INTEGRACIÓN DE UN BRAZO ROBOT EN UNA ESTACÍON DE ALMACENAMIENTO DE UN PRODUCTO FINAL

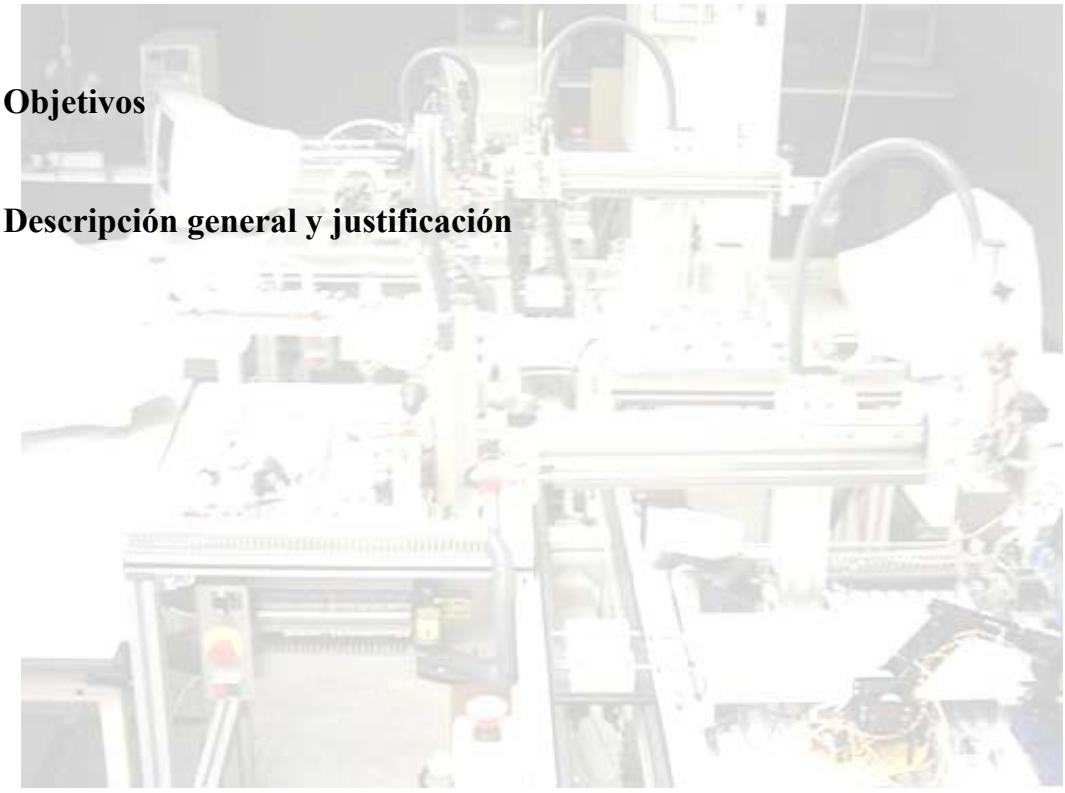
Ingeniero Técnico Industrial, Esp. Electrónica

Nº Registre: 1432/1431

Fernando García Fernández

Raúl Ortiz Campos

INDICE



- Objetivos

- Descripción general y justificación

1- INTRODUCCION

1.1- Objetivos

La finalidad de este proyecto es la integración de un brazo robot controlado por una micro controladora en un proceso automatizado. Dicho brazo irá situado en el punto final de todo el proceso, con la finalidad de almacenar todos los elementos de salida que nos suministra un manipulador.

Este sistema será supervisado por un PC, el cual irá monitorizando el proceso en cuestión. La comunicación entre estos tres elementos se realizará mediante una red Ethernet.

1.2- Descripción general y justificación

El laboratorio de automatización industrial dispone de una célula de fabricación flexible. Dicha célula consta de cinco estaciones distribuidas a lo largo de una cinta transportadora (transfer de producto). Cada estación dispone de un PC los cuales se encuentran interconectados mediante una red de área local (LAN) pudiéndose acceder desde un PC concreto a los otros y compartir cualquier tipo de información entre ellos.

La funcionalidad de la célula flexible es la de un proceso de empaquetado y marcado de medallas de plástico o metal. Este proceso se realiza en las diferentes estaciones de trabajo, configurándose un sistema de producción flexible, donde el transfer es el elemento que transportara los elementos a cada estación.

Antes de que se produzca el transporte de los elementos, se realizara la elección del tipo de medalla por medio del código binario controlado por la estación del transfer.

Código 1	Código 0	Pieza seleccionada
0	0	Medalla de aluminio
0	1	Medalla de plástico
1	0	Llavero de aluminio
1	1	Llavero de plástico

Esta codificación del transportador de piezas o carro, nos permite seleccionar un tipo de pieza entre los cuatro diferentes que disponemos. Una vez codificado el carro, pasará a la primera estación que tiene como función la de introducir una caja vacía dentro del carro. Cuando este proceso se realiza correctamente, el carro pasa a la segunda estación. Esta estación hace una lectura del código binario del carro y dependiendo del que sea, introducirá en la caja un tipo u otro de pieza de entre las cuatro en que disponemos en los dispensadores.

Una vez introducida la pieza, el carro pasa a la tercera estación. En esta estación, se tapa la caja con su debida marca para poder diferenciar de una forma visible y externa el tipo de pieza que contiene la caja ya precintada.

Para acabar, el carro llega a la ultima estación donde dependiendo de su codificación, el manipulador dejara la pieza en una posición u otra.

La quinta estación consiste en un almacén del producto acabado. Dicho almacén está muy limitado ya que es de tipo lineal con una capacidad de almacenamiento muy reducida.

Cómo material complementario de prácticas, el departamento d'ESAII adquirió un robot de tamaño reducido y se nos propuso utilizarlo para complementar la quinta estación, aumentando la capacidad del almacén. Debido a las reducidas capacidades del robot, al final se decidió que la tarea asignada al robot seria la de almacenar en palets todos los elementos de salida del proceso. De esta forma se consigue dotar la célula de un proceso completo desde la elaboración, almacenamiento y paletización para su posterior distribución.

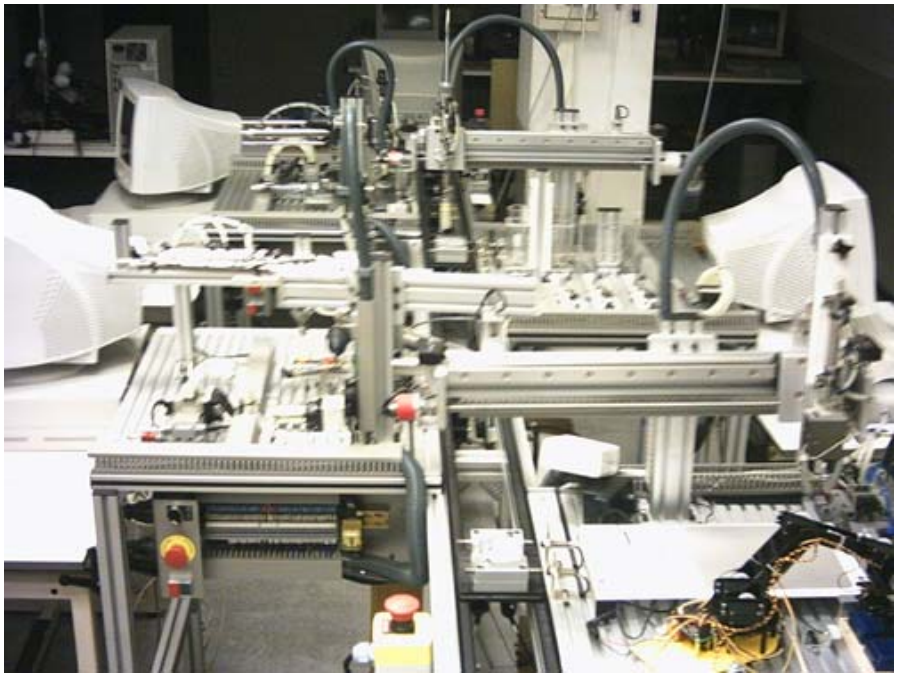


Figura 1.1 Vista de la estación

- **Descripción actual de la maquina**

- **Descripción del Sistema de control y programación**

- **Descripción de las conexiones**
- **Programación**
- **Contado rápido**
    - **Introducción**
    - **Esquema del conexionado**
    - **Programación**
    - **Selecciones de contador de alta velocidad**
    - **Rango de contaje**
    - **Proceso**
    - **Métodos de Reset**
    - **Contaje de interrupción del contador alta velocidad**

2- DESCRIPCION ACTUAL DE LA ESTACION DE TRABAJO

La estación actual consta de una máquina y un sistema de control. El sistema de control disponible es un autómata programable OMRON CPM1, por lo tanto dicho autómata será el utilizado para gobernar la máquina.

2.1- Descripción actual de la maquina

La estación de trabajo (Figura 2.1) consta de diferentes elementos:

- Manipulador.

  El manipulador está constituido por un sinfín con un movimiento horizontal que desplaza un brazo. Este carro consta de un cilindro neumático que realiza un movimiento vertical. Al final del cilindro se dispone de unas pinzas neumáticas que permiten coger objetos.

  El sinfín, a su vez, contiene un encoder que permite registrar la posición exacta del brazo.

Fig. 2.1 Estación de trabajo

- Cinta transportadora.

La cinta transportadora suministra un carro que contiene el objeto a almacenar y que posteriormente recogerá el manipulador.

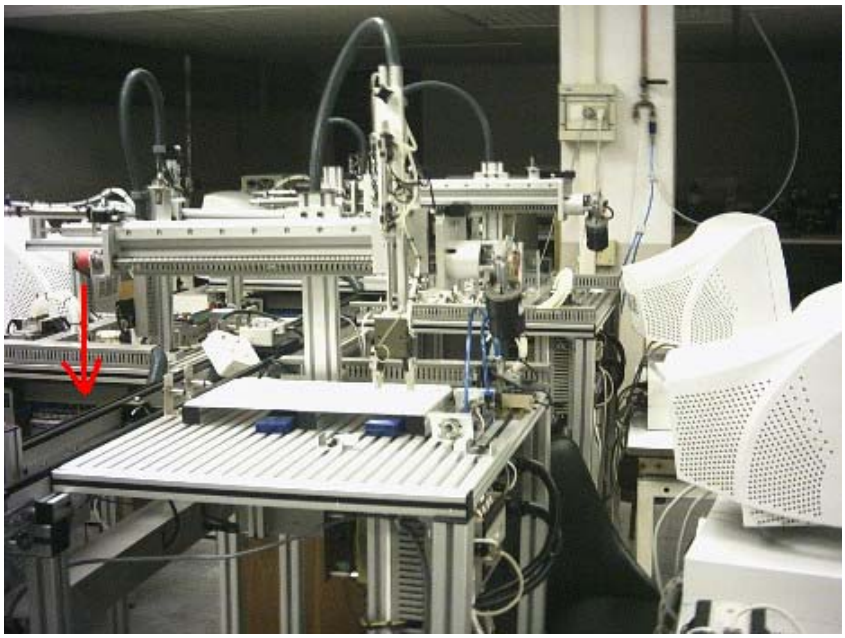


Fig. 2.2 Cinta transportadora

- Botonera.

La botonera esta constituida por un selector que nos permite seleccionar el modo de funcionamiento de la máquina: manual o automático, un pulsador de emergencia y dos pulsadores (marcha y paro).

Fig. 2.3 Botonera

Las especificaciones de funcionamiento actual de la estación se describen a continuación.

- El funcionamiento de la estación depende de la botonera de control, donde diferenciamos dos grandes estados, el funcionamiento automático y el manual.

En la posición automática, podemos poner la estación en funcionamiento con el botón de marcha y nos realizara la siguiente secuencia:

- El proceso se inicia cuando la cinta transportadora suministra el objeto en la posición inicial. El objeto es transportado mediante un transportador de piezas, este está codificado binariamente. La codificación nos indica el contenido de la caja que llega a la estación. Una vez detectada la presencia del elemento transportador se activa el sinfín haciendo que desplace el carro desde su posición inicial a la de recogida del objeto. Seguidamente el cilindro neumático descienda, cierre la pinza, y ascienda para que se produzca el almacenamiento de la pieza, esta pieza se deposita en un sitio fijo en función de la codificación del transportador. En el caso en que se active parada, el proceso finaliza el ciclo.

- Cuando la estación pasa a la posición manual del selector, entra inmediatamente a trabajar en modo paso a paso. Al pulsar el pulsador de marcha hace que la estación realice una operación de la secuencia de funcionamiento y se pare hasta una nueva orden del pulsador. En este caso el pulsador de parada queda inhabilitado.

- Para el caso contrario, al encontrarnos en modo manual (paso a paso) y pasar a modo automático independientemente de si nos encontramos con el proceso en marcha o en paro en un estado, la estación pasara inmediatamente a trabajar de manera automática sin ningún tipo de orden previa como pulsar el botón de marcha.

- Al pulsar el botón de parada de emergencia, sin importar el estado del selector, la estación dejara de funcionar quedando totalmente bloqueada y sin responder a ningún tipo de señal de control. Una vez corregido el motivo de la parada de emergencia y desactivado el pulsador, la estación pasa a condiciones iniciales.

- Seguidamente al presionar el pulsador de marcha, la estación volverá a la posición determinada como posición de emergencia, dejara la pieza en caso de que tenga y volverá a la posición inicial para poder seguir su funcionamiento dependiendo del estado del selector (manual / automático).

2.2 - Descripción del Sistema de control y programación

En nuestra estación de trabajo, disponemos de un PLC Omron modelo CPM1A (CPU-40) donde la conexión entre autómata y los elementos de información se realizara mediante una conexión punto a punto.

2.2.1 - Descripción de las conexiones

Tenemos una serie de entradas enlazadas ya previamente a unas direcciones de dicho PLC al igual que unas salidas que actuaran sobre unos actuadores que modifican el estado del proceso. Este conjunto de entradas y salidas ya las disponemos con unas direcciones específicas y son las siguientes:

ENTRADAS		
DIRECCION	**SIMBOLO**	**DESCRPICION**
000.00	Z4-A	A posición carro (salida A del enconder incremental)
000.01	Z4-B	B posición carro (salida B del enconder incremental)
000.02	Z4-C	C posición carro (salida C del enconder incremental)
000.04	SPE4.1	Pulsador de emergencia
000.05	SP4.2	Pulsador de parada
000.06	SP4.1	Pulsador de marcha
000.07	SS4.1	Selector manual-automático
000.09	DT4.1	Detector eje vertical arriba
000.10	DT4.2	Detector eje vertical abajo
000.11	DT4.3	Sensor inductivo posición inicial del cargador
001.00	DT4.4	Sensor inductivo posición máxima del cargador
001.01	DT4.5	Detector de pinza pieza sujeta
001.02	DT4.6	Sensor inductivo código 1
001.03	DT4.7	Sensor inductivo código 2
001.04	DT4.8	Sensor Óptico palet
001.05	DT4.9	Posición 1 expulsar palet
001.06	DT4.10	Posición 2 expulsar palet

SALIDAS		
DIRECCION	**SIMBOLO**	**DESCRPICION**
10.01	YV4.1	Electroválvula avanzar eje vertical
10.02	YV4.2	Electroválvula cerrar pinza
10.03	KM4.1	Conector avanzar manipulador
10.04	KM4.2	Conector retroceder manipulador
10.05	YV4.3	Avance expulsión
10.07	YV4.4	Retroceso expulsión

2.2.2 – Programación

Una vez disponemos de las entradas y salidas ya especificadas, se accederá al control del brazo mediante dichas direcciones de nuestro PLC. Pero para complementarlas adecuadamente habrá que acceder al programa interno del PLC para la configuración de sus memorias internas y modificación del lenguaje de programa. Esta modificación de programa interno lo realizaremos mediante el Cx-Programmer que trabaja con un tipo de red SYSMAC WAY.

Para una comunicación óptima tendremos que configurar los elementos tal como mostramos a continuación. Empezaremos por abrir el programa Cx-Programmer y cargar un nuevo proyecto, en la ventana que nos aparece la configuramos de la siguiente manera:

- **Nombre de Dispositivo:** Es el nombre que le daremos al proyecto, así que este campo es indiferente el nombre que le demos.
- **Tipo de Dispositivo:** En este campo ponemos el modelo de PLC con el que trabajaremos, en nuestro caso es el que marcamos (CPM1A).
- **Tipo de Red:** Marcamos SYSMACWAY como indicamos en la Fig. 2.4.

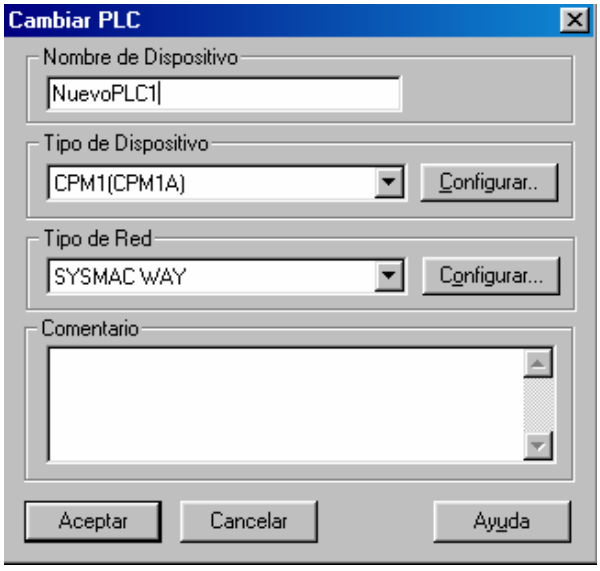


Fig. 2.4

A continuación marcaremos en el botón de configurar que nos queda justo al lado del "Tipo de Red" y accedemos a otra ventana que nos dispondremos a configurar tal y como indicamos en la siguiente imagen.

Nos iremos directamente a la pestaña de UNIDAD puesto que las otras no se verán modificadas. Una vez dentro solo tendremos que tener en cuenta la opción de N. Puerto en el apartado de conexión. Aquí ponemos el puerto al cual se conectara el PLC, en nuestro caso es el puerto **COM2**. En los demás campos se debe mantener su valor por defecto, apretamos aceptar y ya podemos acceder a la plantilla para efectuar la modificación de programa necesaria para nuestro PLC.

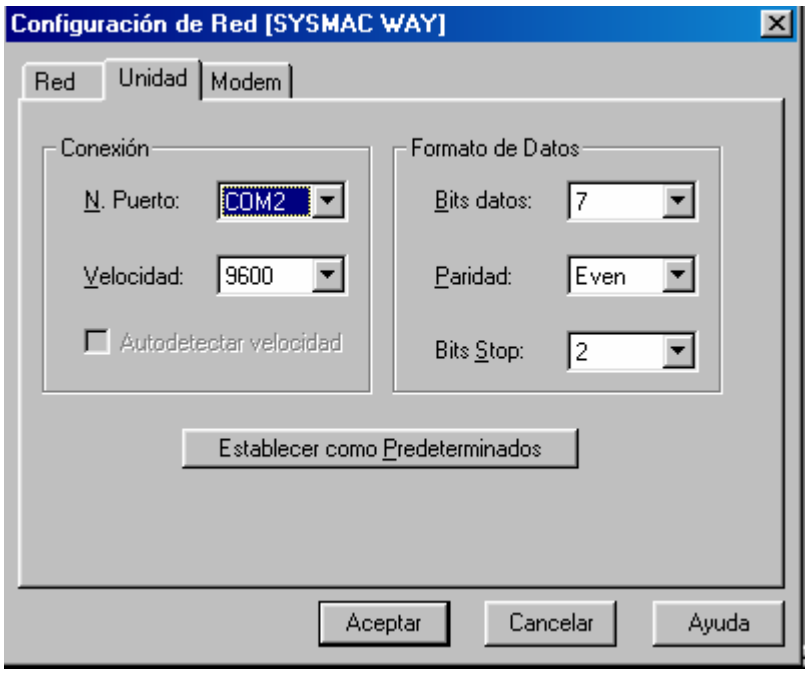


Fig. 2.5

2.2.3- Contado rápido

2.2.3.1-Introducción

Para conocer exactamente la posición del manipulador en todo momento y de este modo detener su desplazamiento en la posición exacta en la cual deseemos que se detenga disponemos de un encoder. Este encoder está fijado directamente al sinfín del carro del manipulador y, durante el movimiento de este, el encoder genera una serie de impulsos que el PLC tendrá que detectar.

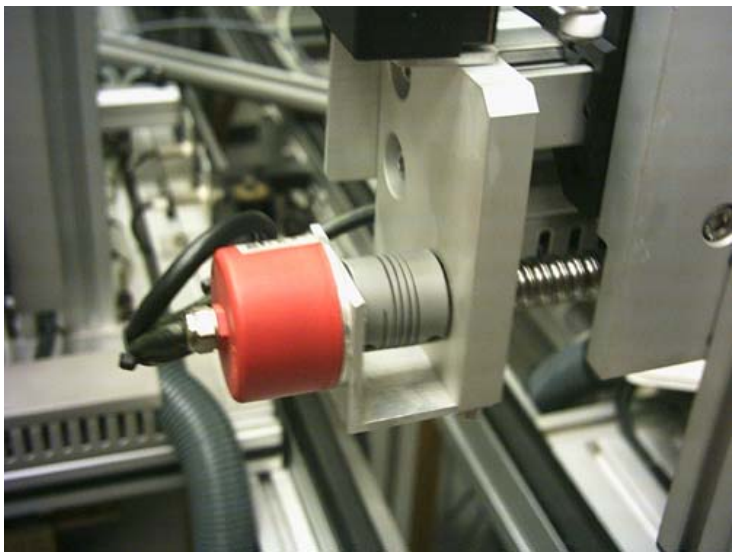


Fig. 2.6 Foto del encoder

Una vez que el PLC recibe los impulsos del encoder, por medio de un contador se registran dichos impulsos para poder interpretarlos y de este modo trabajar con los datos obtenidos del encoder. Pero debido a la alta frecuencia a la que el PLC recibe los impulsos, las entradas configuradas como normales no pueden realizar el contado. Por ello se deben configurar las entradas para contado rápido que el PLC de Omron CPM1-A dispone, y de este modo poder registrar los diferentes impulsos a alta frecuencia que el encoder nos suministra.

2.2.3.2 – Esquema del conexionado

El encoder tiene tres cables que van conectados en las entradas para contado rápido del PLC que se muestran a continuación con las características en función del modo que utilicemos (en nuestro caso el modo es incremental):

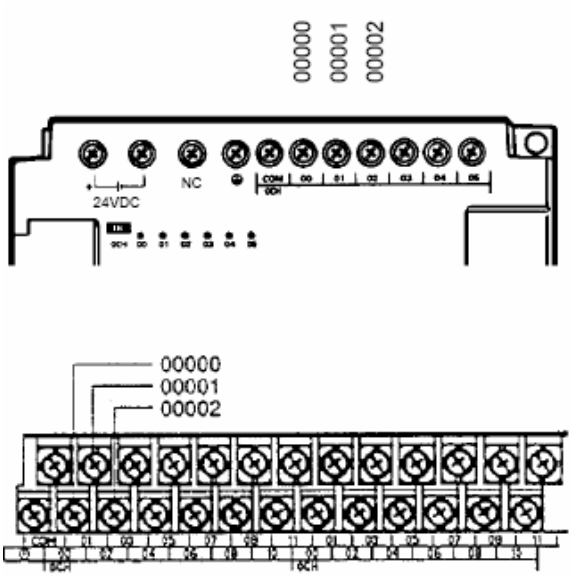


Fig.2.7

Modo	Funciones de entrada	Método de entrada	Frecuencia de contaje	Rango de contaje	Métodos de control
Reversible	00000: Entrada fase A 00001: Entrada fase B 00002: Entrada fase Z	Fase diferencial, 4× entradas	2.5 kHz máx.	–32767 a 32767	Control de valor objeto: Se pueden registrar hasta 16 valores objeto y números de subrutina de interrupción.
Incremental	00000: Entrada contaje 00001: Ver nota. 00002: Entrada reset	Entradas individuales	5.0 kHz máx.	0 a 65535	Control de comparación de zona: Se pueden registrar hasta 8 grupos de valores de límite superior, valores de límite inferior y números de subrutina de interrupción.

Nota En modo incremental, esta entrada (00001) se puede utilizar como entrada normal.

Fig. 2.8

2.2.3.3-Programación

-Activar el contador

Para poder trabajar con los contadores rápidos hay que activar el contador rápido por programa en el CX-Programmer e indicar que utilizamos un contador incremental. Los pasos son los siguientes:

1. Abrir la pestaña Setup

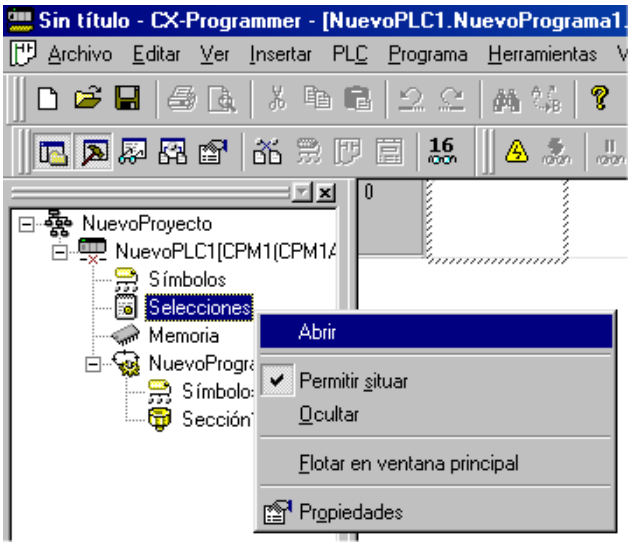


Fig. 2.9

2. Activar el contador y seleccionar el modo del contador en incremental tal y como indicamos en la siguiente figura.

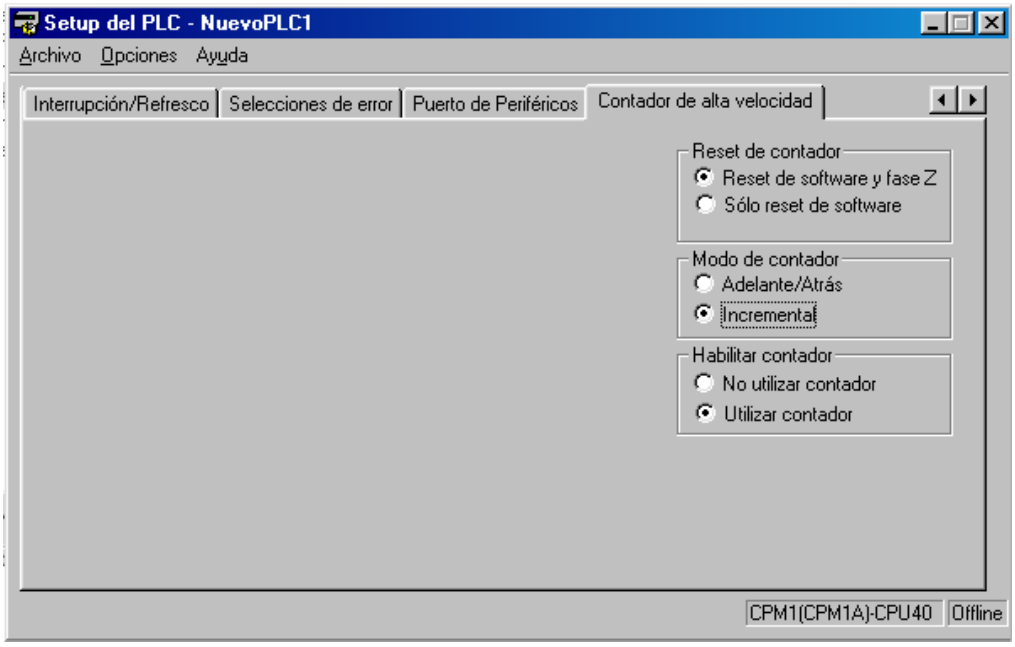
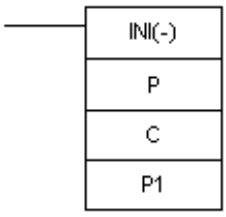


Fig.2.10

-Instrucciones necesarias para la programación:

Para la utilización del contador rápido en el resto del programa es necesario utilizar las tres instrucciones descritas a continuación.

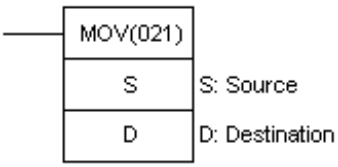


Descripción: Controla la operación del contador de alta velocidad y detiene la salida de pulsos.

Variaciones: Diferencial Ascendente

El puerto (P) especifica el contador de alta velocidad o la salida de pulsos a controlar.

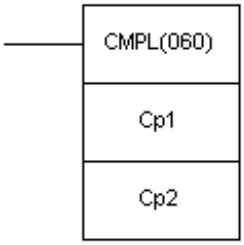
La función de INI(--) está determinada por el dato de control, C (P1 y P1+1 contienen el nuevo PV del contador de alta velocidad cuando se cambia aquel).



Descripción: Copia el contenido de S a D.

Variaciones: Diferencial Ascendente

Rangos: S: Canal fuente IO, AR, DM, HR, TC, LR, #
D: Canal destino IO, AR, DM, HR, LR



Descripción: Une el contenido hexadecimal de 4 dígitos de Cp1+1 con el de Cp1 y el de Cp2+1 con el de Cp2 para crear dos números hexadecimales de 8 dígitos, Cp+1, Cp1 y Cp2+1, Cp2. Luego compara los dos números de 8 dígitos y el resultado se envía a los indicadores GR, EQ y LE en el área de SR.

Variaciones: Diferencial Ascendente

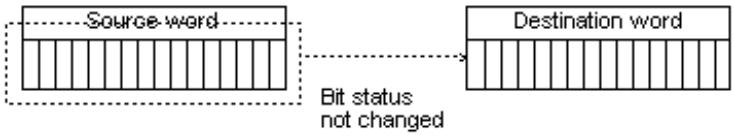


Fig.2.11

Rangos:

Cp1: 1er canal del primer par de canales a comparar IO, AR, DM, HR, TC, LR

Cp2: 1er canal del segundo par de canales a comparar IO, AR, DM, HR, TC, LR

2.2.3.4 - Selecciones de contador de alta velocidad

Cuando se utilice la función de contador de alta velocidad del CPM1A, hay que efectuar las siguientes selecciones en DM 6642.

DM 6642 Bits	Función	Selecciones		
		Incremental	Reversible	No utiliz.
00 a 03	Selecciona el modo: 0: Reversible 4: Incremental	4	0	0 ó 4
04 a 07	Selecciona método de reset: 0: Fase Z + reset de software 1: Reset de Software	0 ó 1	0 ó 1	0 ó 1
08 a 15	Selecciona el contador: 00: Contador no utilizado. 01: Contador utilizado.	01	01	00

2.2.3.5 - Rango de contaje

El contador de alta velocidad del CPM1/CPM1A utiliza operación lineal y el contaje (valor presente) se almacena en SR 248 y SR 249. (Los cuatro dígitos de mayor peso se almacenan en SR 248 y los cuatro de menor peso en SR 249.)

Modo	Rango de contaje
Reversible	De F003 2767 a 0003 2767 (de –32,767 a 32,767) El dígito de la izquierda en SR 248 indica el signo. F es negativo, 0 es positivo.
Incremental	De 0000 0000 a 0006 5535 (de 0 a 65,535)

Se producirá un overflow si el contaje excede el límite superior del rango de contaje y un underflow si el contaje es menor que el límite inferior del rango de contaje.

Error	Incremental	Reversible	Valor presente
Overflow	Se produce cuando se supera 65,535.	Se produce cuando se supera 32,767.	0FFF FFFF
Underflow	---	Se produce cuando desciende de –32,767.	FFFF FFFF

2.2.3.6- Proceso

De un encoder de impulsos se pueden tomar dos tipos de señales. El modo de contaje utilizado para el contador de alta velocidad dependerá del tipo de señal. El modo de contaje y el modo de reset se seleccionan en DM 6642; estas selecciones serán efectivas cuando se conecte la alimentación o se inicie la operación del PLC.

Modo Adelante/Atrás (reversible):

Para entradas se utilizan una señal de dos fases 4X de fase diferencial (fase A y fase B) y una señal de fase Z. El contaje es ascendente o descendente según las diferencias en las señales de 2 fases.

Modo incremental:

Para entradas se utilizan una señal de impulso de una fase y una señal de reset de contaje. El contaje es ascendente de acuerdo con la señal de una fase.

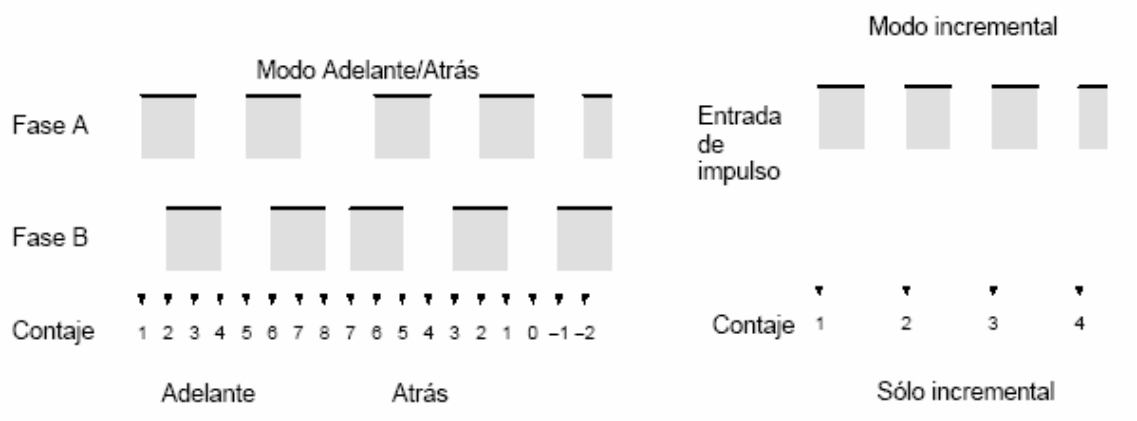


Fig. 2.12

Nota - El contador deberá restaurarse automáticamente cuando se arranque de nuevo, por lo que hay que utilizar uno de los métodos de la siguiente sección. El contador se resetea automáticamente cuando se arranca o para la ejecución del programa.

Las siguientes transiciones de señal se tratan como impulsos adelante (ascendente). Flanco de subida fase A -- flanco de subida fase B -- flanco de bajada fase A -- flanco de bajada fase B. Las siguientes transiciones de señal se tratan como impulsos inversos (descendente). Flanco de subida fase B -- flanco de subida fase A -- flanco de bajada fase B -- flanco de bajada fase A.

El rango de contaje es de --32,767 a 32,767 para modo Adelante/Atrás y de 0 a 65,535 para modo incremental. Las señales de impulso se pueden contar hasta 2.5 kHz en modo Adelante/Atrás y hasta 5.0 kHz en modo incremental.

El modo Adelante/Atrás siempre utiliza entrada de fase diferencial 4X. El número de contajes por cada revolución del encoder será 4 veces la resolución del contador. Seleccionar el encoder basándose en los rangos de contaje permisibles.

2.2.3.7 - Métodos de Reset

Para resetear el PV (ponerlo a 0) del contador, se puede utilizar cualquiera de los dos métodos siguientes.

1- Señal de fase Z + reset software:

El PV se restaura cuando la señal de fase Z (entrada de reset) se pone a ON después de que el bit de reset de contador de alta velocidad (SR 25200) se ponga a ON.

2- Reset Software:

El PV se restaura cuando el bit de reset del contador de alta velocidad (SR 25200) se pone a ON.

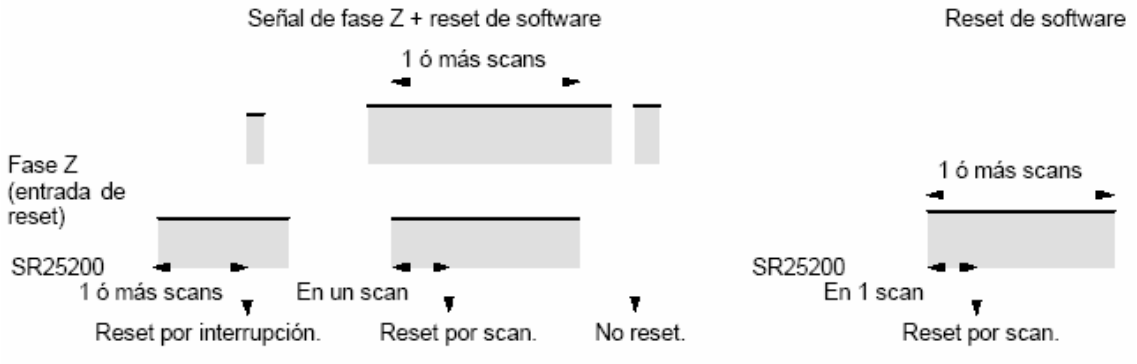


Fig. 2.13

Nota - El bit de reset de contador de alta velocidad (SR 25200) se refresca una vez cada scan, por lo que para que la lectura sea fiable debe estar por lo menos un scan completo en ON.

La "Z" en "fase Z" es una abreviación de cero, en inglés "Zero." Se trata de una señal que indica cuándo ha dado una vuelta completa el encoder.

2.2.3.8 - Contaje de interrupción de contador de alta velocidad

Para las interrupciones de contador de alta velocidad se utiliza, en vez de un "contaje alcanzado" una tabla de comparación. El chequeo de contaje se puede hacer por cualquiera de los dos métodos siguientes. En la tabla de comparación se salvan las combinaciones de rutina de interrupción y las condiciones de comparación (para comparar con el PV).

Valor objeto: En la tabla de comparación se salvan hasta 16 condiciones de comparación (valores objeto y direcciones de contaje) y combinaciones de rutina de interrupción. Cuando el PV del contador y la dirección de contaje concuerdan con las condiciones de comparación, se ejecuta la rutina de interrupción especificada.

Rango de comparación: En la tabla de comparación se salvan ocho condiciones de comparación (límites superior e inferior) y combinaciones de rutina de interrupción. Cuando el PV es mayor que o igual que el límite inferior y menor que o igual que el límite superior, se ejecuta la rutina de interrupción especificada.

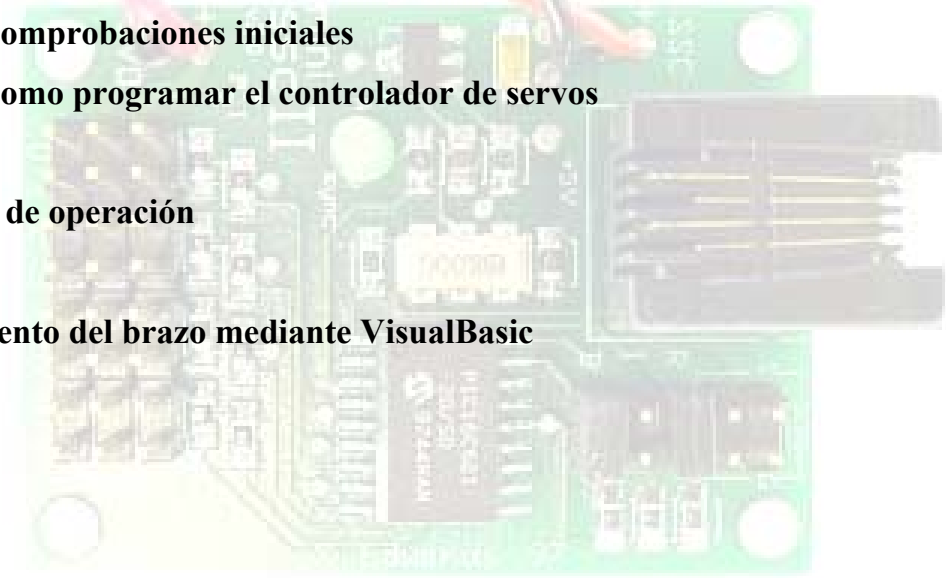
- Descripción del brazo robot

- Configuración de la controladora

- **Conectores y jumpers de configuración**
- **Como configurar el controlador de servos**
- **Como conectar el controlador de servos Mini SSC II**
- **Comprobaciones iniciales**
- **Como programar el controlador de servos**

- Espacio de operación

- Movimiento del brazo mediante VisualBasic



3- BRAZO ROBOT Y SU CONFIGURACION

3.1 Descripción del brazo robot

El brazo robot utilizado es el modelo S300114 de 6 Ejes compuesto por una parte mecánica y una parte electrónica (controladora).

· ***Mecánica.*** Todas las piezas del chasis están hechas en policarbonato negro de 3 mm. Consta de 7 servos Hitec de alta calidad y potencia incluyendo dos micro servos para la apertura y rotación de la pinza manipuladora con lo que se consiguen unos movimientos más precisos.

Fig. 3.1

· ***Electrónica.*** La parte electrónica, consiste en una controladora tipo MINI SSC S310165 que permite controlar desde un puerto serie RS232 hasta 8 servos independientes. Este circuito SSC recibe las ordenes de movimiento de un PC o cualquier otro microcontrolador y genera los pulsos de control de posición de los servos.

Fig. 3.2

3.2 Configuración de la controladora

Una parte muy importante del brazo robot es la controladora de que dispone, esta nos permite hacer una comunicación entre el brazo y el PLC.

3.2.1- Conectores y jumpers de configuración

La figura 3.3 muestra la distribución de la placa de circuito del controlador de servos Mini SSC II con la posición de los conectores y los jumpers de configuración.

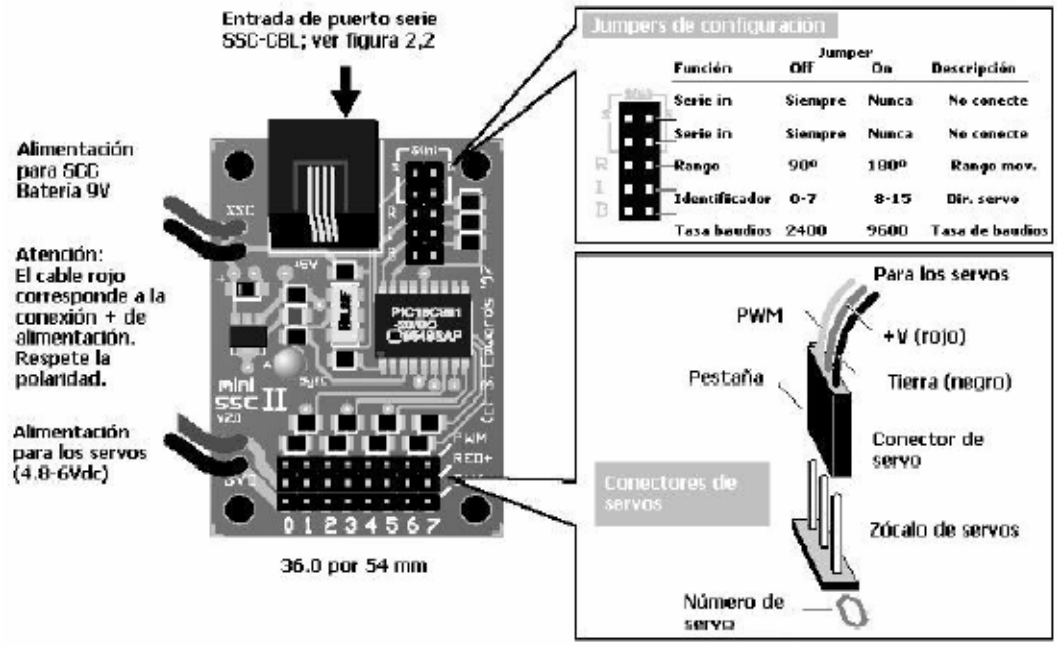


Fig. 3.3 Diseño de la placa de circuito de Mini SSC II

Importante

- No invertiremos la polaridad de la alimentación, ni siquiera de manera temporal, ya que podría destruir los componentes electrónicos.
- El voltaje máximo es de 10 voltios en el terminal de entrada SCC 9V, ya que podría dañar definitivamente la unidad o acortar la vida del mismo.

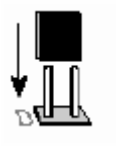
- El conector jack se puede utilizar sólo como entrada serie. Conectar el jack a la línea telefónica podría dañar la unidad.

3.2.2- Como configurar el controlador de servos

La configuración por defecto del controlador Mini SSC II' es la siguiente (ningún jumper conectado):

Velocidad: 2.400 baudios • Servos 0 a 7 • Rango de movimiento = 90º

Para modificar la configuración, instalaremos un jumpers en los pines adecuados como se indica en la ilustración siguiente. Los cambios serán efectivos la próxima vez que activemos el controlador SCC.



A continuación, encontramos los detalles de las opciones de la configuración:

- **Rango:** Sin jumper en R, el controlador Mini SSC II controla servos con un rango de movimiento del 90º. Las posiciones de los servos están expresadas en unidades dentro del rango 0 -254, por lo que cada unidad corresponde con un cambio de 0.36º en la posición de los servos. Con un jumper en R, el controlador Mini SSC II controla servos más allá de los 180º, cada unidad corresponde con un cambio de posición de 0.72º.

Nota acerca del rango de movimiento:

Algunos servos no pueden moverse con un ángulo de 180º. Estos podrían atascarse si se utilizan posiciones inferiores a 50 o superiores a 200 con el controlador Mini SSC II en modo 180º. Por este motivo, comprobaremos el funcionamiento de los

servos nuevos o distintos a los habituales antes de utilizarlos con el modo 180º del controlador Mini SSC II.

Moveremos el servo unas cuantas unidades de una vez hacia el final de su trayecto de movimiento para comprobar si se bloquea.

Si lo hiciera, habrá que utilizar el servo únicamente en modo de 90º, o restringir los valores de posición para definir un rango de movimiento seguro.

Tendremos en cuenta que los servos han sido diseñados para un rango de movimiento de 90º. De esta manera el obtener un rango de 180º es como hacer trampas. No hay nada erróneo en un servo que no pueda ofrecerle un rango de 180º.

- **Identificación:** Sin jumper en I, las direcciones de los servos coincidirán como los números impresos en los zócalos de los servos de 0 a 7. Con un jumper I, el controlador Mini SSC suma 8 direcciones, por lo que el servo conectado a "0" utilizará la dirección 8, conectado a "1" utilizará la 9 ... y conectado a "7" utilizará la dirección 15. Esta opción nos permitirá conectar dos controladores Mini SSC II al mismo puerto serie y controlar los servos de manera individual. Para ello debemos consultar la información relativa al control de varios controladores Mini SSC desde un mismo puerto serie.

- **Baudios:** Sin jumper en B, el controlador Mini SSC II recibe los datos a través del puerto serie a una tasa de 2400 baudios; con un jumper en B, la tasa de baudios es 9600. En cualquier caso, los datos deberían enviarse como 8 bits de datos, sin paridad, 1 (o más) bit(s) de parada; N81 abreviado. Además, se deberían invertir los datos del mismo modo que vienen de uno estándar.

3.2.3- Como conectar el controlador de servos Mini SSC II

- **Alimentación de servos:** Conectaremos el alimentador de los servos (4.8 a 6 V DC) a los cables rojos (+) y negros (–) marcados como SVO en la placa de circuitos del Mini SSC II.

  No invertiremos la polaridad de la alimentación, ni siquiera de manera momentánea, ya que podría dañar los servos. Como fuente de alimentación, utilice 4 pilas alcalinas tipo C o D. Podremos utilizar tal y como hemos hecho en nuestro caso, una alimentación AC, cogemos un alimentador regulado lineal (no conmutado) de 5Vdc a 1A (o superior).

- **Alimentación del controlador Mini SSC II:** Conectamos una pila de 9V al pack de pilas. Podremos utilizar otra fuente de alimentación, para ello cortaremos el cable del pack de pilas y conectamos de 7 a 15Vdc a los cables, + al rojo y – al negro.

- **No se invertirá la polaridad + y –, ni siquiera de manera momentánea.** Si sucediera, se podrían dañar los componentes electrónicos. La entrada de alimentación del SSC está protegida contra cualquier inversión accidental de la batería de 9V. No obstante un voltaje de entrada superior podría deshabilitar esta protección.

- **Entrada por puerto serie:** El controlador de servos Mini SSC II requiere sólo dos conexiones con una placa de circuito impreso:
  - *Conexión de datos por puerto serie.*
  - *Señal de tierra.*

  Hay dos lugares en los que hacer estas conexiones; un conector “jack” de teléfono y un par de pines marcados como S(in) en el zócalo de configuración. No importa cual de ellos se utilice por eso seleccionaremos la conexión más cómoda.

En la siguiente figura, vemos cómo se debe conectar el cable SCC para utilizarlo junto con un conector DB9 de la placa de circuito impreso y otros ordenadores.

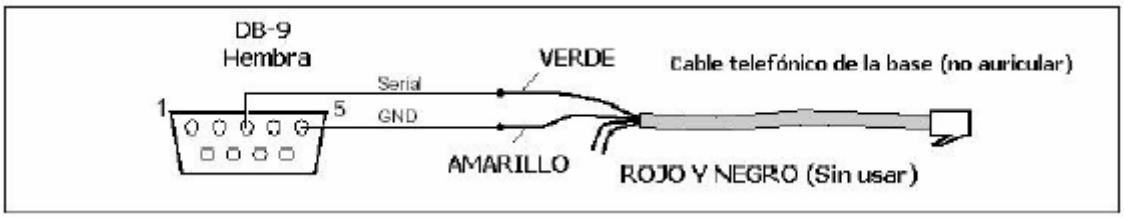


Fig. 3.4

A continuación vemos la figura 3.4 de como será el método de fabricación de este cable debido a que el adaptador DB9-a-modular, viene sin montar. Para montarlo, introducimos los terminales en las clavijas numeradas tal y como se indica en la imagen de abajo. Seguidamente meteremos el cable telefónico en la carcasa del adaptador. Podremos utilizar cualquier cable telefónico diseñado para la conexión con la clavija telefónica de pared. El único requisito es que el cable sea de cuatro hilos (con los cables amarillo, verde, rojo y negros todos conectados). Los cables de dos hilos que normalmente traen los teléfonos más baratos tienen sólo los cables rojo y negro. Estos cables no funcionarán.

En el caso de que se trabaje con puertos de 25 pines, se deberá utilizar un adaptador estándar de DB25-a-DB9 y uno de los cables que se ha descrito. Hoy en día este tipo de puertos de 25 pines ya casi no se utilizan.

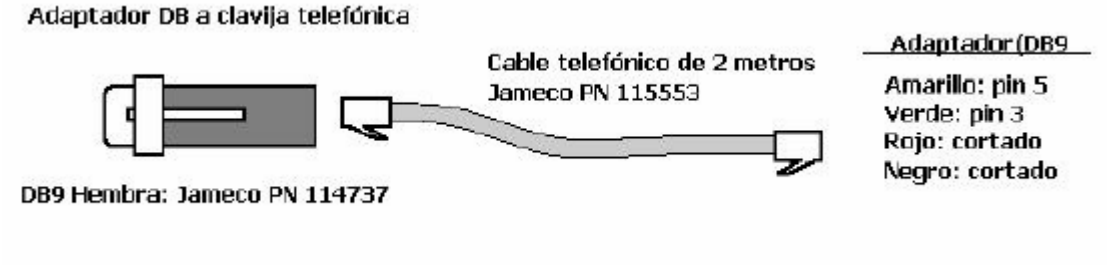


Fig. 3.5

3.2.4- Comprobaciones iniciales

Cuando se conecte por primera vez el controlador de servos Mini SSC II, se iluminará el LED Sync y se moverán todos los servos a la posición central. Una vez estén conectados los servos, la alimentación de los mismos y del controlador Mini SSC, los servos se moverán inmediatamente a su posición central. Si ya se encuentran en dicha posición central, entonces no se moverán mucho, pero se puede comprobar el funcionamiento correcto del controlador moviendo suavemente con los dedos el servo lejos de la posición central. El servo debería intentar resistir el movimiento. En caso contrario, volveremos a comprobar las conexiones de los servos y la alimentación.

Indicador verde: Al encender el sistema, el LED Sync de color verde indica no sólo que controlador Mini SSC II está recibiendo alimentación, sino que su microcontrolador está funcionando correctamente (ha pasado la iniciación de arranque). Si se enciende la luz verde, pero los servos no parecen estar respondiendo correctamente, es probable que haya algún error en las conexiones de los servos, la alimentación de los mismos, o conexiones del puerto serie, y NO en el controlador Mini SSC II propiamente dicho.

Comprobación del puerto serie: Para verificar que se han establecido correctamente las conexiones del puerto serie, ejecutaremos uno de los programas de prueba para puertos serie. Comprobaremos que la tasa de baudios del programa coincide con la configuración del Mini SSC II, y que el pin I/O o puerto comm seleccionado para la salida serie corresponde con el puerto al que está conectado el controlador de servos Mini SSC II.

3.2.5- Como programar el controlador de servos

Para ordenar a un servo que se coloque en una determinada posición es necesario enviar tres bytes con la tasa de baudios adecuada (2400 o 9600 baudios, dependiendo de la configuración del jumper B).

Estos bytes:

Byte 1	Byte 2	Byte 3
[marcador de sincronismo (255)]	[Nº de servo (0-254)	[posición (0-254)]

Estos bytes deben ser enviados como valores de bytes individuales, no como representaciones de texto de los números que podría escribir en un teclado. En el programa que hemos utilizado para crear la monitorización en el PC (VisualBasic), la función **CHR$** es la que nos convierte los números a bytes.

El indicador LED Sync LED de la placa del controlador Mini SSC II nos ayuda a depurar las rutinas de su puerto serie. Se iluminará de manera continua la primera vez que se alimente la placa y se mantendrá encendido hasta que se reciba la primera instrucción de tres bytes. Por lo tanto, el LED se iluminará sólo después de que se haya recibido un marcador de sincronización y una dirección de servo válidos. Se mantendrá encendido hasta que se reciba un byte de posición y después se apagará. Si el programa envía muchos datos al controlador Mini SSC II, aparentemente el LED se iluminará de manera continua, aunque en realidad estará parpadeando muy rápido.

3.3- Espacio de operación

Este brazo adquirido recientemente por el departamento de automatización, estará situado en la salida de la última estación. En un espacio muy reducido, se ha colocado dicho brazo con un conjunto de elementos que nos permitirán realizar una alimentación de palets con su desalojo correspondiente. A todos estos efectos hay que sumar el poco margen de error de que disponemos a la hora de realizar cualquier movimiento del brazo robot debido a que cualquier movimiento brusco haría variar la posición lo suficiente como para que el programa nos comunique un error y no podamos realizar el proceso de almacenamiento. Aunque el error más significativo es que en el brazo observamos un leve margen de error en sus movimientos, un error que se ve acrecentado al disponer de unas cajas muy grandes y una pequeña variación nos supone que el brazo no cogerá la caja.

Por tanto, el espacio en qué tendrá que trabajar el brazo robot será un rango comprendido entre la posición donde el manipulador deje la pieza (ya sea la posición A o B) y la posición donde está ubicado el palet (en cualquiera de sus cuatro posiciones).

3.4- Movimiento del brazo mediante VisualBasic

El encargado de decirle a la controladora del brazo robot las diferentes posiciones de los servos será el programa VisualBasic, el cual mediante la librería mscomm transferirá los tres bytes anteriormente explicados.

Al transferir los datos en teoría no debe haber ningún problema, pero a la práctica se observa que al enviar dos coordenadas a un mismo servo, este realiza movimientos rápidos y bruscos que desestabilizan el brazo robot y provoca que el movimiento teórico contabilizado por programa no corresponda con el movimiento real realizado por el brazo.

Para solventar este problema hay que dar un tiempo de delay al movimiento del brazo para que éste realice un movimiento más lento y así evitar cualquier movimiento brusco.

- Introducción del tiempo de delay mediante VisualBasic

Debido a que la controladora no permite controlar este tiempo de delay (o por lo menos el manual de la controladora no lo indica), ha de proporcionarlo el VisualBasic.

Esto se hace mediante la utilización de un WHILE que enviará una a una las diferentes posiciones con un incremento de +1 respecto el valor anterior. De este modo el brazo robot irá haciendo los diferentes movimientos uno a uno sin movimientos bruscos. La estructura será la siguiente:

```
x  =  valor actual del servo
y  =  valor donde queremos que vaya el servo
While   x < y   then
        x = x + 1
        Transferir x al brazo
End
```

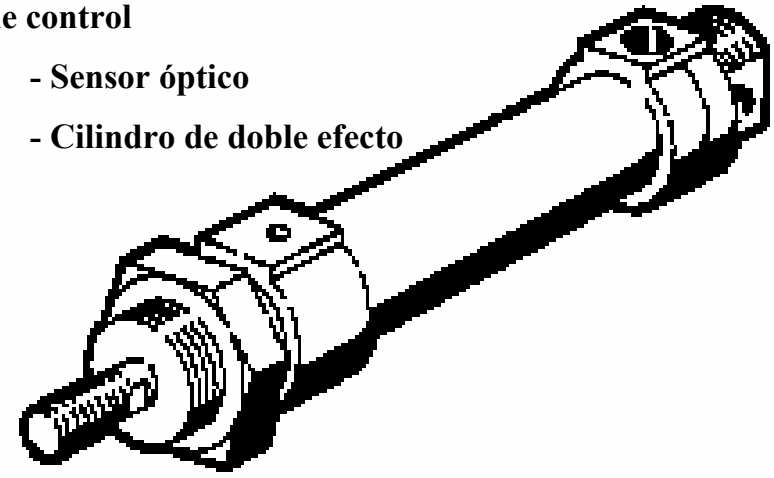
- Visión de la nueva estación

- Sistema de expulsión de palet

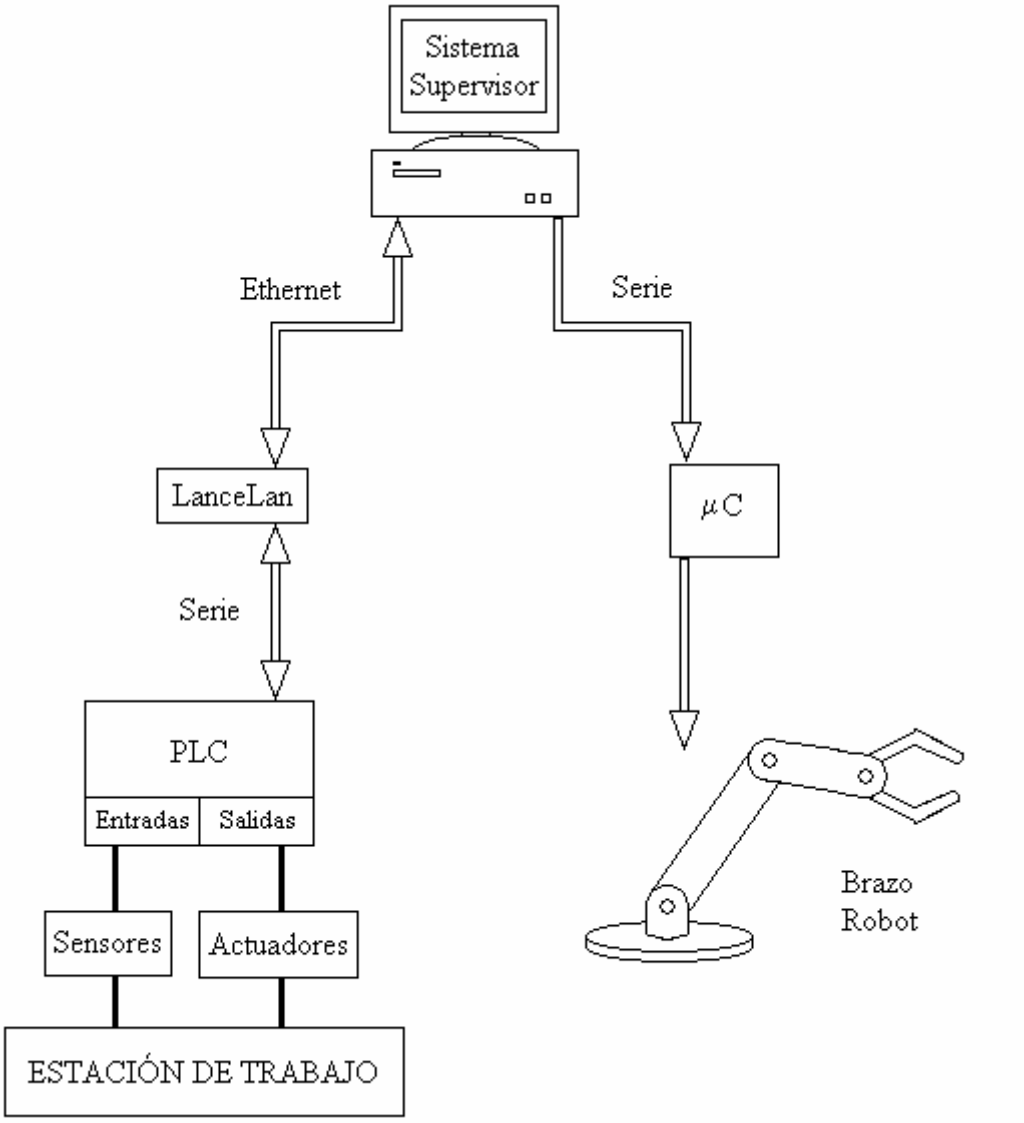
- **Actuador**
- **Cilindro de doble efecto**
- **Sensor óptico**

- Sistema de control

- **Sensor óptico**
- **Cilindro de doble efecto**



4- <u>DESCRIPCCIÓN DE LA ESTACIÓN MODIFICADA</u>



La estación modificada está controlada por un sistema supervisor que hace de intermediario entre el PLC y el brazo robot. La comunicación entre el sistema supervisor y el PLC se hace mediante Ethernet desde el ordenador hasta un modulo LanceLan que convierte la comunicación Ethernet a comunicación serie que se conectará al puerto RS-232C del PLC. Éste a su vez controlará los sensores y actuadotes de la estación de trabajo mediante las entradas y salidas de que dispone.

El sistema supervisor a su vez controla mediante una comunicación en serie la controladora del brazo robot. Esta comunicación es unidireccional desde el PC hasta la controladora. La controladora, a su vez, envía los impulsos necesarios a los diferentes servos del brazo robot para que ejecuten sus respectivos movimientos.

4.1- Visión de la nueva estación

Debido a que la antigua estación estaba inacabada, ya que era una estación para el almacenaje de las piezas y el manipulador únicamente dejaba las piezas en una posición que se consideraba como almacenadas, se ha ampliado el proceso con la instalación de un sistema provisto de un palet y un cilindro neumático para la expulsión de éste una vez lleno. El paso de la posición donde deja el manipulador la pieza hasta la posición exacta del palet se realiza mediante el brazo robot, donde cogerá la pieza dejada del manipulador (con dos posiciones posibles) y la dejará en las cuatro posiciones del palet. Para poder acabar completamente el proceso de toda la estación de trabajo se ha introducido una zona donde las cajas se colocan en un palet mediante el brazo robot, y que una vez que el palet esté lleno de piezas sea expulsado.

Referente a la alimentación y desalojo de palets, se ha montado un conjunto de elementos formados por un actuador (servo válvula), un cilindro de doble efecto y un sensor óptico. En la siguiente figura 4.1 se muestra la distribución de los diferentes elementos en la estación de trabajo:

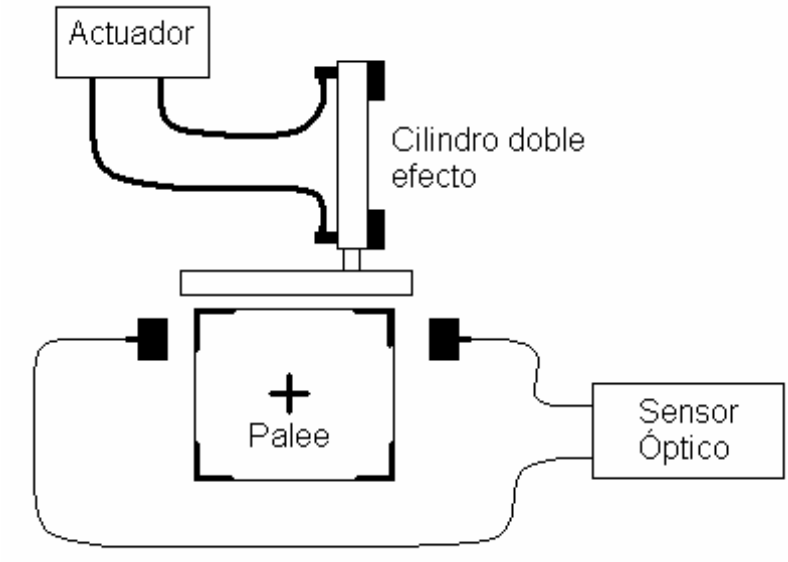


Fig. 4.1 Distribución del sistema de expulsión

4.2- Sistema de expulsión de palet

4.2.1- Actuador

Es una electro válvula de doble efecto que estará conectada eléctricamente al PLC y alimentada mediante aire comprimido del mismo modo que el resto de las estaciones. Ésta suministrará el aire al cilindro neumático en función de las indicaciones del PLC.

4.2.2- Cilindro de doble efecto

Este cilindro permite la expulsión del palet una vez que el PLC indique que las cuatro posiciones del palet están llenas por piezas y que es posible su almacenamiento.

Estos sensores tienen un rango por el cual hay que situarlos en la posición exacta donde está el cilindro. Este rango lo podremos visualizar mediante el LED que contiene el sensor. La figura 4.2 mostrada a continuación muestra la interpretación en función del color que muestre el Led:

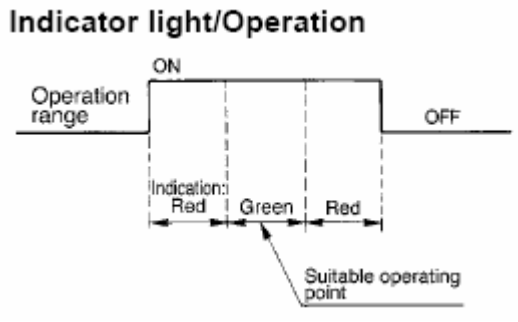
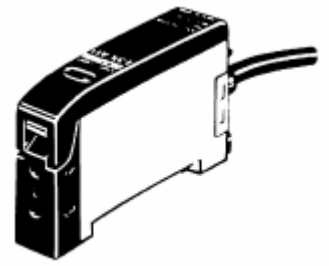


Fig. 4.2

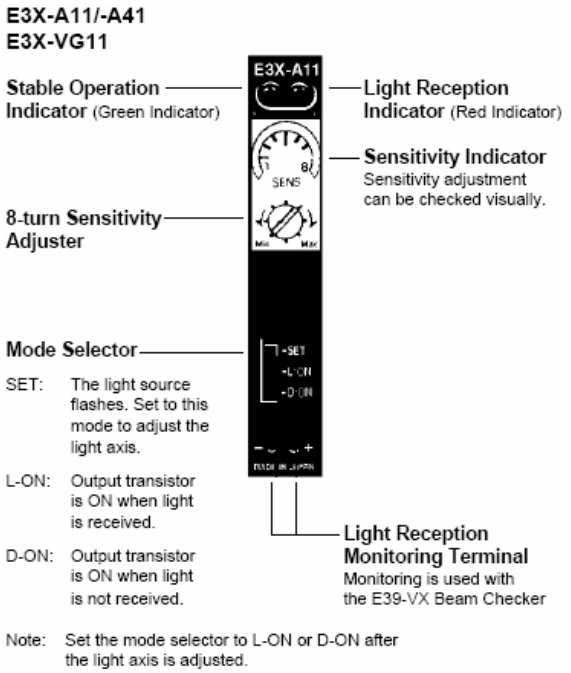
Como se puede observar, cuando el cilindro esté en la posición del sensor, el LED se encenderá. En el caso de que la luz sea roja, indicará que el sensor no está situado del todo correcto aunque nos dará el flanco de señal. En esta situación hay que desplazar el sensor hasta que se active el LED con el color verde y seguidamente fijarlo en esta posición para que de este modo quede debidamente calibrado.

4.2.3- Sensor óptico



Este elemento nos indicará si disponemos de palet o no mediante dos terminales que se sitúan en dos extremos de tal forma que el palet quede comprendido entre estos dos terminales. En el caso de que no estuviera colocado el palet, la luz que transmite el terminal emisor del sensor sería recibida por el terminal receptor de éste. Para cuando el palet evita la transferencia de luz el sensor mandará un cero a la salida que deberá se interpretada por el PLC como que no hay palet y de éste modo el brazo robot no podrá dejar ninguna pieza en el palet ya que no está situado en posición correcta.

- Configuración del sensor óptico:



Para configurar el sensor básicamente hay que seleccionar dos cosas:

1.- La sensibilidad.

Hay que colocar el interruptor *mode selector* en posición *set* y graduar la sensibilidad deseada.

2.- La activación de la señal de salida.

Hay que situar el interruptor *mode selector* en la posición que deseemos que nos dé el flanco de salida; en L-ON para cuando el terminal de recibir del sensor recibe señal de luz, o en D-ON para cuando el terminal de recibir del sensor no recibe señal de luz.

4.3- Sistema de control

4.3.1- Sensor óptico

La forma como el sensor conmuta la señal de salida va en función de si hay o no hay luz, queda bien mostrada en la figura 4.3, la cual muestra que el sensor da un flanco bajo cuando hay una pieza. Esto conlleva un problema con la conexión con el PLC, ya que cuando se integra el elemento, el PLC está predeterminado con una tensión común de 0 voltios y si enviamos con el sensor como flanco el PLC no lo puede

interpretar. Debido a que el PLC CPM-1A no contiene más de un común como referencia para las señales de entrada y que esto no se puede variar, hay que colocar un relee entre el sensor y el PLC para invertir este flanco de 0 voltios a 24V.

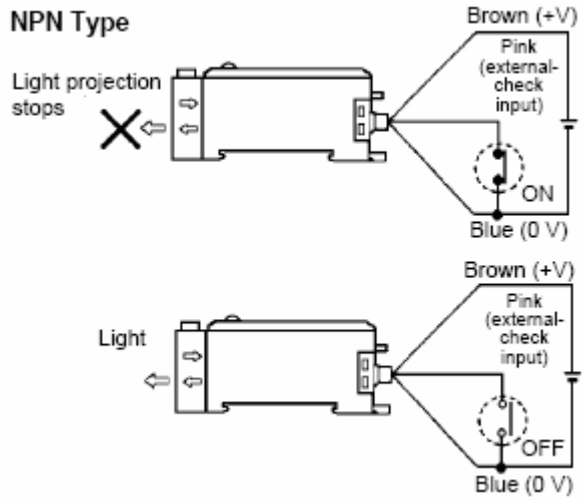


Fig. 4.3

- El esquema que se muestra en la figura 4.4 indica como se hace el conexionado entre el sensor óptico y el PLC.

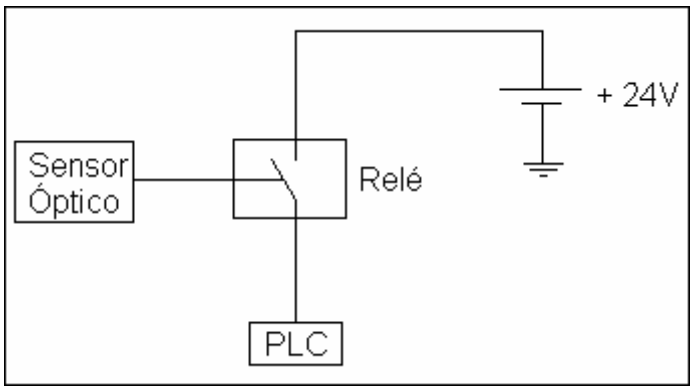
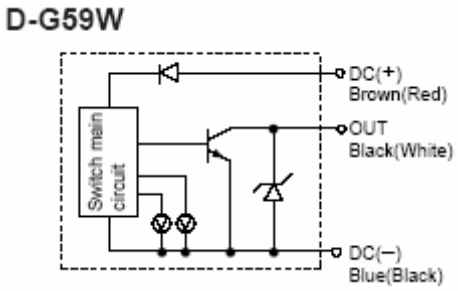


Fig. 4.4

4.3.2- Cilindro de doble efecto

El cilindro posee dos sensores inductivos, cada uno situado en un extremo del cilindro para indicar al PLC en qué posición se encuentra situado el cilindro.

En el momento de hacer el conexionado entre los sensores y PLC existe el mismo problema que en el caso del sensor óptico, en el que el PLC no reconoce el señal que le enviamos por que es un flanco bajo de 0 voltios. En este caso también aplicamos la misma solución en la que utilizamos un relee para que invierta el señal de 0 voltios a +24 voltios.



- El esquema que se muestra en la figura 4.5 indica como se hace el conexionado entre el sensor inductivo y el PLC.

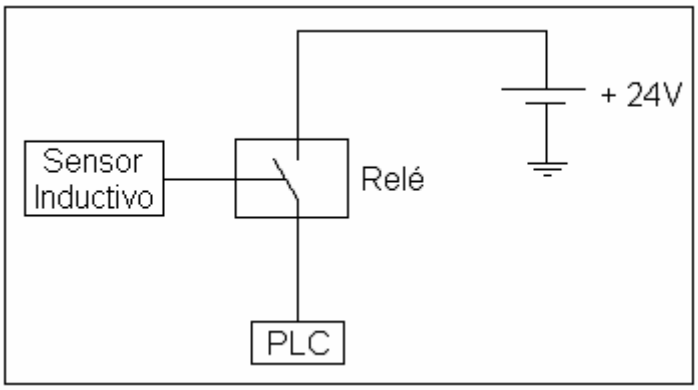


Fig. 4.5

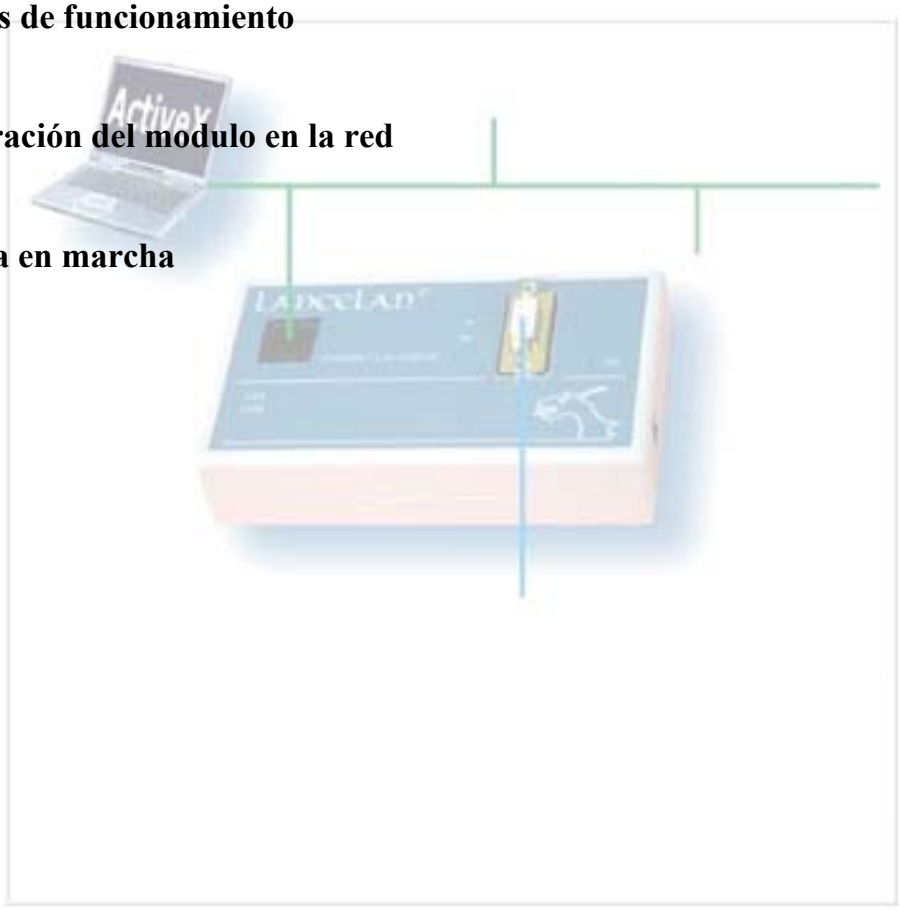
- Introducción

- Funcionalidad del modulo LANCE-1

- Modos de funcionamiento

- Integración del modulo en la red

- Puesta en marcha



5- PROTOCOLO ETHERNET

5.1- Introducción

Por norma general, los protocolos que incorporan los PLC´s y dispositivos serie comunicables son totalmente incompatibles entre sí. Aunque la mayoría de ellos incorporen una conexión serie, la velocidad, estructura de datos, norma física y protocolo, son drásticamente diferentes. Si a todo ello añadimos la enorme sobrecarga de trabajo que supone la implementación de las comunicaciones con estos dispositivos, para los programas que deben procesar los datos obtenidos, nos haremos una idea de la dificultad que conllevan las tareas de comunicación.

El módulo LANCE-1 conecta PLC's y dispositivos serie a una red ethernet, efectuando las pesadas tareas de lectura de los datos del dispositivo asociado poniéndolos a disposición de las aplicaciones cliente, repartidas por la red y con intereses en ellos.

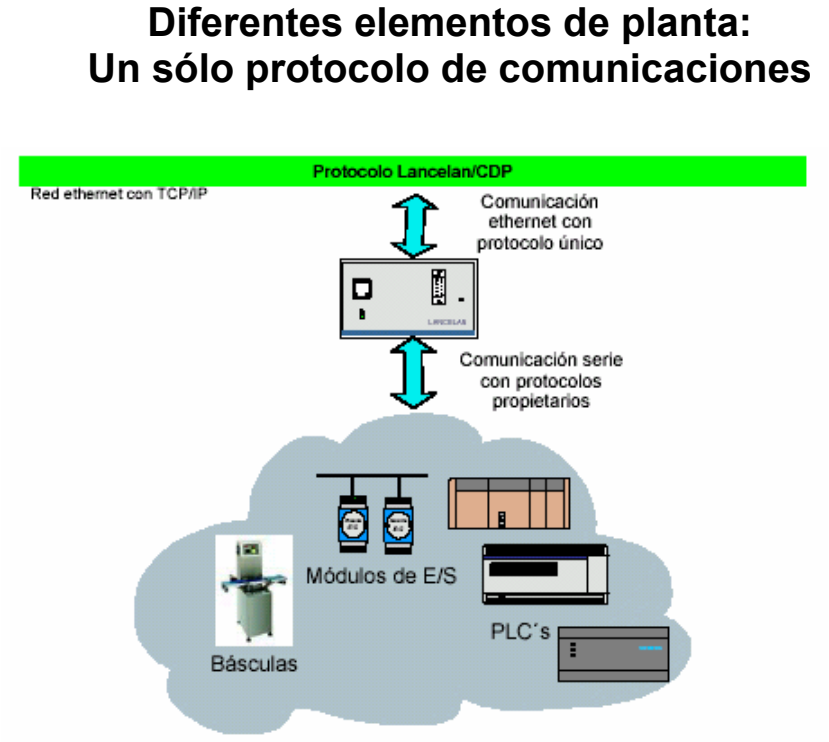
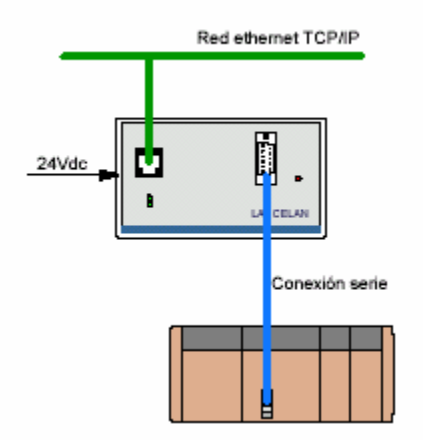


Fig. 5.1

5.2- Funcionalidad del modulo LANCE-1



La principal función del módulo LANCE-1 es la conexión de un PLC o dispositivo serie a la red ethernet, utilizando el protocolo UDP/IP.

El módulo se conecta a la red ethernet mediante un latiguillo de cable UTP enchufado al conector RJ45.

Fig. 5.2

Para la conexión con el PLC usaremos un cable serie enchufado por un extremo al conector sub-D hembra y por el otro, al puerto de programación o comunicaciones del dispositivo.

Desde el momento en que el módulo está conectado a la red y debidamente parametrizado, es posible crear aplicaciones cliente en diferentes PC's conectados a la misma red, las cuales interaccionarán automáticamente con los módulos y, por tanto, con los plc's asociados. Con el módulo LANCE-1 se incluye un control totalmente resuelto a fin de facilitar la programación de dichas aplicaciones cliente en Visual Basic.

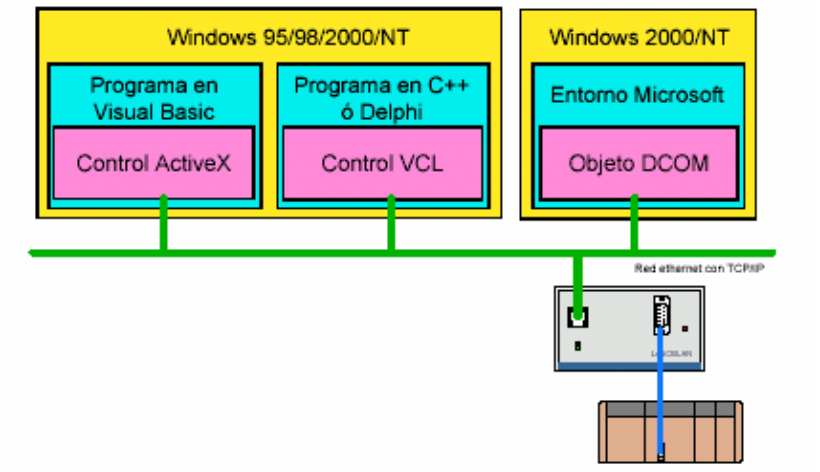


Fig. 5.3

5.3- Modos de funcionamiento

El sistema implementa dos modos de trabajo: Motor local y Transparente. Ambos modos coexisten sin problemas en el módulo ya que utilizan dos puertos IP totalmente independientes. El modo motor local usa el puerto base y el modo transparente el puerto base+2. Como se verá más adelante, el puerto base por defecto es el 3100 aunque puede ser cambiado por configuración.

En nuestro proyecto utilizaremos el modo transparente así que es el que pasamos a describir a continuación:

El modo transparente efectúa el envío y recepción de mensajes entre dispositivos serie, utilizando la red ethernet. Este modo está pensado para permitir el funcionamiento de los programas de programación o supervisión que, por construcción, están pensados para la comunicación punto a punto a través del puerto serie.

Partiendo del punto que el software de programación o programa específico se comunican correctamente con el PLC por medio del puerto serie, podemos intercalar un módulo al lado del PC, conectado al puerto COM y otro módulo al lado del PLC, conectado a su puerto de comunicaciones.

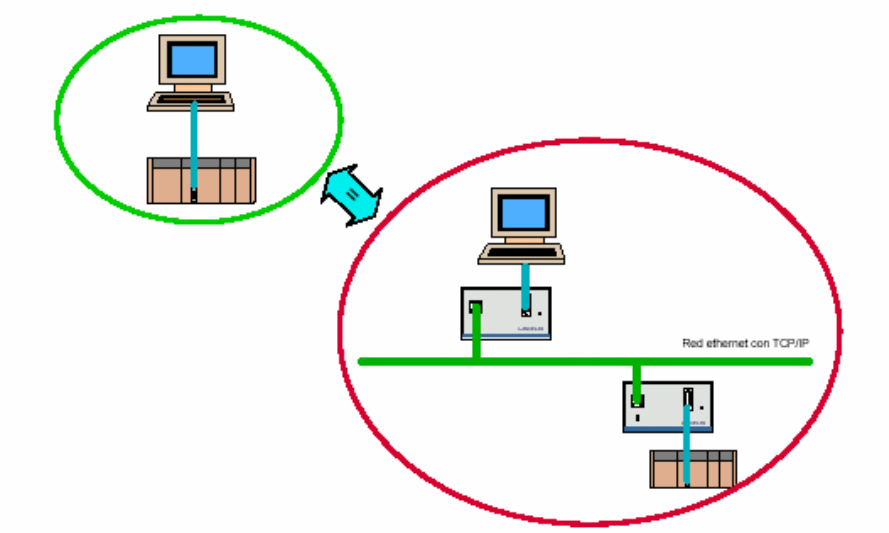


Fig. 5.4

5.4- Integración del modulo en la red

Un punto muy importante debido a que nos causo muchos problemas a la hora de la conexión es el asegurar el perfecto funcionamiento del módulo conectado a una red, es necesario tener en cuenta las siguientes recomendaciones:

· La dirección IP debe ser única y, por ello, debe ser proporcionada por el administrador de la red. La asignación de una dirección IP duplicada puede provocar el mal funcionamiento de la red y la subsiguiente pérdida de datos.

· La máscara de sub-red (netmask) debe ser la adecuada a la dirección IP asignada al módulo. También nos será proporcionada por el administrador de la red.

· Si el módulo debe ser visible desde otra sub-red deberemos incluir la dirección del router (gateway) de nuestra red. Su dirección nos la proporcionará el administrador de la red y la incluiremos en el fichero de configuración del módulo.

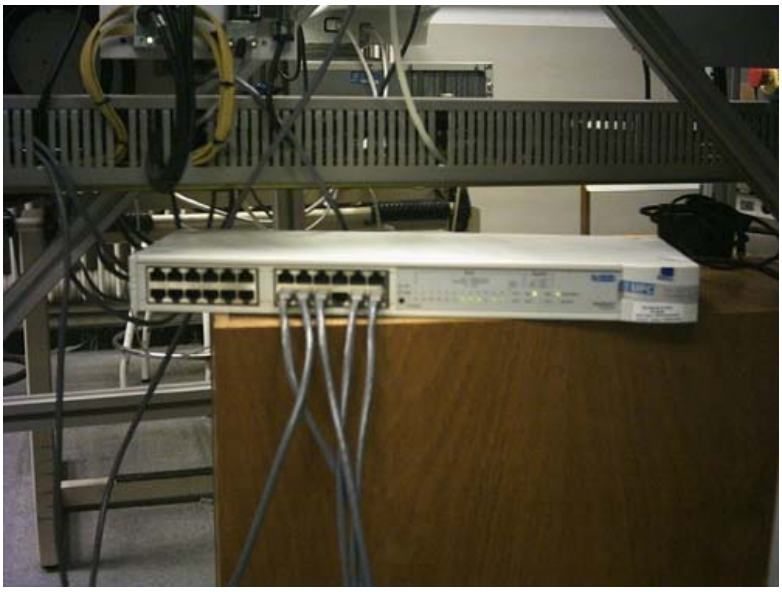


Fig. 5.5 Foto del router

· En el módulo se han definido tres puertos UDP consecutivos a partir del puerto base al cual, si no se ha modificado con la directiva base port, el puerto por defecto es el número 3100. Aunque a los tres se les han asignado los mismos servicios es decir, pueden ser utilizados indistintamente, la utilidad primaria de cada uno de ellos es la siguiente:

. Puerto 3100: Protocolo CDP.
. Puerto 3101: Servicios auxiliares de configuración y monitorización.
. Puerto 3102: Modo transparente.

5.5- Puesta en marcha

Para que el programa no nos dé ningún tipo de problema, hace falta realizar una configuración correcta de todos los apartados que se irán comentando a continuación. El proceso se realizara tal y como se va comentando, siempre desde el primer punto.

1.- Instalar el editor de configuración (LanceLan editor) en un PC: Una vez tengamos el programa instalado, el editor de configuración nos permitirá configurar correctamente los módulos. Para ello lo instalaremos en un PC con sistema operativo Windows y lo ejecutaremos. En este punto será necesario conocer la dirección IP de este PC que es (200.1.1.94), así como la dirección que se le debe asignar al módulo que ya nos viene indicada en el mismo (200.1.1.84).

2.- Conectar el módulo al PC: Inicialmente se debe conectar el módulo al PC para proceder a su configuración, pudiendo ser esta de dos maneras diferentes. Una de ellas permite la conexión serie mediante el COM2, mientras que la otra se realiza con cable ethernet cruzado para una conexión de red punto a punto. Nuestra conexión será esta última, vía cable ethernet, por lo tanto usaremos un latiguillo cruzado con conectores RJ45. Se tendrá en cuenta que para conectar el

módulo con el PC por medio de la red, la parte de la dirección correspondiente a la sub-red debe coincidir en los dos aparatos, así como la correspondiente máscara de sub-red. Por defecto, el módulo viene programado con la dirección 192.168.3.40 pero nosotros la tendremos que modificar a (200.1.1.84) y la máscara de sub-red 255.255.255.0 que se mantendrá tal y como esta dado que no varía su valor.

3.- Conectar la alimentación del módulo: Éste puede ser alimentado a 24Vcc si disponemos de dicha tensión o, podrá usarse el alimentador a 220Vac suministrado conjuntamente (se recomienda). Si está alimentado correctamente, deberá iluminarse el piloto rojo situado en su panel frontal tal y como vemos en la imagen.

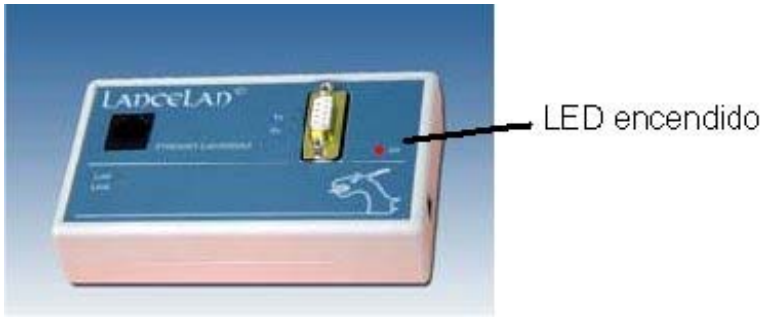
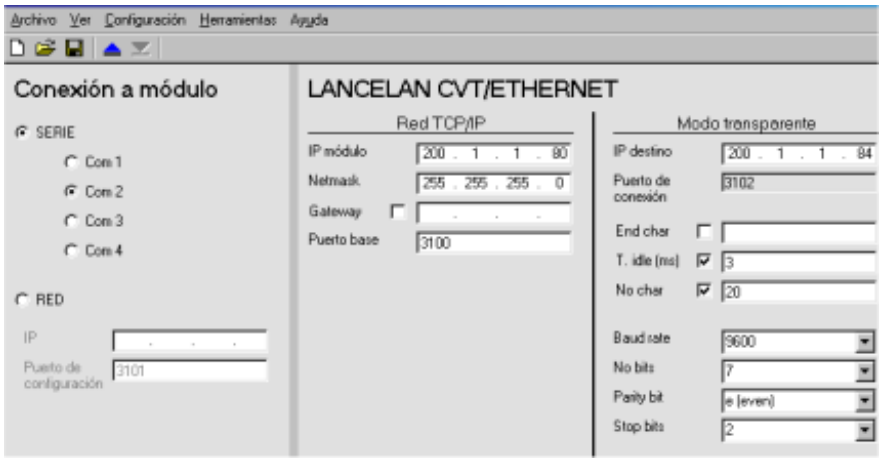


Fig. 5.6

4.- Comprobar que el PC "ve" el módulo: Elegimos la forma de conexión entre ambos, en nuestro caso cogeremos la opción en serie que es la que hemos utilizado a la hora de realizar la configuración del modulo con el puerto COM2.

Arrancaremos el programa editor en el PC y pulsaremos el botón **Cargar configuración del modulo.** En la ventana del programa debe aparecer la lista de parámetros que el módulo soporta.

5.- Editar el fichero de configuración: Con el editor modificaremos los parámetros requeridos. Normalmente los parámetros que cambian son la dirección de IP y el tipo de PLC. La siguiente imagen nos muestra los parámetros utilizados en nuestra configuración. Para guardar los cambios efectuados, pulsamos en el botón **Escribir en el modulo**. Para obtener mas ayuda sobre el editor, pulsaremos el botón F1 o en la barra Menú la opción Ayuda.



En el apartado anterior podemos ver que la configuración del apartado **Red TCP/IP** ya se ha explicado cada apartado, aquí introducimos valores que ya se nos adecuan a nuestras necesidades de programa y comunicación.

Al trabajar en modo transparente también citado con anterioridad, los parámetros son los indicados en la figura que se resumen en:

IP destino. Colocaremos la dirección IP de nuestro modulo transparente (200.1.1.84).

Puerto de conexión. Nos indica en que modo trabaja el modulo, colocaremos el 3102 de esta manera le decimos que trabaje en modo transparente.

End char. Especificación del carácter final de un mensaje entrante. Si utilizamos un carácter imprimible, lo escribiremos directamente. En el caso de utilizar un carácter no imprimible es posible usar su equivalente en decimal delimitado por paréntesis. En nuestro caso no especificaremos nada.

T. idle. Especifica el tiempo mínimo, en milisegundos, para considerar que ha finalizado el mensaje recibido. Con un tiempo de 3 ms ya nos basta.

No Char. Especifica el número de caracteres exactos que componen un mensaje entrante. Daremos por recibido un mensaje cuando se hayan recibido la cantidad especificada por esta directiva.

Baud rate. Velocidad (Baud-Rate) del modo transparente. Los valores válidos son 1200, 2400, 4800, 9600, 19200, 38400 baudios.

No bits. Número de bits de datos del modo transparente. Los valores válidos son: 7 y 8 bits.

Parity bit. Paridad de los datos del modo transparente. Los valores válidos son: **e** (Even), **o** (Odd) y **n** (None).

Stops bits. Número de bits stop del modo transparente. Los valores válidos son 1 y 2.

Una vez introducidos todos los datos en la ventana de configuración del lancelan editor quitaremos la alimentación del modulo. Al realizar esto, haremos que el modulo se actualice con la configuración que hemos escogido y que cada vez que se ponga en funcionamiento, lo haga de la manera especificada por nosotros.

6.- Conectar el módulo a la red ethernet definitiva: Para conectar el módulo a la red se utilizará un cable normal de red del tipo UTP con conectores RJ45 en ambos extremos. Uno de ellos se conecta al módulo y el otro a la red. Si el módulo está alimentado correctamente, debería iluminarse el led rotulado LAN.

Puntualizamos que para un funcionamiento correcto debemos de disponer de un nudo de red de entrada para que transcurra un flujo de datos a través del router y así poder crear una red.

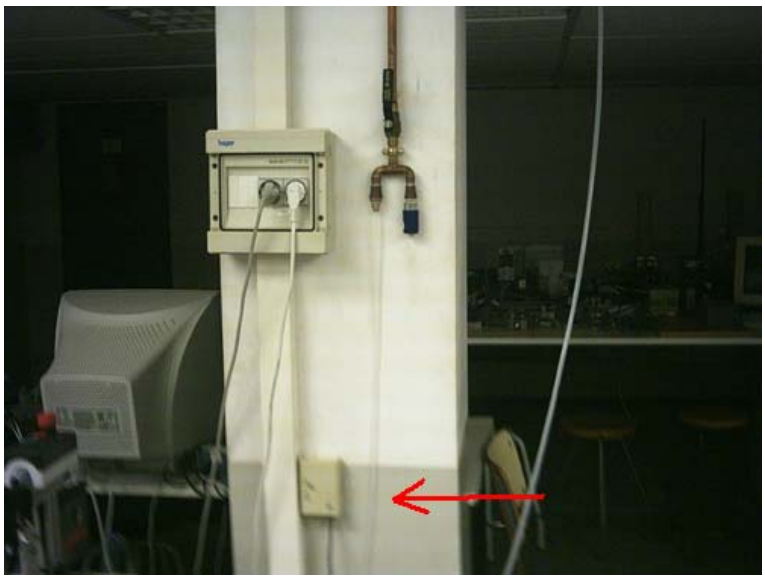


Fig. 5.7 Foto del nudo de red de entrada

En este punto, el módulo ya está adecuadamente conexionado y configurado para su inmediata utilización.

- Integración del brazo robot a la maquina

- Modo de comunicación Brazo Robot-PC

- Introducción al Visual Basic

- Programación de Visual Basic para la comunicación con el brazo robot mediante el puerto COM

6- INTEGRACIÓN DEL BRAZO ROBOT EN LA ESTACIÓN DE TRABAJO

6.1- Integración del brazo robot a la maquina.

La principal limitación de este autómata, es que no permite realizar una conexión directa entre PLC (CPM1A) y la controladora del brazo robot (el modelo CQM de OMRON permite enviar tramas que la controladora puede interpretar y de este modo permitiría la conexión directa PLC-Controladora). Esta limitación es solventada por la introducción de un sistema de supervisión que se llevará a cabo mediante un PC. Dicho PC hará la función de intermediario entre PLC y el brazo robot.

Para poder llegar a crear un movimiento concreto del brazo, tendremos que disponer de un PC con su respectivo programa que nos deje realizar cualquier movimiento deseado por nosotros. Al trabajar inicialmente con Visual Basic en la programación de manipulador, se aprovechó este programa para hacer el sistema de supervisión y no tener que utilizar dos programas distintos. De esta manera desarrollamos un mismo programa que nos permite trabajar con el manipulador y el brazo robot a la vez.

6.2- Introducción al Visual Basic

En la creación del sistema del supervisor, se ha realizado mediante el Visual Basic que es un lenguaje de programación en C con multitud de aplicaciones, como la comunicación mediante el puerto serie o Ethernet.

Al abrir un nuevo documento, aparece una pantalla denominada *form1*, en esta pantalla se podrá ir añadiendo una serie de elementos los cuales se pueden encontrar botones, imágenes, barras, etc.

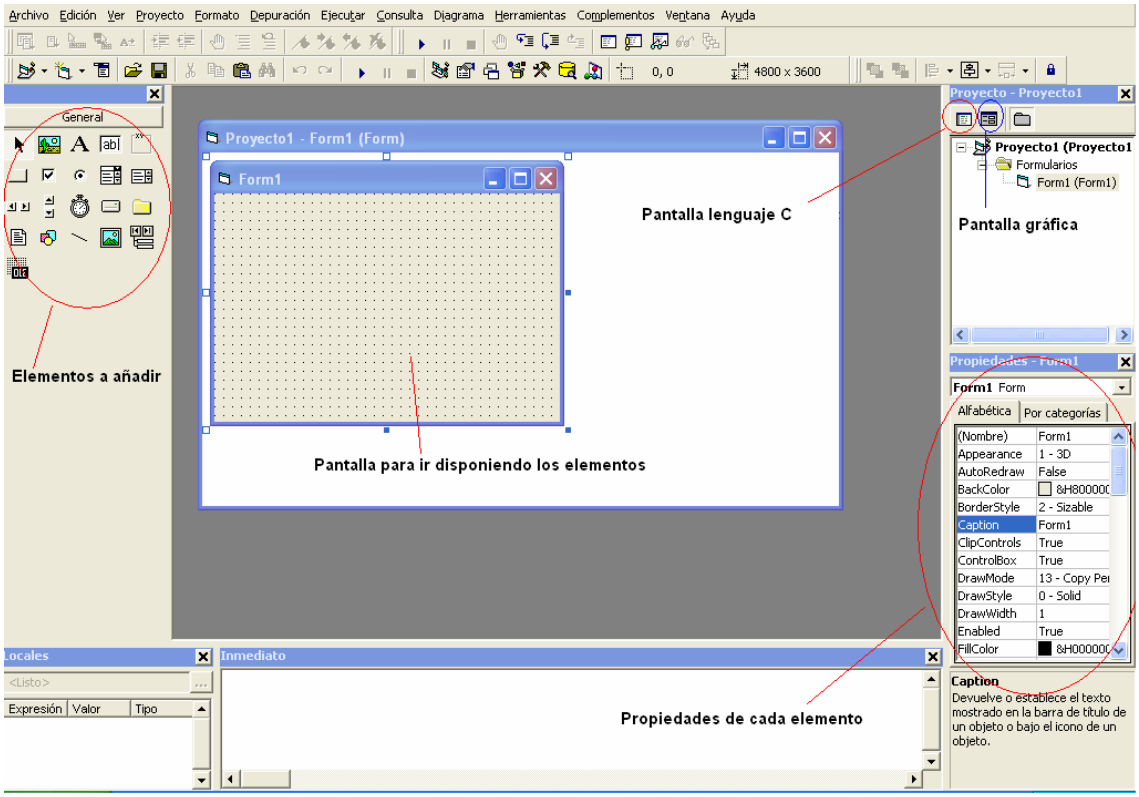


Fig. 6.1 Disposición de los elementos en pantalla

A partir de aquí, se modifica nuestra pantalla según las necesidades de programa. Los elementos se irán añadiendo en la ventana denominada *form1* pero paralelamente a esto, hay que tener en cuenta que cada elemento tiene una función determinada, esta serie de funciones son programadas mediante el lenguaje C.

Las funciones de cada elemento son muy amplias y variables pero las más importantes y específicas de este proyecto son las siguientes:

- **Private Sub Command_Click() >** Esta rutina nos indica que al hacer un clic sobre el botón carga la función que se encuentre en su interior.

- **Private Sub Command_MouseDown() >** Esta rutina indica que en el momento justo de pulsar el botón del mouse carga una función concreta.

- **Private Sub Command_MouseUp() >** Esta rutina indica que en el momento justo de dejar libre el botón del mouse carga una función concreta.

- **Private Sub Form_Load() >** Rutina que al cargar la pantalla inicial iniciará la función que se encuentre en su interior.

- **Private Sub Timer_Timer() >** Dependiendo de un tiempo configurable desde la barra de propiedades, carga la función alojada dentro de la rutina.

- **Private Sub Text_Change() >** Cuando el valor del display se vea modificado, creará la acción determinada.

- **Private Sub ShockwaveFlash() >** Rutina relacionada para cualquier archivo flash, para la carga de animaciones en función del programa interno de la rutina.

Private Sub MSComm_OnComm() > Esta rutina nos iniciará la comunicación a través del puerto COM para poder enviar los datos adecuados en la comunicación PC-Robot. En propiedades de los elementos, se tendrá que configurar previamente una serie de valores para que funcione adecuadamente como son, *comm port* y el apartado de *settings*. Este último apartado viene predeterminado por los datos facilitados por el fabricante de la controladora.

- **Private Sub UDPDriver_Response() >** Esta rutina permite hacer todas las lecturas de las variables y posiciones de memoria que se encuentran en el PLC. A partir de la lectura, se puede trabajar con los datos obtenidos en función del programa.

  Este punto tiene que configurarse previamente para su funcionamiento, que consta de una *dirección IP del modulo*, *el puerto de comunicación*, el *tipo de PLC* y el *nº de estación*.

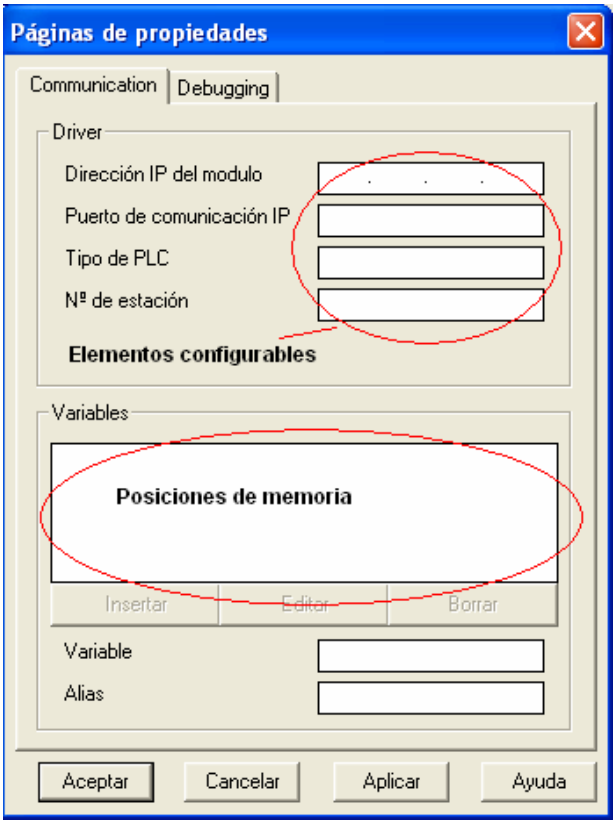


Fig. 6.2 UDP

Con todos estos elementos y su función correspondiente, se desarrolla la presentación de programa pero la funcionalidad la da el lenguaje en C. Por este motivo es necesario tener una base óptima de programación en C. A partir de esta programación se trabajará para hacer lecturas o enviar datos al PLC como es el caso de la carga de las diferentes animaciones en flash o simplemente poder comunicarnos a través de los puertos serie (comando mscomm) o la red de ethernet (UDP).

6.3- Programación de Visual Basic para la comunicación con el brazo robot mediante el puerto COM

Antes de empezar a trabajar con Visual Basic en la programación del protocolo para la comunicación entre brazo robot y el PC mediante el puerto COM, tendremos que cargar previamente unas librerías denominadas (archivos .OCX). De esta manera haremos que el Visual Basic pueda entender la información que le llega a través del puerto serie.

Para cargar estas librerías abrimos el Visual Basic, y en la barra superior de herramientas, marcamos *Proyecto* > *Componentes* y entraremos en el menú de las librerías .OCX. En la pestaña donde pone componentes, buscamos el archivo que pone *Microsoft comm control 6.0* tal y como mostramos en la figura 7.1 y le damos a aplicar.

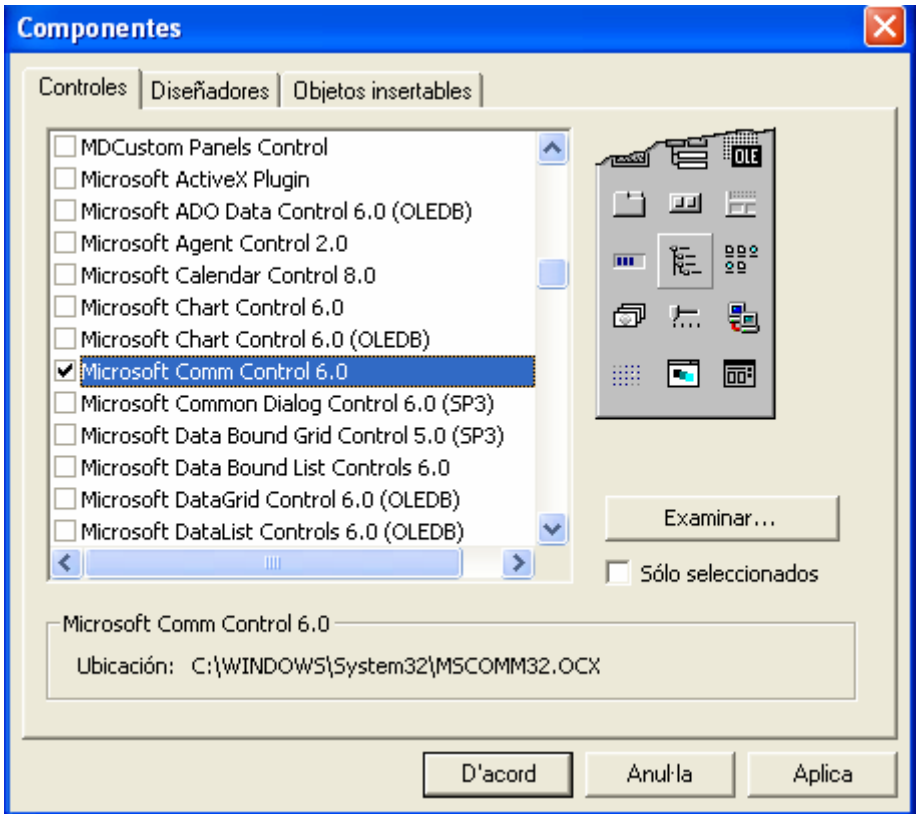


Fig. 7.1

Al activar esta librería no implica que la comunicación ya este realizada entre brazo y PC, esta librería lo único que hace es entender los datos que entran por el puerto serie. Ahora tocaría realizar el programa en C (protocolo) para poder trabajar con esos datos.

El protocolo de comunicación es el siguiente:

```
Option Explicit
'Declaramos variables'
        Public Dato1 As String
        Public Dato2 As String
        Public Dato3 As String
```

```
Private Sub Command1_Click()
'comprueva que el puerto este cerrado para poder abrirlo'
        If MSComm1.PortOpen = False Then
'determina el puerto que hemos seleccionado'
                MSComm1.CommPort = 2
'determina: 9600-Velocidad en Baudios, N-No utiliza ninguna paridad, 8-
Cantidad de bits de envío y recepción por paquete, 1-Determina los bits de
parada'
                MSComm1.Settings = "9600,N,8,1"
'lee todo el buffer de entrada para que quede vacío'
                MSComm1.InputLen = 0
'Abre el puerto seleccionado'
                MSComm1.PortOpen = True
                Me.Caption = "Conectado por el puerto " &
                MSComm1.CommPort
        End If
End Sub
```

```
Private Sub Command2_Click()
        If MSComm1.PortOpen Then
'cierra el puerto'
                MSComm1.PortOpen = False
                Me.Caption = "Desconectado"
        End If
End Sub
```

```
Private Sub Command3_Click()
'envía el texto escrito'

   Dato1 = Chr$(Text1.Text)
   Dato2 = Chr$(Text2.Text)
   Dato3 = Chr$(Text3.Text)

   MSComm1.Output = Dato1
   MSComm1.Output = Dato2
   MSComm1.Output = Dato3

'coloca el texto que enviamos en la pantalla'
   Text4.Text = Text1.Text
   Text1.SetFocus
End Sub
```

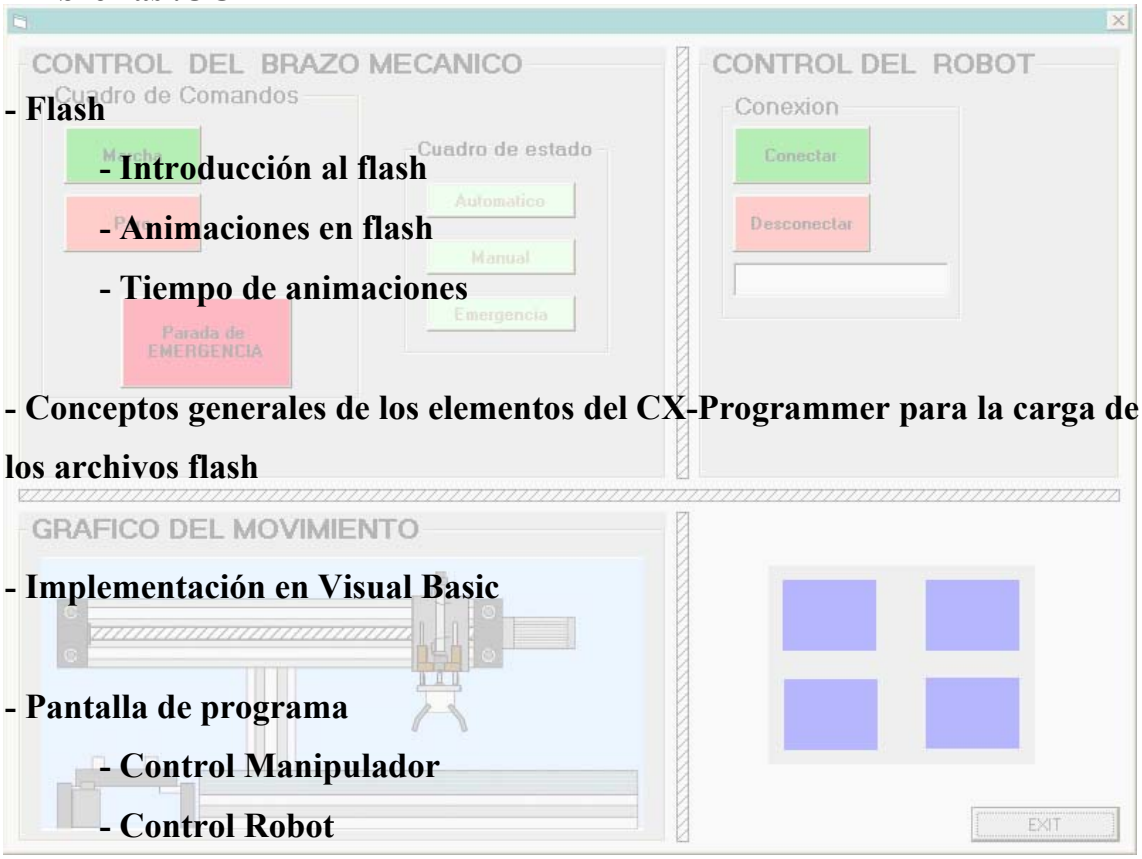
Con este protocolo redactado en el Visual Basic, ya nos permitirá hacer la comunicación, de tal manera que podremos comunicarnos para enviar o recibir valores a trabes del puerto serie.

6.4- Modificaciones del programa de control de la estación

Por medio del sistema supervisor se puede controlar la conexión o desconexión entre el brazo robot y el PC. Si la conexión se realiza correctamente o ha habido algún tipo de problema, mediante el display que aparece en pantalla nos lo indicará. Una vez establecida la conexión entre ambos, el sistema supervisor envía tres tramas para indicar a la controladora del brazo robot a qué servo tiene que mandar las diferentes coordenadas. En la primera trama se envía el bit de sincronismo seguida de la segunda que indica el número de servo al que va destinado la información y en la tercera trama se envía el valor al que tiene que ir el servo seleccionado.

- Introducción

- Librerías .OCX



- Flash

- Introducción al flash

- Animaciones en flash

- Tiempo de animaciones

- Conceptos generales de los elementos del CX-Programmer para la carga de los archivos flash

- Implementación en Visual Basic

- Pantalla de programa

- Control Manipulador

- Control Robot

- Gráfico de movimiento

- Gráfico cajas

7- SISTEMA DE SUPERVISIÓN

7.1- Introducción

Paralelamente al proyecto realizado del control y comunicación de todos los elementos presentes en la estación de trabajo, se ha implementado una nueva aplicación en el PC. Esta nueva aplicación consiste en un sistema de supervisión de toda la actividad de la estación, dando la opción de visualizar todo el proceso y sus respectivas secuencias desde cualquier punto del edificio. Este sistema dota al proyecto de una amplia mayor aplicación debido a que la comunicación PLC – PC la realizamos a través de ethernet y con este sistema de supervisión se puede controlar y visualizar estados desde puntos anexos al edificio como una oficina o desde casa.

Para desarrollar este sistema de supervisión, se ha utilizado una programación en flash donde una vez diseñadas todas las secuencias de comandos se cargaran en el mismo programa utilizado hasta ahora (Visual Basic).

7.2- Librerías .OCX

Al trabajar con Visual Basic podemos observar que la implementación de ficheros .swf (archivos realizados en flash) no es posible. Para poder introducir estos archivos se tendrá previamente que cargar unos archivos denominados **Swflash.ocx** de unas determinadas librerías. Este proceso es el mismo que el comentado en el apartado de la comunicación mediante el puerto COM pero ejecutando otro tipo de archivos .OCX, así que seguiremos los mismos pasos, marcamos *Proyecto* > *Componentes* y entraremos en el menú de las librerías .OCX. En la pestaña donde pone componentes, buscamos la opción ShockwaveFlash para luego agregarla a nuestra caja de herramientas tal y como mostramos en la figura 8.1 y le damos a aplicar.

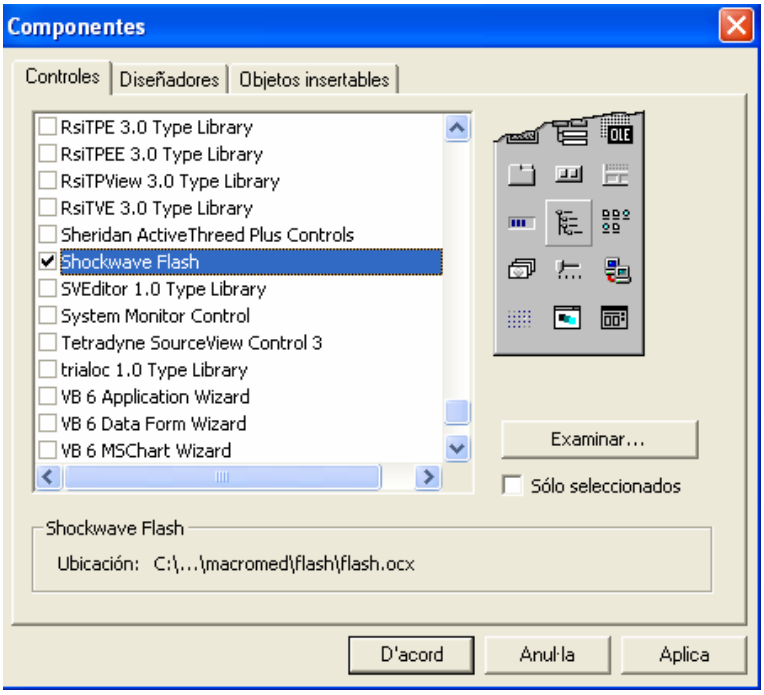


Fig. 7.1

Después del paso anterior, lo más normal es que aparezca el logo de Flash como un control más, claro que cuando se tiene Flash MX instalado simplemente aparece en lugar del logo una hoja en blanco, como la de crear nuevo archivo. Ahora nuestro programa esta preparado para entender los diferentes ficheros que se le mandarán desde el flash aunque habrá que realizar un programa con Visual Basic con el que se indicará cuando y que ficheros cargar. Se debe tener en cuenta que el tamaño con que se dibuje la cuadricula en Visual Basic, será el tamaño con que lo veamos en la aplicación.

7.3- Flash

7.3.1- Introducción al flash

Una vez se ha configurado el Visual Basic y se ha visto que es posible la implementación de archivos .swf debido a que no todos los programas de programación te dejan trabajar con esta aplicación, se procederá a la realización de todos los fotogramas adecuados.

Antes de empezar a desarrollar los diferentes fotogramas, se ha de tener en cuenta los dos grandes estados con los que trabajamos en la estación de trabajo como son el modo automático y el manual. Se hace especial mención a estos estados porque cada uno trabajará con una serie de fotogramas totalmente independientes el uno del otro, debido a que el modo automático se trabajará con unos fotogramas cíclicos (la secuencia se repite al ser automático) mientras que el manual (paso a paso) se ha de hacer por etapas validándolas a una orden determinada por el usuario.

Empezamos cargando el programa Macromedia Flash y vemos que se dispone de una hoja en blanco no configurada. El primer paso que haremos será configurar el tamaño que vendrá determinado por el tamaño que le hemos dado a la cuadricula del Visual Basic. Este tamaño será de 430mm x 190mm, para cambiar este valor, tendremos que abrir el submenú de propiedades situado en la parte inferior de la pantalla. Una vez abierto veremos una pequeña ventana la cual esta conformada por las opciones de *dimensiones, coincidir, color de fondo, velocidad fotograma y unidades de regla.* De momento solo nos centraremos en la opción de dimensiones, donde pondremos el tamaño ya citado, al modificar este valor y darle a aceptar, veremos como el espacio de que disponíamos para crear la animación se verá levemente reducido. En la siguiente imagen se muestra bien como configurar y preparar la hoja para el nuevo tamaño.

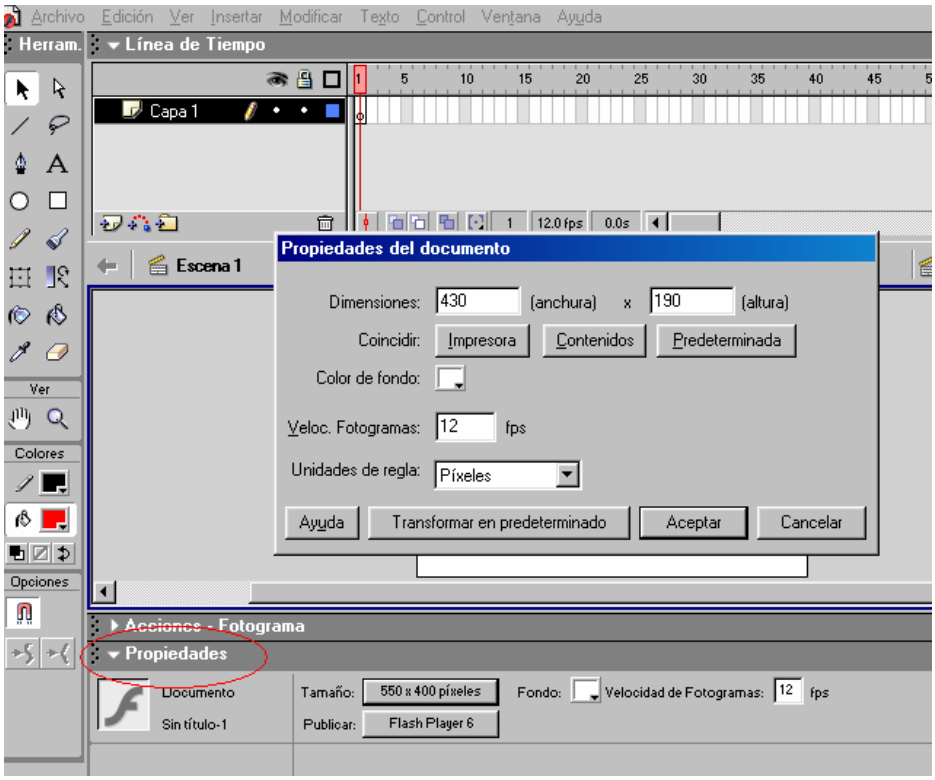


Fig. 7.2 Foto de la modificación de tamaño

Otro punto muy importante a la hora de crear la animación, es que al darle movimiento a los diferentes gráficos, tenemos que hacer unos elementos independientes de los otros, debido a que las pinzas por ejemplo no tendrán el mismo movimiento que el pistón que las hace subir o bajar su posición. Para poderlos hacer independientes entre si, se dibujarán en capas diferentes, es decir, pinzas en una capa, pistón en otra, caja en otra, etc.

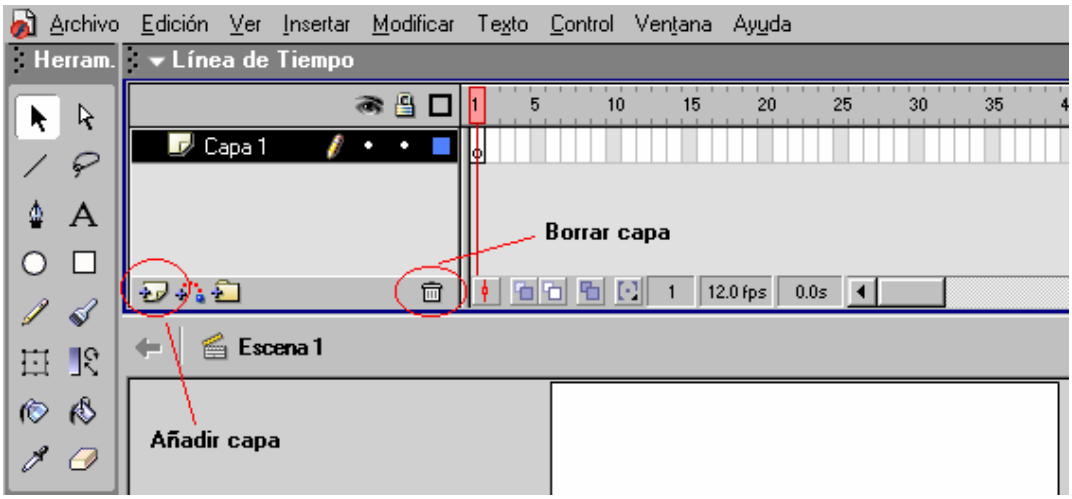


Fig. 7.3 Edición de capas

Basta con ir añadiendo capas hasta conseguir el número adecuado para cada conjunto de animaciones, de tal forma que el resultado final de capas viene a ser el que se muestra en la figura 8.4.

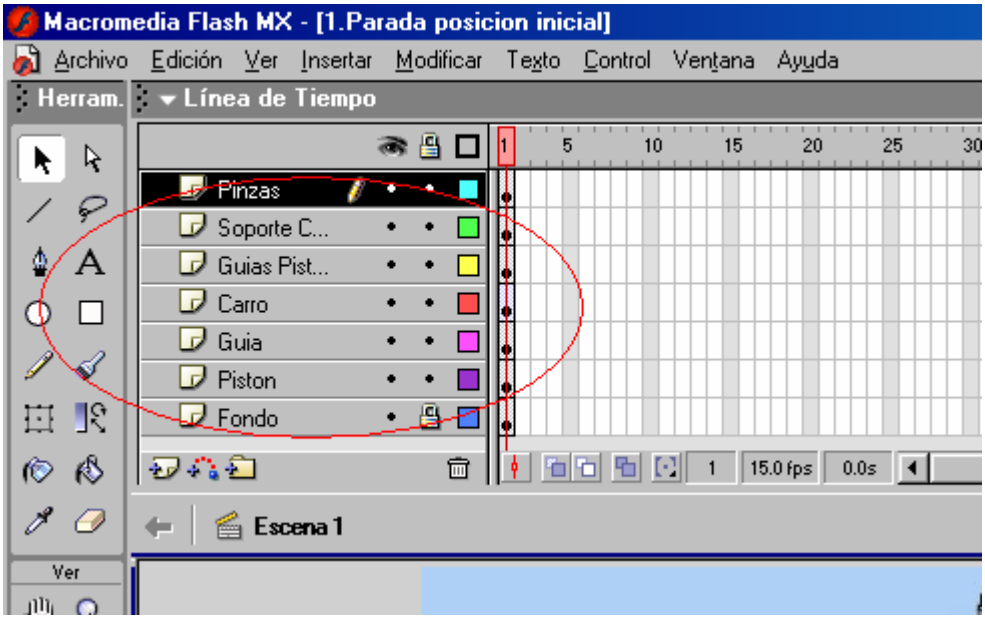


Fig. 7.4 Foto de las diferentes capas

En este punto el programa ya esta configurado de manera adecuada para empezar a crear las animaciones. Se crearán todos los objetos y el movimiento se realizará utilizando un sistema de interpolación. Este sistema de interpolación, se localiza en la barra situada en la parte superior de la imagen y dispone de unos bloques temporales.

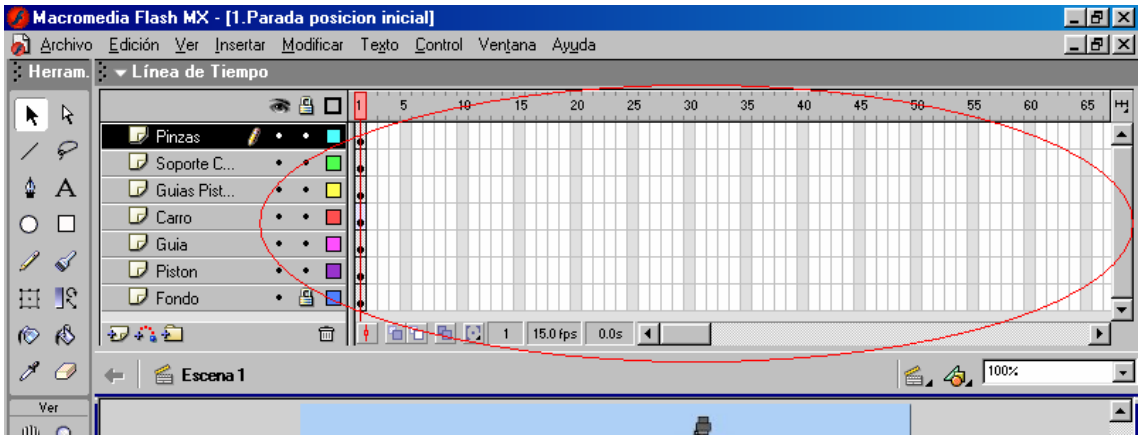


Fig. 7.5 Foto de los bloques temporales

IMPORTANTE. Este sistema de movimiento se realizará independiente entre capas debido a que cada objeto tendrá su movimiento específico por tanto, cada capa dispondrá de su movimiento concreto.

Marcamos una capa para trabajar con un elemento concreto y le damos una posición inicial en los bloques de tiempo que será el denominado fotograma inicial, dejamos una separación de bloques en blanco hasta poner el fotograma final que nos indicará el final del movimiento. Este espacio en blanco de bloques que dejamos, determinará el tiempo que tardará en hacer el movimiento, el número de bloques que dejamos entre el fotograma inicial y el final, será el responsable de que el movimiento sea rápido o lento, a mayor número de bloques, más lento

Fig. 7.6 Foto de la duración de tiempos

Este proceso habrá que hacerlo para cada capa porque cada objeto tendrá su movimiento y su tiempo determinado. Obsérvese que el movimiento no tiene porque coincidir con el movimiento real del objeto, será un movimiento aproximado y fijado por la base de tiempo entre bloques.

7.3.2- Animaciones en flash

Como se ha comentado anteriormente, las animaciones están divididas en dos subgrupos, unas son las que pertenecen al modo automático, y las otras al modo manual.

Además de esta condición, se ha de tener el cuenta el tiempo de duración de cada interpolación, éste es un factor muy importante debido que un tiempo incorrecto hará variar totalmente el tiempo de visualización del programa conllevando un desfase temporal manipulador – simulación. El tiempo de interpolación se controla en flash mediante los fotogramas por segundo.

El modo automático estará compuesto por:

- **Parada.** Esta es una imagen, no tiene movimiento debido a que es el momento de reposo en que se encuentra el manipulador al arrancar o al parar.

- **Inicio.** Animación que realiza el movimiento del carro desde la posición inicial hasta la posición de coger pieza.

- **RetrocesoA.** Animación que realizará el conjunto de movimientos de bajar pistón, cerrar pinza, subir pistón, ir a la posición A para dejar la pieza y acto seguido volver a la posición de coger pieza. Este retroceso ira en función de los inductores de posición que nos indicaran la posición A o B. En caso de que los inductores continúen en la posición A, el proceso se repetiría debido a que está configurado como un bucle.

- **RetrocesoB.** Animación que realizará el conjunto de movimientos de bajar pistón, cerrar pinza, subir pistón, ir a la posición B para dejar la pieza y acto seguido volver a la posición de coger pieza. Este retroceso ira en función de los inductores de posición que nos indicaran la posición A o B. En caso de que los inductores continúen en la posición B, el proceso se repetiría debido a que está configurado como un bucle.
- **RetrocesoCorto.** Animación que realiza el movimiento que se produce de la posición B a la posición de reposo inicial en caso de darle a parada.

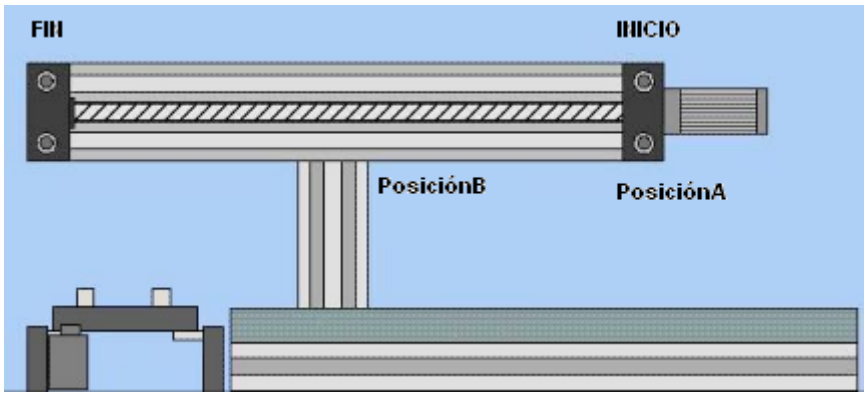


Fig. 7.7 Foto esquema de posiciones

Vemos que en el modo automático, empezará ejecutándose la animación de parada para inmediatamente al darle a marcha se ejecuta el inicio del movimiento del carro hasta la posición final del carro.

A partir de aquí ira en función de los inductores de posición que nos indicaran si tiene que ejecutar el RetrocesoA o RetrocesoB. Ejecuta uno de los dos, hace la acción de coger la pieza, ir a la posición adecuada (A o B) para dejarla y retroceder de nuevo al punto final. Una vez aquí vuelve a hacer lectura de los inductores para saber si ha de repetir secuencia o cambiar a la otra posición. En caso de darle al botón de parada, si el carro se desplaza a la posición A no habrá problema debido a que coincide con la posición inicial y se pararía a final de ciclo. El problema es en la posición B, que al dejar la pieza debe cargar otra animación que lo hará desplazarse hasta la posición inicial, es decir, esta animación solo se ejecutará al pulsar el botón de parada.

El modo manual estará compuesto por:

- **Parada.** Esta es una imagen, no tiene movimiento debido a que es el momento de reposo en que se encuentra el manipulador al arrancar o dar a parada.

- **Inicio.** Animación que realiza el movimiento del carro desde la posición inicial hasta la posición de coger pieza.

- **Bajar Pistón.** Animación que realiza el movimiento de bajar el pistón en la posición final.

- **Cierra Pinza.** Cerrará pinza para coger pieza en la posición final del carro.

- **Subir Pistón.** Animación de subir pistón en la posición final del carro, con pieza cogida.

A partir de aquí hará el proceso A o B en función de los inductores de posición:

Posición A:

- **RetrocesoA.** Solo realizará el desplazamiento desde la posición final del carro hasta la posición A.

- **Bajar PistónA.** Animación que realiza el movimiento de bajar el pistón en la posición A.

- **Abrir PinzaA.** Abrirá pinza para dejar pieza en la posición A.

- **Subir PistónA.** Animación de subir pistón en la posición A del carro, sin pieza cogida.

- **RetornoA.** Desplazamiento del carro desde la posición A a la posición final.

Posición B:

- **RetrocesoB.** Solo realizará el desplazamiento desde la posición final del carro hasta la posición B.

- **Bajar PistónB.** Animación que realiza el movimiento de bajar el pistón en la posición B.

- **Abrir PinzaB.** Abrirá pinza para dejar pieza en la posición B.

- **Subir PistónB.** Animación de subir pistón en la posición B del carro, sin pieza cogida.

- **RetornoB.** Desplazamiento del carro desde la posición B a la posición final.

En el modo manual, se puede ver que hay muchas mas etapas debido a que no se puede hacer un ciclo cerrado como en el caso del automático. Las animaciones se harán paso a paso de manera que al finalizar un proceso quede en situación de reposo hasta nueva orden. La secuencia del proceso es igual que la del automático, las posiciones A o B vienen determinadas por los inductores de posición, pero en este caso no nos dirán de cargar una animación cíclica sino un conjunto de animaciones (bajar pistónA, Abrir pinzaA, etc.) al trabajar en modo manual. Al crear estas animaciones se inserta un bit de stop a cada una para que no ejecute el bucle automáticamente sino que esté en reposo hasta nueva orden.

El modo de parada de emergencia estará compuesto por:

- **Emergencia.** Animación donde no hay movimiento, solo unos indicadores de que nos encontramos en una para de emergencia.

Paralelamente a estas dos maneras de operar con las animaciones, esta el modo de parada de emergencia. En caso de estar activa esta parada, inmediatamente todas las animaciones quedarían anuladas independientemente del modo en que se encuentren y ejecutaría la animación especifica que no desaparecerá hasta quedar solucionado el error.

7.3.3- Tiempo de las animaciones

Como se ha comentado en la introducción, un apartado muy importante es el tiempo de duración de cada animación para no crear un desfase considerable al ejecutar un proceso completo. Para calcular el tiempo se ha cronometrado los ciclos completos y cada movimiento por separado para después introduciros en el flash.

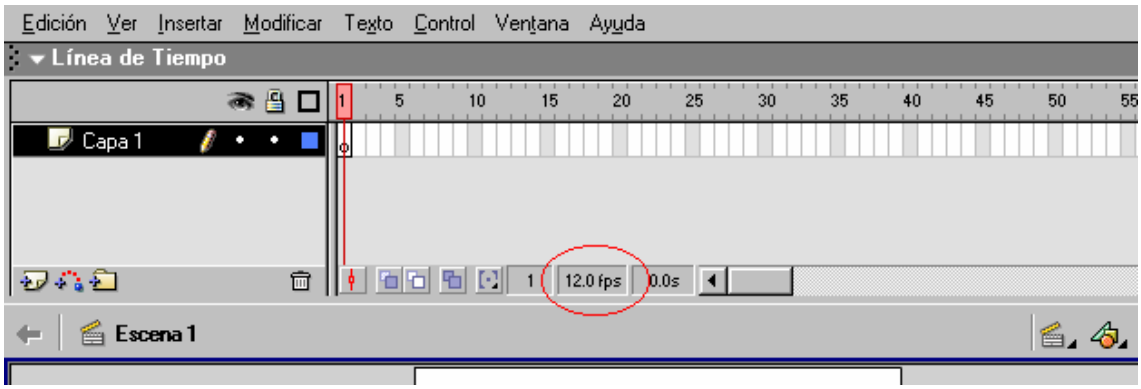


Fig. 7.8 Foto indicador de fps

Para modificar este valor se hace un doble clic encima para que aparezca una ventana emergente. En esta ventana se modifican los parámetros que se marcan en la figura de más abajo.

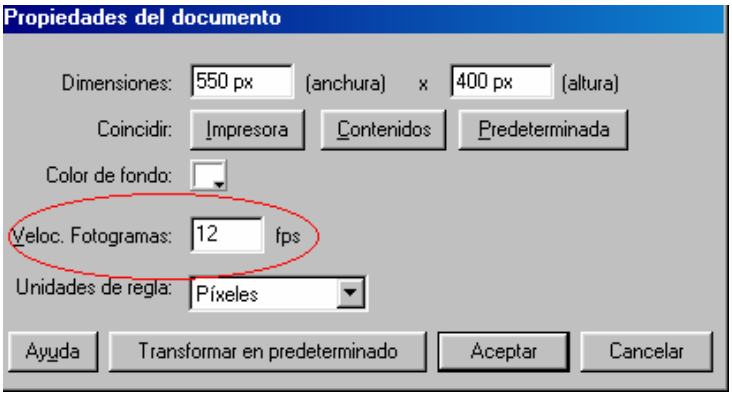


Fig. 7.9

A continuación se adjuntan las tablas con tiempos en segundos y los fotogramas por segundo que son los utilizados en flash.

MANUAL		
ANIMACION	**Tiempo (Fps)**	**Tiempo (segundos)**
Parada	No tiene	estática
Inicio	7 fps	18.1s
Bajar Pistón	20 fps	0.5s
Cierra Pinza	1 fps	0.2s
Subir Pistón	20 fps	0.5s
RetrocesoA	7 fps	18.1s
Bajar PistónA	20 fps	0.5s
Abrir PinzaA	1 fps	0.2s
Subir PistónA	20 fps	0.5s
RetornoA	7 fps	18.1s
RetrocesoB	7 fps	14.5s
Bajar PistónB	20 fps	0.5s
Abrir PinzaB	1 fps	0.2s
Subir PistónB	20 fps	0.5s
RetornoB	7 fps	14.5s

AUTOMATICO		
ANIMACION	**Tiempo (Fps)**	**Tiempo (segundos)**
Parada	No tiene	estática
Inicio	7 fps	18.1s
RetrocesoA	6.3 fps	38.1s
RetrocesoB	7.2 fps	26.2s
RetrocesoCorto	7.2 fps	3.6s

En la parada de emergencia no tendremos en cuenta este factor de tiempo debido a que es una animación en bucle que durará hasta que el problema quede resuelto, es decir que el tiempo puede tener un margen de oscilación muy amplio.

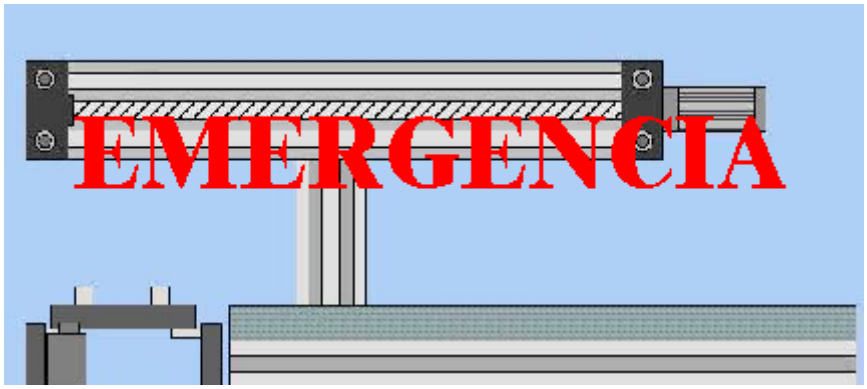


Fig. 7.10 Parada emergencia

7.4- Conceptos generales de los elementos del CX-Programmer para la carga de los archivos flash

Una vez disponemos de todas las animaciones realizadas, habrá que implementarlas en el Visual Basic. Al disponer de una sola cuadricula para cargar todas las animaciones en la pantalla, se tendrá que realizar un programa que en función de unas características determinadas (sensores, contadores, etc..) cargue una animación u otra. En visual Basic este programa no se puede hacer debido a que seria una programación casi imposible por la cantidad de opciones que puede haber a lo largo del funcionamiento del manipulador. De esta manera el programa se realizará en el Cx-Programmer mientras que el Visual Basic lo utilizaremos para leer las posiciones de memoria activadas y de esta manera cargar la animación adecuada.

Aprovechando el mismo programa que tenemos hecho para el control del manipulador, insertaremos otra línea de comandos denominada *animaciones*.

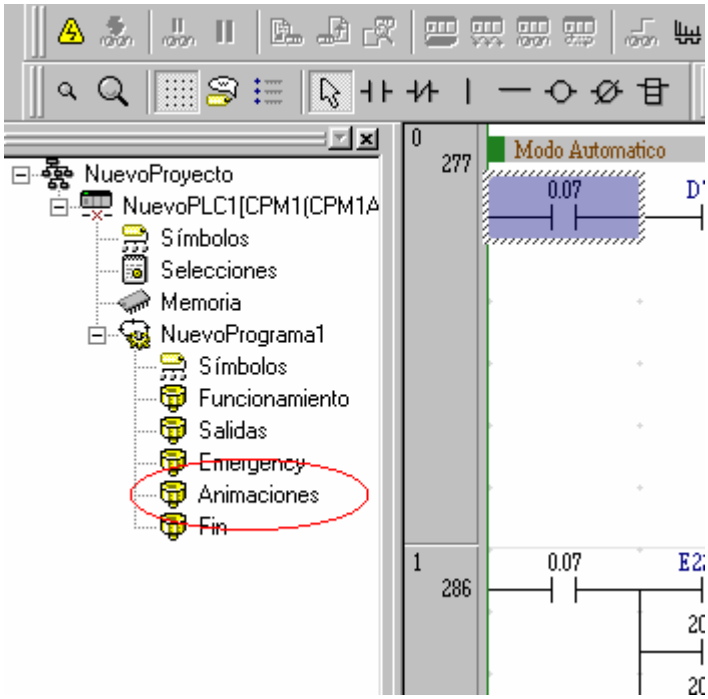


Fig. 7.11

Este programa irá paralelamente al del control del brazo para que este no pueda interferir en ninguna posición de memoria y dar conflicto de funcionamiento.

Al empezar a implementar el lenguaje de contactos en el Cx-Programmer, hay que destacar los diferentes subapartados que tenemos que tener. Estos subapartados irán en función de los estados en que se encuentre el brazo robot que son:

- *Modo automático*
- *Modo manual*
- *Para de emergencia*
- *De manual a automático*
- *De automático a manual*

Las tres primeras partes del programa irán independientes entre si, pero al realizar los posibles pasos de un estado a otro, implica que dependerán y habrá que realizar modificaciones de programa para que se pueda pasar de un estado a otro en cualquier momento. Hay que tener en cuenta que para el paso de un estado a otro, se puede realizar en cualquier momento, esto quiere decir que el programa tiene que estar dispuesto para leer todas las posibles combinaciones.

Las posiciones de memoria se activarán mediante un SET y se desactivarán por un RESET, de esta manera dispondremos de un grupo de memorias que serán las encargadas de activar o no cada imagen, debido a que cada memoria esta vinculada a una animación de flash. A parte del SET y RESET, el tercer bloque utilizado es el TIM puesto que en algunas posiciones precisamos de un cierto retardo de activación.

Una vez sabemos que las memorias vinculadas a las animaciones las activaremos a través de los diferentes SETs, tenemos que saber como activar dichas memorias. Los diferentes SETs los activaremos dependiendo de los diferentes sensores, contadores, temporizadores, etc. que nos encontramos en nuestra estación de trabajo. Así pues para cargar la animación de bajar pistón en la posición final por ejemplo, se tendría que encontrar activo el selector (manual o auto), el sensor de final de posición, pinza cerrada y de esta manera con todas las posiciones de memoria con que trabajamos.

7.5- Implementación en Visual Basic para la carga de animaciones en flash

Para poder utilizar la animación correcta en el momento adecuado al movimiento que realiza la estación, es necesaria la aplicación UDP Driver que ya utilizamos para la comunicación con el PLC descrita anteriormente, puesto que será el PLC mediante diferentes posiciones de memoria quien determinará en qué posición están los elementos de la estación. De este modo, el VisualBasic en ningún momento controlará las animaciones sino que se limitará a hacer lecturas de las memorias y en cuanto se active la memoria determinada por el PLC, únicamente cargará la animación previamente indicada sin poder manipular lo que ésta contenga.

El comando UDP Driver no viene implementado en Visual Basic, para ello se ha de cargar una nueva librería que nos permita hacer estas lecturas de memoria. Para cargar esta librería hay que abrir en el Visual Basic la barra superior de herramientas, y en la pestaña *Proyecto* > *Componentes* entraremos en el menú de las librerías .OCX. En la pestaña donde pone componentes, buscamos el archivo que pone *Lancelan UDPDriver* y le damos a aplicar.

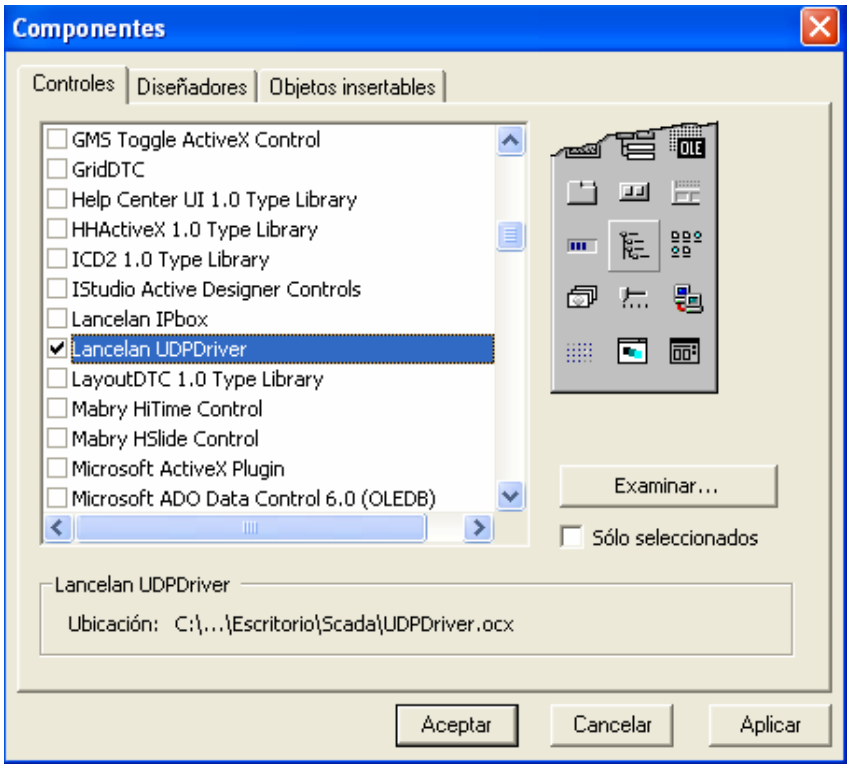


Fig. 7.12

En las propiedades del UDP Driver insertamos las diferentes posiciones de memoria con sus respectivos nombres indicados en las siguientes tablas para los estados automático y manual:

AUTOMÁTICO		
Núm. Animación	**Movimiento**	**Posición de Memoria**
1	Parada	IR202.01
2	Inicio	IR202.02
3	Retroceso A	IR 202.03
4	Retroceso B	IR 202.04
5	Retroceso Corto	IR 202.06

MANUAL		
Núm. Animación	**Movimiento**	**Posición de Memoria**
1	Parada	IR 202.01
02	Inicio	IR 203.01
03	Bajar Pistón	IR 203.02
04	Cerrar Pinza	IR 203.03
05	Subir Pistón	IR 203.04
06	Retroceso B	IR 203.05
07	Bajar Pistón B	IR 203.06
08	Abrir Pinza B	IR 203.07
09	Subir Pistón B	IR 203.08
10	Retorno B	IR 203.09
11	Retroceso A	IR 203.10
12	Bajar Pistón A	IR 203.11
13	Abrir Pinza A	IR 203.12
14	Subir Pistón A	IR 203.13
15	Retorno A	IR 203.14

-Método para cargar la animación en el sistema supervisor:

Para que sea posible una comunicación constante entre los diferentes movimientos de la estación y el sistema supervisor, es necesario que la parte de programación para cargar la animación esté incluida en la rutina UDPDriver_Response. Esto permitirá que para cualquier variación en el estado de las memorias del PLC se provoque una interrupción en esta rutina de tal modo que podamos hacer una lectura de la posición de memoria que el PLC considera necesaria para la ejecución de la animación correcta.

La forma de programar en el VisualBasic es la siguiente:

```
Private Sub UDPDriver1_Response(ByVal Item As String, ByVal Value As String)

Case "1.Parada"

    If Value = "1" Then
       Flash1.Movie = (App.Path & "\1.Parada.swf")
    End If

End Select

End Sub
```

De este modo, cuando el PLC de el valor a la memoria “1.Parada” (configurada previamente en el UDPDriver) el valor de estado “1”, se activará el comando que contenga el *Case* y mediante el comando Flash1.Movie se cargará la animación “1.Parada.swf” que estará ubicada en la misma carpeta que el resto del programa.

7.6- Pantalla de programa

El sistema de supervisión presenta un aspecto compacto para un manejo sencillo y a su vez rápido. La pantalla del programa se ha dividido en cuatro partes para que cada una aloje un control concreto e independiente entre si. De esta manera se ha conseguido un acceso rápido a todos los posibles controles sin que se haya de ir abriendo y cerrando distintas ventanas. Como se aprecia en la figura 8.12 a la vez que se controlan los diferentes estados del programa, observamos los estados directos tanto del Manipulador como del robot.

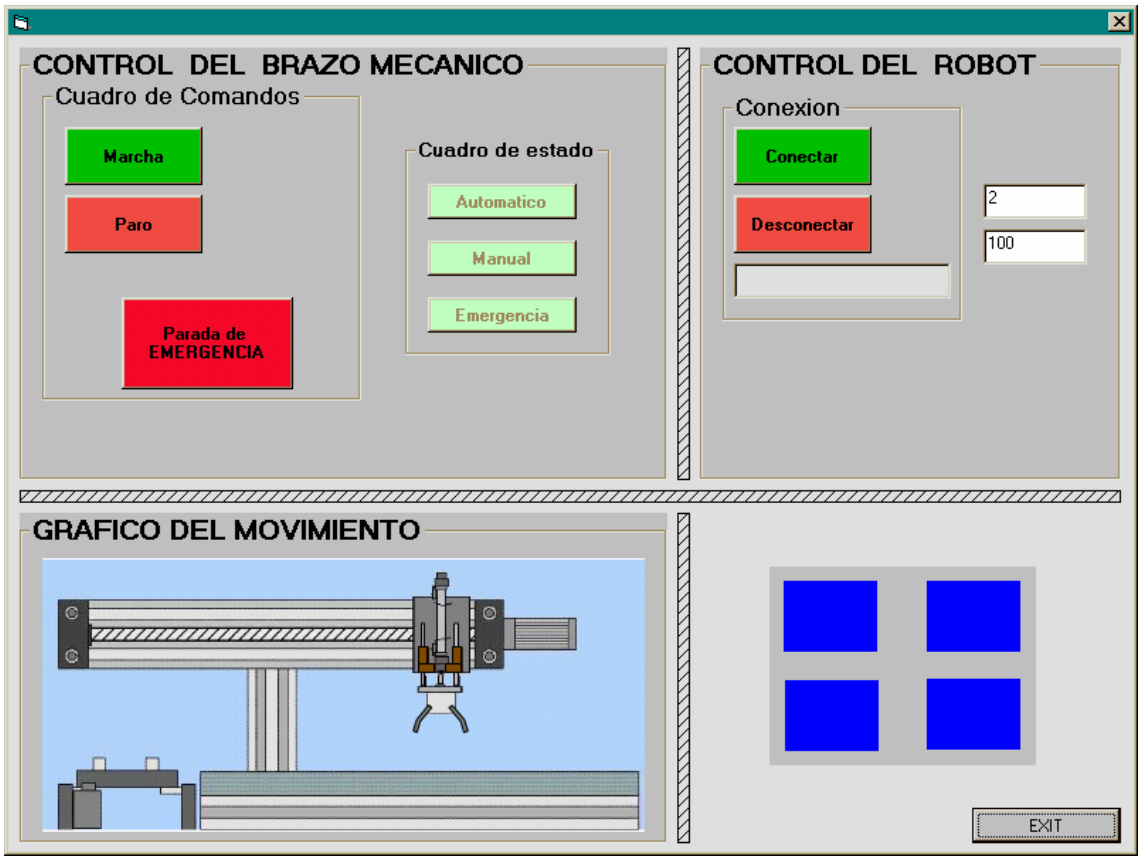


Fig. 7.13 Presentación del sistema de supervisión

Las cuatro partes en que se encuentra dividida la pantalla son las siguientes:

7.6.1- Control del Manipulador

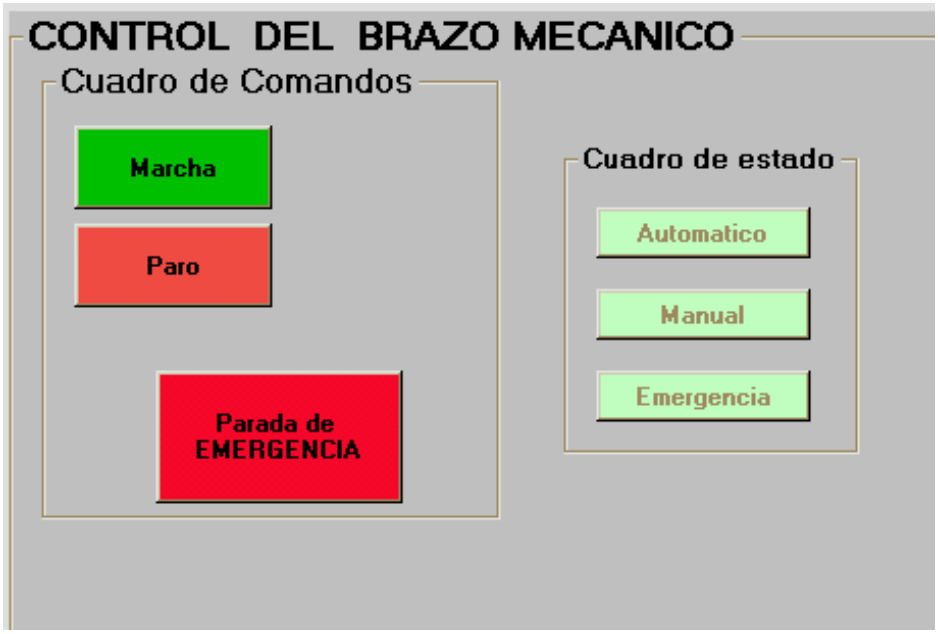


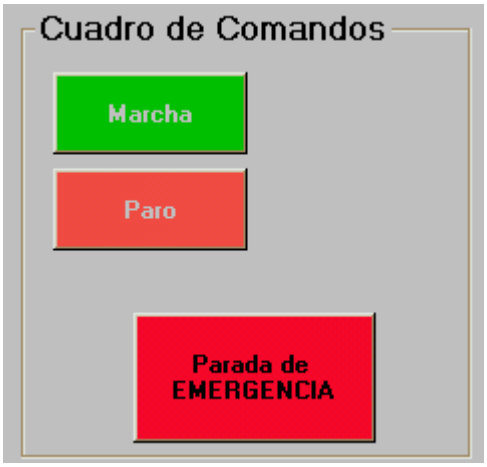
Fig. 7.14 Control del Manipulador

Esta parte del programa es la encargada de interactuar con el Manipulador. Su función global es la misma que la botonera situada en la estación de trabajo, de esta manera podemos controlar el manipulador desde el PC sin tener que depender de dicha botonera.

En la figura 8.13 se aprecia como esta subdividida en dos partes, el cuadro de comandos y el cuadro de estado. Como su nombre indica, el primer cuadro nos permite realizar el control de los diferentes movimientos del manipulador, formado por tres botones con una funcionalidad concreta, *marcha*, *paro* y *parada de emergencia* mientras que el segundo es el encargado de mostrarnos mediante unos LED's el modo de funcionamiento en que se encuentra el manipulador en cada momento. Al activarse uno de los estados vemos como el LED se ve modificado.

- **Cuadro de comandos**

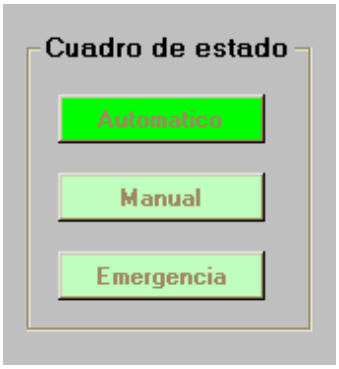
Al manipular los botones de Marcha y Paro no se ven modificados, es decir que no sufren ningún efecto visual. Por el contrario, la parada de emergencia si que sufre modificación.



En la imagen se puede apreciar como al pulsar la para de emergencia, inmediatamente los botones de marcha y paro pasan a bloquearse y no podrán estar activos hasta que la causa de la parada de emergencia quede solucionada.

Fig. 7.15

- **Cuadro de estado**



Los estados de automático y manual tienen el mismo efecto visual al activarse, un color un poco mas intenso en caso de validarse el estado correcto.

Fig. 7.16

Al activarse la parada de emergencia, el indicador es diferente a los del funcionamiento manual o automático. En este caso el LED se activará con un color rojo intenso que ofrece un efecto visual más contrastado.

Fig. 7.17

Al activarse este indicador de parada de emergencia, la pantalla principal se vera modificada.

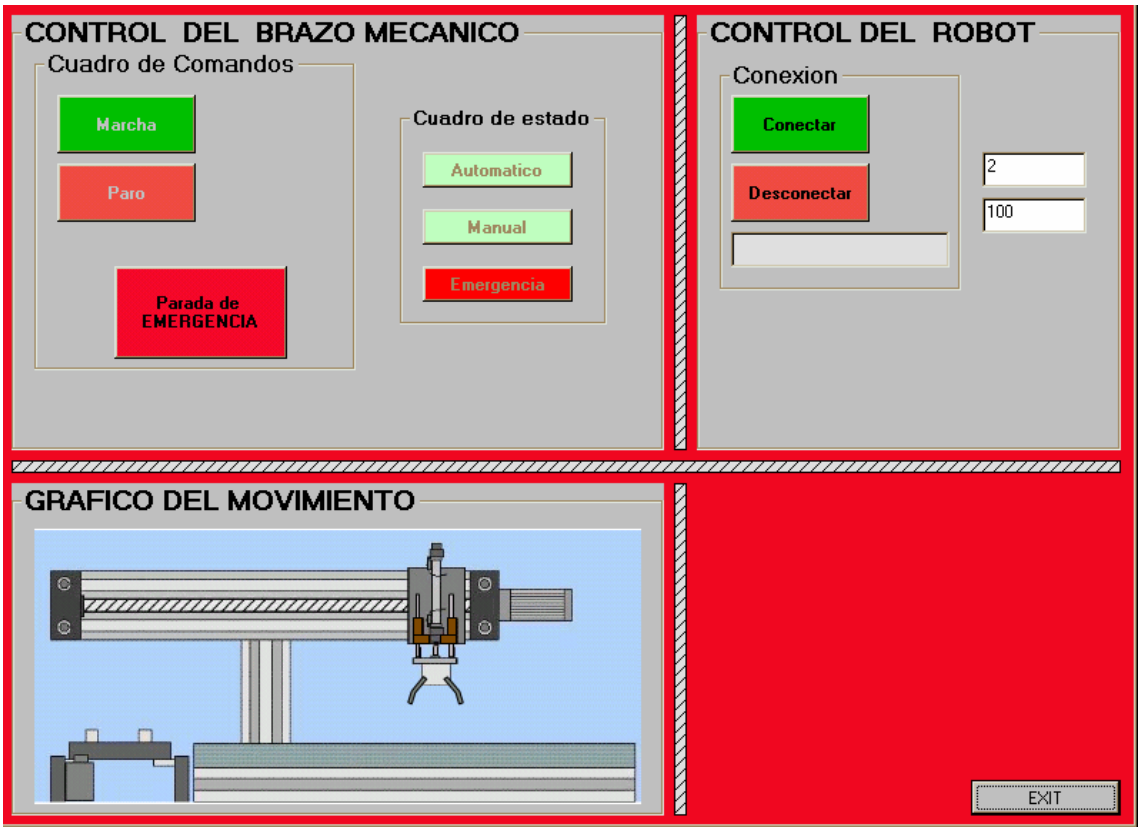


Fig. 7.18

7.6.2- Control del Robot

Esta parte del programa se encarga de controlar el robot. Al ser un proceso semiautomático debido a que el movimiento del robot esta controlado por el PLC, el usuario solo se limitará a conectar o desconectarlo del puerto serie con los botones de *conectar* y *desconectar*.

El display inferior indica si el sistema se encuentra conectado o no.

Fig. 7.19

7.6.3- Gráfico de movimiento

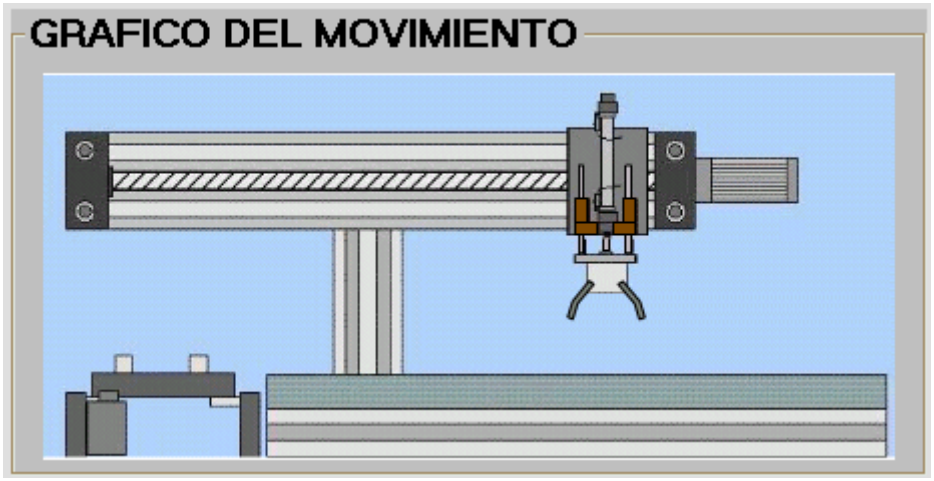


Fig. 7.20

Esta parte de la pantalla esta compuesta por una única ventana que es la encargada de mostrar el movimiento y función del manipulador en cada momento. Las imágenes mostradas son animaciones creadas con flash, de esta manera el programa nos ofrece la posibilidad de visualizar los siguientes estados:

- *Modo automático*
- *Modo manual*
- *Para de emergencia*

Además de estos estados, también se visualizarán las diferentes transiciones entre fases.

7.6.4- Gráfico de cajas

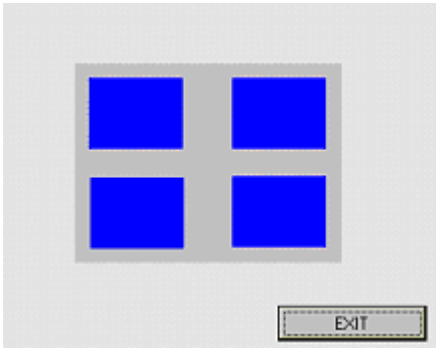


Fig. 7.21

Únicamente está formado por un gráfico y un botón. El gráfico hace referencia a las cajitas depositadas en el palet de manera que conforme se vayan depositando dichas cajas, se activará un bloque azul. Al completar los cuatro bloques azules, el palet estará listo para ser almacenado y dar entrada a otro palet nuevo.

Por otra parte el botón que encontramos es el de EXIT y como su nombre indica nos permite salir del programa.

- Limitaciones

- Incidencias

- Posibles ampliaciones

8- CONCLUSIONES FINALES

8.1- Limitaciones

A pesar de que teóricamente la comunicación entre el PLC y el sistema supervisor debe ser instantánea, a la práctica no sucede así. Cuando hay un cambio en las memorias del PLC, tarda unas décimas de segundo en establecer la comunicación, enviar el valor por parte del PLC, y recibir la transmisión de datos el ordenador. Todo esto comporta que no haya cambios instantáneos en el sistema supervisor, por consecuencia, se puede apreciar un pequeño desfase a la hora de ejecutar las diferentes animaciones.

Además, todo este problema se agrava a la hora de cargar las animaciones en flash y visualizarlas en el sistema supervisor, sobre todo en las ocasiones más extremas. Por ejemplo, cuando la estación se pone en marcha y tarda un tiempo en enviar, se provoca un desfase en el movimiento entre el manipulador y su animación en el supervisor con la consecuencia de que el manipulador llegue a su punto final y la simulación siga en movimiento.

8.2- Incidencias

- La alimentación del brazo robot se hace mediante una pila de 9 voltios para la controladora y 2x1,5 voltios para los servos. Respecto a la alimentación de los servos sucede que consumen rápidamente las 2 pilas y para un funcionamiento normal es un problema cambiar las pilas constantemente. Por tanto, se ha modificado este sistema y se han substituido las pilas por una fuente de alimentación de 6 voltios en C.C.

- En algunos momentos el PLC puede generar un error mediante el cual queda totalmente bloqueado. Para resolver esta situación hay que poner a cero la memorias IR y DM del PLC, y a continuación hacer un reset al PLC mediante cortarle la alimentación por unos segundos.

- Problemas de los servos del brazo robot, el control de posición de los brazos, se realiza en lazo abierto, no se dispone de un instrumento de medida (como un potenciómetro) que nos indique la posición relativa de un brazo respecto al otro, ni respecto a la proyección del brazo sobre el plano.

8.3- Posibles ampliaciones

- Existe un comando que algunos PLC's contienen (el TXD), que permite el envío de datos directamente por el puerto RS-232C del PLC y daría la posibilidad de una conexión directa entre el PLC y el brazo robot. Pero debido a que el PLC CPM1-A que es el que estaba ya implementado en la estación de trabajo no permite esta transferencia, es necesario un intermediario que en nuestro caso es el PC. Por tanto, una posible ampliación podría ser sustituir el PLC CPM1-A por uno que permita esta conexión, como puede ser el CS1G de OMRON entre otros.

- La controladora que viene suministrada con el brazo robot (la MINI SSC S310165) únicamente permite la recepción de datos, con lo cual enviamos una serie de tramas mediante el ordenador y en ningún momento podemos conocer realmente si se ha recibido o ejecutado la información correctamente. Para poder ampliar la aplicación seria conveniente cambiar la controladora por una que permita una recepción y transmisión de datos entre el ordenador y el brazo robot.

- El rango de movimiento del brazo robot es limitado y únicamente permite 180º como máximo, lo que implica a la práctica en esta estación de trabajo que haya solo dos posiciones donde el manipulador deja la pieza en contraste con las cuatro que la antigua estación tenía. Por ello podría modificarse los servos y cambiarlos por otros que permitan un mayor rango de trabajo para que de este modo el brazo robot pueda acceder a más posiciones y con mayor flexibilidad en los movimientos.

- Incorporar sensores externos que permitan conocer en todo momento el estado del brazo robot, para poder corregir la trama enviada a cada uno de los servos y trabajar en lazo cerrado. Otras funciones de control de estos sensores externos serían el control de cada posición de piezas en el palet y la correcta colocación de estas en el palet.

- En caso de que el brazo dispusiera de mayor rango de movimiento, el sistema de almacenamiento se modificaría por otro mayor y más complejo como podría ser un sistema de estanterías fijas.

9- BIBLIOGRAFÍA

-**Direcciones WEB**:

www.lancelan.com *(Página oficial del módulo LanceLan)*
www.superrobotica.com *(Página oficial del brazo robot)*
www.lawebdelprogramador.com *(Página de consulta sobre Visual Basic)*
www.weblogs.golemproject.com/msdn *(Página de consulta sobre Visual Basic)*

-**Libros consultados**:

VISUAL BASIC 6.0 INICIACIÓN Y REFERENCIA; Antonio Muñoz Clemente, Luis Joyanes Aguilar (Ed. McGraw-Hill).

Manual avanzado Visual basic 6; Matías Blazquez (Ed. Anaya Multimedia).

FLASH MX; Besley, Kristian Bhangal, Sham Farr, Amanda(Ed. Anaya Multimedia).

Redes; Jesús Sánchez Allende, Joaquín López Lérida (Ed. McGraw-Hill)

-Manuales empleados:

Manual Lancelan

Manual OMRON CPM1-A