PRÁCTICAS PRESENCIALES

estudios abiertos
SEAS
GRUPO SANVALERO

AUTÓMATAS PROGRAMABLES AVANZADO

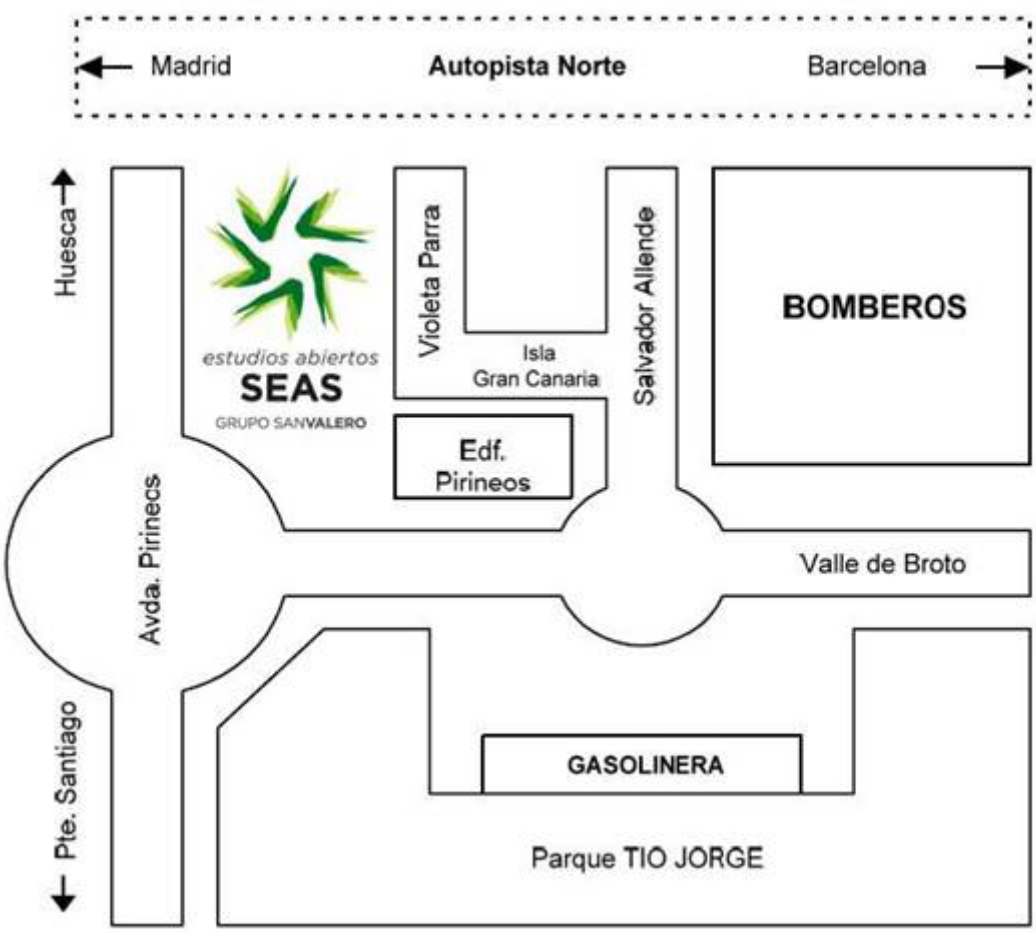
Área: Autómatas programables avanzado

LUGAR DE CELEBRACIÓN

Instalaciones de Centro San Valero, en c/ Violeta Parra 9.

50015 Zaragoza.

Horario: consultar la convocatoria de la práctica en Campus.



Aclaración:

Para las prácticas realizadas en c/ Violeta Parra 9 Centro San Valero, el acceso a las instalaciones se realizará por la entrada del edificio de Centro San Valero, no por la entrada del edificio de SEAS.

Entrada Fundación San Valero

Las líneas de autobús que tienen parada en las proximidades de Fundación San Valero son: 29, 36, 35, 45, 42 y Ci1.

Para más información visitar la página Web de Avanza. https://zaragoza.avanzagrupo.com/

DESCRIPCIÓN:

Durante la jornada presencial se pondrán en práctica los conocimientos adquiridos durante el estudio del módulo correspondiente a la asignatura de Autómatas Programables Nivel Avanzado.

REQUISITOS:

Es requisito para la realización de la práctica, haber cursado la asignatura de Autómatas Programables y haber trabajado las 4 primeras unidades didácticas de la asignatura de Autómatas Programables Nivel Avanzado.

OBJETIVOS DE LA PRÁCTICA:

- 1. Conocer los pasos a seguir a la hora de realizar la automatización de máquinas.
- 2. Conocer los elementos empleados en detección y su tratamiento por parte del PLC.
- 3. Conocer los actuadores y su tratamiento por parte del autómata.
- 4. Trabajar con módulos de función y módulos de datos.
- 5. Conocer el entorno de desarrollo de aplicaciones para autómatas SIEMENS.
- 6. Conocer los fundamentos de programación en lenguaje AWL.
- 7. Conocer los modos de estructuración y ordenación a la hora de programar
- 8. (Funciones).

PROPUESTA DE LA PRÁCTICA:

1. Automatización de una máquina. (Control de cintas transportadoras).
2. Direccionamiento de los distintos dispositivos a conectar al PLC.
3. Estructuración del programa. Uso de funciones y módulos de datos.
4. Desarrollo del programa.
5. Transferencia y modos de funcionamiento del autómata.
6. Pruebas de funcionamiento del sistema.

DESARROLLO DE LA PRÁCTICA:

Planteamiento

Basándonos en el diagrama técnico de la máquina y la secuencia de funcionamiento, deberemos realizar un estudio preliminar de cara a determinar las necesidades a nivel de equipamiento. En función del número de detectores y actuadores realizaremos el dimensionado de entradas y salidas necesarias en nuestro PLC.

Estructuración y programación

Una vez planteadas las pautas básicas se procederá a estructurar la aplicación mediante funciones, generando aquellas que sean necesarias mediante el entorno de desarrollo.

Transferencia y pruebas de funcionamiento

Una vez finalizada la programación de los distintos bloques de programa, se procederá a realizar la transferencia de estos al PLC, prestando atención a los distintos modos de funcionamiento de estos equipos.

Con el programa transferido a la memoria del autómata, simularemos mediante el equipo la funcionalidad del sistema.

BIBLIOGRAFIA RECOMENDADA:

- Manual de las asignaturas de Autómatas Programables y Autómatas Programables Nivel Avanzado de SEAS.

MATERIALES NECESARIOS:

Para la realización de las prácticas, se utilizarán los siguientes equipos, que se encuentran en el aula (sólo es necesario traer el guion de prácticas impreso):

- Autómata programable S7-300 CPU 314C – 2PN/DP.
- Ordenador personal con cable ethernet y entorno de desarrollo TIA Portal.

DURACIÓN DE LA PRÁCTICA:

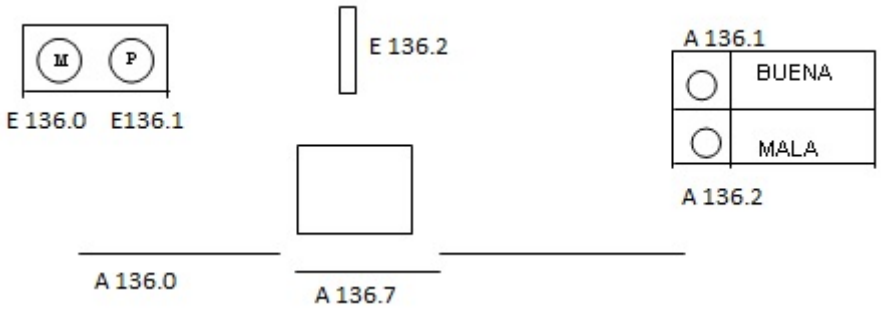
4 horas.

SOLUCIÓN DE LA PRÁCTICA:

Tenemos una cinta transportadora que lleva los productos. Lo que queremos es que los productos que se salgan del peso estipulado tanto por arriba como por abajo se desechen.

Además, tenemos unas luces indicadoras de producto bueno o de producto malo. Queremos que mientras está la pieza debajo de la célula detectora, se encienda la luz correspondiente.

Diagrama técnico:



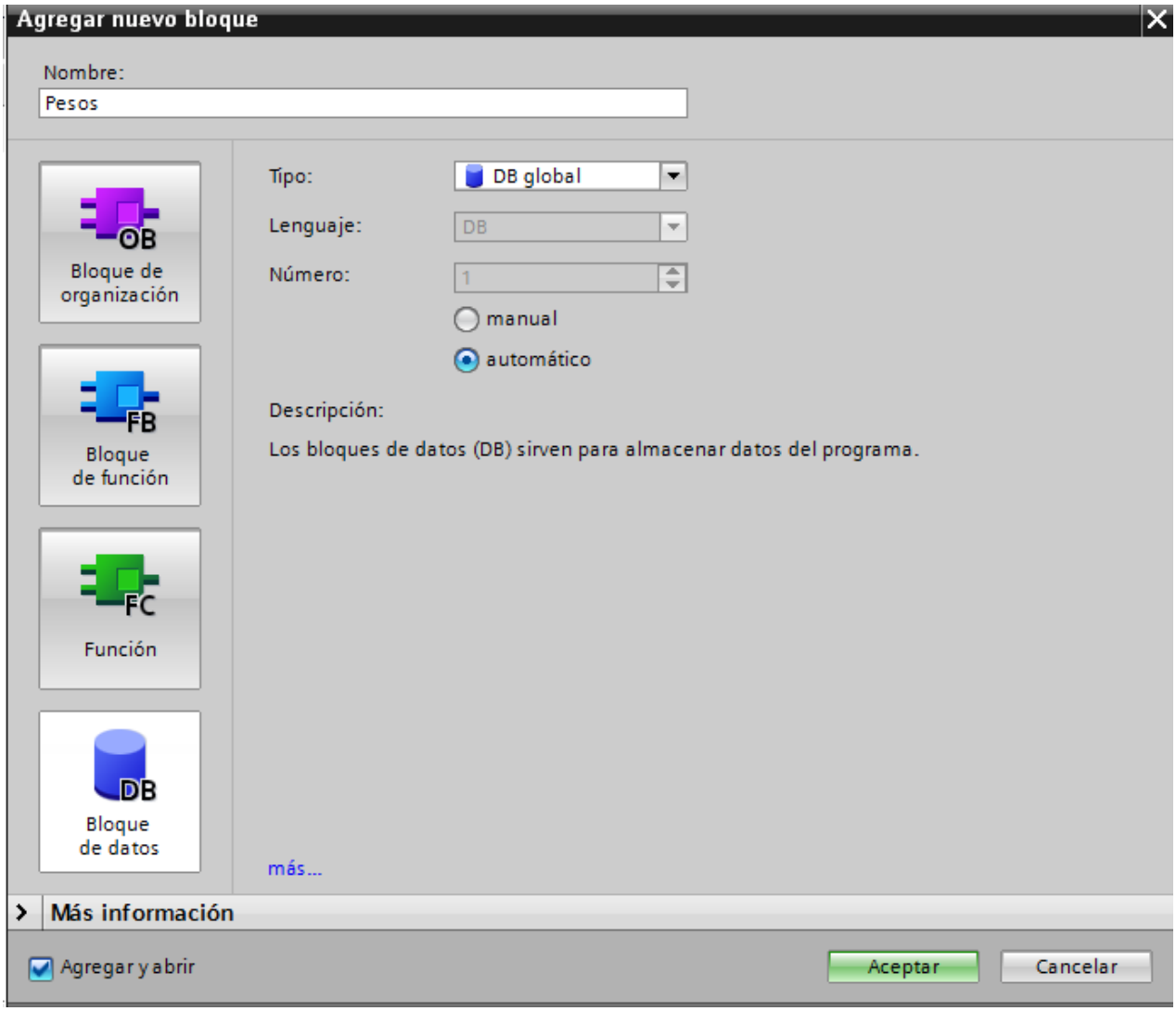
Para ello vamos a tener en un bloque de datos (DB) los límites de peso tanto superior como inferior.

Con el byte EB137 de entradas vamos a simular los pesos de las piezas que circulan por la cinta. Introduciremos los valores de forma binaria con cada bit de EB137.

Queremos que cuando la pieza tenga un peso comprendido entre los límites correctos se encienda una luz que nos indique que la pieza es correcta. Cuando pase una pieza que su peso se salga de los límites queremos que nos indique una luz que la pieza es mala y a la vez que se abra una trampilla para que la pieza salga de la cinta.

Teoría y ejemplo previo:

Vamos a crear un bloque de datos nuevo. En estos bloques lo único que vamos a introducir son datos. No vamos a programar nada. Va a ser una tabla con datos los cuales podemos leer y podemos escribir nuevos valores en los datos.



El primer campo será el peso máximo que puede tener la pieza y el segundo el peso mínimo. Establecemos unos valores de arranque, por ejemplo 64 y 32 respectivamente.

Pesos

	Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Visible en ..	Valor de a..	Comentario
1	Static							
2	Peso_Max	Int	...	64	☑	☑	☐	
3	Peso_Min	Int	...	32	☑	☑	☐	
4	<Agregar>				☐	☐	☐	

Ahora que hemos creado el DB vamos a realizar la programación que compare el peso de la pieza entre los límites del DB. Lo primero que debemos realizar es la asignación de simbólicos a las direcciones de entrada y salida, es decir, cumplimentar la tabla de variables.

Tabla de variables estándar

	Nombre	Tipo de datos	Dirección	Rema...	Visibl...	Acces...	Comentario
1	Marcha	Bool	%E136.0		☑	☑	
2	Paro	Bool	%E136.1		☑	☑	
3	Sensor	Bool	%E136.2		☑	☑	Sensor de pieza en posición pesaje
4	Peso	Byte	%EB137		☑	☑	Valor binario de peso pieza
5	Cinta	Bool	%A136.0		☑	☑	Cinta de arrastre de piezas
6	Luz_BUENA	Bool	%A136.1		☑	☑	Peso dentro de rango
7	Luz_MALA	Bool	%A136.2		☑	☑	Peso fuera de rango
8	Trampilla	Bool	%A136.7		☑	☑	Trampilla expulsora
9	<Agregar>				☑	☑	

Seguidamente realizaremos la programación en OB1.

Segmento 1: Marcha y paro

```
1  //Marcha
2        U     "Marcha"          %E136.0
3        S     "Cinta"           %A136.0   Cinta de arrastre de piezas
4
5  //Paro
6        U     "Paro"            %E136.1
7        R     "Cinta"           %A136.0   Cinta de arrastre de piezas
8
9  //Llega pieza
10       U     "Sensor"          %E136.2   Sensor de pieza en posición…
11       SPB   M001
12       R     "Luz_BUENA"       %A136.1   Peso dentro de rango
13       R     "Luz_MALA"        %A136.2   Peso fuera de rango
14       R     "Trampilla"       %A136.7   Trampilla expulsora
15       BEA
```

Segmento 2: Lectura peso

```
1  M001: NOP 0
2
3  //¿El peso es menor que el máximo?
4        L     "Peso"            %EB137    Valor binario de peso pieza
5        L     "Pesos".Peso_Max  DB1.???
6        <I
7        =     "Tag_1"           %M0.0
8
9  //¿El peso es mayor que el mínimo?
10       L     "Peso"            %EB137    Valor binario de peso pieza
11       L     "Pesos".Peso_Min  DB1.???
12       >I
13       =     "Tag_2"           %M0.1
```

Segmento 3: Dejo pasar o expulso

```
1  //Compruebo si estoy dentro de rango
2        U     "Tag_1"                %M0.0
3        U     "Tag_2"                %M0.1
4        S     "Luz_BUENA"            %A136.1   Peso dentro de rango
5        R     "Luz_MALA"             %A136.2   Peso fuera de rango
6
7  //Si no lo estoy expulso pieza
8        NOT
9        S     "Luz_MALA"             %A136.2   Peso fuera de rango
10       R     "Luz_BUENA"            %A136.1   Peso dentro de rango
11       S     "Trampilla"            %A136.7   Trampilla expulsora
```

Hemos visto cómo hacer la comparación de una señal analógica que leemos de forma binaria, pero en la práctica lo que haremos es leer una señal analógica. Trata de modificar este programa leyendo el canal 0 del PLC, que se corresponde a “PEW 800”.

Cuando modifiques el programa debes tener en cuenta que la entrada analógica PEW800 nos entrega un valor entre 0 y 27648, que debe ser escalado, por ejemplo, mediante la función de sistema FC105 (SCALE).

El valor que nos devuelve esta función es un número real, en lugar de un entero, lo que implica modificar las comparaciones.