Escola Superior d'Enginyeries Industrials, Aeroespacial i Audiovisual de Terrassa
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Trabajo final de grado

**Grado en Ingeniería industrial electrónica y automática**

# ESTUDIO DE LAS ETAPAS DE AUTOMATIZACIÓN DE UN PROCESO INDUSTRIAL: COMUNICACIONES Y OPERACIÓN

**MEMORIA**

**Autor:** Nil Munuera Cano

**Director:** Miguel Delgado Prieto

**Co-Director:** Ángel Fernández Sobrino

**Convocatoria:** Enero 2020

## Resumen:

Se realiza un estudio de la sucesión de etapas que se llevan a cabo en la automatización de un proceso industrial.

Se inicia el estudio haciendo referencia a la evolución de la automatización industrial en los procesos productivos, y el efecto que supone la aprición de las tecnologías de comunicación.

Posteriormente se procede al estado del arte de los controladores lógicos programables, y su importancia en los sistemas automatizados.

Conociendo las características de los controladores, se realiza la Implementación de tecnologias de comunicación mediante los buses de campo. Introducción a los buses de comunicación profibus Dp, y CAN con sus respectivas tramas de datos, así como las redes ethernet y sus respectivos protocolos de aplicación en el modelo OSI.

Aplicar la infromación anterior para elaborar el diseño de los procesos de automatización de la célula flexible, para su posterior traducción a lenguaje de programación de dichos diseños.

Para finalizar, elaborar una validación de las pruebas realizadas en el diseño y programación de los automatismos.

## Abstract:

Study of the all of stages that are carried out in the automation of an industrial process.

The study begins by referring to the evolution of industrial automation in production processes, and the effect that it implies the aparition of new comunication tecnologies.

Subsequently it proceed to the state of the art of programmable logic controllers, and their relevance in automated systems.

Knowing the characteristics of the controllers, the implementation of communication technologies is developed through field buses. Introduction to comunication buses profibus Dp, and CAN with its respectives data frames, as well as ethernet networks and their application protocols in the OSI model.

Apply the information below to elaborate the design of the automation processes in the flexible cell, for its subsequent translation into the programming language of these designs.

Finally, prepare a validation of the tests performed in the design and programming of the automatisms.

## Agradecimientos:

Quiero expresar mi agradecmiento a MIGUEL DELGADO PRIETO como director del proyecto por darme la oportunidad de elaborar el trabajo final de grado en el laboratorio de automatización, así como el soporte en la realización del mismo.

También agradecer a ÁNGEL FERNÁNDEZ SOBRINO como coordinador del laboratorio por orientarme en el desarrollo de las aplicaciones.

En especial agradezo a mis familiares y a mi pareja TANIA por dedicarme su apoyo en todo momento durante la realización del proyecto.

# ÍNDICE:

# ÍNDICE DE ILUSTRACIONES:

# ÍNDICE DE TABLAS:

# 1. Introducción:

## 1.1. Descripción del trabajo:

El trabajo documentado en esta memoria consiste en un estudio de las etapas necesarias a la hora de implementar un automatismo programado sobre un proceso industrial, abarcando desde el diseño, la integración, la programación y validación.

En concreto, en este caso, el sistema sobre el que desarrolla el estudio es una célula flexible ubicada en el laboratorio de automatización industrial aplicada de la Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa. ESEIAAT, UPC de Terrassa.

## 1.2. Objetivos del estudio:

El objetivo principal es desarrollar las etapas y procedimientos técnicos necesarios para la correcta automatización de un proceso industrial, donde intervienen controladores programables, protocolos de comunicación, detectores y actuadores.

Para la consecución de este objetivo principal, se identifican tres objetivos parciales:

- La configuración e integración de los módulos disponibles para poder realizar la comunicación con el software de programación de los controladores programables, y hacer posible una aplicación secuencial sobre el proceso industrial a automatizar y su validación...
- Hacer que la célula funcione de manera automática, sin colisiones entre componentes.

Implementar una aplicación basada en la ordenación de bandejas según un número de producto, y a partir de dicha aplicación se procede al almacenamiento en el pulmón disponible en dicho orden, para luego descargar los palets del pulmón a la línea en un nuevo ciclo de trabajo.

## 1.3. Alcance del estudio:

El alcance del estudio propuesto incluye el seguimiento de las etapas del proceso industrial que se quiere automatizar, así, se contemplan los siguientes aspectos

- Descripción del proceso industrial a automatizar.
- Identificación de los elementos de control, detección, mando, y las señales del controlador lógico programable ya existentes.
- Uso de las comunicaciones industriales existentes: buses de campo que intervienen en el sistema.
- Diseñar los diagramas de flujo del automatismo programable.
- Programación del automatismo.
- Validación funcional y operacional.

## 1.4. Requisitos del estudio:

Para realizar el estudio se deben tener en cuenta los requisitos a nivel técnico y de conocimientos:

- Utilización de PLC de Schneider, software Unity Pro XL, y entorno físico de la célula flexible disponible.
- Conocimientos sobre buses de campo y redes Ethernet.
- Protocolo Modbus aplicado a RS-485 y TCP/IP.
- En términos de programación, robustez en los programas, y que funcionen de manera automática.

## 1.5. Contexto tecnológico:

Las revoluciones industriales indicaron transformaciones del modelo industrial, con el objetivo de mejorar los procesos productivos.

Durante la primera revolución industrial en el siglo XVIII, se produjeron avances tecnológicos que implicaron un desarrollo de maquinaria utilizando energía de vapor.

La segunda revolución industrial marcó cambios trascendentes de las ciencias y tecnologías en el siglo XX, como la aparición y explotación de los combustibles fósiles, y el uso de la energía eléctrica en la producción.

En la tercera revolución industrial surgieron las tecnologías de comunicación, explotando los recursos de la informática y la electrónica, con el objetivo de automatizar procesos industriales. Todo ello desembocó en la llegada del primer controlador lógico programable a la industria de la mano de Modicon, lo que representó un gran avance en la automatización industrial.

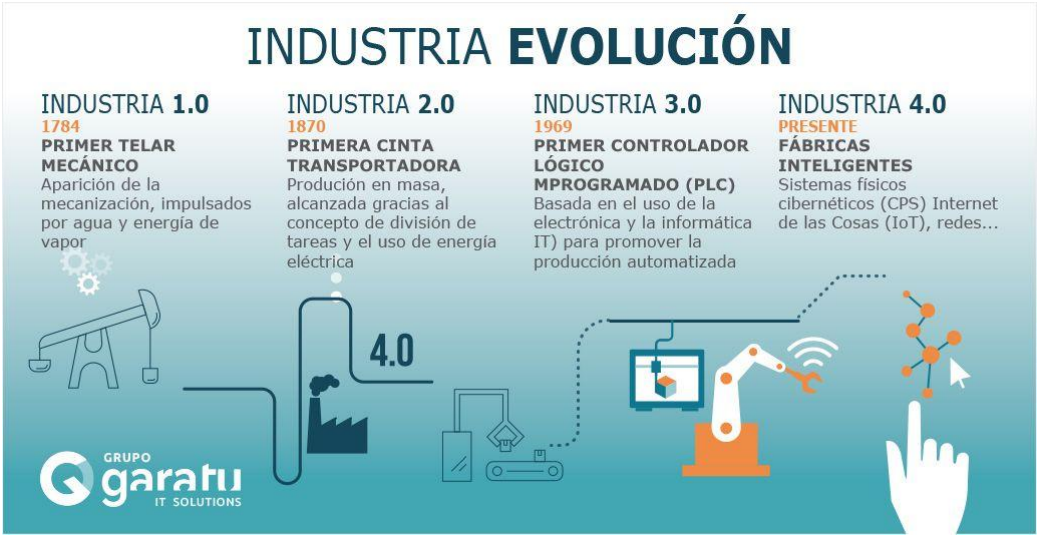



Ilustración 1: Evolución industria

La automatización industrial se inició con el objetivo de emplear máquinas industriales que fuesen capaces de realizar tareas repetitivas y mecánicas, las cuales siempre eran realizadas por el ser humano. Con ello se consigue una mayor eficiencia en dichas tareas.

La automatización industrial empezó teniendo un gran impacto en la industria del automóvil, ya que se pretendía lograr la reducción de los tiempos y costes de fabricación de automóviles, así como el incremento de la producción.

Posteriormente, con la implantación del PC y de internet, se pudo desarrollar más el campo de la automatización, y tal y como se ha mencionado anteriormente se introdujo en la industria el primer controlador lógico programable (PLC).

La aparición de las tecnologías de comunicación y los buses industrial, con sus respectivos protocolos de red generó la urgencia de transformar los procesos digitales, de tal forma que la arquitectura de los sistemas de automatización industrial pudiera interactuar con los sistemas de comunicación en un proceso productivo.

Para conseguir establecer una eficiencia en los procesos productivos, se elaboró una estructura de niveles jerárquicos, con el objetivo de definir una red industrial con mejor fiabilidad, así como estandarizar las características de las tecnologías de comunicación.

### 1.5.1. Industria 4.0:

La industria 4.0, también conocida como la cuarta revolución industrial, determina una nueva etapa en la industria de la humanidad, donde la automatización industrial juega un papel de máxima importancia.

La industria 4.0 se centra sobre la implementación de nuevas tecnologías, con la implantación de fábricas inteligentes (Smart Factories), con el objetivo de hacer que las máquinas intercambien información entre sí, y poder adaptarse a los requerimientos de la producción en tiempo real.

Para ello es necesario un alto nivel de comunicaciones entre los equipos de red y los medios de producción.

# 2. Consideraciones previas, el controlador lógico programable:

Un PLC es un computador industrial programable que mediante el procesamiento de unas señales de Entrada/salida, permite controlar un proceso industrial secuencial de forma automática.

El PLC almacena información de las señales de entrada (captadores), y la trasmite a las señales de salida (actuadores), para que en función de los captadores y las ordenes de programación se pueda controlar un proceso industrial de forma secuencial.


Ilustración 2: Arquitectura de un PLC

## 2.1. Antecedentes:

Los PLC se implementaron en los años 60 para la sustitución de la circuitería con relés lógicos electromecánicos e interruptores utilizando la lógica combinacional, que eran los sistemas convencionales en aquella época, en la cual la tecnología no había llegado hasta el punto de poder prescindir del conexionado punto a punto.

El fin con el que los PLC se implementaron fue para poder conseguir que un sistema pueda trabajar de forma autónoma mediante unas órdenes de control dadas por un usuario, y ejecutadas por el controlador. De esta forma también se consigue liberar al ser humano de tareas mecanizadas y repetitivas.

En los años 70 surgieron los PLC en la automoción, para conseguir reducir los costes de fabricación de automóviles, ya que se necesitaba un sistema con robustez, y que fuese modificable de manera sencilla. Así surgió el primer PLC, que fue nombrado MODICON 084, el cual fue utilizado en procesos industriales en 1968.

A partir de entonces aparecieron los protocolos de comunicación, para que fuera posible formar una red de controladores, donde los PLC pueden intercambiar información entre ellos, lo que hace más sencillo el control de un proceso industrial en tiempo real.

El PLC trabaja con una serie de instrucciones, que según las ordenes de control, permite la ejecución de aplicaciones, las cuales hace posible el funcionamiento de un proceso industrial de forma secuencial.

Para la interacción del usuario sobre el PLC, se diseñó el lenguaje Ladder, que es muy intuitivo y similar a los diagramas esquemáticos de los circuitos con lógica cableada mediante relés, con lo cual resultaba más sencilla la programación de un PLC.

### 2.1.1. Evolución:

Actualmente los PLC son un referente a nivel de automatización de procesos, ya que están dotados de sistemas de comunicación entre sí, considerándolos ordenadores industriales. La tendencia actual es crear sistemas industriales utilizando redes de controladores programables.

También se han desarrollado sistemas SCADA, donde la interfaz HMI hace posible la monitorización y control de un proceso productivo en tiempo real, mediante pantallas táctiles.

### 2.1.2. Fabricantes:

- **Siemens:** La compañía alemana Siemens es una de las principales en la fabricación de autómatas en el ámbito internacional. Fabrican PLC conocidos como S7, también disponen de software de alto nivel como Tia Portal.
- **Schneider electrics:** Fabricante francés también importante en el sector de la automatización industrial. En este proyecto todos los PLC utilizados son de tipo BMX de la marca Schneider con el software Unity Pro XL.
- **Omron:** La empresa japonesa Omron surgió como fabricante de componentes electrónicos, pero ha extendido su mercado con la fabricación de PLC industriales.

### 2.1.3. Tipos de PLC:

En la industria existen diferentes tipos de PLC, aunque se pueden diferenciar en 2 grandes tipos:

- **Modulares:** Estos PLC de gama alta están divididos en ciertos módulos, lo que les hace más flexibles.

  De esta forma resulta sencillo ampliar el sistema, ya que se pueden añadir módulos en el caso de que se necesiten mas señales de entrada/salida en el proceso, o cualquier otra necesidad de ampliación.

Ilustración 3: PLC Modular

- **Compactos:** PLC basados en una estructura compacta de todos los elementos, con señales de entrada/salida limitadas.


Ilustración 4: PLC compacto

### 2.1.4. Elementos de un PLC:

Un controlador lógico programable consta de diferentes bloques interconectados entre sí, los cuales tienen diferentes funciones en un proceso de ejecución de los programas de usuario:

#### 2.1.4.1. Unidad central del proceso (CPU):

Un elemento fundamental de un PLC es la unidad central del proceso (CPU), que es considerado el cerebro del PLC, donde se halla el microprocesador, que hace posible la correcta lectura del programa de usuario ejecutando un control de errores de software y hardware.

Previamente a la ejecución del programa ejecuta la lectura de las entradas, las cuales son almacenadas en la memoria RAM volátil. Los datos del programa de usuario son almacenados en la memoria interna de la CPU (usualmente EEPROM).

- **Ciclo de SCAN:** tiempo que tarda un autómata programable en ejecutar todas las operaciones de programa:


Ilustración 5: Ciclo SCAN de un PLC

Las salidas del programa se actualizan en cada ciclo de Scan del autómata.

#### 2.1.4.1.1. Tipos de memoria:

- **Memoria RAM:** memoria de acceso aleatorio. Dicha memoria almacena los datos del programa de usuario durante su evaluación. Es una memoria volátil, con lo cual su contenido desaparece una vez se pierde el suministro de energía eléctrica con la CPU.

- **Memoria EPROM:** memoria de solo lectura, programable y se puede borrar. La memoria EPROM es no volátil, con lo cual conserva el contenido cuando hay desconexión eléctrica de la CPU.
  En esta memoria se almacena el programa de usuario una vez generado y analizado. Para eliminar el contenido de la EPROM es necesaria la exposición a rayos ultravioletas.

- **Memoria EEPROM:** similar a la EPROM, pero esta memoria puede ser borrada y reprogramada eléctricamente, no es necesario emitir rayos ultravioletas.

### 2.1.4.2. Módulos E/S:

Los Módulos de entrada/salida tienen la función de conectar las señales del PLC con los componentes físicos del sistema.

Los módulos de entrada/ salida pueden ser analógicos o digitales, cuya elección se hace según las señales que se utilicen en el proceso industrial.

Los módulos de entrada recogen la información del estado de las señales de entrada, y en función de dichas señales, por órdenes de programa, ejecutan la activación de las salidas correspondientes del modulo de salidas.

### 2.1.4.3. Bastidor:

El rack (bastidor), es un elemento sobre el que se montan todos los módulos del PLC. Dispone de slots, que son las ranuras donde se introduce el modulo correspondiente.

El bastidor se encarga de que mecánicamente los módulos estén correctamente fijados, y también las interconexiones eléctricas necesarias entre algunos módulos.

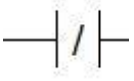
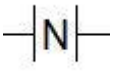
## 2.2. Lenguajes de programación:

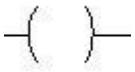
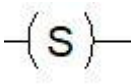
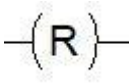
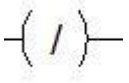
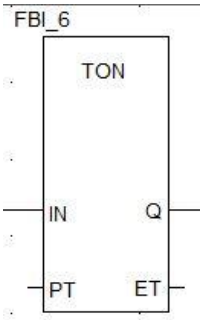
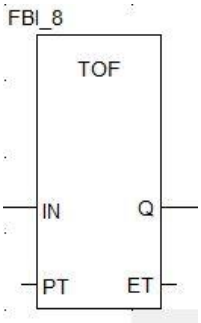
Cuando aparecieron los PLC, surgió el lenguaje ladder, que es muy intuitivo y similar a los diagramas esquemáticos de circuitos con automatismos cableados (relés).

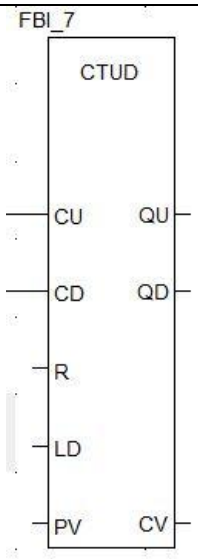
Actualmente existen más tipos de lenguajes de programación. Algunos de los más utilizados en la industria son:

- **Ladder:** Lenguaje de contactos anteriormente nombrado. Fue el primer lenguaje de programación que surgió con los PLC, dada su similitud con un esquema de relés. Algunos elementos relevantes són:

Tabla 1: lenguaje ladder

| Identificación | Símbolo | Función |
|---|---|---|
| Contacto NO |  | Tipo BOOL. Estado 0 en reposo. Transfiere conexión cuando está en estado 1. |
| Contacto NC |  | Tipo BOOL. Estado 1 en reposo. Transfiere conexión cuando está en estado 0. |
| Flanco positivo |  | Transfiere conexión en la transición de estado 0 a 1 de la variable asociada, durante un ciclo de scan. |
| Flanco negativo |  | Transfiere conexión en la transición de estado 1 a 0 de la variable asociada, durante un ciclo de scan. |

| | | |
|---|---|---|
| Bobina |  | Se activa cuando hay conexión en la entrada, y se transfiere a la salida. |
| Bobina set |  | Se activa cuando hay conexión en la entrada. Su estado de activación es 1 y aunque pierda la conexión permanece en estado 1 hasta que se active el reset de la misma variable. |
| Bobina reset |  | Se activa cuando hay conexión en la entrada. Su estado de activación es 0, y aunque pierda la conexión permanece en estado 0. |
| Bobina negativa |  | Se activa cuando hay conexión a la entrada. Su estado de activación es 0. Cuando no está activa, el estado de la variable asociada es 1. |
| Temporizador TON |  | Temporizador con retardo a la conexión:<br>• IN: cuando se produce conexión a esta entrada empieza a contar el tiempo.<br>• Q: salida del temporizador. Se activa una vez transcurre el tiempo.<br>• PT: tiempo establecido.<br>• ET: tiempo actual. |
| Temporizador TOF |  | Temporizador con retardo a la desconexión:<br>• IN: cuando se produce conexión a esta entrada empieza a contar el tiempo.<br>• Q: salida del temporizador. Se activa cuando recibe conexión en la entrada IN. Se desactiva una vez transcurre el tiempo.<br>• PT: tiempo establecido.<br>• ET: tiempo actual. |

| | | |
|---|---|---|
| Contador CTUD |  | Contador incrementa/decremental:<br>• CU: incrementa contador.<br>• CD: decrementa contador.<br>• R: reset contador.<br>• LD: cargar valor al contador.<br>• PV: valor de preselección del contador.<br>• CV: valor actual de conteo.<br>• QU: activación salía cuando se realiza un conteo en CU.<br>• QD: activación salía cuando se realiza un conteo en CD. |

El lenguaje ladder está organizado por segmentos, que són ejecutados de arriba hacia abajo y de derecha a izquierda. Un ejemplo de un segmento sencillo en una línea de programa sería:

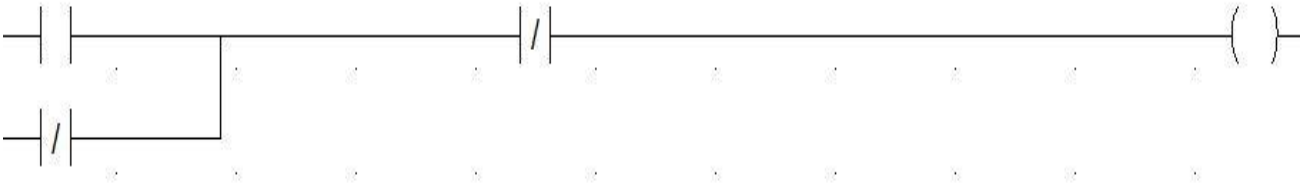


Ilustración 6: segmento ladder

- **ST:** Lenguaje de alto nivel similar a C, lenguaje de texto estructurado con sentencias de control, bucles, etc.
  Es un lenguaje que permite multitud de posibilidades de programación distinta, y aporta más robustez a las aplicaciones, ya que tiene menos restricciones que el resto de lenguajes.
  Algunas de las sentencias más utilizadas són:
  - **IF:** Sentencia de control, que ejecuta una acción si se cumple una condición.
    - **ELSE:** en el caso de que no se cumpla la condición, ejecuta una acción diferente.

```
IF (suma=1) THEN
    variable:=variable+1;
    ELSE
    variable:=0;
END_IF;
```

Ilustración 7: ejemplo sentencia IF/ELSE

El ejemplo indica que si la variable booleana vale 1, otra variable se suma una unidad, y si la condición no se cumple pone la variable a 0.

- **While:** bucle de repetición. Mientras una condición sea cierta, ejecuta una acción cíclicamente hasta que no se cumpla la condición.

```
WHILE (suma=1) DO
    variable:=variable+1;
END_WHILE;
```

Ilustración 8: Bucle WHILE

El ejemplo indica que mientras la variable booleana suma sea otra variable se irá sumando una unidad hasta que la condición no sea cierta.

- **FOR: bucle de repetición.** Se utiliza con una variable contador que indica el número de veces que realiza el bucle, o hasta que se cumpa una condición predeterminada.
  Es útil para moverse por las posiciones de un vector.

```
FOR variable:=1 TO 10 DO
    vector[variable]:=1;
END_FOR;
```

Ilustración 9: Bucle FOR

Como se puede ver, se utiliza una variable que marca la posición actual, y hace la función de recorrer el vector y poner todos los coeficientes del vector de 10 posiciones con un valor 1.

- **SFC:** lenguaje de programación mediante diagramas de bloques secuencial (GRAFCET), en el que intervienen 3 elementos importantes:
    - Etapas: división del proceso industrial en pequeños subprocesos.
    - Transiciones. Condiciones para poder cambiar de etapa.
    - Acciones: activaciones de señales de salida en cada etapa, por ejemplo activar una electroválvula de un cilindro neumático, o un piloto de señalización.

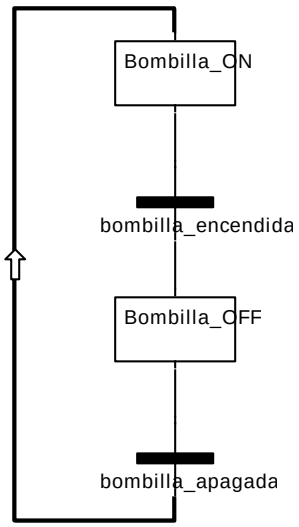



Ilustración 10: Lenguaje SFC

# 3. Consideraciones previas, las comunicaciones Industriales:

En la industria las comunicaciones juegan un papel fundamental para que un proceso industrial automatizado pueda funcionar a pleno rendimiento, ya que es la forma en la que los diferentes sistemas que intervienen en el proceso pueden comunicarse entre sí, transmitiendo la información de todo el sistema a tiempo real.

Existen 3 características importantes en la aplicación de un sistema de comunicación de datos industrial:

- **Velocidad de datos:** cantidad de datos en la red por cada envío.
- **Velocidad de transmisión:** velocidad a la que se produce la transferencia de datos en la red.
- **Velocidad de respuesta:** velocidad dada por el tiempo que tarda en responder el dispositivo a una orden realizada.

Dado que en cada sistema puede existir un grado de restricción en estas características, existen unos niveles de comunicación de datos para poder cumplir con las exigencias en cada caso.

La pirámide de comunicaciones (CIM), establece una estructura de niveles en la fabricación. Los cinco grandes niveles son:

- **Fábrica:** formado por ordenadores con gran volumen de datos tanto a nivel de fábrica, como de ingeniería. Redes WAN.
- **Planta:** formado por ordenadores a nivel de redes LAN.
- **Célula:** es una ampliación del nivel de campo**.**
- **Campo**: incluye los PLC's, con buses de campo (profibus, CAN, etc…).
- **Proceso:** Buses de sensores y actuadores (A-si).

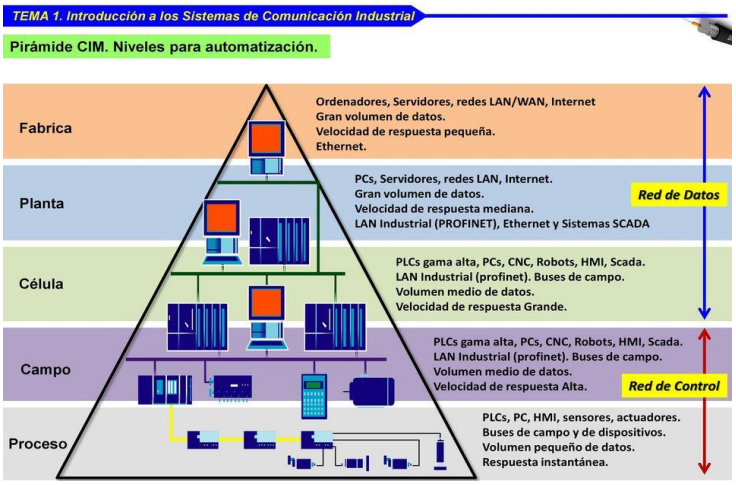



Ilustración 11: Pirámide CIM

## 3.1. Norma física RS-485:

RS-485 es un estándar de comunicaciones ubicado en la capa física de comunicación, la cual fue introducida en 1983.

Se trata de una interface de topología multipunto capaz de permitir la comunicación entre 32 equipos con emisor y receptor en un bus de comunicación. Dado que el bus de datos es común para todos los equipos, se dispone de un estado de habilitación para evitar posibles colisiones entre datos en el canal de comunicación.

La norma RS-485 igual que sus predecesoras, también permite eliminar los ruidos que pueden interferir en el canal de comunicación.

Los niveles de las señales eléctricas de la norma RS-485 son:

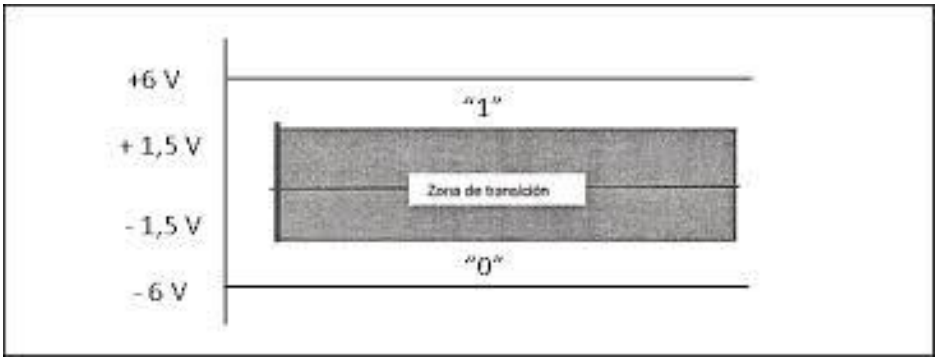



Ilustración 12: Niveles señales eléctricas interface RS-485

**Protocolo Half-Duplex:**

La norma RS-485 utiliza un protocolo de transmisión de datos Half-Duplex, cuyas características son:

- Conexión a 2 hilos, con masa común.
- Transmisión Half-Duplex: cada equipo puede enviar y recibir, sin embargo no es posible simultáneamente.
- Norma RS-485 permite la configuración de una red de hasta 64 dispositivos, con 32 emisores y 32 receptores.
- Dato resultante de comunicación obtenido de la diferencia de señal eléctrica entre los dos hilos del canal de comunicación (señal diferencial balanceada).
- Velocidades permitidas de hasta 100 Kbps para distancias de 1200 m, y hasta 10 Mbps para distancias de 15 m.

El medio físico utilizado en la norma RS-485 es el cableado con par trenzado tipo UTP, que consiste en grupos de conductores de cobre con aislamiento entrelazados en pares, de tal forma que se evitan las interferencias de señales externas, y así se consigue una mejor transmisión de datos.


Ilustración 13: Par trenzado UTP

En la norma RS-485 se utiliza en este caso un par trenzado apantallado, ya que al ser una trasmisión Half-Duplex no se pueden enviar datos y recibir al mismo tiempo.

**Métodos de transmisión:**

- **Transmisión serie:**
  - La transmisión se realiza bit a bit, con una línea de comunicación.
  - Es la más utilizada para transmisión de datos a larga distancia.
  - Volumen de datos bajo.
- **Transmisión paralelo:**
  - Transmisión se realiza de carácter (8 Bytes), a otro carácter.
  - Transmisión de bits simultanea.
  - Necesidad de las mismas líneas de comunicación que numero de caracteres.
  - Longitudes bajas (máximo 20 metros).
  - Velocidades más elevadas que en la transmisión serie.

Pese a que la transmisión en paralelo posee altas velocidades, en la norma RS-485 se utiliza con transmisión serie, ya que resulta mucho menos costosa su instalación. Además al ser la norma RS-485 una interface física industrial, la comunicación serie posee más fiabilidad.

**Transmisión serie:**

En la transmisión serie se necesita un canal de comunicación entre emisor y receptor.

Para la sincronización entre los equipos de la red existe una señal de reloj, que se puede configurar de dos formas diferentes: asíncrona y síncrona.

- **Transmisión serie asíncrona:** Configuración local de los parámetros de cada equipo de la red (señal de reloj deshabilitada).
  - **Bit de START:** Bit en estado 0 lógico. Es una señal enviada por el emisor de datos a los demás equipos, indica el empaquetamiento de datos.
  - **Bits de datos:** Indica la longitud de bits de cada carácter de transmisión según el código ASCII (configuración 7 u 8 bits).

- **Bit de paridad:** bit enviado posteriormente a los bits de datos. Realiza un control de errores.
  - **Estados bit paridad:**
    - Paridad PAR
    - Paridad IMPAR
    - No hay paridad (NONE).

- **Bit de STOP:** Bit que se envía al final de la comunicación, posterior al bit de paridad, si existe. Este bit pone la señal eléctrica a nivel alto (bit estado 1 lógico), que es el estado normal de la señal de comunicación cuando no hay ningún dato en transmisión.

  Posteriormente se puede ver un ejemplo de una trama de comunicación serie asíncrona:

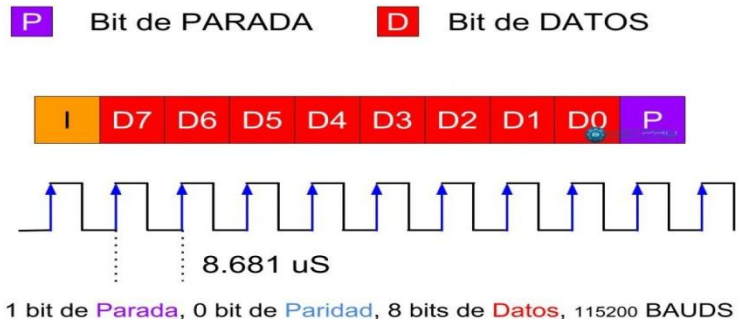



Ilustración 14: Trama Serie asíncrona

- **Transmisión síncrona:** en este tipo de transmisión se configura la señal de reloj individual para cada emisor con el resto de equipos de la red. La configuración de la señal de reloj puede ser de dos tipos:
  - **Señal de reloj independiente:** es una señal que une el emisor con el resto de equipos y sincroniza el envío y la recepción de los datos.
  - **Señal de reloj en los datos:** sistemas con reloj propio en el que se utiliza una codificación Manchester para detectar la velocidad de transmisión de los datos por el canal de comunicación.

## 3.2. Buses de campo:

La función principal de un bus de campo es simplificar la instalación del conexionado punto a punto de los elementos de un proceso industrial., y conseguir automatizar un proceso en tiempo real, de tal forma que los PLC que forman una red consigan intercambiar información entre sí.

### 3.2.1. Profibus DP:

Profibus DP (periferia descentralizada) es un estándar abierto de comunicaciones para buses de campo que se emplea para la interconexión de dispositivos de campo de entrada y salida:
En cuanto a su medio físico, trabaja bajo las especificaciones de la norma física RS-485. Se usa un par diferencial con cable trenzado, previsto para comunicación Half-duplex, cuya velocidad de transmisión varía entre 9.6Kbits/s y 12Mbits/s. Se colocan unas resistencias terminadoras en cada extremo de la red.

Se aplica para el intercambio de un volumen medio de datos a gran velocidad entre un controlador principal, que actúa como maestro de la red, y el resto de equipos conectados y distribuidos por la red Profibus son denominados esclavos.

Profibus DP siempre diagnostica un posible error en la red o en algún dispositivo durante la comunicación de un dato, así como también es inmune al ruido que pueda interferir en el canal de comunicación.

Profibus DP incluye dos tipos de dispositivos en la red de comunicación:

- **Activos:** dispositivos controladores de la red, los cuales inician la comunicación y preguntan por la información a los elementos pasivos. Existe un modelo de equipos activos:
    - Maestros DP clase 1 (DPM1): Autómata programable que tiene asignadas funciones de control a los esclavos de la red.

- Maestros DP clase 2 (DPM2): equipos con funciones de configuración y diagnóstico. Unidades específicas de programación.

- **Pasivos:** dispositivos esclavos que no pueden iniciar la comunicación, pero pueden responder a las peticiones de los maestros. Existe un modelo de dispositivos pasivos:
  - Esclavo DP: Dispositivo que ejecuta la lectura del estado de sus entradas, y envía información a las salidas que están conectadas en el.

Un ejemplo de un sistema maestro–esclavo con profibus DP sería el siguiente:

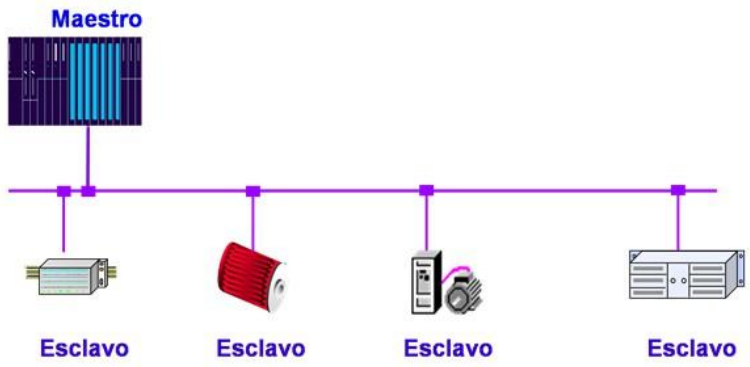



Ilustración 15: sistema maestro/esclavo en Profibus DP

**Arquitectura:**

El protocolo profibus-DP sigue la siguiente arquitectura a nivel de OSI:

- **Capa 1:** Medio físico de transmisión, que es RS-485 con fibra óptica
- **Capa 2:** protocolo de enlace al bus, Fieldbus Data Link (FDL). Establece el recorrido del testigo (token).

No se aplican las capas desde la 3 hasta la 7.

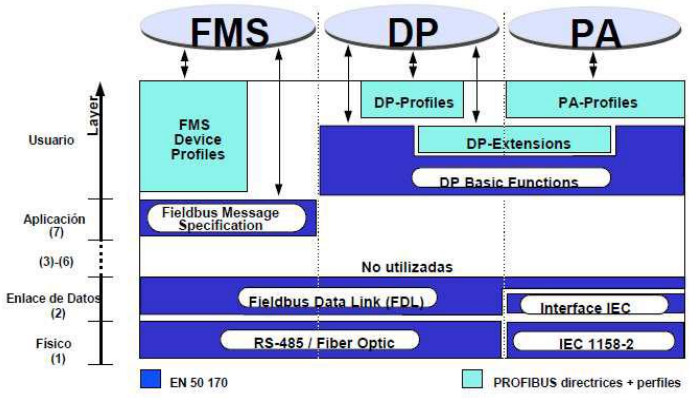



Ilustración 16: Arquitectura Profibus DP

## Topologia de red:

La topología de red es la representación física de la manera de intercambiar datos dentro de una red industrial.

En profibus DP se pueden utilizar dos tipos de topologías de red:

- **Bus lineal**: se trata de una línea de comunicación donde van conectados todos los nodos de la red. El bus admite 32 equipos, pero se puede expandir hasta 127 mediante repetidores.

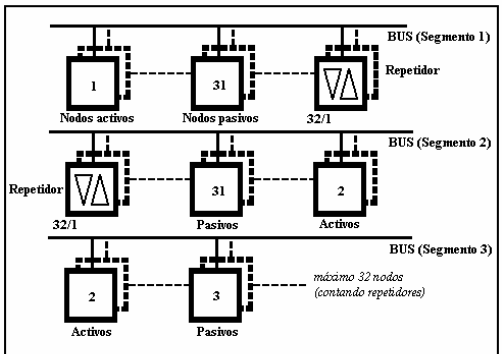



Ilustración 17: Topología bus lineal

- **Árbol**: Se puede considerar un sistema de bus con un único maestro.

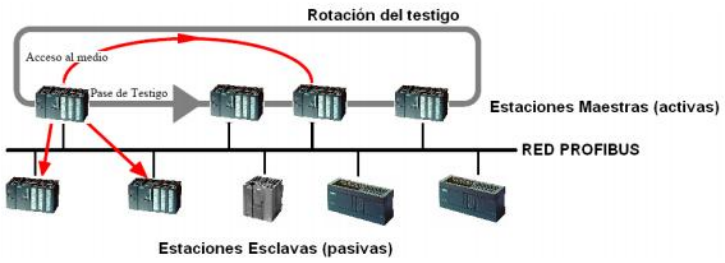
**Método acceso al bus:**

Hasta ahora se había mencionado el método de transmisión de datos mediante maestro-esclavo, que es el protocolo que sigue profibus en la comunicación entre el maestro de la red y los esclavos.

En sistemas multimaestro Profibus DP utiliza el método de acceso al bus de paso de testigo (token) para la comunicación entre los maestros de la red.

Cuando una estación posee el testigo, las demás estaciones estarán actuando como esclavos, incluyendo el resto de equipos maestros que no poseen el testigo.

A continuación se puede ver la representación del método de paso de testigo:




*Ejemplo configuración por paso de testigo entre estaciones maestras en la red*

Ilustración 18: Método paso de testigo

Tabla 2: Resumen características Profibus DP

| Característica: | Profibus DP norma EN 50 170: |
|---|---|
| Método acceso: | Paso testigo entre maestros, y maestro-esclavo |
| Velocidad transmisión: | 9,6 kbit/s hasta 12 Mbit/s |
| Volumen datos: | Hasta 246 bytes<br>Longitud trama : hasta 255 bytes |
| Nº máximo de equipos: | 32 estaciones por segmento, en total 127 usando repetidores |

| | |
|---|---|
| Tamaño de la red: | Eléctrica: 9,6 km<br>Óptica: 150 km |
| Medio transmisión: | Eléctrico: par trenzado apantallado<br>Óptico: cables fibra óptica<br>Sin cables: infrarrojos |
| Tiempo ciclo: | 5-10 ms |
| Topología: | Bus lineal, árbol. Con resistencias terminadoras |

**Profibus DP V2 (Última versión):**

La última versión de profibus DP surgió en 2002, cuyos maestros pueden realizar las siguientes funciones:

- Configuración.
- Parametrización.
- Lectura cíclica de salidas.
- Lectura datos diagnóstico.
- Lectura/escritura acíclica.
- Reconocimiento alarmas.
- Sincronización reloj entre todos los equipos.
- Comunicación directa entre los esclavos.

**Trama Profibus DP:**

La trama de datos de profibus DP se clasifica en 3 tipos:

- Longitud fija sin campo de datos.
- Longitud fija con campo de datos.
- Longitud variable con campo de datos.

Transmisión asíncrona: caracteres de 11 bits (Bit Start, 8 bits de datos, bit de paridad, bit Stop).

A continuación se puede observar un ejemplo de los tipos de tramas de Profibus:

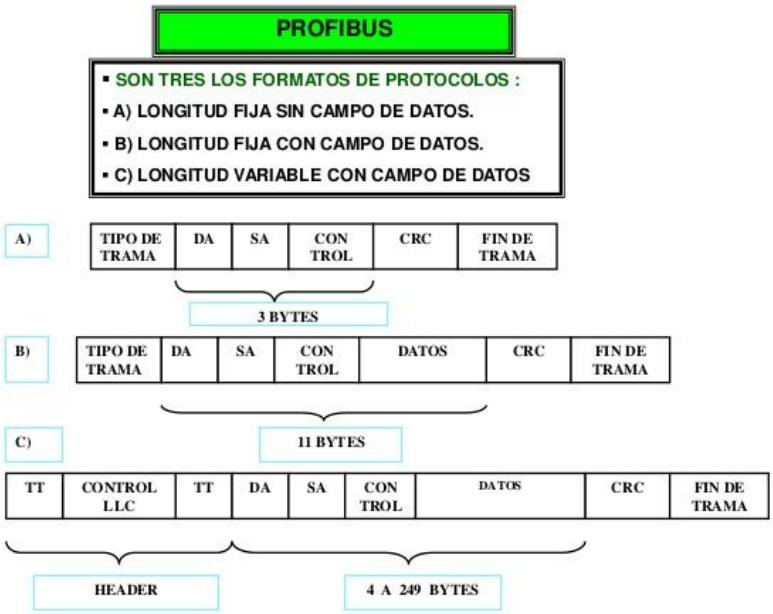



Ilustración 19: Tipos de tramas Profibus

### 3.2.2. Bus CAN:

CAN surgió en la automoción, fue desarrollada por el alemán Robert Bosch a través del protocolo Intel. Su robustez y su topología de bus mediante el método de acceso múltiple a la red lo hacían muy útil en sistemas automatizados en tiempo real.

Actualmente CAN es muy utilizado por fabricantes de controladores industriales, tanto en sistemas digitales, como en sistemas distribuidos de tiempo real.

En cuanto al medio físico, el bus CAN trabaja con una línea de bus a dos hilos trenzados, los cuales se denominan:

- CAN_high ( CAN_H): Señal de nivel alto.
- CAN_low (CAN_L): Señal de nivel bajo.

**Niveles de tensión:**

El bus de tensión de un sistema CAN tiene dos estados:

- **Dominante:** en estado dominante se produce una diferencia de tensión entre los cables de nivel alto y nivel bajo. Tensión diferencial de 2 V aproximadamente.
- **Recesivo:** Señales de nivel alto y bajo están en el mismo nivel. Tensión diferencial aproximadamente 0V.

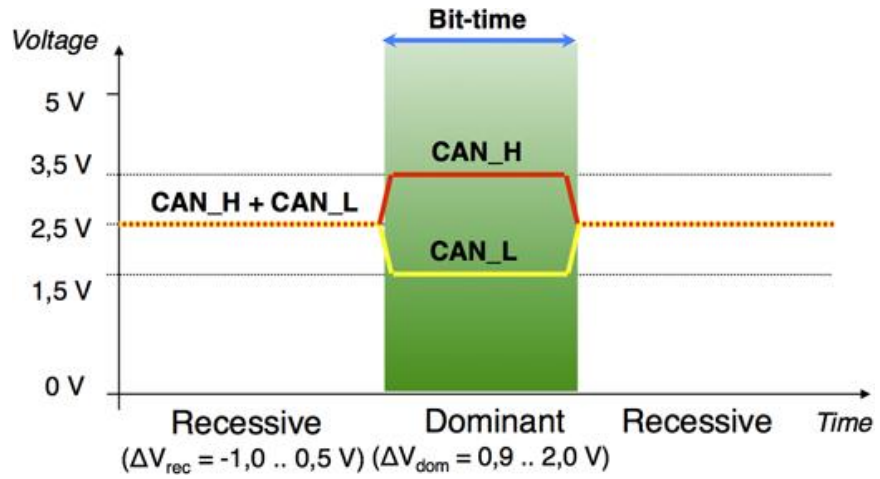



Ilustración 20: Niveles de tensión CAN

Existen 2 tipos de buses según su velocidad:

- CAN de baja velocidad (hasta 125 kbit/s).
- CAN de alta velocidad (hasta 1 Mbit/s).

La estructura del bus CAN con los nodos sería la siguiente:

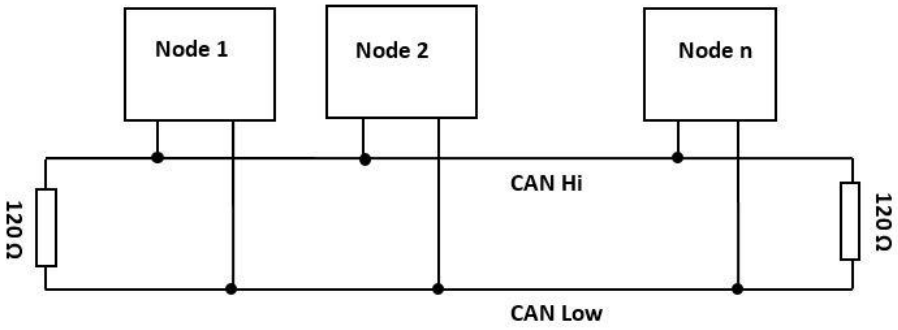



Ilustración 21: Bus CAN

El bus CAN permite la conexión de hasta 128 equipos en la red (1 maestro y 127 esclavos). La velocidad de transmisión viene determinada por una tabla según las longitudes y el tipo de cables utilizados.

Tabla 3: Velocidad- longitud bus CAN

| Velocidad de transmisión | Tiempo de bit | Longitud máxima |
|---|---|---|
| 1 Mbps | 1 µs | 30 m |
| 800 kbps | 1.25 µs | 50 m |
| 500 kbps | 2 µs | 100 m |
| 250 kbps | 4 µs | 250 m |
| 125 kbps | 8 µs | 500 m |

**Tramas Bus CAN:**

En el bus CAN existen cuatro tipos de tramas de información:

- **Trama de datos:** estas tramas son las que incluyen la información que circula por el canal de comunicación.
  Existen dos formatos de la trama de datos:
  - Formato base: trama de datos más corta, que tiene los siguientes campos:

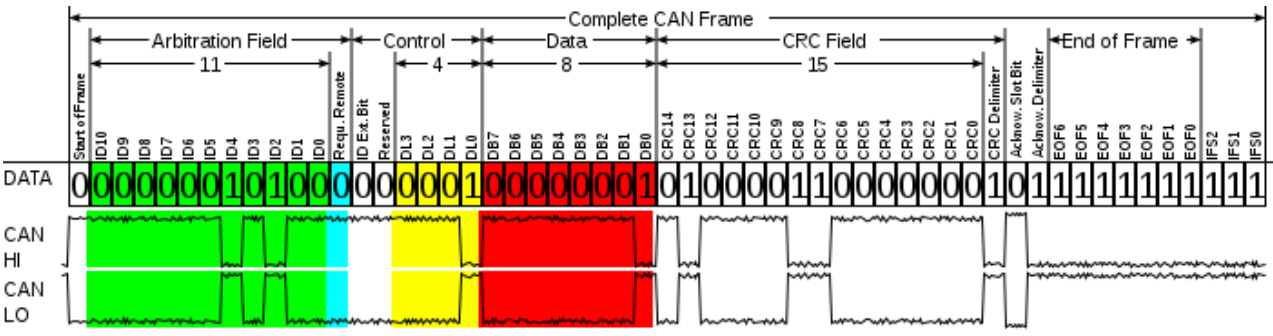



Ilustración 22: trama de datos CAN formato base

- SOF (start of frame): 1 bit que marca el incio de la trama.
- Arbitration field: Campo que define las prioridades la trama. Está formado por los siguientes bits:
  - Identificador: 11 bits de identificación. También representan la prioridad de la trama.
  - RTR: 1 bit de petición de transmisión remota. El bit está en estado 0 (dominante) para la trama de datos, y a 1 (recesivo) para tramas remotas.
- Control field: Campo de control de la trama. Contiene los siguientes bits:
  - IDE: bit de extensión de identificador. En formato base el bit debe estar a 0 (dominante).
  - R0: bit reservado.
  - DLC: 4 bits que indican el numero de bytes de datos a enviar. Valor entre 0 y 8.
- Data: Datos de la trama, que están entre 0 y 64 bits, cuya longitud de los datos a enviar viene dada por el código DLC o el de longitud de datos.
- CRC field:
  - 15 bits de verificación por redundancia cíclica. Código que indica la correcta transmisión de los datos.
  - Delimitador CRC: bit siempre en estado 1 (recesivo).
- ACK field: Campo de reconocimiento de datos obtenidos.
  - ACK: bit de acuse de recibo del mensaje. Debe estar en estado 1 (recesivo) para el emisor, y en estado 0 (dominante) para el receptor.
  - Delimitador ACK: debe estar en estado 1 (recesivo).
- EOF (end of frame): 7 bits que indican el fin de la trama. Deben estar todos los bits en estado 1 (recesivo).

- Formato extendido: trama más extensa. La mayoría de campos coinciden con el formato base, aunque tienen algunas diferencias:

- dos identificadores: uno de 11 bits, y otro segundo identificador de 18 bits, lo que hace un total de 29 bits para la identificación.
- SRR: Dispone de un bit de sustitución de transmisión remota, que debe estar en estado 1 (recesivo).

- **Trama remota:**
  El formato de la trama es el mismo que en la trama de datos, pero con el bit RTR en estado 1 (recesivo). Además la trama remota no requiere de bits de datos, va implícito dentro del código de longitud de datos (DLC).

- **Trama de error:**
  Son tramas generadas cuando un nodo detecta un mensaje erróneo. Contiene los siguientes campos:

  - **Error flag:** son los bits indicadores de error, que se clasifican en 2 tipos:
    - **Activos:** 6 bits en estado 0 (dominante). Detectan error en el fin de la trama (EOF).
    - **Pasivos:** 6 bits en estado 1 (recesivo). Detectan error en el campo de reconocimiento (ACK).

- **Trama de sobrecarga:**
  Similar a una trama de error activo, pero la trama de sobrecarga siempre se debe generar en el espacio entre una trama y otra, ya que ofrece un retardo a las tramas.

- **Separación entre tramas:** se basa en separar una trama de datos de la siguiente trama remota, que están separadas por 3 bits recesivos. Las tramas de error y sobrecarga no cumplen la separación entre tramas

**CANOpen:**

Protocolo utilizado en el bus CAN. En CANopen se utiliza el método de acceso al medio CSMA/CD, que es de acceso múltiple. Consiste en una lectura bit a bit del mensaje, y si se detectan diferencias entre transmisor y receptor, pasa arbitrariamente a otro receptor.

Modelo Comunicaciones CANopen:

- **Maestro/esclavo:** Un único maestro. Resto de equipos de la red son esclavos.
- **Cliente/servidor:** solo 2 equipos, un cliente hace una petición y un servidor ejecuta una tarea.
- **Productor/consumidor:** Dispone de un productor y uno o varios consumidores.

## 3.3. Modelo OSI:

Teniendo en cuenta las dificultades que entrañaban las comunicaciones entre redes de distintas características, no se conseguía transmitir la información correctamente. Para ello se estableció un modelo de una serie de capas, en las cuales se emplea un protocolo para cada una de ellas. De esta forma se pueden estandarizan las comunicaciones entre distintos sistemas industriales.

El modelo OSI es un estándar de comunicaciones para la interconexión de sistemas abiertos, que fue creado en 1983 por ISO, el cual consta de 7 capas:

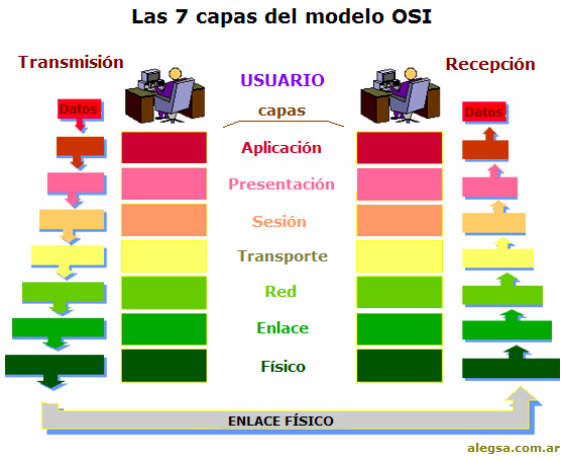



Ilustración 23: Modelo OSI

- **Capa 1 – Física:** El nivel más bajo del modelo OSI. Se encarga de la topología de la red que se va a utilizar, y las conexiones desde el PC a la red:
    - Garantizar la conexión, sin garantizar el rendimiento.
    - Medio físico de la conexión (cableado, normativa).
    - Características materiales, y funcionales de la interface.
    - Transmisión de flujo de bits.
    - Señales eléctricas entre las conexiones.

- **Capa 2 - Enlace:** Fiabilidad en la transmisión de datos, direccionamiento, control de errores, y control de flujo de datos.

- **Capa 3 - Red:** Capa que se encarga del enrutamiento de la información entre redes.
- **Capa 4 - Transporte:** Transporte de los datos hasta el equipo de destino.
- **Capa 5 - Sesión:** Se encarga de controlar el enlace entre dos equipos que están en sesión de ejecución de una transmisión de datos.
- **Capa 6 - Presentación:** Tiene como objetivo hacer que la representación de los datos sea legible y entendible por cualquier sistema, independientemente de la red física utilizada.
  Se podría decir que esta capa es el traductor de la información para que la máquina de destino pueda reconocer los datos.
  La capa de presentación cumple 3 requisitos de tratamiento de los datos obtenidos:
    - Formatear la información
    - Cifrado de datos
    - Comprimir los datos.

- **Capa 7 - Aplicación:** Se encarga de definir los protocolos de intercambio de datos.

## 3.4. Ethernet:

Ethernet es una estándar de redes de área local, que utiliza el método de acceso al medio de acceso múltiple (CSMA/CD).

Ethernet define las características del medio físico y las comunicaciones al nivel de enlace, así como los formatos de las tramas de datos del sistema, siguiendo el modelo OSI.

**Trama Ethernet:**

El formato de la trama Ethernet es el siguiente:

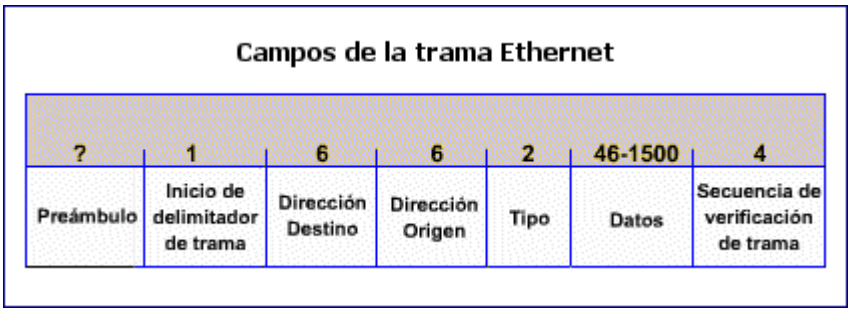



Ilustración 24: formato trama Ethernet

- **Preámbulo:** conjunto de bits identificadores de la red Ethernet.
- **Inicio trama (SOF):** 8 bits delimitadores de inicio de trama, que terminan en dos bits en estado 1. Sincronización de la recepción de la trama.
- **Direcciones destino y origen:** Direcciones físicas de las máquinas de envío y destino.
- **Tipo:** especifica protocolo de la capa superior.
- **Datos:** datos de la trama. Ethernet mínimo 46 bytes.
- **Secuencia de verificación (FCS):** contiene un código de redundancia cíclia (CRC) de 4 bytes, generado por el emisor para el control de errores en la trama. El Checksum (verificación) de la cabecera IP lo identifica como mensaje IP. Tamaño máximo paquetes en Ethernet de 1500 bytes.

### 3.4.1. Modelo OSI en Ethernet:

El modelo OSI para las redes Ethernet tiene la estructura similar al modelo OSI original:

#### 3.4.1.1. Medio físico:

Ethernet utiliza como medio físico un cable denominado RJ45, que dispone de 8 pines.

##### 3.4.1.1.1. Cableado RJ45:

RJ-45 es una interfaz física que se utiliza usualmente para cables de red Ethernet. Hay dos tipos de cables de red: directos y cruzados. En este caso los cables utilizados son directos.

El cable de red Ethernet tiene 8 pines, que son terminaciones para 4 pares trenzados:

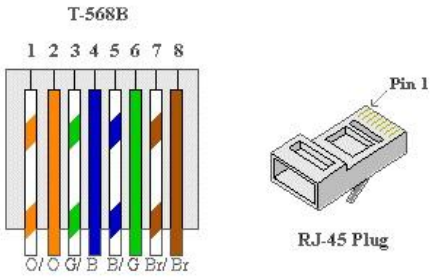



Ilustración 25: Cable RJ-45 T568B

Los cables de red directos de este sistema cumplen la normativa EIA/TIA-568B, que tiene el siguiente código de colores:

Tabla 4: código colores cable RJ-45

| Pin | Color |
|---|---|
| 1 | Blanco y naranja |
| 2 | Naranja |
| 3 | Blanco y verde |
| 4 | Azul |
| 5 | Blanco y azul |
| 6 | Verde |
| 7 | Blanco y marrón |
| 8 | Marrón |

### 3.4.1.2. Protocolo IP:

El protocolo de internet (IP) está clasificado en la capa de red del modelo OSI. IP es el encargado de la distribución de paquetes nombrados datagramas en campos de 4 bytes.

Para poder realizar el envío almacena la información del paquete en direcciones, fragmentando los datos si es necesario. El protocolo IP gestiona la IP del origen y de destino de los datagramas. Para la gestión de los datagramas por la red son necesarios los routers.

El protocolo IP también puede recibir datos de la capa de transporte, que tiene asociado el protocolo TCP.

**Direcciones IP:**

Conjunto de números que identifican una interfaz de red de un dispositivo que utiliza el protocolo IP.

Las direcciones IP se representan con el formato XXX.XXX.XXX.XXX, donde cada conjunto de números separados por un punto corresponde a 8 bits binarios que en decimal es un digito que va de 0 a 255. Los 4 conjuntos de 8 bits se denominan octetos, con lo cual una dirección IP contiene 32 bits.

IPv4 es la versión del protocolo IP que se utiliza en este estudio, la cual tiene la cabecera:

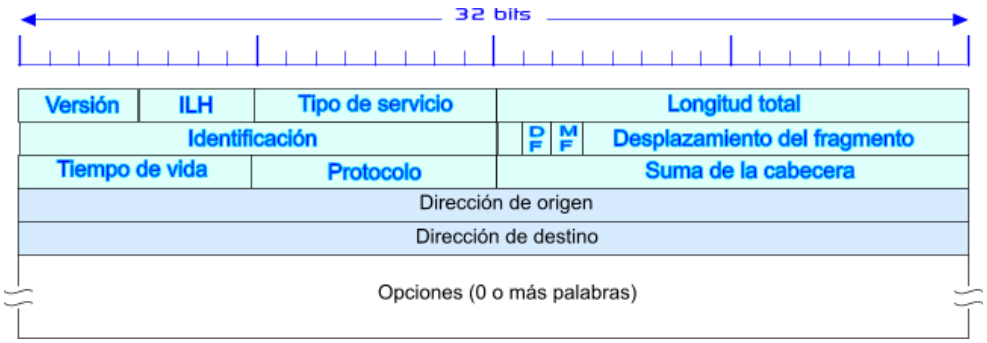



Ilustración 26: cabecera IPv4

### 3.4.1.3. Protocolo TCP:

Protocolo de control de la transmisión, que opera en la capa de transporte del modelo OSI. Tiene la función de que la transmisión se realice sin errores y en el orden correcto de transmisión. El protocolo IP envía a TCP los datos en segmentos.

**Características TCP:**

- Permite ordenar segmentos de datos que recibe del protocolo IP.
- Monitoreo de datos.
- Longitud variable de los segmentos de datos.
- En caso de segmentos de gran tamaño, fracciona dichos segmentos formando datagramas

### 3.4.1.4. Modbus:

Modbus es un protocolo de comunicaciones que trabaja en los niveles 1 (Físico), 2 (enlace) y 7(aplicación) del modelo OSI. Fue diseñado en 1979 para los PLC de Modicon, para uso en aplicaciones industriales.

Modbus se utiliza para la comunicación interna entre dispositivos electrónicos, que pueden ser PLC u otro tipo de controladores industriales.

Modbus posee la gran ventaja de que puede manejar grandes bloques de datos sin restricciones. El protocolo modbus es capaz de control grandes redes de equipos y crear bases de datos en un PC. Trabaja con la topología maestro/esclavo.

En modbus cada equipo tiene una dirección y puede enviar datos, aunque usualmente hay un equipo maestro. Al enviar una trama todos los equipos la reciben, pero solo la ejecuta el nodo receptor. Hay una excepción que es el broadcast, que hace que se puedan enviar datos simultáneamente a varios receptores.

En cuanto al medio físico, modbus en este caso funciona acorde a las especificaciones de la norma física RS-485, anteriormente mencionada en la comunicación con el bus de campo profibus DP.

**Modbus TCP/IP:**

Para la comunicación entre redes Ethernet se utiliza el protocolo modbus encapsulado sobre TCP, formando Modbus TCP/IP, que es el protocolo utilizado para enviar datos entre los PLC de la célula.

**Trama modus TCP/IP:**

La trama de modbus sobre redes Ethernet mediante protocolo TCP/IP sería de la siguiente manera:

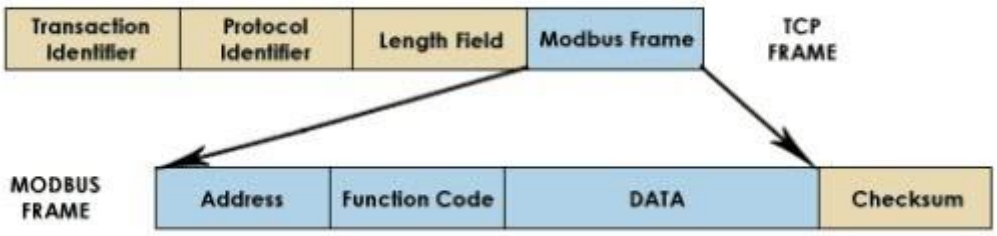



Ilustración 27: Trama Modbus sobre TCP/IP

- **Identificador de transacción:** longitud 2 bytes. Sincronización entre cliente y servidor.
- **Identificación protocolo:** longitud de 2 bytes. Valor 0 indica protocolo Modbus TCP/IP.
- **Longitud de campo:** Longitud 2 bytes. Indica número de bytes en la trama actual.
- **Trama Modbus:**
  - **Dirección:** Byte de dirección del esclavo.
  - **Código de función:** 1 byte de códigos de función.
  - **Bytes de datos:** número determinado de bytes de datos de respuesta.
  - **Checksum (verificación):** En este caso no es necesario ya que las capas inferiores ya disponen de protección de suma de verificación checksum.

# 4. Descripción del sistema:

El sistema está formado por dos grandes bloques principales: la célula flexible y el pulmón.

## 4.1. Célula flexible:

La célula está compuesta de estaciones de trabajo, y está controlada por PLC'S, los cuales trabajan con distintos buses de comunicación a nivel de campo. En este caso se trabaja con Profibus DP, CAN, y el estándar de redes Ethernet para el ciclo del pulmón. Hay una pequeña estación que trabaja con el bus de sensores y actuadores A-si, pero para este proyecto no se va a utilizar dicha línea.

La célula está planteada para uso académico, no obstante está aplicada a un posible caso real en la industria, ya que pretende simular el funcionamiento de una línea de producción de piezas, que son transportadas por palets por las diferentes estaciones de trabajo.

Se dispone de distintas cintas transportadoras principales, que delimitan las líneas de trabajo. Una cinta es controlada por profibus DP, y dispone de 2 islas de entradas y salidas, donde van conectados los elementos de control y mando de la línea. Hay zonas de las cintas, que son puntos de comunicación de dos líneas, donde se traslada el estado de las señales que van a intervenir en el siguiente proceso.

Ilustración 28: Célula flexible

- **Ciclo del recorrido de los palets:**

Los palets se pueden introducir manualmente desde la línea profibus, o bien pueden ser descargados por el pulmón. Cuando son introducidos en la línea profibus se colocan en la plataforma PT08, y circulan por la célula según las ordenes de programa.

Posteriormente los palets van llegando a la línea CAN, en la plataforma PT15, para luego poder ser almacenados en el pulmón según las necesidades de la línea.

### 4.1.1. Islas Advantys:

Las islas Advantys son periferias de entrada/salida, que están conectadas a los PLC principales mediante Ethernet. Son islas Schneider del tipo STB. A continuación de un ejemplo de una periferia:

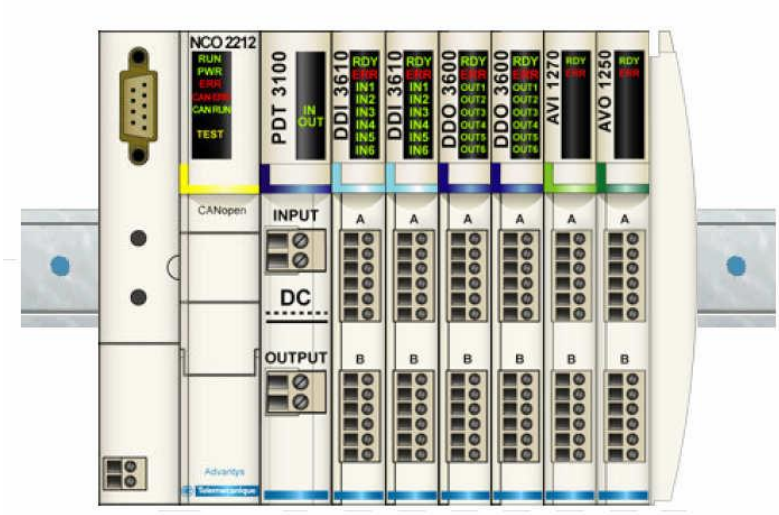



Ilustración 29: Islas Advantys

### 4.1.2. Líneas:

- **Profibus DP:** Comprende las estaciones 5,6 y 7, con sus respectivas islas Advantys.

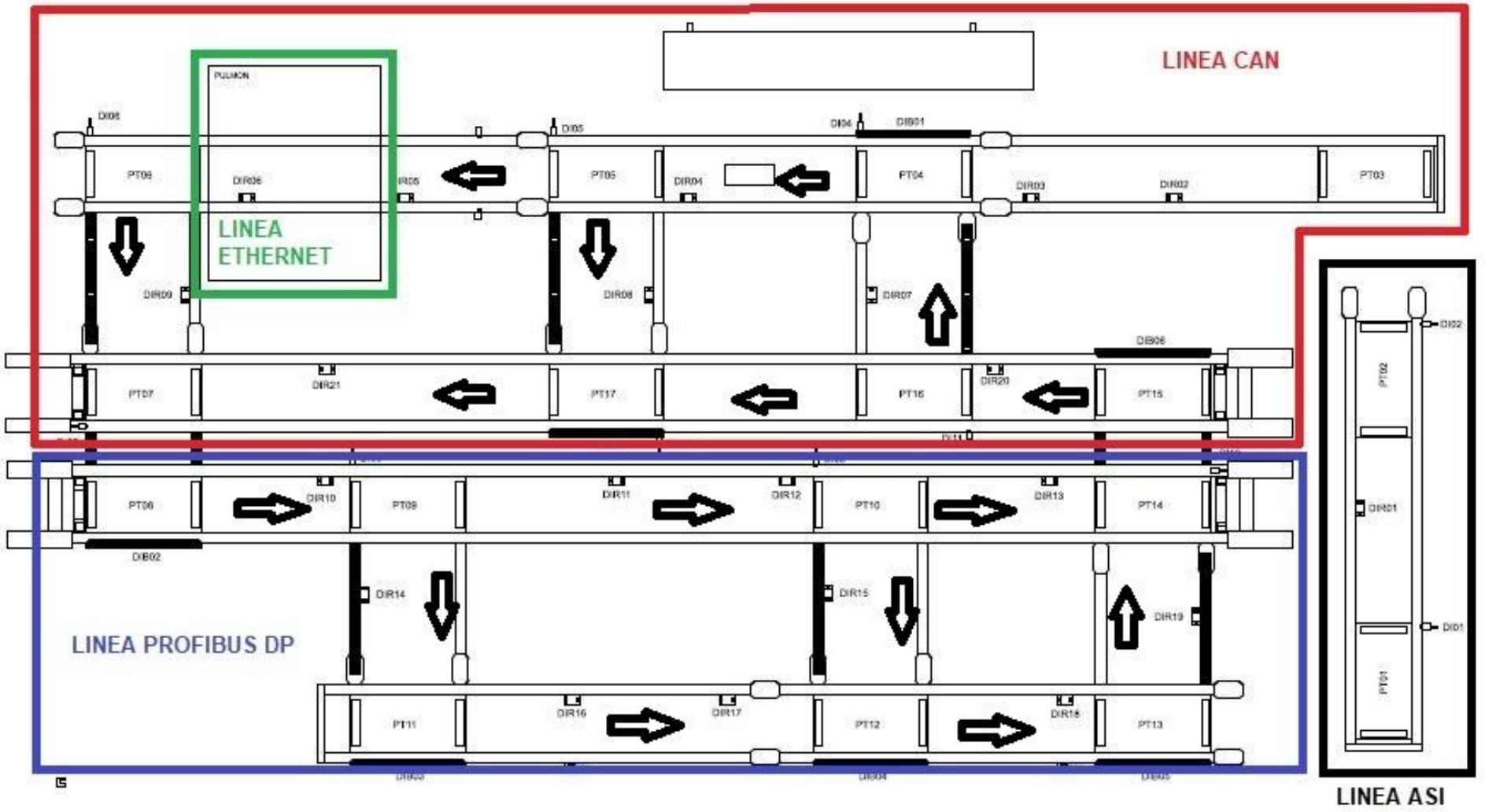
**Puntos de comunicación con línea CAN:**

La línea profibus comunica con la línea CAN en la plataforma neumática PT07, donde mediante una orden de programa a la línea CAN le envía el estado de la plataforma siguiente (PT08), para que si no hay ningún palet presente en dicha plataforma, pueda ser transferido a la PT08, y el palet entra en la línea profibus DP.

Hay otro punto de comunicación entre profibus y CAN en la PT14, donde la profibus envía una orden de programa a la línea CAN, para saber si la siguiente plataforma de la línea CAN (PT15), está disponible para que el palet pueda ser traspasado de la PT14 a la plataforma PT15. Una vez el palet es transferido a la PT15, entra en la línea CAN.

- **CAN:** Comprende las Estaciones 1, 2, 3 y 4. Además de la comunicación explicada en el apartado de profibus DP, la línea CAN también comunica con la línea A-si, y con el pulmón.

- **Ethernet:** El pulmón trabaja en dicha línea y tiene la función de cargar las piezas que le llegan desde la línea CAN, y también descargarlas sobre la propia línea CAN.




Las flechas indican la dirección de la bandeja por la célula.

Ilustración 30: Plano líneas Célula

### 4.1.3. Componentes:

La célula dispone de componentes de detección, accionamiento, elementos neumáticos, etc. Todos los elementos hacen posible el funcionamiento de la célula, mediante las órdenes de programa que se ejecuten para dichos componentes.

#### 4.1.3.1. Sensores:

Elementos captadores de las señales físicas del proceso industrial, las cuales son procesadas por los módulos de entradas del PLC, para que el programa almacene su estado, con el objetivo de transmitir dicho estado a los actuadores siempre siguiendo las órdenes del programa de usuario.

##### 4.1.3.1.1. Sensores Fotoeléctricos:

Sensores de detección de objetos mediante haces de luz. Existen dos tipos principales de sensores fotoeléctricos:

- **Barrera emisor-receptor:** se basa en dos componentes, los cuales uno emite un haz de luz y el receptor percibe la luz emitida. El emisor y el receptor están separados una cierta distancia, que es lugar por donde pasa el objeto. Cuando el objete se sitúe en medio del emisor y receptor, la luz reflejada en el receptor debe ser mucho menor que en el emisor, ya que el objeto hace de pantalla. En ese momento se produce la detección.

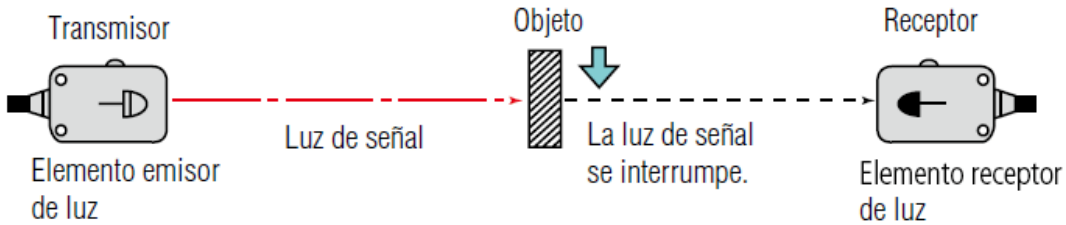



Ilustración 31: sensor fotoeléctrico barrera emisor-receptor

- **Reflexión espejo:** El emisor y el receptor son el mismo componente. El mecanismo de este sensor se basa en que cuando pasa un objeto, hace reflejar la luz de vuelta al emisor, y entra en estado de detección. Mientras no se detecte ningún objeto, la luz no se refleja al emisor.

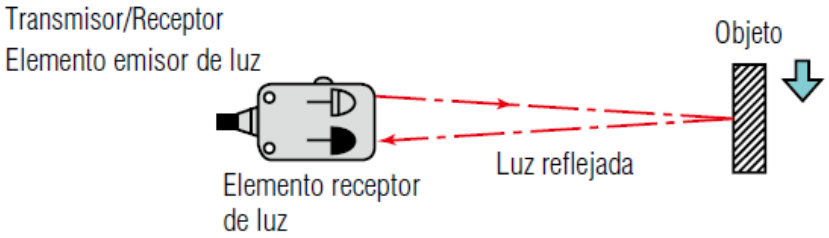



Ilustración 32: sensor fotoeléctrico reflexión por espejo

#### 4.1.3.1.2. Sensores inductivos:

Los sensores inductivos tienen la función de detectar objetos metálicos cuando pasan por encima de dicho sensor, ya que el sensor tiene una bobina interna, que cuando circula corriente por ella se genera un campo magnético. Cuando un metal se coloca en la zona de detección del sensor, a causa de las corrientes de Focault, se genera un campo magnético en la dirección opuesta al que genera la bobina. Entonces se genera una variación de la amplitud de onda del sensor, entrando en etapa de detección.

Cuando la señal del sensor está conectada a un PLC Se ha de tener en cuenta el tipo de sensor:

- **Normalmente abierto (NO):** su estado de reposo es un 0 lógico, y cuando entra en estado de detección pasa a estado 1 lógico.

- **Normalmente cerrado (NC):** es la inversa del NO, su estado reposo es 1, y al entrar en detección pasa a 0.

En la célula se utilizan sensores inductivos, para detectar el paso de una bandeja metálica en un retenedor. Hay una variante que es el detector inductivo basculante, que se utiliza para detectar el palet en una plataforma neumática. Está insertado en un mecanismo basculante que al llegar una bandeja es desplazado y hace que cambie el estado de la señal ya sea un sensor normalmente abierto o cerrado.

Ilustración 33: Detector inductivo basculante

Los sensores de la célula disponen de un retenedor, que es un actuador neumático, concretamente un cilindro de simple efecto activado mediante electroválvula, y con retorno por muelle. Cuando pasa una bandeja por el sensor, según las ordenes de programa el retenedor baja para dejar que la bandeja pase a la siguiente estación.

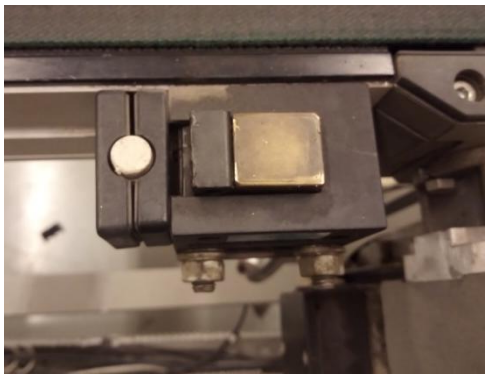


Ilustración 34: sensor inductivo con retenedor

### 4.1.3.2. Pre actuadores:

Son los elementos que mediante una señal eléctrica permite la activación de los actuadores.

- **Electroválvulas:** válvulas electromecánicas que permiten controlar el paso de un fluido. La activación de un relé solenoide produce un cambio del estado de la válvula, la cual puede estar abierta, que en tal caso permite el paso del fluido, o cerrada, que no deja pasar el fluido.

### 4.1.3.3. Actuadores neumáticos:

- **Retenedores:** Como se ha mencionado en el apartado anterior, son cilindros neumáticos de simple efecto con retorno por muelle.

- **Plataformas**: plataformas neumáticas que se encuentran en las cintas transportadoras.

  Pueden ser de simple efecto, las cuales tienen 2 posiciones: posición de reposo y subida, o de doble efecto con 3 posiciones: subida, reposo y bajada.
  Cuando una bandeja circula por la cinta y se encuentra una plataforma, según las ordenes de programa dejará pasar la bandeja bajando la plataforma.

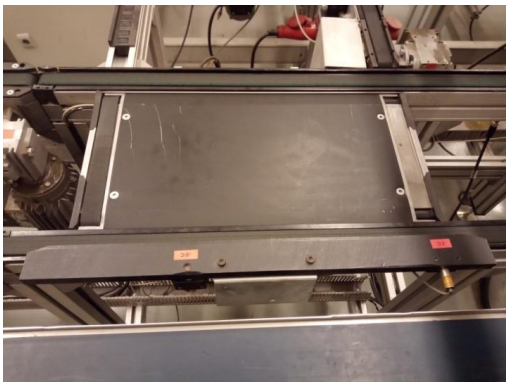


Ilustración 35: Plataforma neumática

Las plataformas neumáticas tienen un sensor inductivo para detectar la llegada de la bandeja a dicha plataforma.

### 4.1.3.4. Motores:

Los motores eléctricos trifásicos se utilizan para la activación de las cintas transportadoras, que hacen posible el desplazamiento de las bandejas metálicas por la célula. Las plataformas neumáticas también disponen de cintas transportadores con sus respectivos motores.

Ilustración 36: Motores eléctricos cintas transportadoras

### 4.1.3.5. Elementos de mando y señalización:

La célula dispone de una botonera, donde se hallan 3 pulsadores para activar los motores de las tres cintas transportadoras principales con sus respectivos pulsadores de paro, y un pulsador de parada de emergencia.

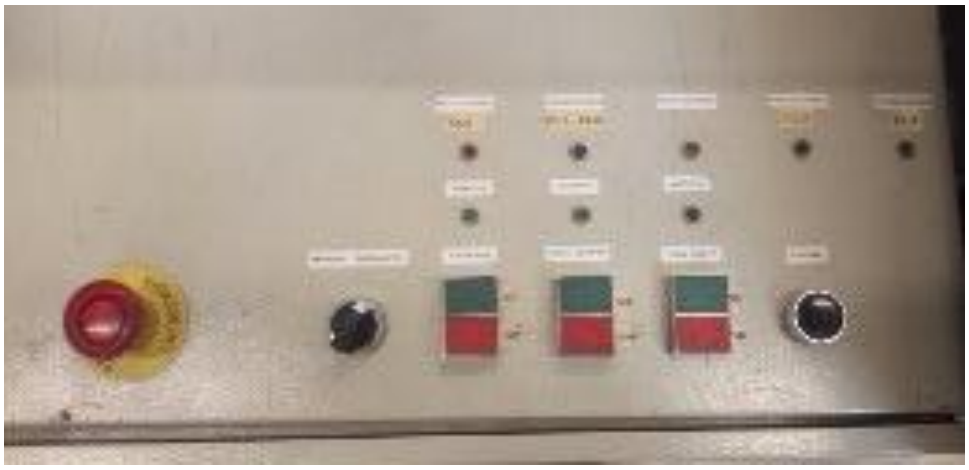


Ilustración 37: Cuadro de mando cintas transportadoras

## 4.2. Pulmón:

El pulmón consiste en un almacenador de palets, que mediante 4 cadenas de retención montadas sobre dos ejes, y accionadas por un motor eléctrico, puede cargar o descargar palets según las necesidades del proceso productivo en ejecución.

Ilustración 38: Pulmón de almacenamiento

### 4.2.1. Componentes:

El pulmón dispone de una serie de componentes para su funcionamiento, igual que la célula.

#### 4.2.1.1. Sensores:

Elementos captadores de la señal, como anteriormente se mencionó para la célula flexible:

- **Sensores Fotoeléctricos:** Hay distintos tipos de sensores fotoeléctricos instalados en el pulmón.

    - **Barrera emisor-receptor:** se utilizan para detectar las aletas de la cadena de retención. Hay tres sensores de este tipo en el pulmón:

- **Detector cadena retención lenta en carga:** cuando está cargando una bandeja, y la aleta de la cadena es detectada, disminuye su velocidad. En la ilustración está situado delante de los otros sensores.
- **Detector parada cadena de retención:** Cuando la aleta es detectada en este sensor, se detiene la cadena de retención. En la ilustración es el sensor que está situado en el centro.
- **Detector cadena retención lenta en descarga:** cuando está descargando una bandeja, y la aleta de la cadena es detectada, disminuye su velocidad. En la ilustración es el que está situado detrás de los otros sensores.

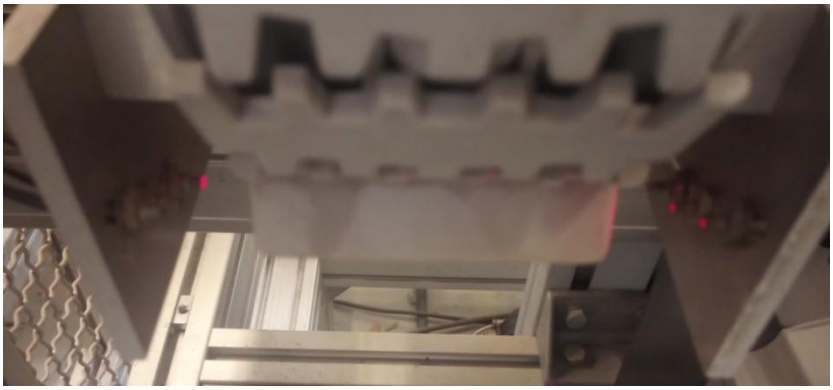


Ilustración 39: Sensores Fotoeléctricos de aleta

- **Reflexión por espejo:** hay 2 sensores de este tipo: En la parte superior e inferior: indican si el pulmón está lleno o vacío.

- **Sensores inductivos:** Los sensores con retenedor mencionados anteriormente en la célula flexible.

#### 4.2.1.2. Actuadores:

- **Actuador lineal:** Es un cilindro eléctrico activado mediante un servomotor. Tiene la función de levantar la bandeja para que la cadena de retención la pueda cargar, o también para poder descargar la bandeja.

Ilustración 40: Actuador lineal

- **Cadena de retención:** tal y como se ha mencionado en anteriores apartados, se trata de cuatro cadenas montadas sobre dos ejes motores, que se activan mediante un motor eléctrico.
  La cadena de retención dispone de aletas con una distancia de separación simétrica. En dichas aletas es donde se alojan las bandejas que se pueden cargar o descargar.

Ilustración 41: Cadena de retención

### 4.2.1.3. Motores:

- **Motor cadena de retención:** La cadena de retención es activada por un motor eléctrico que dispone de un freno de seguridad. Para mover la cadena de retención hay que desenclavar el freno.

Ilustración 42: Motor cadena de retención

- **Motor actuador líneal:** El actuador lineal se activa mediante un servomotor, que funciona mediante dos encoders.

Ilustración 43: Motor actuador lineal

#### 4.2.1.4. Variador de frecuencia:

El variador de frecuencia es elemento que gestiona las velocidades de la cadena de retención con las frecuencias de 20 Hz para velocidad rápida, y 5 Hz para velocidades lentas, aunque los parámetros pueden ser modificados manualmente.

El variador de frecuencia tiene conectados las señales de activación de los motores, y a s vez las señales que recibe del PLC.


Ilustración 44: Variador de frecuencia

### 4.2.1.5. Elementos de mando y señalización:

- **Cuadro de mando:** El pulmón tiene un cuadro de mando, el cual se puede utilizar para ejecutar el ciclo manual de programa, ya que dispone de los siguientes elementos de mando:
  - **Pulsadores:**
    - Condiciones iniciales
    - Subir / bajar actuador lineal
    - Subir / bajar cadena de retención
    - Selector ciclo manual/automático
    - Seta de emergencia
    - Rearme
  - **Pilotos de señalización:**
    - Condiciones iniciales
    - Puertas abiertas
    - Cadena retención/actuador lineal en marcha
    - Bloqueo variador de frecuencia
    - Fallo protecciones de los motores
    - Pulmón lleno/vacío

Ilustración 45: Cuadro de mando

# 5. Diseño del automatismo programable:

En esta sección se incluyen los diseños de los automatismos que se consideran para programar sobre el sistema disponible..

El diseño del automatismo se debe hacer como paso previo a la programación de cada una de las aplicaciones que se consideren, ya que para elaborar el programa de automatización se ha de trabajar sobre el diseño realizado.

Para ello se elaboran diagramas de flujo, que consisten en dividir la aplicación considerada en un conjunto de acciones, las cuales se organizan en etapas, y las cuales tienen una condición para pasar de una a otra, que se denomina transición.

Cada etapa desarrolla una o varias acciones en el proceso, por ejemplo activar una salida digital del PLC (i.e. para bajar un cilindro neumático, activar un relé, o activar un piloto de señalización).

**Funcionamiento de la célula flexible:**

En la célula los elementos principales son los conjuntos de sensores inductivos con retenedor, y las plataformas neumáticas. Para poder gestionar los movimientos de las bandejas por la línea, se debe elaborar un proceso que elabore las transiciones que deben cumplir los retenedores y las plataformas, para que la bandeja pueda circular o bloquearse según las condiciones.

## 5.1. Diagrama sensor/retenedor:

EL método de programación propuesto para el conjunto de sensores y los retenedores se basa en tres estados del sensor/retenedor:

- **Reposo (REST):** estado en el que no hay detección en el sensor, que es en el estado que se encuentra al inicio del proceso.
- **Petición (READY):** es el estado en el que se encuentra en el momento que se produce una detección de bandeja en el sensor. Es un estado para consultar la existencia de otra bandeja en la siguiente estación, ya que no puede ponerse en movimiento hasta que no esté libre la siguiente posición.

- **Avance (MOVE):** Periodo donde la bandeja está en movimiento, hasta llegar a la siguiente posición.

A continuación se puede observar el diagrama de flujo del sistema sensor/retenedor:

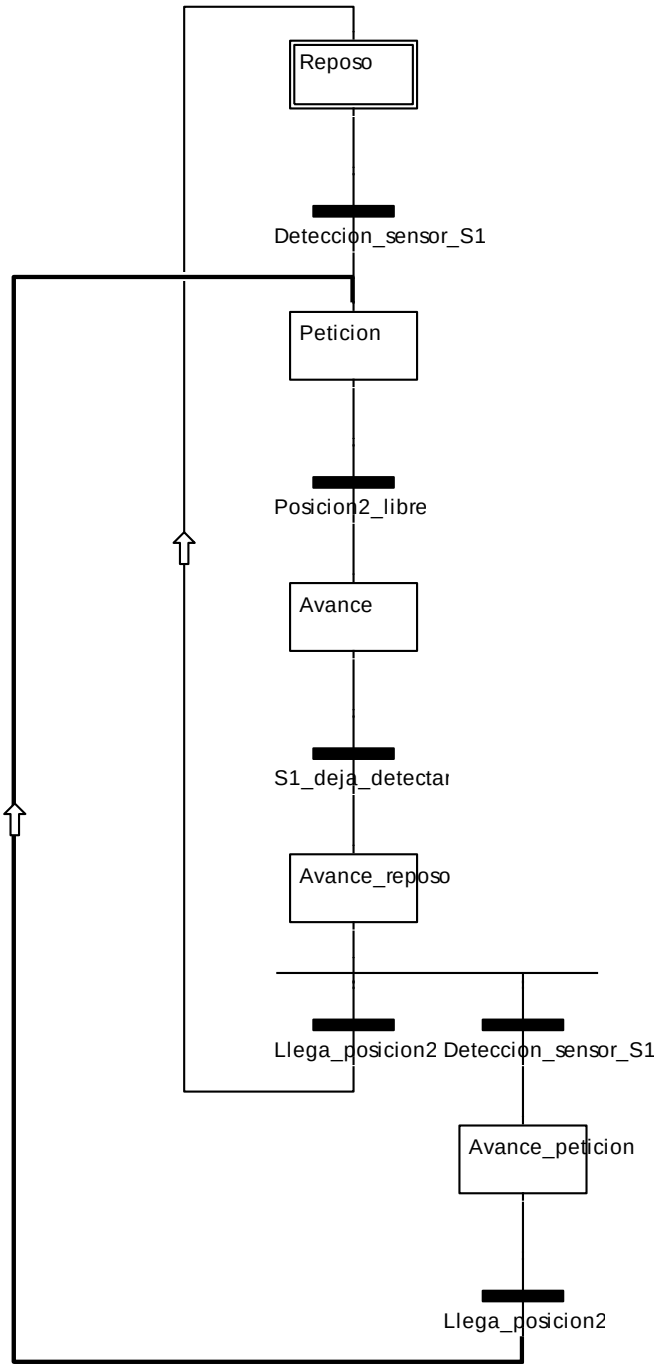



Ilustración 46: Diagrama de flujo retenedor

**Coincidencia de dos estados:** Como se puede observar en el diagrama de flujo, hay ciertos momentos donde en el sistema el sensor/retenedor está en dos estados a la vez, ya que en el proceso puede haber multitud de bandejas circulando por la línea.

- **Avance-reposo:** Cuando el retenedor está en avance, solo se activa la salida del retenedor durante un pequeño tiempo hasta que el sensor deje de detectar, ya que la bandeja dispone de una ranura en el extremo que hace que el retenedor bloquee la bandeja. Una vez pasa esa ranura, el retenedor puede volver a subir, y no bloqueará la bandeja ni el sensor la detectará. Esto se hace por seguridad, ya que en caso contrario podría haber colisión con otra pieza.
  Cuando se deja de detectar bandeja, el retenedor pasa a estado reposo pero se mantiene el avance hasta que llegue a la siguiente posición.

- **Avance-petición:** Se produce cuando la bandeja esta en movimiento hacia la siguiente posición y se produce la detección de la bandeja de la posición anterior, entonces el retenedor pasa a estado petición, pero no desactiva el avance, ya que no ha llegado a la siguiente posición. Cuando llega a la siguiente posición se desactiva el avance, y permanece en petición hasta que la siguiente posición esté disponible.

## 5.2. Diagrama plataforma:

Es el mismo método utilizado en los retenedores, ya que tiene los mismos 3 estados: reposo, petición y avance. También se produce la coincidencia de dos estados de avance-reposo, y avance-petición.

La diferencia es que el detector inductivo es basculante, y que hay plataformas que tienen 3 estados.

## 5.3. Diagrama línea profibus:

La línea profibus está delimitada desde la plataforma PT08 hasta la PT14 (ver ilustración 9). Entonces se trata de hacer un diagrama de flujo principal basado en una sucesión de etapas, y dentro de cada etapa se encontrará su diagrama de flujo, que puede ser un retenedor (DIR…) o una plataforma (PT…).

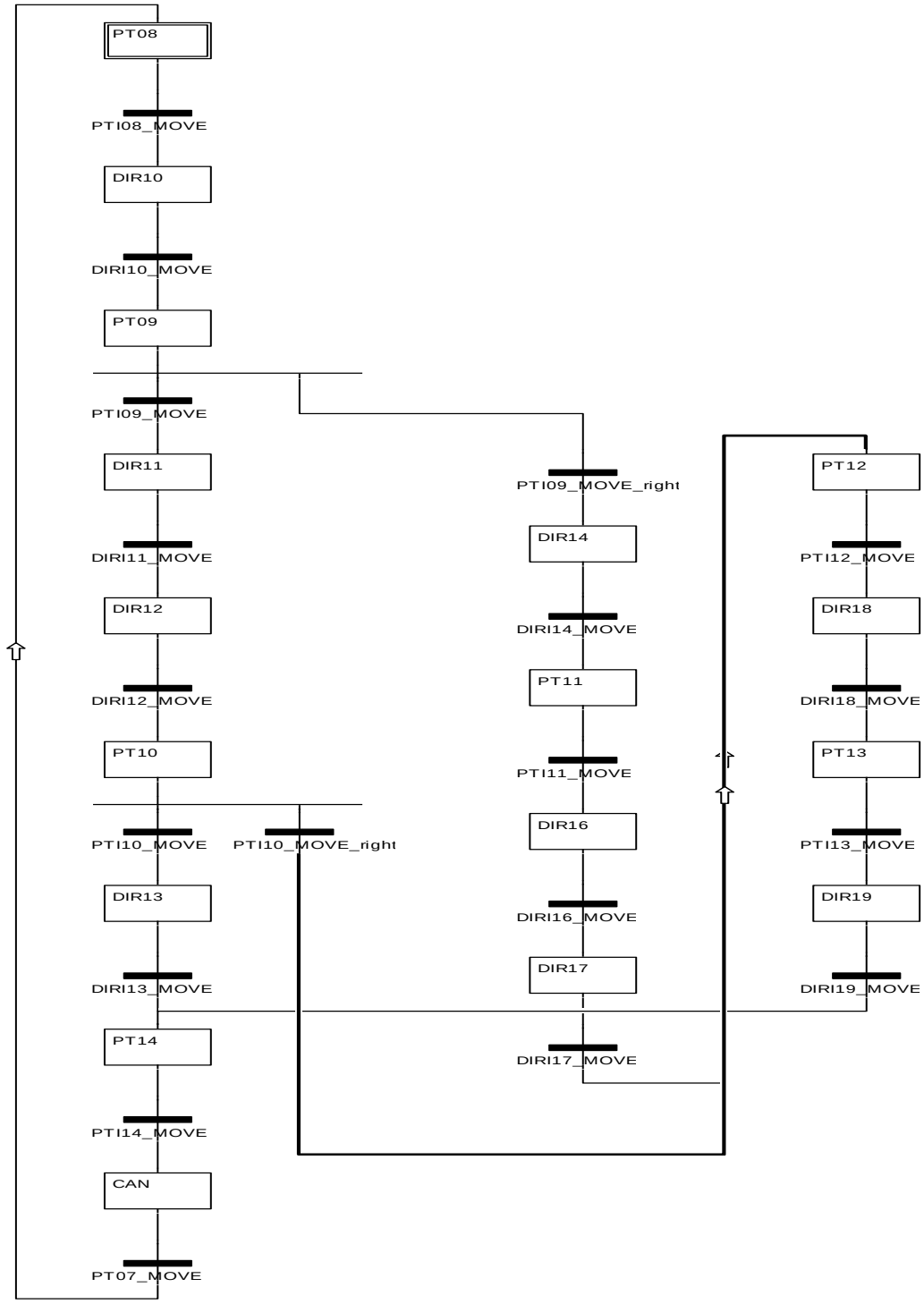



Ilustración 47: Diagrama flujo línea Profibus

## 5.4. Diagrama línea CAN:

La línea CAN comienza en la plataforma PT15, y termina en la PT07. Se elabora de la misma manera que el de profibus pero siguiendo el recorrido de las etapas de la línea CAN (ver ilustración 9):

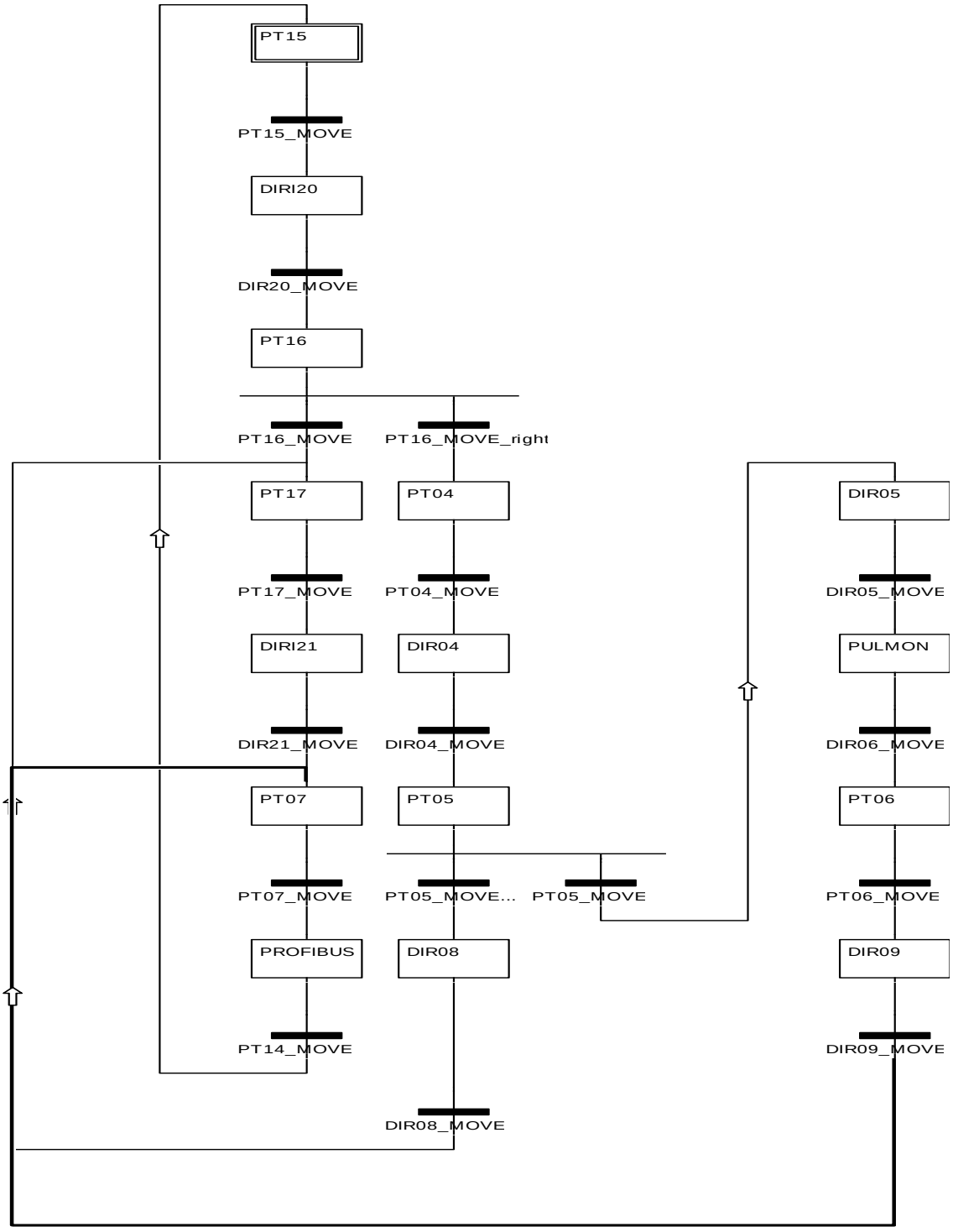



Ilustración 48: Diagrama flujo línea CAN

## 5.5. Diagrama Pulmón:

El pulmón se divide en 3 diagramas principales:

- **Condiciones iniciales:** el sistema tiene unos requisitos iniciales para el inicio de un ciclo de trabajo.

  Para que el sistema se ponga en condicionales se debe pulsar el botón verde de condiciones iniciales de la botonera, entonces el sistema seguirá las siguientes fases:

  - Actuador lineal abajo
  - Cadena retención con una aleta en posición de carga.

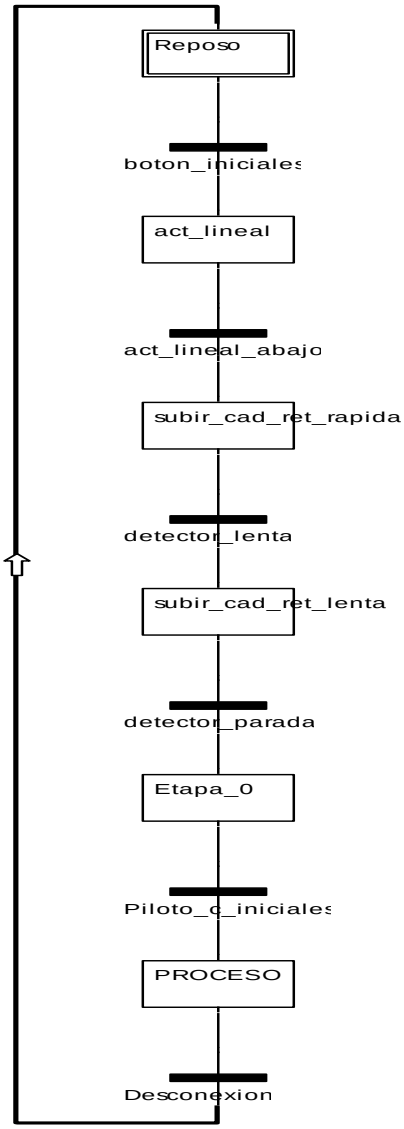



Ilustración 49: Condiciones iniciales pulmón

- **Cargar palet:** a continuación se puede observar el diagrama de flujo de la carga de una bandeja al pulmón.

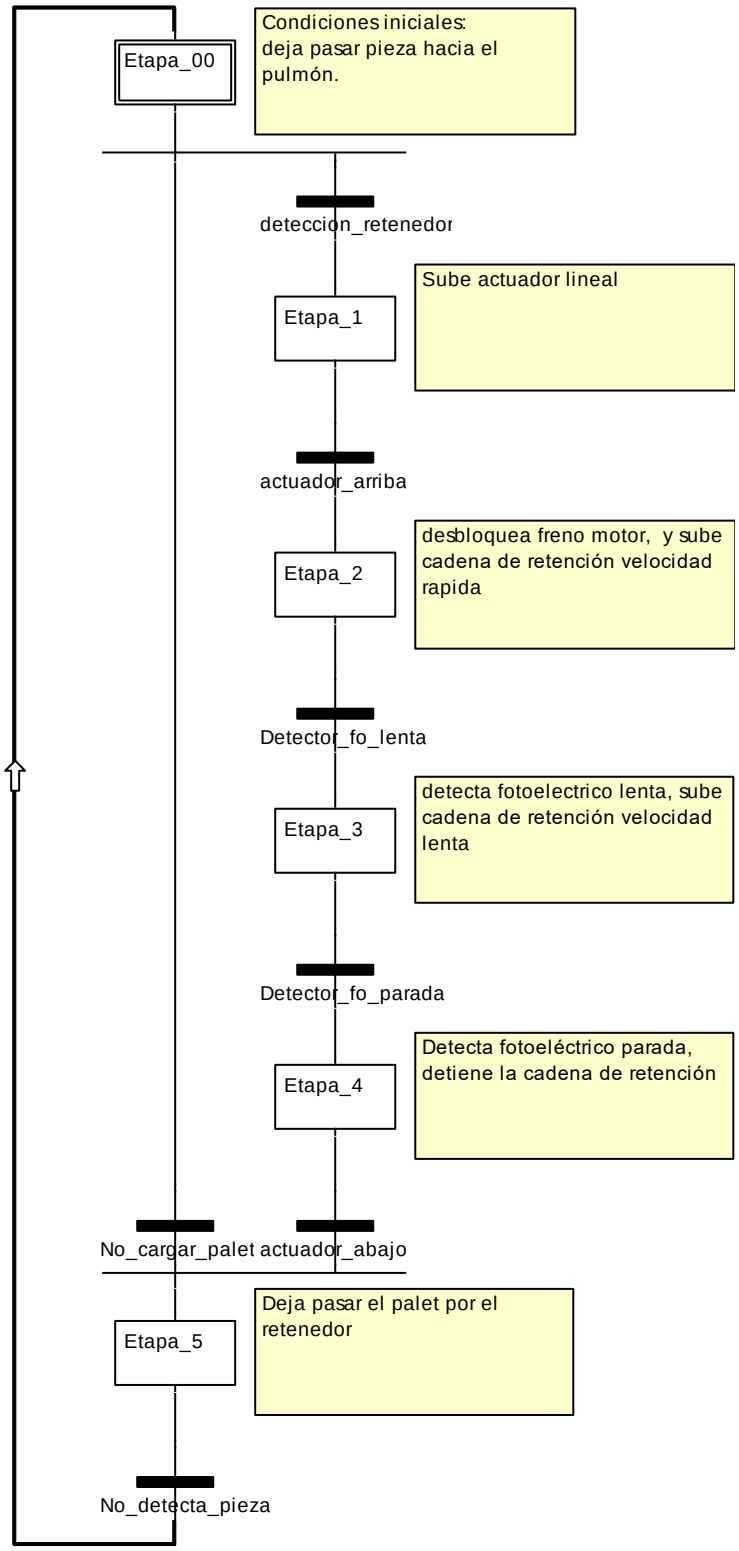



Ilustración 50: Diagrama carga bandeja

- **Descargar palet:** Diagrama de flujo de la descarga de una bandeja del pulmón.

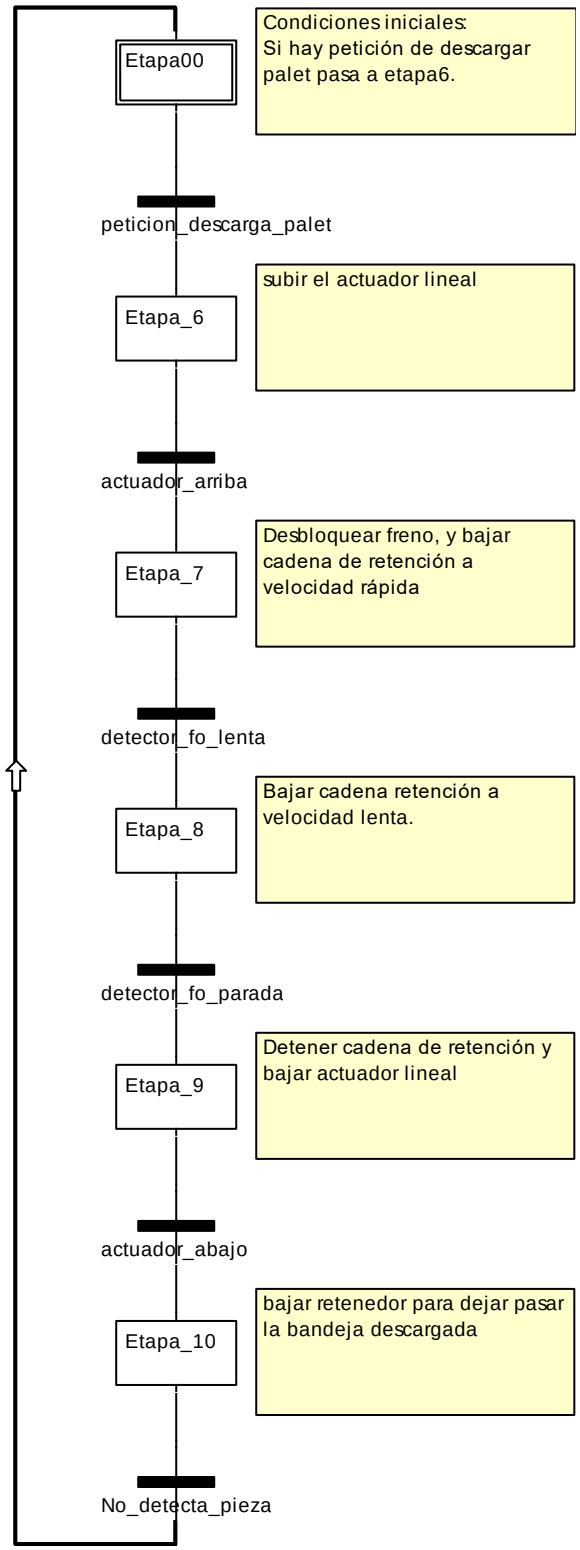



Ilustración 51: Descargar bandeja pulmón

# 6. Programación:

Posteriormente al diseño realizado, se deben traducir dichos diseños a lenguaje de programación ladder. Para ello se han utilizado marcas de programa, que son señales de memoria, las cuales pueden actuar como salidas y entradas dependiendo del método de programación del proceso.

Se ha decidido estructurar el programa en secciones según su aplicación en el sistema, de tal manera que se pueda tener una estructura ordenada de las funciones que se realizan.

Para la programación de los retenedores, se ha decidido el lenguaje ladder, ya que resulta más intuitivo utilizar contactos y bobinas para definir las transiciones y la activación de las salidas correspondientes.

**Descripción aplicación de ordenación de piezas:**

Una vez las líneas estén automatizadas y el pulmón en funcionamiento. se elabora una aplicación de carácter académico que trata de ordenar las piezas, de tal forma que se puedan cargar en el pulmón en dicho orden. Para ello sabiendo en qué punto se encuentra en cada pieza se trata de elaborar una función que relacione los números de las bandejas.

Una vez las bandejas están cargadas en el pulmón se define un nuevo orden de trabajo para ejecutar un nuevo ciclo.

## 6.1. Software:

El software utilizado es el Unity pro XL, que es el usual para PLC de Schneider. El entorno de programación está basado en proyectos. En este caso se necesita un proyecto diferente para cada línea (Profibus DP, CAN y el pulmón).

### 6.1.1. Configuración:

Lo primero que hay que configurar al abrir el Unity pro xl, es añadir los módulos del PLC dependiendo del proyecto que sea. Para ello se entra en configuración en el apartado 0: Bus PLC, y se van insertando los módulos necesarios. Se puede ver un ejemplo de la línea profibus con sus módulos insertados en el bus del sistema:

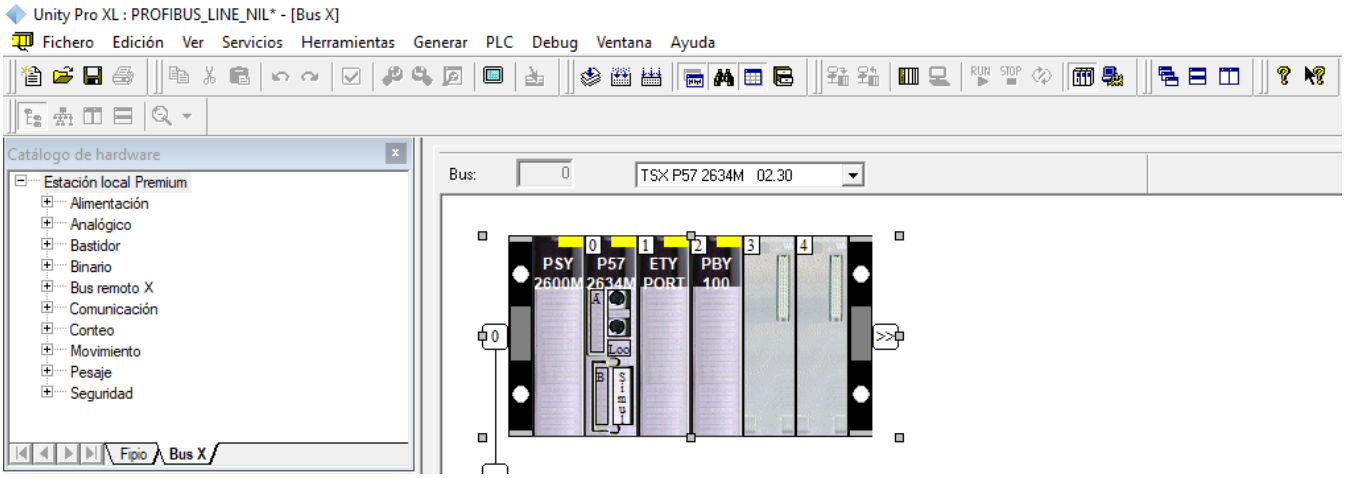



Ilustración 52: Configuración línea Profibus

Cada proyecto está organizado por secciones, que es donde se elaboran los programas. Al crear una sección te da una serie de lenguajes de programación para la sección:

Ladder (LD), estructurado (ST), Grafcet (SFC), y lenguaje de bloques (FBD). A continuación se puede ver un ejemplo ce la creación de la sección de la cinta central de la línea Profibus:

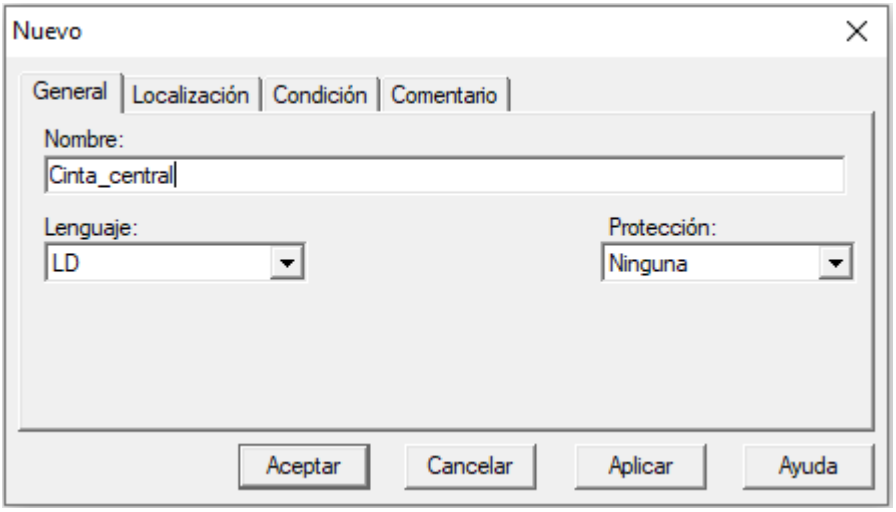



Ilustración 53: sección cinta central

### 6.1.2. Programa retenedor:

**Transición etapas de un retenedor (ejemplo DIR20):**

1. Estado reposo (DIR20_REST): al inicio siempre está activado.
2. Estado petición (DIR20_READY): se alcanza cuando está en estado reposo, y se produce una detección en el sensor inductivo (DIR20).
3. Estado avance (DIR20_MOVE) se alcanza en las siguientes condiciones:
    a. Está en estado de petición, y no está en avance.
    b. Siguiente posición en reposo (PT16_REST).
    c. Señal de bloqueo de seguridad del retenedor desactivada.
    d. No hay parada de emergencia.

Cuando está en avance y deja de detectar el sensor, pasa a estado avance_reposo hasta que detecte el sensor de la plataforma PT16.

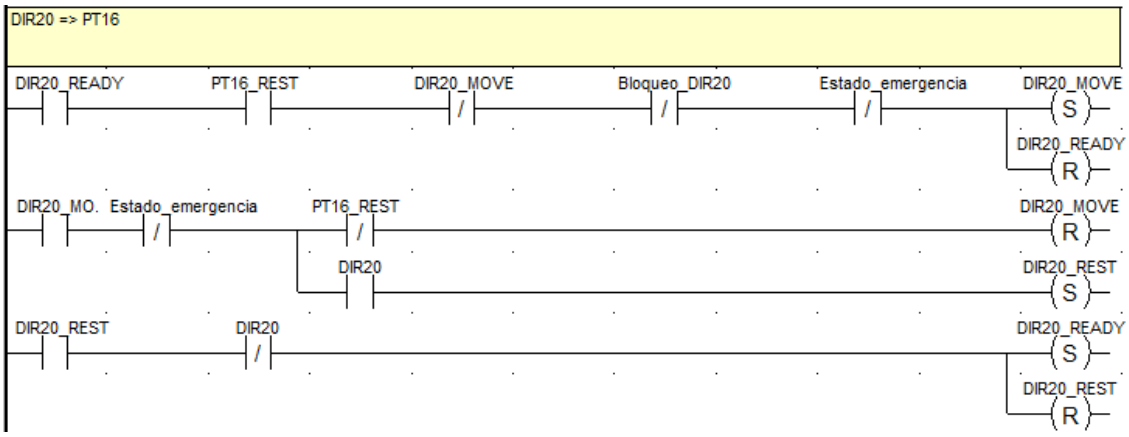



Ilustración 54: Programa de la transición de etapas de los retenedores

**Activaciones salidas:**

Si el retenedor está en avance (DIR20_MOVE), y no hay ningún otro estado a la vez, baja el retenedor hasta que deje de detectar la bandeja y pase 1 segundo del temporizador a la desconexión.

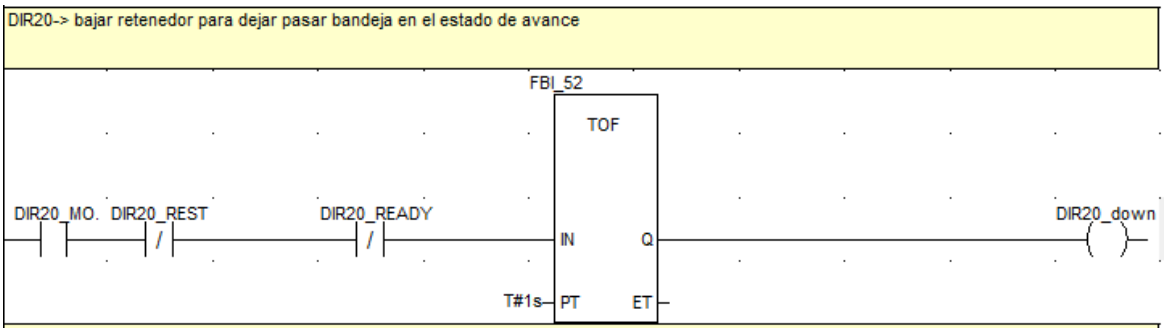



Ilustración 55. Activación salida retenedor

### 6.1.3. Programa plataforma:

**Transición etapas:**

1. Estado reposo (PT…_REST): estado de inicio de ciclo.
2. Estado petición (PT…_READY): se produce cuando se encuentra en estado reposo hay detección en el sensor inductivo basculante (DIB06).
3. Estado avance (PT…_MOVE): se alcanza en las siguientes condiciones:
    a. Está en estado de petición, y no está en avance.
    b. Siguiente posición en reposo (PT…_REST o DIR…REST).
    c. Señal de bloqueo de seguridad plataforma desactivada.
    d. No hay paro de emergencia.

    En el reset del estado de avance hay dos casos de programación:

    - caso 1: cuando deja de detectar el sensor de la plataforma, pasa a estado avance_reposo. A continuación se puede ver un ejemplo de programación de una plataforma (PT15) aplicando el caso 1:

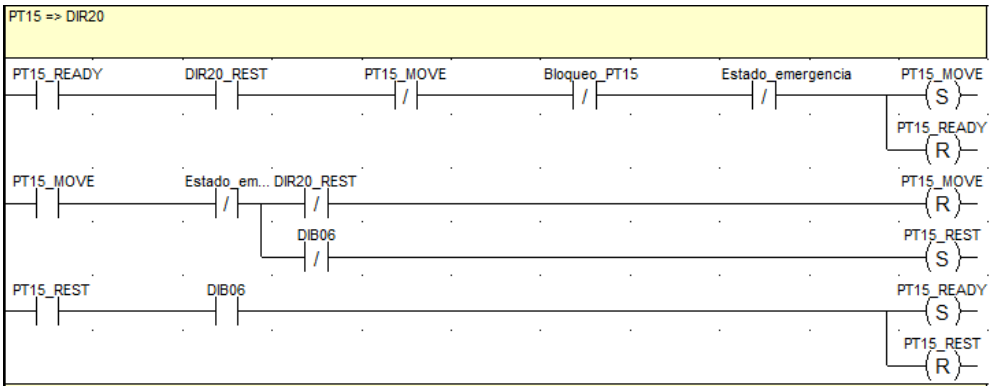



Ilustración 56: programación plataforma caso 1.

- caso 2: No se activa el reposo hasta que no llegue a la siguiente estación. Se utiliza para los cambios de cinta en las cintas transversales.

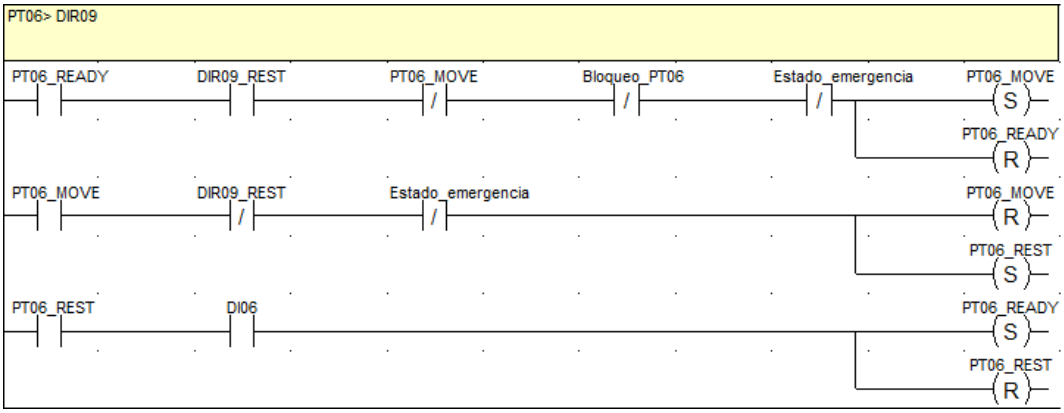



Ilustración 57: Programación plataforma caso 2

**Activaciones salidas:**

En la activación de la salida de la plataforma se debe tener en cuenta si es de 2 estados (subida y reposo), o de 3 estados (subida, reposo y bajada). También se deben tener en cuenta las cintas transportadoras transversales.

Las cintas transportadoras transversales sirven para cambiar de una cinta a otra, y para que una bandeja pueda pasar por las cintas transversales ha de estar en estado de subida.

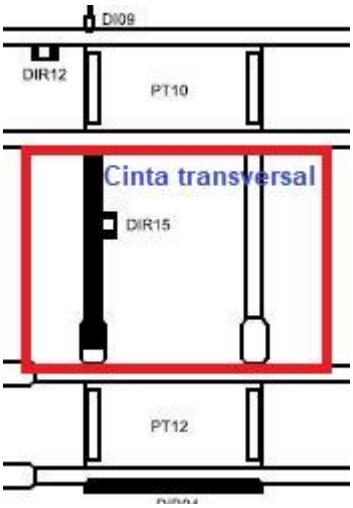



Ilustración 58: Cinta transportadora transversal

Las plataformas de 2 estados se clasifican en 2 casos según su estado al inicio de un ciclo de trabajo:

- **Estado inicial subida:** las plataformas que simplemente dejan pasar una bandeja sin pasar por las cintas transversales deben tener un estado inicial de subida cuando la plataforma no está en avance, por si han de bloquear la bandeja, o por si reciben la bandeja desde una cinta transversal. Cuando la plataforma está en estado de avance debe estar en reposo para dejar avanzar la bandeja. Un ejemplo sería la PT12:

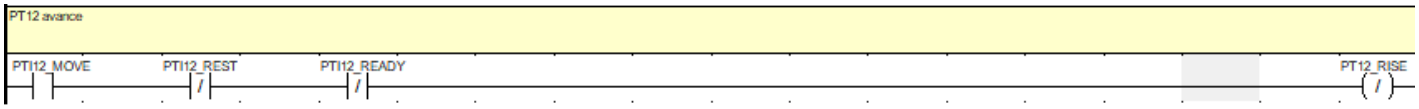



Ilustración 59: activación salida plataforma PT12

Se puede observar que cuando la PT12 está en avance, y no está activado ningún estado más a la vez, desactiva el estado de subida de la plataforma.

- **Estado inicial reposo:** se utiliza para las plataformas que deben pasar por las cintras transversales para cambiar de cinta al dejar pasar la bandeja. Dichas plataformas deben estar en estado de reposo cuando la plataforma no está en estado de avance, para dejar que la bandeja llegue. Para que la bandeja avance, cuando la plataforma esté en avance debe subir la plataforma para pasar por la cinta transversal. Un ejemplo sería la PT06:

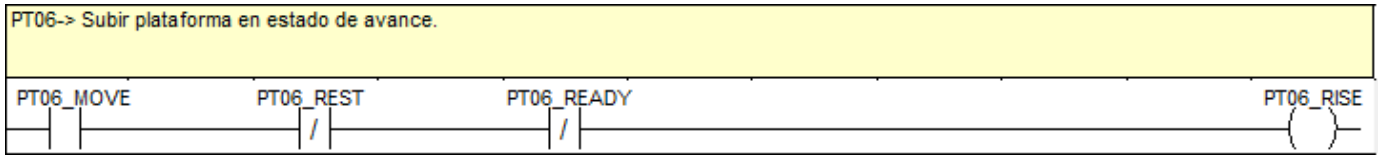



Ilustración 60: activación salida plataforma PT06

Cuando PT06 está en avance sin ningún estado a la vez activado, activa la subida de la plataforma.

Las plataformas de 3 estados también pueden tener el estado de reposo inicial o el de subida. Se diferencian en que tiene 2 señales de activación, la señal de subida y bajada.

### 6.1.4. Programación línea Profibus:

Una vez hecho el diagrama de flujo de la línea profibus, hay que traducirlo a lenguaje ladder o estructurado dependiendo del proceso a realizar. El programa de la línea profibus está compuesto por las siguientes secciones:

- **Reset:** sección en lenguaje estructurado (ST). Se utiliza para inicializar las variables de programa que sean necesarias de un estado inicial concreto. Se actualizan en cada ciclo de Scan.
  Se deben inicializar las variables de los estados de los retenedores/plataformas:
  - Reposo (REST) debe estar en estado 1 al inicio del ciclo.
  - Petición (READY) y avance (MOVE) deben estar en estado 0 al iniciar el sistema.
- **Programas cintas transportadoras:** Esta sección se ha realizado en lenguaje ladder (LD), ya que es una sucesión de etapas de retenedores y plataformas. El sistema dispone de 3 cintas transportadoras principales. Para cada cinta se deben programar 2 secciones, ya que cada cinta se divide en dos grandes programas:
  - **Transición etapas:** se basa en la transición por las diferentes estaciones de la célula.
    - **Cinta_central**: transición de las estaciones de trabajo en la central.
    - **Cinta_derecha1:** transición estaciones de la cinta derecha desde la plataforma PT09 a la PT12.
    - **Cinta_derecha2:** transición estaciones trabajo desde la PT12 a la PT14.

- **Activaciones salidas:** activación de las salidas del PLC necesarias para que los actuadores entren en funcionamiento.

- **Tracking:** sección que se ha decidido elaborar en lenguaje estructurado (ST), ya que se utilizan sentencias de control y variables compuestas (vectores).
  Esta sección tiene la función de elaborar un seguimiento de las bandejas, de tal forma que se pueda saber en todo momento donde esta cada bandeja y en qué punto del proceso.

  A la hora de elaborar la función de seguimiento de bandejas, hay que tener en cuenta las intersecciones que hay en la trayectoria de las bandejas en la línea profibus (ver plano ilustración 21).

  En la línea profibus hay 4 intersecciones, entonces se deben definir vectores entre cada intersección, tanto en la entrada como de salida. Si coinciden 2 vectores en un punto solo se define uno. Entonces los vectores son los siguientes:
  - **PT08 _INICIO:** es el vector de inicio del sistema, que indica que piezas se van a introducir en la célula, y transfiere sus valores a los siguientes vectores.
  - **PT08 -> PT09:** vector de entrada a la plataforma PT09, que tiene dos direcciones de salida.
  - **PT09 -> PT10:** vector seguir recto desde plataforma PT09 a la PT10. Coincide con el vector de entrada a la PT10.
  - **PT09 -> PT12:** vector salida PT09, con bifurcación hacia la derecha hasta PT12. Coincide con uno de los vectores de entrada a PT12.
  - **PT10 -> PT14:** vector seguir recto desde PT10 hasta la PT14.
  - **PT10 -> PT12:** vector de salida PT10 con bifurcación hacia la derecha hasta la PT12. Coincide con el otro vector de entrada a PT12.

- **PT12 -> PT14:** vector de salida de la PT12, en dirección a la PT14.
- **PT14 -> PT15:** vector salida PT14. Comunicación con línea CAN.

Además de los vectores, se necesitan unas variables que vayan indicando la posición actual del vector, que son los contadores.
Se define un contador para las entradas y salidas de las estaciones donde se producen intersecciones.

Cada vez que entre una bandeja de la estación en cuestión, transfiere el valor del vector de entrada correspondiente en la posición del contador al vector de salida en la posición que indique el contador, e incrementa el valor del contador para pasar a la siguiente posición del vector.
A continuación se puede ver un ejemplo para poder comprender mejor la metodología:

```
IF (PTI09_MOVE AND %M82=FALSE)THEN

    %M82:=TRUE;
    ArrayPT09_PT10[C3_DIRI11]:=ArrayPT08_PT09[C2_PTI09];
    C2_PTI09:=C2_PTI09+1;
    C3_DIRI11:=C3_DIRI11+1;
    IF (C2_PTI09=PIEZAS_TOTALES) THEN
        C2_PTI09:=0;
    END_IF;

END_IF;
```

Ilustración 61: Tracking Plataforma PT09

En el ejemplo se tiene el vector de entrada (ArrayPT08_PT09), el vector de salida (ArrayPT09_PT10), y los respectivos contadores, uno de entrada (C2_PTI09), y otro de salida (C3_DIRI11).

Cuando se produce un flanco positivo del estado de avance recto de la PT09 (sale una pieza de PT09 a PT10), se transfiere el valor del vector de entrada en la posición actual del contador de entrada al vector de salida en la posición actual del vector de salida.
Acto seguido se incrementan los contadores de entrada y salida para pasar a la siguiente posición de los vectores.

Si el contador de entrada alcanza el valor del total de las piezas que hay en la célula se pone a 0.

**Resumen de las funciones de la sección tracking:**

- Definir vectores de máximo 10 posiciones (máximo de piezas en la línea) en las entradas y salidas de las intersecciones.
- Definir contadores de entrada y salida de las estaciones en las que se producen intersecciones.
- Si se cumplen las condiciones de entrada o salida pieza (flanco ascendente de la petición o el avance), transferir valor vector de entrada en la posición del contador de entrada al vector de salida en la posición contador de salida.
- Incrementar contadores.
- Reset contador entrada si alcanza el número de piezas totales.

- **Ordening:** es una función empleada en lenguaje de texto estructurado (ST) dada la utilización de sentencias de control, que se basa en una vez realizado el seguimiento de las bandejas, poder ordenarlas según el número de bandeja indicado en los vectores.
  Este algoritmo es empleado únicamente en la línea profibus, que es la que se encarga de ordenar las piezas.

**Metodología de programación:**

- Definir una variable contador que empieza en 1, y marca el número de pieza actual en el orden correcto: contenido de la variable ha de ser enviado a la línea Can, para saber si debe cargar la pieza en el pulmón.
- Se compara la variable con el vector en la bifurcación actual.
- Coincide valor: avanza la pieza hasta el pulmón.
- Si el valor no coincide hay 2 casos posibles:
    - **Intersección tipo 1:**
        - Tiene 1 entrada y 2 salidas. Cuando llega a la entrada espera pieza estación anterior. Compara valores vectores en las 2 posiciones, si el valor de la estación actual es mayor que la anterior, la pieza gira hacia la derecha. En caso contrario avanza recto.

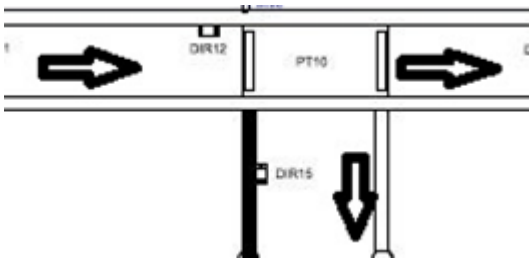



Ilustración 62: Bifurcación tipo 1

    - **Intersección tipo 2:**
        - Tiene 2 entradas y 1 salida. Compara los valores del vector de las 2 entradas. La que tiene el número de bandeja menor avanza.

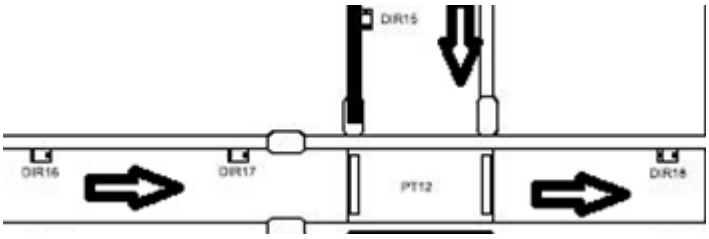



Ilustración 63: Bifurcación tipo 2

### 6.1.5. Programación línea CAN:

El programa de la línea CAN combina secciones en lenguaje ladder y en lenguaje estructurado. El programa consta de las siguientes secciones:

- **Reset:** sección en lenguaje estructurado (ST). Se utiliza para inicializar las variables de programa igual que en la línea Profibus.
- **Programas de las cintas:** programación de las cintas como en la anterior línea.
- **Activación salidas:** Activación de todas las señales de salida necesaria, siguiendo el mismo procedimiento que en la línea profibus.
- **Tracking:** La misma función que en la línea profibus pero con las estaciones de la línea CAN. En este caso no hay vector de inicio. Recibe el vector de la línea profibus.

### 6.1.6. Programación Pulmón:

El programa del pulmón dispone de las siguientes secciones:

- **Reset:** sección en lenguaje estructurado (ST). Se utiliza para inicializar las variables de programa.
- **Alarmas:** gestión y activación de las alarmas que tiene el sistema para su seguridad:
  - Puertas abiertas:
  - Pulmón lleno/vacio.
  - Fallo protecciones motores.
  - Bloqueo variador de frecuencia.
  - Palet con carga.
- **Estados:** Modos de operación del sistema.
  - Ciclo manual/automático
  - Parada de emergencia
- **Reposo:** Poner el proceso en condiciones iniciales.

- **Cargar_palet:** Función de cargar palet en el pulmón. Consiste en traducir las etapas del diagrama de flujo en lenguaje ladder, que se utilizan señales de bits de memoria (ejemplo: %MW100.1).
- **Descargar_palet:** Función de descargar palet usando bits de memoria para generar la sucesión de etapas.
- **Activación_salidas:** Activación de las salidas del PLC necesarias, que pueden ser activadas con varias condiciones.

# 7. Validación:

La etapa de validación se basa en un diagnóstico de las funciones que se han ido realizando en el estudio. Define todas las pruebas realizadas a nivel de funcionamiento o de programación.

## 7.1. Pruebas funcionales:

Las pruebas funcionales se emplean con los componentes y son independientes del código realizado. Es una verificación de que los componentes cumplen la función principal que desempeñan. Las pruebas funcionales que se han empleado en el proceso son:

Tabla 5: Pruebas funcionales

| Prueba | Diagnóstico |
|---|---|
| Detección sensores | Funcionalidad correcta |
| Activación actuadores | Los actuadores funcionan correctamente |
| Activación motores | Motores en funcionamiento óptimo. |
| Elementos de mando y señalización | Funcionalidad correcta. |

## 7.2. Pruebas operacionales:

Las pruebas operacionales, se basan en un diagnóstico de las funciones de programación realizadas. Son pruebas que evalúan una aplicación sobre el sistema, la cual incluye la sucesión de activaciones de componentes. Las pruebas funcionales empleadas en el sistema son:

Tabla 6: Pruebas operacionales

| Operación | Diagnóstico |
|---|---|
| Programación retenedores/plataformas | Funcionalidad correcta |
| Cargar pieza en el pulmón | Funcionalidad correcta |
| Descargar pieza del pulmón | Funcionalidad correcta |
| Aplicación temporizadores | Funcionalidad correcta |
| Aplicación de contadores | Funcionalidad correcta |
| Seguimiento bandejas | Funcionalidad correcta |
| Ordenación bandejas | Funcionalidad correcta |

# 8. Conclusiones:

Como conclusión, este proyecto se ha enfocado en la aplicación de las tecnologías de comunicación a nivel de campo para poder aplicarlas sobre un proceso industrial.

Se ha conocido el funcionamiento de los PLC y su impacto tanto en la automatización industrial, como a nivel de gestión de procesos productivos. El estudio también ha hecho referencia a los protocolos de red utilizados en las redes industriales.

Se han cumplido las expectativas en el sentido de que la línea puede trabajar de manera automática y en comunicación con el pulmón, y también se ha implementado el algoritmo de ordenación de las bandejas con un resultado satisfactorio.

**Posible continuación del estudio:**

Implementación de un nuevo controlador programable de Schneider para la línea profibus, con el fin de añadir nuevas señales en los controladores y ofrecer más posibilidades al proceso industrial.

Otra posible continuación es diseñar aplicaciones que impliquen diferentes tratamientos en las estaciones de trabajo, como la implementación de una plataforma de pesado de piezas presente en la célula flexible. Con ello se puede ampliar la funcionalidad de la aplicación.

# 9. Bibliografía:

**[1]** ieec.es. *Información sobre PLC.* [En línea] [01/09/2019]. Recuperada de:

<http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE6_1_2.pdf>

**[2]** logicbus.com. *Automatización.* [En línea] [10/09/2019]. Recuperada de:

<https://www.logicbus.com.mx/automatizacion.php>

**[3]** logicbus.com. *Automatización.* [En línea] [10/09/2019]. Recuperada de:

<https://www.logicbus.com.mx/automatizacion.php>

**[4]** Infoplc.net. *Ciclo de Scan autómatas.* [En línea] [15/09/2019]. Recuperada de: <https://www.infoplc.net/blogs-automatizacion/item/101930-ciclo-scan-automata-plc>

**[5]** wikipedia.org. *Sensor inductivo.* [En línea] [17/09/2019]. Recuperada de:

<https://es.wikipedia.org/wiki/Sensor_inductivo>

**[6]** Keyence.com. *Sensor fotoeléctrico.* [En línea] [17/09/2019]. Recuperada de:

<https://www.keyence.com.mx/ss/products/sensor/sensorbasics/photoelectric/info/index.jsp>

**[7]** wikipedia.org. *Bus CAN.* [En línea] [15/11/2019]. Recuperada de:

<https://es.wikipedia.org/wiki/Bus_CAN>

**[8]** wikipedia.org. *Modbus.* [En línea] [29/11/2019]. Recuperada de:

<https://es.wikipedia.org/wiki/Modbus>

**[9]** davidrojasticsplc.wordpress.com. *Arquitectura de los PLC.* [En línea] [10/12/2019]. Recuperada de:

<https://davidrojasticsplc.wordpress.com/category/7-arquitectura-de-los-plc/>

**[10]** Vicente Guerrero, Lluís Martínez. *Comunicaciones Industriales*.1ª Edición. 2009 Marcombo S.A.ISBN: 978-84-267-1574-6.

**[11]** wikipedia.org. *CANopen.* [En línea] [10/12/2019]. Recuperada de:

<https://es.wikipedia.org/wiki/CANopen>

**[12]** ingmecafenix.com. *Tipos de PLC*. [En línea] [10/12/2019]. Recuperada de:

<https://www.ingmecafenix.com/automatizacion/que-es-un-plc/>

**[13]** hetpro-store.com. *Puerto serial.* [En línea] [10/12/2019]. Recuperada de:

<https://hetpro-store.com/TUTORIALES/puerto-serial/>

**[14]** maleon.blogspot.com. *Trama Modbus TCP.* [En línea] [10/12/2019]. Recuperada de:

<http://maleonhe.blogspot.com/2010/03/modbus-tcpip.html>

**[15]** cic.es. *Evolución de la industria.* [En línea] [08/01/2020]. Recuperada de:

<https://www.cic.es/scada-y-la-industria-4-0/>