cierra una imagen.

>>B=IMCLOSE(A, MASCARA) Realiza la operación morfológica "closing" sobre la imagen de entrada A usando la máscara especificada por el parámetro MASCARA, el cual debe ser una matriz.



En la figura de la derecha se observa el resultado de la operación "closing" que es la aplicación de una dilatación más una erosión, con una mascara de 3x3 formada por unos.

BWAREAOPEN.- Quita objetos pequeños.

>> B=BWAREAOPEN(A, P, CONN) Elimina de una imagen binaria todos los componentes conectados (objetos) que tengan menos de P pixeles, produciendo otra imagen binaria B. La conectividad por defecto es 8 para dos dimensiones y 26 para tres dimensiones. Es posible especificar la conectividad a través del parámetro CONN.

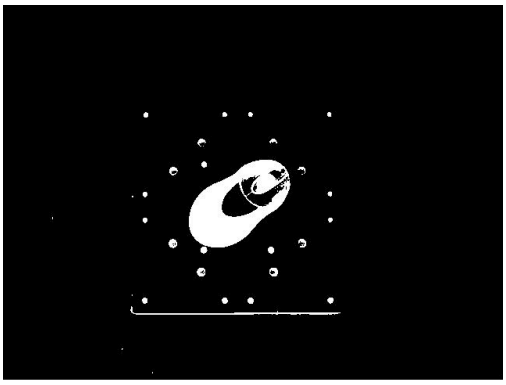


Imagen A

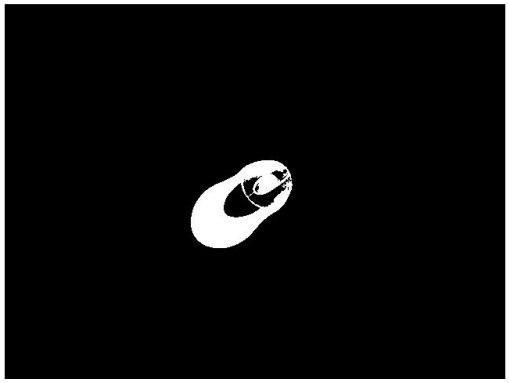


Imagen B

Se observa en la figura de la derecha el resultado de realizar una operación BWAREAOPEN con un umbral P igual a 1000 pixeles.

4.4.5.- Extracción de información de la imagen.

4.4.5.1.- Obtención del contorno.

El siguiente paso será, después de las operaciones morfológicas, el de obtener el perímetro del objeto a agarrar. Para ello necesitaremos hallar el perímetro, es decir, el conjunto de pixeles que forman el borde exterior del objeto. Para ello usaremos el "Image Processing Tools" de Matlab, el cual incluye una función para encontrar el perímetro.

BWPERIM.- Encuentra pixeles del perímetro en una imagen binaria.

>>B=BWPERIM(A, CONN) Devuelve una imagen binaria formada solamente por los pixeles del perímetro de los objetos de la imagen A. Un píxel pertenece al perímetro si tiene como valor 1 y esta conectado con al menos otro píxel de valor 1. La conectividad por defecto es 4 para dos dimensiones, aunque puede especificarse mediante el parámetro CONN.

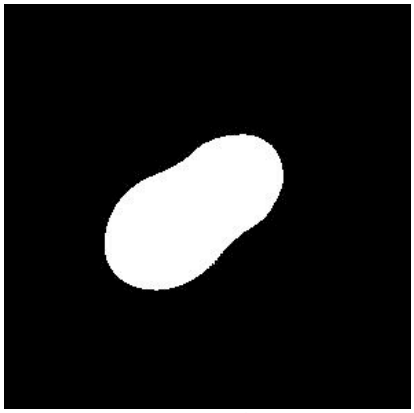


Imagen A

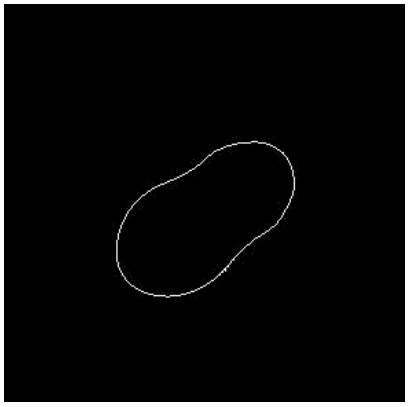


Imagen B

El resultado, imagen B, es una imagen binaria formada por los pixeles frontera que forman el objeto de la imagen A.

También es un buen momento para asegurarnos que nuestro sistema no ha cometido un fallo, gracias a la función bwboundaries perteneciente al "Image Processing Toolbox" que se describe a continuación:

BWBOUNDARIES.- Traza regiones frontera en una imagen binaria.

>> [B, L]=BWBOUNDARIES(BW, OPTION) Traza la frontera exterior de objetos y la frontera de los agujeros dentro de esos objetos. La imagen de entrada BW debe ser una imagen binaria donde los pixeles que forman el objeto deben tener de valor 1 y los pixeles que forman el fondo valor 0.

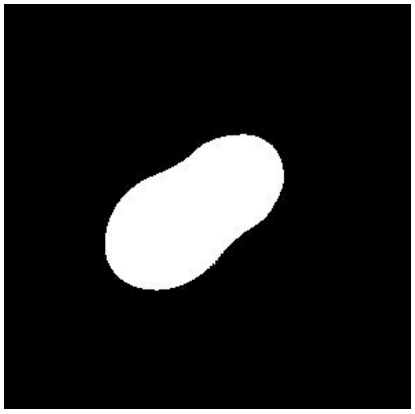
B es un vector de celdas Px1, donde P es el número de objetos y agujeros. Cada celda contiene una matriz de Qx2, donde Q es el número de pixeles frontera que forman el correspondiente objeto. Cada fila de esa matriz Qx2 contiene las coordenadas de cada píxel frontera.

El número de objetos viene representado por L y es igual a max(L(:)). Además se le pueden especificar algunas opciones como:

'noholes' – aumenta la velocidad del algoritmo buscando solo la frontera de objetos.

'holes' – por defecto o cuando se especifica, el algoritmo buscará tanto las fronteras de objetos como las de agujeros.

```
>> [B, L]=bwboundaries(BW,'noholes');
>> numRegiones=max(L(:));
>> imshow(label2rgb(L));
numRegiones=1
```



Gracias a esta función, si el número de objetos encontrados es superior a uno, podremos decirle al programa de control que vuelva a tomar una foto y reajustar los parámetros para repetir el proceso, ya que solo se admite una pieza por bandeja.

4.4.5.2.- Obtención del centro de gravedad.

Una vez obtenido el perímetro del objeto situado encima de las bandejas, lo siguiente a realizar será obtener su centro de gravedad para poder enviarle al robot las coordenadas para posicionarse encima del centro de gravedad del objeto a agarrar.

Para ello el "Image Processing Toolbox" posee una función que facilitará esta tarea:

REGIONPROPS.- Calcula propiedades de las regiones de la imagen.

>> STATS=REGIONPROPS(L, PROP) Calcula una serie de propiedades por cada región catalogada en la matriz L. Los elementos positivos de L corresponden a diferentes regiones. Por ejemplo L=1 corresponde a la región 1, L=2 corresponde con la región 2, y así sucesivamente. El valor devuelto STATS es un vector de estructuras de tamaño max(L(:)). Las celdas del vector de estructuras contienen propiedades de cada región.

PROP es un parámetro que puede tomar dos valores:

'all' – Todas las propiedades de la región del objeto se calcularán.

'basic' – Por defecto o especificando este valor, solamente se calcularan el 'Area', 'Centroid', y 'BoundingBox'.

Ahora se comentan las propiedades calculadas que resultan más interesantes:

'Area'.- Es el número de pixeles que forman la región.

'BoundingBox'.- Es el menor rectángulo que contiene a esa región. El valor devuelto es un vector [inicio, ancho] donde inicio es la coordenada de la esquina superior izquierda del rectángulo y ancho especifica el ancho del rectángulo para cada dimensión.

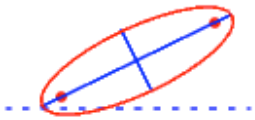
'Centroid'.- Es el centro de gravedad de la región. El valor devuelto es un vector 1xndims(L) donde el primer elemento de 'Centroid' es la coordenada x del centro de gravedad y la segunda es la coordenada y. Todos los demás elementos de 'Centroid' están ordenados por dimensión.

'MajorAxisLength'.- Es la longitud en pixeles del eje mayor de la elipse que tiene los mismos momentos que la región.

´MinorAxisLength'.- Es la longitud en pixeles del eje menor de la elipse que tiene los mismos momentos que la región.

'Eccentricity'.- Es la excentricidad de la elipse que tiene los mismos momentos que la región. El valor está entre el 0 y el 1, una excentricidad de valor cero seria igual a un círculo mientras que un valor cercano a uno correspondería con una línea o segmento.

'Orientacion'.- Es el ángulo entre el eje x y el eje mayor de la elipse que tiene los mismos momentos que la región.



4.4.5.3.- Obtención de los puntos de agarre.

Para obtener los puntos de agarre que necesitamos para orientar la pinza del robot, utilizaremos la imagen binaria que contiene la silueta. Tras analizar la imagen binaria con la silueta mediante el comando:

>> [B, L]=bwboundaries(BW,'noholes');

En la variable de salida B se almacenan los perímetros de todas las regiones encontradas, en nuestro caso solo tendremos una. Con el comando:

>> A=B(1);

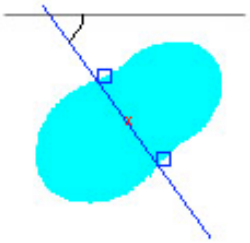
Obtendremos una variable A formada por dos columnas y tantas filas como puntos formen la silueta de la región 1.

Lo siguiente será dividir la imagen por la mitad a partir del centro de gravedad obtenido anteriormente, respecto al eje X o al eje Y independientemente. Con esto se consigue obtener dos puntos de agarre, uno para cada mitad de la imagen.

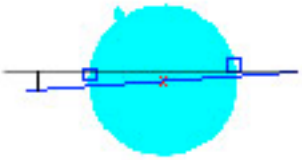
Para encontrar el punto de agarre de cada mitad, se recorre todo el perímetro de la silueta perteneciente a la mitad seleccionada, comprobando en cada píxel su distancia al centro de gravedad, siendo el punto de agarre el píxel perteneciente al perímetro que minimice esta distancia.

Una vez obtenidos los dos puntos de agarre, se procede a calcular el ángulo que forman el eje de coordenadas X y la recta que pasa por los dos puntos de agarre.

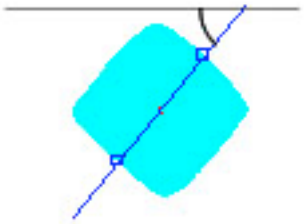
Este ángulo calculado es el ángulo que se enviará al robot para que oriente la pinza de forma que se produzca un agarre correcto. Queda por decir que el ángulo obtenido es el ángulo formado por el margen superior de la imagen con la recta que une los puntos de agarre y pasa por el centro de gravedad.



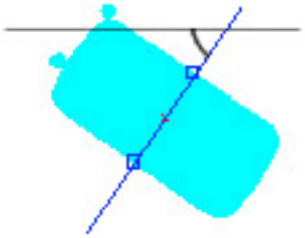
Ángulo de agarre=54.2726



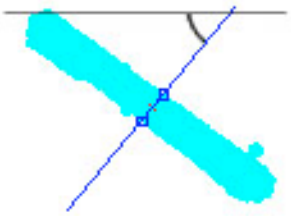
Ángulo de agarre=-3.8141



Ángulo de agarre=-51.1592



Ángulo de agarre=-55.9050



Ángulo de agarre=-51.0090

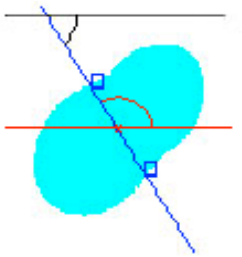
Ángulo de agarre=59.9314

Para obtener el ángulo de agarre en el sistema de referencia del robot, se programa una función cuyo nombre es calculaAngulo.m y se describe a continuación:

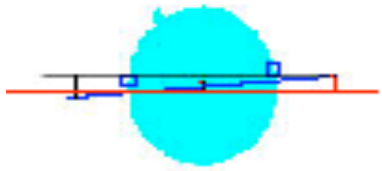
****

```
%Nos devuelve el ángulo correcto que hay que enviarle al robot
function newAngulo=calculaAngulo(anguloAgarre)
%Dependiendo de si el ángulo de agarre es positivo o negativo
if (anguloAgarre>0)
%Se calcula el equivalente en el segundo cuadrante
   newAngulo=90+(90-anguloAgarre);
else
%Se le calcula el equivalente en el primer cuadrante
   newAngulo=-anguloAgarre;
end
```

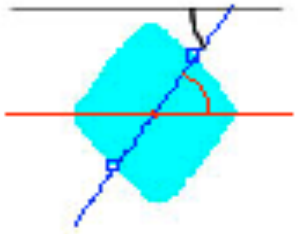
****



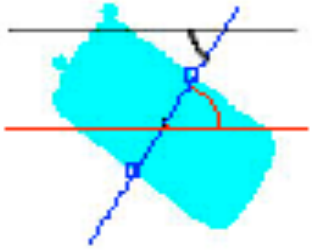
Ángulo de agarre=125.7274



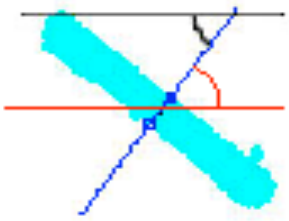
Ángulo de agarre=3.8141



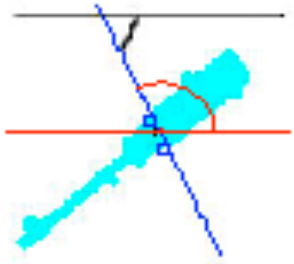
Ángulo de agarre=51.1592



Ángulo de agarre=55.9050



Ángulo de agarre=51.0090



Ángulo de agarre=120.0686

4.4.6.- Programación del sistema de visión artificial.

Para la realización cíclica de la tarea de devolver las coordenadas para posicionar el robot y la orientación de las pinzas del robot para el agarre de la pieza de forma correcta, se crea el siguiente programa que tiene como entrada una imagen capturada por la cámara Web. El programa se comenta a continuación:

****

```
%Función que tiene como entrada una imagen capturada por la cámara
%Tiene como parámetro la matriz A que contiene la información de la
%imagen devuelve las coordenadas del centro de gravedad para colocar el
%brazo robot, el ángulo respecto al eje X para colocar las pinzas y el
%numero de elementos que contiene la imagen

function [Xcentro,Ycentro,anguloRobot,numRegiones]=
calculaCord(imColor)
%Se realiza la transformación de la imagen en color a imagen en escala de
%grises
   imGris=rgb2gray(imColor);

%Se calcula el nivel óptimo de gris para umbralizar la imagen
   level=graythresh(imGris);

%Se umbraliza la imagen convirtiéndola en binara usando la imagen en
%escala de grises y el nivel de gris calculado
   imBn=im2bw(imGris,level);

%Invertimos la imagen para que el objeto sea de valor 1 y el fondo de
%valor 0
   imInv=invertir(imBn);

%Se realiza una operación morfológica para eliminar los bordes
   sinBordes=imclearborder(imInv);

%Se rellenan los huecos de la imagen
   sinHuecos=imfill(sinBordes);

%Usando la función bwareaopen eliminamos las regiones que tengan
%menos de 2000 pixeles
   silueta=bwareaopen(sinHuecos,2000);
```

```
%Se calcula la silueta del objeto a agarrar
   [B,L]=bwboundaries(silueta,'noholes');

%Se obtiene el número de regiones que hay en la imagen, si es mayor de
%uno se cometió un fallo
   numRegiones=max(L(:));

%Se obtiene las propiedades de la región
   stats=regionprops(L,'all');

%Obtenemos las coordenadas del centro de gravedad
   centro=[stats.Centroid];
   Xcentro=centro(1);
   Ycentro=centro(2);

%Se obtienen las coordenadas de los puntos del perímetro, outline es una
%matriz que tiene tantas filas como puntos. En la columna 2 se encuentran
%las coordenadas X de todos los puntos del perímetro, en la columna uno
%las coordenadas Y Copiamos todos los parámetros de B que sean de la
%región 1
   outline=B{1};

%Buscamos los puntos de agarre óptimos, para ello dividiremos la imagen
%por la mitad respecto al eje X y calculamos un óptimo para cada parte
%Obtenemos el tamaño de outline
   tam=size(outline);

%Declaramos dos variables para poder hacer el bucle
   minD1=800;
   minD2=800;

%Iniciamos el bucle donde tam(1) contiene el numero de filas que hay en
%outline
   for i=1:tam(1)
      %Para la primera mitad de la imagen
      if(outline(i,2)<=Xcentro)
         min1x=outline(i,2)-Xcentro;
         min1y=outline(i,1)-Ycentro;
         d1=((min1x)^2+(min1y)^2)^(1/2);
         if(d1<minD1)
            minD1=d1;
            punto1=i;
         end
```

```
    %Para la segunda mitad de la imagen
    else
        min2x=outline(i,2)-Xcentro;
        min2y=outline(i,1)-Ycentro;
        d2=((min2x)^2+(min2y)^2)^(1/2);
        if(d2<minD2)
            minD2=d2;
            punto2=i;
        end
    end
end

%Obtenemos las coordenadas de los puntos de agarre
Xagarre1=outline(punto1,2);
Yagarre1=outline(punto1,1);
Xagarre2=outline(punto2,2);
Yagarre2=outline(punto2,1);

%Calculamos la distancia entre puntos de agarre
long=(((Xagarre2-Xagarre1)^2)+((Yagarre2-Yagarre1)^2))^0.5;

%Calculamos el ángulo que forma la línea que pasa por los dos
%puntos de agarre respecto al eje X
anguloAgarre=asind((Yagarre2-Yagarre1)/long)

%Para calcular el ángulo a enviar al robot usamos la función
%calculaAngulo
anguloRobot=calculaAngulo(anguloAgarre);

%Mostramos por pantalla los resultados obtenidos.
%
%
subplot(3,3,1)

%Se muestra la imagen en color de entrada
imshow(imColor)
title('Imagen en color')
subplot(3,3,2)

%Se muestra la imagen en escala de grises
imshow(imGris)
title('Imagen en escala de grises')
subplot(3,3,3)
```

```
%Se muestra la imagen binara
imshow(imBn)
title('Imagen binaria')
subplot(3,3,4)

%Se muestra la imagen binaria invertida
imshow(imInv)
title('Imagen binaria Invertida')
subplot(3,3,5)

%Se muestra la imagen binaria sin bordes
imshow(sinBordes)
title('Imagen binaria sin bordes')
subplot(3,3,6)

%Se rellenan los huecos de la imagen binaria
imshow(sinHuecos)
title('Imagen binaria sin huecos')
subplot(3,3,7)

%Se obtiene la silueta del objeto presente en la imagen binaria
imshow(silueta)
title('Imagen binaria solo el objeto')
subplot(3,3,8)

%Se muestra la región calculada
imshow(label2rgb(L));
title('Silueta del objeto')
subplot(3,3,9)

%Se muestra la imagen de la región junto con el centro de gravedad y
%los puntos de agarre
imshow(label2rgb(L))
title('Centro de gravedad y eje de agarre')
hold on

%Dibujamos los puntos de agarre
plot(outline(punto1,2),outline(punto1,1),'bs')
plot(outline(punto2,2),outline(punto2,1),'bs')

%Dibujamos el punto del centro de gravedad
plot(Xcentro,Ycentro,'rx')
```

```
    %Dibujamos la recta que pasa por el centro de gravedad y es paralela a
    %la recta que pasa por los puntos de agarre
    newAngulo=anguloAgarre;
    largo=100;
    XejeIni=Xcentro-(largo*cosd(newAngulo));
    YejeIni=Ycentro-(largo*sind(newAngulo));
    XejeFin=Xcentro+(largo*cosd(newAngulo));
    YejeFin=Ycentro+(largo*sind(newAngulo));
    line([XejeIni XejeFin],[YejeIni YejeFin])
    hold off
end
```

****

A continuación se comentan los resultados obtenidos al aplicar este algoritmo al tratamiento de imágenes tomadas por la cámara Web, en el apartado 5.1.

5.-Resultados y conclusiones.

Tras la incorporación del último algoritmo, los resultados obtenidos en el presente proyecto son:

- Se realizo un estudio para determinar las características necesarias que debía cumplir el nuevo autómata empleado en el presente proyecto.

- Se realizó el cableado, conexionado, la configuración y programación del sistema de transporte mediante bandejas, controlado por el autómata S7-224xp.

- Se realizó el cambio de baterías del robot RH-5AH55 y se estableció de nuevo el origen de coordenadas del mismo.

- Se programó el robot RH-5AH55 y se estableció comunicación entre el robot y el PC.

- Mediante la tarjeta de adquisición de datos, se estableció comunicación entre el autómata y el PC. También, mediante la tarjeta de adquisición de datos, se controló la apertura y cierre de la pinza del robot.

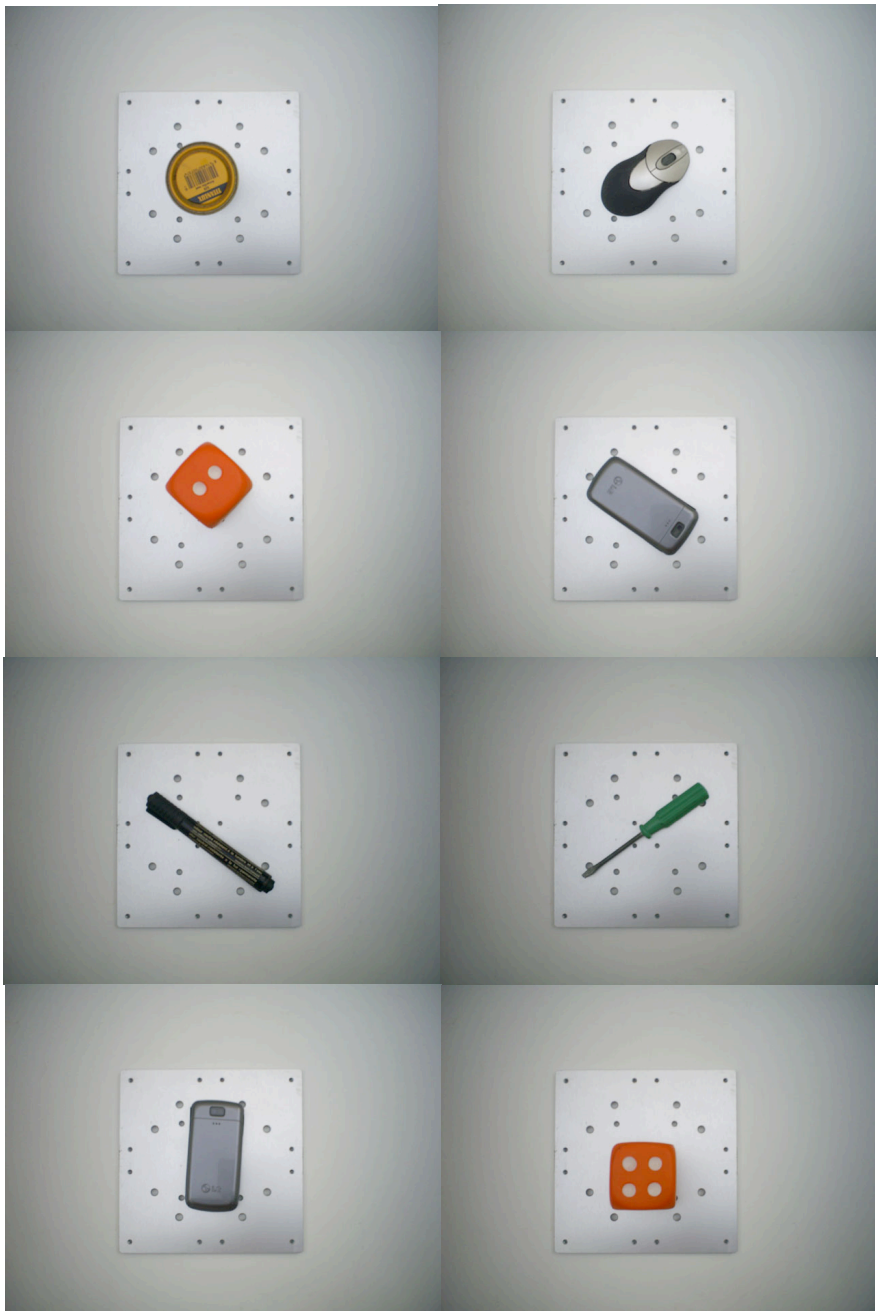
- Se estableció una comunicación para la adquisición de datos entre la cámara Web y el ordenador.

- Se diseñó, programó e implementó un sistema de control de visión artificial para el correcto agarre de cualquier pieza contenida en las bandejas del sistema de transporte.

- Se realizó la puesta a punto del sistema neumático mediante aire comprimido que da soporte a la pinza del brazo robot y a la cinta transportadora de bandejas.

5.1.- Resultados obtenidos del sistema de visión artificial.

En la siguiente imagen están las capturas realizadas por la cámara Web propuestas para diferentes objetos, las capturas tienen una dimensión de 640x480 pixeles y son imágenes en color RGB.



Para los experimentos, se han utilizado diferentes objetos comunes. Los resultados obtenidos se comentaran a continuación junto con el resultado tras aplicar la función calculaCord a la cual pasaremos como parámetro cada una de las imágenes.

Se mostrarán por cada imagen: la propia imagen en color, la imagen en escala de grises, la imagen en binario, se invertirá la imagen, se limpiarán los bordes, se rellenarán los huecos, se eliminarán las regiones con áreas menores a 2000 pixeles y por último se mostrará el centro de gravedad, los puntos de agarre y el eje de agarre a utilizar por el robot.

Los valores devueltos por la función calculaCord serán:

- La coordenada X del centro de gravedad.
- La coordenada Y del centro de gravedad.
- El ángulo de agarre para posicionar la articulación J3 del robot.
- El número de regiones dentro de la imagen capturada.

Imagen 1

Imagen en color

Imagen en escala de grises

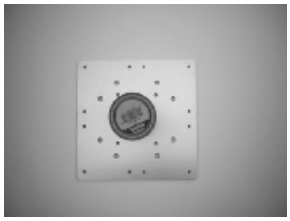


Imagen binaria

Imagen binaria Invertida

Imagen binaria sin bordes

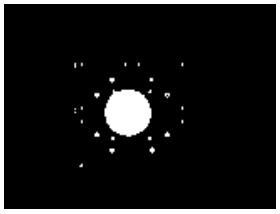


Imagen binaria sin huecos

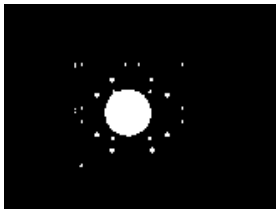
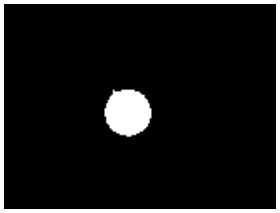


Imagen binaria solo el objeto



Silueta del objeto

Centro de gravedad y eje de agarre

Los valores devueltos por la función son:

X = 290.9983

Y = 254.5257

anguloAgarre = 3.8141

numreg = 1

Se puede observar como el agarre a realizar sería óptimo para sujetar un objeto de forma cilíndrica como el de la imagen. El número de regiones es 1, lo cual es correcto, dentro de la imagen solo hay un objeto del cual se obtienen el centro de gravedad, los puntos de agarre y el ángulo de agarre.

Imagen 2

Imagen en color

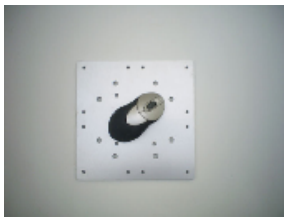


Imagen en escala de grises

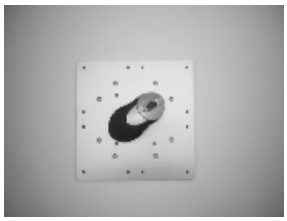


Imagen binaria

Imagen binaria Invertida

Imagen binaria sin bordes

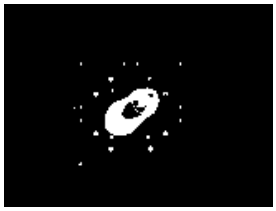


Imagen binaria sin huecos

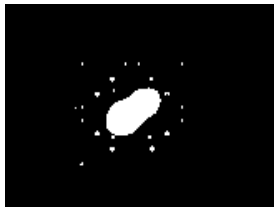


Imagen binaria solo el objeto

Silueta del objeto

Centro de gravedad y eje de agarre

Los valores devueltos por la función son:

X = 302.1880

Y = 255.3685

anguloAgarre = 125.7274

numreg = 1

El objeto a agarrar, en este caso, es un ratón inalámbrico de ordenador, siguiendo los pasos del programa, se obtiene el contorno, el centro de gravedad y el ángulo de agarre óptimo. Los resultados obtenidos son buenos y no hay fallo ya que solo hay un objeto detectado.

Imagen 3

Imagen en color

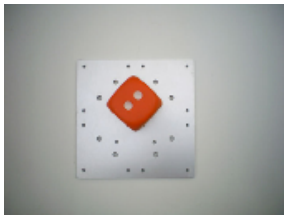


Imagen en escala de grises

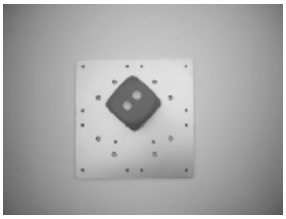


Imagen binaria

Imagen binaria Invertida

Imagen binaria sin bordes

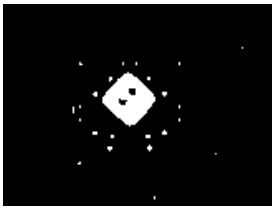


Imagen binaria sin huecos

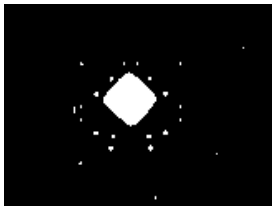


Imagen binaria solo el objeto

Silueta del objeto

Centro de gravedad y eje de agarre

Los valores devueltos por la función son:

X = 299.1570

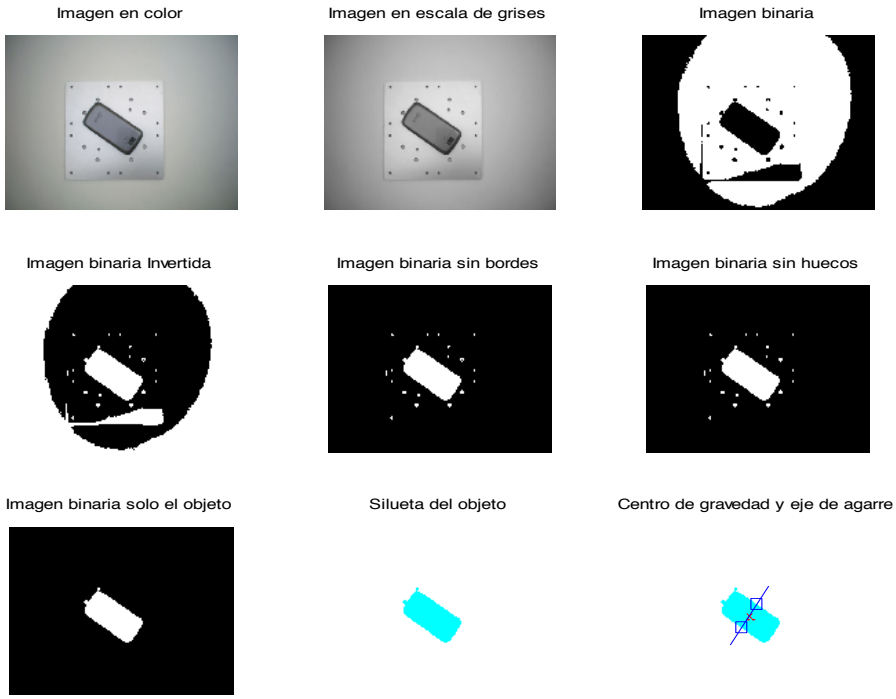
Y = 227.0965

anguloAgarre = 51.1592

numreg = 1

Se puede observar como los dos círculos del dado no han interferido en el calculo del perímetro del objeto, el ángulo de agarre es correcto y los puntos de agarre óptimos, el número de regiones es uno, lo cual, es correcto.

Imagen 4



Los valores devueltos por la función son:

X = 296.3903

Y = 256.3243

anguloAgarre = 55.9050

numreg = 1

En la imagen de la silueta se puede observar como hay algunas regiones que no pertenecen al objeto debidas a algunos de los agujeros de la bandeja de aluminio pero esto no influye de manera negativa y de forma significante a la hora de calcular un centro de gravedad y un ángulo de agarre óptimo.

Imagen 5

Imagen en color

Imagen en escala de grises

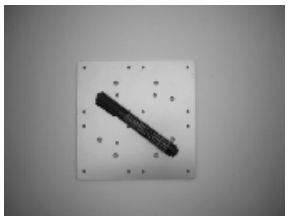


Imagen binaria

Imagen binaria Invertida

Imagen binaria sin bordes

Imagen binaria sin huecos

Imagen binaria solo el objeto

Silueta del objeto

Centro de gravedad y eje de agarre

Los valores devueltos por la función son:

X = 304.3146

Y = 273.9527

anguloAgarre = 51.0090

numreg = 1

Los resultados para la imagen del rotulador son aceptables, aunque observando la imagen binaria que solo contiene el objeto se ha colado una región perteneciente a un agujero de la bandeja. Aun así los puntos de agarre y el ángulo de agarre son aceptables.

Imagen 6

Imagen en color

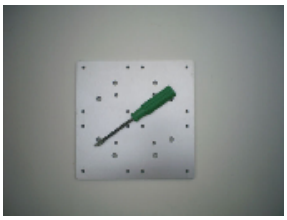


Imagen en escala de grises

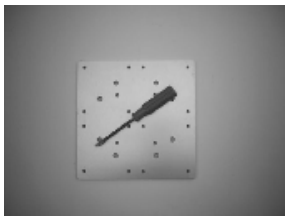


Imagen binaria

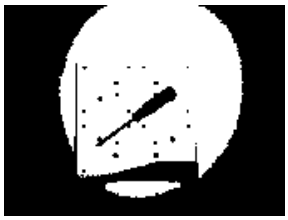


Imagen binaria Invertida

Imagen binaria sin bordes

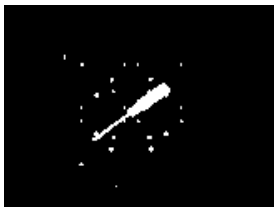


Imagen binaria sin huecos

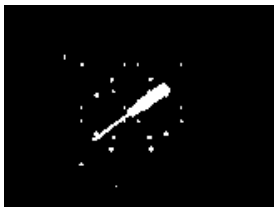


Imagen binaria solo el objeto

Silueta del objeto

Centro de gravedad y eje de agarre

Los valores devueltos por la función son:

X = 320.9746

Y = 241.9791

angulAgarre = 120.0686

numreg = 1

En el caso del destornillador también se aprecia como en la punta de este se ha colado una región perteneciente al fondo de la imagen. Aun así volvemos a tener unos resultados aceptables al obtener el centro de gravedad y el ángulo de agarre.

Imagen 7

Imagen en color

Imagen en escala de grises

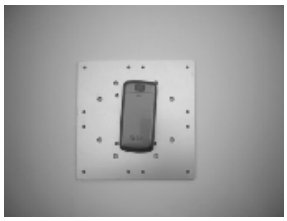


Imagen binaria

Imagen binaria Invertida

Imagen binaria sin bordes

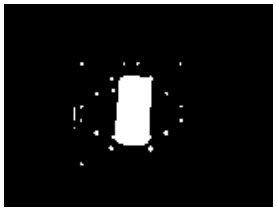


Imagen binaria sin huecos

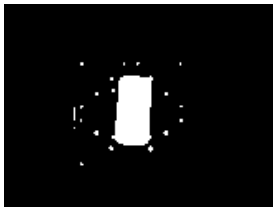


Imagen binaria solo el objeto

Silueta del objeto

Centro de gravedad y eje de agarre

Los valores devueltos por la función son:

X = 307.7740

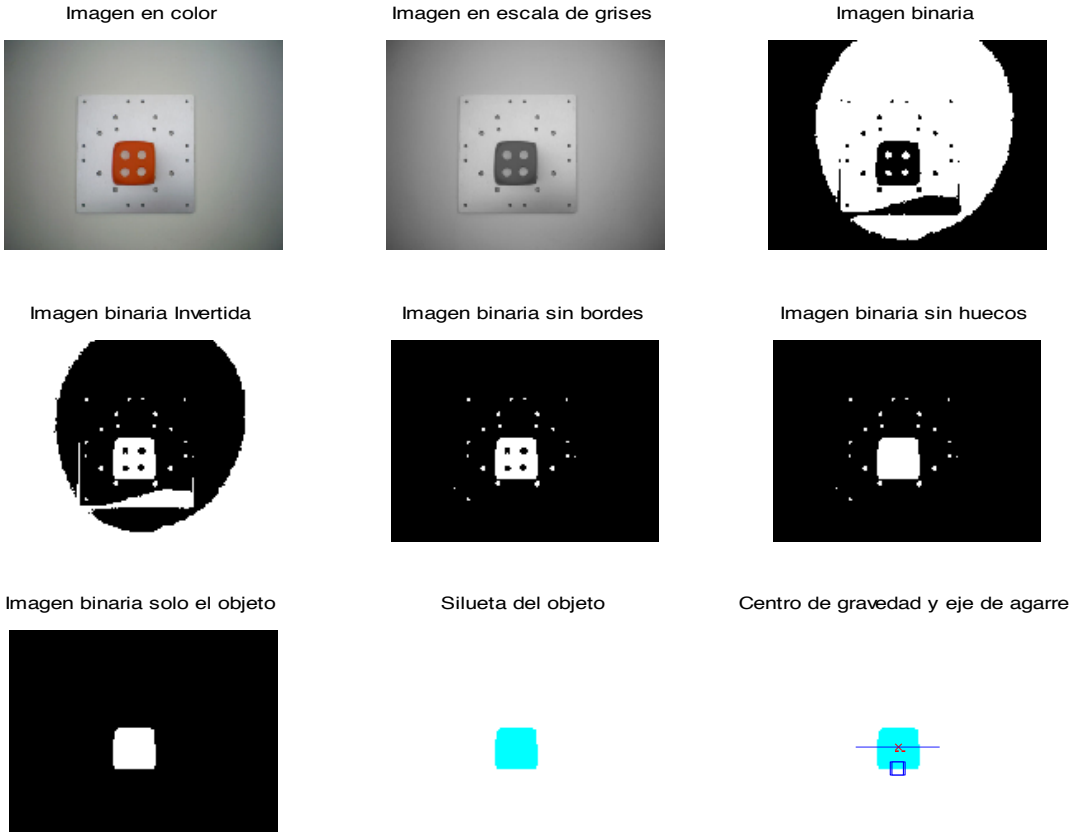
Y = 253.4549

anguloAgarre = 174.1440

numreg = 1

En la imagen obtenida podemos observar como los puntos de agarre, el centro de gravedad y el eje de agarre son aceptables para la correcta sujeción de la pieza mediante la pinza del brazo robot.

Imagen 8



Los valores devueltos por la función son:

X = 298.9685

Y = 283.0693

anguloAgarre = 0

numreg = 1

En este caso los puntos de agarre obtenidos no son aceptables, se han obtenido dos puntos de agarre juntos, lo cual será debido a irregularidades en la silueta del objeto. De todas formas con los resultados obtenidos el robot podría realizar un correcto agarre de la pieza.

5.2.- Entorno experimental.

Para realizar el experimento se dispondrá de la cinta transportadora controlada por el autómata, el brazo robot Mitsubishi RH-5AH55, con cuatro grados de libertad y configuración SCARA, el PC y la cámara Web.

La pinza de dos dedos a utilizar es del modelo HDT-1680 de la clase chelic, de accionamiento neumático y con un recorrido de 80mm, lo que permite realizar agarres adecuados. Los dedos de la pinza han sido diseñados específicamente para este trabajo, y permiten utilizar diferentes rangos de apertura en función del montaje:

- Rango 1: entre 4 y 84mm.
- Rango 2: entre 14 y 94mm.
- Rango 3: entre 24 y 104mm.
- Rango 4: entre 34 y 114mm.

De este modo, es posible realizar multitud de experimentos distintos con la misma pinza y los mismos dedos. El rango a utilizar en los experimentos es el rango 1.

5.3.- Resultados obtenidos en experimento real.

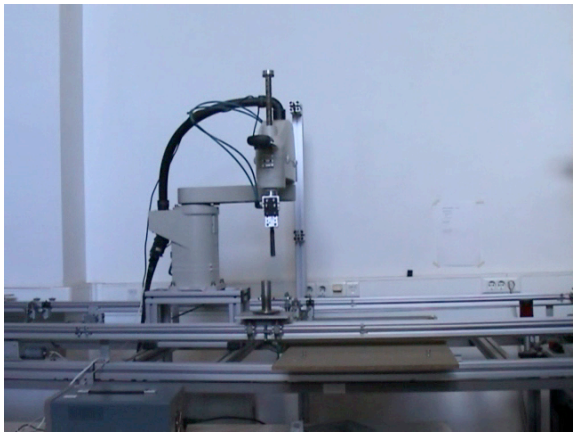
Para el experimento real de agarre se utilizo un objeto común y se realizaron pruebas como las llevadas a cabo con el sistema de visión artificial. La secuencia del experimento es la siguiente:

1. Se deposita una pieza en la bandeja, con una posición y orientación indeterminadas.

2. Se pone en funcionamiento el autómata al presionar “MARCHA” en el cuadro de mandos de la cinta transportadora.

3. La bandeja llega a la estación número 5, entonces se inmoviliza la bandeja y se detiene la cinta

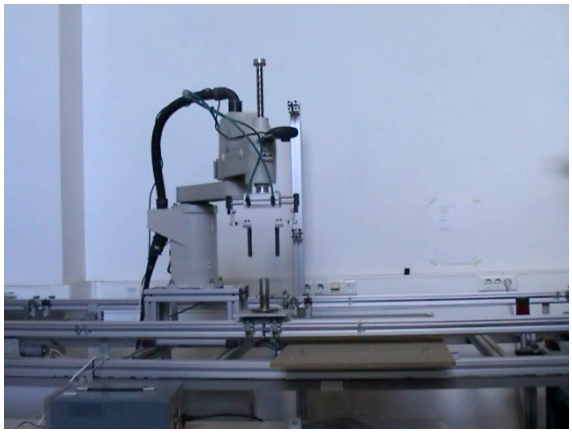
4. El robot se desplaza hacia la posición de captura de imágenes (necesario dado que la cámara es solidaria al robot).



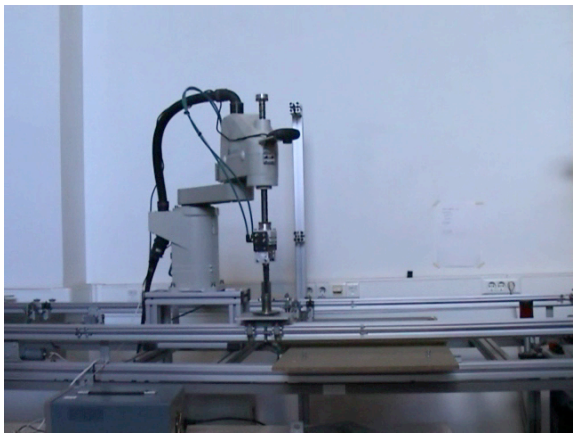
5. Se adquiere la imagen del objeto y se extrae el contorno del mismo.

6. Se calculan los puntos de agarre y orientación de la pinza, utilizando la función calculaCord.m.

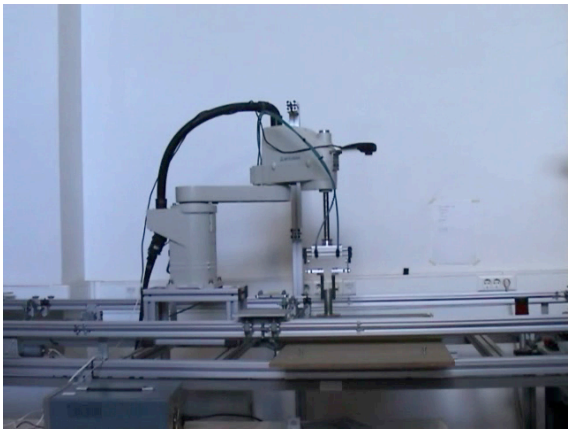
7. Se coloca el robot en la posición y orientación de agarre calculadas.



8. Desciende el brazo robot hasta la altura prefijada de agarre y se acciona la pinza para agarrar el objeto.



9. Se desplaza el objeto hasta la posición de destino, donde se deposita.



10. Se quitan los topes y los frenos que inmovilizaban la bandeja.

5.4.-Trabajos futuros.

Para finalizar, partiendo del trabajo ya realizado, se proponen algunas guías para futuros trabajos con el equipo descrito en este proyecto de fin de carrera:

- Diseño de una interfaz hombre maquina para la correcta supervisión y control de la célula de fabricación flexible del presente proyecto de fin de carrera.

- Programar un diagnostico para poder testear desde el PC, tanto los demás componentes de la célula de fabricación flexible como la comunicación entre ellos.

- Realizar una gestión completa de errores del proceso para su análisis y posterior corrección de problemas, tanto del sistema de visión artificial, como del resto de los componentes.

- Realizar la compra y su posterior incorporación al robot del modulo de entradas y salidas del mismo, el cual establecería el control de la pinza del brazo robot y la comunicación entre robot y autómata (se evitaría hacer uso de la tarjeta de adquisición conectada al PC)

- Incorporación de otros sistemas autónomos, como el del almacén automático (Profibus FMS) donde el robot sería el encargado del transporte de una pieza entre el sistema de transporte automático mediante bandejas y el almacén automático, como continuación de la célula de fabricación flexible

6.-ANEXOS.

6.1.- Programación en STEP 7.

6.1.1.-Introducción a la programación de autómatas.

Se podría dividir la programación de autómatas en varios pasos:

1. Definir el sistema de control, que debe hacer, en que orden.
2. Identificar las señales de entrada y salida del autómata.
3. Representar el sistema de control mediante un modelo.
4. Asignar las direcciones de entrada/salida o internas del autómata a las correspondientes del modelo.
5. Codificar la representación del modelo de control mediante un lenguaje de programación.
6. Cargar el programa en la memoria del autómata desde el PC.
7. Depurar el programa y obtener una copia de seguridad.

6.1.2.-Definición del sistema de control.

Para la correcta definición del sistema de control será necesario el uso de herramientas de representación basadas en símbolos que permitan obtener toda la información posible acerca del sistema de control.

Según los símbolos utilizados en el modelo, la representación puede ser:

- Proposicional: descripciones literales.
- Algebraicas: funciones booleanas y aritméticas.
- Grafica: esquemas de contactos, diagramas lógicos, técnicas Grafcet.

6.1.3.-Asignación de las direcciones de entrada/salida o internas del autómata.

La memoria de datos del autómata se compone de datos y objetos. En la parte de datos, almacenamos la memoria de variables, la imagen del proceso de las entradas (I), la imagen del proceso de las salidas (Q), marcas internas (M) y marcas especiales (SM). En la parte de Objetos se almacenaran temporizadores, contadores, Entradas analógicas, salidas analógicas, acumuladores y contadores rápidos.
Direccionamiento de la imagen del proceso de las entradas (I):

La CPU lee las entradas físicas al comienzo de cada ciclo y escribe los correspondientes valores en la imagen del proceso de las entradas según el siguiente formato:

Bit	I[direcc.del byte].[direcc.del bit] I0.1
Byte	I[tamaño][direcc.del byte inicial] IB3

Direccionamiento de la imagen del proceso de las salidas (Q):

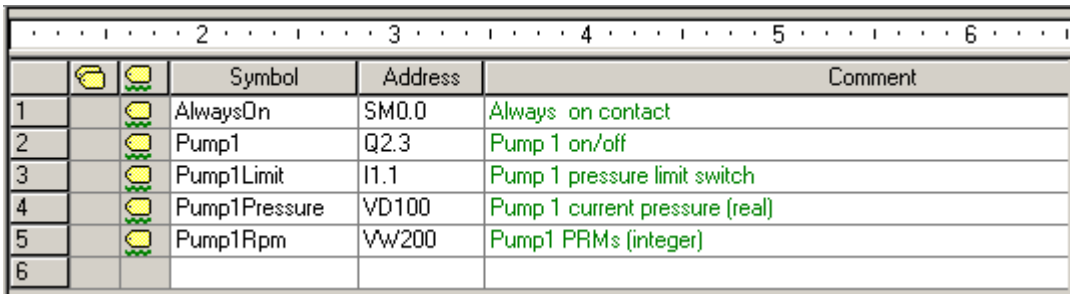
Al final de cada ciclo, la CPU copia en las salidas físicas el valor almacenado en la imagen del proceso de las salidas según el siguiente formato:

Bit	Q[direcc.del byte].[direcc.del bit] Q2.1
Byte	Q[tamaño][direcc.del byte inicial] QB6

En la tabla de símbolos es posible definir y editar los símbolos a los que pueden acceder los nombres simbólicos en cualquier parte del programa. Es posible crear varias tablas de símbolos. La tabla de símbolos incluye también una ficha que contiene los símbolos definidos por el sistema utilizables en el programa de usuario. La tabla de símbolos se denomina también tabla de variables globales.

A los operandos de las operaciones se pueden asignar direcciones absolutas o simbólicas. Una dirección absoluta utiliza el área de memoria y un bit o un byte para identificar la dirección. Una dirección simbólica utiliza una combinación de caracteres alfanuméricos para identificar la dirección.

En los programas SIMATIC, los símbolos globales se asignan utilizando la tabla de símbolos. En los programas IEC, los símbolos globales se asignan utilizando la tabla de variables globales.



	Symbol	Address	Comment
1	AlwaysOn	SM0.0	Always on contact
2	Pump1	Q2.3	Pump 1 on/off
3	Pump1Limit	I1.1	Pump 1 pressure limit switch
4	Pump1Pressure	VD100	Pump 1 current pressure (real)
5	Pump1Rpm	VW200	Pump1 PRMs (integer)
6			

Para asignar un símbolo a una dirección:

1. Haga clic en el icono ”Tabla de símbolos” en la barra de navegación para abrir la tabla de símbolos.
2. En la columna ”Nombre simbólico”, teclee el nombre del símbolo (p. ej. ”Entrada1”). Un nombre simbólico puede comprender 23 caracteres como máximo.
3. En la columna ”Dirección”, teclee la dirección (p. ej. I0.0).
4. Si está utilizando la tabla de variables globales (IEC), introduzca un valor en la columna ”Tipo de datos” o seleccione uno del cuadro de lista.

Es posible crear varias tablas de símbolos. No obstante, una misma cadena no se puede utilizar más de una vez como símbolo global, ni en una misma tabla ni en tablas diferentes.

6.1.4.- Representación del sistema de control.

Para definir correctamente el funcionamiento de nuestro sistema de control recurriremos al método gráfico ‘Grafcet’.

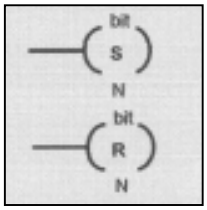
El ‘Grafcet’ es un diagrama de flujos que describe la evolución de un proceso que se pretende controlar, dividiéndolo por etapas, indicando que acciones hay que realizar en cada etapa y especificando las condiciones que se deben cumplir para la transición de una etapa a otra.

En base a este diagrama se puede obtener la secuencia de acciones que debe realizar el autómata.

6.1.5.-Programación del sistema de control.

Una vez que se ha obtenido el diagrama Grafcet que representa el proceso, hay que realizar la tarea de transformación de este en un algoritmo de control y su posterior programación sobre un autómata programable

Para ello usaremos el lenguaje de programación KOP, el cual es un lenguaje de programación basado en contactos. A cada una de las etapas en las que se divide el Grafcet se le asocia una variable interna. La condición de transición es la encargada de activar la etapa siguiente y desactivar la anterior, para ello se usan las instrucciones set y reset.



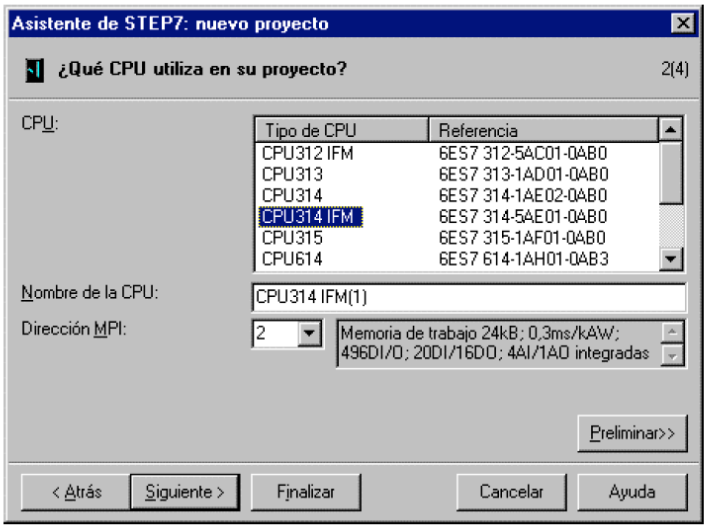
6.1.6.- Arrancar el programa y crear un proyecto.

Para arrancar el programa Step-7:

Inicio > Programas > Simatic > STEP7 > Administrador SIMATIC

Crear un nuevo proyecto:

1. De forma automática al arrancar el programa aparece la siguiente pantalla con el Asistente de STEP7 para configurar un nuevo proyecto.



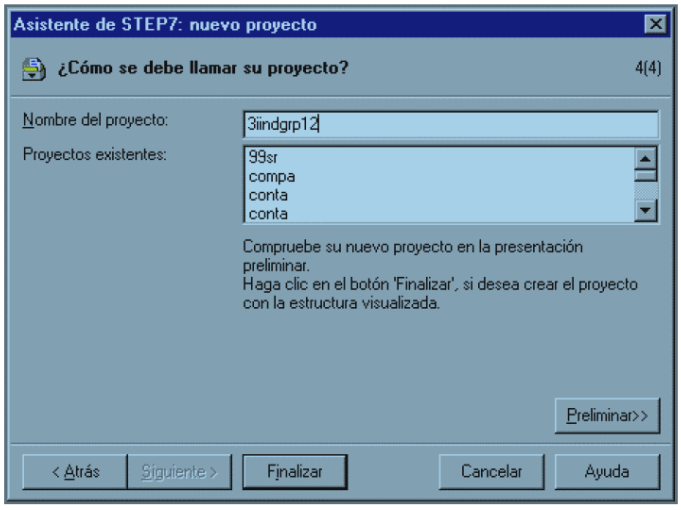
En esta ventana se selecciona el PLC a utilizar: CPU-224; y la dirección de red del PLC para poder comunicar con el desde el PC y cargar el programa. Si no hay otra indicación, la dirección correcta es MPI=2. Una vez hecha la selección, pulsar el botón de “Siguiente”.

2. Aparece la segunda pantalla del Asistente:

En ella se configuran los bloques que se va a utilizar el programa del PLC y el lenguaje por defecto en el que se va a realizar la programación. Siempre se debe seleccionar el bloque OB1. Éste se corresponde con la tarea cíclica. Se pueden utilizar otros bloques que tienen una función específica, como programas sincronizados con el tiempo.

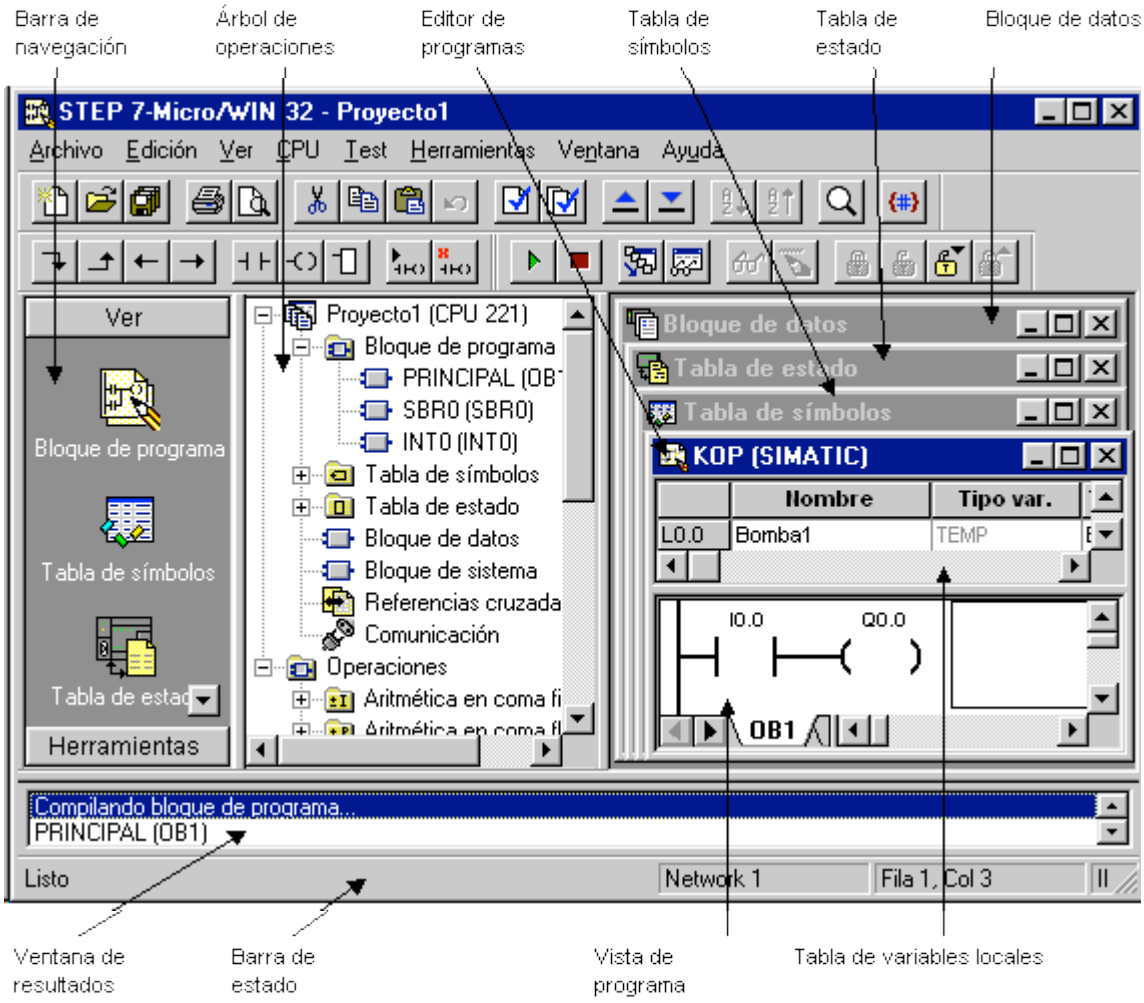
Seleccionar KOP como lenguaje por defecto para programar. KOP se corresponde con el lenguaje de contactos. Una vez realizadas las selecciones, pulsar “Siguiente”.

3.-En la tercera pantalla del Asistente se selecciona el nombre del proyecto. Y a continuación pulsar “Finalizar”.



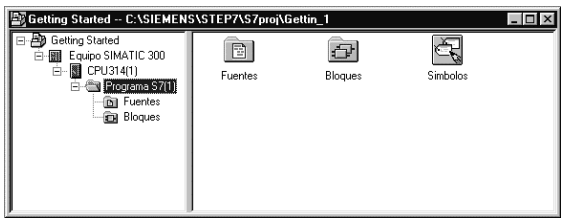
6.1.7.- Ventana principal de STEP 7.

Al cerrar el Asistente de Step 7 aparecerá el administrador Simatic con la ventana del proyecto abierta. Desde esta ventana se accede a todas las funciones de STEP 7.

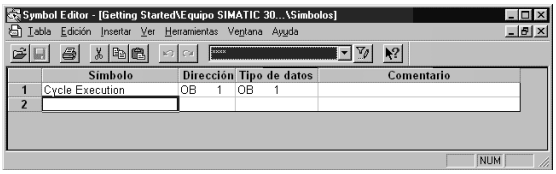


En la ventana izquierda se ve la estructura del proyecto y en la de la derecha se aparecen los objetos y carpetas que contiene la carpeta seleccionada en la ventana izquierda.

Uso del editor de símbolos



Navegue en la ventana del proyecto "Getting Started“ hasta la carpeta **Programas S7 (1)** y abra el icono **Símbolos** con un doble clic.



Por el momento nuestra tabla de símbolos contiene sólo el bloque de organización OB1.

	Símbolo	Dirección	Tipo de datos
1	Cycle Execution	OB 1	OB 1
2			

Haga clic en el símbolo **Cycle Execution** y sobrescríbalo para nuestro ejemplo con "Programa principal“.

	Símbolo	Dirección	Tipo de datos
1	Programa principal	OB 1	OB 1
2	Lámpara verde	A 4.0	BOOL

En la línea 2 introduzca "Lámpara verde“ y "A 4.0“. El tipo de datos se añade automáticamente.

Comentario

Para introducir el comentario del símbolo haga clic en la columna 'Comentario' de la línea 1 ó 2. Al finalizar una línea pulse la tecla E**ntrar** para insertar otra.

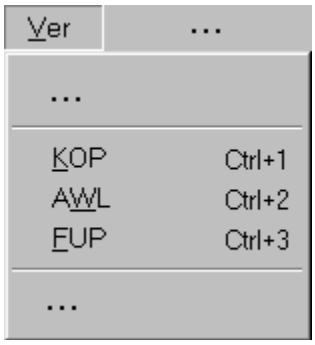
	Símbolo	Dirección	Tipo de datos
1	Programa principal	OB 1	OB 1
2	Lámpara verde	A 4.0	BOOL
3	Lámpara roja	A 4.1	BOOL

En la línea 3 introduzca "Lámpara roja" y "A 4.1“ y confirme con **Entrar**.

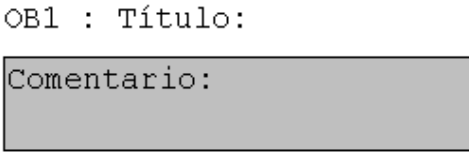
6.1.8.- Programar el OB1 en KOP.

A continuación vamos a programar una conexión en serie, una conexión en paralelo y la operación 'Flip-flop de activación/desactivación' en KOP (esquema de contactos).

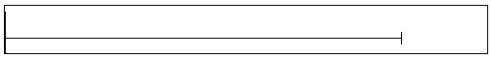
Programar una conexión en serie en KOP



Si no está ajustado el lenguaje de programación KOP, ajústelo eligiendo el comando de menú **Ver > KOP**.

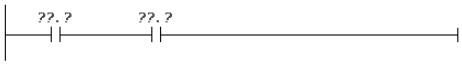


Haga clic en el área **Título** del OB1 e introduzca, p.ej., "Programa principal. Se ejecuta cíclicamente".



Seleccione un circuito para el primer elemento.

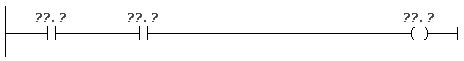
Haga clic en el botón que representa un contacto normalmente abierto en la barra de herramientas e insértelo.



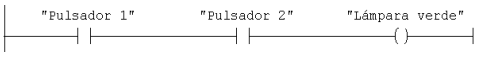
Repita el procedimiento e inserte un segundo contacto abierto.

Inserte una bobina al final del circuito.

Para completar la conexión en serie



sólo falta asignar las direcciones a los contactos normalmente abiertos y a la bobina.



Haga clic en **??.?** e introduzca el nombre simbólico "Pulsador 1" (no olvide las comillas).
Confirme con **Entrar**. Repita el proceso para asignar los demás nombres.

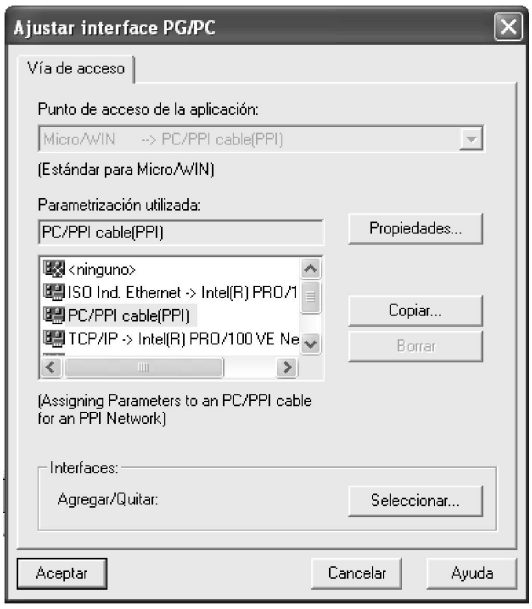
Con este botón insertaremos otro segmento.

Con este botón insertaremos una rama paralela.

6.1.9.- Configurar comunicación PG/PC.

Para configurar los parámetros de la comunicación entre el PC y la CPU habrá que seguir estos pasos:

1. En la ventana de STEP 7, haga click en el icono “Comunicación” o elija el comando de menú Ver > Comunicación. Aparecerá el cuadro de dialogo “Configurar comunicación”.



2. En el cuadro de dialogo "Configurar comunicación", haga doble click en el icono del cable PC/PPI. Aparecerá el cuadro de dialogo "Ajustar interface PG/PC".

3. Haga click en el botón "Propiedades" para acceder al cuadro de dialogo donde se visualizan las propiedades del interface. La velocidad de transferencia debe estar ajustada a 9.600 bit/s y el puerto de comunicaciones a seleccionar será el 2.

6.1.10.- Cargar y ejecutar el programa en el PLC.

En la pantalla de edición, seleccione Sistema de destino -> Cargar. Previamente hay que asegurarse que el autómata esta con la llave en la posición RUN-P. Para ejecutar el programa pulsen en pantalla el botón "Ejecutar" o cambien la posición de la llave a RUN.

6.1.11.-Subrutinas y rutinas de interrupción.

SUBRUTINAS

Las subrutinas facilitan la estructura desprograma. Las operaciones utilizadas en el programa principal determinan la ejecución de la subrutina en cuestión.

Cuando el programa principal llama a una subrutina para que esta se ejecute, la subrutina procesa su programa hasta el final, después el sistema retorna el control al segmento de programa principal desde donde se llamó a la subrutina.

Las subrutinas se utilizan para estructurar o dividir el programa en bloques más pequeños, por lo tanto, más fáciles de gestionar.

Para utilizar una subrutina en el programa principal es preciso realizar tres tareas:

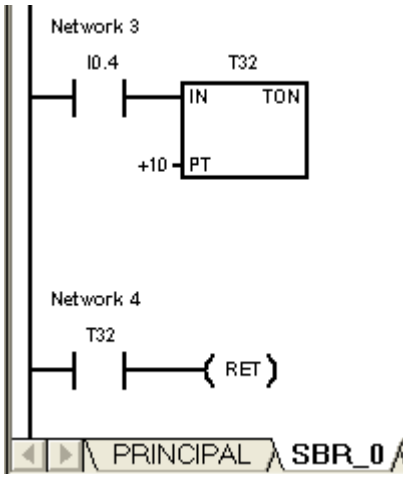
- Crear la subrutina.
- Definir los parámetros en la tabla de variables de la subrutina.
- Llamar a la subrutina desde la unidad de organización del programa en cuestión (por ejemplo, exprograma principal OB1 o desde otra subrutina).

Crear una subrutina:

A continuación se explican los pasos para crear una subrutina:

- En el menú Edición, elija los comandos Insertar > Subrutina.
- En la ventana del editor de programas, haga click con el botón derecho del ratón y elija el comando Insertar > subrutina del menú emergente.

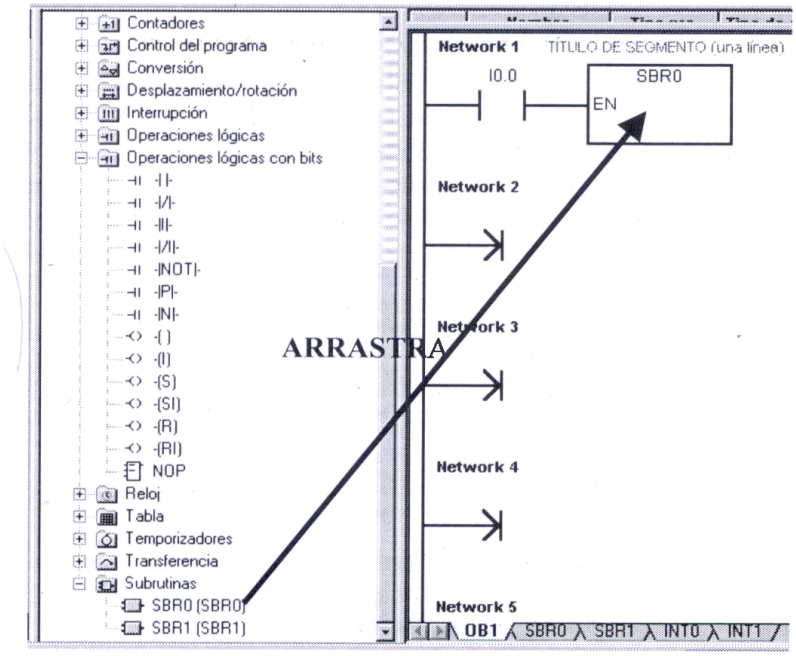
El editor de programas cambia de la anterior unidad de organización del programa visualizada a la nueva subrutina. En el borde inferior del editor de programas aparece una nueva ficha correspondiente a la nueva ficha correspondiente a la nueva subrutina.



Ahora se puede trabajar en la nueva subrutina, o bien retornar a la unidad de organización del programa donde se encontraba previamente.

Llamar a una subrutina:

Para llamar a una subrutina, desde el programa principal OB1 simplemente hay que arrastrar el icono subrutina (SBR0 o SBR1 dependiendo de el número de subrutina) dentro del programa OB! En el momento que queramos que la subrutina empiece a trabajar.



RUTINAS DE INTERRUPCIÓN

Antes de poder llamar a una rutina de interrupción es preciso establecer un enlace entre el evento que provoca la interrupción y la parte de programa que se desee ejecutar cuando se presente el evento asociado.

La operación Asociar interrupción (ATCH) sirve para asignar el evento de interrupción a una parte del programa. También es posible asociar varios eventos de interrupción a una única rutina de interrupción. Por el contrario, no se puede asociar un evento a distintas rutinas de interrupción.

Cuando se produce un evento estando habilitadas las interrupciones, se ejecuta únicamente la última rutina de interrupción asociada a dicho evento.

El procesamiento de interrupciones permite reaccionar rápidamente ante determinados eventos internos o externos. Las rutinas de interrupción se deben estructurar de forma que una vez ejecutadas determinadas tareas devuelvan el control al programa principal.

Para ello es conveniente crear rutinas de interrupción cortas con indicaciones precisas, de manera que se puedan ejecutar rápidamente sin interrumpir otros procesos durante períodos demasiado largos. Si no se observan estas medidas, es posible que se produzcan estados imprevistos que podrían afectar a los equipos controlados por el programa principal. Al utilizar interrupciones, conviene atenerse al lema de ”cuanto más breve, mejor”.

Cuando se asocia un evento a una rutina de interrupción, se habilita automáticamente el evento. Si se inhiben todos los eventos de interrupción, entonces cada vez que se presente la interrupción, se pondrá en cola de espera hasta que las interrupciones se habiliten de nuevo.

También es posible inhibir ciertos eventos de interrupción, eliminando la asociación entre evento y la correspondiente rutina mediante la operación DTCH. Esta operación retorna la interrupción a un estado inactivo o ignorado.

Las operaciones Inhibir todos los eventos de interrupción (DISI), Habilitar todos los eventos de interrupción
(ENI), Definir modo para contador rápido (HDEF) y Finalizar programa principal (END) no se pueden utilizar
en las rutinas de interrupción.

Crear una rutina de interrupción:

Para crear una rutina de interrupción, pulse el botón derecho del ratón en cualquier parte del editor KOP y seleccione Insertar > Interrupción. En ese momento se crea una ficha en el lado inferior del editor KOP, indicando la rutina de interrupción insertada.

En un programa se permiten 128 rutinas de interrupción como máximo. La CPU procesa las interrupciones según su prioridad y después en el orden que aparezcan. Solo se ejecutará una rutina de interrupción a la vez. Las interrupciones que se presenten mientras se está ejecutando otra interrupción se ponen en cola de espera para ser procesadas.

Desde una rutina de interrupción se puede llamar a un nivel de anidado de subrutinas. Los acumuladores y la pila lógica son compartidos por la rutina de interrupción y por la subrutina invocada.

Puesto que las interrupciones pueden afectar a la lógica de los contactos, las bobinas y los acumuladores, el sistema almacena la pila lógica, los acumuladores y las marcas especiales (SM) que indican el estado de los acumuladores y las operaciones, volviéndolos a cargar posteriormente. De este modo se evitan perturbaciones en el programa principal causadas por derivaciones a rutinas de interrupción o desde ellas.

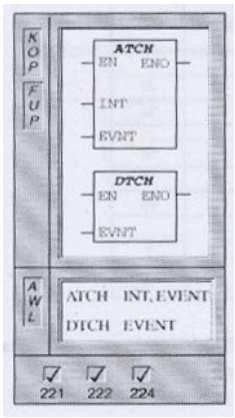
A continuación se muestra una tabla que contiene todos los eventos de interrupción junto con su descripción y las CPU`s soportadas:

Evento	Descripción		CPU 221 CPU 222	CPU 224	CPU 224XP, CPU224XPsi CPU 226
0	I0.0	Flanco positivo	Sí	Sí	Sí
1	I0.0	Flanco negativo	Sí	Sí	Sí
2	I0.1	Flanco positivo	Sí	Sí	Sí
3	I0.1	Flanco negativo	Sí	Sí	Sí
4	I0.2	Flanco positivo	Sí	Sí	Sí
5	I0.2	Flanco negativo	Sí	Sí	Sí
6	I0.3	Flanco positivo	Sí	Sí	Sí
7	I0.2	Flanco negativo	Sí	Sí	Sí
8	Puerto 0	Recibir carácter	Sí	Sí	Sí
9	Puerto 0	Transmisión finalizada	Sí	Sí	Sí
10	Interrupción temporizada 0	SMB34	Sí	Sí	Sí
11	Interrupción temporizada 1	SMB35	Sí	Sí	Sí
12	HSC0	CV=PV (valor actual = valor predeterminado)	Sí	Sí	Sí
13	HSC1	CV=PV (valor actual = valor predeterminado)		Sí	Sí
14	HSC1	Cambio de sentido		Sí	Sí
15	HSC1	Puesta a 0 externa		Sí	Sí
16	HSC2	CV=PV (valor actual = valor predeterminado)		Sí	Sí
17	HSC2	Cambio de sentido		Sí	Sí
18	HSC2	Puesta a 0 externa		Sí	Sí
19	PLS0	Interrupción Valor de contaje de impulsos PTO	Sí	Sí	Sí
20	PLS1	Interrupción Valor de contaje de impulsos PTO	Sí	Sí	Sí
21	Temporizador T32	Interrupción CT=PT	Sí	Sí	Sí
22	Temporizador T96	Interrupción CT=PT	Sí	Sí	Sí
23	Puerto 0	Recepción de mensajes finalizada	Sí	Sí	Sí
24	Puerto 1	Recepción de mensajes finalizada			Sí
25	Puerto 1	Recibir carácter			Sí
26	Puerto 1	Transmisión finalizada			Sí
27	HSC0	Cambio de sentido	Sí	Sí	Sí
28	HSC0	Puesta a 0 externa	Sí	Sí	Sí
29	HSC4	CV=PV (valor actual = valor predeterminado)	Sí	Sí	Sí
30	HSC4	Cambio de sentido	Sí	Sí	Sí
31	HSC4	Puesta a 0 externa	Sí	Sí	Sí
32	HSC3	CV=PV (valor actual = valor predeterminado)	Sí	Sí	Sí
33	HSC5	CV=PV (valor actual = valor predeterminado)	Sí	Sí	Sí

Asociar/Desasociar interrupciones:

La operación Asociar interrupción asocia el número de una rutina de interrupción (INT) a un evento de interrupción (EVENT), habilitando así el mismo.

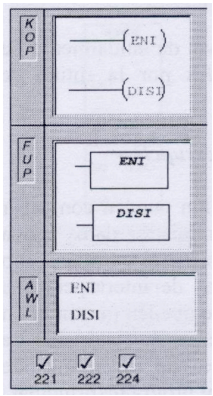
La operación Desasociar interrupción desasocia un evento0 de interrupción (EVENT) de todas las rutinas de interrupción, deshabilitando así este evento.



Habilitar/Inhibir todos los eventos de interrupción:

La operación habilitar todos los eventos de interrupción habilita la ejecución de todos los eventos de interrupción asociados.

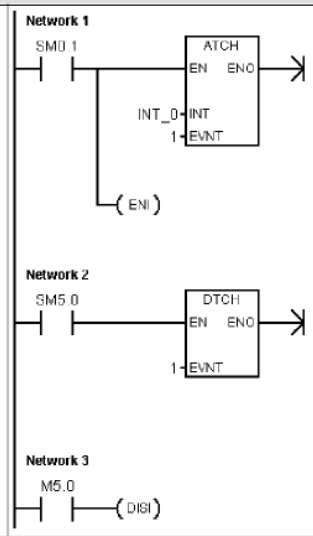
La operación Inhibir todos los eventos de interrupción deshabilita la ejecución de todos los eventos asociados.

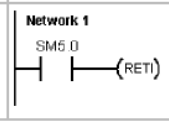


Cuando la CPU pasa a modo RUN, las interrupciones se inhiben. Estando en modo RUN, se pueden habilitar todos los eventos de interrupción con la operación global ENI. La operación DISI permite poner las interrupciones en cola de espera, pero no llamar a ninguna rutina de interrupción.

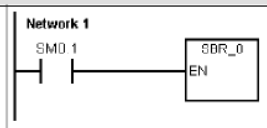
Ejemplos de rutinas de interrupción.

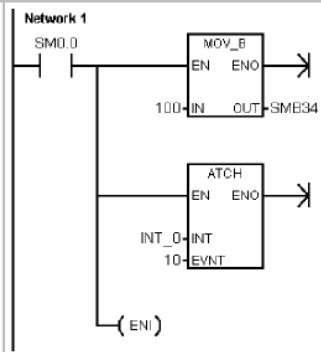
Ejemplo: Operaciones de interrupción

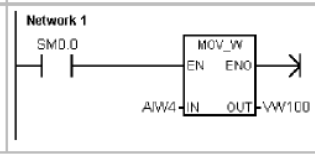
PRINCIPAL		Network 1 //En el primer ciclo:
		//1º Definir la rutina de interrupción INT_0 para que
		// sea una interrupción de flanco negativo para I0.0
		//2º Habilitar todos los eventos de interrupción.
		LD SM0.1
		ATCH INT_0, 1
		ENI
		Network 2 //Si se detecta un error de E/S,
		//inhibir la interrupción Flanco positivo en I0.0.
		//Este segmento es opcional.
		LD SM5.0
		DTCH 1
		Network 3 //Si M5.0 está activada,
		//inhibir todas las interrupciones.
		LD M5.0
		DISI

INT 0		Network 1 //Rutina de interrupción de flanco negativo en I0.0: //Retorno condicionado debido a un error de E/S. LD SM5.0 CRETI

Ejemplo: Interrupción temporizada para leer el valor de una entrada analógica

PRINCIPAL		Network 1 //Llamar a la subrutina 0 en el primer ciclo. LD SM0.1 CALL SBR_0

SBR 0		Network 1 //1º Ajustar de 0 a 100 ms el intervalo de tiempo de la
		//interrupción temporizada.
		//2º Asociar la interrupción temporizada 0 (evento 10) a
		INT_0.
		//3º Habilitar todos los eventos de interrupción.
		LD SM0.0
		MOVB 100, SMB34
		ATCH INT_0, 10
		ENI

INT 0		Network 1 //Leer el valor de AIW4 cada 100 ms. LD SM0.0 MOVW AIW4, VW100

6.1.12.- Áreas de memoria y funciones del S7-200.

Descripción	CPU 221	CPU 222	CPU 224	CPU 224XP, CPU 224XPsi	CPU 226
Tamaño del programa de usuario con edición en runtime sin edición en runtime	 4096 bytes 4096 bytes	 4096 bytes 4096 bytes	 8192 bytes 12288 bytes	 12288 bytes 16384 bytes	 16384 bytes 24576 bytes
Tamaño de los datos de usuario	2048 bytes	2048 bytes	8192 bytes	10240 bytes	10240 bytes
Imagen de proceso de las entradas	I0.0 a I15.7	I0.0 a I15.7	I0.0 a I15.7	I0.0 a I15.7	I0.0 a I15.7
Imagen de proceso de las salidas	Q0.0 a Q15.7	Q0.0 a Q15.7	Q0.0 a Q15.7	Q0.0 a Q15.7	Q0.0 a Q15.7
Entradas analógicas (sólo lectura)	AIW0 a AIW30	AIW0 a AIW30	AIW0 a AIW62	AIW0 a AIW62	AIW0 a AIW62
Salidas analógicas (sólo escritura)	AQW0 a AQW30	AQW0 a AQW30	AQW0 a AQW62	AQW0 a AQW62	AQW0 a AQW62
Memoria de variables (V)	VB0 a VB2047	VB0 a VB2047	VB0 a VB8191	VB0 a VB10239	VB0 a VB10239
Memoria local (L)[1]	LB0 a LB63	LB0 a LB63	LB0 a LB63	LB0 a LB63	LB0 a LB63
Área de marcas (M)	M0.0 a M31.7	M0.0 a M31.7	M0.0 a M31.7	M0.0 a M31.7	M0.0 a M31.7
Marcas especiales (SM)	SM0.0 a SM179.7	SM0.0 a SM299.7	SM0.0 a SM549.7	SM0.0 a SM549.7	SM0.0 a SM549.7
Sólo lectura	SM0.0 a SM29.7	SM0.0 a SM29.7	SM0.0 a SM29.7	SM0.0 a SM29.7	SM0.0 a SM29.7
Temporizadores	256 (T0 a T255)	256 (T0 a T255)	256 (T0 a T255)	256 (T0 a T255)	256 (T0 a T255)
Retardo a la conexión con memoria 1 ms	T0, T64	T0, T64	T0, T64	T0, T64	T0, T64
10 ms	T1 a T4 y T65 a T68	T1 a T4 y T65 a T68	T1 a T4 y T65 a T68	T1 a T4 y T65 a T68	T1 a T4 y T65 a T68
100 ms	T5 a T31 y T69 a T95	T5 a T31 y T69 a T95	T5 a T31 y T69 a T95	T5 a T31 y T69 a T95	T5 a T31 y T69 a T95
Retardo a la conexión/desconexión 1 ms	T32, T96	T32, T96	T32, T96	T32, T96	T32, T96
10 ms	T33 a T36 y T97 a T100	T33 a T36 y T97 a T100	T33 a T36 y T97 a T100	T33 a T36 y T97 a T100	T33 a T36 y T97 a T100
100 ms	T37 a T63 y T101 a T255	T37 a T63 y T101 a T255	T37 a T63 y T101 a T255	T37 a T63 y T101 a T255	T37 a T63 y T101 a T255
Contadores	C0 a C255	C0 a C255	C0 a C255	C0 a C255	C0 a C255
Contadores rápidos	HC0 a HC5	HC0 a HC5	HC0 a HC5	HC0 a HC5	HC0 a HC5
Relés de control secuencial (S)	S0.0 a S31.7	S0.0 a S31.7	S0.0 a S31.7	S0.0 a S31.7	S0.0 a S31.7
Acumuladores	AC0 a AC3	AC0 a AC3	AC0 a AC3	AC0 a AC3	AC0 a AC3
Saltos a metas	0 a 255	0 a 255	0 a 255	0 a 255	0 a 255
Llamadas a subrutinas	0 a 63	0 a 63	0 a 63	0 a 63	0 a 127
Rutinas de interrupción	0 a 127	0 a 127	0 a 127	0 a 127	0 a 127
Detectar flanco positivo/negativo	256	256	256	256	256
Lazos PID	0 a 7	0 a 7	0 a 7	0 a 7	0 a 7
Puertos	Puerto 0	Puerto 0	Puerto 0	Puerto 0, puerto 1	Puerto 0, puerto 1

1 STEP 7-Micro/WIN (versión 3.0 o posterior) reserva LB60 a LB63.

6.2.- Programación del robot RH-5AH.

El lenguaje de programación usado para la programación del robot es el MELFA-BASIC IV, a continuación se explica tanto el formato y la sintaxis como la configuración y parámetros de funciones y comandos.

La programación del robot se realizará desde el teaching pendant introduciendo por teclado el código del programa a la vez que se observa por el display.

6.2.1.- Introducción a MELFA BASIC IV.

1. Nombre del programa

El nombre del programa puede ser especificado usando 12 caracteres. Sin embargo el panel de operaciones solo puede mostrar 4 caracteres por lo que se recomienda usar solo 4 caracteres para nombra al programa. Los caracteres que pueden ser usados son los siguientes:

Clase	Caracteres usados
Caracteres alfabéticos	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Caracteres numéricos	0 1 2 3 4 5 6 7 8 9

Si un programa ha sido especificado con más de 4 caracteres, el programa no podrá ser seleccionado desde el panel de operaciones. Se aconseja describir los programas por números. Si un programa ejecuta un sub-programa por medio de la instrucción CALLP, se podrían usar más de 4 caracteres para nombrar dicho sub-programa.

2. Línea de comandos

Ejemplo de cómo escribir una línea:

10	MOV	P1	WTH M OUT(17)=1
1)	2)	3)	4)

1)Línea No.	Numero que determina el orden de ejecución dentro del programa. Las líneas de código se van ejecutando en orden ascendente.
2)Palabra comando	Instrucción que específica al robot que movimiento o tarea realizar.
3)Datos	Variables y datos numéricos necesarios para cada instrucción.
4)Comando adicional	Código necesario cuando se añaden más tareas al robot.

3. Variables

Los siguientes tipos de variables pueden ser usadas en un programa:

- Variable de usuario: Determinadas por su nombre de variable y su uso.
- Variable de control de usuario: Estas variables pueden ser leídas o escritas en el programa.
- Variable de control de sistema: Estas variables solo pueden ser leídas en un programa.
- Variables del sistema: Son variables predeterminadas tanto el nombre como el valor.

Cada variable esta categorizada dentro de las siguientes clases:

Position variable	El nombre de la variable comienza por P y contiene el valor de las coordenadas ortogonales del robot.
Joint variable	El nombre de la variable comienza por J y contiene el valor del ángulo de las articulaciones del robot.
Arithmetic variable	El nombre de la variable comienza por M y contiene un valor numérico (entero, real, etc.).
Character variable	Al final del nombre de la variable se le añade un $, contiene una cadena de caracteres.

4. Sentencia

Una sentencia es la minima unidad que configura un programa, y esta formada por una palabra de comando y un dato asociado al comando.
Ejemplo)

MOV	P1
Palabra de comando	Dato

5. Comandos adicionales

Las palabras de comando pueden unirse mediante la palabra WTH, pero esta limitado a solo comandos de movimiento. Esto permite que algunos comandos sean ejecutados en paralelo junto a un comando de movimiento.
Ejemplo)

MOV P1	WTH	M OUT (17)=1
Comando	Palabra unión	Comando adicional

6.2.2.- Variables de posición.

La sintaxis correcta para una variable de posición se muestra a continuación:

(100,	100,	300,	180,	0,	180,	0,	0)	(7,	0)
1	2	3	4	5	6	7	8	9	10

Número	Descripción
1	Eje X valores de coordenadas para la pinza
2	Eje Y valores de coordenadas para la pinza
3	Eje Z valores de coordenadas para la pinza
4	Eje A (postura de la pinza del robot)
5	Eje B (postura de los ejes del robot)
6	Eje C (postura de los eje del robot)
7	Eje L1 (eje adicional 1)
8	Eje L2 (eje adicional 2)
9	Flag de estructura 1
10	Flag de estructura 2

Ejemplo) P1=(300, 100, 400, 180, 0, 180, 0, 0)(7, 0)

Es posible referirse a un dato de coordenadas en particular de una variable de posición. En este caso añadir "." Y el nombre de el eje de coordenadas después del nombre de la variable.
Ejemplo)

P1.X El valor devuelto será la coordenada X de la posición P1

Tipos de datos y significado de coordenadas, postura y ejes adicionales:

[Formato] X, Y, Z, A, B, C, L1, L2
[Significado] X, Y, Z: Datos de coordenadas. Es la posición del fin de la mano del robot en coordenadas X Y Z.
A, B, C: Datos de postura. Es el ángulo de la postura. (las unidades son milímetros o grados).
L1, L2: Datos de los ejes adicionales. Estas son las coordenadas para ejes adicionales. (las unidades son milímetros o grados).

Tipo y significado de los datos del Flag de la estructura:

[Formato] FL1, FL2
[Significado] FL1: Datos sobre la postura
Ejemplo)

7 = & B 0 0 0 0 0 1 1 1 (Número en binario)
1/0 = Volteado/no volteado
1/0 = Encima/debajo
1/0 = Izquierda/derecha

Queda decir que no es necesario escribir los datos de coordenadas y postura para todos los 8 ejes. Si se omiten los datos de algunos ejes, el robot lo procesará como no definidos.

6.2.3.- Variables de entrada / Salida.

Los siguientes tipos de variables de entrada/salida pueden usarse. Estas vienen provistas de antemano por las variables de estado del robot.

Nombre	Explicación
M_IN	Para señales de bit de entrada
M_INB	Para señales de byte de entrada
M_INW	Para señales de palabra de entrada
M_OUT	Para señales de bit de salida
M_OUTB	Para señales de byte de salida
M_OUTW	Para señales de palabra de salida
M_DIN	Para registros de entrada del CC-link
M_DOUT	Para registros de salida del CC-link

M_IN/M_INB/M_INW

[Función] Devuelve el valor de la señal de entrada.
M_IN: Devuelve un bit.
M_INB: Devuelve un byte (8bits).
M_INW: Devuelve una palabra (16bits).

[Formato]

<Variable numérica> = M_IN(<Expresión numérica>)
<Variable numérica> = M_INB(<Expresión numérica>)
<Variable numérica> = M_INW(<Expresión numérica>)

[Programa de referencia]

10 M1 = M_IN(0)	M1 contendrá el valor de la señal de entrada 0.
20 M2 = MINB(0)	M2 contendrá la información de 8 bits que empieza por la señal de entrada 0.
30 M3 = M_INB(3)	M3 contendrá la información de 8 bits que empiezan por la señal de entrada 3.
40 M4 = M_INW(5)	M4 contendrá la información de 16 bits que empiezan por la señal de entrada 5.

[Explicación]

(1) Devuelve el estado de la señal de entrada.
(2) M_INB y M_INW devolverá la información de 8 o 16 bit empezando por el número especificado.
(3) Aunque el número de señal puede ser tan largo como 32767, solo los números de señal que correspondan con el hardware devolverá un valor valido.
(4) Esta variable solo lee los datos.

M_OUT/M_OUTB/M_OUTW

[Función] Escribe la señal de salida externa.

[Formato]

M_OUT (<Expresión numérica>) = <Variable numérica>
M_OUTB (<Expresión numérica>) = <Variable numérica>
M_OUTW (<Expresión numérica>) = <Variable numérica>

[Programa de referencia]

10 M_OUT(2)=1 Pone a 1 la señal de salida 2.
20 M_OUTB(2)=&HFF Pone en ON 8-bits desde la señal de salida 2.
30 M_OUTW(2)=&HFFFF Pone en ON 16-bits desde la señal de salida

[Explicación]

(1) Se utiliza cuando se escriben señales de salida.
(2) A partir del numero 900 se usaran para las señales de entrada salida de la mano.
(3) Números desde el 6000 en adelante están asignados o hacen referencia a los datos del CC-Link.

6.2.4.- Funciones.

OPEN (Open)

[Función] Abre un archivo o una línea de comunicación.

[Formato]

OPEN "<Descripción del archivo"> AS [#] <No. De archivo>

[Terminología]

<Descripción de archivo>: Describe el nombre del archivo o la línea de comunicación. Para usar una línea de comunicación, establecer "<Nombre de archivo de la línea de comunicación>". Cuando no se use para una línea de comunicación, establecer "<Nombre de archivo>".

Tipo	Nombre	Método de acceso
Archivo	Descrito con 16 caracteres o menos	INPUT, PRINT, APPEND
Línea de comunicación	COM1: Estándar RS-232C COM2: El conjunto de parámetros en "COMDEV" ….. COM8: El conjunto de parámetros en "COMDEV"	Omitido = solo Random mode

Designando el método de acceso al archivo:

- Omitido = random mode. Puede ser omitido cuando se trate de una línea de comunicación.
- INPUT = modo de entrada. Lee desde un archivo.
- OUTPUT = modo de salida. Crea un nuevo archivo de salida.
- APPEND = modo de salida. Añade la salida al final de un archivo existente.

[Programa de referencia]

```
10 OPEN "COM1:" AS #1        Abre una línea estándar RS-232C como 1.
30 PRINT #1, P_CURR          Envía la posición actual a un dispositivo
                             externo.
40 INPUT #1, M1, M2, M3      Recibe de un dispositivo externo las
                             variables M1, M2 y M3.
50 P_01.X = M1
60 P_01.Y = M2
70 P_01.C = M3               Copia a los datos globales.
80 CLOSE                     Cierra los archivos abiertos.
90 END
```

[Explicación]

(1) Abre un archivo especificado en <Nombre de archivo> usando el numero de archivo. Se usa ese número para leer o escribir en el archivo.
(2) Una línea de comunicación es tratada como un archivo.

CLOSE (Close)

[Función] Cierra el archivo designado, incluyendo líneas de comunicación.

[Formato]

CLOSE [#] <Número de archivo>

[Terminología]

<Número de archivo>: Especifica el número de archivo que hay que cerrar. Solo se admiten constantes numéricas. Si se omite, todos los archivos abiertos serán cerrados.

[Programa de referencia]

```
10 OPEN "COM1:" AS #1        Abre el "COM1:" como archivo número 1.
                .
                .
110 CLOSE #1                 Cierra el archivo número 1. "COM1:".
                .
                .
200 CLOSE                    Cierra todos los archivos abiertos.
```

MOV (Move)

[Función] Operación de interpolación de articulaciones, mueve desde la actual posición a la posición destino.

[Formato]

MOV <Posición destino>
TYPE <Constante 1>, <Constante 2> [Condiciones añadidas]

[Terminología]

<Posición destino>	Posición final de la operación de interpolación.
<Constante 1>	1/0: Rodeo/atajo. El valor por defecto es 1(rodeo).
<Constante 2>	Invalido (Se especifica con 0)
<Condiciones añadidas>	Las sentencias WTH y WTHIF pueden ser usadas.

[Programa de referencia]

```
10 MOV P1 TYPE 1,0
20 MOV J1
30 MOV P3 WTH M_OUT(17)=1
40 MOV P4+P5,50.0 TYPE 0,0 WTHIF M_IN(18)=1,M_OUT(20)=1
```

[Explicación]

(1) La diferencia del ángulo de la articulación para cada eje se interpola desde el punto de partida hasta el punto final.
(2) Usando las sentencias WTH y WTHIF las señales de entrada y salida se pueden sincronizar con el movimiento.
(3)La constante numérica 1 para TYPE especifica la cantidad de posturas de interpolación.
(4) La operación de atajo especifica que la postura de interpolación entre el punto inicial y el final sea en la dirección de menos movimiento.
(5) Si la posición final está fuera del rango de movimiento, aunque la operación atajo haya sido seleccionada, el eje puede moverse con la operación rodeo en otra dirección.
(5) La constante numérica 2 para TYPE no se utiliza.
(6) Si se pausa mientras se esta ejecutando una operación MOV, al volver a continuar el robot vuelve a la posición inicial y repite la operación MOV.

MVS (Move S)

[Función] Operación de interpolación lineal de movimiento entre la posición actual y la posición de destino.

[Formato]

MVS <Posición final> [distancia]
TYPE <Constante 1>, <Constante 2> <Condición añadida>

[Terminología]

<Posición final>	Posición final de la interpolación lineal.
<Distancia>	Especifica la distancia de separación en el eje Z de la herramienta.
<Constante 1>	1/0: Rodeo/atajo. El valor por defecto es 1(rodeo).
<Constante 2>	1/0: 3-ejes XYZ/Rotación equivalente.
<Condiciones añadidas>	Las sentencias WTH y WTHIF pueden ser usadas.

[Programa de referencia]

```
10 MVS P1
20 MVS P1, 100.0 WTH M_OUT(17)=1
30 MVS P4+P5, 50.0 WTHIF M_IN(18)=1
```

[Explicación]

(1) Movimiento de interpolación lineal es un tipo de movimiento donde el robot se mueve desde el punto inicial al punto final en línea recta.
(2) La postura se interpola desde el punto inicial al punto final.
(3) Si la ejecución es pausada, cuando vuelva a continuar el robot volverá a la posición inicial mediante interpolación de articulaciones y después volverá a empezar.
(4) Si los 3-ejes XYZ son seleccionados con la constante 2, la constante 1 queda invalidada.

PRINT (Print)

[Función] Escribe datos dentro de un archivo. Todos los datos están en el formato ASCII.

[Formato]

PRINT #<Número de archivo>, [Expresión]

[Terminología]

<Número de archivo>	Es el número de archivo, descrito desde el 1 al 8.
<Expresión>	Describe una operación numérica, una posición o una cadena de caracteres.

[Programa de referencia]

10 OPEN “COM1” AS #1	Abre una línea estándar RS-323C como 1.
20 PRINT #1, “PRINT TEST”	Escribe la cadena de caracteres PRINT TEST en la salida.
30 PRINT #1, “DATA=”,DATA	Escribe la cadena de caracteres “DATA=” seguido del valor de DATA.
40 PRINT #1	Devuelve el acarreo en cuestión.
50 END	Fin del programa.

[Explicación]

(1) Si no se especifica ninguna expresión, entonces se devolverá un valor de acarreo.
(2) EL número de caracteres para cada <Expresión> está limitado a 14 caracteres. Cuando se escriba en un dispositivo externo múltiples valores se usará una coma para separarlos. Si se usa un “;” en vez de “,”, a la salida no habrán espacios en blanco entre los valores separados por “;”.
(3) Ocurrirá un error si la función PRINT es utilizada sin antes haber abierto un canal de comunicación con la función OPEN.

INPUT (Input)

[Función] Lee datos desde un archivo o una línea de comunicación. Solo podrá recibir datos escritos en caracteres ASCII.

[Formato]

INPUT #<Número de archivo>, <Dato de entrada>

[Terminología]

<Número de archivo>	Especifica del número 1 al 8 el archivo o línea de comunicación.
<Dato de entrada>	Describe el nombre de la variable donde se almacenaran los datos de entrada.

[Programa de referencia]

10 OPEN "COM1:" AS #1	Asigna una línea de comunicación RS-232C al archivo 1.
20 INPUT #1,M1	Almacena en la variable M1 los datos leídos en el archivo 1.

[Explicación]

(1) Los datos de entrada leídos desde el archivo 1 abierto por la sentencia OPEN, son sustituidos en la variable. Si la sentencia OPEN no ha sido ejecutada aparecerá un error
(2) El tipo de datos de entrada y el tipo de variable que se sustituye debe ser el mismo.
(3) Cuando se escriben variables con múltiples nombres, utilizar una coma entre dichos nombres como delimitador.
(4) Cuando la sentencia INPUT es ejecutada el Status será "Standby for input", los datos de entrada son escritos en las variables al mismo tiempo que los datos de acarreo (CR y LF) son introducidos.
(5) Si el protocolo (en caso de puerto estándar: el parámetro "CRPC232" es cero) del puerto esta especificado para un PC es necesario añadir "PRN" a la cabeza de los datos enviados desde el PC.
(6) Si el número de elementos introducidos es mayor que el número de argumentos en la sentencia INPUT, estos serán leídos e ignorados.
(7) Al ejecutar las sentencias END o CLOSE se borraran todos los datos almacenados en el buffer.

6.2.5.-Parámetros de comunicación RS232

En la siguiente tabla se muestran los datos de configuración para la comunicación PC-robot mediante el puerto serie RS232:

DATO	VALOR
Velocidad de transmisión	9600 bps
Número de bits de datos	8 Bits
Paridad	Par
Carácter de finalización	"CR"
Número de bits para terminar un byte	2 Bits

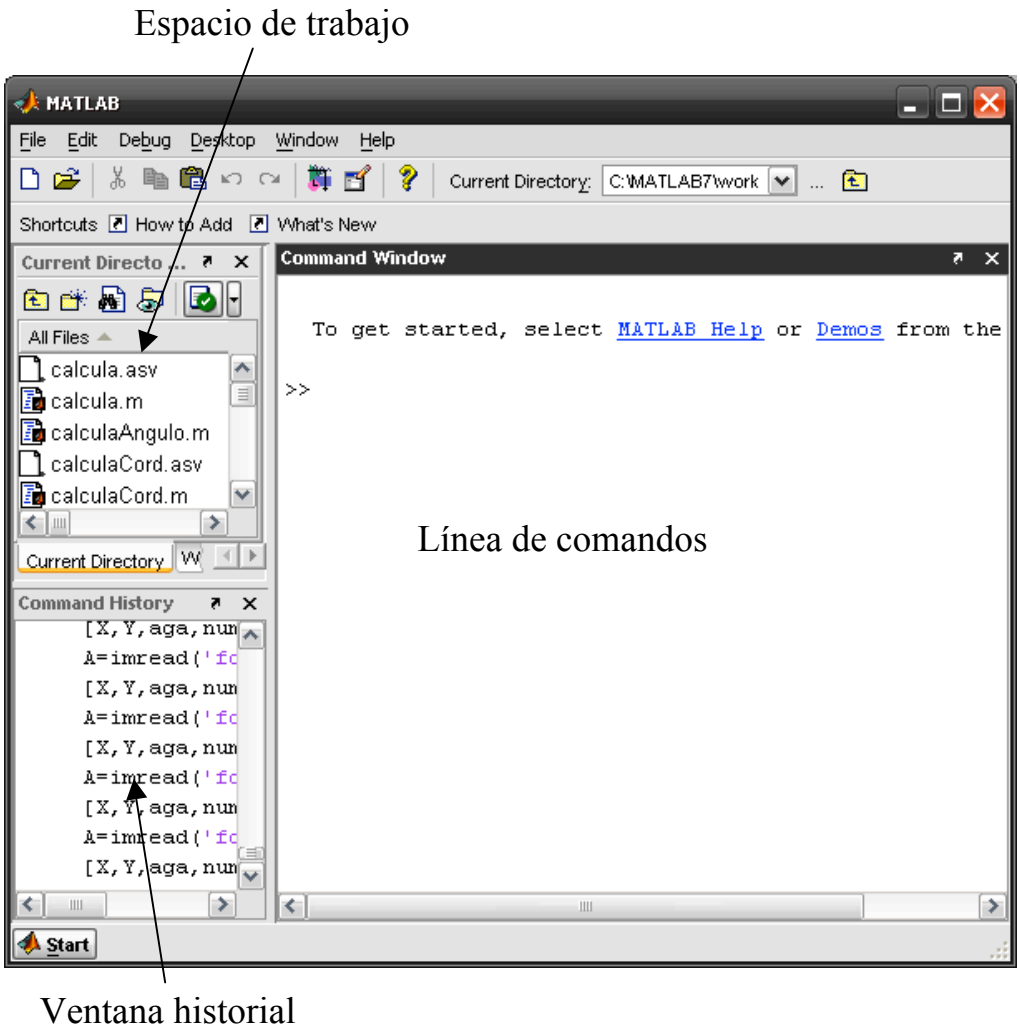
Estos datos de configuración son necesarios para poder establecer la comunicación entre el PC y la controladora del robot. Si se introducen algún dato de comunicación de forma incorrecta la controladora no comprenderá los datos recibidos por el PC y los rechazará.

En el caso de configurar correctamente la comunicación entre el PC y la controladora del robot, se crean buffers de entrada y salida, es decir, los datos que van llegando por el puerto RS232 quedan almacenados y numerados según el orden de entrada para luego procesarlos en el mismo orden en que han llegado, con la finalidad de no perder ningún dato recibido y conseguir una sincronización real. Lo mismo pasa con los datos de salida.

6.3.- Programación con MATLAB.

Matlab (abreviatura de MATrix LABoratory, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Es una potente herramienta de cálculo matemático, para la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes).



Las “toolbox” utilizadas en este proyecto son:

- Image Adquisition Toolbox
- Image Processing Toolbox

Las cuales se pueden descargar tras obtener la licencia de las páginas:

http://www.mathworks.es/products/imaq/tryit.html

http://www.mathworks.es/products/image/tryit.html

6.3.1.- Vectores y matrices.

Para definir un vector fila, basta introducir sus coordenadas entre corchetes:

```
>>v=[1 2 3]     % Vector de 3 coordenadas

v=
1 2 3
```

Si queremos declarar un vector de coordenadas equiespaciadas entre dos dadas, por ejemplo, que la primera valga 0, la última 20 y la distancia entre coordenadas sea 2, basta poner:

```
>>vect1=0:2:20

vect1 =
0 2 4 6 8 10 12 14 16 18 20
```

A las coordenadas de un vector se accede sin más que escribir el nombre del vector y, entre paréntesis, su índice:

```
>>vect2(3)

ans =
4
```

Las matrices se escriben como los vectores, pero separando las filas mediante un punto y coma; así una matriz 3x3:

```
>>M=[1 2 3;4 5 6;7 8 9]

M =
1 2 3
4 5 6
7 8 9

>>M'                % Su traspuesta (su adjunta)

ans =
1 4 7
2 5 8
3 6 9
```

A los elementos de una matriz se accede sin más que escribir el nombre de la matriz y, entre paréntesis, los respectivos índices:

```
>>M(1,3)    % Elemento en la primera fila y tercera columna de la
              matriz M

ans =
7
```

También se puede acceder a una fila o columna completas,

```
>>mat(:,2)    % Segunda columna de mat

ans =
2
5
0

>>mat(2,:)    % Su segunda fila

ans =
4 5 6
```

Existen algunas matrices definidas previamente; por ejemplo, la matriz identidad,

```
>>eye(3)    % eye se pronuncia en inglés como I

ans =
1 0 0
0 1 0
0 0 1
```

la matriz nula,

```
>>zeros(3)

ans =
0 0 0
0 0 0
0 0 0
```

o la matriz cuyos elementos valen todos 1:

```
>>ones(3)

ans =
1 1 1
1 1 1
1 1 1
```

Se puede conocer el tamaño de una matriz y la longitud de un vector:

```
>>size(mat) % Dimensiones de la matriz mat (número de filas y de
              columnas)

ans =
3 3

>>length(v) % Longitud del vector (número de coordenadas)

ans =
3
```

Las operaciones habituales entre *arrays* (suma, resta y producto escalar de vectores; suma, resta, producto y potencia de matrices) se representan con los operadores habituales:

```
>>z=v*w'  % Producto escalar (producto de matrices 1x3 por 3x1)

>>mat^2   % Matriz mat elevada al cuadrado
```

También pueden efectuarse multiplicaciones, divisiones y potencias de *arrays*, entendiéndolas como elemento a elemento. El operador utilizado para ellas es el habitual precedido por un punto; es decir:

```
>>v.*w     %Vector formado por el producto de las respectivas
              coordenadas   %ans(i)=v(i)*w(i)

>>mat.^2  % Matriz cuyos  elementos son los de mat elevados
              al cuadrado: ans(i,j)=mat(i,j)^2
```

6.3.2.- Gráficas de funciones.

MATLAB tiene un gran potencial de herramientas gráficas. Se pueden dibujar los valores de un vector frente a otro (de la misma longitud) mediante la función plot.

PLOT.- Dibuja una gráfica de dos dimensiones.

>> PLOT(X, Y, 'Propiedad', 'Propiedad') traza el vector Y versus el vector X. Si X o Y es una matriz, entonces el vector es trazado versus las filas y columnas de la matriz. La tercera entrada consiste de uno a tres caracteres que especifican un color y/o o un tipo de marcador. La lista de colores y marcadores es como sigue:

Símbolo	Color	Símbolo	Marcador
'y'	Amarillo	.	Punto
'm'	Violeta	O	Círculo
'c'	Celeste	X	Cruz x
'r'	Rojo	+	Suma
'g'	Verde	-	Sólido
'b'	Azul	*	Estrella
'w'	Blanco	:	Punteado
'k'	Negro	-.	Punto y raya
		--	A trazos

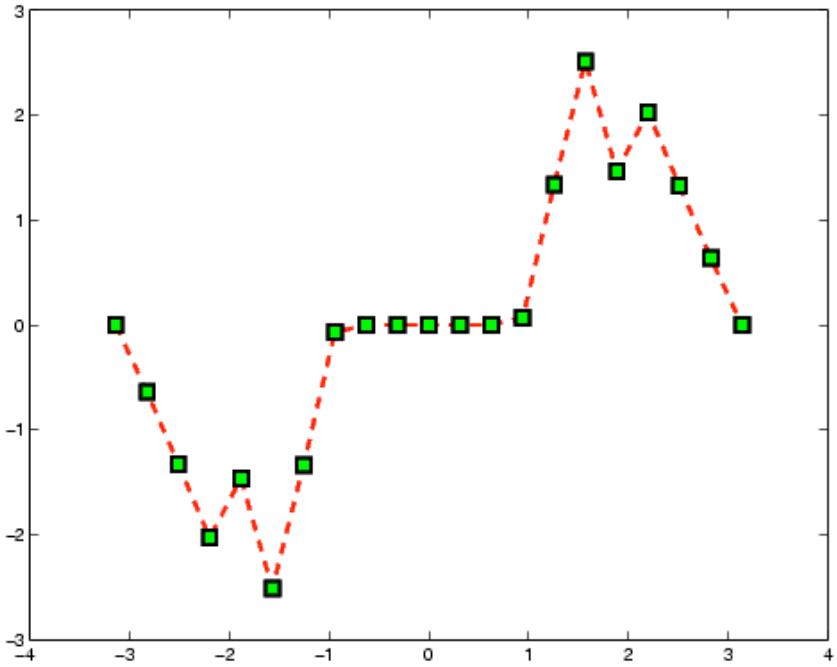
Además, también se pueden especificar otros parámetros de las gráficas, que nos ayudaran a mejorar el aspecto de las mismas:

- LineWidth: Especifica el ancho de la línea que une los puntos.
- MarkerEdgeColor: Determina el color del marcador.
- MarkerFaceColor: Especifica el color de la cara o las áreas cerradas.
- MarkerSize: Especifica el tamaño del marcador.

Veamos un ejemplo:

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...
              'MarkerEdgeColor','k',...
              'MarkerFaceColor','g',...
              'MarkerSize',10)
```

Produce esta gráfica:



Añadiendo títulos, ejes, y anotaciones.

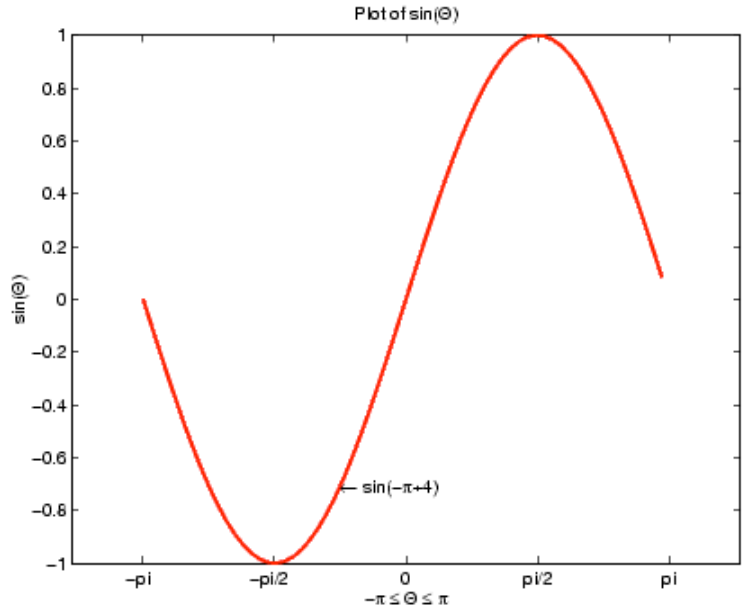
Otra cosa que puede ser importante en sus gráficos es el etiquetado. Puede darle un título al gráfico (con el comando title), etiqueta al eje x (con el comando xlabel), o al eje y (con el comandoylabel), así como incluir texto en la figura presente. Todos los comandos mencionados se emiten luego de ejecutarse el comando plot actual.

Más aún, puede incorporarse texto en el mismo gráfico mediante el comando text y el comando gtext. El primer comando requiere conocer las coordenadas donde quiere ubicar la cadena de texto. El comando es text(xcor,ycor,'textstring'). El otro comando, gtext('textstring'), no necesita saber las coordenadas exactas. Sólo mueva la mira en cruz al lugar deseado con el ratón y haga clic en la posición que quiere que se inserte el texto.

Por ejemplo, añadiendo las siguientes sentencias podremos especificar las etiquetas de los ejes y el título de la gráfica:

```
>> xlabel('-\pi \leq \Theta \leq \pi')
>> ylabel('sin(\Theta)')
>> title('Plot of sin(\Theta)')
>> text(-pi/4,sin(-pi/4),'\leftarrow sin(-\pi\div4)',...
      'HorizontalAlignment','left')
```

Obtendremos una gráfica con las etiquetas, título y texto como la gráfica que se muestra a continuación:



Otros comandos que pueden usarse con el comando plot son:

Comando	Descripción
Clf	Borra el gráfico actual, lo deja en blanco.
Figure	Abre una nueva ventana gráfica, se conserva la anterior
Close	Cierra la ventana de la figura actual
Loglog	Igual que Plot, pero los ejes se escala en Log10
Semilogx	Igual que Plot, pero el eje x se escala en Log10
Semilogy	Igual que Plot, pero el eje y se escala en Log10
grid	Agrega una grilla al gráfico

En una misma figura puede ponerse más de una línea empleando el comando subplot. El comando subplot le permite separar la figura en tantas figuras como se quiera, y ponerlas todas en una figura. Para usar este comando, Copie la siguiente línea de código e insértela en la ventana de comandos del Matlab o en un archivo-m:

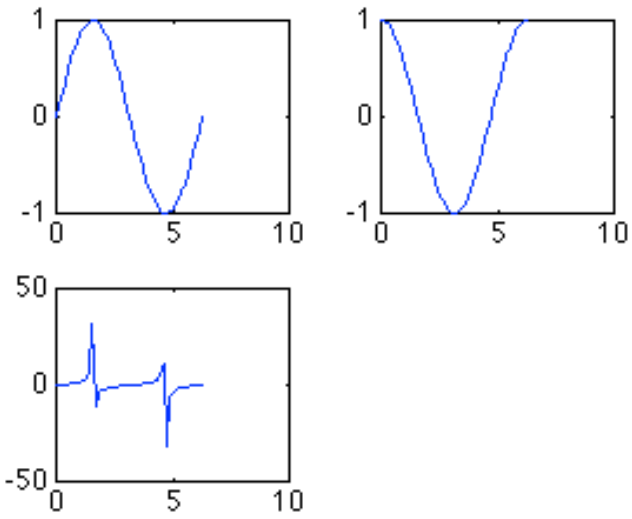
>> SUBPLOT(M, N, P) Este comando divide la figura en una matriz de m renglones y n columnas, por lo tanto crea m x n gráficos en una figura. Por ejemplo, suponga quisiera ver una senoide, un coseno, y una onda tangente graficadas en la misma figura, pero no en los mismos ejes. El siguiente archivo-m lo hará:

```
>> x = linspace(0,2*pi,50);
>> y = sin(x);
>> z = cos(x);
>> w = tan(x);

>> subplot(2,2,1)
>> plot(x,y)
>> subplot(2,2,2)
>> plot(x,z)
>> subplot(2,2,3)
>> plot(x,w)
```

No es necesario rellenar todos los gráficos m x n creados con subplot, simplemente quedaran huecos vacíos en las coordenadas que no hayan sido asignadas.

La siguiente única figura contiene las gráficas de un senoide, un coseno, y una onda tangente graficadas en la misma figura:



Agregando más juegos de parámetros al comando plot, puede graficar tantas funciones en la misma figura como se quiera. Cuando traza muchas cosas en el mismo gráfico es útil diferenciar las diferentes funciones mediante color y marcadores. Puede lograrse el mismo efecto usando los comandos hold on y hold off. Por ejemplo:

```
>> x = linspace(0,2*pi,50);
>> y = sin(x);
>> plot(x,y,'r')
>> z = cos(x);
>> hold on
>> plot(x,z,'gx')
>> hold off
```

Recuerde que siempre que use el comando hold on, todas las figuras serán generadas en un conjunto de ejes, sin borrarse la figura anterior, hasta que se aplique el comando hold off .

6.3.3.- Programación con MATLAB.

Para escribir un programa con MATLAB habrá que crear un fichero que tenga extensión .m y contenga las instrucciones. Esto se puede hacer con cualquier editor de textos, pero tiene algunas ventajas usar el editor propio de MATLAB llamado medit.

MATLAB trabaja con memoria dinámica, por lo que no es necesario declarar las variables que se van a usar. Por esta misma razón, habrá que tener especial cuidado y cerciorarse de que entre las variables del espacio de trabajo no hay ninguna que se llame igual que las de nuestro programa (proveniente, por ejemplo, de un programa previamente ejecutado en la misma sesión), porque esto podría provocar conflictos.

A menudo, es conveniente reservar memoria para las variables (por ejemplo, si se van a utilizar matrices muy grandes); para ello, basta con asignarles cualquier valor. Del mismo modo, si se está usando mucha memoria, puede ser conveniente liberar parte de ella borrando (clear) variables que no se vayan a usar más.

Un programa escrito en MATLAB admite la mayoría de las estructuras de programación al uso y su sintaxis es bastante estándar. En los siguientes ejemplos se muestra la sintaxis de algunas de estas estructuras (if, for, while,...).

Ejemplo: Calcular la suma de los n primeros términos de la sucesión 1, 2x, 3x^2, 4x^3, ...

```
>> n=input('¿Cuántos términos quieres sumar?  ');
>> x=input('Dame el valor del numero x  ');
>> suma=1;
>> for i=2:n
       suma=suma+i*x^(i-1);
>> end
>> disp('El valor pedido es')
>> disp(suma)
```

6.3.4.- Image Processing Toolbox.

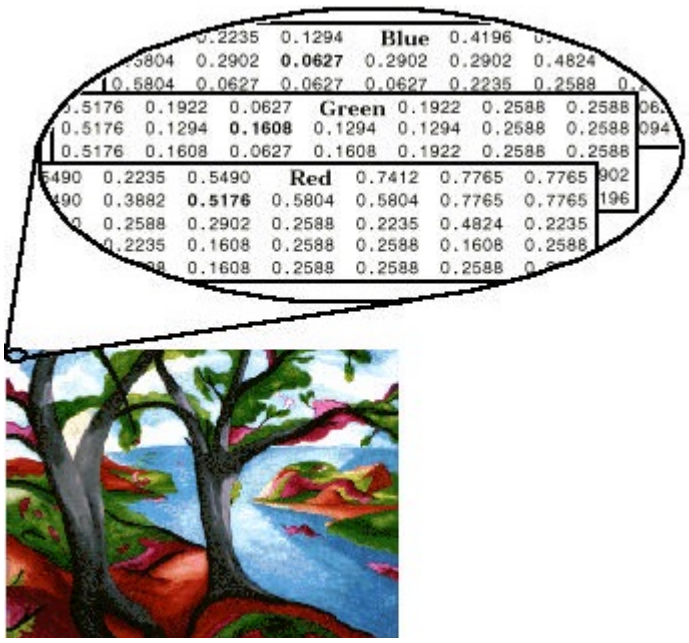
En este apartado veremos los distintos tipos de imágenes soportadas y como son representados por Matlab.

6.3.4.1.- Las imágenes en Matlab

La estructura básica de datos en Matlab es el array, es decir, un conjunto ordenado de elementos reales o complejos. El array es por tanto un objeto acertado para la representación de imágenes, que serán matrices reales con datos de color e intensidad.

Matlab representa la mayoría de las imágenes como arrays bidimensionales (matrices), donde cada elemento de la matriz representa un píxel. Por ejemplo, una imagen de 200 filas y 300 columnas será representada por Matlab como una matriz de 200x300. Algunas imágenes como las RGB requieren un array tridimensional, donde cada plano de la dimensión representa la información de intensidades en rojo, verde y azul respectivamente.

Este convenio hace que trabajar con imágenes en Matlab venga siendo como trabajar con cualquier tipo de matrices, lo que nos proporciona todas las facilidades que Matlab ofrece para manipular este tipo de datos.



6.3.4.2.- Tipos de datos

Por defecto, Matlab almacena a mayoría de los datos en arrays de tipo *double*. Los datos en este arrays son representados como números de 64 bits en coma flotante.

Todas las funciones y capacidades de Matlab trabajan con este tipo de arrays. Pero para el procesado de imágenes esta representación de los datos no siempre será la más acertada.

Supongamos una imagen 1000x1000 pixeles. Si optamos por la representación anterior esta imagen ocuparía 8 megabytes. Para reducir los requerimientos de memoria, Matlab soporta el almacenamiento de imágenes en arrays de tipo *uint8* (entero sin signo de 8 bits) y *uint16* (entero sin signo de 16 bits)

6.3.4.3.- Tipos de imágenes en Matlab

El Image Processing Toolbox de Matlab soporta 4 tipos básicos de imágenes:

- Imágenes indexadas.
- Imágenes de intensidad o en escala de grises.
- Imágenes binarias.
- Imágenes RGB.

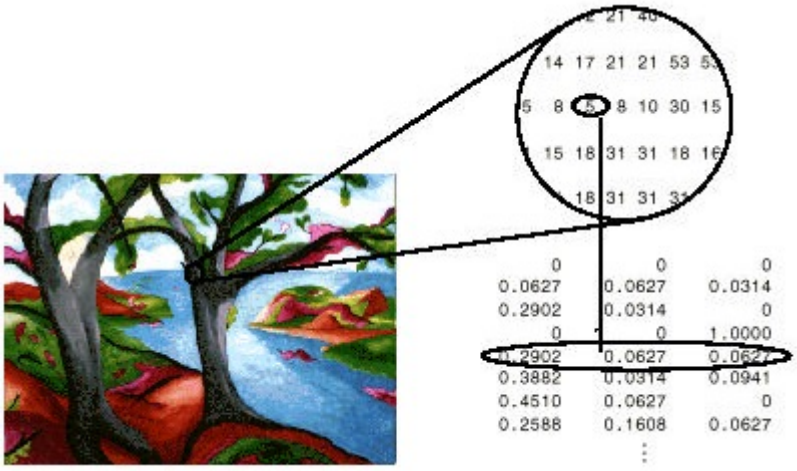
En este apartado veremos como Matlab y el Image Processing Toolbox representan cada uno de este tipo de imágenes.

6.3.4.4.- Imágenes indexadas

Una imagen indexada consiste en una matriz X (matriz imagen), y un mapa de color. El mapa de color es un array mx3 de tipo doble que contiene números reales en coma flotante de 64 bits dentro de un rango [0 1].

Cada fila del mapa representa los componentes rojo, verde y azul de una coordenada concreta. Cada elemento de la matriz X representa un píxel de la imagen, su valor indica un elemento del mapa de color.

Por lo tanto la color de cada píxel de la imagen esta determinado usando el correspondiente valor de X como un índice del mapa de color.
La siguiente figura ilustra la estructura de una imagen indexada:



Los pixeles de una imagen están representados por enteros que son punteros a colores almacenados en el mapa de color. La relación entre los valores de la matriz X y el mapa de color dependen del tipo que sea la matriz X.

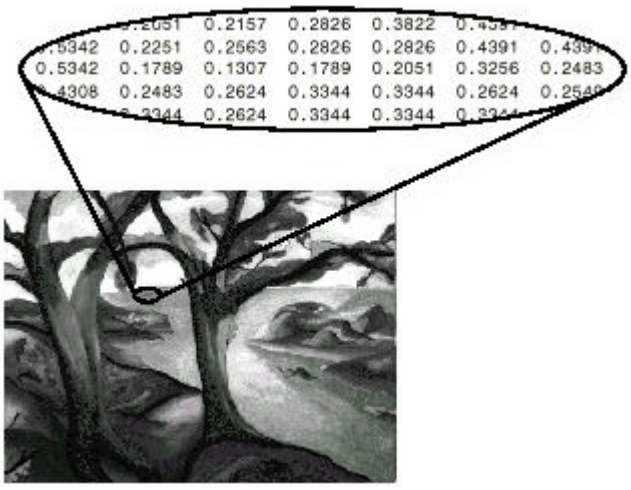
Lo más usual y lo más recomendado, en imágenes indexadas es que la matriz imagen sea de tipo uint8 y que el mapa de color sea de 256 filas. Además la mayoría de funciones de Image Processing Toolbox operan con imágenes indexadas en este formato.

6.3.4.5.- Imágenes en escala de grises

Una imagen en escala de grises es representada en Matlab por una matriz A tal que sus valores representan intensidades de gris. La matriz A puede ser de tipo double, uint8 o uint16.

El valor de intensidad 0 normalmente representa el negro y el máximo (255) normalmente representa el blanco (según si la matriz a es del tipo double, uint8 o uint16.

La figura siguiente representa una imagen en escala de grises de tipo double.



Lo más usual es encontrarse con imágenes en escala de grises representadas por matrices de tipo uint8 y por lo tanto con 256 niveles de gris.

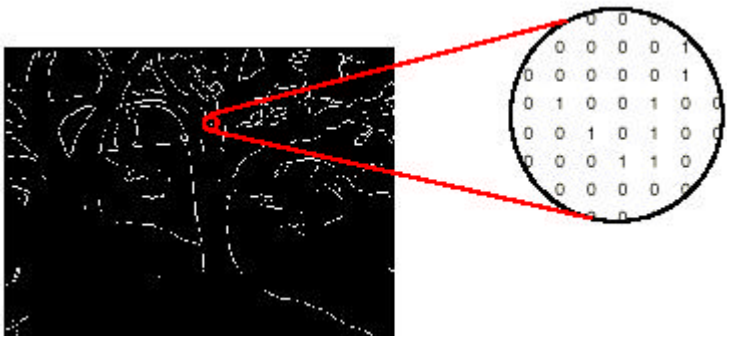
6.3.4.6.- Imágenes binarias

En una imagen binaria cada píxel puede tener solo dos valores, normalmente 0 (negro) y 1 (blanco). Una imagen binaria es almacenada como una matriz bidimensional de ceros y unos.

Las imágenes binarias pueden guardarse en arrays de tipo double o uint8. Un array de tipo uint8. Se recomienda el uso de un array de tipo uint8 antes que un array de tipo double por que el primero usa menos memoria.

Cualquier función del Image Processing Toolbox que devuelva una imagen binaria devolverá un array lógico de tipo uint8. El Image Processing Toolbox usa un flag para indicar el rango de los datos de un array de tipo uint8. Si el flag está en ON el rango será [0, 1], si el flag esta en OFF Matlab asume que el rango es [0, 255].

La figura siguiente muestra una imagen binaria.



6.3.4.7.- Imágenes RGB

Una imagen RGB también llamada imagen en color verdadero (truecolor) son almacenadas en Matlab como un array mxnx3 que definen los componentes rojo, verde y azul de cada píxel. Las imágenes RGB no usan una paleta de color, es decir un colormap, como las indexadas.

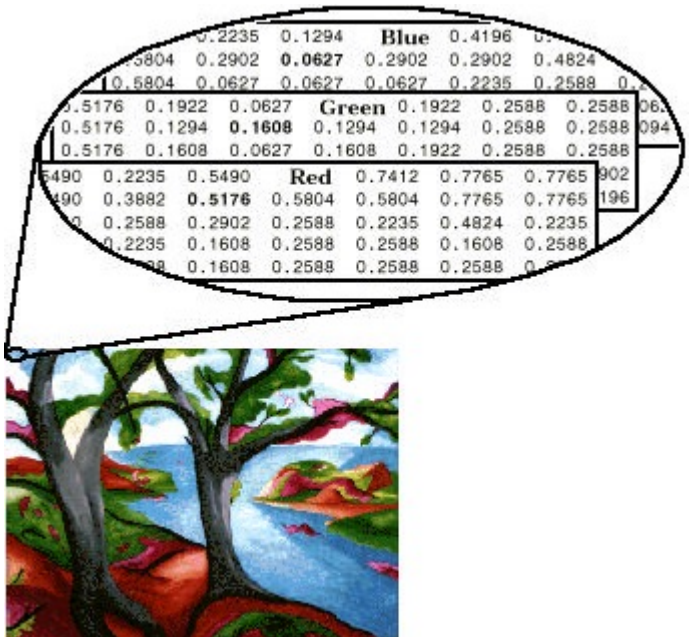
El color de cada píxel viene determinado por la combinación de las intensidades roja, verde y azul guardadas en cada plano de color en la posición del píxel.

Las imágenes RGB representan cada píxel con 24 bits, ocho para cada componente de color, por lo tanto el rango es de 16 millones de colores.

Un array que represente a una imagen RGB puede ser de tipo double, uint8 o uint16. En un array RGB de tipo double, cada componente de color es un vlaor entre cero y uno.

Un píxel cuyas componentes de color sean (0, 0, 0) será un punto negro y si fuesen (1, 1, 1) sería un punto blanco. Las tres coordenadas de color para cada píxel son guardadas en este orden, primero el rojo, después el verde y por último el azul.

La siguiente imagen es una imagen RGB de tipo double.



Para determinar la coordenada del píxel (2, 3) tenemos que mirar la tabla de tres valores almacenados en (2, 3,1:3). En el caso de la figura el color del píxel sería 0.5176 0.1608 0.0627.

El Image Processing Toolbox de Matlab dispone de comandos para leer, escribir y visualizar varios tipos de formatos de imagen:

- BMP (Microsoft Windows Bitmap)
- HDF (Hierarchical Data Format)
- JPEG (Joint Photographic Experts Group)
- PCX (Paintbrush)
- PNG (Portable Network Graphics)
- TIFF (Tagged Image File Format)
- XWD (X Window Dump)

6.3.5- Objetos y funciones de comunicación.

A continuación se explica los comandos, objetos y funciones de Matlab necesarios para establecer comunicación entre el ordenador y los dispositivos externos a través de puertos series.

El objeto serial

[Sintaxis]

obj = serial('puerto')
obj = serial('puerto', 'propiedad', 'valorPropiedad')

[Argumentos]

Nombre	Descripción
'puerto'	El nombre del puerto serie 'COM1', 'COM2',..
'Propiedad'	Propiedad del puerto serie 'Baudrate', 'Paridad', 'terminator'
'valorPropiedad'	Valor de la propiedad del puerto serie
'obj'	El objeto puerto serie

[Descripción]

obj = serial('puerto') Crea un objeto puerto serie asociado con el puerto serie especificado en 'puerto', Si el puerto no existe o está en uso, no será posible conectar el puerto serie al dispositivo.

obj = serial('puerto', 'propiedad', 'valorPropiedad',...) Crea un objeto puerto serie con las propiedades especificadas por el nombre de la propiedad y el valor de la propiedad. Si se especifica alguna propiedad errónea, se devolverá un error y el objeto serie no se creará.

Para poder comunicarse con el dispositivo, el objeto creado debe estar conectado al puerto serie con la función fopen.

Cuando se crea el puerto serie, la propiedad 'Status' tiene el valor 'close' (cerrado). Una vez que el objeto esta conectado con el puerto serie mediante la función fopen, la propiedad 'Status' se configurará en 'open' (abierto). Solamente un objeto puerto serie puede estar conectado a un puerto serie al mismo tiempo.

[Observaciones]

Cuando se crea un objeto puerto serie las siguientes propiedades son configuradas automáticamente:

- La propiedad 'Type' viene dada por el objeto serial.
- La propiedad 'Name' viene dada por la concatenación de el objeto serie con el puerto especificado en la función serial.
- La propiedad 'Port' viene dada por el puerto especificado en la función serial.

Además se puede especificar los nombres de las propiedades sin escribir el nombre completo. Por ejemplo, todos los siguientes comandos son validos:

```
s = serial('COM1','BaudRate',4800);
s = serial('COM1','baudrate',4800);
s = serial('COM1','BAUD',4800);ç
```

[Ejemplos]

Este ejemplo crea el objeto puerto serie s1 asociado al puerto serie COM1:

```
>> s1 = serial('COM1');
```

El tipo, nombre y las propiedades del puerto son configuradas automáticamente, en el siguiente ejemplo.

```
>> get(s1,{'Type','Name','Port'})
ans =
   'serial'    'Serial-COM1'    'COM1'
```

Para especificar propiedades mientras se crea el objeto:

```
>> s2 = serial('COM2','BaudRate',1200,'DataBits',7);
```

Para preguntar al dispositivo:

```
>> fprintf(s1, '*IDN?');
>> idn = fscanf(s1);
```

La funcion fopen

[Sintaxis]

Fopen(obj)

[Argumentos]

Nombre	Descripción
obj	Puede ser un puerto serie o un array que contenga el nombre del puerto serie.

[Descripción]

Fopen(obj) conecta el objeto puerto serie obj a un dispositivo externo conectado al puerto serie del obj.

[Observaciones]

Antes de poder realizar una operación de lectura o escritura, el objeto puerto serie debe estar conectado al dispositivo mediante la función fopen. Cual el objeto esta conectado al dispositivo:

- Los datos pendientes en el buffer de entrada o de salida se borran.
- La propiedad 'Status' se configura a 'open'.
- Las propiedades 'BytesAvailable', 'ValuesReceived' y 'BytesToOutput' se establecen en 0.

Se devolverá un error si se intenta realizar una operación de lectura o escritura mientras obj no este conectado al dispositivo. Solo se puede conectar un objeto puerto serie a un dispositivo.

Algunas propiedades solo se pueden leer mientras el objeto puerto serie está conectado al dispositivo, y deben ser configuradas antes de usar fopen. Algunas de estas propiedades son; 'InputBufferSize', 'OutputBufferSize',..

Los valores de algunas propiedades solo son verificados después de conectar obj al dispositivo. Si alguna de esas propiedades están configuradas de forma incorrecta, se devolverá un error cuando se ejecute fopen para conectar obj al dispositivo.

[Ejemplo]

Este ejemplo crea un objeto puerto serie ‘s’, lo conecta mediante fopen, escribe y lee datos de texto y después desconecta s del dispositivo:

```
>> s = serial('COM1');
>> fopen(s)
>> fprintf(s,'*IDN?')
>> idn = fscanf(s);
>> fclose(s
```

La funcion fclose

[Sintaxis]

Fclose(obj)

[Argumentos]

Nombre	Descripción
obj	Puede ser un puerto serie o un array que contenga el nombre del puerto serie.

[Descripción]

La función fclose desconecta un objeto puerto serie del dispositivo que se encuentre en el puerto especificado por obj.

[Observaciones]

Si obj se desconecta con existo, entonces la propiedad ‘Status’ se configura al valor ‘cerrado’ y la propiedad ‘RecordStatus’ se configura a ‘off’. Se puede volver a conectar el obj al dispositivo mediante la función fopen.

Se devolverá un error si se ejecuta fclose mientras se están escribiendo datos de froma asincrona, en este caso, se debe abortar la operación con el comando stopasync o esperar a que termine la operación de escritura.

La función fprintf

[Sintaxis]

```
fprintf(obj,'cmd')
fprintf(obj,'format','cmd')
fprintf(obj,'cmd','mode')
fprintf(obj,'format','cmd','mode')
```

[Argumentos]

Nombre	Descripción
Obj	El objeto puerto serie
‘cmd’	La cadena de caracteres escrita en el dispositivo
‘format’	Especificación de conversión del lenguaje C
‘mode’	Especifica si los datos se escriben de forma sincrónica o asincrónica.

[Descripción]

fprintf(obj, ‘cmd’) Escribe la cadena de caracteres cmd en el dispositivo conectado a obj. El formato por defecto es %s\n. La operación de escritura es sincrónica y bloquea la línea de comandos hasta que la ejecución se ha completado.

fprintf(obj, ‘format’, ‘cmd’) Escribe la cadena usando el formato especificado por el formato. El formato es una conversión del lenguaje C especificada. Las especificaciones de conversión implican el carácter % y los caracteres de conversión d, i, o, u, x, X, F, E, E, G, G, C y S.

fprintf (obj, 'formato', 'cmd', 'mode') Escribe la cadena usando el formato especificado. Si el modo es síncrono, cmd está escrito de forma sincrónica. Si el modo es asíncrono, se escribe cmd de forma asíncrona. Si el modo no es especificado la operación de escritura será síncrona.

[Observaciones]

Antes de poder escribir texto en el dispositivo, este debe estar conectado a obj con la función fopen. Se devolverá error si se intenta realizar una operación de escribir mientras el dispositivo no está conectado a obj.

[Operación de escritura síncrona/asíncrona]

Por defecto, el texto se escribe en el dispositivo externo de forma síncrona y la línea de comandos permanece bloqueada hasta que la operación termina. Se puede seleccionar la forma asíncrona configurando la propiedad 'mode' a ' 'async'. Para escrituras asíncronas se cumple que:

- El valor de la propiedad 'BytesToOutput se refresca continuamente para reflejar el número de bytes en el buffer de salida.

- La función M-file especificada por la propiedad OutputEmptyFcn se ejecuta cuando el buffer de salida está vacio.

Se puede determinar cuando una operación de escritora asíncrona está en progreso con la propiedad 'TransferStatus'.

[Ejemplo]

Se crea el puerto serie s, se conecta a un dispositivo externo y se escribe el comando RS232? con la función fprintf:

```
>> s = serial('COM1');
>> fopen(s)
>> fprintf(s,'RS232?')
```

La función fscanf

[Sintaxis]

```
A = fscanf(obj)
A = fscanf(obj,'format')
A = fscanf(obj,'format',size)
[A,count] = fscanf(...)
[A,count,msg] = fscanf(...)
```

[Argumentos]

Nombre	Descripción
obj	El objeto puerto serie
‘format’	Especificación de conversión del lenguaje C
size	Número de valores a leer
A	Datos leídos del dispositivo y convertido a texto
count	Número de valores leidos
msg	Mensaje que indica si la operación de lectura no se realizó con éxito

[Descripción]

A = fscanf(obj) Lee datos desde el dispositivo externo conectado a obj, y los devuelve en A. Los datos son convertidos a texto usando el formato %c.

A = fscanf(obj, ‘format’) Lee datos y los convierte de acuerdo al formato. ‘format’ especifica la conversión del lenguaje C. Las especificaciones de conversión implican el carácter % y los caracteres de conversión d, i, o, u, x, X, F, E, E, G, G, C, y s

A = fscanf(obj, ‘format’, ‘size’) Lee el número de valores especificados en size. Los valores validos para size son:

- n: Lee al menos n valores y los escribe en un vector columna.
- [m, n]; Lee al menos m x n valores llenando una matriz m x n en orden de columnas.

El parámetro size no puede ser infinito, se devolverá un error si el número de valores especificado no se puede almacenar en el buffer de entrada. Con la propiedad InputBufferSize se especifica, en bytes, el tamaño del buffer de entrada. Un valor ASCII es un byte.

[A, count] = fscanf(…) Devuelve el número de valores leidos.

[A, count, msg] = fscanf(…) Devuelve un mensaje de alerta si la operación de lectura no ha terminado de forma correcta.

[Descripción]

Antes de poder realizar una operación de lectura desde el dispositivo externo, este debe estar conectado a obj mediante la función fopen. Un objeto puerto serie conectado tiene el valor de la propiedad 'Status' en 'open'. Se devolverá un error en el caso de realizar una operación de lectura mientras obj no esté conectado.

Cada vez que se usa fscanf el valor de la propiedad 'ValuesReceived' incrementa, el incremento será igual al número de valores leídos incluyendo el 'terminator'.

[Ejemplos]

Crear un objeto puerto serie y conectarlo a un dispositivo externo:

```
>> s = serial('COM1');
>> fopen(s);
```

Uso de fprintf para enviar un parámetro de configuración y realizar una petición de datos:

```
>> fprintf(s,'PosicionFoto = (285,135,0,180,0,180,0,0)(0,0)')
>> fprintf(s,'PosicionFoto?')
```

Uso de fscanf para leer los valores recibidos. La operación termina al leer el teminator.

```
>> PosFoto = fscanf(s)
>> PosFoto=
    (285,135,0,180,0,180,0,0)(0,0)
```

Desconectar el objeto puerto serie del dispositivo externo y liberar memoria del espacio de trabajo borrando s:

```
>> fclose(s)
>> delete(s)
>> clear(s)
```

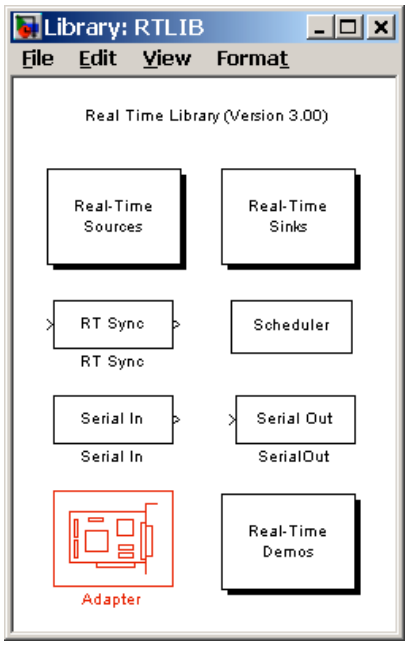
6.4.- Puesta en marcha de la tarjeta de adquisición de datos.

Para utilizar la tarjeta usaremos un toolbox de Matlab denominado 'Extended Real Time Toolbox'. Este paquete permite trabajar con la tarjeta de adquisición de datos directamente desde Matlab e incluso desde esquemas Simulink, ofreciendo así una gran potencia y facilidad en la etapa de desarrollo de un sistema de control.

El Extended Real Time Toolbox (en adelante RTT) permite la comunicación directa entre el sistema de adquisición de datos y Matlab / Simulink; es decir, permite acceder a la tarjeta desde la línea de comando de Matlab o directamente desde un esquema Simulink.

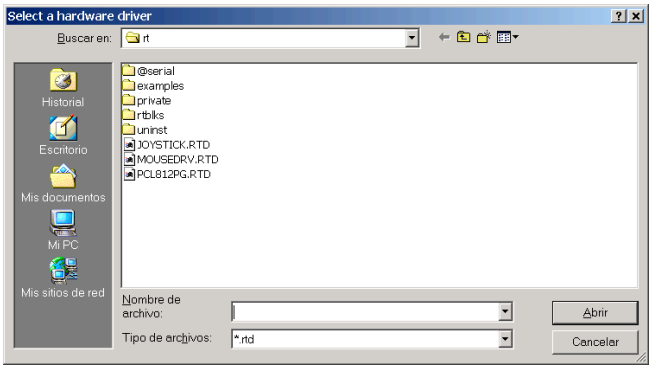
La utilización de la RTLIB en un esquema Simulink pasa por dos fases: primero la inclusión y configuración del driver del hardware de nuestra tarjeta y segundo la correcta utilización de los diferentes bloques de entrada/salida en tiempo real.

El acceso a dicha librería se puede realizar desde la línea de comandos de Matlab mediante el comando **rtlib**. Al ejecutar dicho comando aparecerá una ventana de Simulink como la que se muestra en la siguiente figura:



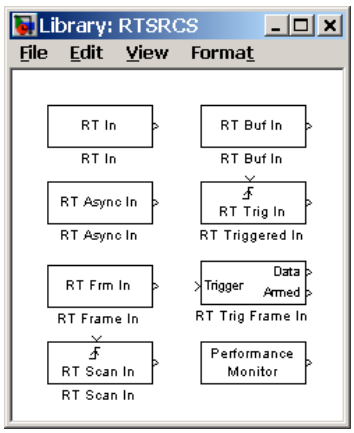
A partir de aquí abriremos un nuevo esquema Simulink y podemos comenzar a trabajar con los bloques de la RTLIB. El primer paso es incluir en nuestro esquema un bloque adaptador. El bloque Adapter es un bloque que representa la tarjeta de adquisición de datos y hace las veces de driver

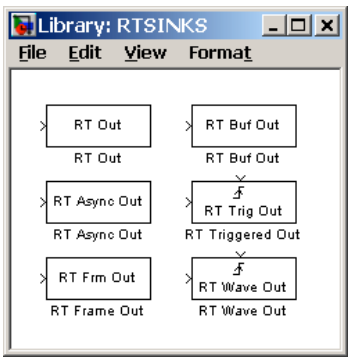
o interfaz entre Simulink y la tarjeta. Se utiliza para disponer dentro del esquema Simulink de los canales de la tarjeta de adquisición de datos. Una vez incluido dicho bloque en el esquema Simulink, lo primero que hay que hacer es configurar el mismo para que utilice el driver de la tarjeta que poseemos. Para ello haremos doble clic con el ratón sobre el mismo y se abrirá una ventana como la que se muestra:



En nuestro caso, el driver que debemos cargar es el correspondiente al fichero PCL812PG.RTD, para ello lo seleccionamos y pulsamos '*Aceptar*'. A partir de este momento ya tenemos cargado en nuestro esquema Simulink los drivers de la tarjeta ACL – 8112PG con las opciones por defecto. En el caso de querer cambiar dichas opciones haremos de nuevo doble clic sobre el bloque *Adapter* y se abrirá una ventana de selección de parámetros. En principio y a menos que se indique lo contrario, se trabajará con las opciones del driver por defecto por lo que no será necesario realizar este último paso (se pulsará OK sin cambiar ningún parámetro).

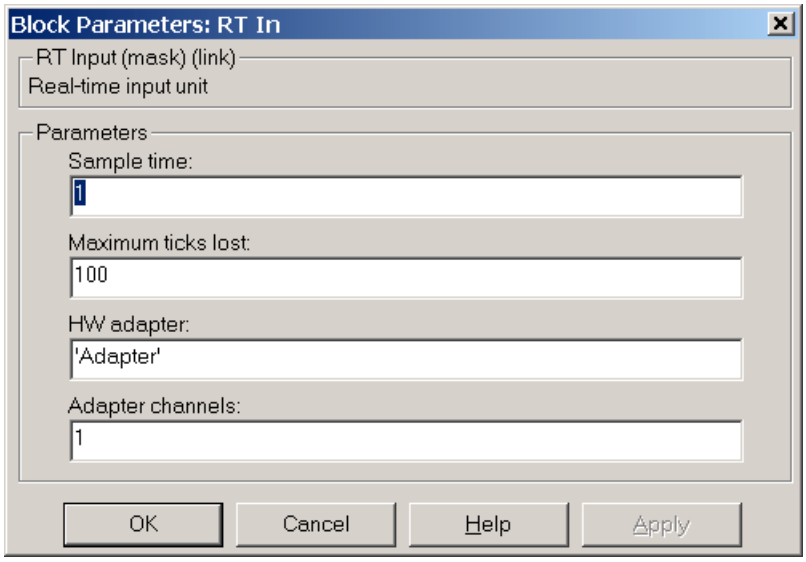
Ahora ya podemos empezar a utilizar los bloques de entrada/salida de la librería. Se dispone de dos tipos de bloques: *sources* y *sinks*. Para acceder a los mismos haremos doble clic sobre los bloques *Real – Time Sources* o *Real – Time Sinks* respectivamente. De esta forma aparecerán las ventanas correspondientes tal y como se muestra en la siguiente figura:





Como se puede apreciar aparecen multitud de bloques tanto en una como en otra ventana. En este documento sólo se van a analizar aquellos que se van a utilizar en la práctica. Dichos bloques son *'RT In'* y *'RT Out'* ya que son los mejores para realizar el control en tiempo real.

El bloque *'RT In'* está diseñado para la adquisición de datos en tiempo real. Para configurar este bloque haremos doble clic sobre el mismo, apareciendo así una ventana como la que se muestra a continuación:



Veamos los parámetros configurables:

- *Sample Time:* especifica el periodo de muestreo al cual se debe realizar la adquisición de los datos.
- *Maximum ticks lost:* especifica el máximo número de 'tics' (muestreos) que se pueden perder antes de mostrar un mensaje de error. Esto es un mecanismo de "seguridad" para asegurar que la adquisición se ha realizado de forma correcta (no ha habido excesivas pérdidas de datos) ya que hay que tener en cuenta que este sistema en tiempo real se ejecuta sobre Matlab y Simulink, y éstos a su vez sobre Windows por lo que cabe la posibilidad de que en un determinado instante el sistema esté tan cargado que no se pueda realizar la adquisición con la frecuencia correcta. Esto, claro está, depende tanto de la potencia del microprocesador como de la cantidad de recursos de la máquina que están ocupados (aplicaciones, ventanas abiertas, etc.).

- *HW adapter:* indica el nombre del bloque *Adapter* incluido en el esquema Simulink. Esto es así pues cabe la posibilidad de tener en un mismo esquema más de un adaptador para gestionar dos sistemas de adquisición distintos.

- *Adapter channels:* especifica el canal analógico de entrada al que se quiere acceder. En este caso tendremos 8 canales numerados del 1 al 8 correspondiendo a los canales de entrada analógica 0 a 7 de la tarjeta de adquisición.

Los parámetros a fijar en el bloque *RT Out* son idénticos a los anteriores (y por tanto la ventana de configuración) con la salvedad de que en este caso se trata de un bloque de salida y se emplea para generar las señales necesarias con el mínimo retardo posible.

En este caso, el parámetro *Sample Time* especifica el periodo al cual se actualiza el canal de salida correspondiente en la ACL – 8112PG, *Maximum ticks lost*, el número de veces que no se ha podido actualizar dichos canales debido a la carga del sistema, *HW adapter* es idéntico al caso anterior, y *Adapter channels* especifica el canal de salida analógica al que se accede que en este caso son 2, numerados del 1 al 2 correspondiendo a los canales de salida analógica 0 a 1 de la tarjeta ACL – 8112PG.

6.5- Problemas con MATLAB y AMD.

Quizás no sea muy común este problema, pero al menos ocurre con los PC`S que tienen procesador AMD y se utiliza la versión del 2004-2006 de Matlab7.

Antes de la solución, recordaremos el problema: Se instala Matlab7 sin ninguna dificultad, pero al ejecutarlo, se muestran todos sus menús y ventanas y de inmediato se cae, sin informar de que sucede…

La solución que funciono en mi caso (para AMD 32bits y algunos AMD de 64bits):

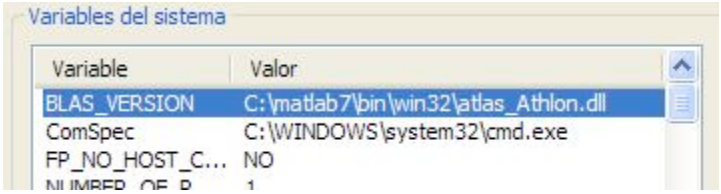
1. Ir a propiedades del sistema (click derecho en Mi PC, Propiedades).

2. Entrar en opciones avanzadas y luego en variables de entorno.

3. Abajo, en variables de sistema, hacer click en Nueva.

4. El nombre de la variable a colocar es BLAS_VERSION.

5. El valor de la variable es la ruta y el archivo atlas_Athlon.dll de la carpeta Matlab. Por defecto cuando instalas es C:matlab7binwin32atlas_Athlon.dll.



Al finalizar se debe aceptar todo para que se guarde la configuración y sin necesidad de reiniciar el PC, ejecutar Matlab7 y funcionará.

7.- Bibliografía.

(1) Autómatas Programables. Teoría y práctica. Nicolás M. Aracil, Miguel Almonacid Kroeger, Roque J. Saltarén Pazmiño y Rafael Puerto Manchón.

(2) Indicaciones relativas al producto CD de instalación de la versión de STEP 7.

(3) Autómata programable S7-224xp. Configuración, instalación y datos de la CPU.

(4) Manual de instrucciones el robot Mitsubishi RH-5AH55

(5) Ayuda online de MATLAB 7.

(6) Aprendizaje mediante árboles de decisión para la síntesis de agarres robóticos, Tesis doctoral Cesar Fernández Peris.

Páginas Web de interés:

http://jhproject.wordpress.com/2007/09/30/solucion-de-matlab-7-en-amd-y-xp/

http://www.mathworks.es/products/image/demos.html?file=/products/demos/image/IntroIPdemo/launchpage.html

http://issuu.com/miguelbueno/docs/s7200_manual_tecnico/342

http://www.mitsubishi-automation.es/

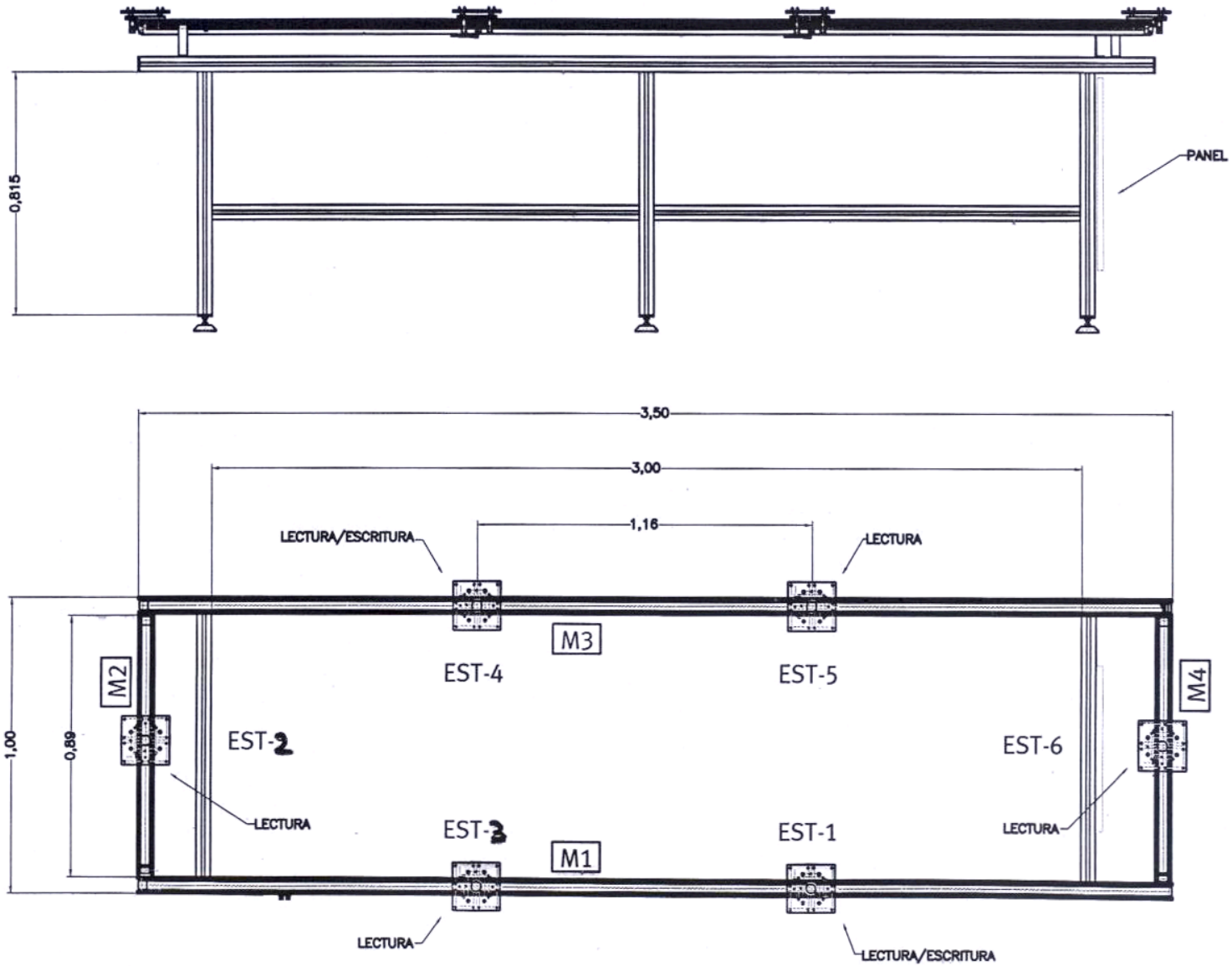
8.- Planos.

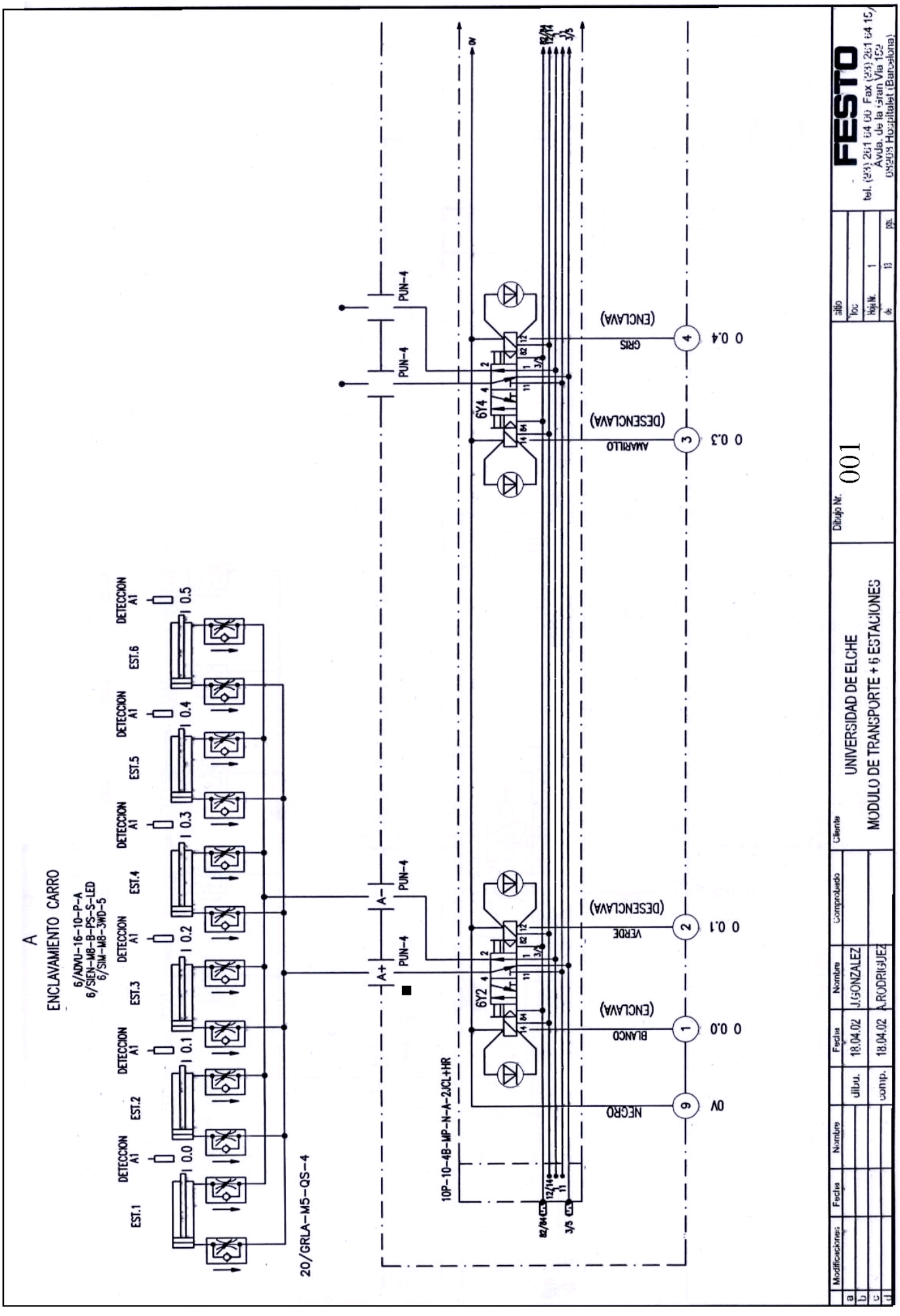
A continuación se adjuntan los planos de los componentes de la célula de fabricación flexible, el modulo de transporte y el robot industrial Mitsubishi.

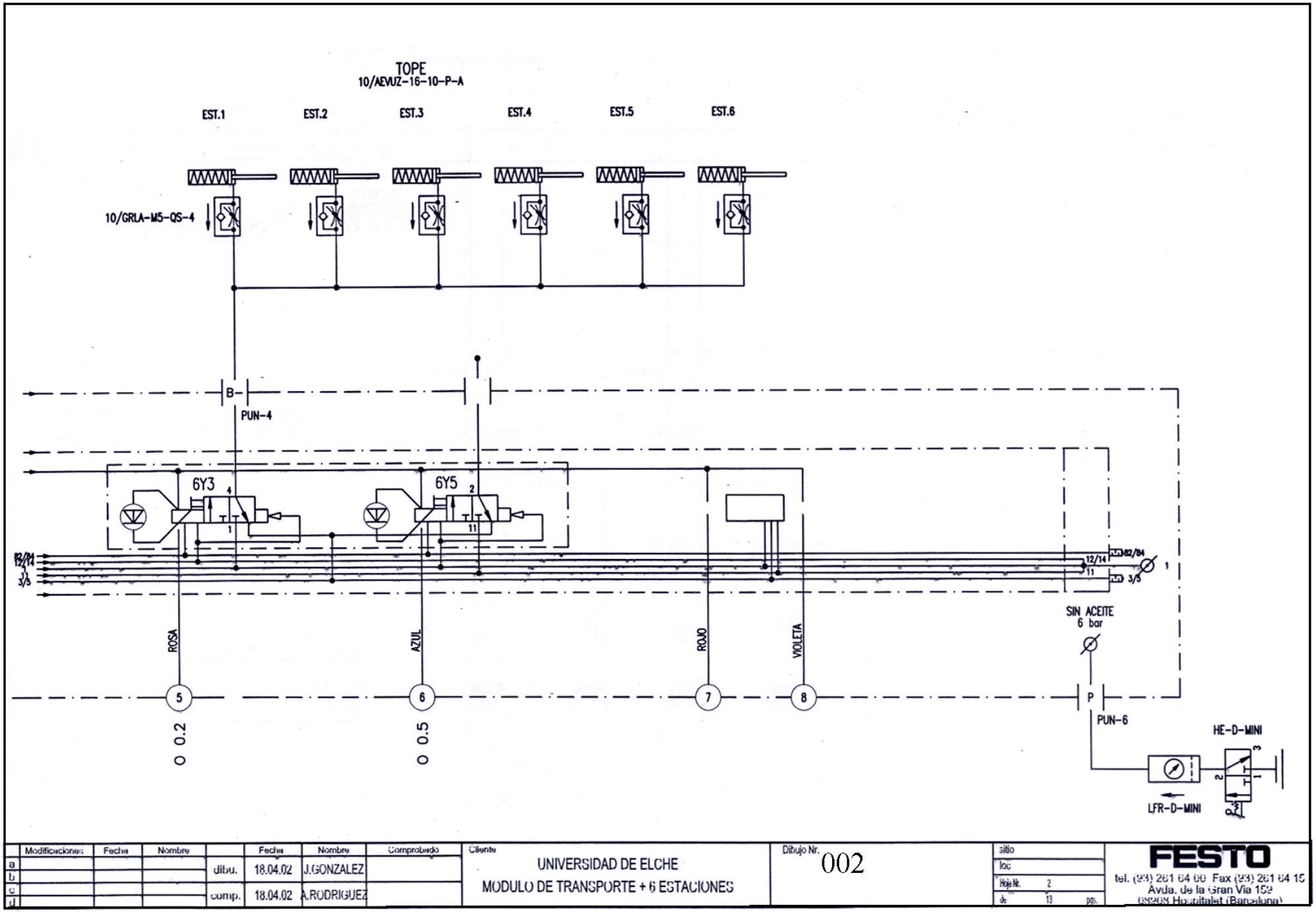
Configuración especial para la universidad de Elche
6 estaciones y codificación de carros
Módulo de transporte de 3,5 x 1 mts

PLANO 01532

FESTO




A
ENCLAVAMIENTO CARRO
6/ADVU-16-10-P-A
6/SIEN-M8-B-PS-S-LED
6/SIM-M8-3WD-5
20/GRLA-M5-QS-4
10P-10-4B-MP-N-A-2ACL+HR
EST.1
EST.2
EST.3
EST.4
EST.5
EST.6
DETECCION A1
I 0.0
I 0.1
I 0.2
I 0.3
I 0.4
I 0.5
PUN-4
A+
A-
6Y2
6Y4
0V
NEGRO
BLANCO
VERDE
AMARILLO
GRIS
(ENCLAVA)
(DESENCLAVA)
O 0.0
O 0.1
O 0.3
O 0.4
Modificaciones
Fecha
Nombre
dibu.
comp.
18.04.02
18.04.02
J.GONZALEZ
A.RODRIGUEZ
Comprobado
Cliente
UNIVERSIDAD DE ELCHE
MODULO DE TRANSPORTE + 6 ESTACIONES
Dibujo Nr.
001
Hoja Nr. 1
de 13 pág.
FESTO
tel. (93) 261 64 00 Fax (93) 261 64 15
Avda. de la Gran Vía 152
08908 Hospitalet (Barcelona)



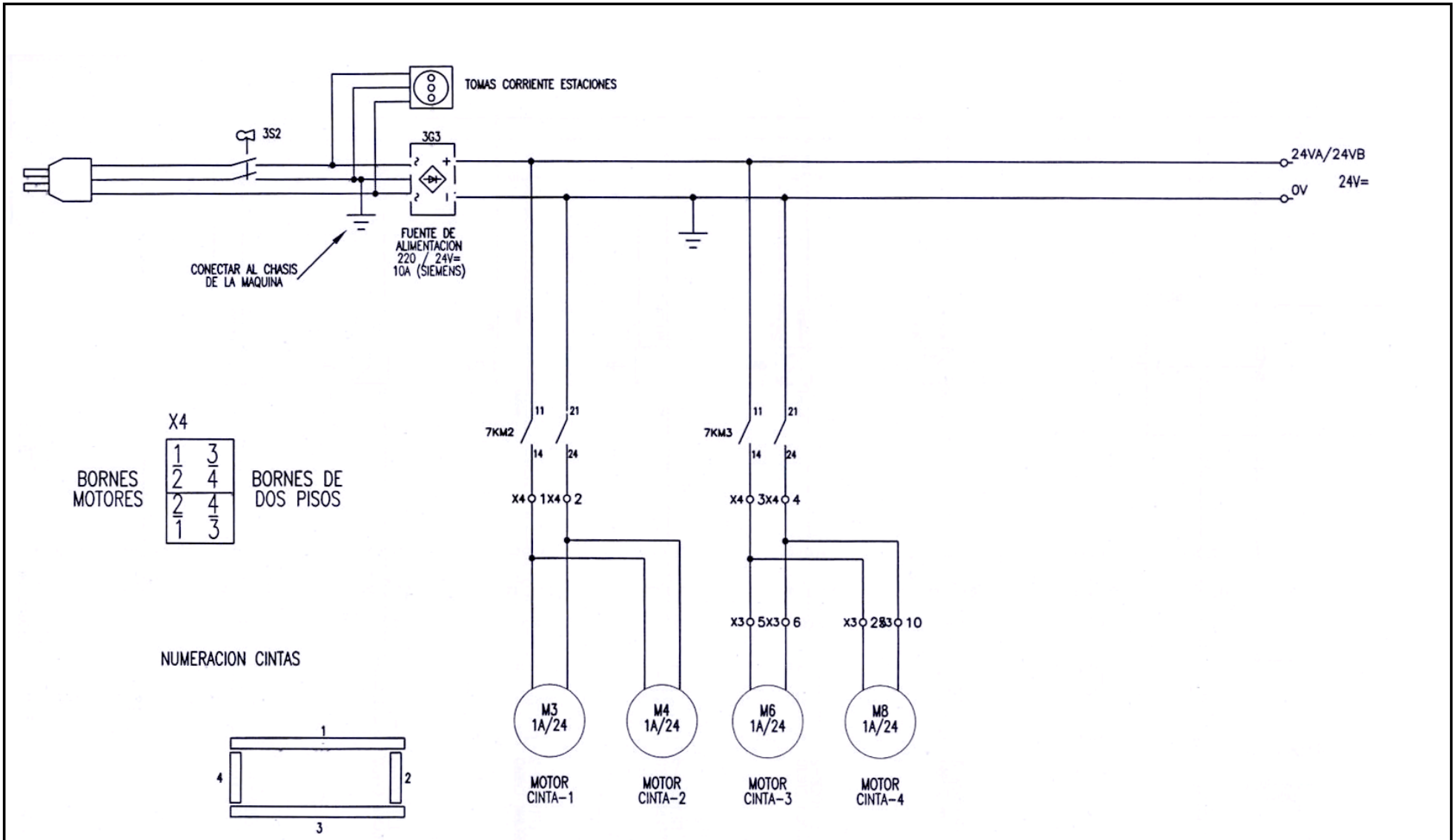
	Modificaciones	Fecha	Nombre		Fecha	Nombre	Comprobado
a				dibu.	18.04.02	J.GONZALEZ	
b							
c				comp.	18.04.02	A.RODRIGUEZ	
d							

Cliente: UNIVERSIDAD DE ELCHE
MODULO DE TRANSPORTE + 6 ESTACIONES

Dibujo Nr. 002

sitio
loc
Hoja Nr. 2
de 13 pos.

FESTO
tel. (93) 261 64 00 Fax (93) 261 64 15
Avda. de la Gran Vía 152
08908 Hospitalet (Barcelona)



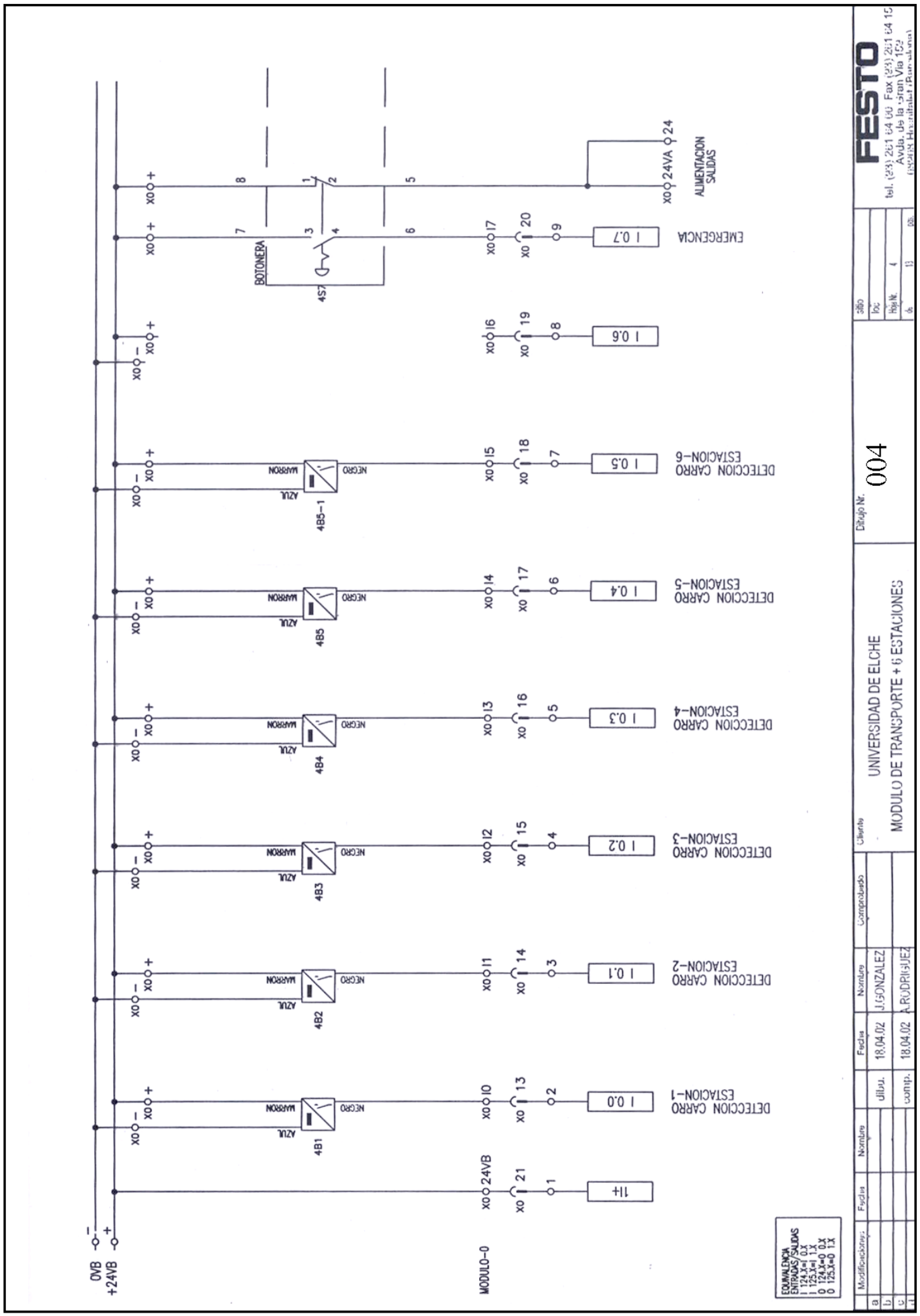
	Modificaciones	Fecha	Nombre		Fecha	Nombre	Comprobado
a				dibu.	18.04.02	J.GONZALEZ	
b							
c				comp.	18.04.02	A.RODRIGUEZ	
d							

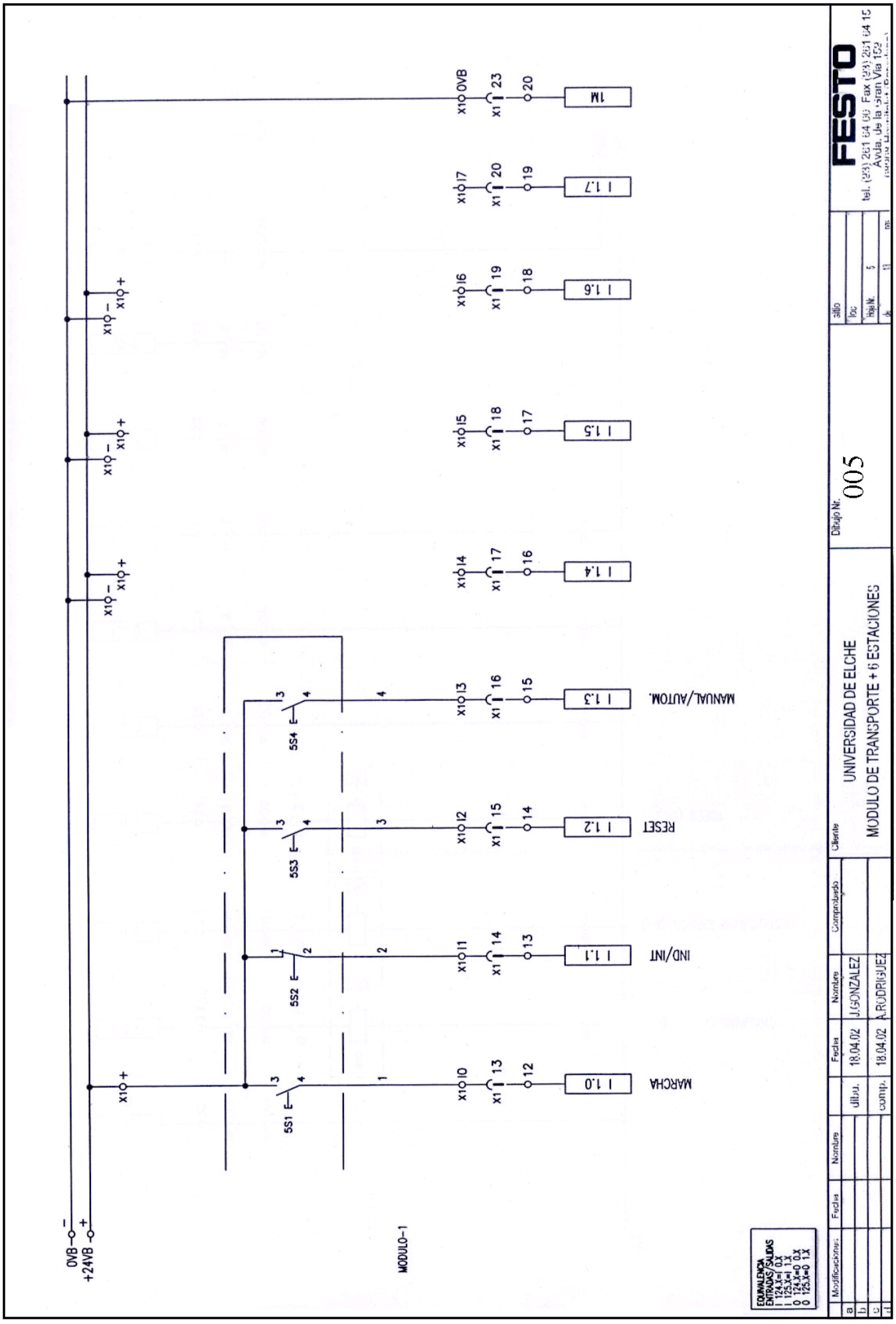
Cliente: UNIVERSIDAD DE ELCHE
MODULO DE TRANSPORTE + 6 ESTACIONES

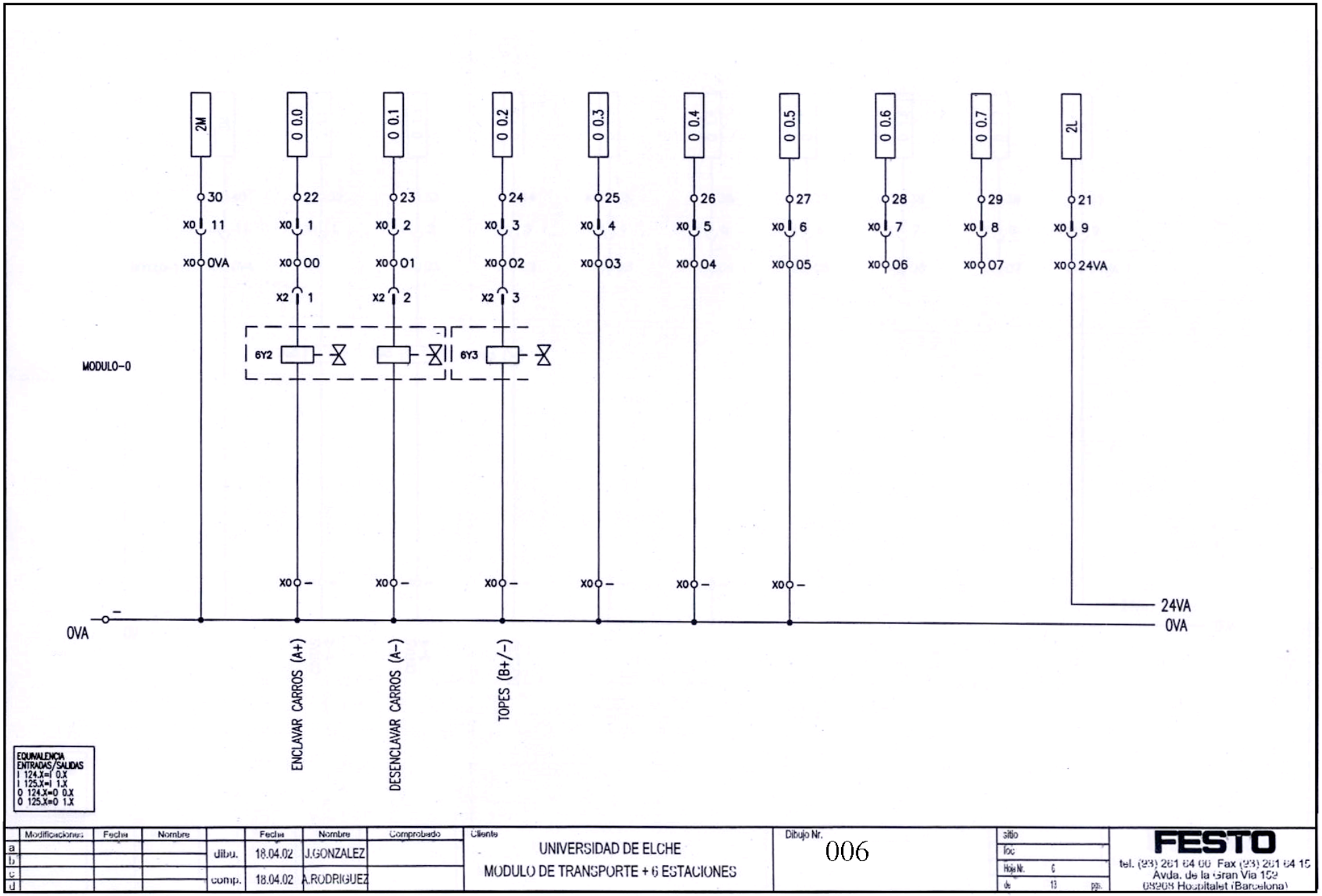
Dibujo Nr.: 003

sitio
loc
Hoja Nr. 3
de 13 pág.

FESTO
tel. (93) 261 64 00 Fax (93) 261 64 15
Avda. de la Gran Via 152
08908 Hospitalet (Barcelona)


0VB
+24VB
MODULO-0
24VB
11+
DETECCION CARRO ESTACION-1
DETECCION CARRO ESTACION-2
DETECCION CARRO ESTACION-3
DETECCION CARRO ESTACION-4
DETECCION CARRO ESTACION-5
DETECCION CARRO ESTACION-6
EMERGENCIA
BOTONERA
4S7
ALIMENTACION SALIDAS
24VA
4B1
4B2
4B3
4B4
4B5
4B5-1
AZUL
MARRON
NEGRO
UNIVERSIDAD DE ELCHE
MODULO DE TRANSPORTE + 6 ESTACIONES
Dibujo Nr.
004
FESTO


0VB
+24VB
MODULO-1
MARCHA
IND/INT
RESET
MANUAL/AUTOM.
5S1
5S2
5S3
5S4
1M
UNIVERSIDAD DE ELCHE
MODULO DE TRANSPORTE + 6 ESTACIONES
005
FESTO



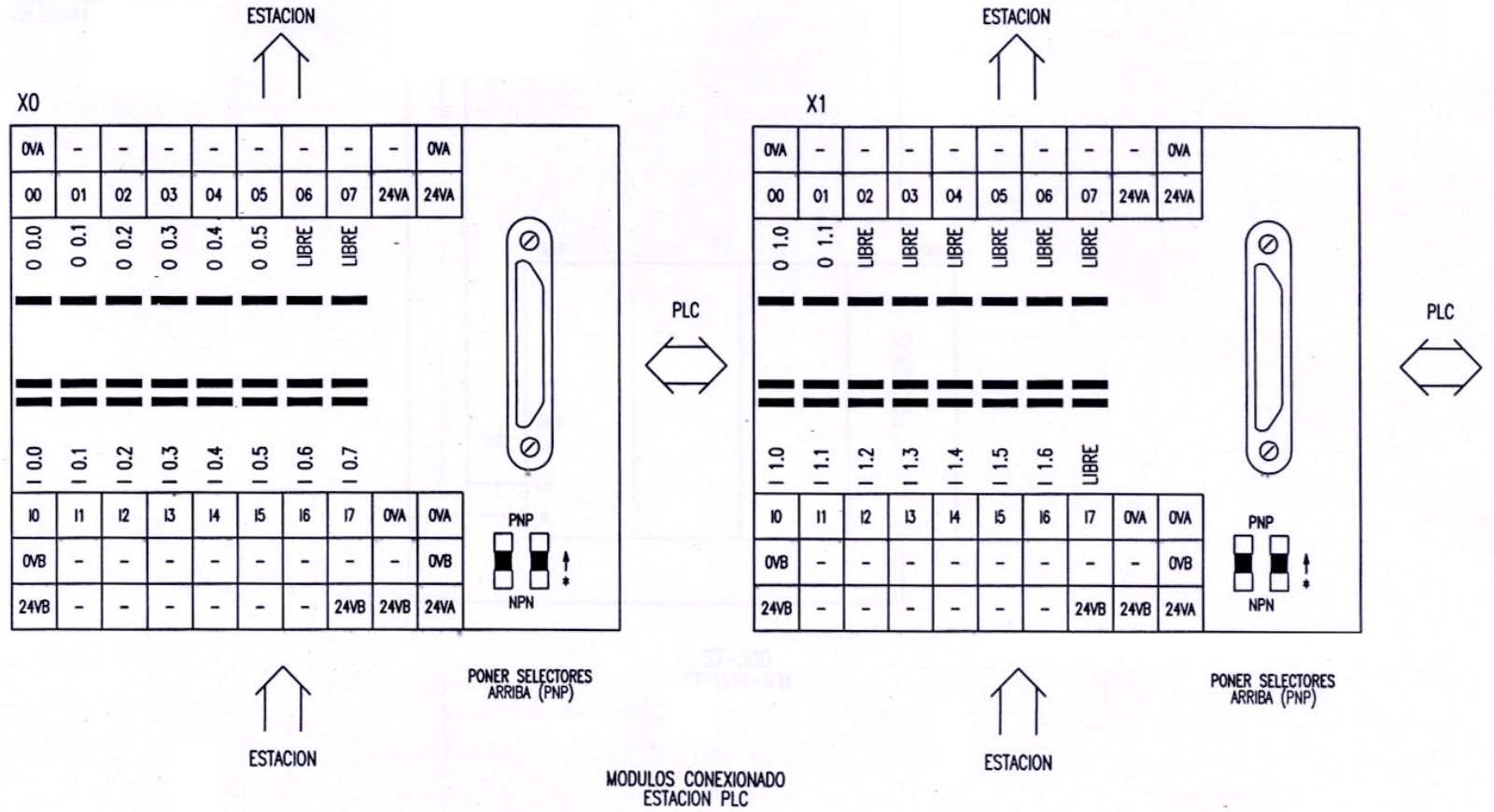
	Modificaciones	Fecha	Nombre		Fecha	Nombre	Comprobado
a				dibu.	18.04.02	J.GONZALEZ	
b							
c				comp.	18.04.02	A.RODRIGUEZ	
d							

Cliente: UNIVERSIDAD DE ELCHE
MODULO DE TRANSPORTE + 6 ESTACIONES

Dibujo Nr.: 006

sitio
loc
Hoja Nr. 6
de 13 pgs.

FESTO
tel. (93) 261 64 00 Fax (93) 261 64 15
Avda. de la Gran Vía 152
08908 Hospitalet (Barcelona)



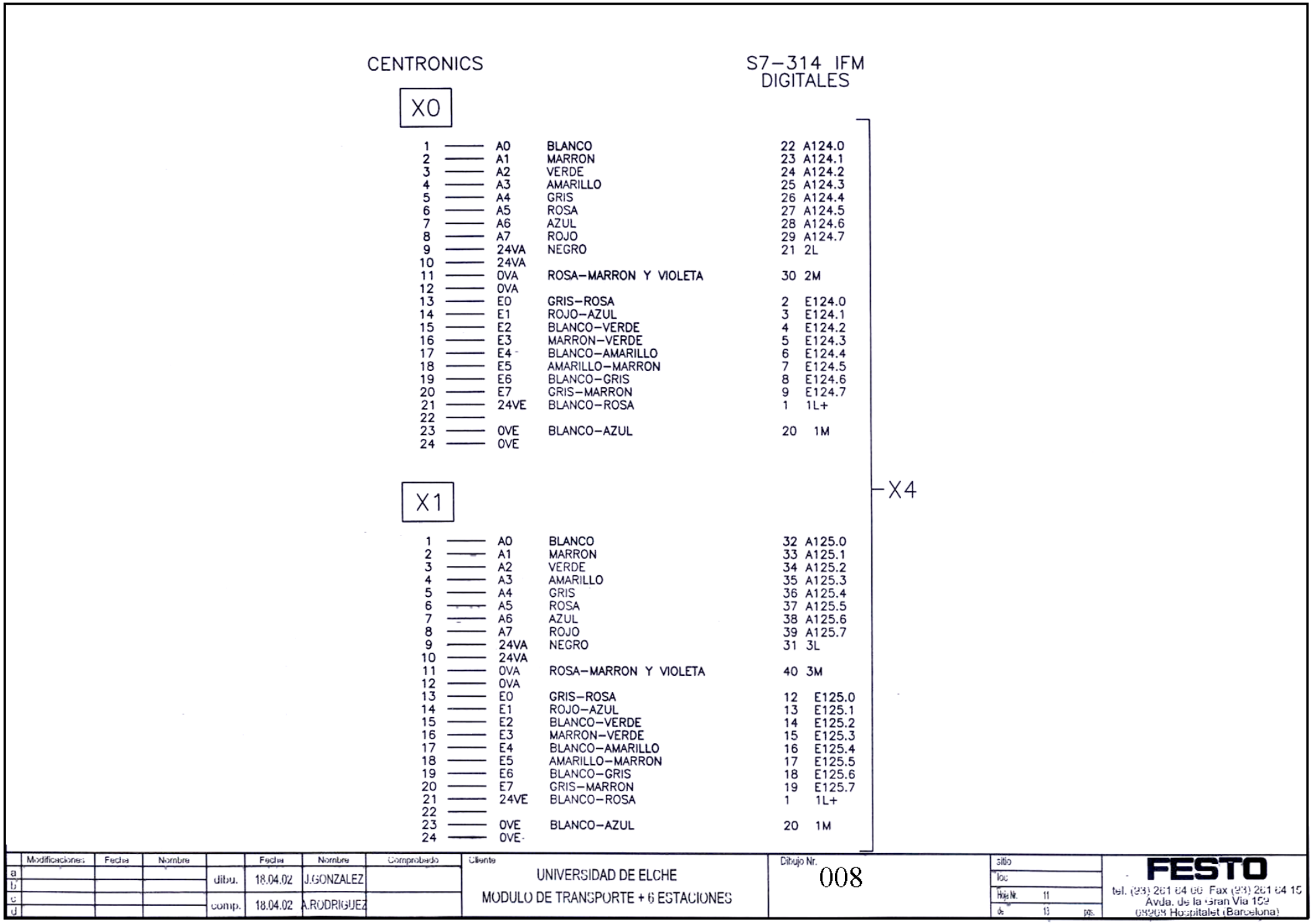
	Modificaciones	Fecha	Nombre		Fecha	Nombre	Comprobado
a				dibu.	18.04.02	J.GONZALEZ	
b							
c				comp.	18.04.02	A.RODRIGUEZ	
d							

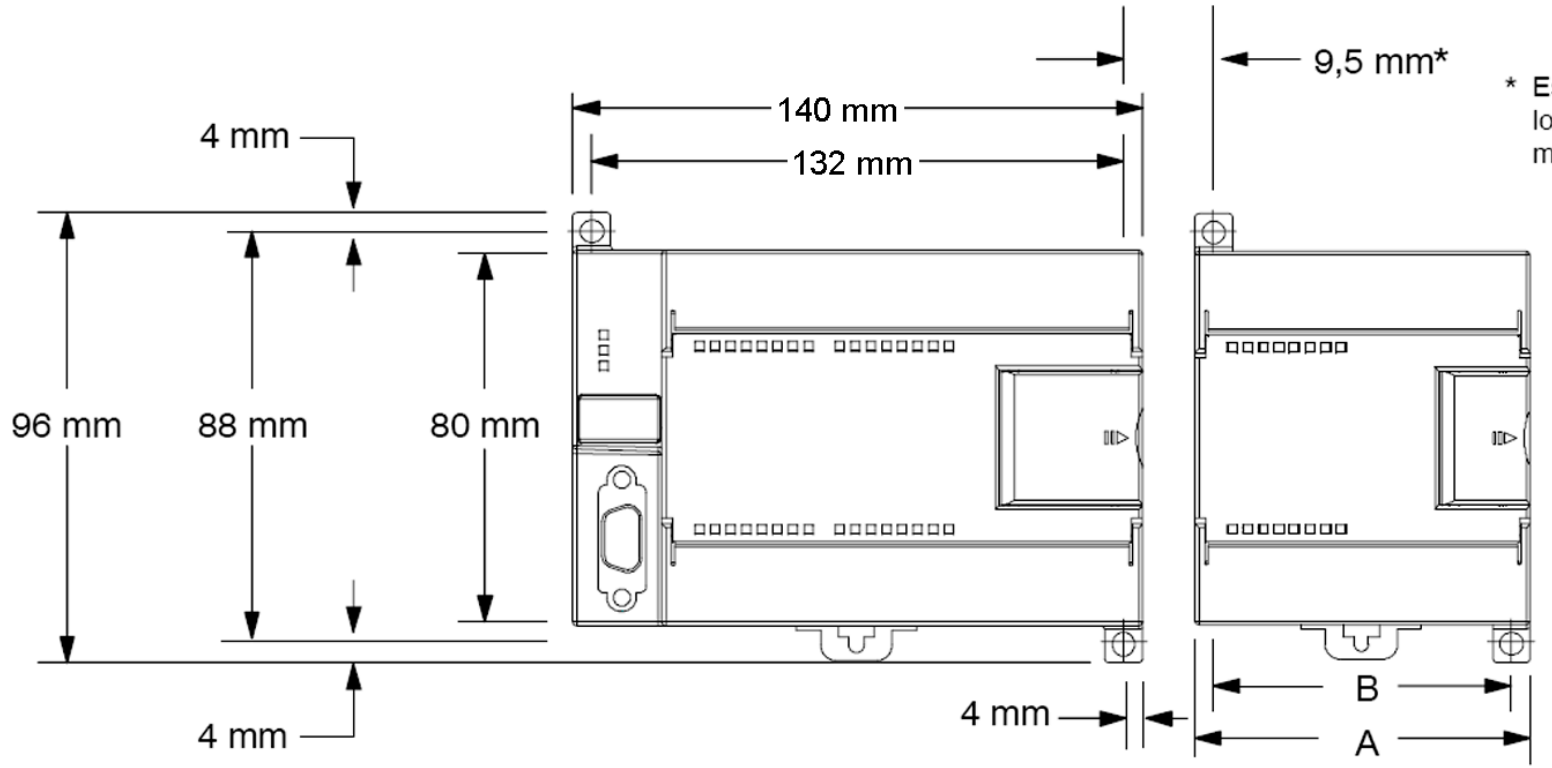
Cliente: UNIVERSIDAD DE ELCHE
MODULO DE TRANSPORTE + 6 ESTACIONES

Dibujo Nr.: 007

sitio
loc
Hoja Nr. 5
de 13 pos.

FESTO
tel. (93) 261 64 00 Fax (93) 261 64 15
Avda. de la Gran Via 152
08908 Hospitalet (Barcelona)


CENTRONICS
S7–314 IFM
DIGITALES
X0
1 A0 BLANCO 22 A124.0
2 A1 MARRON 23 A124.1
3 A2 VERDE 24 A124.2
4 A3 AMARILLO 25 A124.3
5 A4 GRIS 26 A124.4
6 A5 ROSA 27 A124.5
7 A6 AZUL 28 A124.6
8 A7 ROJO 29 A124.7
9 24VA NEGRO 21 2L
10 24VA
11 0VA ROSA–MARRON Y VIOLETA 30 2M
12 0VA
13 E0 GRIS–ROSA 2 E124.0
14 E1 ROJO–AZUL 3 E124.1
15 E2 BLANCO–VERDE 4 E124.2
16 E3 MARRON–VERDE 5 E124.3
17 E4 BLANCO–AMARILLO 6 E124.4
18 E5 AMARILLO–MARRON 7 E124.5
19 E6 BLANCO–GRIS 8 E124.6
20 E7 GRIS–MARRON 9 E124.7
21 24VE BLANCO–ROSA 1 1L+
22
23 0VE BLANCO–AZUL 20 1M
24 0VE
X1
1 A0 BLANCO 32 A125.0
2 A1 MARRON 33 A125.1
3 A2 VERDE 34 A125.2
4 A3 AMARILLO 35 A125.3
5 A4 GRIS 36 A125.4
6 A5 ROSA 37 A125.5
7 A6 AZUL 38 A125.6
8 A7 ROJO 39 A125.7
9 24VA NEGRO 31 3L
10 24VA
11 0VA ROSA–MARRON Y VIOLETA 40 3M
12 0VA
13 E0 GRIS–ROSA 12 E125.0
14 E1 ROJO–AZUL 13 E125.1
15 E2 BLANCO–VERDE 14 E125.2
16 E3 MARRON–VERDE 15 E125.3
17 E4 BLANCO–AMARILLO 16 E125.4
18 E5 AMARILLO–MARRON 17 E125.5
19 E6 BLANCO–GRIS 18 E125.6
20 E7 GRIS–MARRON 19 E125.7
21 24VE BLANCO–ROSA 1 1L+
22
23 0VE BLANCO–AZUL 20 1M
24 0VE
X4
Modificaciones
Fecha
Nombre
dibu. 18.04.02 J.GONZALEZ
comp. 18.04.02 A.RODRIGUEZ
Comprobado
Cliente
UNIVERSIDAD DE ELCHE
MODULO DE TRANSPORTE + 6 ESTACIONES
Dibujo Nr.
008
FESTO
tel. (93) 261 64 00 Fax (93) 261 64 15
Avda. de la Gran Via 152
08908 Hospitalet (Barcelona)



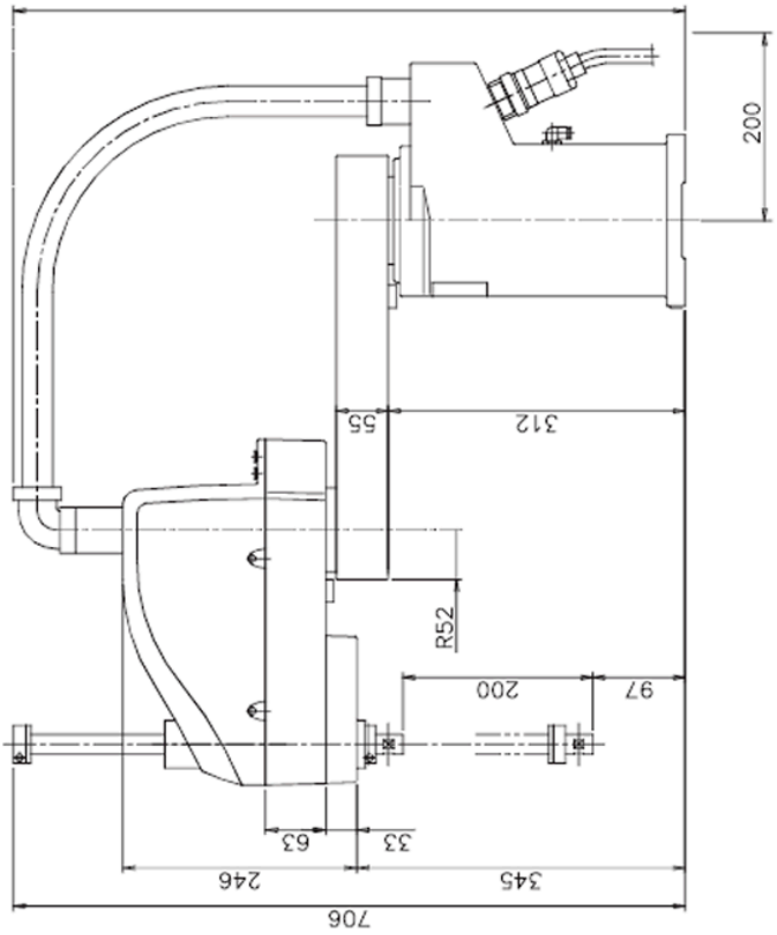
	Modificaciones	Fecha	Nombre		Fecha	Nombre	Comprobado
a				dibu.	10.02.09	J.FGALVEZ	
b							
c				comp.	10.02.09	J.FGALVEZ	
d							

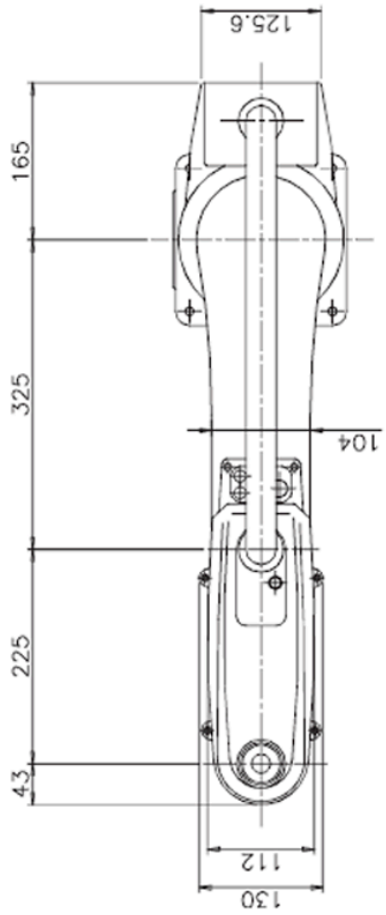
Cliente: UNIVERSIDAD DE ELCHE
AUTOMATA PROGRAMABLES

Dibujo Nr.: 009

sitio
loc
Hoja Nr. 11
de 13 pgs.

SIEMENS





MITSUBISHI ELECTRIC

	Modificaciones	Fecha	Nombre
a			
b			
c			
d			

	Fecha	Nombre
dibu.	10.02.09	JFGALVEZ
comp.	10.02.09	JFGALVEZ

Comprobado

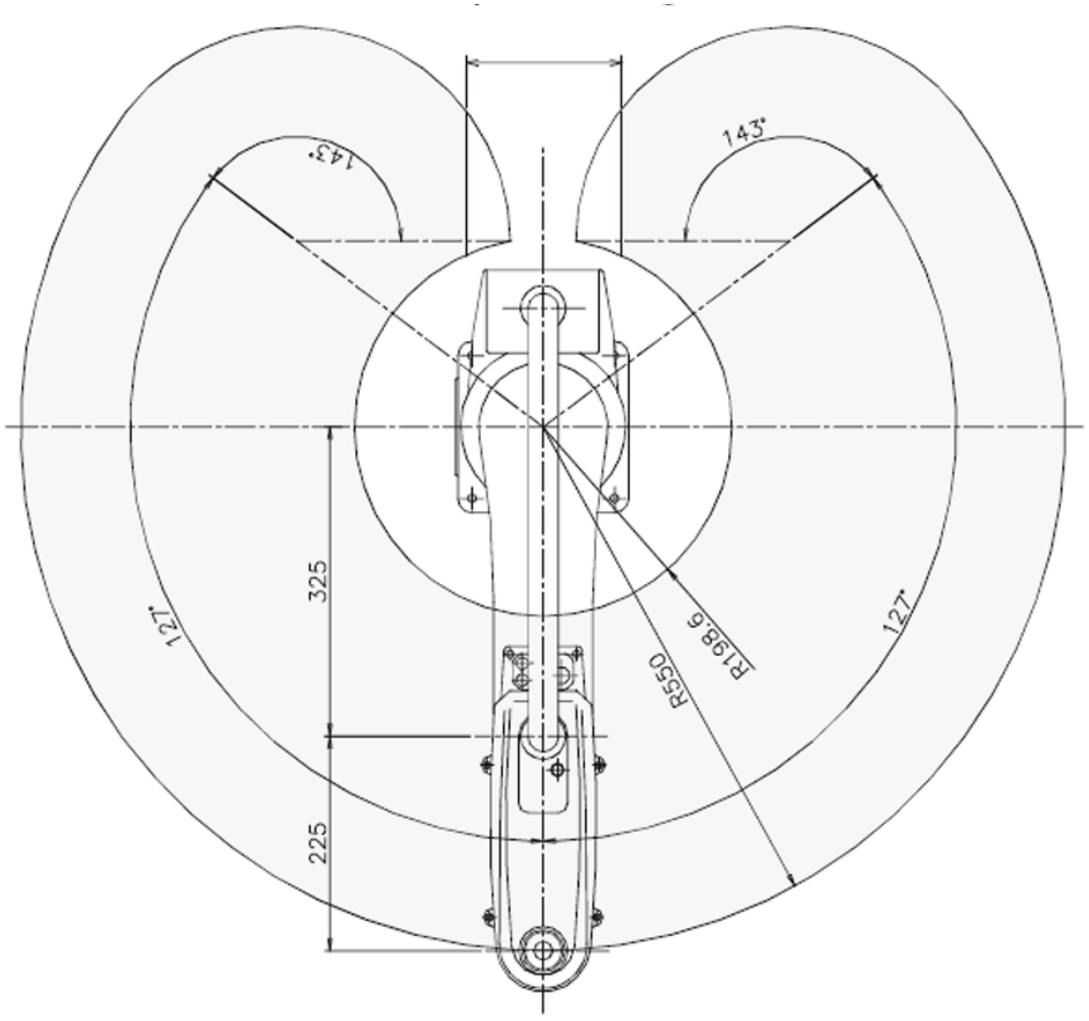
Cliente

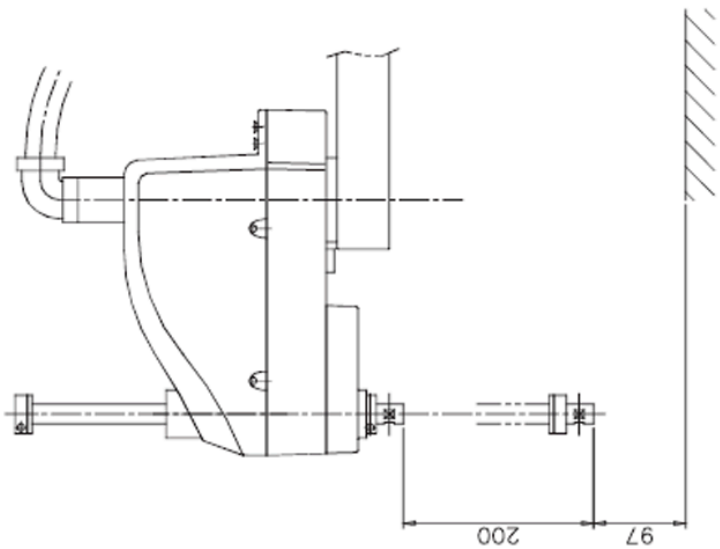
UNIVERSIDAD DE ELCHE
ROBOT RH-5AH

Dibujo Nr. 010

Escala

Hoja Nr. 11





Modificaciones:	Fecha	Nombre
a		
b		
c	dibuj.	
d	comp.	

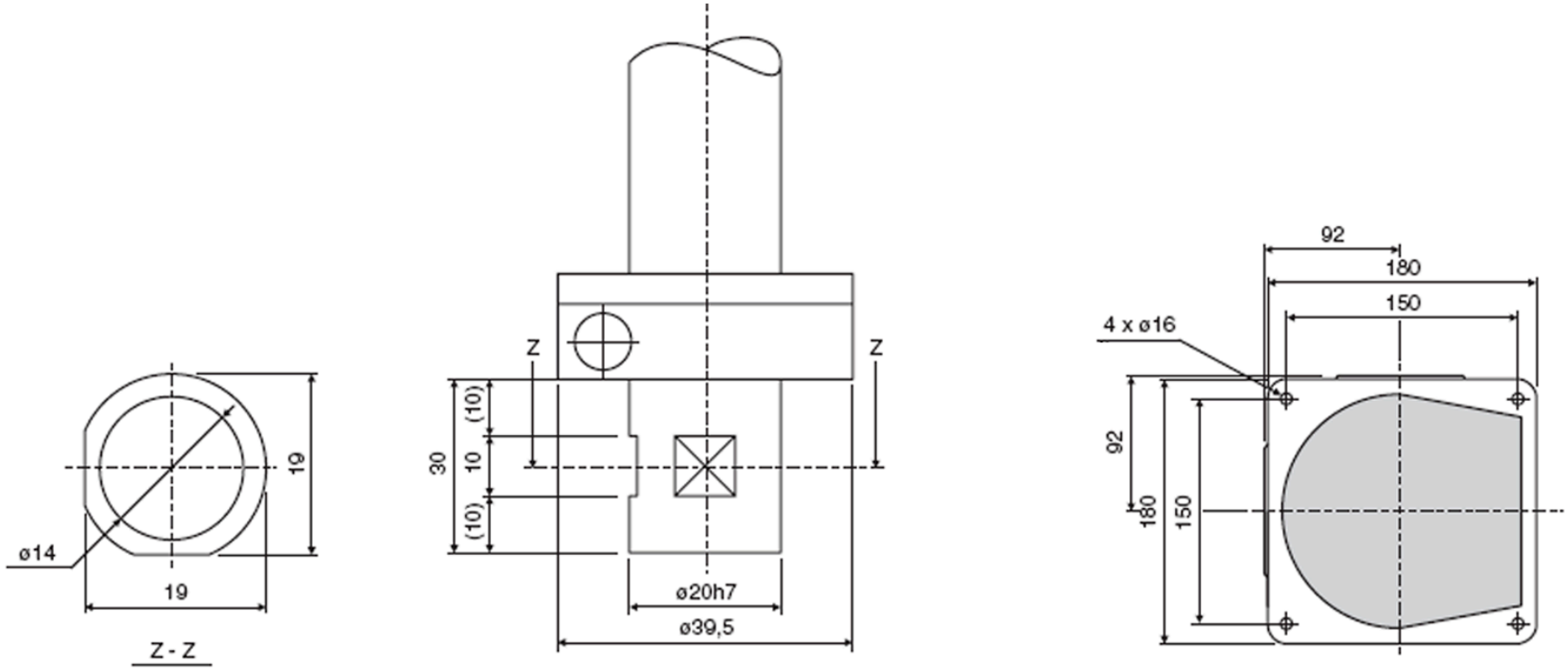
	Fecha	Nombre	Comprobado
dibuj.	10.02.09	JFGALVEZ	
comp.	10.02.09	JFGALVEZ	

Cliente: UNIVERSIDAD DE ELCHE
ROBOT RH-5AH

Dibujo Nr. 011

Masa	
Escala	
Hoja Nr.	11
de	13
pos.	

MITSUBISHI ELECTRIC



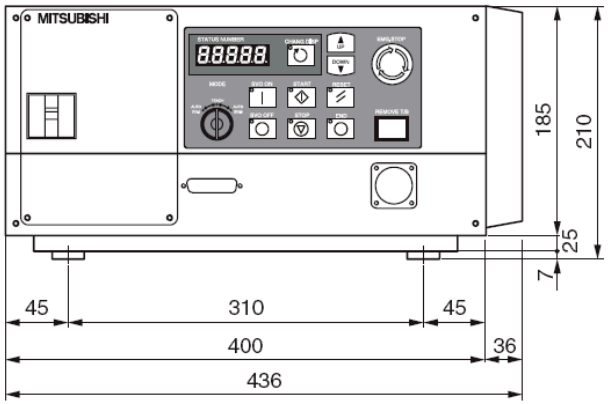
	Modificaciones	Fecha	Nombre		Fecha	Nombre	Comprobado
a				dibu.	10.02.09	J.FGALVEZ	
b							
c				comp.	10.02.09	J.FGALVEZ	
d							

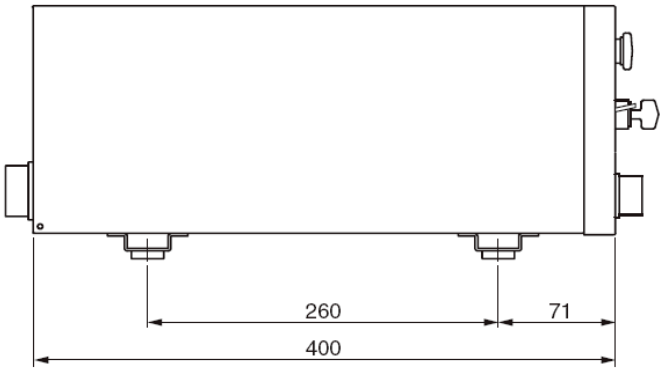
Cliente: UNIVERSIDAD DE ELCHE
ROBOT RH-5AH

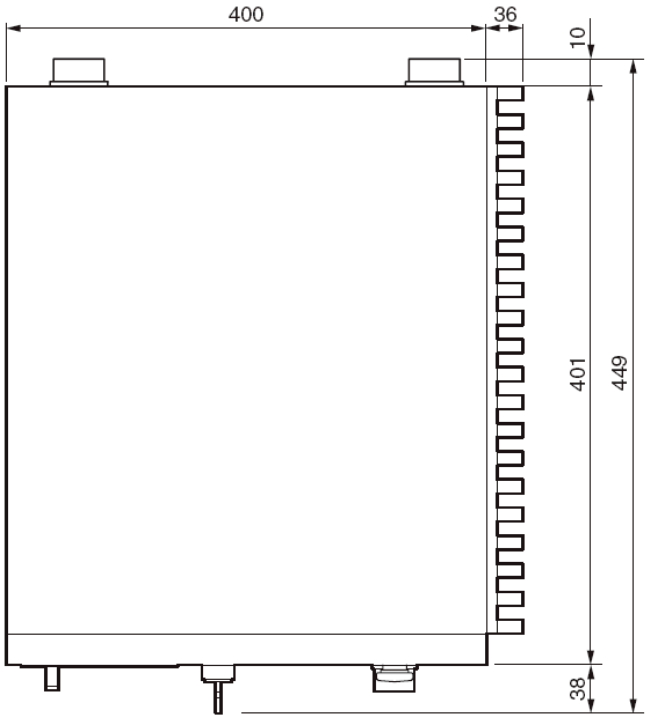
Dibujo Nr.: 012

sitio
Hoja Nr. 11
de 13 pgs.

MITSUBISHI ELECTRIC







	Modificaciones	Fecha	Nombre		Fecha	Nombre	Comprobado
a				dibu.	10.02.09	JFGALVEZ	
b							
c				comp.	10.02.09	JFGALVEZ	
d							

Cliente: UNIVERSIDAD DE ELCHE
CONTROLADORA ROBOT RH-5AH

Dibujo Nr.: 013

sitio
Hoja Nr. 11
de 13 pgs.

MITSUBISHI ELECTRIC