Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

**Adrian Fresco Iglesias**

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

# Control de una impresora de etiquetas mediante HMI

Febrero de 2015

Este proyecto, Control de una impresora de etiquetas mediante HMI, ha sido elaborado por Adrian Fresco Iglesias 71895262E a petición de la empresa Alcoa Avilés y en el desarrollo del mismo, se ponen a prueba los conocimientos adquiridos en el "Master en ingeniería de automatización e informática industrial" relacionados con la Automatización industrial.

En el desarrollo del vigente proyecto han participado José Ángel Sirgo como tutor por parte de Universidad de Oviedo y Abelardo Ablanedo como tutor de la empresa ALCOA.

# Índice

# 1. ENUNCIADO DEL PROYECTO

## 1.1 Objeto

La fábrica de aluminio primario de ALCOA avilés genera el aluminio producto en dos formatos diferentes, lingote o tocho.

El tocho es formato más solicitado entre los clientes y uno de los fines más habituales de este es la fabricación de ventanas.

El lingote está perdiendo importancia ya que la mayor parte de las ventas en este formato es empleado para la fabricación de automóviles y aeronáutica, actualmente sectores en crisis.

Las sierra de tochos, elemento estudio de este proyecto, es la encargada de cortar y empaquetar dicho producto. El operario de dicha maquina es capaz de hacer modificaciones en el producto final, pudiendo modificar el tamaño de los tochos así como el número de tochos incluidos en un paquete. Para cumplir estas especificaciones, así como otras, la maquina se sirve de un HMI que sirve de interfaz con el operario encargado de la maquina. Se crea así la necesidad de de imprimir etiquetas para poder identificar los paquetes de tochos que se envían a los clientes. En dicha etiqueta va incluidos datos de interés como son el formato, el número de colada, la aleación, el número de piezas incluido, el número de paquete, el peso, así como, el cliente.

En el sistema actual se dispone de una báscula y un HMI conectados a un PLC por medio de switch. Desde el HMI, el operario introduce una serie de datos, mientras que desde la báscula, llega al PLC la información del peso del paquete. El PLC envía este conjunto de datos a un PC desde el que se imprime la etiqueta, y se genera un fichero de texto para enviar al servidor que gestiona los datos de producción.

Para este proyecto se pretende eliminar el Pc que gobierna la impresora, conectando esta directamente a la red ethernet mediante un switch. También

se pretende que se envíen los datos al PC, en el que se gestionará su transmisión al servidor de producción.

Un punto interesante y de posible mejora, es que en las etiquetas se disponga de código de barras y QR.


Etiqueta estudio del proyecto

## 1.2 Amplitud y alcance del proyecto

En este documento detallaremos diferentes aspectos necesarios para la creación de este proyecto, como son los esquemas de conexiones, componentes utilizados, el código fuente del programa y se detallará como se utiliza y está programada dicha aplicación.

## 1.3 Peticionario

El proyecto ha sido desarrollado a petición de la empresa ALCOA Avilés para la sierra de tochos ubicada en la planta de fundición, con el fin de eliminar el Pc existente.

### 1.3.1 Alcoa

Alcoa es una empresa estadounidense, la tercera más grande productora de aluminio en el mundo detrás de Rio Tinto-Alcan, y Rusal.

Establecida en Pittsburgh, Pensilvania en 1888, adoptó el nombre de Aluminum Co. of América en 1907, del cual se deriva el mote actual. Alcoa introdujo el papel aluminio en 1910 y encontró usos para el aluminio en las incipientes industrias de aviación y del automóvil.

Alcoa cuenta en la actualidad con cuatro centros de producción en distintas comunidades de España y una oficina central en Madrid. Son líderes en la producción de aluminio en España, único productor de alúmina y aluminio primario.

Alcoa quiere apostar por las plantas que utilizan tecnología moderna para fabricar aluminio, como la de Lugo, e ir descartando las que emplean un sistema obsoleto, como la de Avilés y A Coruña. De este tipo quedan 13 en la multinacional y todas siguen usando tecnología Söderberg. Un sistema antiguo, con más de 50 años, que exige más personal y gasto eléctrico

La planta de ALCOA Inespal Avilés es productora de aluminio primario. Alcoa genera este producto de dos formas diferentes: mediante el uso de cubas Söderberg, aluminio nuevo, o mediante un proceso de reciclado de chatarra de aluminio, horno de reciclado.

El aluminio liquido, en sus dos vertientes, llega a la planta de fundición donde se solidifica y se le da forma de lingote o tocho para su posterior comercialización.

En la planta de fundición es donde está ubicada la sierra de tochos 2, máquina estudio de este proyecto. La sierra de tochos ya estaba operativa, solicitando por parte de Alcoa una modernización del software que permitiese eliminar un Pc ubicado en las inmediaciones de la misma.

## 1.4 Estado del arte

Para el desarrollo de dicho proyecto, y tal y como se había solicitado por la empresa, se ha solicitado información para el uso de la impresora mediante la red Ethernet gobernado desde el PLC. Esta solución es muy cómoda debido a la proximidad del switch a la impresora de trabajo y a que actualmente el PLC ya se encuentra conectado a la red.

Después de un estudio minucioso, nos damos cuenta de que los sistemas son incompatibles. Esto es debido a que la impresora se comunica mediante el protocolo TCP/IP y el PLC de Schneider lo hace mediante TCP/IP ModBus con los servicios preestablecidos, lo que hace que ambos sistemas sean incompatibles, teniendo que disponer de un PC.

A continuación y vista la imposibilidad de atacar el problema desde el PLC, se comprueban los componentes existentes en la instalación. Se opta por desarrollar el proyecto en el HMI, ya que esta pantalla se encuentra próxima a la impresora de trabajo y es un modelo actual que cuenta con diferentes puertos para la comunicación, como es el puerto serie, paralelo, usb y Ethernet.

Ahora tenemos que decidir por qué puerto del HMI nos comunicamos con la impresora.

El primer puerto que se comprueba es el Ethernet, ya que dicho puerto era el que en un principio se nos había pedido el desarrollo del proyecto, pero no es posible esta comunicación debido a limitaciones en el software de programación.

A continuación, se comprueba el USB, solicitando información de la compatibilidad de nuestra impresora Toshiba. La respuesta de Schneider es que “dicha impresora no es compatible con ese puerto ya que se necesita una impresora que soporte PCL3 o PCL5” protocolo del que no dispone nuestra impresora.

Por último hemos comprobado el puerto serie, ya que ambos componentes dispone de él. Se ha solicitado información y se comprueba que el software de programación del HMI permite el manejo avanzado de dicho puerto.

La programación del HMI se hace mediante el software Vijeo Designer, dicho software es el proporcionado por el fabricante de la pantalla y en el cual, aparte de la programación gráfica, permite el desarrollo de Scripts de Java para la programación de tareas especiales no preestablecidas por el software.

Para la comunicación entre ambos sistemas, HMI e impresora, se ha empleado el código ASCII, código estandarizado para la comunicación.

Para el correcto funcionamiento de la impresora, se ha procedido a la configuración mediante comandos de programación del lenguaje Java. Para realizar dicha configuración, ya que estos parámetros difieren dependiendo del fabricante, se ha tenido que hacer un estudio de la impresora, teniendo que solicitar información útil de dicho periférico al fabricante, Toshiba, para posteriormente trabajar con ella.

Una vez ajustado los parámetros propios de la impresora, se ha procedido a la programación del contenido a imprimir.

Por último hay que ajustar el contenido al hueco disponible en las etiquetas, por lo que habrá que hacer numerosas pruebas para el ajuste del contenido impreso, debido a que las etiquetas objeto de este proyecto ya vienen pre-impresas y nuestro software tiene que imprimir una serie de caracteres como son el peso, la aleación, formato, numero de piezas, …., datos de carácter alfanuméricos que completen los espacios en blanco incluidos en dichas etiquetas.

El sistema que forma objeto de este proyecto, incluye un autómata programable (PLC) Telemecanique de la compañía Schneider electric.

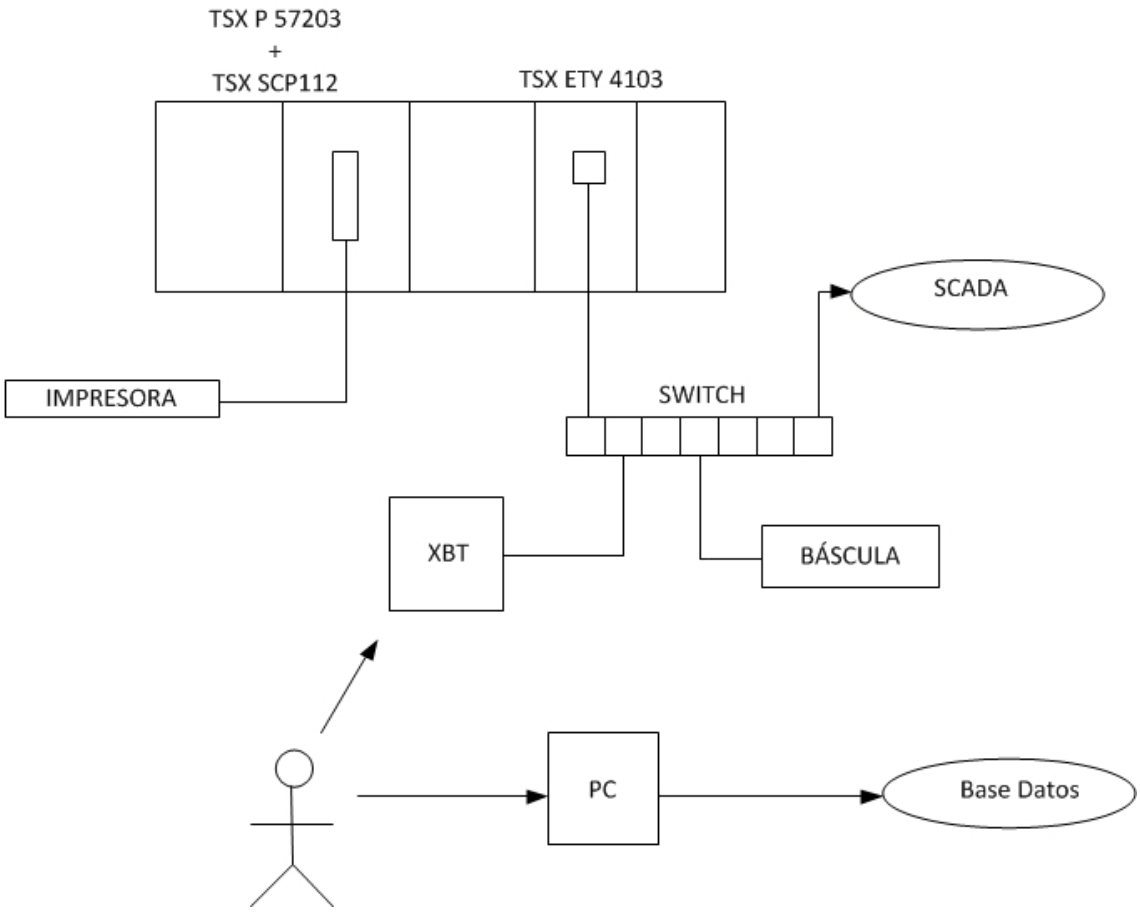
Dicho PLC está formado por una fuente de alimentación PSY 2600, una CPU TSX 57203, un puerteo de ethernet ETY 4103, además de unas series de entradas salidas encargadas del correcto funcionamiento de la máquina así como una serie de módulos para el contaje de los encoders.

También se dispone de una pantalla táctil XBT GT 6330, también del fabricante Schneider, anteriormente citada y donde se va a desarrollar el proyecto.

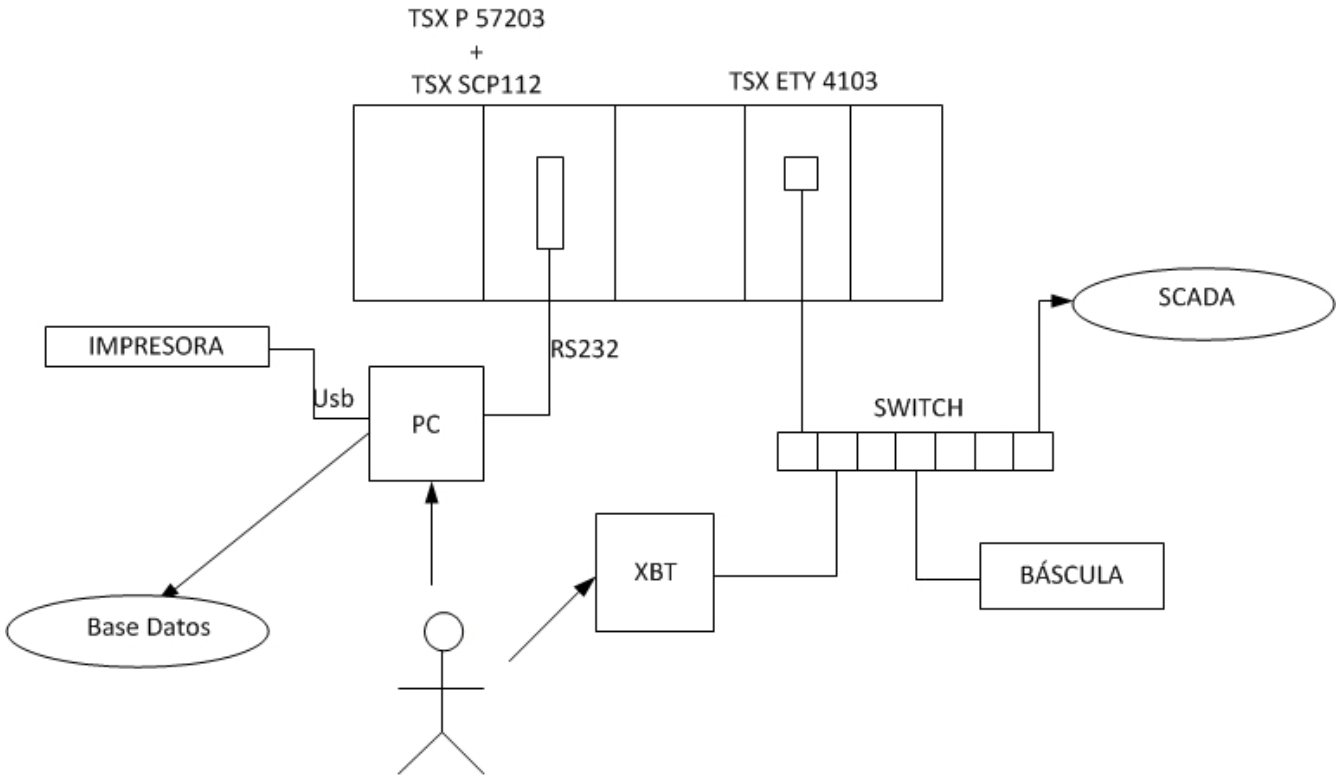
# 1.5 Instalaciones

En las siguientes figuras se muestran la evolución de las instalaciones, así como, los componentes donde se ha desarrollado el proyecto.

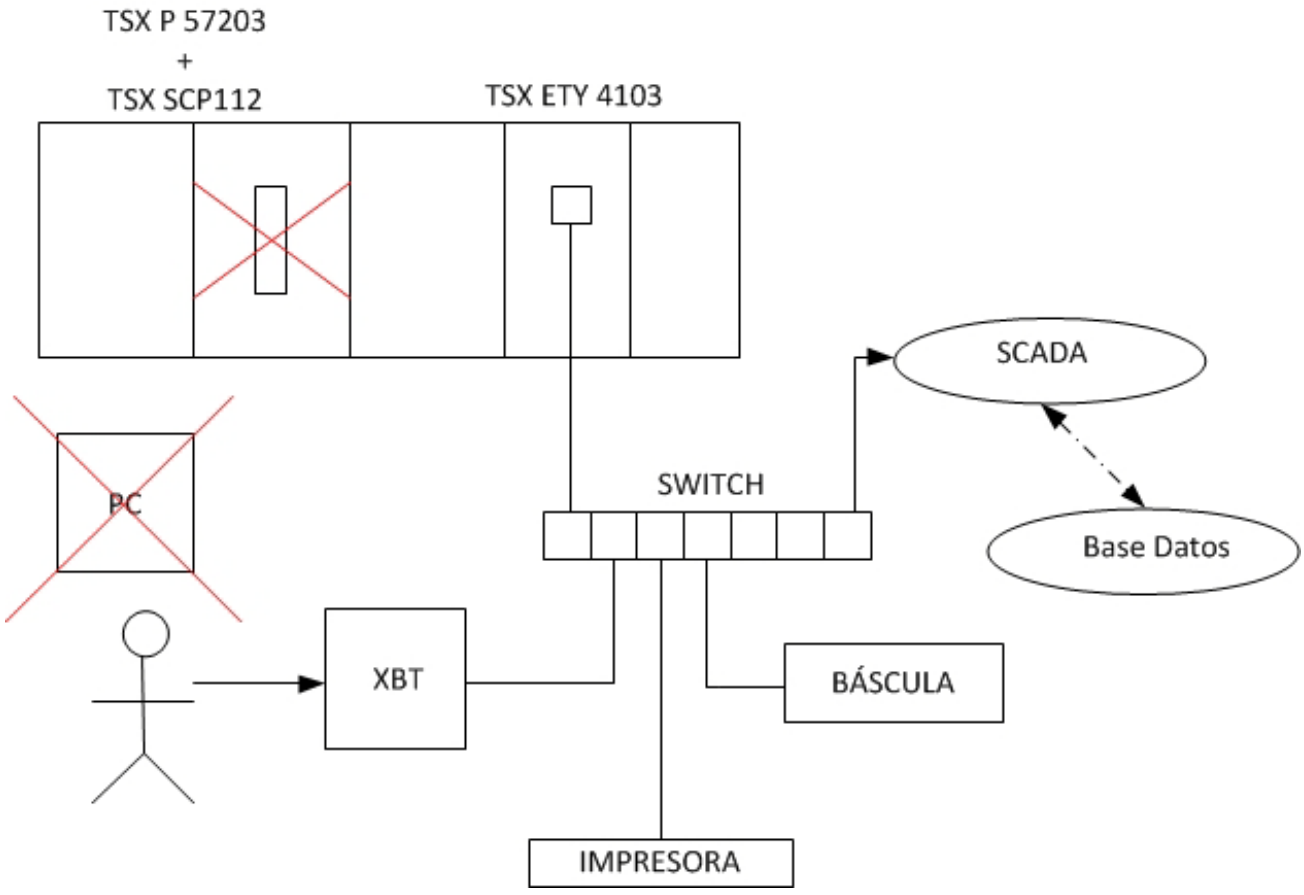
En la figura siguiente se muestra como estaban hace años conectadas las instalaciones objeto de este proyecto. En la figura se puede apreciar que la impresora estudio de dicho proyecto se encuentra conectada mediante un enlace serie al PLC.



En la figura siguiente se muestra como se encuentran las instalaciones en la actualidad. En la figura se puede apreciar que la impresora está conectada por medio de un cable Usb y gobernada por un Pc, el cual recibe los datos del PLC por medio de un enlace serie. Dicho Pc es el que la empresa desea eliminar de sus instalaciones.

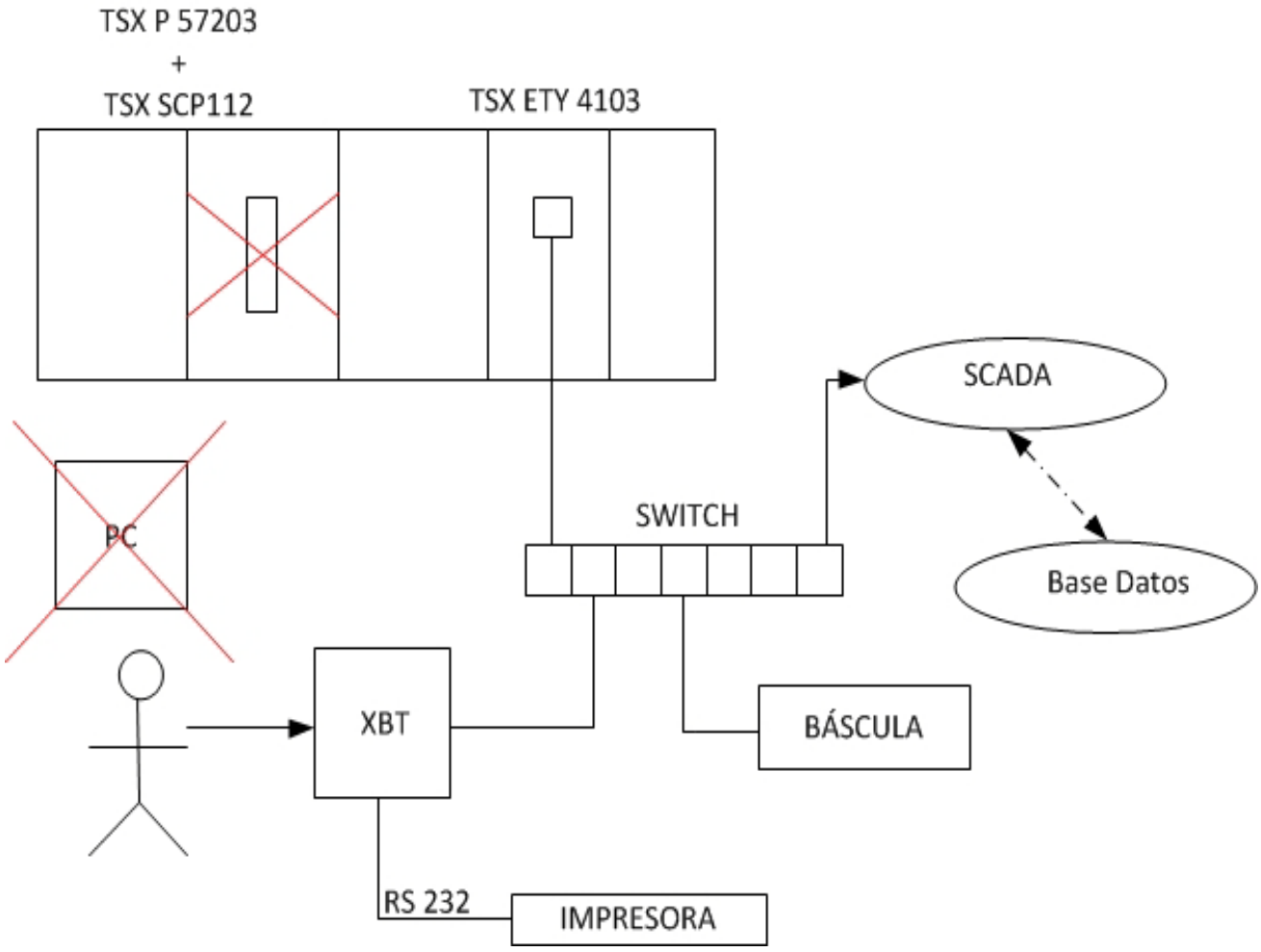


En la figura siguiente, se muestra el esquema solicitado por la empresa para el desarrollo de este proyecto. En él, se puede observar como desaparecen los enlaces serie y donde destaca que la mayoría de componentes vayan conectados al switch. Este esquema no se ha podido realizar debido a la incompatibilidad de protocolos




.

Por último se muestra el esquema que se ha desarrollado, donde destaca la conexión serie entre la pantalla HMI y la impresora. La pantalla HMI realizará la función de cerebro manejando la impresora, extrayendo los datos necesarios del PLC y enviándoselo tanto a esta, como a un registro de datos y el SCADA.



# 2. DESCRIPCIÓN DE LOS COMPONENTES

A continuación se describirá cada uno de los componentes utilizados en el sistema diseñado: PLC, impresora, HMI

## 2.1 PLC

Un controlador lógico programable o autómata programable, más conocido por sus siglas en inglés PLC (programmable logic controller), es una computadora utilizada en la automatización industrial, para automatizar procesos electromecánicos, tales como el control de la maquinaria de la fábrica en líneas de montaje o atracciones mecánicas.

Dentro de la definición de autómata programable entra el concepto de «caja negra» en la que existen, por una parte, unos terminales de entrada (o captadores) a los que se conectan señales analógicas o digitales (valor todo/nada) como por ejemplo pulsadores, finales de carrera, fotocélulas, detectores...; y por otra, unos terminales de salida (o actuadores) a los que se conectarán los elementos físicos sobre los que actuar, como por ejemplo bobinas de contactores, electroválvulas, lámparas., etc… de forma que la actuación de estos últimos está en función de las señales de entrada que estén activadas en cada momento, según el programa almacenado.

Los PLC son utilizados en muchas industrias y máquinas. A diferencia de las computadoras de propósito general, el PLC está diseñado para múltiples señales de entrada y de salida, rangos de temperatura ampliados, inmunidad al ruido eléctrico y resistencia a la vibración y al impacto. Los programas para el control de funcionamiento de la máquina se suelen almacenar en baterías copia de seguridad o en memorias no volátiles. Un PLC es un ejemplo de un sistema de tiempo real «duro», donde los resultados de salida deben ser producidos en respuesta a las condiciones de entrada dentro de un tiempo limitado, de lo contrario no producirá el resultado deseado.

Normalmente se requiere un PLC para:

-Reemplazar la lógica de relés para el comando de motores, máquinas, cilindros, neumáticos e hidráulicos, etc.

-Reemplazar temporizadores y contadores electromecánicos.

-Actuar como interfaz entre una PC y el proceso de fabricación.

-Efectuar diagnósticos de fallas y alarmas.

-Controlar y comandar tareas repetitivas y peligrosas.

-Regulación de aparatos remotos desde un punto de la fábrica.

Sus principales beneficios son:

-Menor cableado, reduce los costos y los tiempos de parada de planta.

-Reducción del espacio en los tableros.

-Mayor facilidad para el mantenimiento y puesta en servicio

-Flexibilidad de configuración y programación, lo que permite adaptar fácilmente la automatización a los cambios del proceso.

### 2.1.1 Principios básicos

Un controlador lógico programable o PLC está compuesto por dos elementos básicos:

La CPU, (Central Processing Unit) o Unidad Central de Procesamiento y la interfaz de Entradas y Salidas

El procesador, la memoria y la fuente de alimentación le otorgan la inteligencia necesaria al controlador la CPU lee la información en las entradas provenientes de diferentes dispositivos de censados (pulsadores, finales de carrera,

censores inductivos, medidores de presión, etc.), ejecuta el programa de almacenando en la memoria y envía los comandos a las salidas para los dispositivos de control (pilotos luminosos, contactores, válvulas, solenoides, etc.)

El proceso de lectura de Entradas, ejecución del programa y control de las Salidas se realiza en forma repetitiva y se conoce como SCAN.

Finalmente la fuente de alimentación suministra todas las tensiones necesarias para la correcta operación de la CPU y el resto de los componentes.

La aplicación se realiza mediante un software apropiado para PC.

Uno de los lenguajes que se puede utilizar para la programación de PLCs, es el Diagrama de Flujo Secuencial o SFC (anteriormente denominado Grafcet), reconocido como el lenguaje gráfico mejor adaptado a la expresión de la parte secuencial de la automatización de la producción.

El SFC representa la sucesión de las etapas en el ciclo de producción. La evolución de ciclo, Etapa por Etapa, se controla por una "Transición" ubicada entre cada etapa.

A cada una de las etapas le puede corresponder una o varias acciones. Para que el SFC evolucione a la siguiente etapa, debe cumplirse la/s condición/es ubicada en la transición entre ambas etapas.

Para asegurar la estandarización de los lenguajes de programación de los PLCs, y asegurarle al usuario una única forma de programar, sin importar la marca comercial del PLC, ha sido establecida la norma IEC 1131−3 que fija criterios en tal sentido.

Así, la norma define los lenguajes de programación: Ladder(contactos), Lista de instrucciones, Estructurado (Similar el Pascal), Bloques de Función y Diagrama Flujo de Secuencial (SFC o Grafcet).

Según el tipo de PLC que se escoja, podrá tener uno o más de estos lenguajes.

Cuando la aplicación crece en complejidad dado el tipo de señales a manejar, es posible incrementar la capacidad de Entradas/Salidas. Además permite el control de señales, tanto digitales como analógicas.

Un concepto que cada día es más necesario aplicar, es la comunicación entre PLCs o con un sistema de supervisión (SCADA) o con una base de datos. EL PLC dispone módulos de comunicación diseñados con este fin.

## 2.1.2 CPU TSX P 57203 características

A continuación, se describen las características de la CPU del PLC empleado en el desarrollo de este proyecto:

**Procesadores con software PL7**

| Tipo de procesador | | TSX 5720<br>16 racks máx. |
|---|---|---|
| Número de entradas/salidas | TON | 1.024 |
| en racks | analógicas | 80 |
| Regulación integrada | No | |
| Vías de funciones específicas (contaje, posicionamiento, pesaje) | | 24 |
| Bus | sistema de cableado AS-Interface | 4 |
| | máquina CANopen | 1 |
| | Campo InterBus, Profibus DP | 1 |
| Redes (Ethernet, Modbus Plus, Fipway) | | 1 |
| Capacidad de memoria | integrada | 48/64 K pals. datos/prog. (5) |
| | con ampliación PCMCIA | 48/64 pals. datos (5)/160 K pals. prog. |
| Tiempo de ejecución | booleana | 0,19 µs |
| para una instrucción | en palabra o aritmética | 0,25 µs |
| Referencias | sin puerto integrado | TSXP57203M |
| | Ethernet integrado | TSXP572623M |
| | fipio integrado | TSXP57253M |
| | Ethernet y Fipio integrado | TSXP572823M |

## 2.1.3 Módulo Ethernet TSX ETY 4103 características

A continuación, se describen las características del módulo ethernet del PLC empleado en el desarrollo de este proyecto:

| Tipo de módulo | | Red Ethernet TCP/IP | | | | |
|---|---|---|---|---|---|---|
| Velocidad | | 10 Mbits/s | 10/100 Mbits/s | | | |
| Servicios básicos | | Ethway, TCP/IP(Uni-TE,Modbus) | TCP/IP(Uni-TE,Modbus) | | | |
| Transparent Ready | Clase | C10 | B30 | B30 | C30 | D10 |
| | Global Data | – | Sí | Sí | Sí | – |
| | I/O Scanning | – | Sí | Sí | Sí | – |
| | TCP Open | Sí | – | – | Sí | – |
| Servidor web | Servicios básicos | Sí | Sí | Sí | Sí | Sí |
| | Servicios FactoryCast | Sí | – | – | Sí | – |
| | Servicios FactoryCast HMI | – | – | – | – Sí | |
| Referencias | | TSXETY110WS | TSXP57 | TSXETY4103 | TSXETY5103 | TSXWMY100 |

# 2.2 Sistemas de visualización y supervisión.HMI

## 2.2.1 Introducción

Sistema de visualización y control para procesos industriales desde terminal gráfico. HMI viene de las siglas de "Human Machine Inteface", es decir: Interfaz Hombre Máquina. Se trata de un elemento especialmente diseñado para establecer los parámetros necesarios de la máquina sin necesidad tener conocimiento de programación de PLC´s o derivados, sino que requiere exclusivamente del conocimiento de la operación de la máquina.

Proporciona una interactividad a través de la comunicación con los dispositivos de campo y puede controlar el proceso de forma automática, junto con el PLC.

Además, puede proveer de toda la información que se genera en el proceso a diversos usuarios, tanto del mismo nivel como de otros supervisores dentro de la empresa que requieran contacto con la operación de la máquina, así como control de calidad, supervisión, mantenimiento, etc.

La comunicación se realiza mediante buses de campo o redes LAN. Todo esto se ejecuta normalmente en tiempo real (Runtime), y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos

## 2.2.2 Objetivos

Un HMI cumple varios objetivos:

-Permiten una fácil e intuitiva interacción con la máquina a través de su capacidad táctil.

-Son elementos fácilmente integrables en arquitecturas abiertas, capaces de contener aplicaciones que pueden crecer o adaptarse según las necesidades cambiantes de la empresa.

-Se comunican con total facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión).

-Posibilidad de crear paneles de alarma, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.

-Generación de históricos de señal de planta, que pueden ser volcados para su proceso sobre una hoja de cálculo.

-Ejecución de programas, que modifican la ley de control, o incluso anulan o modifican las tareas asociadas al autómata, bajo ciertas condiciones (siempre que se implemente en el programa del PLC)

-Posibilidad de programación numérica, que permite realizar cálculos aparte de los realizados en el PLC, con lo que podemos distribuir de una forma más adecuada la carga de cálculos a realizar.

-Con ellas, se pueden desarrollar aplicaciones para ordenadores, con captura de datos, análisis de señales, presentaciones en pantalla, envío de resultados a disco e impresora, etc. a través de la carga de Runtime en el PC.

-Además, todas estas acciones se llevan a cabo mediante un paquete de funciones que incluye zonas de programación en un lenguaje de uso general (como Java), lo cual confiere una potencia muy elevada y una gran versatilidad.

### 2.2.3 XBT GT 6330 características

A continuación, se describen las características del HMI empleado en el desarrollo de este proyecto:

**Principal**

| | |
| --- | --- |
| Estatus comercial | Comercializado |
| Gama de producto | Magelis XBTGT |
| Tipo de producto o componente | Panel de pantalla táctil avanz |
| Tipo de pantalla | LCD TFT a color retroiluminado |

| | |
|---|---|
| Color de pantalla | 65536 colores |
| Resolución de la pantalla | 800 x 600 pixels SVGA |
| Tamaño de pantalla | 12,1 pulg. |
| Tipo de software | Software de configuración |
| Designación de software | Vijeo Designer |
| Sistema operativo | Magelis |
| Nombre del procesador | CPU RISC |
| Frecuencia de procesador | 266 MHz |
| Descripción de memoria | Copia seg. datos SRAM 512 kB batería litio<br>Memoria de aplicaciones flash EPROM 32 MB |
| Pares de nueces | Alimentación blq term rosca extrbls<br>Entrada digital blq term rosca extrbls<br>Enlace serie COM2 RJ45 RS485 <= 187,5 kbit/s MPI Siemens (187,5 kbitios/s)<br>Enlace serie COM1 RJ12/M8 RS232C/RS422/RS485 <= 115,2 kbits/s<br>Salida audio blq term rosca extrbls<br>3 salidas digitales blq term rosca extrbls<br>Ethernet TCP/IP RJ45<br>2 puertos maestros USB tipo A (V1.1) |
| Resistencia a descargas electroestáticas | 6 kV IEC 61000-4-2 nivel 3 |

| | |
|---|---|
| Dimensiones de corte | 301,5(+1/-0) x 227,5(+1/-0) mm |

## Complementario

| | |
|---|---|
| Zona sensible al tacto | 1024 x 1024 |
| Panel táctil | Analógico |
| Vida útil de la luz posterior | 50000 horas |
| Brillo | 8 niveles por panel táctil |
| Fuente del carácter | ASCII (caracteres europeos)<br>Chino (chino simplificado)<br>Japonés (ANK, kanji)<br>Coreano<br>Taiwanés (chino tradicional) |
| [Us] Tensión de alimentación | 24 V CC |
| Alimentación | Fuente de alimentación externa |
| Límites tensión alimentación | 19.2...28.8 V |
| Corriente de entrada | <= 30 A |
| Consumo de potencia en W | 30 W |
| Señal local | 1 LED (verde o naranja) para funcionamiento normal/fallo |

| | |
|---|---|
| | iluminación contraluz |
| Número de páginas | Limitado por capacidad de memoria interna |
| Protocolos | Uni-TE Telemecanique Modicon<br>Protocolos de terceros Siemens Simatic<br>Protocolos de terceros Rockwell Automation Allen-Bradley<br>Protocolos de terceros Omron Sysmac<br>Protocolos de terceros Mitsubishi Melsec<br>Modbus Telemecanique Modicon<br>Modbus TCP Telemecanique Modicon<br>Modbus Plus Telemecanique Modicon<br>FIPWAY Telemecanique Modicon |
| Reloj en tiempo real | Incorporado |
| Tipo de memoria | 1 ranura para tarjeta Compact Flash (de 128 MB a 1 GB) |
| Tipo ranura integr. | Para 1 tarjeta de comunicación fieldbus 1 (Device Net, Profibus DP) |
| Puerto Ethernet | 10BASE-T/100BASE-TX |
| Montaje de producto | Montaje empotrado |
| Modo de fijación | Por 4 abrazaderas de rosca<br>Con 4 clips de resorte |
| Front material | Aleación de aluminio |

| | |
|---|---|
| Material del envolvente | PPT |
| DESC | CE |
| Anchura | 313 mm |
| Altura | 239 mm |
| Profundidad | 56 mm |
| Peso del producto | 3 kg |

## Entorno

| | |
|---|---|
| Inmunidad a microcortes | <= 10 ms |
| Normas | EN 61131-2<br>FCC Class A<br>IEC 61000-6-2<br>UL 1604<br>UL 508<br>CSA C22.2 No 14 |
| Certificados de producto | Zona ATEX 2/22<br>CSA Clase 1 División 2 T4A<br>CSA Clase 1 División 2 T5<br>C-Tick<br>CULus<br>UL Clase 1 División 2 T4A<br>UL Clase 1 División 2 T5 |

| | |
|---|---|
| Temperatura ambiente de trabajo | 0...50 °C |
| Temperatura ambiente de almacenamiento | -20...60 °C |
| Humedad relativa | 10...90 % sin condensación |
| Altitud máxima de funcionamiento | < 2000 m |
| Grado de protección IP | IP65 (panel frontal) de acuerdo con IEC 60529<br>IP20 (panel trasero) de acuerdo con IEC 60529 |
| Grado de protección NEMA | NEMA 4X panel frontal (uso inter.) |
| Resistencia a los choques | 15 gn para 11 ms de acuerdo con IEC 60068-2-27 |
| Resistencia a las vibraciones | 3.5 mm de acuerdo con IEC 60068-2-6 (f = 5...9 Hz)<br>1 gn de acuerdo con IEC 60068-2-6 (f = 9...150 Hz) |
| Resistencia a campos electromagnéticos | 10 V/m de acuerdo con IEC 61000-4-3 |
| Resistencia a transitorios rápidos | 2 kV de acuerdo con IEC 61000-4-4 nivel_3 |

# 2.3 Impresora

Como indica su nombre, la impresora es el periférico que la computadora utiliza para presentar información impresa en papel u otro medio. Las primeras impresoras nacieron muchos años antes que el PC e incluso antes que los monitores (el otro dispositivo de salida por excelencia), siendo durante años el método más usual para presentar los resultados de los cálculos en aquellas primitivas computadoras, que previamente usaban tarjetas y cintas perforadas.

## 2.3.1 Definición de impresora térmica

Es un dispositivo electromecánico de alta velocidad, que tiene la función de recibir información digital, para por medio de calor, un haz de luz y una cinta entintada, plasmarla en la hoja, (también hay una variante en la cual, el haz de luz graba directamente sobre papel especial sin necesidad de cintas). Imprime básicamente en color negro aunque actualmente se encuentran en el mercado algunas que tienen la capacidad de imprimir en 2 ó 3 colores, se utiliza para la impresión de comprobantes de compra.

## 2.3.2 Como funciona una impresora térmica

Estas impresoras son libres de impacto, básicamente se utilizan para impresión monocromo. Cuentan internamente con chips y circuitos electrónicos que reciben órdenes y almacenan los datos para imprimirlos:
-La impresora térmica recibe la orden desde la computadora de lo que va a imprimir.

-La impresora térmica almacena los datos recibidos en una memoria RAM interna también llamada Buffer.

-Un mecanismo electromecánico acomoda la hoja acorde a las especificaciones que envía la computadora.

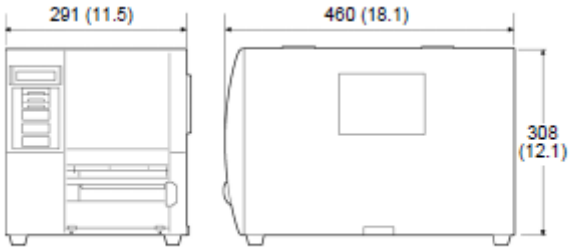
-Internamente cuentan con un sistema que por medio de un haz de luz, forma la imagen en una cinta especial.

-La cinta pasa sobre la hoja y se presiona sobre la misma, por medio de calor, se fija la tinta a la hoja.

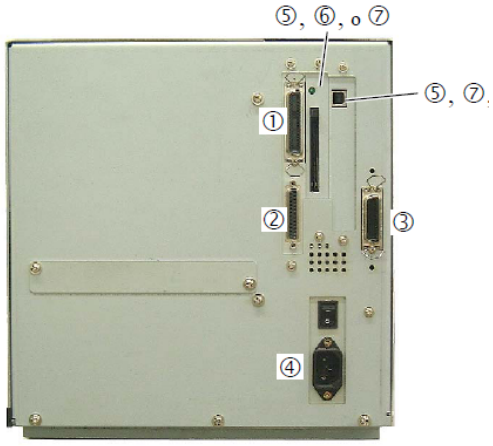
-Esto se repite hasta terminar los datos almacenados. Dependiendo el modelo de impresora térmica, esta puede enviar la señal hacia la computadora de que terminó de imprimir.

## 2.3.3 Impresora Toshiba B-SX4T

**Vista general:**




Dimensiones en mm (pulgadas)




① Conector Interface Paralelo (Centronics)
② Conector Interface Serie (RS-232C)
③ Conector de Expansión I/O (Opcional)
④ Entrada de Alimentación
⑤ Conector Interface USB (Opcional)
⑥ Slot para Tarjeta PCMCIA (Opcional)
⑦ Conector Interface de Red LAN (Opcional)

**Interfaz:**

## ***LAN***

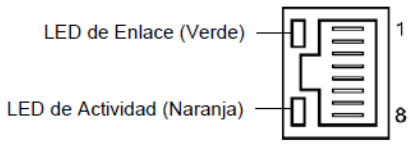
Capa Física: IEEE802.3 10BASE-T/100BASE-TX

Número de puertos: 1

Conector: RJ-45

LED de estado: LED de Enlace

LED de Actividad




| LED | Estado del LED | Estado de la RED |
|---|---|---|
| Enlace | ON | Detectado enlace de 10Mbps o 100Mbps. |
| | OFF | Enlace no detectado.<br>** No se puede realizar la comunicación mientras el LED de Enlace se encuentre apagado.* |
| Actividad | ON | Comunicando |
| | OFF | Inactivo |

Cable LAN: 10BASE-T: UTP categoría 3 o categoría 5

100BASE-TX: UTP categoría 5

Longitud del Cable: Longitud del segmento Máx. 100 m

## ***Interface Serie***

-Tipo: RS-232C

-Modo de Comunicación: Full duplex

| | | |
|---|---|---|
| -Velocidad de: | 2400 bps | 19200 bps |
| transmisión | 4800 bps | 38400 bps |
| | 9600 bps | 115200 bps |

| | | |
|---|---|---|
| -Sincronización: | Start-stop synchronization | |
| -Bit de Inicio: | 1 bit | |
| -Bit de Parada | 1 bit | |
| | 2 bits | |
| -Longitud de Datos: | 7 bits | |
| | 8 bits | |
| -Paridad: | No | |
| | PAR | |
| | IMPAR | |
| -Detección de Errores: | Paridad | |
| | Trama | |
| | Desbordamiento | |
| -Protocolo: | “Unprocedure communication” | |
| -Códigos entrada Datos: | código ASCII | código Shift JIS Kanji |
| | código JIS8 | código gráfico de 8 bits |
| | código JIS Kanji | código Europeo de caracteres de 8 bits |
| Buffer de recepción: | 1M byte | |

***Conector:***

| Pin Nº | Señal |
|---|---|
| 1 | FG |
| 2 | RD (Received Data |
| 3 | TD (Transmit Data) |
| 4 | CTS (Clear to Send) |
| 5 | RTS (Request to Send) |
| 6 | DTR (Data Terminal Ready) |
| 7 | SG (Signal Ground) |
| 20 | DSR (Data Set Ready) |

# 2.4 SCADA

Proviene de las siglas Supervisory Control and Data Adquisition (Supervisión, Control y Adquisición de Datos), son aplicaciones de software diseñadas con la finalidad de controlar y supervisar procesos a distancia. Se basan en la adquisición de datos de procesos remotos.

Este tipo de sistema es diseñado para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (PLC´s) y controlando el proceso de forma automática desde una computadora.

Además, envía la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa, es decir, que permite la participación de otras áreas, como por ejemplo: control de calidad, supervisión, mantenimiento, etc.

Las tareas de supervisión y control generalmente están más relacionadas con el software SCADA, en él, el operador puede visualizar en la pantalla del computador cada una de las estaciones remotas que conforman el sistema, los estados de éstas, las situaciones de alarma y tomar acciones físicas sobre algún equipo lejano.

Todo esto se ejecuta normalmente en tiempo real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos.

Un término clave en la definición, al que muchas veces no se le da adecuada atención, es el de supervisión, que significa que un operador humano es el que al final tiene la última decisión sobre operaciones, usualmente críticas de una planta industrial.

En nuestro sistema, hemos empleado el Scada como sistema de captación y visualización de los datos impresos en planta.

### 2.4.1 Componentes de un sistema scada

Se pueden ubicar a los componentes de un SCADA en dos grupos principales:

Software: Es un programa que permite construir la interfaz hombre –máquina y este debe ser capaz de restringir el acceso de las personas al sistema y generar señales de alarma en caso de fallo.

Hardware: Un sistema SCADA necesita ciertos componentes de hardware en su sistema para poder tratar y gestionar la información captada.

# 3. COMUNICACIÓN

## 3.1 Introducción

La automatización en la industria ha seguido un proceso gradual, aplicando la tecnología disponible en cada momento. Esto ha dado lugar a las denominadas “islas automatizadas”, término empleado para designar a una serie de equipos aislados entre sí y dedicados a una máquina o parte del proceso. Dichos equipos pueden ser PLC’s, ordenadores de diseño y gestión, controles numéricos, actuadores, sensores, etc.

El desarrollo de las comunicaciones y su aplicación en la industria ha permitido la implantación de redes industriales que facilitan la comunicación entre estas “islas automatizadas”, aumentando el rendimiento y las posibilidades de control. Entre las innumerables ventajas del empleo de redes de comunicación industrial, podemos destacar las siguientes:

-Visualización y supervisión de todo el proceso productivo.

-Mayor velocidad en la toma de datos.

-Mejora del rendimiento del proceso al realizar el control en su conjunto.

-Posibilidad de intercambio de datos entre diferentes sectores del proceso y departamentos.

-Posibilidad de programación y control a distancia sin tener que estar en campo.

-Ahorro energético.

Dependiendo del tipo de instalación, la implantación de una red de comunicación industrial puede ser o no rentable, lo que obliga a un estudio previo antes de su utilización.

## 3.2 Definición de comunicación

Cuando hablamos de comunicación de datos, conviene distinguir entre datos, que definimos como el conjunto de diferentes estados que puede adoptar una variable, e información, que es el resultado de procesar e interpretar esos datos, que en muchos casos, serán redundantes para garantizar una comunicación fiable.

Definiremos comunicación como el proceso de intercambio de datos, de cuyo análisis posterior se obtiene la información.

En la comunicación de datos intervienen varios elementos.

**Emisor/Receptor:** equipos que intervienen en la comunicación (ordenadores, PLC's, periféricos, etc.).

**Canal:** Medio físico capaz de propagar las señales (cable eléctrico, fibra óptica, aire, etc.).

**Mensaje:** datos que se transfieren entre el emisor y el receptor.

Las comunicaciones están sometidas a ruidos y perturbaciones de la misma naturaleza de las que circulan por el canal y que afectan negativamente a la transmisión. Además, los interlocutores deben utilizar el mismo código. En caso contrario el receptor no podría transformar los datos recibidos en información.

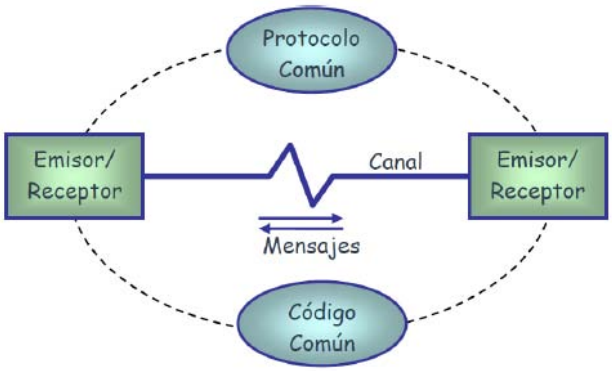
Como consecuencia de un flujo de datos bidireccional surge la necesidad de emplear un protocolo, que no es sino un conjunto de reglas que regulan el flujo de la información, y que establecen:

Quién y cómo comienza el diálogo.

Quién puede transmitir en cada momento.

Cómo termina la comunicación.

Además de este protocolo de regulación de flujo se deben establecer mecanismos de control de detección de errores y recuperación de datos en caso de error.




## 3.3 Modos de comunicación

La comunicación entre dos equipos (transmisión punto a punto) se puede producir de tres modos distintos, dependiendo de la dirección del flujo de datos:

***Simplex*:** si la comunicación se realiza en un solo sentido, desde un equipo emisor a un equipo receptor. Es el modo de comunicación más sencillo.

***Semi-dúplex*:** o half-duplex, si la comunicación se realiza en ambos sentidos, pero no simultáneamente. En este caso el canal compartido para la comunicación es el mismo para las transmisiones en ambos sentidos, por lo que se deben utilizar protocolos que regulen quién accede al canal en cada momento.

***Dúplex completo*:** o full-duplex, cuando la comunicación se puede realizar en ambos sentidos simultáneamente. Para ello, debe existir un medio físico de transmisión en cada sentido.

## 3.4 Arquitectura de las redes de comunicación

Se define como arquitectura de la red al conjunto de técnicas más importantes empleadas para su diseño y que además controlan su funcionamiento. Los aspectos más importantes en la arquitectura son:

**Topología**

Las principales topologías utilizadas son: estrella, bus, árbol y anillo.

**Medios de transmisión**

Se trata de los diferentes medios de soporte físicos que se emplean para la transmisión de la señal, como pueden ser los cables eléctricos, fibra óptica u ondas de radio.

**Métodos de acceso al medio**

Dado que los canales de transmisión de datos en las redes de comunicación son compartidos (un mensaje transmitido es escuchado por todos los demás), debe disponerse de un protocolo de acceso al medio que regule quién puede transmitir en cada momento.

Este problema no se presente en sistemas elementales, como en los enlaces punto a punto RS-232, donde al existir tan sólo dos interlocutores conectados mediante un canal fullduplex, cualquiera de ellos puede transmitir en cada momento.

Los métodos de acceso al medio definen cómo se inicia, se envía y finaliza la propagación de un mensaje por la red sin interferir o afectar a los demás sistemas. Los principales métodos empleados para tal fin son:

**Maestro/Esclavo:** Un dispositivo (maestro) indica cuándo puede transmitir cada uno de los dispositivos restantes (esclavos) mediante consulta secuencial de los mismos. Dada su sencillez y fiabilidad es muy utilizado en buses de campo industriales (Modbus, Profibus), y en la comunicación entre autómatas y dispositivos de entrada/salida remotos.

**Métodos de contienda:** Los dispositivos de la red compiten por acceder al medio de transmisión común. Sólo uno conseguirá iniciar la comunicación. Es el utilizado en redes Ethernet (802.3).

**Paso de testigo:** o Token-Passing, en este método se va pasando un paquete de datos especial, denominado testigo, entre los diferentes dispositivos. Sólo podrá transmitir el que esté en posesión de dicho testigo. Existen diferentes variantes, con o sin prioridad, y se utilizan con diferentes topologías (Token-Ring, Token-Bus).

## 3.5 Protocolos empleados

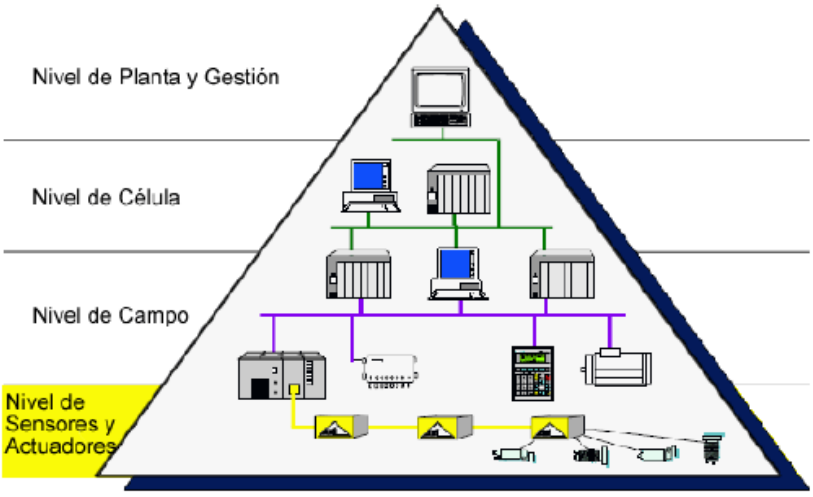
Podemos clasificar los protocolos, dependiendo únicamente del tiempo en la transmisión/recepción, como:

**Determinista:** cuando el tiempo es fijo siempre y conocido, como por ejemplo un sistema de comunicación AS-i, que tarda 5ms en realizar la emisión/recepción de 31 esclavos y 10ms para 62 esclavos. También son deterministas las redes Profibus y Profinet

**No Determinista o probabilístico:** cuando el tiempo es aleatorio, es decir, no siempre es el mismo y por tanto no es conocido. Un ejemplo es la red Ethernet que utiliza el método de acceso por contienda CSMA/CD.

## 3.6 Niveles en una red industrial

La integración de los diferentes equipos y dispositivos existentes en una planta se hace dividiendo las tareas entre grupos de procesadores con una organización jerárquica. Así, dependiendo de la función y el tipo de conexiones, se suelen distinguir cuatro niveles




**Nivel de Sensores y Actuadores:** es el nivel más próximo al proceso. Aquí encontramos las máquinas con las que opera la empresa, y con ellas, todos los sensores y actuadores para la toma de medidas y realizaciones de control sobre el proceso.

**Nivel de Campo:** integra pequeños automatismos (PLC's compactos, PID's, multiplexores de e/s, etc.) en subredes o "islas". En el nivel más alto de estas redes podemos encontrar uno o varios autómatas modulares actuando como maestros de la red. En este nivel se emplean los buses de campo.

**Nivel de Célula:** enlaza las células de fabricación o zonas de trabajo. A este nivel se sitúan los autómatas de gama alta y los ordenadores dedicados al diseño, control de calidad, programación, etc. En este nivel es donde se suelen utilizar las rede tipo LAN (MAP o Ethernet).

**Nivel de Planta y Gestión:** este es el nivel más alto y se encarga de integrar los siguientes niveles en una estructura de fábrica o varias fábricas. Se suelen emplear estaciones de trabajo que establecen la conexión entre el proceso productivo y la gestión (ventas, stocks, etc.). Las rede empleadas son del tipo LAN o WAN (para plantas situadas en diferentes lugares).

Esta estructura no es universal, varía con el tamaño del proceso y sus características particulares. Además, para cualquiera de los niveles, no hay un estándar universalmente aceptado que cubra todos los aspectos desde el nivel físico al de aplicación (si nos referimos al modelo OSI de ISO).

# 3.7 Buses de campo

Un bus de campo es un sistema de transmisión de información (datos) que simplifica enormemente la instalación y operación de máquinas y equipamientos industriales utilizados en procesos de producción.

El objetivo de un bus de campo es sustituir las conexiones punto a punto entre los elementos de campo y el equipo de control a través del tradicional bucle de corriente de 4-20mA.

Típicamente son redes digitales, bidireccionales, multipunto, montadas sobre un bus serie, que conectan dispositivos de campo como PLC's, transductores, actuadores y sensores. Cada dispositivo de campo incorpora cierta capacidad de proceso, que lo convierte en un dispositivo inteligente, manteniendo siempre un costo bajo. Cada uno de estos elementos será capaz de ejecutar funciones simples de diagnóstico, control o mantenimiento, así como de comunicarse bidireccionalmente a través del bus.

Las características fundamentales que el bus de campo debe cumplir, en lo referente a la conexión de dispositivos, son:

**Interconectividad**: al bus se deben poder conectar de forma segura dispositivos de diferentes fabricantes que cumplan el protocolo. Es el nivel mínimo, y no proporciona, en principio, ninguna ventaja.

**Interoperabilidad**: los dispositivos de diferentes fabricantes funcionan satisfactoriamente en el mismo bus.

**Intercambiabilidad**: los dispositivos de un fabricante pueden ser sustituidos por otros equivalentes, de otro fabricante, y seguir funcionando. Este es el objetivo final, y sólo se consigue si las especificaciones son completas y se dispone de un sistema de prueba y validación

## 3.7.1 Ventajas e inconvenientes de un bus de campo

Los buses de campo, si son correctamente elegidos para la aplicación, ofrecen numerosas ventajas, como:

**Flexibilidad**

El montaje de un nuevo instrumento supone la simple conexión eléctrica al bus y una posterior configuración/programación, normalmente remota (desde la sala de control). Si se trata de buses abiertos, resultará posible la conexión de instrumentos de distintos fabricantes al mismo bus.

**Seguridad**

Transmisión simultánea de señales de diagnóstico de sensores y actuadores, permitiendo así instalaciones más seguras.

**Precisión**

Transmisión totalmente digital para variables analógicas.

**Facilidad de mantenimiento**

Resulta posible diagnosticar el funcionamiento incorrecto de un instrumento y realizar calibraciones de forma remota desde la sala de control. Esto permite localizar rápidamente conexiones erróneas en la instalación, con lo que los errores de conexión son menores y más rápidamente solucionados (reducción de los tiempos de parada y pérdidas de producción).

**Reducción de la complejidad** del sistema de control en términos de hardware:

Reducción drástica del cableado.

Se elimina la necesidad de grandes armarios de conexiones para el control del equipamiento asociado.

Reducción del número de PLC’s.

Reducción de tiempo de instalación y personal necesario para ello.

Por el contrario, el **principal inconveniente** que ofrece la utilización de un bus de campo es la posible rotura del cable de bus. Esto conllevaría la caída de todos los elementos que estuvieran conectados al bus y probablemente una parada general del proceso. Hay también que advertir que en la actualidad los buses de campo son muy robustos ante interferencias y entornos agresivos.

### 3.7.2 Situación actual de la estructura de redes: incompatibilidad

Como hemos contemplado anteriormente, hasta ahora se pueden distinguir tres tipos distintos de estructuras en una red de comunicación industrial según en el nivel de la pirámide CIM. Pues bien, esta situación está cambiando en la actualidad y, debido a las características tan atractivas que ofrecen las redes LAN Ethernet, algunos buses de campo ya establecidos como Modbus, Profibus, etc., están adoptando este protocolo, con lo cual, obtenemos una red a medio camino entre ambas. Estas redes, aunque están basadas en Ethernet, han modificado el protocolo para resolver los problemas de indeterminismo que presenta, así como la adaptación de los dispositivos Ethernet a los entornos industriales. En todo caso esas opciones no son gratuitas. La tarjeta Ethernet empieza a encarecerse cuando se la dota de robustez para un entorno industrial.

Parece difícil que Ethernet tenga futuro a nivel de sensor, aunque puede aplicarse en nodos que engloban conexiones múltiples de entrada-salida.

Como conclusión, Ethernet está ocupando un área importante entre las opciones para redes industriales, llegando a penetrar en los niveles bajos de la pirámide CIM.

### 3.7.3 Niveles OSI en un bus de campo

Idealmente, las especificaciones de un bus de campo deberían cubrir los siete niveles OSI, aunque lo más frecuente es que implementen sólo tres:

**Nivel físico:**

Especifica el tipo de conexión, naturaleza de la señal, tipo de medio de transmisión, etc. Normalmente, las especificaciones de un determinado bus admiten más de un tipo de medio físico. Los más comunes son de tipo RS485 o con conexiones en bucle de corriente.

**Nivel de enlace:**

Se especifican los protocolos de acceso al medio (MAC) y de enlace (LLC). En este nivel se definen una serie de funciones y servicios de la red mediante códigos de operación estándar.

**Nivel de aplicación:**

Es el dirigido al usuario, y permite la creación de programas de gestión y presentación, apoyándose en las funciones estándar definidas en el nivel de enlace. En este nivel se define el significado de los datos. Las aplicaciones suelen ser propias de cada fabricante (no hay un nivel de aplicación estándar para buses de campo).

# 3.8 Ethernet

## 3.8.1Introducción

Ethernet, al que también se conoce como IEEE 802.3, es el estándar más popular para las LAN, usa el método de transmisión de datos llamado Acceso múltiple con detección de portadora y detección de colisiones (CSMA/CD). Antes de que un nodo envíe algún dato a través de una red Ethernet, primero escucha y se da cuenta si algún otro nodo está transfiriendo información; de no ser así, el nodo transferirá la información a través de la red. Todos los otros nodos escucharán y el nodo seleccionado recibirá la información. En caso de que dos nodos traten de enviar datos por la red al mismo tiempo, cada nodo se dará cuenta de la colisión y esperará una cantidad de tiempo aleatoria antes de volver a hacer el envío.

Cada paquete enviado contiene la dirección de la estación destino, la dirección de la estación de envío y una secuencia variable de bits que representa el mensaje transmitido. El dato transmitido procede a 10 millones de bits por segundo y el paquete varia en una longitud de 64 a 1518 bytes, así el tiempo de transmisión de un paquete en la Ethernet esta en un rango de 50 a 1200 microsegundos dependiendo de su longitud. La dirección de la estación de destino normalmente es referida por una única interfaz de red. Cada estación recibe una copia de cada paquete, pero ignora los paquetes que son dirigidos a otras computadoras y procesa solamente los que son dirigidos a ella.

Las velocidades de envío de paquetes utilizando la tecnología Ethernet son de 10 Mbps (Ethernet estándar), 100 Mbps (Fast Ethernet – 100BaseX) y de 1000 Mbps utilizando el Gigabit Ethernet cuya especificación se encuentra respaldada por la IEEE con número 802.3z, el cual cumple los siguientes objetivos:

• Permite realizar operaciones de envío y recepción de datos a una velocidad de 1000 Mbps.

• Usa el formato de Ethernet 802.3.

• Usa el método de acceso CSMA/CD con soporte para un repetidor por dominio de colisión.

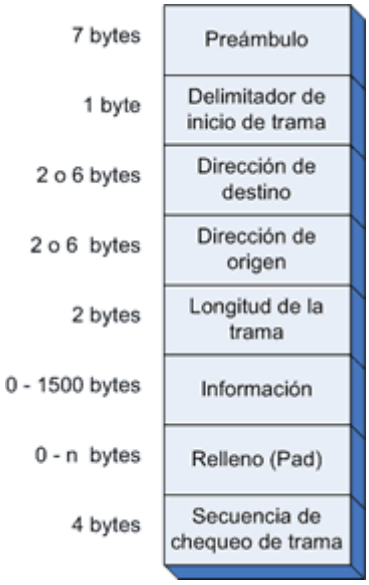
• Las direcciones de retorno son compatibles con las tecnologías 10BASE-T y 100Base-T.

### 3.8.2 Trama de transmisión CSMA/CD

Se define a una trama de transmisión como el grupo de bits en un formato particular con un indicador de señal de comienzo de la trama.

El formato de la trama permite a los equipos de red reconocer el significado y propósito de algunos bits específicos en la trama. Una trama es generalmente una unidad lógica de transmisión conteniendo información de control para el chequeo de errores y para el direccionamiento.

El formato de la trama CSMA/CD (IEEE 802.3) se muestra a continuación en la siguiente figura:




Los distintos componentes de la trama CSMA/CD son responsables de las siguientes tareas:

Preámbulo

Es responsable de proveer sincronización entre los dispositivos emisor y receptor.

El delimitador de inicio de trama

Indica el comienzo de una trama de datos. Está formado de la siguiente secuencia de 8 bits, 10101011.

Dirección de destino y dirección de origen

Pueden tener una longitud tanto de 2 bytes como de 6 bytes. Ambas direcciones, origen y destino, deben tener la misma longitud en todos los dispositivos de una red dada. El campo dirección de destino específica la estación o estaciones a las cuales están dirigidos los datos. Una dirección que referencia a un grupo de estaciones es conocida como dirección de grupo de multicast, o dirección de grupo de multidifusión. Una dirección que referencia a todas las estaciones de una red es conocida como dirección de difusión. La dirección de origen identifica a la estación que está haciendo la transmisión.

Longitud de la trama

Indica la longitud del campo de datos que se encuentra a continuación. Es necesaria para determinar la longitud del campo de datos en los casos que se utiliza un campo Pad (campo de relleno).

Información

Contiene realmente los datos transmitidos. Es de longitud variable, por lo que puede tener cualquier longitud entre 0 y 1500 bytes.

Relleno

Un campo Pad o de relleno es usado para asegurar que la trama alcance la longitud mínima requerida. Una trama debe contener mínimo un número de bytes para que las estaciones puedan detectar las colisiones con precisión.

Secuencia de chequeo de trama

Es utilizada como mecanismo de control de errores. Cuando el dispositivo emisor ensambla la trama, realiza un cálculo en los bits de la trama. El algoritmo usado para realizar este cálculo siempre genera como salida un valor de 4 bytes. El dispositivo emisor almacena este valor en el campo de chequeo de secuencia de la trama. Cuando el receptor recibe la trama, realiza el mismo cálculo y compara el resultado con el del campo de chequeo de secuencia de la trama. Si los dos valores coinciden, la transmisión se asume como correcta. Si los dos valores son diferentes, el dispositivo de destino solicita una retransmisión de la trama.

## 3.9 Modbus

Modbus es un protocolo de comunicación serie desarrollado y publicado por Modicon en 1979. En su origen el uso de Modbus estaba orientado exclusivamente al mundo de los controladores lógicos programables o PLC´s de Modicon. No hace falta más que echar un vistazo al mercado industrial actual para darse cuenta que, a día de hoy, el protocolo Modbus es el protocolo de comunicaciones más común utilizado en entornos industriales, sistemas de telecontrol y monitorización. Lo que implica de forma implícita que: tanto a nivel local como a nivel de red, en su versión TCP/IP, seguirá siendo uno de los protocolos de referencia en la industria.

El objeto del protocolo Modbus es bien sencillo: La transmisión de información entre distintos equipos electrónicos conectados a un mismo bus. Existiendo en dicho bus un solo dispositivo maestro (Master) y varios equipos esclavos (Slaves) conectados.

En su origen estaba orientado a una conectividad a través de líneas serie como pueden ser RS-232 o RS-485, pero con el paso del tiempo han aparecido variantes como la Modbus TCP, que permite el encapsulamiento del Modbus serie en tramas Ethernet TCP/IP de forma sencilla. Esto sucede porque desde un punto de vista de la torre OSI, el protocolo Modbus se ubica en la capa de aplicación.

El hecho que se haya extendido su uso hasta convertirse en el protocolo más estandarizado en el sector industrial se debe a varias razones diferenciales respecto a otros protocolos:

El estándar Modbus es público, lo que permite a los fabricantes desarrollar dispositivos tanto Master como Slave. Este hecho facilita el acceso a la información y estructura del protocolo .

Desde un punto de vista técnico, su implementación es muy sencilla y en consecuencia el tiempo de desarrollo se acorta considerablemente respecto a otros protocolos en los que se complica la estructura de las tramas y en consecuencia el acceso a los datos que no están almacenados en estructuras complejas.

La transmisión de información no está comprometida a ningún tipo de datos. Lo que implica cierta flexibilidad a la hora del intercambio de información. Esto quiere decir, que si se transmite un dato de 16bits de información su

representación no está sujeta a ninguna restricción, por lo que puede tratarse de un dato tipo Word con signo, un entero sin signo de 16bits o la parte alta de una representación tipo Float de 32bits, etc.La representación del valor vendrá definida por la especificación que el fabricante dé del dispositivo, lo que permite la representación de un amplio rango de valores.

### 3.9.1 Funcionamiento y elementos de una red ModBUS

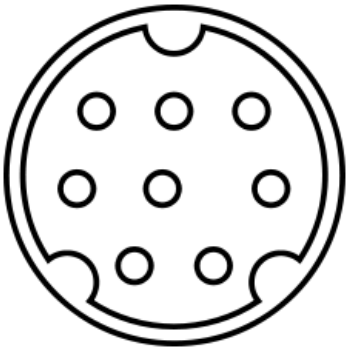
El funcionamiento tiene una base muy sencilla: El Master pregunta y los Slaves responden o actúan en función de lo que este diga.

Un dispositivo conectado al bus ejerce de maestro solicitando información del resto de dispositivos conectados que ejercen como esclavos y son quienes suministran la información al primero. Según el estándar Modbus y dada su implementación, en una red Modbus habrá un Master y hasta un máximo de 247 dispositivos Slaves. Esta limitación está determinada por el simple hecho que en una trama Modbus la dirección del esclavo se representa con un solo Byte, existiendo algunas direcciones reservadas para propósitos específicos como broadcast, etc.

Lo dicho, en una red Modbus todos los dispositivos esclavos deben tener una dirección asignada que debe estar comprendida entre la 1 y la 247.Desde un punto de vista práctico, no pueden coexistir dos dispositivos esclavos con la misma dirección Modbus. Dentro de la trama Modbus RTU, la dirección del esclavo corresponde al primer byte. En una red Modbus el Master no sólo puede ejercer la función de recompilar información de los esclavos mediante preguntas, sino que puede interactuar con ellos o alterar su estado, pudiendo escribir además de leer información en cualquiera de ellos.

Con el paso de los años y según la evolución de las redes de comunicaciones entre dispositivos electrónicos, así como de la conectividad entre dispositivos, han ido apareciendo variantes del protocolo Modbus que estaba pensado en su inicio para redes implementadas sobre líneas serie. La evolución más utilizada/conocida es la que se conoce como Modbus TCP, una “versión” del protocolo Modbus que permite la implementación de este protocolo sobre redes Ethernet, en consecuencia, aumenta el grado de conectividad. Está “versión” del protocolo encapsula la trama base del protocolo Modbus en la capa de aplicación TCP/IP de forma sencilla.

# 3.10 Puerto serie



Un puerto serie o puerto serial, es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente. La comparación entre la transmisión en serie y en paralelo se puede explicar usando una analogía con las carreteras. Una carretera tradicional de un sólo carril por sentido sería como la transmisión en serie y una autovía con varios carriles por sentido sería la transmisión en paralelo, siendo los vehículos los bits que circulan por el cable

## 3.10.1 Introducción

En tecnologías básicas, un puerto serie es una interfaz física de comunicación en serie a través de la cual se transfiere información mandando o recibiendo un bit. A lo largo de la mayor parte de la historia de los ordenadores, la transferencia de datos a través de los puertos de serie ha sido generalizada. Se ha usado y sigue usándose para conectar los ordenadores a dispositivos como terminales o módems. Los ratones, teclados, y otros periféricos también se conectaban de esta forma.

Mientras que otras interfaces como Ethernet, FireWire, y USB mandaban datos como un flujo en serie, el término "puerto serie" normalmente identifica el hardware más o menos conforme al estándar RS-232, diseñado para interactuar con un módem o con un dispositivo de comunicación similar.

Actualmente en la mayoría de los periféricos serie, la interfaz USB ha reemplazado al puerto serie por ser más rápida. La mayor parte de los

ordenadores están conectados a dispositivos externos a través de USB y, a menudo, ni siquiera llegan a tener un puerto serie.

El puerto serie se elimina para reducir los costes y se considera que es un puerto heredado y obsoleto. Sin embargo, los puertos serie todavía se encuentran en sistemas de automatización industrial y algunos productos industriales y de consumo.

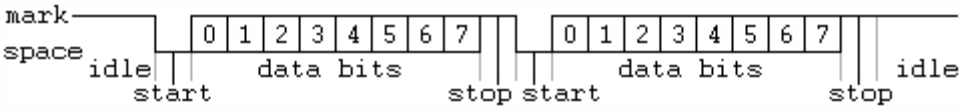
Los dispositivos de redes, como los enrutadores y switches, a menudo tienen puertos serie para modificar su configuración. Los puertos serie se usan frecuentemente en estas áreas porque son sencillos, baratos y permiten la interoperabilidad entre dispositivos. La desventaja es que la configuración de las conexiones serie requiere, en la mayoría de los casos, un conocimiento avanzado por parte del usuario y el uso de comandos complejos si la implementación no es adecuada.

### 3.10.2 Puerto serie asincrónico

A través de este tipo de puerto la comunicación se establece usando un protocolo de transmisión asíncrono. En este caso, se envía en primer lugar una señal inicial anterior al primer bit de cada byte, carácter o palabra codificada. Una vez enviado el código correspondiente, se envía inmediatamente una señal de stop después de cada palabra codificada.

La señal de inicio (start) sirve para preparar al mecanismo de recepción o receptor, la llegada y registro de un símbolo, mientras que la señal de stop sirve para predisponer al mecanismo de recepción para que tome un descanso y se prepare para la recepción del nuevo símbolo.

La típica transmisión start-stop es la que se usa en la transmisión de códigos ASCII a través del puerto RS-232, como la que se establece en las operaciones con teletipos.




El puerto serie (también conocido como COM o serie RS-232) es del tipo asincrónico, utiliza cableado simple desde 3 hilos hasta 25 y conecta computadoras o microcontroladores a todo tipo de periféricos, desde terminales a impresoras y módems pasando por ratones.

La interfaz entre el RS-232 y el microprocesador generalmente se realiza mediante el chip UART 8250 (computadoras de 8 y 16 bits, PC XT) o el 16550 (IBM Personal Computer/AT y posteriores).

La norma RS-422, similar al RS-232, es un estándar utilizado en el ámbito industrial.

### 3.10.3 Puertos serie modernos

Uno de los defectos de los puertos serie iniciales era su lentitud en comparación con los puertos paralelos -hablamos de 19.2 kbits por segundo- sin embargo, con el paso del tiempo, están apareciendo multitud de puertos serie de alta velocidad que los hacen muy interesantes ya que presentan las ventajas del menor cableado y solucionan el problema de la merma de velocidad usando un mayor apantallamiento, y más barato, usando la técnica del par trenzado. Por ello, el puerto RS-232, e incluso multitud de puertos paralelos, se están sustituyendo reemplazándose por los nuevos puertos serie como el USB, el FireWire o el Serial ATA.

# 3.11 RS 232

**RS-232** (Recommended Standard 232, también conocido como EIA/TIA RS-232C) es una interfaz que designa una norma para el intercambio de una serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE ( Equipo de Comunicación de datos), aunque existen otras en las que también se utiliza la interfaz RS-232

*Conector RS-232 (DB-9 hembra).*

En particular, existen ocasiones en que interesa conectar otro tipo de equipamientos, como pueden ser computadores. Evidentemente, en el caso de interconexión entre los mismos, se requerirá la conexión de un DTE (*Data Terminal Equipment*) con otro DTE. Para ello se utiliza una conexión entre los dos DTE sin usar módem, por ello se llama: null módem ó módem nulo.

El RS-232 consiste en un conector tipo DB-25 (de 25 pines), aunque es normal encontrar la versión de 9 pines (DE-9, o popularmente mal denominados DB-9), más barato e incluso más extendido para cierto tipo de periféricos (como el ratón serie del PC).

La interfaz RS-232 está diseñada para imprimir documentos para distancias cortas, de hasta 15 metros según la norma, y para velocidades de comunicación bajas, de no más de 20 kbps. A pesar de esto, muchas veces se utiliza a mayores velocidades con un resultado aceptable. La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal simplex, half duplex o full duplex. En un canal simplex los datos siempre viajarán en una dirección, por ejemplo desde DCE a DTE. En un canal half duplex, los datos pueden viajar en una u otra dirección, pero sólo durante un determinado periodo de tiempo; luego la línea debe ser conmutada antes que los datos puedan viajar en la otra dirección. En un canal full duplex, los datos pueden viajar en ambos sentidos simultáneamente. Las líneas de handshaking de la

RS-232 se usan para resolver los problemas asociados con este modo de operación, tal como en qué dirección los datos deben viajar en un instante determinado.

Si un dispositivo de los que están conectados a una interfaz RS-232 procesa los datos a una velocidad menor de la que los recibe deben de conectarse las líneas handshaking que permiten realizar un control de flujo tal que al dispositivo más lento le de tiempo de procesar la información. Las líneas de "hand shaking" que permiten hacer este control de flujo son las líneas RTS y CTS. Los diseñadores del estándar no concibieron estas líneas para que funcionen de este modo, pero dada su utilidad en cada interfaz posterior se incluye este modo de uso.

### 3.11.1 Los circuitos y sus definiciones

Las UART o U(S)ART (Transmisor y Receptor Asíncrono Universal) se diseñaron para convertir las señales que maneja la CPU y transmitirlas al exterior. Las UART deben resolver problemas tales como la conversión de tensiones internas del DCE con respecto al DTE, gobernar las señales de control, y realizar la transformación desde el bus de datos de señales en paralelo a serie y viceversa. Debe ser robusta y deberá tolerar circuitos abiertos, cortocircuitos y escritura simultánea sobre un mismo pin, entre otras consideraciones. Es en la UART en donde se implementa la interfaz.

Generalmente, cuando se requiere conectar un microcontrolador (con señales típicamente entre 3.3 y 5 V) con un puerto RS-232 estándar, se utiliza un driver de línea, típicamente un MAX232 o compatible, el cual mediante dobladores de tensión positivos y negativos, permite obtener la señal bipolar (típicamente alrededor de +/- 6V) requerida por el estándar.

Para los propósitos de la RS-232 estándar, una conexión es definida por un cable desde un dispositivo al otro. Hay 25 conexiones en la especificación completa, pero es muy probable que se encuentren menos de la mitad de éstas en una interfaz determinada. La causa es simple, una interfaz full duplex puede obtenerse con solamente 3 cables.

Existe una cierta confusión asociada a los nombres de las señales utilizadas, principalmente porque hay tres convenios diferentes de denominación (nombre común, nombre asignado por la EIA, y nombre asignado por el CCITT).

Las convenciones que se usan son las siguientes:

| Tensión | Señal | Nivel lógico | Control |
|---|---|---|---|
| +3 a +15 V | Espacio | 0 | On |
| -3 a -15 V | Marca | 1 | Off |

Los valores de tensión se invierten con respecto a los valores lógicos. Por ejemplo, el valor lógico positivo corresponde a la tensión negativa. También un 0 lógico corresponde a la señal de valor verdadero ó activada. Por ejemplo, si la línea DTR está al valor 0 lógico, se encuentra en la gama de tensión que va desde +3 a +15 V, entonces DTR está listo

El canal secundario a veces se usa para proveer un camino de retorno de información más lento, de unos 5 a 10 bits por segundo, para funciones como el envío de caracteres reloj, en principio sobre un canal half duplex. Si el módem usado acepta esta característica, es posible para el receptor aceptar o rechazar un mensaje sin tener que esperar el tiempo de conmutación, un proceso que usualmente toma entre 100 y 200 milisegundos.

### 3.11.2 Características eléctricas de cada circuito

Los siguientes criterios son los que se aplican a las características eléctricas de cada una de las líneas:

La magnitud de una tensión en circuito abierto no excederá los 25V.

El conductor será apto para soportar un corto con cualquier otra línea en el cable sin daño a sí mismo o a otro equipamiento, y la corriente de cortocircuito no excederá los 0,5A.

Las señales se considerarán en el estado de MARCA, (nivel lógico “1”), cuando la tensión sea más negativa que - 3V. Las señales se considerarán en el estado de ESPACIO, (nivel lógico ”0”), cuando la tensión sea más positiva que +3V. La gama de tensiones entre -3V y +3V se define como la región de transición, donde la condición de señal no está definida.

La impedancia de carga tendrá una resistencia a DC de menos de 7000Ω al medir con una tensión aplicado de entre 3 a 25 V pero mayor de 3000Ω cuando se mida con una tensión de menos de 25V..

Cuando la resistencia de carga del terminal encuentra los requerimientos de la regla 4 anteriormente dicha, y la tensión de la terminal de circuito abierto está a 0V, la magnitud del potencial de ese circuito estará en el rango de 5 a 15 V.

El driver de la interfaz mantendrá una tensión entre -5 a –15V para representar una condición de MARCA. El mismo driver mantendrá una tensión de entre 5 V a 15 V para simbolizar una señal de ESPACIO. Obsérvese que esta regla junto con la Regla 3, permite 2V de margen de ruido. En la práctica, se utilizan –12 y 12 V respectivamente.

El driver cambiará la tensión de salida hasta que no se excedan 30V/µs, pero el tiempo requerido a la señal para pasar de –3 V a +3 V de la región de transición no podrá exceder 1 ms, o el 4% del tiempo de un bit.

La desviación de capacitancia del terminal no excederá los 2500 pF, incluyendo la capacitancia del cable. Obsérvese que cuando se está usando un cable normal con una capacitancia de 40 a 50 pF/Pie de longitud, esto limita la longitud de cable a un máximo de 50 Pies, (15 m). Una capacitancia del cable inferior permitiría recorridos de cable más largos.

La impedancia del driver del circuito estando apagado deberá ser mayor que 300 Ω.

Existen en el mercado muchos circuitos integrados disponibles, (los chips 1488 y 1489, Max 232, etc) los cuales implementan drivers y receptores TTL, para una RS-232 de forma compatible con las reglas anteriores.

# 3.12 Codificación de datos

El código Morse fue el primero en utilizarse para las comunicaciones de larga distancia. Fue inventado por Samuel F. B. Morse en 1844. Este código está compuesto por puntos y guiones (una especie de código binario). Se usaba para realizar comunicaciones en forma mucho más rápida que Pony Express, el servicio de correo de Estados Unidos en ese entonces. El telegrafista, quien debía tener un perfecto conocimiento del código, era una figura clave en esa época.

El 10 de marzo de 1876, el Dr. Graham Bell creó el teléfono: un invento revolucionario para el envío de señales de voz a través de cables. Un hecho interesante es que la Cámara de Representantes recientemente resolvió nombrar a Antonio Meucci como el verdadero inventor del teléfono. De hecho, Meucci presentó una solicitud de patente en 1871, pero sólo la renovó hasta 1874.

Las líneas telegráficas dieron lugar a las teleimpresoras: máquinas que podían codificar y decodificar caracteres utilizando el código Baudot (para ese entonces, los caracteres se codificaban utilizando 5 bits y sólo se disponía de 32 caracteres).

En la década de 1960, se adoptó el código ASCII (American Standard Code for Information Interchange) como el nuevo estándar. Con ASCII, los caracteres se pueden codificar utilizando 8 bits y se obtienen 256 caracteres posibles.

## 3.12.1 Código ASCII

El código ASCII (siglas en ingles para American Standard Code for Information Interchange (Estándar para el intercambio de Información), es decir Código Americano.

Fue creado en 1963 por el Comité Estadounidense de Estándares o "ASA", este organismo cambio su nombre en 1969 por "Instituto Estadounidense de Estándares Nacionales" o "ANSI" como se lo conoce desde entonces.

Este código nació a partir de reordenar y expandir el conjunto de símbolos y caracteres ya utilizados en aquel momento en telegrafía por la compañía Bell. En un primer momento solo incluía letras mayúsculas y números, pero en 1967

se agregaron las letras minúsculas y algunos caracteres de control, formando así lo que se conoce como US-ASCII, es decir los caracteres del 0 al 127. Así con este conjunto de solo 128 caracteres fue publicado en 1967 como estándar, conteniendo todos lo necesario para escribir en idioma ingles.

En 1981, la empresa IBM desarrolló una extensión de 8 bits del código ASCII, llamada "pagina de código 437", en esta versión se reemplazaron algunos caracteres de control obsoletos, por caracteres gráficos. Además se incorporaron 128 caracteres nuevos, con símbolos, signos, gráficos adicionales y letras latinas, necesarias para la escrituras de textos en otros idiomas, como por ejemplo el español. Así fue como se sumaron los caracteres que van del ASCII 128 al 255.
IBM incluyó soporte a esta página de código en el hardware de su modelo 5150, conocido como "IBM-PC", considerada la primera computadora personal. El sistema operativo de este modelo, el "MS-DOS" también utilizaba el código ASCII extendido.
Casi todos los sistemas informáticos de la actualidad utilizan el código ASCII para representar caracteres, símbolos, signos y textos (279) .

### 3.14.1 Tabla código ASCII

**Caracteres ASCII de control**

| | | |
|---|---|---|
| 00 | NULL | (carácter nulo) |
| 01 | SOH | (inicio encabezado) |
| 02 | STX | (inicio texto) |
| 03 | ETX | (fin de texto) |
| 04 | EOT | (fin transmisión) |
| 05 | ENQ | (consulta) |
| 06 | ACK | (reconocimiento) |
| 07 | BEL | (timbre) |
| 08 | BS | (retroceso) |
| 09 | HT | (tab horizontal) |
| 10 | LF | (nueva línea) |
| 11 | VT | (tab vertical) |
| 12 | FF | (nueva página) |
| 13 | CR | (retorno de carro) |
| 14 | SO | (desplaza afuera) |
| 15 | SI | (desplaza adentro) |
| 16 | DLE | (esc.vínculo datos) |
| 17 | DC1 | (control disp. 1) |
| 18 | DC2 | (control disp. 2) |
| 19 | DC3 | (control disp. 3) |
| 20 | DC4 | (control disp. 4) |
| 21 | NAK | (conf. negativa) |
| 22 | SYN | (inactividad sínc) |
| 23 | ETB | (fin bloque trans) |
| 24 | CAN | (cancelar) |
| 25 | EM | (fin del medio) |
| 26 | SUB | (sustitución) |
| 27 | ESC | (escape) |
| 28 | FS | (sep. archivos) |
| 29 | GS | (sep. grupos) |
| 30 | RS | (sep. registros) |
| 31 | US | (sep. unidades) |
| 127 | DEL | (suprimir) |

**Caracteres ASCII imprimibles**

| | | | | | |
|---|---|---|---|---|---|
| 32 | espacio | 64 | @ | 96 | ` |
| 33 | ! | 65 | A | 97 | a |
| 34 | " | 66 | B | 98 | b |
| 35 | # | 67 | C | 99 | c |
| 36 | $ | 68 | D | 100 | d |
| 37 | % | 69 | E | 101 | e |
| 38 | & | 70 | F | 102 | f |
| 39 | ' | 71 | G | 103 | g |
| 40 | ( | 72 | H | 104 | h |
| 41 | ) | 73 | I | 105 | i |
| 42 | * | 74 | J | 106 | j |
| 43 | + | 75 | K | 107 | k |
| 44 | , | 76 | L | 108 | l |
| 45 | - | 77 | M | 109 | m |
| 46 | . | 78 | N | 110 | n |
| 47 | / | 79 | O | 111 | o |
| 48 | 0 | 80 | P | 112 | p |
| 49 | 1 | 81 | Q | 113 | q |
| 50 | 2 | 82 | R | 114 | r |
| 51 | 3 | 83 | S | 115 | s |
| 52 | 4 | 84 | T | 116 | t |
| 53 | 5 | 85 | U | 117 | u |
| 54 | 6 | 86 | V | 118 | v |
| 55 | 7 | 87 | W | 119 | w |
| 56 | 8 | 88 | X | 120 | x |
| 57 | 9 | 89 | Y | 121 | y |
| 58 | : | 90 | Z | 122 | z |
| 59 | ; | 91 | [ | 123 | { |
| 60 | < | 92 | \ | 124 | \| |
| 61 | = | 93 | ] | 125 | } |
| 62 | > | 94 | ^ | 126 | ~ |
| 63 | ? | 95 | _ | | |

**ASCII extendido (Página de código 437)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 128 | Ç | 160 | á | 192 | └ | 224 | Ó |
| 129 | ü | 161 | í | 193 | ┴ | 225 | ß |
| 130 | é | 162 | ó | 194 | ┬ | 226 | Ô |
| 131 | â | 163 | ú | 195 | ├ | 227 | Ò |
| 132 | ä | 164 | ñ | 196 | ─ | 228 | õ |
| 133 | à | 165 | Ñ | 197 | ┼ | 229 | Õ |
| 134 | å | 166 | ª | 198 | ã | 230 | µ |
| 135 | ç | 167 | º | 199 | Ã | 231 | þ |
| 136 | ê | 168 | ¿ | 200 | ╚ | 232 | Þ |
| 137 | ë | 169 | ® | 201 | ╔ | 233 | Ú |
| 138 | è | 170 | ¬ | 202 | ╩ | 234 | Û |
| 139 | ï | 171 | ½ | 203 | ╦ | 235 | Ù |
| 140 | î | 172 | ¼ | 204 | ╠ | 236 | ý |
| 141 | ì | 173 | ¡ | 205 | ═ | 237 | Ý |
| 142 | Ä | 174 | « | 206 | ╬ | 238 | ¯ |
| 143 | Å | 175 | » | 207 | ¤ | 239 | ´ |
| 144 | É | 176 | ░ | 208 | ð | 240 | ≡ |
| 145 | æ | 177 | ▒ | 209 | Ð | 241 | ± |
| 146 | Æ | 178 | ▓ | 210 | Ê | 242 | ‗ |
| 147 | ô | 179 | │ | 211 | Ë | 243 | ¾ |
| 148 | ö | 180 | ┤ | 212 | È | 244 | ¶ |
| 149 | ò | 181 | Á | 213 | ı | 245 | § |
| 150 | û | 182 | Â | 214 | Í | 246 | ÷ |
| 151 | ù | 183 | À | 215 | Î | 247 | ¸ |
| 152 | ÿ | 184 | © | 216 | Ï | 248 | ° |
| 153 | Ö | 185 | ╣ | 217 | ┘ | 249 | ¨ |
| 154 | Ü | 186 | ║ | 218 | ┌ | 250 | · |
| 155 | ø | 187 | ╗ | 219 | █ | 251 | ¹ |
| 156 | £ | 188 | ╝ | 220 | ▄ | 252 | ³ |
| 157 | Ø | 189 | ¢ | 221 | ¦ | 253 | ² |
| 158 | × | 190 | ¥ | 222 | Ì | 254 | ■ |
| 159 | ƒ | 191 | ┐ | 223 | ▀ | 255 | nbsp |

# 4. PROGRAMAS UTILIZADOS

A continuación se va a mostrar todos los programas utilizados para la realización de este proyecto.

**-OPENOFFICE:** Programa utilizado para la redacción de la memoria del proyecto. También se utilizará para la realización del presupuesto.

**-PL7 pro:** empleado para la programación de las variables a guardar del PLC.

**-Visio:** Programa empleado para la realización de los esquemas.

**-Suit Vijeo :** Programa utilizado para la implementación de la app. Vijeo de la casa Schneider electric también proporciona más herramientas a parte del desarrollo de HMI, como es el Vijeo IDS para base de datos o Vijeo Citect para SCADA.

## 4.1 Vijeo Designer

La programación del terminal táctil (HMI) se hace mediante el software Vijeo Designer de Schneider Electric.

Vijeo Designer se ejecutará en cualquier PC con Windows XP o Windows 7. Admite la simulación (se observa el resultado final de la programación) de la aplicación ampliada, sin necesidad de disponer de dicho terminal ..

Admite la producción de imágenes de vídeo. La oferta Magelis Vijeo Designer proporciona acceso a nuevos tipos de aplicaciones. El usuario puede visualizar el procesos en la pantalla HMI.

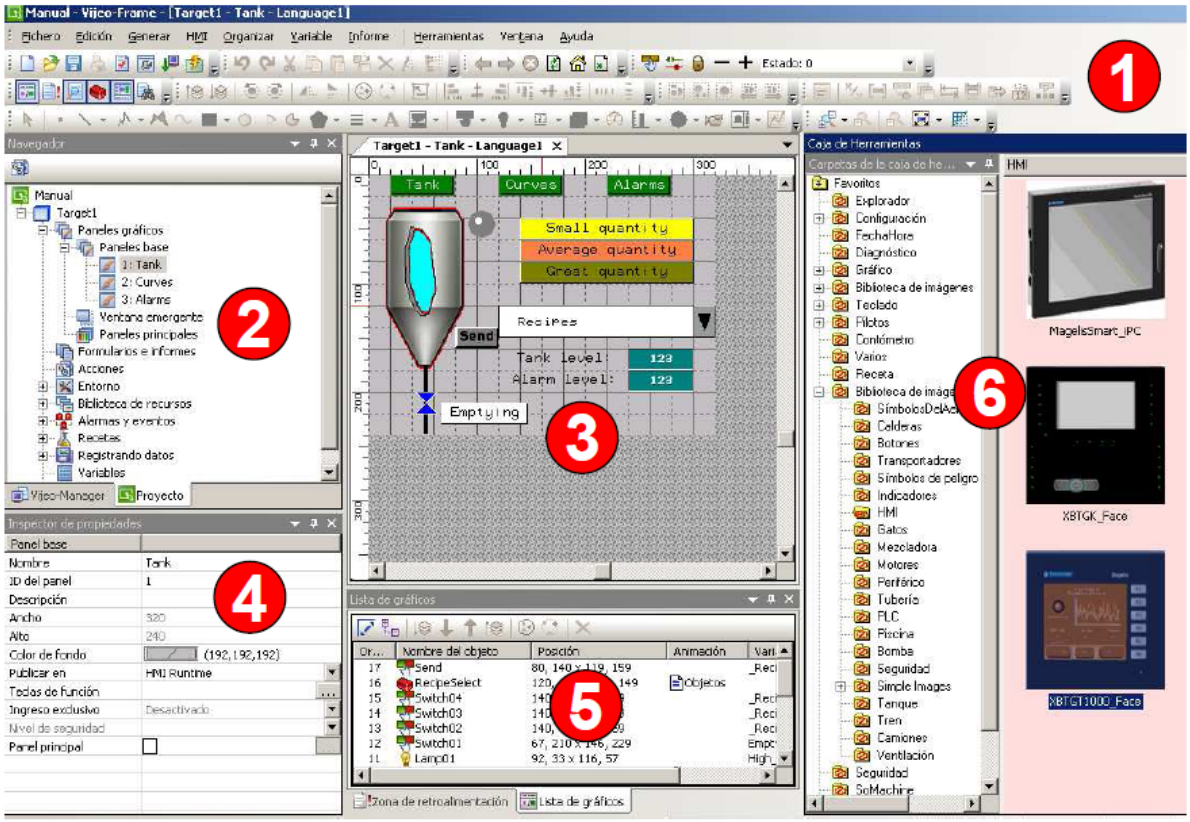
Vijeo Designer utiliza la conectividad Ethernet TCP/IP de Magelis y admite, por tanto, el acceso remoto WEB Gate, el uso compartido de datos de aplicaciones entre terminales, la transferencia de fórmulas y registros para variables, y mucho más, todo ello con una total seguridad.

Las aplicaciones pueden adoptar una naturaleza internacional, gracias a la capacidad de Vijeo Designer de admitir hasta 10 idiomas simultáneamente en un proyecto (38 alfabetos disponibles en el terminal XBT GT).El interface y la

documentación de Vijeo Designer se ofrecen en 6 idiomas: español, inglés, francés, alemán, italiano y chino simplificado.

### 4.1.1 Interface

El interface del software de Vijeo Designer tiene diferentes barras de herramientas y ventanas que ayudan a una mayor comprensión y orden a la hora de realizar la aplicación por el usuario.




**1. Barras de herramientas:** En esta área se muestra el menú contextual y las barras de iconos de las diferentes acciones que se pueden realizar en el proyecto.

**2. Navegador:** Esta área muestra el árbol de las diferentes partes que puede tener la aplicación.

**3. Área de Trabajo:** Es la zona donde el usuario tendrá que trabajar para crear la aplicación.

**4. Inspector de Propiedades:** En esta área se muestra las propiedades que se pueden parametrizar de los objetos seleccionados.

**5. Zona de retroalimentación/Lista de gráficos:** Según la pestaña seleccionada: En la zona de retroalimentación se verá los mensajes de compilación (errores de compilación y advertencias), en cambio en la pestaña de listado de gráficos muestra todos los objetos que hay en ese momento en el panel que hay en el área de trabajo.

**6. Caja de herramientas:** Podemos ver los diferentes objetos e imágenes que ya hay creados y guardados en la librería y que podemos incluir en nuestra aplicación y que nos ayudarán a realizarla.

### 4.1.2 Editor gráfico

El editor gráfico de Vijeo Designer ofrece un interface coherente para objetos simples así como para objetos más sofisticados. Permite a los desarrolladores de aplicaciones crear vistas basadas fácilmente en:

-Objetos simples para configurar:

-Puntos, líneas, rectángulos, elipses, arcos.

-Gráficos de barras, medidores, depósitos, rellenos, gráficos de sectores, curvas.

-Polilíneas, polígonos, polígonos regulares, curvas Bézier, escalas. □

-Textos, imágenes o resumen de alarmas, etc.

-Objetos avanzados preconfigurados: interruptores, botones de opción, indicadores, botones, depósitos, gráficos de barras, potenciómetros, selectores, campos de texto o número, listas enumeradas, etc.

### 4.1.3 Animaciones de objetos

La animación de 8 tipos de objetos gráficos permite la creación de vistas animadas basadas en:

-Pulsación del panel táctil.
-Cambio de color.
-Relleno.
-Movimiento.
-Rotación.

-Tamaño.
-Visibilidad.
-Visualización de valor asociado

### 4.1.4 Funciones avanzadas

Vijeo Designer, basado en las nuevas tecnologías de la información, ofrece un gran número de funciones avanzadas para el procesamiento de un volumen de datos mayor de una forma más rápida y fiable:

-Gestión de datos multimedia en los formatos más conocidos:

-Visualización de imágenes (archivos jpeg, bmp, emf, y png).

-Procesamiento y visualización de texto (archivos .txt).

-Procesamiento de mensajes de sonido (archivos .wav).

-Registros de alarmas o curvas grabados para la transferencia y el almacenamiento de datos.

-Gestión de alarmas. Todas las variables pueden categorizarse como “Alarmas” y pueden personalizarse con respecto a la visualización y el reconocimiento. Estas alarmas analógicas de tipo límite y booleano pueden imprimirse en tiempo real.

-Transferencia de aplicaciones multimodo: a través de un enlace serie Ethernet y una tarjeta de memoria Compact Flash (en terminales multifunción).

-Copia de seguridad de la aplicación en el terminal o iPC para facilitar el mantenimiento.

-Intercambio de datos sencillo entre el PC y el terminal con la herramienta Administrador de Datos.

-Servidor FTP integrado para descargar/cargar a través de Ethernet TCP/IP y restaurar registros en terminales XBT G/GT.

-Pueden activarse simultáneamente una comunicación de múltiples puertos para terminales multifunción, 2 enlaces serie y 1 red Ethernet.

### 4.1.5 Acciones

Vijeo Designer tiene preestablecidas una serie de acciones con el fin de facilitar el trabajo a los programadores.

Estas acciones pueden ser incluidas en los botones, como es el caso de este proyecto, o también nos da la posibilidad de lanzar esas acciones de las siguientes maneras:

**-Periódico:** Ejecuta la acción de forma periódica, según se haya configurado en la configuración del disparador.

-**Programado:** Ejecuta la acción según el programa en la configuración del disparador. Este tipo de disparador está disponible sólo para Acciones.

**-Condicional:** Ejecuta la acción, dependiendo de si se cumplen ciertas condiciones definidas en la configuración del disparador.

**-Evento:** Ejecuta una acción cuando se realiza un evento en el destino, según se haya configurado en los parámetros del disparador.

**-Alarma de la variable:** Ejecuta la acción cuando el estado de la alarma de la variable cambia

**-Grupo/Categoría de alarmas:** Ejecuta la acción cuando el número de alarmas activas del Grupo de alarmas o Categoría de alarmas cambia de 0 a vuelve a 0, según la configuración en la ajustes del disparador.

A continuación se enumera las diferentes acciones disponibles en el software de programación:

**-Bit**: permite modificar el valor de un bit único de datos

**-Palabra:** escribe un valor constante o un valor o una variable al entero o al flotante

**-Cadena:** permite escribir una cadena o añadir una variable a la cadena

**-Panel:** Permite el cambio de un panel a otro.

**-Emergente** (ventana emergente): permite tanto abrir como cerrar ventanas emergentes.

**-Alarma**: Lanza la señal de alarma preestablecida.

**-Sonido:** Permite lanzar un sonido cuando se acciona.

**-Idioma:** Permite desarrollar la app en más de un idioma.

**-Script:** Permite al programador desarrollar el código utilizando el lenguaje de programación Java.

**-Sistema:** Realiza funciones del sistema como puede ser el reinicio del terminal, configuración, registro de eventos.

**-Retraso**: Crea un retraso para la ejecución de la siguiente acción.

**-Video**: Lanza el video seleccionado.

**-Información:** Escribe un valor solicitado de información del sistema en una variable cadena.

# 5. MANUAL DEL USUARIO

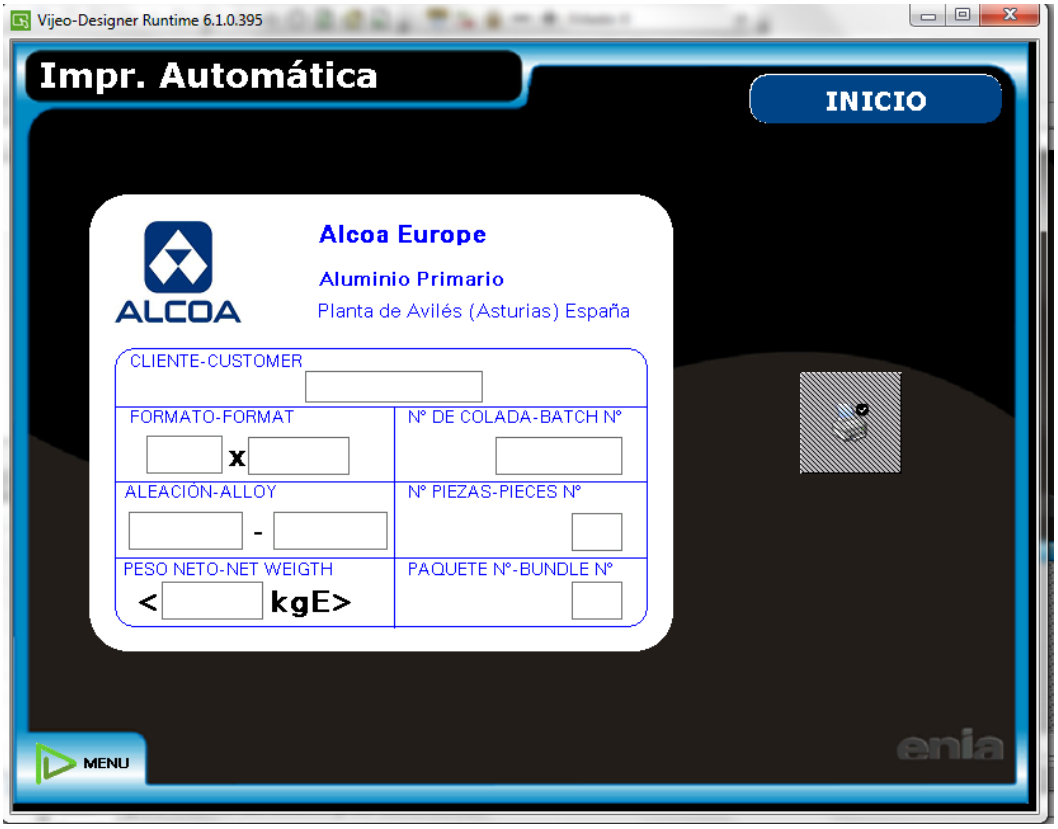
Este manual del usuario está basado en el conocimiento de la aplicación anterior, en la cual se han actualizado alguna de las pantallas existentes y se han desarrollado otras nuevas:

## 5.1 Inicio

El menú de inicio es igual que en el pasado, añadiendo los botones de base de datos, buscador e impresión manual y automática, para el cumplimiento de las nuevas especificaciones requeridas. Para acceder a estos nuevos submenús bastará con presionar encima del interruptor deseado.

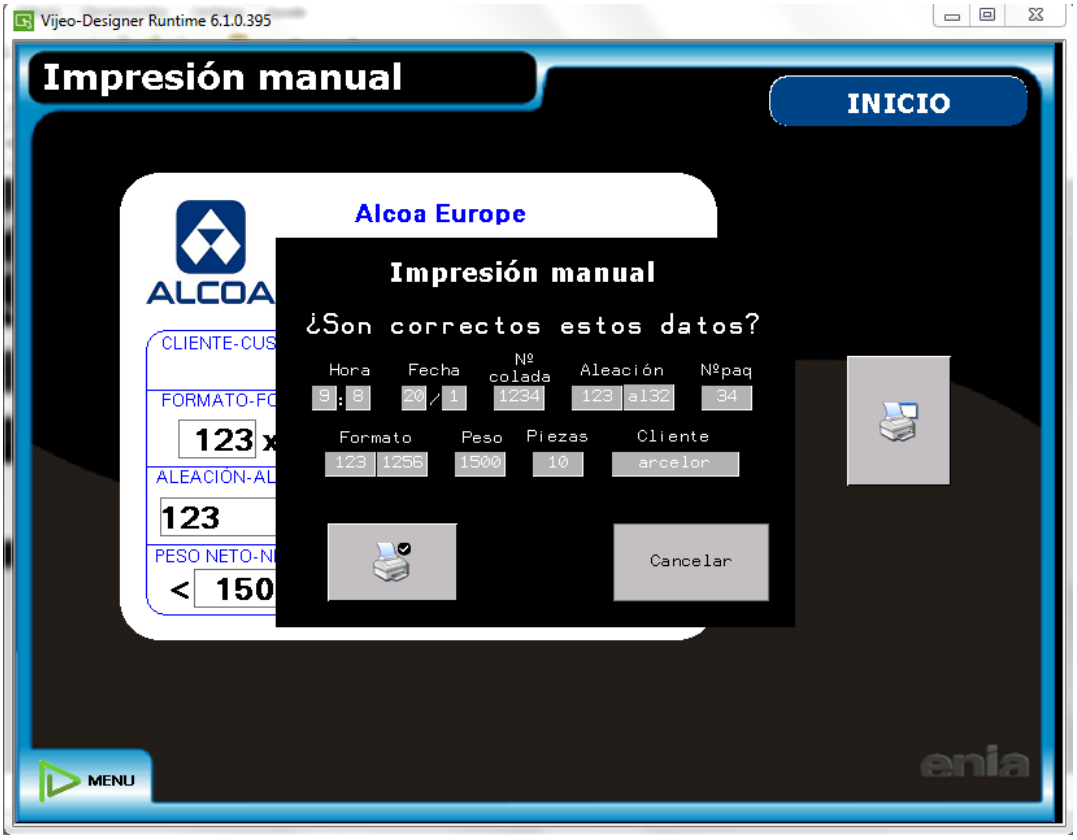
## 5.2 Impresión automática

El manejo de este submenú de la app es igual que en el pasado, siendo todos los campos autocompletados y disponiendo solo de dos botones, el de inicio, situado en la parte superior de la pantalla, cuya función es la de regresar al menú principal y el botón de imprimir, situado a la derecha de la pantalla, cuya función es continuar con el proceso de impresión y captura de datos. Una vez presionado dicho botón, la aplicación mostrará la ventana emergente para la comprobación de los datos.

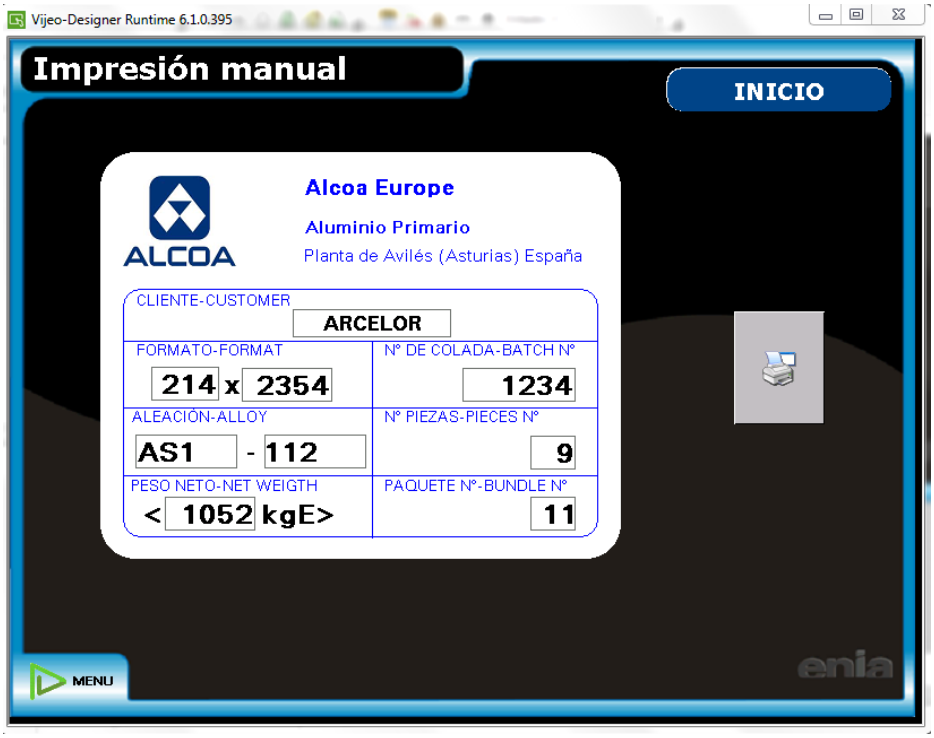


### 5.2.1 Pantalla emergente 1 Comprobación de datos

Pantalla emergente diseñada para la verificación de los datos que se van a imprimir y guardar. En caso de que los datos sean correctos, presionamos el botón con una impresora dibujada lo que permite que el proceso de impresión culmine. La app mostrará el mensaje “impresión en curso” que automáticamente se cerrará con el transcurso de unos segundos. En el caso de que se detecte algún error en alguno de los campos, presionamos el botón cancelar, el cual nos situará en la ventana impresión manual para que el usuario complete manualmente los campos.



# 5.3 Impresión manual




En aspecto y funcionamiento, esta pantalla de la app es similar a la del punto anterior, “impresión automática”. La diferencia reside en que, en la impresión automática, los campos se auto rellenan y en la impresión manual es el usuario el que rellena los campos. Para rellenar dichos campos se dispone de un teclado virtual que aparecerá en la pantalla al presionar sobre el campo a rellenar. Dependiendo del campo seleccionado el teclado que aparecerá será numérico o alfanumérico. Una vez rellenado los campos, podremos continuar con el proceso de imprimir y guardar los datos presionando el botón con una impresora dibujada, situado en la parte derecha de la pantalla. Una vez accionado dicho botón pueden aparecer dos pantallas diferentes:

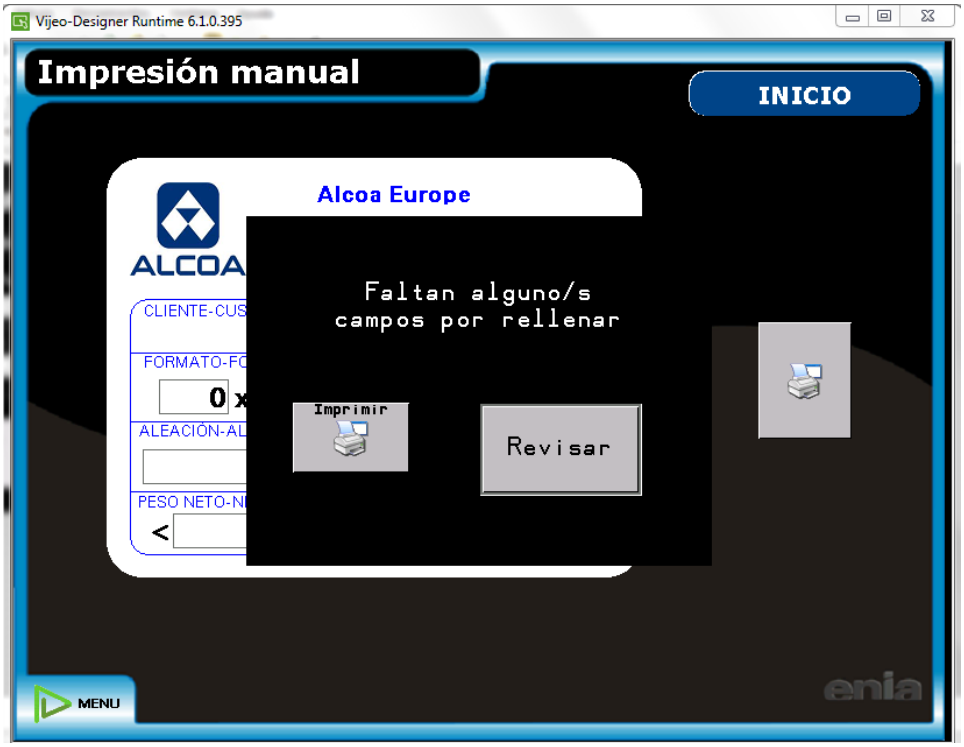
-Falta algún campo por rellenar

-Comprobación de los datos

### 5.3.1 Pantalla emergente 2: Falta algún campo por rellenar

En caso de que quede algún campo sin completar o mal completado (se necesiten más caracteres de los escritos), aparecerá una ventana emergente informándonos del suceso, donde podremos ignorara el aviso (botón imprimir) y continuar con el proceso de impresión o volver a revisar el formulario (botón revisar), regresando a la  pantalla de impresión manual.

En el caso de que continuemos con el proceso, el sistema mostrará la pantalla emergente 1, comprobación de datos, anteriormente descrita.



## 5.4 Base de datos


Pantalla desarrollada para mostrar, modificar o borrar los datos guardados e impresos por la app. En esta pantalla se puede observar ordenados por filas, los últimos datos impresos y cada columna representa un campo en concreto.

En esta pantalla de la aplicación se pueden distinguir una serie de botones. Hay que distinguir los botones para desplazarse por la lista de datos y los botones de operación:

**Botones para desplazarse por la lista de datos:**

 **Subir**: botón encargado de subir el listado de uno en uno

 **Subir rápido:** botón encargado de subir el listado en 9 unidades y así mostrar datos nuevos en todos sus visualizadores.

 **Bajar**: botón encargado de bajar el listado de uno en uno.

 **Bajar rápido**: botón encargado de bajar el listado en 9 unidades y así mostrar datos nuevos en todos sus visualizadores.

 **Seleccionar:** botón encargado de seleccionar el dato a tratar.

**Botones de operación:**

 **Eliminar**: Botón encargado de eliminar los datos de la etiqueta seleccionada de la app. Para realizar esta operación debemos seleccionar cual es la etiqueta que deseamos eliminar, presionando sobre una de las flechas situadas en la parte izquierda de la pantalla y posteriormente presionar el botón de eliminar. En caso de que no se realice ninguna selección el programa nos informará mediante una ventana emergente con el texto" realiza una selección correcta" que automáticamente se cerrará. Si se ha realizado una selección y para culminar el proceso de eliminación de datos, el sistema nos mostrará, mediante una pantalla emergente, los datos a eliminar. Si estos datos son correctos, culminaremos el proceso presionando el icono de la papelera. En caso de que se detecte algún error, el usuario podrá cancelar el proceso de eliminación presionando el botón cancelar.

Nota: la eliminación de los datos solo es para el listado interno de la app.


**Modificar**: Botón encargado de modificar los datos de la etiqueta seleccionada de la app. Para ello, e igual que en el caso anterior, debemos seleccionar cual es la etiqueta que deseamos modificar, empleando las flechas situadas en la parte izquierda de la pantalla y posteriormente presionar el botón de modificar.

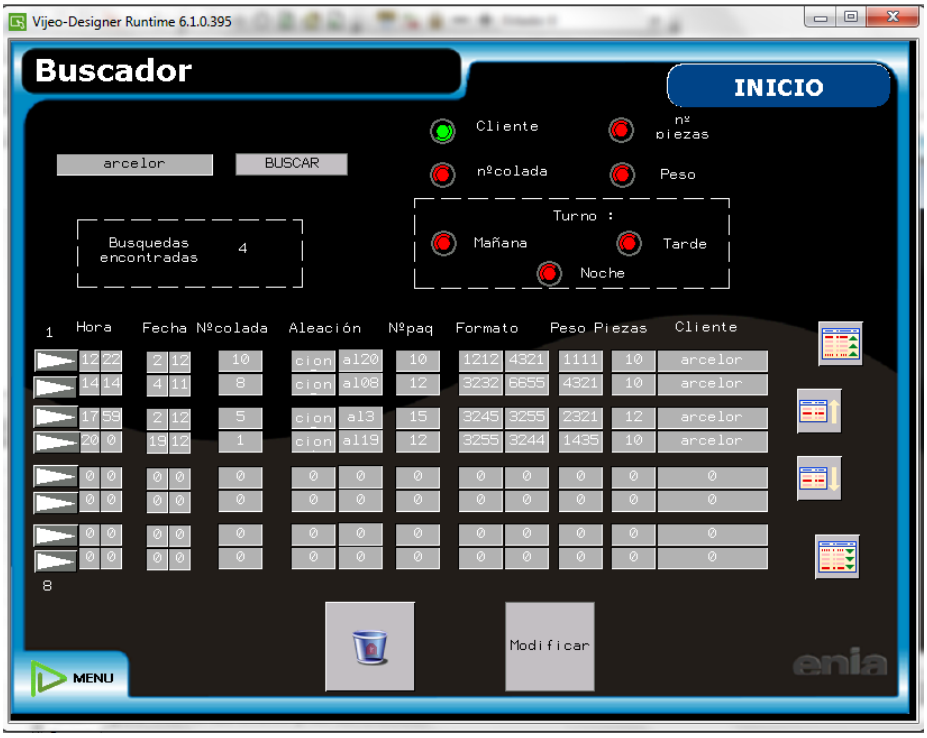
Nota: la modificación no sustituye la antigua etiqueta.

**Inicio**: al presionar este botón la app regresará a la pantalla principal.

## 5.5 Buscador

Pantalla desarrollada para que el usuario pueda buscar, borrar y modificar los datos generados. Para este fin, el usuario dispone de un buscador donde podrá escribir datos alfanuméricos y de una serie de selectores para que el usuario pueda seleccionar el campo donde desee realizar la búsqueda. También se ha incluido un buscador por turno, el cual muestra las etiquetas generadas en los intervalos horarios de cada turno.

En esta pantalla vamos a distinguir 3 partes: Buscador, visualizador y botones de acción.



### 5.5.1 Buscador

Para realizar una búsqueda basta con que el usuario escriba, mediante el teclado emergente que aparecerá al presionar en el campo buscador, los datos alfanuméricos que desee buscar. A continuación, el usuario debe escoger, mediante los selectores ubicados arriba a la derecha de la pantalla, el campo donde quiere realizar la búsqueda y por último pulse el botón “BUSCAR”. El sistema mostrará las coincidencias en un listado ordenado cronológicamente e informará del número de coincidencias encontradas.

En el caso de que el usuario desee buscar por turno, bastará con que presione, mediante el selector ubicado arriba a la derecha de la pantalla, el turno que quiere ver. Para culminar la búsqueda, presionamos el botón “BUSCAR”. El sistema mostrará las etiquetas encontradas ordenadas cronológicamente.

También, se ha desarrollado una serie de búsquedas especiales para que en el caso de que el usuario no recordase el nombre o dato concreto de la búsqueda. A continuación se explican los tipos de búsquedas:

**-abcd**: En caso de que se rellene el campo del buscador sin “*”, el sistema realizará una búsqueda literal. El sistema busca que coincida el dato o texto íntegramente, sin distinguir entre mayúsculas y minúsculas.

**-*abcd:** En caso de que se rellene el campo del buscador con un “*” al comienzo, el sistema buscará los datos que terminen con dicho texto o valor. El sistema busca dicha terminación sin distinguir entre mayúsculas y minúscula.

**-abcd*:** En caso de que se rellene el campo del buscador con un “*” al final, el sistema buscará los datos que comiencen con dicho texto o valor. El sistema busca dicho comienzo sin distinguir entre mayúsculas y minúscula.

**-*Abcd*:** En caso de que se rellene el campo del buscador con dos “*” uno a cada extremo, el sistema buscará los datos que contengan con dicho texto o valor, independientemente de donde se encuentre este, ya sea al principio final o en medio. El sistema busca dicho comienzo sin distinguir entre mayúsculas y minúscula.

### 5.5.2 Botones para desplazarse por la lista de datos:

Situado en la zona media baja de la pantalla, es el encargado de desplazarnos por el visualizador de datos buscados. Su manejo es igual que en el de la pantalla base de datos, teniendo una serie de botones para visualizar el total de los datos.

 **Subir**: botón encargado de subir el listado de uno en uno.

 **Subir rápido**: botón encargado de subir el listado en 9 unidades y así mostrar datos nuevos en todos sus visualizadores.

 **Bajar:** botón encargado de bajar el listado de uno en uno.

 **Bajar rápido:** botón encargado de bajar el listado en 9 unidades y así mostrar datos nuevos en todos sus visualizadores.

 **Seleccionar:** botón encargado de seleccionar el dato a tratar.

### Acción

En la parte inferior de esta pantalla el usuario dispone dos botones, al igual que en la pantalla de “base de datos”, para realizar dos acciones diferentes: borrar y modificar

**Modificar**: Botón encargado de modificar los datos de la etiqueta seleccionada de la app. Para ello, e igual que en el caso anterior, debemos seleccionar cual es la etiqueta que deseamos modificar, empleando las flechas situadas en la parte izquierda de la pantalla y posteriormente presionar el botón de modificar. A continuación, el sistema se posiciona en la pantalla “Impresión manual”, con los datos de la etiqueta seleccionada para que el usuario los modifique Este proceso está descrito en el manual del usuario, en el apartado “impresión manual”.

Nota: la modificación no sustituye la antigua etiqueta.

**Inicio**: al presionar este botón la app regresará a la pantalla principal.

**Eliminar**: Botón encargado de eliminar los datos de la etiqueta seleccionada de la app. Para realizar esta operación debemos seleccionar cual es la etiqueta que deseamos eliminar, empleando las flechas situadas en la parte izquierda de la pantalla y posteriormente presionar el botón de eliminar. En caso de que no se realice ninguna selección, el programa nos informará mediante una ventana emergente con el texto “Realice una selección correcta”, que automáticamente se cerrará. Si se ha realizado una selección correcta y para culminar el proceso de eliminación de datos, el sistema nos mostrará, mediante una pantalla emergente, los datos que vamos a eliminar. Si estos datos son correctos, culminaremos el proceso presionando el icono de la papelera. En caso de que se detecte algún error, el usuario podrá cancelar el proceso de borrado presionando el botón “Cancelar”.

Nota: la eliminación de los datos solo es para el listado interno de la app.

# 6. MANUAL DEL PROGRAMADOR

Esta app se ha desarrollado en base a la app ya existente, añadiendo los puntos necesarios para la visualización, modificación e impresión de los datos sin necesidad de utilizar un PC.

La programación de esta app se ha llevado a cabo en una pantalla HMI, modelo XBT GT 6330.

## 6.1 Configuración

A continuación se va a comentar los parámetros introducidos para el control del periférico. Para realizar dicha configuración debemos modificar los parámetros de la impresora, utilizando los cursores ubicados en la misma y si fuese necesario con la ayuda del manual del usuario de la impresora, incluido en los archivos del proyecto. Los parámetros son los siguientes:

| | |
| --- | --- |
| -Puerto del HMI: | COM1 |
| Interfaz serie | RS-232 |
| -Velocidad de transmisión | 38400 bps |
| -Bit de paridad | Par |
| -Bit de parada | 1 |
| -Longitud de los datos | 8bits |

# 6.2 Listado de variables

A continuación se procede a explicar las variables empleadas en la app. Para mayor comprensión del documento se ha decidido separar las variables en función de su finalidad.

### 6.2.1 Base de datos

Variables donde se guarda el valor de las etiquetas impresas y que realizan la función de base de datos:

| **Base de datos** | | |
|---|---|---|
| **Nombre** | **tipo** | **Explicación** |
| bd_aleación1 | String[50] | Contiene el valor 1 del tipo de aleación de las etiquetas impresas |
| bd_aleación2 | String[50] | Contiene el valor 2 del tipo de aleación de las etiquetas impresas |
| bd_cliente | String[50] | Contiene los clientes de las etiquetas impresas |
| bd_formato1 | String[50] | Contiene el valor 1 del formato de las etiquetas impresas |
| bd_formato2 | String[50] | Contiene el valor 2 del formato de las etiquetas impresas |
| bd_num_colada | String[50] | Contiene el número de colada de las etiquetas impresas |
| bd_num_paquete | String[50] | Contiene el número de paquete de las etiquetas impresas |
| bd_num_piezas | String[50] | Contiene el número de piezas de las etiquetas impresas |
| bd_hora | Int[50] | Contiene la hora de las etiquetas impresas |
| bd_minuto | Int[50] | Contiene los minutos de las etiquetas impresas |
| bd_dia | Int[50] | Contiene el día de las etiquetas impresas |
| bd_mes | Int[50] | Contiene el número de mes de las etiquetas impresas |

### 6.2.2 Impresión manual

Variables encargadas de la recogida de datos de la pantalla “Etiqueta manual”

| **Impresión manual man_etiq** | | |
|---|---|---|
| **Nombre** | **tipo** | **Explicación** |
| man_etiq_aleación1 | String | Contiene el valor 1 del tipo de aleación de las etiquetas impresas |
| man_etiq_aleación2 | String | Contiene el valor 2 del tipo de aleación de las etiquetas impresas |
| man_etiq_cliente | String | Contiene los clientes de las etiquetas impresas |
| man_etiq_formato1 | String | Contiene el valor 1 del fortmato de las etiquetas impresas |

| | | |
|---|---|---|
| man_etiq_formato2 | String | Contiene el valor 2 del formato de las etiquetas impresas |
| man_etiq_num_colada | String | Contiene el número de colada de las etiquetas impresas |
| man_etiq_num_paquete | String | Contiene el número de paquete de las etiquetas impresas |
| man_etiq_num_piezas | String | Contiene el número de piezas de las etiquetas impresas |

### 6.2.3 Cursores

Variables encargadas del desplazamiento y selección de los valores de las listas de datos, ya sea en la pantalla "buscador", como en la pantalla " Base de datos".

| **Selectores de desplazamiento de las listas de datos** | | |
|---|---|---|
| **nombre** | **tipo** | **Explicación** |
| sel_0 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 0. |
| sel_1 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 1. |
| sel_2 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 2. |
| sel_3 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 3. |
| sel_4 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 4. |
| sel_5 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 5. |
| sel_6 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 6. |
| sel_7 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 7. |
| sel_8 | BOOL | Variable encargada de seleccionar la etiquetas deseada de la posición 8. |
| pos_vect_0 | Int | Variable encargada de desplazar los datos de la posición 0. Esta variable se modifica con los cursores ubicados en las listas de datos |
| pos_vect_1 | Int | Variable encargada de desplazar los datos de la posición 1. Esta variable se modifica con los cursores ubicados en las listas de datos |
| pos_vect_2 | Int | Variable encargada de desplazar los datos de la posición 2. Esta variable se modifica con los cursores ubicados en las listas de datos |
| pos_vect_3 | Int | Variable encargada de desplazar los datos de la posición 3. Esta variable se modifica con los cursores ubicados en las listas de datos |
| pos_vect_4 | Int | Variable encargada de desplazar los datos de la posición 4. Esta variable se modifica con los cursores ubicados en las listas |

| | | |
|---|---|---|
| pos_vect_5 | Int | Variable encargada de desplazar los datos de la posición 5. Esta variable se modifica con los cursores ubicados en las listas de datos |
| pos_vect_6 | Int | Variable encargada de desplazar los datos de la posición 6. Esta variable se modifica con los cursores ubicados en las listas de datos |
| pos_vect_7 | Int | Variable encargada de desplazar los datos de la posición 7. Esta variable se modifica con los cursores ubicados en las listas de datos |
| pos_vect_8 | Int | Variable encargada de desplazar los datos de la posición 8. Esta variable se modifica con los cursores ubicados en las listas de datos |

### 6.2.4 Buscador

Variables utilizadas en la pantalla “Buscador” de la app.

| **buscador** | | |
|---|---|---|
| **nombre** | **tipo** | **Explicación** |
| busc_aleación1 | String[50] | Contiene el valor 1 del tipo de aleación de las etiquetas buscadas. |
| busc_aleación2 | String[50] | Contiene el valor 2 del tipo de aleación de las etiquetas buscadas. |
| busc_cliente | String[50] | Contiene los clientes de las etiquetas buscadas |
| busc_formato1 | String[50] | Contiene el valor 1 del formato de las etiquetas buscadas |
| busc_formato2 | String[50] | Contiene el valor 2 del formato de las etiquetas buscadas |
| busc_num_colada | String[50] | Contiene el número de colada de las etiquetas buscadas |
| busc_num_paquete | String[50] | Contiene el número de paquete de las etiquetas buscadas |
| busc_num_piezas | String[50] | Contiene el número de piezas de las etiquetas buscadas |
| busc_posicion | String[50] | Contiene la posición donde se encuentra en la base de datos de los datos buscados |
| busc_encontradas | Int | Contiene el valor de las coincidencias encontradas en el buscador |
| dato_buscado | String[50] | Copia de la base de datos del campo indicado en el buscador |
| busc_tipo_1 | BOOL | Variable relacionada con el selector para escoger la búsqueda cliente |
| busc_tipo_2 | BOOL | Variable relacionada con el selector para escoger la búsqueda número de colada |
| busc_tipo_3 | BOOL | Variable relacionada con el selector para escoger la búsqueda número de piezas |
| busc_tipo_4 | BOOL | Variable relacionada con el selector para escoger la búsqueda peso |
| busc_turno_manana | BOOL | Variable relacionada con el selector para escoger la |

| | | |
|---|---|---|
| | | búsqueda turno de mañana |
| busc_turno_tarde | BOOL | Variable relacionada con el selector para escoger la búsqueda turno tarde |
| busc_turno_noche | BOOL | Variable relacionada con el selector para escoger la búsqueda turno noche |

### 6.2.5 Variables compartidas

Variables encargadas de mandar la información al PLC para su posterior extracción por parte del SCADA y la base de datos. Estas variables son de carácter global por lo que tenemos que indicar la posición de memoria

| | | **Comp_ :compartir con el servidor** | |
|---|---|---|---|
| **Nombre** | **Tipo** | | **Explicación** |
| comp_aleación1 | String | %M1065 | Contiene el último valor del tipo de aleación 1de las etiquetas impresas para el envío al PLC |
| comp_aleación2 | String | %M1066 | Contiene el último valor del tipo de aleación 2de las etiquetas impresas para el envío al PLC |
| comp_cliente | String | %M1067 | Contiene el último dato de clientes de las etiquetas impresas para el envío al PLC |
| comp_formato1 | String | %M1068 | Contiene el último valor del formato 1de las etiquetas impresas para el envío al PLC |
| comp_formato2 | String | %M1069 | Contiene el último valor del formato2 de las etiquetas impresas para el envío al PLC |
| comp_num_colada | String | %M1070 | Contiene el último número de colada de las etiquetas impresas para el envío al PLC |
| comp_num_paquete | String | %M1071 | Contiene el último número de paquete de las etiquetas impresas para el envío al PLC |
| comp _num_piezas | String | %M1072 | Contiene el último número de piezas de las etiquetas impresas para el envío al PLC |
| comp_peso | String | %M1073 | Contiene el último valor del peso para el envío al PLC |
| comp_hora | DInt | %MD607 | Contiene la último hora de las etiquetas impresas para el envío al PLC |
| comp_minuto | DInt | %MD608 | Contiene los último minutos de las etiquetas impresas para el envío al PLC |
| comp_dia | DInt | %MD609 | Contiene el último día de las etiquetas impresas para el envío al PLC |
| comp_mes | DInt | %MD610 | Contiene el último número de mes de las etiquetas impresas para el envío al PLC |
| comp_new | BOOL | %M1121 | Esta variable realiza un flanco ascendente cuando se genera una serie de datos nuevos |

# 6.3 Programación

Este documento está desarrollado con el fin de ayudar a terceras personas a la modificación de la app desarrollada. El proyecto se ha desarrollado en el Software Vijeo Designer, el cual incluye a parte de la programación visual, un editor de código Java para la programación de funciones especificas. Con el fin de facilitar la comprensión, explicaremos cada pantalla y cada acción de programación, por separado, para su mayor comprensión.

**Nota:** Para mayor comprensión, los script donde se ha incluido la programación están incluidos en el *Anexo I del presente documento.*

A continuación se muestran las pantallas que conforman la app y su programación:

## 6.3.1 Inicio

Se han añadido una serie de nuevos botones, en los cuales, se han incluido las siguientes acciones y programación:


**Acciones**:
-cambio de panel(imp_automático)


**Acciones**:
-cambio de panel(imp_manual)


**Acciones**:
-Script(1)
-Cambio de panel(buscador)


**Acciones**:
-Script(1)
-Cambio de panel(base datos)

### 6.3.2 Impresión automática

Esta pantalla se ha modificado la programación ya existente. La nueva programación es la siguiente:


**Acciones**:

-Abrir pantalla emergente centrado(imp_auto)

#### *6.3.2.1 Pantalla emergente(impr_auto)*

Se han añadido dos botones, en los cuales, se han incluido las siguientes acciones y programación:


**Acciones**:

-Cerrar pantalla emergente (imp_auto)

**Acciones**:

-Script(2)
-Script(3)
-Script(19)
-Set bit(comp_new)
-Cerrar emergente(impr_auto)
-Abrir emergente(imprimiendo)
-Retraso(2s)
-Cerrar emergente(imprimiendo)
-Script(4)
-Reset bit(comp_new)
-Cambio panel(inicio)

### 6.3.3 Impresión manual

Se han añadido una serie de nuevos botones, en los cuales, se han incluido las siguientes acciones y programación:


**Acciones**:

Script(5)
Decisión( si campos rellenos => -Abrir emergente(imp_manual)
no campos rellenos=>-Abrir emergente(faltan_campos))

#### *-6.3.3.1 Pantalla emergente(imp_manual)*

**Acciones**:

-Script(2)
-Script(6)
-Script(19)
-Set bit(comp_new)
-Cerrar pantalla emergente(impr_manual)
-Abrir pantalla emergente(imprimiendo)
-Retraso(2s)
-Script(4)
-Reset bit(comp_new)
-Cambiar panel(inicio)


**Acciones**:

-Cerrar emergente(impr_manual)

#### *-6.3.3.2 Pantalla emergente(faltan_campos)*


**Acciones**:

-Abrir pantalla emergente(imp_manual)


**Acciones**:
-Cerrar pantalla emergente(faltan_campos)
-Cambiar panel(impresión_manual)

### 6.3.4 Base de datos

Se han añadido una serie de nuevos botones, en los cuales, se han incluido las siguientes acciones y programación:

**Acciones**:
-Script(7)

**Acciones**:
-Script(8)

**Acciones**:
-Script(9)

**Acciones**:
-Script(10)

**Acciones**:
-Decisión(si hay_seleccion=> -Script(11)
-Abrir pantalla emergente(borrado)

si no hay_seleccion=>
-Abrir pantalla emergente(seleccion)
-Retraso(2s)
-Cerrar pantalla emergente(selecion)


**Acciones**:

-Decisión(si hay_seleccion=> -Script(12)
-Abrir panel(impresión_manual)

si no hay_seleccion=>
-Abrir pantalla emergente(selecion)
-Retraso(2s)
-Cerrar pantalla emergente(seleccion)

### 6.3.5 Buscador

Se han añadido una serie de nuevos botones, en los cuales, se han incluido las siguientes acciones y programación:

**Acciones**:

-Script(7)

**Acciones**:

-Script(8)

**Acciones**:

-Script(13)

**Acciones**:

-Script(14)

**Acciones**:

-Decisión(si hay_seleccion=> -Script(11)
-Abrir pantalla emergente(borrado)

si no hay_seleccion=> -Abrir pantalla emergente(seleccion)
-Retraso(2s)
-Cerrar pantalla emergente(seleccion)


**Acciones**:
-Decisión(si hay_seleccion=> -Script(12)
-Abrir panel(impresión_manual)
si no hay_seleccion=>
-Abrir pantalla emergente(seleccion)
-Retraso(2s)
-Cerrar pantalla emergente(seleccion)


**Acciones**:
-Script(15)
-Script(16)
-Script(17)

### *6.3.5.1 Pantalla emergente(borrado)*

**Acciones**:
-Script(18)
-Cerrar pantalla emergente(borrado)
-Abrir pantalla emergente(eliminados)
-Retraso(2s)
-Cambiar panel(inicio)


**Acciones**:
-Cerrar pantalla emergente(borrado)

# 7. Planificación

A continuación se muestra la planificación por etapas realizada para este proyecto.

**ETAPA 1: DOCUMENTACIÓN**

La primera etapa consistió en la documentación acerca de los componentes que forman el sistema, así como, los protocolos de comunicación que emplean

**ETAPA 2: DISEÑO**

Durante este período se diseñó la comunicación del sistema y donde se iba a desarrollar el proyecto

**ETAPA 3: DESARROLLO DEL CÓDIGO FUENTE**

Se implementó el código fuente par el manejo del sistema

**ETAPA 4: REDACCIÓN DE LOS DOCUMENTOS**

Una vez finalizada la etapa de documentación y casi terminado el diseño se comenzó con la redacción de todos los documentos necesarios.

## 7.1 Diagrama de Gantt

| Mes | Septiembre | | | | Octubre | | | | Noviembre | | | | Diciembre | | | | Enero | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Semana | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| Documentación | | | | | | | | | | | | | | | | | | | | |
| Diseño | | | | | | | | | | | | | | | | | | | | |
| Desarrollo | | | | | | | | | | | | | | | | | | | | |
| Memoria | | | | | | | | | | | | | | | | | | | | |

# 8. Bibliografía

A continuación se enumera el material de ayuda utilizado para el desarrollo de este proyecto:

http://www.schneider.com/
Fabricante del PLC y HMI

http://www.ate.uniovi.es
Información general del Área de Tecnología Electrónica

Documentación de la asignatura: Oficina técnica.
Información de la estructura de un proyecto

Documentación de la asignatura: Integración de sistemas
Información de las redes de comunicación

www.toshiba.es
http://www.toshibatec-ris.com
Información de la impresora

es.wikipedia.org
Información general

https://juncalcepeda.wordpress.com/
Manual de programación de Java

http://lineadecodigo.com
Manual de programación de Java

http://www.automatas.org
Foro de PLC

http://todoimpresora.blogspot.com
Blog referido a los protocolos de las impresoras

# ANEXO I: CÓDIGO FUENTE DEL PROGRAMA

## Script(1)

```
//vuelve a reiniciar los índices de los vectores de las listas
        pos_vect_0.write(0);
        pos_vect_1.write(1);
        pos_vect_2.write(2);
        pos_vect_3.write(3);
        pos_vect_4.write(4);
        pos_vect_5.write(5);
        pos_vect_6.write(6);
        pos_vect_7.write(7);
        pos_vect_8.write(8);
```

## Script(2 )

```
//script desarrollado para desplazar los diferentes datos de los vectores
        int j= 0;
        String aux;
        int aux_int;

        for (j=48; j>=0; j--)        //Bucle que desplaza los datos para hacer sitio al nuevo dato
        {
        aux_int=bd_hora[j].getIntValue();
        bd_hora[j+1].write(aux_int);

        aux_int=bd_minuto[j].getIntValue();
        bd_minuto[j+1].write(aux_int);

        aux_int=bd_mes[j].getIntValue();
        bd_mes[j+1].write(aux_int);

        aux_int=bd_dia[j].getIntValue();
        bd_dia[j+1].write(aux_int);

        aux=bd_num_paquete[j].getStringValue();
        bd_num_paquete[j+1].write(aux);

        aux=bd_num_colada[j].getStringValue();
        bd_num_colada[j+1].write(aux);


        aux=bd_formato1[j].getStringValue();
        bd_formato1[j+1].write(aux);

        aux=bd_formato2[j].getStringValue();
        bd_formato2[j+1].write(aux);

        aux=bd_num_piezas[j].getStringValue();
```

```
bd_num_piezas[j+1].write(aux);

aux=bd_cliente[j].getStringValue();
bd_cliente[j+1].write(aux);

aux=bd_peso[j].getStringValue();
bd_peso[j+1].write(aux);

aux=bd_aleacion1[j].getStringValue();
bd_aleacion1[j+1].write(aux);

aux=bd_aleacion2[j].getStringValue();
bd_aleacion2[j+1].write(aux);}
```

## *Script(3)*

```
//Captura de datos. Copia de lo que se muestra en la pantalla impresión automática
//(mdEtiq_.....)a las variables del registro de datos interno (bd_.......) todo ello en  minúsculas
        String aux;
        int aux_int;


        aux_int=_Hour.getIntValue();
        bd_hora[0].write(aux_int);

        aux_int=_Minutes.getIntValue();
        bd_minuto[0].write(aux_int);

        aux_int=_Day.getIntValue();
        bd_dia[0].write(aux_int);

        aux_int=_Month.getIntValue();
        bd_mes[0].write(aux_int);


        aux=mdEtiq_Paquete.getStringValue();
        aux=aux.toLowerCase();          //se pasan a minúsculas para que estén todos iguales
        bd_num_paquete[0].write(aux);

        aux=mdEtiq_Colada.getStringValue();
        aux=aux.toLowerCase();
        bd_num_colada[0].write(aux);

        aux=mdEtiq_Diametro.getStringValue();
        aux=aux.toLowerCase();
        bd_formato1[0].write(aux);


        aux=mdEtiq_Longitud.getStringValue();
        aux=aux.toLowerCase();
```

```
bd_formato2[0].write(aux);


aux=mdEtiq_Piezas.getStringValue();
aux=aux.toLowerCase();
bd_num_piezas[0].write(aux);

aux=mdEtiq_Cliente.getStringValue();
aux=aux.toLowerCase();

aux=mdEtiq_Peso.getStringValue();
aux=aux.toLowerCase();
bd_peso[0].write(aux);


aux=mdEtiq_Aleacion_Num.getStringValue();
aux=aux.toLowerCase();
bd_aleacion1[0].write(aux);

aux=mdEtiq_Aleacion_Char.getStringValue();
aux=aux.toLowerCase();
bd_aleacion2[0].write(aux);
```

## *Script(4)*

//Script desarrollado para manejar la impresión.

```
// 1. Caracteres de control de la Impresión . Crea un script de variables local para los
comandos de la impresora: nueva línea, alimentación de línea y retorno del carruaje.
Estas variable simplificarán la escritura, lectura y mantenimiento del script.

byte LineFeed = (byte) 0x0a;        // Avance de línea
byte CarriageReturn = (byte) 0x0d;  // Retorno de carro
byte FormFeed = (byte) 0x0c;        // Avance de página

// 2. Buffer de la impresora y otras variables. El script también requiere de la matriz de
byte y de carácter para el búfer de impresión; de cadenas de almacenaje temporal, y de
variables enteras para varios índices.
byte PrintBuffer[] = new byte[1024];
char[] sendToPrinterTMP; // matriz de caracteres temporal
byte sendToPrinter[];    // matriz de bytes para cada variable
int numOfChar = 0,       // longitud de la cadena
i, j, k, BufferPosition = 0;

// 3. Imprimir los valores de las variables
        for (i=0; i<5; i++)
 {
// a. Crear una cadena con los valores a imprimir, teniendo que concatenar cadenas de
datos a imprimir en la misma línea y teniendo que introducir espacios para situar el
texto impreso dentro del campo correcto
 String LocalMessage=" " ;
                if (i==0){
```

```
      LocalMessage.concat(" = "+ "         " + bd_cliente[0].getStringValue());
         }
      if (i==1){
   LocalMessage.concat(" = "+ bd_aleacion1[0].getStringValue()+
bd_aleacion2[0].getStringValue()+ "        " + bd_num_colada[0].getStringValue());
      }
        if (i==2){
          LocalMessage.concat(" = "+ bd_aleacion1[0].getStringValue()+
bd_aleacion2[0].getStringValue()+"        " + bd_num_piezas[0].getStringValue() );
          }              if (i==3){
      LocalMessage.concat(" = "+ bd_peso[0].getStringValue() + "        " +
bd_num_paquete[0].getStringValue());
          }
//en caso de querer añadir el código de barras impreso, realizar otro if
introduciendo los valores de control proporcionados en el manual del
programador de la impresora

  // b. Convertir desde la cadena a un carácter de matriz El script emplea primero el
número de caracteres en la cadena para definir el tamaño de una nueva matriz de byte.
A continuación, el script convierte la variable de cadena local en una matriz de
caracteres.
   numOfChar = LocalMessage.length();
  sendToPrinter = new byte[numOfChar]; // defina el tamaño de una matriz de bytes
  sendToPrinterTMP = LocalMessage.toCharArray();

   // c. Convierte la matriz de un carácter unicode a una matriz ASCII byte. El script
encasilla cada carácter en la matriz de carácter en un byte y lo asigna al elemento
correspondiente en la nueva matriz byte creada. Cada elemento en la matriz byte es
equivalente a un carácter ASCII
   for (j=0; j<numOfChar; j++)
   {
    sendToPrinter[j] = (byte)sendToPrinterTMP[j];
   }

   // d. Mueve caracteres a una matriz de búfer de impresión. El script copia los
caracteres del byte desde una matriz de byte temporal a la matriz de bytePrintBuffer.
    for (k=0; k<j; k++)
    {
    PrintBuffer[BufferPosition++] = sendToPrinter[k];
    }


// e. Cambia a una nueva línea. El script agrega caracteres de formato de impresión al
final del mensaje para cambiar a la siguiente línea.
    PrintBuffer[BufferPosition++] = LineFeed;
    PrintBuffer[BufferPosition++] = CarriageReturn;
}

// 4. Cambie a una nueva página. el script agrega al final un carácter de control de
alimentación de página.
PrintBuffer[BufferPosition++] = FormFeed;
```

*// 5. Envíe el búfer a un controlador de script*
*Print_Variables.write(true, BufferPosition+1, PrintBuffer);*
*El script envía una matriz de byte PrintBuffer al controlador del script.*

*El primer parámetro es true porque el búfer de recepción (puerto en serie del controlador de script) no es usado para recibir ninguna información desde la impresora.*

*El segundo parámetro emplea la variable BufferPosition porque rastrea el número del último elemento usado en la matriz de byte PrintBuffer. Necesitamos agregar 1 a BufferPosition porque los números de elementos de la matriz empiezan en la posición 0.*

*El tercer parámetro es la matriz de byte PrintBuffer, la misma que almacena todos los valores de las variables y caracteres de control de impresión en la matriz entera PrintValues.*

## *Script(5)*

```
//Comprueba que todos los campos están completos y su valor no es 0
//si se quiere que un campo tenga un tamaño determinado.modificar el if
//correspondiente
                String aux;
                int aux_int;
                campos_rellenos.write(0);

                aux=man_etiq_cliente.getStringValue();
                aux_int=aux.length();
                if (aux_int>0 && !aux.equalsIgnoreCase("0")){

                  aux=man_etiq_formato1.getStringValue();
                  aux_int=aux.length();
                     if (aux_int>0 && !aux.equalsIgnoreCase("0")){

                        aux=man_etiq_formato1.getStringValue();
                        aux_int=aux.length();
                            if(aux_int>0 && !aux.equalsIgnoreCase("0")){

                              aux=man_etiq_formato2.getStringValue();
                               aux_int=aux.length();
                                  if(aux_int>0 && !aux.equalsIgnoreCase("0")){

                                    aux=man_etiq_aleacion1.getStringValue();
                                     aux_int=aux.length();
                                         if(aux_int>0 && !aux.equalsIgnoreCase("0")){

                                         aux=man_etiq_aleacion2.getStringValue();
                                          aux_int=aux.length();
                                    if(aux_int>0 && !aux.equalsIgnoreCase("0")){

                                    aux=man_etiq_num_colada.getStringValue();
                                     aux_int=aux.length();
                            if(aux_int>0 && !aux.equalsIgnoreCase("0")){
```

```
                aux=man_etiq_num_paquete.getStringValue();
                 aux_int=aux.length();
          if(aux_int>0 && !aux.equalsIgnoreCase("0")){

                aux=man_etiq_peso.getStringValue();
                 aux_int=aux.length();
 if(aux_int>0 && !aux.equalsIgnoreCase("0")){

campos_rellenos.write(1);

}}}}}}}}}
```

## *Script(6)*

```
//Captura de datos. Copia de lo que se muestra en la pantalla impresión manual
//(man_etiq_.....)a las variables del registro de datos interno (bd_.......) todo ello en minúsculas
aux_int=_Hour.getIntValue();
bd_hora[0].write(aux_int);

aux_int=_Minutes.getIntValue();
bd_minuto[0].write(aux_int);

aux_int=_Day.getIntValue();
bd_dia[0].write(aux_int);

aux_int=_Month.getIntValue();
bd_mes[0].write(aux_int);

aux=man_etiq_num_paquete.getStringValue();
aux=aux.toLowerCase(); / /pasar a minusculas
bd_num_paquete[0].write(aux);

aux=man_etiq_num_colada.getStringValue();
aux=aux.toLowerCase();
bd_num_colada[0].write(aux);

aux=man_etiq_formato1.getStringValue();
aux=aux.toLowerCase();
bd_formato1[0].write(aux);

aux=man_etiq_formato2.getStringValue();
aux=aux.toLowerCase();
bd_formato2[0].write(aux);

aux=man_etiq_num_piezas.getStringValue();
aux=aux.toLowerCase();
bd_num_piezas[0].write(aux);

aux=man_etiq_cliente.getStringValue();
```

```
aux=aux.toLowerCase();
bd_cliente[0].write(aux);

aux=man_etiq_peso.getStringValue();
aux=aux.toLowerCase();
bd_peso[0].write(aux);

aux=man_etiq_aleacion1.getStringValue();
aux=aux.toLowerCase();
bd_aleacion1[0].write(aux);

aux=man_etiq_aleacion2.getStringValue();
aux=aux.toLowerCase();
bd_aleacion2[0].write(aux);
```

## *Script(7)*

```
//script encargado de subir en 8 unidades la lista de datos hasta llegar a la
//posición 0
int aux=0;
aux=pos_vect_0.getIntValue();
aux=aux-8;

if (aux<0)
{
pos_vect_0.write(0);
pos_vect_1.write(1);
pos_vect_2.write(2);
pos_vect_3.write(3);
pos_vect_4.write(4);
pos_vect_5.write(5);
pos_vect_6.write(6);
pos_vect_7.write(7);
pos_vect_8.write(8);
}
if (aux>=0)
{
aux=pos_vect_0.getIntValue();
aux=aux-8;
pos_vect_0.write(aux);

aux=pos_vect_1.getIntValue();
aux=aux-8;
pos_vect_1.write(aux);


aux=pos_vect_2.getIntValue();
aux=aux-8;
pos_vect_2.write(aux);
```

```
aux=pos_vect_3.getIntValue();
aux=aux-8;
pos_vect_3.write(aux);

aux=pos_vect_4.getIntValue();
aux=aux-8;
pos_vect_4.write(aux);

aux=pos_vect_5.getIntValue();
aux=aux-8;
pos_vect_5.write(aux);

aux=pos_vect_6.getIntValue();
aux=aux-8;
pos_vect_6.write(aux);

aux=pos_vect_7.getIntValue();
aux=aux-8;
pos_vect_7.write(aux);

aux=pos_vect_8.getIntValue();
aux=aux-8;
pos_vect_8.write(aux);
}
```

## *Script(8)*

```
//script encargado de subir en 1 unidad la lista de datos hasta llegar a la
//posición 0

int aux=0;
aux=pos_vect_0.getIntValue();

if (aux>0)
{
aux=pos_vect_0.getIntValue();
aux=aux-1;
pos_vect_0.write(aux);

aux=pos_vect_1.getIntValue();
aux=aux-1;
pos_vect_1.write(aux);


aux=pos_vect_2.getIntValue();
aux=aux-1;
pos_vect_2.write(aux);

aux=pos_vect_3.getIntValue();
aux=aux-1;
```

```
pos_vect_3.write(aux);

aux=pos_vect_4.getIntValue();
aux=aux-1;
pos_vect_4.write(aux);

aux=pos_vect_5.getIntValue();
aux=aux-1;
pos_vect_5.write(aux);

aux=pos_vect_6.getIntValue();
aux=aux-1;
pos_vect_6.write(aux);

aux=pos_vect_7.getIntValue();
aux=aux-1;
pos_vect_7.write(aux);

aux=pos_vect_8.getIntValue();
aux=aux-1;
pos_vect_8.write(aux);
}
```

## *Script(9)*

```
//script encargado de bajar en 1 unidad la lista de datos hasta llegar a última posición
int aux=0;
aux=pos_vect_8.getIntValue();

if (aux<49)
{
aux=pos_vect_0.getIntValue();
aux=aux+1;
pos_vect_0.write(aux);

aux=pos_vect_1.getIntValue();
aux=aux+1;
pos_vect_1.write(aux);

aux=pos_vect_2.getIntValue();
aux=aux+1;
pos_vect_2.write(aux);

aux=pos_vect_3.getIntValue();
aux=aux+1;
pos_vect_3.write(aux);

aux=pos_vect_4.getIntValue();
```

```
aux=aux+1;
pos_vect_4.write(aux);

aux=pos_vect_5.getIntValue();
aux=aux+1;
pos_vect_5.write(aux);

aux=pos_vect_6.getIntValue();
aux=aux+1;
pos_vect_6.write(aux);

aux=pos_vect_7.getIntValue();
aux=aux+1;
pos_vect_7.write(aux);

aux=pos_vect_8.getIntValue();
aux=aux+1;
pos_vect_8.write(aux);
}
```

## *Script(10)*

//script encargado de bajar en 8 unidades la lista de datos hasta llegar a la posición 49

```
int aux=0;
aux=pos_vect_8.getIntValue();
aux=aux+8;

if (aux>49)
{
pos_vect_0.write(41);
pos_vect_1.write(42);
pos_vect_2.write(43);
pos_vect_3.write(44);
pos_vect_4.write(45);
pos_vect_5.write(46);
pos_vect_6.write(47);
pos_vect_7.write(48);
pos_vect_8.write(49);
}

if (aux<=49)
{
aux=pos_vect_0.getIntValue();
aux=aux+8;
pos_vect_0.write(aux);
```

```
aux=pos_vect_1.getIntValue();
aux=aux+8;
pos_vect_1.write(aux);

aux=pos_vect_2.getIntValue();
aux=aux+8;
pos_vect_2.write(aux);

aux=pos_vect_3.getIntValue();
aux=aux+8;
pos_vect_3.write(aux);

aux=pos_vect_4.getIntValue();
aux=aux+8;
pos_vect_4.write(aux);

aux=pos_vect_5.getIntValue();
aux=aux+8;
pos_vect_5.write(aux);

aux=pos_vect_6.getIntValue();
aux=aux+8;
pos_vect_6.write(aux);

aux=pos_vect_7.getIntValue();
aux=aux+8;
pos_vect_7.write(aux);
aux=pos_vect_8.getIntValue();
aux=aux+8;
pos_vect_8.write(aux);
}
```

## *Script(11)*

```
//guardado para saber cual se quiere eliminar sabiendo el selector  accionado y
//teniendo en cuenta que el vector de datos se desplaza con los cursores para
//poder visualizar la lista completa de datos
    int aux_int=99;
    String aux;

    int tipo_0, tipo_1, tipo_2, tipo_3, tipo_4,
     tipo_5, tipo_6,tipo_7 ,tipo_8;


    tipo_0=sel_0.getIntValue();
    tipo_1=sel_1.getIntValue();
    tipo_2=sel_2.getIntValue();
    tipo_3=sel_3.getIntValue();
    tipo_4=sel_4.getIntValue();
```

```
tipo_5=sel_5.getIntValue();
tipo_6=sel_6.getIntValue();
tipo_7=sel_7.getIntValue();
tipo_8=sel_8.getIntValue();

//guardado para saber cual se quiere eliminar sabiendo el selector accionado y
//teniendo en cuenta que el vector de datos se desplaza con los cursores para
poder
//visualizar la lista completa de datos

if (tipo_0 != 0)
{
aux_int=(pos_vect_0.getIntValue())+1;
sel_0.write(0);
busc_pos_borrado.write(aux_int);
}

if (tipo_1 == 1)
{
aux_int=(pos_vect_1.getIntValue())+1;
busc_pos_borrado.write(aux_int);

}

if (tipo_2 == 1)
{
aux_int=(pos_vect_2.getIntValue())+1;
busc_pos_borrado.write(aux_int);

}

if (tipo_3 == 1)
{
aux_int=(pos_vect_3.getIntValue())+1;
busc_pos_borrado.write(aux_int);
}

if (tipo_4 == 1)
{
aux_int=(pos_vect_4.getIntValue())+1;
busc_pos_borrado.write(aux_int);
}

if (tipo_5 == 1)
{
aux_int=(pos_vect_5.getIntValue())+1;
busc_pos_borrado.write(aux_int);
        }
if (tipo_6 == 1)
{
aux_int=(pos_vect_6.getIntValue())+1;
busc_pos_borrado.write(aux_int);
}

if (tipo_7 == 1)
{
```

```
aux_int=(pos_vect_7.getIntValue())+1;
busc_pos_borrado.write(aux_int);
}

if (tipo_8 == 1)
{
aux_int=(pos_vect_8.getIntValue())+1;
busc_pos_borrado.write(aux_int);
}
```

## *Script(12)*

```
 //guardado para saber cual se quiere modificar sabiendo el selector accionado y
//teniendo en cuenta que el vector de datos se desplaza con los cursores para poder visualizar
//la lista completa de datos
int pos;
int aux_int;
String aux;
int tipo_0, tipo_1, tipo_2, tipo_3, tipo_4,
 tipo_5, tipo_6,tipo_7 ,tipo_8;

tipo_0=sel_0.getIntValue();
tipo_1=sel_1.getIntValue();
tipo_2=sel_2.getIntValue();
tipo_3=sel_3.getIntValue();
tipo_4=sel_4.getIntValue();
tipo_5=sel_5.getIntValue();
tipo_6=sel_6.getIntValue();
tipo_7=sel_7.getIntValue();
tipo_8=sel_8.getIntValue();


if (tipo_0 != 0)
{
pos=pos_vect_0.getIntValue();
sel_0.write(0);
}
if (tipo_1 != 0)
{
pos=pos_vect_1.getIntValue();
sel_1.write(0);
}
if (tipo_2 != 0)
{
pos=pos_vect_2.getIntValue();
sel_2.write(0);
}
if (tipo_3 != 0)
{
pos=pos_vect_3.getIntValue();
sel_3.write(0);
}
if (tipo_4 != 0)
{
```

```
pos=pos_vect_4.getIntValue();
sel_4.write(0);
}
if (tipo_5 != 0)
{
pos=pos_vect_5.getIntValue();
sel_5.write(0);
}
if (tipo_6 != 0)
{
pos=pos_vect_6.getIntValue();
sel_6.write(0);
}
if (tipo_7 != 0)
{
pos=pos_vect_7.getIntValue();
sel_7.write(0);
}
if (tipo_8 != 0)
{
pos=pos_vect_8.getIntValue();
sel_8.write(0);
}
//Copia los datos en los referentes a impresión manual para que el operario
//proceda mediante la pantalla impresión manual
aux=bd_num_paquete[pos].getStringValue();
man_etiq_num_paquete.write(aux);

aux=bd_num_colada[pos].getStringValue();
man_etiq_num_colada.write(aux);

aux=bd_formato1[pos].getStringValue();
man_etiq_formato1.write(aux);

aux=bd_formato2[pos].getStringValue();
man_etiq_formato2.write(aux);

aux=bd_num_piezas[pos].getStringValue();
man_etiq_num_piezas.write(aux);

aux=bd_cliente[pos].getStringValue();
man_etiq_cliente.write(aux);

aux=bd_peso[pos].getStringValue();
man_etiq_peso.write(aux);

aux=bd_aleacion1[pos].getStringValue();
man_etiq_aleacion1.write(aux);

aux=bd_aleacion2[pos].getStringValue();
man_etiq_aleacion2.write(aux);
```

# *Script(13)*

```
//script encargado de bajar 1 unidad la lista de datos hasta llegar a la posición
//49 o la última búsqueda encontrada
        int aux=0 , encontradas=0;
        aux=pos_vect_7.getIntValue();
        encontradas=busc_encontradas.getIntValue()-1;

        if (aux<49 && aux<encontradas )
        {
        aux=pos_vect_0.getIntValue();
        aux=aux+1;
        pos_vect_0.write(aux);

        aux=pos_vect_1.getIntValue();
        aux=aux+1;
        pos_vect_1.write(aux);


        aux=pos_vect_2.getIntValue();
        aux=aux+1;
        pos_vect_2.write(aux);

        aux=pos_vect_3.getIntValue();
        aux=aux+1;
        pos_vect_3.write(aux);

        aux=pos_vect_4.getIntValue();
        aux=aux+1;
        pos_vect_4.write(aux);

        aux=pos_vect_5.getIntValue();
        aux=aux+1;
        pos_vect_5.write(aux);

        aux=pos_vect_6.getIntValue();
        aux=aux+1;
        pos_vect_6.write(aux);

        aux=pos_vect_7.getIntValue();
        aux=aux+1;
        pos_vect_7.write(aux);
        }
```

# *Script(14)*

```
//script encargado de bajar en 8 unidades la lista de datos hasta llegar a la
//posición 49 o la última búsqueda encontrada
                int aux=0, encontradas=0;
                aux=pos_vect_7.getIntValue();
                encontradas=busc_encontradas.getIntValue()-1;

                if ((aux+8)>49 )
                {
                pos_vect_0.write(41);
                pos_vect_1.write(42);
                pos_vect_2.write(43);
                pos_vect_3.write(44);
                pos_vect_4.write(45);
                pos_vect_5.write(46);
                pos_vect_6.write(47);
                pos_vect_7.write(48);
                pos_vect_8.write(49);
                }
                if ((aux+8)<=49 && aux<encontradas)
                {
                aux=pos_vect_0.getIntValue();
                aux=aux+8;
                pos_vect_0.write(aux);

                aux=pos_vect_1.getIntValue();
                aux=aux+8;
                pos_vect_1.write(aux);


                aux=pos_vect_2.getIntValue();
                aux=aux+8;
                pos_vect_2.write(aux);

                aux=pos_vect_3.getIntValue();
                aux=aux+8;
                pos_vect_3.write(aux);

                aux=pos_vect_4.getIntValue();
                aux=aux+8;
                pos_vect_4.write(aux);

                aux=pos_vect_5.getIntValue();
                aux=aux+8;
                pos_vect_5.write(aux);

                aux=pos_vect_6.getIntValue();
                aux=aux+8;
                pos_vect_6.write(aux);

                aux=pos_vect_7.getIntValue();
                aux=aux+8;
                pos_vect_7.write(aux);
```

```
aux=pos_vect_8.getIntValue();
aux=aux+8;
pos_vect_8.write(aux);
}
```

## *Script(15)*

//comprueba que tipo de búsqueda está marcada y copia los datos a una
//variable intermedia para comparar ambas cadenas

```
int j=0,tipo_1=0, tipo_2=0, tipo_3=0, tipo_4=0;
String aux;
//elimina los selectores
pos_vect_0.write(0);
pos_vect_1.write(1);
pos_vect_2.write(2);
pos_vect_3.write(3);
pos_vect_4.write(4);
pos_vect_5.write(5);
pos_vect_6.write(6);
pos_vect_7.write(7);
pos_vect_8.write(8);
//comprueba el tipo de busqueda
tipo_1=busc_tipo_1.getIntValue();
tipo_2=busc_tipo_2.getIntValue();
tipo_3=busc_tipo_3.getIntValue();
tipo_4=busc_tipo_4.getIntValue();
//vacia vector intermedio para comparar las cadenaas
for (j=0; j<60; j++)
{
 busc_posicion[j].write(-1);//donde se encontraba el dato en bd... -1 diferenciarlo
de la ultima pos de bd
dato_buscado[j].write(0);

busc_aleacion1[j].write(0);
busc_aleacion2[j].write(0);
busc_formato1[j].write(0);
busc_formato2[j].write(0);
busc_num_colada[j].write(0);
busc_num_paquete[j].write(0);
busc_num_piezas[j].write(0);
busc_cliente[j].write(0);

busc_peso[j].write(0);
busc_mes[j].write(0);
busc_dia[j].write(0);
busc_hora[j].write(0);
busc_minuto[j].write(0);
}
//copia de los datos de la bd referido al selector seleccionado
```

```
if ( tipo_1 ==1)
{
        for (j=0; j<60; j++)
{
aux= bd_cliente[j].getStringValue();
dato_buscado[j].write(aux);
}}

if ( tipo_2 ==1)
{
        for (j=0; j<60; j++)
{
aux= bd_num_colada[j].getStringValue();
dato_buscado[j].write(aux);
}}
if ( tipo_3 ==1)
{
        for (j=0; j<60; j++)
{
aux= bd_num_piezas[j].getStringValue();
dato_buscado[j].write(aux);
}}
if ( tipo_4 ==1)
{
        for (j=0; j<60; j++)
{
aux= bd_peso[j].getStringValue();
dato_buscado[j].write(aux);
}}
```

## *Script(16)*

```
/realiza la búsqueda utilizando un switch para ello
        //Caso 0        .busq.literal                   caso2 comienzo
        //Caso 1        : terminacion                   caso 3 contenga


int caso= 0,j=0, aux_int;
String buscador;
String asterisco;
String aux;


asterisco= "*";
buscador=cadena.getStringValue();
buscador=buscador.toLowerCase();
////////////buscar caso/////////////
//si hay solo escritos * ó **
if(buscador.equalsIgnoreCase("*") || buscador.equalsIgnoreCase("**")){
        busc_encontradas.write(0);
```

```
            buscador="|";
            caso=7;
}
//si empieza por *
if (buscador.startsWith(asterisco)){
buscador=buscador.substring(1,buscador.length( ));//elimina el *
caso=caso+1;
}
//si acaba por *
if (buscador.endsWith(asterisco)){
            buscador=buscador.substring(0,buscador.length()-1); //elimina el *

caso=caso+2;
}



////////////////////////////////////////////////////SWITCH//////////////////////////
switch (caso)
{
//////////////////////////////////////////CASO 0/////////////////////////////////////////////////////////
case 0 :    //[instrucciones que se ejecutan cuando la expresión es igual al
            //valor 0..ni empieza ni acaba por *.
               /busqueda literal

  for (int i=0; i<60; i++)
    {
    aux=dato_buscado[i].getStringValue();

        if (aux.equalsIgnoreCase(buscador) && bd_dia[i].getIntValue()>0)   //si es
//verdadero copia las variables para visualizarlas..se añadió el filtro de dia ya
//que no existe el dia 0 y evitamos errores cuando la bbdd no está llena
          {
      aux=bd_cliente[i].getStringValue();
      busc_cliente[j].write(aux);

      aux_int=bd_hora[i].getIntValue();
      busc_hora[j].write(aux_int);

      aux_int=bd_minuto[i].getIntValue();
      busc_minuto[j].write(aux_int);

      aux_int=bd_dia[i].getIntValue();
      busc_dia[j].write(aux_int);

      aux_int=bd_mes[i].getIntValue();
      busc_mes[j].write(aux_int);

      aux=bd_cliente[i].getStringValue();
      busc_cliente[j].write(aux);

      aux=bd_peso[i].getStringValue();
      busc_peso[j].write(aux);

      aux=bd_aleacion1[i].getStringValue();
      busc_aleacion1[j].write(aux);
```

```
        aux=bd_aleacion2[i].getStringValue();
        busc_aleacion2[j].write(aux);

        aux=bd_formato1[i].getStringValue();
        busc_formato1[j].write(aux);

        aux=bd_formato2[i].getStringValue();
        busc_formato2[j].write(aux);

        aux=bd_num_colada[i].getStringValue();
        busc_num_colada[j].write(aux);

        aux=bd_num_paquete[i].getStringValue();
         busc_num_paquete[j].write(aux);

        aux=bd_num_piezas[i].getStringValue();
         busc_num_piezas[j].write(aux);



        busc_posicion[j].write(i);//donde se encontraba el dato para poder tratarlo
                                  //cn pulsadores de selección
             j=j+1;


           }
      }

break;

//////////////////////////////////////////CASO 1//////////////////////////////////////////////////////////////


case 1 :                  //[instrucciones que se ejecutan cuando la expresión es
             //igual al valor 1.empieza por * y hay que buscar la terminación dada

 for (int i=0; i<60; i++)
    {
    aux=dato_buscado[i].getStringValue();

        if (aux.endsWith(buscador) && bd_dia[i].getIntValue()>0)   //si es
verdadero copia las variables para visualixarlas
           {
        aux=bd_cliente[i].getStringValue();
        busc_cliente[j].write(aux);

        aux=bd_hora[i].getStringValue();
        busc_hora[j].write(aux);

        aux=bd_minuto[i].getStringValue();
        busc_minuto[j].write(aux);

        aux=bd_dia[i].getStringValue();
        busc_dia[j].write(aux);
```

```
      aux=bd_mes[i].getStringValue();
      busc_mes[j].write(aux);

      aux=bd_cliente[i].getStringValue();
      busc_cliente[j].write(aux);

      aux=bd_peso[i].getStringValue();
      busc_peso[j].write(aux);

      aux=bd_aleacion1[i].getStringValue();
      busc_aleacion1[j].write(aux);

      aux=bd_aleacion2[i].getStringValue();
      busc_aleacion2[j].write(aux);

      aux=bd_formato1[i].getStringValue();
      busc_formato1[j].write(aux);

      aux=bd_formato2[i].getStringValue();
      busc_formato2[j].write(aux);

      aux=bd_num_colada[i].getStringValue();
      busc_num_colada[j].write(aux);

      aux=bd_num_paquete[i].getStringValue();
      busc_num_paquete[j].write(aux);
      aux=bd_num_piezas[i].getStringValue();
       busc_num_piezas[j].write(aux);

      busc_posicion[j].write(i);//vector que contiene la posicion de donde se copio
el dato de la bbdd

        j=j+1;

      }    }

break;

//////////////////////////////////////////CASO 2//////////////////////////////////////////////////////////

case 2 :
//[instrucciones que se ejecutan cuando la expresión es igual al valor 2. ];
//termina por *, hay que buscar los datos que empiezen por la cadena dada
for (int i=0; i<60; i++)
    {
    aux=dato_buscado[i].getStringValue();

        if (aux.startsWith(buscador) && bd_dia[i].getIntValue()>0)   //si es
verdadero copia las variables para visualixarlas
          {
      aux=bd_cliente[i].getStringValue();
      busc_cliente[j].write(aux);

      aux=bd_hora[i].getStringValue();
      busc_hora[j].write(aux);
```

```
      aux=bd_minuto[i].getStringValue();
      busc_minuto[j].write(aux);

      aux=bd_dia[i].getStringValue();
      busc_dia[j].write(aux);

      aux=bd_mes[i].getStringValue();
      busc_mes[j].write(aux);

      aux=bd_cliente[i].getStringValue();
      busc_cliente[j].write(aux);

      aux=bd_peso[i].getStringValue();
      busc_peso[j].write(aux);

      aux=bd_aleacion1[i].getStringValue();
      busc_aleacion1[j].write(aux);

      aux=bd_aleacion2[i].getStringValue();
      busc_aleacion2[j].write(aux);

    aux=bd_formato1[i].getStringValue();
      busc_formato1[j].write(aux);

      aux=bd_formato2[i].getStringValue();
      busc_formato2[j].write(aux);


      aux=bd_num_colada[i].getStringValue();
      busc_num_colada[j].write(aux);

      aux=bd_num_paquete[i].getStringValue();
      busc_num_paquete[j].write(aux);

        aux=bd_num_piezas[i].getStringValue();
       busc_num_piezas[j].write(aux);

        busc_posicion[j].write(i);//vector que contiene la posicion de donde se
copio el dato para bd

        j=j+1;

      }    }

break;

case 3 :
//[instrucciones que se ejecutan cuando la expresión es igual al valor 3];
//termina y empieza por /, hay que buscar los datos que contengan  la cadena
//dada
 for (int i=0; i<60; i++)
    {
    aux=dato_buscado[i].getStringValue();

       if (aux.indexOf(buscador)>-1 && bd_dia[i].getIntValue()>0)   //si es
verdadero copia las variables para visualixarlas
```

```
    {
aux=bd_cliente[i].getStringValue();
busc_cliente[jj].write(aux);

aux=bd_hora[i].getStringValue();
busc_hora[jj].write(aux);

aux=bd_minuto[i].getStringValue();
busc_minuto[jj].write(aux);

aux=bd_dia[i].getStringValue();
busc_dia[jj].write(aux);

aux=bd_mes[i].getStringValue();
busc_mes[jj].write(aux);

aux=bd_cliente[i].getStringValue();
busc_cliente[jj].write(aux);

aux=bd_peso[i].getStringValue();
busc_peso[jj].write(aux);

aux=bd_aleacion1[i].getStringValue();
busc_aleacion1[jj].write(aux);

aux=bd_aleacion2[i].getStringValue();
busc_aleacion2[jj].write(aux);

aux=bd_formato1[i].getStringValue();
busc_formato1[jj].write(aux);

aux=bd_formato2[i].getStringValue();
busc_formato2[jj].write(aux);


aux=bd_num_colada[i].getStringValue();
busc_num_colada[jj].write(aux);

aux=bd_num_paquete[i].getStringValue();
busc_num_paquete[jj].write(aux);

aux=bd_num_piezas[i].getStringValue();
 busc_num_piezas[jj].write(aux);

busc_posicion[jj].write(i);//vector que contiene la posicion de donde se copio
el dato para bd

  jj=jj+1;
   }    }

break;

  }

  busc_encontradas.write(j);
```

# *Script(17)*

//script desarrollado para la búsqueda por turnos

```
int hora1=0, hora2=0;
int aux_int;
int j=0;
String aux;

//si hay algún selector de búsqueda poir turno activado
if (busc_turno_manana.getIntValue()==1 || busc_turno_tarde.getIntValue()==1 ||
busc_turno_noche.getIntValue()==1)
{
//copia las horas para buscar…modificar en caso de que cambie el horario de
//los turnos
if (busc_turno_manana.getIntValue()==1){
        hora1=6;
    hora2=14;

        }
        if (busc_turno_tarde.getIntValue()==1){
        hora1=14;
    hora2=22;
        }
if (busc_turno_noche.getIntValue()==1){
        hora1=22;
    hora2=6;

}

////////////buscar caso////////////
for (int h=hora1; h!=hora2; h++)
  {
        if (h==24){ // horas de 0 a 23
h=0;
}
for (int i=0; i<49; i++){

      if(h == bd_hora[i].getIntValue()){

        if (bd_dia[i].getIntValue()>0){   //empleado solo al principio...cuando la
//bd no está entera llena

        aux_int=bd_dia[i].getIntValue();
    busc_dia[j].write(aux_int);

         aux_int=bd_hora[i].getIntValue();
    busc_hora[j].write(aux_int);

    aux_int=bd_minuto[i].getIntValue();
    busc_minuto[j].write(aux_int);
```

```
aux_int=bd_mes[i].getIntValue();
 busc_mes[jj].write(aux_int);

      aux=bd_cliente[i].getStringValue();
 busc_cliente[jj].write(aux);


 aux=bd_peso[i].getStringValue();
 busc_peso[j].write(aux);

 aux=bd_aleacion1[i].getStringValue();
 busc_aleacion1[j].write(aux);

  aux=bd_aleacion2[i].getStringValue();
 busc_aleacion2[j].write(aux);

 aux=bd_formato1[i].getStringValue();
 busc_formato1[j].write(aux);

  aux=bd_formato2[i].getStringValue();
 busc_formato2[j].write(aux);

 aux=bd_num_colada[i].getStringValue();
 busc_num_colada[j].write(aux);

 aux=bd_num_paquete[i].getStringValue();
 busc_num_paquete[j].write(aux);

 aux=bd_num_piezas[i].getStringValue();
  busc_num_piezas[j].write(aux);

 busc_posicion[j].write(i);
j=j+1;
}}   }   }
  busc_encontradas.write(j);
}
```

## *Script(18)*

```
//Script encargado de borrar el dato seleccionado de la base de datos
//empleando un marcador asociado al dato

int aux_int=99;
String aux;
int j=0;
busc_encontradas.write(0);
for ( j=0; j<49; j++)
{
busc_posicion[j].write(-1);
busc_aleacion1[j].write(0);
busc_aleacion2[j].write(0);
busc_formato1[j].write(0);
busc_formato2[j].write(0);
busc_num_colada[j].write(0);
busc_num_paquete[j].write(0);
busc_num_piezas[j].write(0);
busc_cliente[j].write(0);

busc_peso[j].write(0);
busc_mes[j].write(0);
busc_dia[j].write(0);
busc_hora[j].write(0);
busc_minuto[j].write(0);
}



aux_int=busc_pos_borrado.getIntValue();
//rotamos todos los datos de la base de datos para así suplir el hueco que deja
los datos eliminados
for (j=aux_int; j<=49; j ++)
{
aux_int=bd_hora[j].getIntValue();
bd_hora[j-1].write(aux_int);

aux_int=bd_minuto[j].getIntValue();
bd_minuto[j-1].write(aux_int);

aux_int=bd_mes[j].getIntValue();
bd_mes[j-1].write(aux_int);

aux_int=bd_dia[j].getIntValue();
bd_dia[j-1].write(aux_int);

aux=bd_num_paquete[j].getStringValue();
bd_num_paquete[j-1].write(aux);

aux=bd_num_colada[j].getStringValue();
```

```
bd_num_colada[j-1].write(aux);

aux=bd_formato1[j].getStringValue();
bd_formato1[j-1].write(aux);

aux=bd_formato2[j].getStringValue();
bd_formato2[j-1].write(aux);

aux=bd_num_piezas[j].getStringValue();
bd_num_piezas[j-1].write(aux);

aux=bd_cliente[j].getStringValue();
bd_cliente[j-1].write(aux);

aux=bd_peso[j].getStringValue();
bd_peso[j-1].write(aux);

aux=bd_aleacion1[j].getStringValue();
bd_aleacion1[j-1].write(aux);

aux=bd_aleacion2[j].getStringValue();
bd_aleacion2[j-1].write(aux);
}
//encargado de borrar el último dato de la lista
bd_hora[49].write(0);
bd_minuto[49].write(0);
bd_mes[49].write(0);
bd_dia[49].write(0);
bd_num_paquete[49].write(0);
bd_num_colada[49].write(0);
bd_formato1[49].write(0);
bd_formato2[49].write(0);
bd_num_piezas[49].write(0);
bd_cliente[49].write(0);
bd_peso[49].write(0);
bd_aleacion1[49].write(0);
bd_aleacion2[49].write(0);


//borra los selectores de opciones
sel_0.write(0);
sel_1.write(0);
sel_2.write(0);
sel_3.write(0);
sel_4.write(0);
sel_5.write(0);
sel_6.write(0);
sel_7.write(0);
sel_8.write(0);
```

## *Script(19)*

```
//Script encargado de guardar los datos para ser compartidos… igualar la
//posición 0 del vector de base de datos interno( bd_  ) a las variables
//compartidas con el PLC (comp_)

        String aux;


    aux=bd_hora[0].getStringValue();
    comp_hora.write(aux);

    aux=bd_minuto[0].getStringValue();
    comp_minuto.write(aux);

    aux=bd_dia[0].getStringValue();
    comp_dia.write(aux);

    aux=bd_mes[0].getStringValue();
    comp_mes.write(aux);

    aux=bd_cliente[0].getStringValue();
    comp_cliente.write(aux);

    aux=bd_peso[0].getStringValue();
    comp_peso.write(aux);

    aux=bd_aleacion1[0].getStringValue();
    comp_aleacion1.write(aux);

    aux=bd_aleacion2[0].getStringValue();
    comp_aleacion2.write(aux);

    aux=bd_formato1[0].getStringValue();
    comp_formato1.write(aux);

    aux=bd_formato2[0].getStringValue();
    comp_formato2.write(aux);


    aux=bd_num_colada[0].getStringValue();
    comp_num_colada.write(aux);

    aux=bd_num_paquete[0].getStringValue();
    comp_num_paquete.write(aux);

    aux=bd_num_piezas[0].getStringValue();
  comp_num_piezas.write(aux);
```