Anexo TFG

Rafael Bello Ferrer
Titulación: Grado en ingeniería informática
Área del trabajo: Ingeniería del software

Consultor: Oriol Martí Girona
Profesor: Santi Caballe Llobet

Índice

1. Fases del ciclo de vida.

Las fases del ciclo de vida de un proyecto son las siguientes [1]:

1) Fase 1 (aprobación del proyecto): Se identifica un determinado problema y se conceptualiza en forma de proyecto. En esta fase se analiza la viabilidad técnica y económica del proyecto, además del análisis de los posibles riesgos durante el desarrollo del mismo. Al finalizar esta fase normalmente queda aprobada una propuesta de proyecto, que quizás haya competido entre otras propuestas que han sido finalmente descartadas.
2) Fase 2 (definición del proyecto): Se presenta documentación de propuesta del proyecto, donde consta una definición clara y concisa del proyecto a realizar y se analizan en detalle los requerimientos del mismo. En esta fase se presentarán los objetivos y el contexto del proyecto y se definirá una planificación inicial tanto en tiempo como en recursos para llevarlo a cabo. Por tanto, y habiendo realizado un pre-análisis ya de riesgos en la primera fase, aquí se profundizará en la identificación y análisis de los mismos, con el objetivo de determinar el alcance de su impacto durante el desarrollo del proyecto y emprender medidas correctoras en caso de ser necesario.
3) Fase 3 (planificación del proyecto): En esta fase se revisa y profundiza en la documentación de especificaciones y planificación del proyecto. Es definitiva, se trata de reconocer los hitos y hechos más importantes en el proyecto y situarlos en el tiempo previsto para llevarlo a cabo (planificación orientada a objetivos). En caso de que el proyecto sea realizado por un equipo, en esta fase se identificarán los roles y se distribuirá el trabajo para todo el equipo.
4) Fase 4 (ejecución del proyecto): En esta fase cobran vital importancia los trabajos de seguimiento y reporte de la evolución del proyecto. Esto conllevará la mayoría de las veces en la dedicación de recursos, tanto humanos como materiales, a la gestión de cambios e incidencias que puedan surgir durante el desarrollo del proyecto.
5) Fase 5 (cierre del proyecto): Realización de pruebas de rendimiento y funcionales del sistema, en el caso de este proyecto, prototipo, realizado durante las fases anteriores. Este es el momento de comprobar si el producto obtenido se corresponde y cumple con las especificaciones iniciales y los usuarios a los que va destinado lo aprueban. En esta fase se debe entregar la documentación completa del proyecto, incluyendo una evaluación técnica y económica de su desarrollo. Finalmente, también se entregará documentación relativa a los planes de mantenimiento y revisiones futuros (tanto a nivel software como a nivel hardware).

En esencia, estas 5 fases se pueden resumir en el siguiente diagrama:

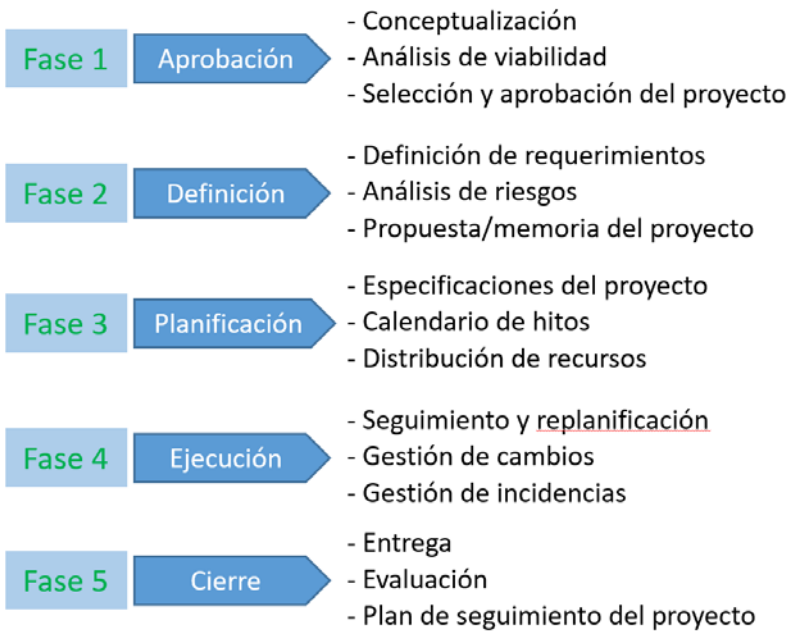


Figura 1. Fases y puntos clave del ciclo de vida de un proyecto [1]

2. Maqueta ferroviaria.

Este apartado describe brevemente la solución adoptada en cuanto al tamaño adecuado de la maqueta de modelismo para simular la infraestructura típica de un entorno ferroviario. La construcción de esta maqueta consta de un trabajo en marquetería utilizado como estructura base para disponer los elementos del sistema de control automatizado y los componentes de modelismo ferroviario en escala H0 que constituyen la representación miniaturizada de un posible escenario ferroviario real. Acompañando a esta base de marquetería se hace imprescindible adquirir material ferroviario para realizar la automatización de su control. La selección de escala es una de las primeras decisiones a tomar cuando se planifica la construcción de una maqueta de modelismo ferroviario. De hecho, la selección de una u otra escala determina la selección de un determinado fabricante u otro de modelismo especializado en sistemas ferroviarios. Es frecuente que un determinado fabricante de modelismo ferroviario a escala considere como no rentable la replicación del parque completo de una determinada operadora ferroviaria y, aún menos, su disponibilidad en todos los factores de escala estandarizados. Además, entre los clientes coleccionistas y aficionados los hay que prefieren centrarse en adquirir réplicas del material motor (locomotoras) y otros que prefieren dedicarse casi en exclusiva al material remolcado (vagones de pasajeros, mercancías, mantenimiento, …). Incluso, hay quien decide centrarse en material ferroviario de época, como por ejemplo locomotoras de vapor, y otros que prefieren los trenes modernos, como los trenes de alta velocidad y locomotoras eléctricas. Esta demanda ha introducido una cierta heterogeneidad de material ferroviario en las maquetas ferroviarias a nivel mundial. Esta diversidad de material demandada por los clientes propició que estos fabricantes de modelismo produjesen sus réplicas a escala cumpliendo unos estándares, y, permitiendo, al igual que ocurre en los sistemas ferroviarios reales, la coexistencia del material de diferentes fabricantes sobre un mismo escenario. Estos estándares determinan la escala de la carrocería del material rodante, el ancho de vía y el tipo de enganches de las máquinas locomotoras y vagones, entre otros detalles. Dicho de otra manera, la relación de reducción de material ferroviario en miniatura se expresa con la idea de “Escala”. De esta forma, existen varios estándares de escala respetados por la totalidad de los fabricantes de modelismo en general, y ferroviario en particular.

Uno de los puntos más importantes que determinan las proporciones de las escalas de modelismo ferroviario es el ancho de vía existente en la vida real. Estos anchos de vía “reales” se agrupan en cuatro

grupos para su conversión en modelos a escala, dando como resultado una relación entre la escala y el ancho de vía y designando esta relación como "trocha" o "galga". De esta forma, el estándar dicta que cuando la nomenclatura de la escala no incluye una letra "m", "e" o "i", entonces la escala se refiere a un ferrocarril de la vida real con un ancho de vía >1250, mientras que, si se trata de un ferrocarril real de vía estrecha, es decir, con ancho de vía <1250, entonces se adiciona a la nomenclatura de escala una letra "m", "e", o "i". Los siguientes ejemplos ilustran la forma de designar a las diferentes escalas existentes:

- Ejemplo 1: Réplica de material rodante de vía **normal** en escala 1:87 → Escala H0, vía H0, ancho de vía de 16,5 mm
- Ejemplo 2: Réplica de material rodante de vía **normal** en escala 1:160 → Escala N, vía N, ancho de vía de 9 mm
- Ejemplo 3: Réplica de material rodante de vía **métrica** en escala 1:45 → Escala 0, vía 0**m**, ancho de vía de 22,5 mm

Debido a que no en todos los países se utiliza el mismo ancho de vía "real", los fabricantes de modelismo deciden las proporciones del material ferroviario rodante para adaptarlo a una escala determinada.

Entre los factores de escala más conocidos destacan la escala H0 y la escala N, aunque existen otros factores de escala adicionales y depende de los fabricantes de modelismo adoptar unos u otros dependiendo incluso del país donde se encuentren.

La escala N ha sido considerada durante mucho tiempo como la "gran estrella" del modelismo para espacios reducidos. La proporción del estándar N para el material de esta escala es de 1:160, lo que conlleva que el ancho de vía sea de 9 mm. Los fabricantes de modelismo ferroviario en escala N son capaces de replicar con un nivel de detalle aceptable cualquier material ferroviario existente en la realidad, desde material rodante a cualquier tipo de infraestructura ferroviaria para montar una maqueta de gran realismo. A la vez, el reducido tamaño de los materiales y el aumento del catálogo de material de la inmensa mayoría de los fabricantes en esta escala año tras año, hace posible la construcción de un circuito de mayor tamaño que en otras escalas, como es el caso de la H0 que se introduce a continuación.

La escala H0 es, casi con toda probabilidad, la más extendida a nivel mundial. Su tamaño, pequeño pero suficiente como para replicar cualquier característica de los sistemas ferroviarios reales, la hace idónea para disfrutar de este hobby en un ámbito doméstico. Según el estándar H0, el factor de escala es de 1:87, lo que conlleva que el ancho de vía de sea de 16,5 mm. El catálogo de los fabricantes de modelismo en escala H0 es bastante amplio desde hace ya tiempo, renovando incluso su material con nuevos modelos cada año.

Vistas estas dos escalas, y siendo dos de las más utilizadas en España, es necesario tomar una decisión en cuanto a la escala a utilizar para seguir trabajando en la maqueta propuesta en este proyecto. La elección de una escala adecuada se basa fundamentalmente en los siguientes factores:

- Espacio disponible para la maqueta
- Escenario que se quiere representar, es decir, tener en cuenta el espacio de maqueta disponible para representar estaciones, recorridos y detalles con la mayor coherencia posible. Por ejemplo, si se representan grandes estaciones en un espacio reducido la maqueta va a quedar desproporcionada y va a transmitir falta de realismo.
- Coste económico del conjunto de la maqueta. Es importante tener en cuenta que el material ferroviario construido en las escalas más utilizadas, como son la N y la H0, seguramente tendrá en la mayoría de casos un precio más competitivo, puesto que la producción de los fabricantes es mayor debido a la gran demanda.
- Nivel de detalle del material ferroviario a escala: Aun utilizando la escala N, el material replicado presenta un nivel de detalle y realismo inferior al que podemos conseguir en la escala H0. La miniaturización de los componentes ayuda a la construcción de maquetas más grandes en menos

espacio, pero como contrapartida, debemos evaluar qué nivel de detalle queremos o necesitamos para nuestra maqueta. Un menor tamaño del modelo a escala implica más dificultad para el fabricante en la réplica de todos y cada uno de los detalles del modelo real replicado.

- Cantidad de material ferroviario existente en el mercado para cada escala: Ya se ha comentado anteriormente que los fabricantes se decantan por utilizar mayoritariamente unas u otras escalas dependiendo de varios factores, como son la demanda existente, el país donde se encuentren, la preferencia por unas u otras compañías ferroviarias de la vida real, las "épocas ferroviarias" que se quieren replicar, etc. Hoy por hoy, tanto la escala H0 en primer lugar, como la escala N en segundo lugar, son altamente populares y los fabricantes disponen de gran cantidad de material en ellas. Aunque a mayor distancia, los fabricantes también aumentan paulatinamente su stock de material en escalas Z, H0e, TT, etc.

  En cuanto a la escala, para este proyecto se han tenido en cuenta fundamentalmente las escalas N y H0. Los motivos para seleccionar estas escalas como preferentes son:

- Alta disponibilidad de material en los fabricantes europeos: Algunos fabricantes disponen de material en otras escalas, pero no es tan abundante como en las escalas N y H0.

- Precios del material más comedidos que en otras escalas: Al disponer el fabricante de más stock de material en las escalas N y H0, los precios suelen ser más competitivos que en otras escalas y también fomenta la competencia de precios y variedad entre fabricantes.

- Disponibilidad de espacio para la maqueta: Puesto que se dispone de un espacio comedido para el trabajo con la maqueta, es lógico seleccionar las escalas de menor tamaño como las preferentes a utilizar para el proyecto. Por otra parte, plantearse trabajar con escalas demasiado reducidas dificulta la utilización de parte del hardware de control, como, por ejemplo, los sensores o incluso la forma en la que se gestionan los cambios de vía. En este caso, tanto la escala N como la escala H0 disponen de un tamaño relativamente adecuado al espacio disponible para la maqueta. En concreto, la escala H0, aunque en el caso de este proyecto no permite por su tamaño crear un escenario ferroviario amplio, se adapta muy bien al espacio del que en este caso se dispone.

- Nivel de detalle y realismo a lograr: Sin lugar a dudas, trabajando en escala H0 los fabricantes son capaces de representar con gran nivel de detalle todo el material ferroviario disponible. Aunque en escala N el nivel de detalle es aceptable, es evidente que utilizar un tamaño mayor como el de la escala H0 proporciona más capacidad para recrear con mayor precisión cualquier detalle del material real utilizado como modelo.

Sopesando todos los puntos anteriores, ambas escalas aportan características muy acordes a los requerimientos económicos, de espacio y de disponibilidad de material necesarios para este proyecto. Sin embargo, la escala H0 aporta además un nivel de realismo mayor que la escala N del material ferroviario. Por tanto, la escala de modelismo ferroviario seleccionada es la escala H0. Por otra parte, y una vez diseñado el recorrido ferroviario a representar en la maqueta mediante el software WinTrack (se explica con detalle más adelante), se plantea todo el material ferroviario a escala necesario para proceder a su compra. Finalmente, la solución adoptada para este proyecto ha sido la compra de dos kits de iniciación de modelismo del fabricante Roco (Modelleisenbahn GmbH) en escala H0, uno de los fabricantes más prestigiosos en el mundo del modelismo ferroviario. La adquisición de los componentes de este fabricante proporciona parte del sistema hardware a controlar, y constituye la base principal para el desarrollo del proyecto. Este fabricante dispone de gran material para el "modelismo ferroviario", desde tramos de vía, semáforos y señalización en general, desvíos, locomotoras de varios tipos y épocas, vagones, piezas eléctricas y centralitas de control digital, réplicas de estaciones de tren y edificios, pasos a nivel, etc. El catálogo completo de productos se puede consultar en su página web:

http://www.roco.cc/es/home/index.html

Cabe destacar que también influye en esta decisión el hecho de que la compra de material a través de "kits de inicio" resulta más económica que comprando todo el material por separado. El inconveniente con esta decisión es que, por ejemplo, las vías con balasto incluidas en los kits de inicio son menos realistas que las vías sin balasto que luego son "personalizadas" por el maquetista, al ser el balasto de las primeras pre-fabricado en plástico. Sin embargo, el objetivo del proyecto radica en gran medida en la funcionalidad del prototipo para realizar pruebas de software de control y no tanto en la decoración de la propia maqueta. Por este motivo, la adquisición de los kits de inicio parece, en este caso, la decisión adecuada.

3. El sistema DCC.

En este apartado se desarrollará con más detenimiento el sistema DCC, ya que el material ferroviario utilizado en la realización del proyecto es de Roco y utiliza este sistema para su control, por lo que es necesario conocer su funcionamiento básico para la realización de este proyecto.

El sistema DCC: Protocolo DCC

La aplicación del control digital al modelismo ferroviario data de los años 80. Entre los primeros equipos que permitían la conducción simultánea de varios trenes se encuentra el creado por Lenz en 1980 (para varios fabricantes, entre ellos Arnol y Roco). A partir de 1985 Lenz comercializa su propio sistema "Digital Plus by Lenz", que no dejara de mejorar y que a partir de 1995 servirá de base a la NMRA (National Model Railroad Association) para sus normas de modelismo digital. Estas normas se denominan DCC (Digital Command Control).

La llegada de la tecnología digital a las maquetas ha supuesto un avance importante en el modelismo ferroviario, permitiendo dos mejoras básicas:

a) Mejorar el realismo del manejo de la maqueta.

b) Simplificar el montaje y cableado de la maqueta.

Los montajes en analógico requerían un elevado número de componentes y cableados complejos en comparación con la simplicidad del sistema digital en este terreno. Además, gracias al sistema digital se puede establecer un control por software que antes era impensable.

Se habla de sistemas analógicos cuando los valores o magnitudes que pueden tomar las variables o señales del sistema son de tipo continuo, como por ejemplo la temperatura. Por el contrario, se habla de sistemas digitales cuando las magnitudes que se manejan se representan a través de valores discretos en lugar de continuos (por ejemplo, un interruptor, el cual puede estar abierto o cerrado).

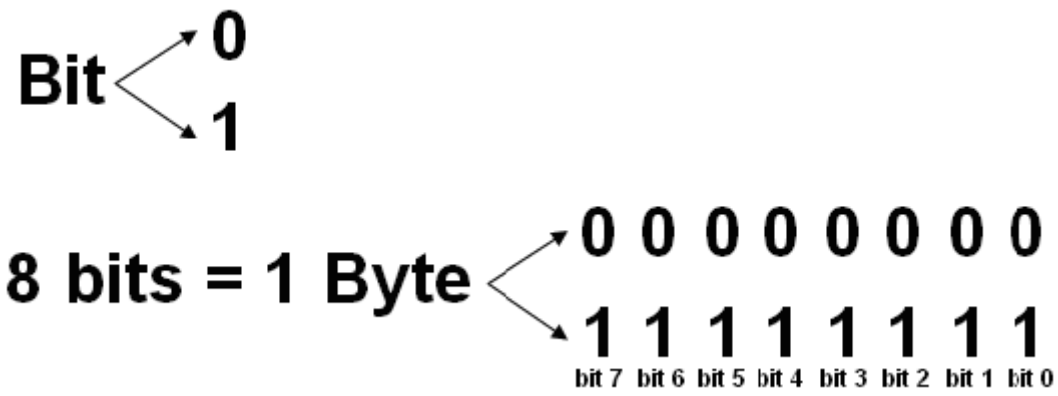
A pesar de su aparente limitación, estos dos estados que convencionalmente se representan por "0" y "1" y gracias a la lógica binaria, pueden ser utilizados para múltiples operaciones y funciones. En los sistemas digitales, como los ordenadores, estos dos estados en que se basa toda la lógica del sistema se representan utilizando dos niveles de tensión eléctrica ("bajo" y "alto") claramente diferenciados.

Los estados de una señal digital son capaces de representar la cantidad mínima de información que puede transmitirse. Es lo que en informática y comunicaciones se conoce como "bit".

Una secuencia de 8 bits se conoce como "byte". Un byte permite codificar, de forma "comprensible" para los sistemas informáticos y digitales 256 valores (un número decimal entre 0 y 255). Son estos valores en su forma binaria o decimal los que se utilizan para programar los descodificadores a través de sus "Variables de Configuración" (o CV's). Los bits de un byte se consideran de derecha a izquierda, de forma que el bit situado más a la derecha se considera el bit "menos significativo" y, en general, se denomina como bit 0, mientras que el bit situado más a la izquierda se considera como el bit "más

significativo" y se le suele denominar bit 7 (algunos fabricantes como Lenz numeran los bits de 1 a 8, en lugar de 0 a 7).

Para convertir un número binario en decimal, se debe sumar el producto de cada dígito binario, comenzando por la derecha, multiplicado por 2 elevado a la potencia correspondiente a su posición en el digito binario, comenzando por 0. Esto es más fácil si se considera que cada "bit" posee un "peso" específico dentro del byte. Este "peso" es lo que se utiliza para "traducir" un número binario a decimal (se multiplica el valor de cada bit por su peso y se suman los resultados). Se puede ver esto en la siguiente tabla:



BYTE								
Bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
"peso"	128	64	32	16	8	4	2	1
Decimal = (bi7 x 128) + (bit6 x 64) + (bit5 x 32)+ (bit4 x 16) + (bit3 x 8) + (bit2 x 4) + (bit1 x 2) + (bit0 x 1)								
Ejemplos 00000000 = 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0 10010110 = 128 + 0 + 0 + 16 + 0 + 4 + 2 + 0 = 150 11111111 = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255								

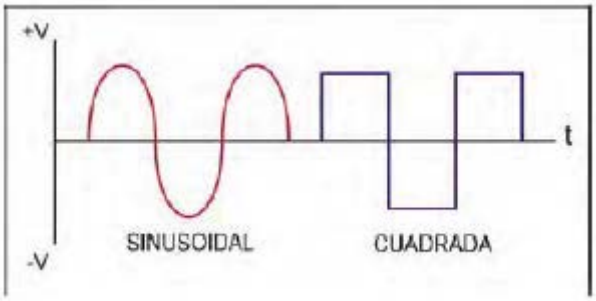
Convertir un número decimal a binario es algo más complicado, pero tampoco se necesita realizar esta conversión de forma habitual.

En el control "analógico" tradicional, las vías son alimentadas por un transformador cuyo voltaje de salida es posible variar de forma continua y en función del cual los motores adquieren más o menos velocidad. Como consecuencia todas las locomotoras que se encuentren en la misma vía reaccionarán de forma parecida, con pequeñas diferencias, dependientes de las características de sus motores y sistemas de transmisión.

Por el contrario, en un control digital las vías son alimentadas por una corriente de voltaje constante, sobre la que se superpone una codificación generada por la central digital. En la locomotora, el descodificador reconoce dichas señales, las interpreta (descodifica) y en función de las mismas suministra corriente al motor con el voltaje y polaridad necesarios para que se desplace según las órdenes recibidas desde la central. La posibilidad de "modular" una corriente eléctrica para que pueda transmitir información además de energía es conocida y utilizada desde hace mucho tiempo. De hecho, en modelismo ferroviario se han utilizado sistemas que utilizaban esta propiedad. Sin embargo, la mayoría de estos sistemas continuaban manejando señales de tipo analógico que producían problemas de

rendimiento y capacidad, limitando su uso. Por este motivo los sistemas actuales optan por una estrategia diferente: utilizar señales digitales.

Se utiliza corriente alterna, pero, a diferencia de la corriente alterna de cualquier entorno habitual de la actualidad (cuya polaridad varia en el tiempo de forma progresiva), se utiliza un tipo de corriente alterna que cambia de forma "brusca". Es decir, al contrario que la corriente alterna normal que genera una onda de tipo sinusoidal, en este caso se obtendrá una onda de forma cuadrada. Se puede comprobar esto en el siguiente gráfico:



Una onda cuadrada permite ser modificada (modulada) en amplitud o anchura de los pulsos. Utilizando los medios y sistemas adecuados, esta modificación de la anchura de los pulsos puede ser utilizada para transmitir información, de una forma parecida a lo que sucede con las ondas de radio. Las posibilidades de transmitir información por este sistema son muchas, pudiendo utilizarse múltiples "protocolos" o idiomas diferentes que, por desgracia, suelen resultar incompatibles entre sí.

Por ello, la NMRA (National Model Railroad Association), que agrupa a las asociaciones de modelismo americanas, decidió seleccionar uno de estos protocolos o lenguajes, concretamente el conocido como DCC (Digital Command Control), originalmente desarrollado por Lenz, como estándar.

Dicho estándar define una serie de parámetros "obligatorios" que todos los fabricantes deben necesariamente cumplir y que definen:

a) La forma de la señal eléctrica enviada a la vía, es decir, como se traduce, en forma de señal eléctrica, un uno y un cero.

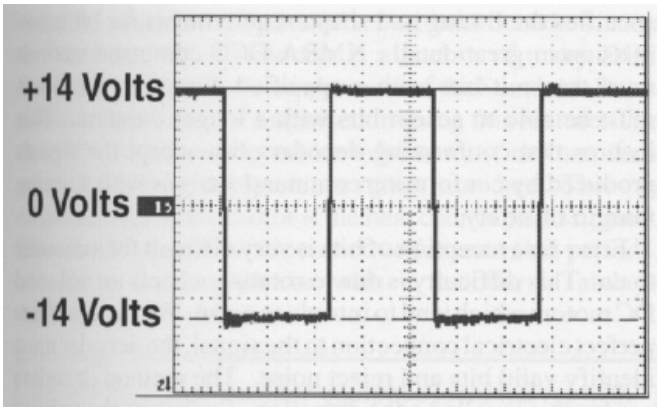
b) Cómo se codifican las órdenes en binario y como se envían.

c) Qué órdenes pueden intercambiarse entre el control y los receptores, qué significado tienen y qué información compone cada orden (también denominado "paquete" de información).

d) Una serie de parámetros "libres" no obligatorios, aunque recomendables. Estos parámetros permiten cierto juego a los fabricantes para implementar posibilidades no contempladas en el estándar DCC.

Además, existen una serie de normas o definiciones "recomendadas" que los fabricantes pueden utilizar para dotar a sus sistemas de más prestaciones sin salirse del estándar.

La señal eléctrica del sistema DCC es una onda cuadrada bipolar cuyos valores de pico de tensión dependen de la escala, aunque normalmente se establece en aproximadamente +14V y -14V, codificándose los valores alto y bajo (0 y 1) mediante un cambio en la amplitud o duración de los pulsos, es decir, un cambio de frecuencia.

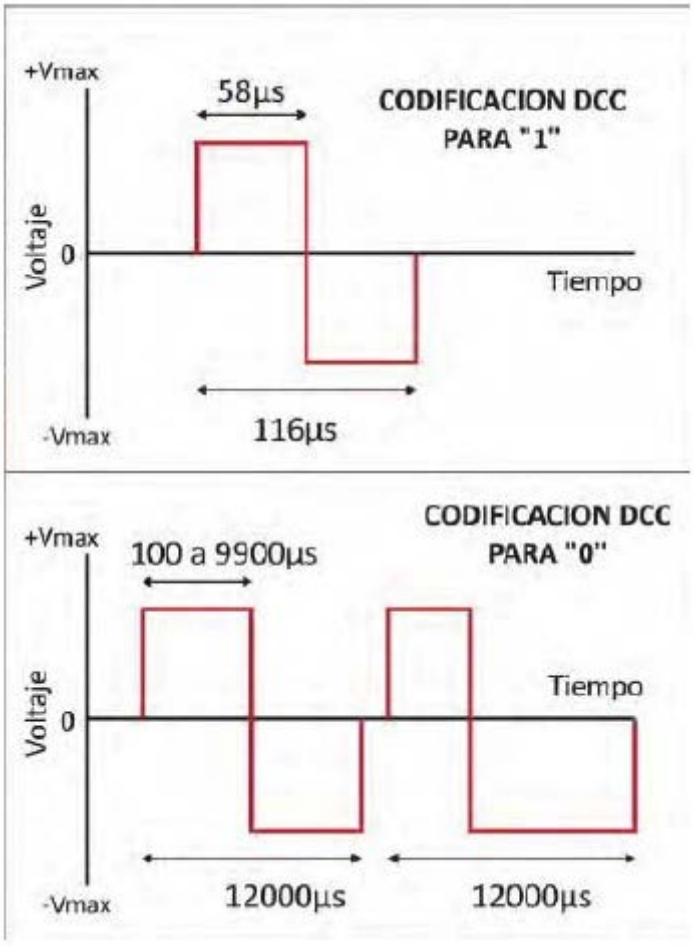


De esta forma, y con una cierta tolerancia, un bit “1” se identifica con un pulso corto y un bit “0” con uno largo. Esta identificación atiende a ciertas normas:

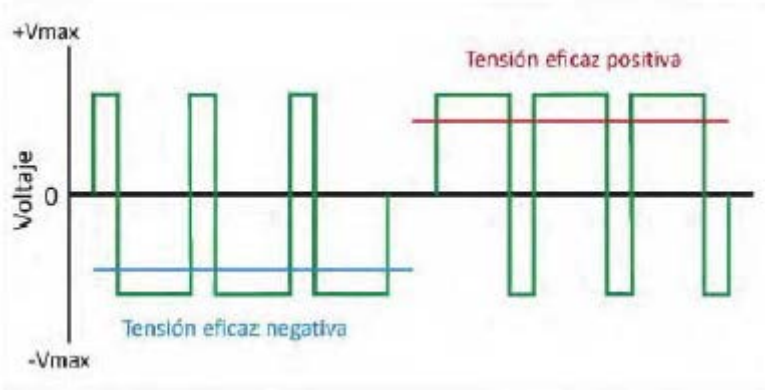
a) Un bit con valor 1 se codifica con una transición entre los máximos valores positivo y negativo, con una duración de 116 µs, en la que el pulso positivo y el negativo tienen la misma duración.

b) Un bit con valor 0 se codifica igualmente con una transición entre los máximos valores positivo y negativo, pero con una duración de cada pulso entre 100 y 9900 µs, en la que el pulso positivo y negativo pueden tener diferente duración, pero sin que la duración total exceda los 12000 µs.

Esta norma está recogida también por las normas NEM europeas (NEM 670 que corresponde al Standard NMRA S9.1).

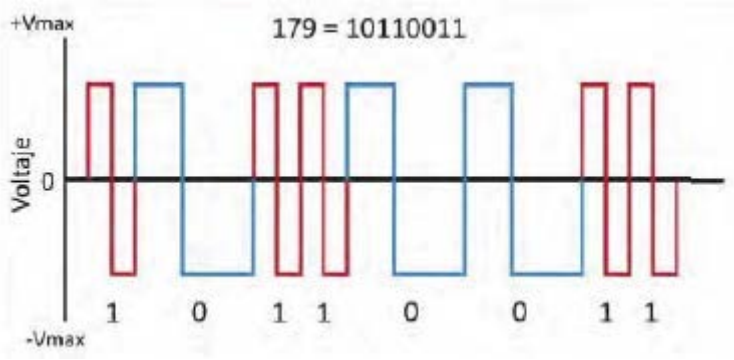


Dado que en el sistema DCC las vías están permanentemente bajo tensión alterna y que el decodificador instalado en cada locomotora es el encargado, en función de las órdenes recibidas, de alimentar con un voltaje mayor o menor el motor de la locomotora, a priori resulta imposible hacer circular una locomotora analógica. Sin embargo, existe una posibilidad real, aunque poco recomendable, de hacer funcionar una única locomotora analógica en un sistema digital DCC. Esta se basa en el concepto de "tensión eficaz" de la corriente alterna y a ello se debe el que la norma permita que en el caso de los bits "0" la duración de la semionda positiva o negativa sea diferente.



La diferente duración de la semionda positiva o negativa del bit "0" permite generar una corriente eficaz cuyo voltaje y polaridad se pueden modificar. De esta forma se consigue controlar la velocidad y sentido de una locomotora analógica sin interferir con el funcionamiento de las locomotoras dotadas de decodificador. Esta posibilidad, disponible a través de la dirección "0", no es recomendable, pues debido a las características de la corriente se generan sobre el motor muchos efectos indeseables (calentamiento, vibraciones, ruidos) que pueden dañar el motor de la locomotora analógica. Sin embargo, no existe este problema en las locomotoras digitales, dado que el motor no es alimentado directamente con la corriente de vía. En el sistema digital la corriente de la vía, antes de llegar al motor de la locomotora, es previamente transformada y filtrada por el decodificador en una corriente, también pulsante, pero regular y de alta frecuencia que evita estos efectos. Esto es similar a la corriente generada por los controladores analógicos de corriente pulsante diseñados para mejorar el rendimiento de los motores. De hecho, la mayoría de fabricantes incorporan en los decodificadores la posibilidad de modificar las características de la corriente de salida del decodificador para mejorar el rendimiento de los diferentes tipos de motores.

En la siguiente gráfica se puede comprobar lo explicado hasta el momento:



La corriente generada puede contener información, de forma que, por ejemplo, el número 179 = 10110011 se representaría por la secuencia de pulsos representada en la gráfica. Con lo visto hasta el momento ya se pueden enviar instrucciones a través de las vías, sin embargo, falta por determinar las condiciones de recepción e interpretación por parte del decodificador. Cuando se envía información de esta forma, se debe garantizar que esas instrucciones sean reconocidas sin errores por el decodificador al que van dirigidas y éste, además, debe saber que lo que ha recibido no contiene errores. Para ello, a los bits que contienen las instrucciones propiamente dichas, se les añaden otras señales de control. A la secuencia de órdenes propiamente dicha se le denomina "paquete".

Una secuencia de control básica DCC (paquete DCC de acuerdo con el estándar NMRA S-92 → Communications Standards for DCC) estaría formada por los siguientes elementos:

1. **PREAMBLE** → Es una secuencia de al menos 10 bits con el valor "1". Sirve para "avisar" de que va a ser enviado un mensaje. Realmente esta secuencia de bits "1" sirve para que los decodificadores identifiquen y se sincronicen con el principio del paquete. Las centrales digitales deben enviar un mínimo de 14 bits "1" completos en el preámbulo. Por su parte, los decodificadores no deben admitir como preámbulos válidos aquellos con menos de 10 bits "1" (lo comentado anteriormente), ni deben requerir más de 12 bits "1" completos para recibir correctamente el paquete.
2. **PACKET STAR BIT** (Bit de inicio del paquete) → Es un bit de valor "0" que tras la secuencia del preamble, indica el comienzo de las órdenes.
3. **PAQUETE DE DATOS** → De tamaño variable entre 3 y 6 bytes. Está formado por:
    a. ADDRESS DATA BYTE (Byte de dirección): Conjunto de 1 o 2 bytes con la "dirección" del decodificador al que van destinadas las instrucciones, de forma que estas sean obedecidas exclusivamente por este y no por el resto. El hecho de identificar a los destinatarios mediante direcciones implica que los decodificadores deben ser capaces de retener y reconocer su propia dirección, que debe poder ser fácilmente configurable por el usuario. Las centrales digitales pueden restringir el rango de direcciones válidas soportadas, de manera que no estén disponibles todas a las que el campo dirección podría dar lugar, siempre que este hecho se indique claramente en la documentación.
    b. INSTRUCTION DATA BYTE (Byte de instrucciones): Conjunto de 1 a 3 bytes con las instrucciones propiamente dichas. También puede contener una dirección o un dato, o incluso información para detección de errores.
    c. ERROR DATA BYTE (Byte de error): Es un código de control que permite al decodificador determinar si el paquete se ha recibido correctamente.

    NOTA: Cada uno de estos tres elementos se separan por un bit a "0".
4. **PACKET END BIT** (Bit de fin de paquete): Es un bit de valor "1" que indica el final de la secuencia, pudiendo ser también el primer bit del preámbulo del paquete siguiente si no se utilizan bits de relleno entre paquetes.

Un ejemplo gráfico de paquete básico DCC puede ser el siguiente:

Preámbulo	0	Byte de dirección	0	Byte de datos	...	1

Entre el byte de datos y el bit "1" final puede haber repetición de bytes de datos separados por bits "0".

El estándar NMRA S-92 (Communications Standards for DCC) define el siguiente conjunto de 4 paquetes básicos que constituyen el mínimo exigible para asegurar interoperabilidad entre sistemas diferentes y cuya estructura, que es conforme a la general de un paquete DCC es:

111111111111	0	0AAAAAAA	0	01DCSSSS	0	EEEEEEEE	1

- Velocidad y dirección para decodificadores de locomotora:

111111111111	0	0AAAAAAA	0	01DCSSSS	0	EEEEEEEE	1

El byte de dirección (0AAAAAAA) contiene la dirección del decodificador destinatario. Con los 7 bits disponibles para codificar la dirección y teniendo en cuenta las direcciones reservadas, se admiten direcciones en el rango 1-127. El byte de instrucción (01DCSSSS) contiene la información de velocidad y sentido para el decodificador. Consta del campo fijo '01' que identifica la instrucción de velocidad/dirección, el bit D de dirección (el valor '1' indica hacia delante), cuatro bits para codificar el paso de velocidad y un bit C que puede servir como bit adicional (el menos significativo) para codificar el paso de velocidad (para el caso de usar 28 pasos) o para el control de las luces de la locomotora (caso del uso de 14 pasos de velocidad en modo compatible con modelos antiguos de decodificador). En el caso de usar sólo los bits SSSS para codificar la velocidad, de los 16 valores posibles se reservan dos para indicar parada (normal y de emergencia), con lo que quedan 14 pasos posibles para el uso normal. En el caso de usar los bits CSSSS, de los 32 valores posibles, se reservan 4 (dos para parada normal y dos para parada de emergencia), con lo que quedan 28 posibles para el uso normal. A continuación, se incluye la tabla de codificación de pasos de velocidad. En los casos de parada en los que el bit C='1', el bit de dirección puede ignorarse para las funciones sensibles a la dirección.

CSSSS	**Velocidad**	**CSSSS**	**Velocidad**	**CSSSS**	**Velocidad**	**CSSSS**	**Velocidad**
00000	Parada	00100	Paso 5	01000	Paso 13	01100	Paso 21
10000	Parada	10100	Paso 6	11000	Paso 14	11100	Paso 22
00001	Parada de emergencia	00101	Paso 7	01001	Paso 15	01101	Paso 23
10001	Parada de emergencia	10101	Paso 8	11001	Paso 16	11101	Paso 24
00010	Paso 1	001100	Paso 9	01010	Paso 17	01110	Paso 25
10010	Paso 2	10110	Paso 10	11010	Paso 18	11110	Paso 26
00011	Paso 3	00111	Paso 11	01011	Paso 19	01111	Paso 27
10011	Paso 4	10111	Paso 12	11011	Paso 20	11111	Paso 28

El byte de detección de errores (EEEEEEEE) se calcula realizando la operación OR exclusivo (0 XOR 0 =0, 0 XOR 1 = 1, 1 XOR 0 = 1, 1 XOR 1 = 0) bit a bit de los bytes de dirección y velocidad. Todo decodificador que recibe un paquete, debe calcular el resultado y compararlo con el valor recibido. Si la comparación no es exacta, puede ignorar el paquete recibido.

- Reset (para todos los decodificadores):

111111111111	0	00000000	0	00000000	0	00000000	1

Cuando un decodificador recibe este paquete debe parar inmediatamente la locomotora que tenga bajo su control, borrar su memoria volátil (incluidos datos de dirección y velocidad) y volver al estado normal tras un arranque. Tras el envío de un paquete de reset, hasta transcurridos 20 mseg, la central no debe enviar paquetes con la dirección en el rango 01100100-01111111 (100-127), salvo que se pretenda entrar en el modo de servicio.

- Vacío (para todos los decodificadores):

111111111111	0	11111111	0	00000000	0	11111111	1

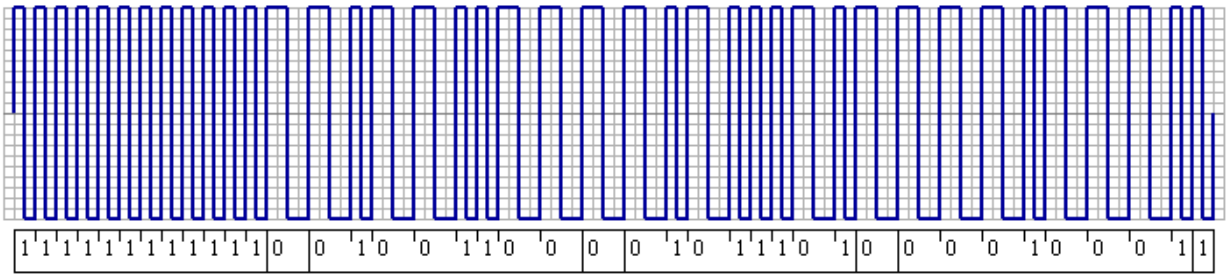
Cuando un decodificador recibe este paquete, que incluye un byte de dirección con valor 255 (reservado), no debe hacer nada, aunque para él debe contar como si se tratara de un paquete normal dirigido a otro decodificador. Este paquete es un paquete típico "de relleno" utilizado por las centrales digitales cuando no tienen otros paquetes que transmitir y desean mantener en modo digital a los decodificadores que puedan estar en la instalación.

- Parada general (para todos los decodificadores):

111111111111	0	00000000	0	01DC000S	0	EEEEEEEE	1

Si S='0', todos los decodificadores realizarán una parada normal (de acuerdo con la tasa de frenado con la que se hayan configurado) del motor de la locomotora que controlen. Si S = '1', la parada será de emergencia (corte inmediato de la tensión al motor). En este caso, si C = '1', puede ignorarse opcionalmente el valor del bit de dirección D para las funciones sensibles a la dirección.

La siguiente figura muestra un paquete que indica a la locomotora 76 que se ponga en el paso de velocidad 24 marchando hacia atrás.



En el documento de prácticas recomendadas NMRA RP-9.2.1 DCC Extended Packet Formats se extiende el conjunto de paquetes básicos con un amplio conjunto de paquetes en formato extendido. Hay que indicar que, aunque dicho documento no es un estándar NMRA, la mayoría de los fabricantes incorporan en sus centrales y decodificadores lo incluido en él, habida cuenta de la ampliación de capacidades de control que ello supone (direcciones extendidas, modo de servicio, control de multitracción, 128 pasos de velocidad, etc). A continuación, se describe el conjunto de formatos de paquete extendidos, que siguen la norma general en cuanto a estructura de un paquete DCC válido, pero en vez de incluir 3 bytes de datos separados por '0', pueden incluir entre 3 y 6 bytes de datos separados por '0'. Para la descripción se utiliza:

- A para definir un bit del campo de dirección.
- C para denotar un bit de un campo de instrucción.
- D para denotar un bit de un campo de datos.
- U para denotar un bit de valor indefinido (válido '0' o '1').
- E para denotar un bit de detección de error.

El primer byte de todo paquete de formato extendido contiene la dirección primaria de su destinatario. Esta dirección primaria puede tener valores dentro del siguiente espacio de direcciones (hay que aclarar que, para el caso de direcciones con más de 7 bits, el valor correspondiente a los bits de la dirección primaria debe completarse con bits adicionales del siguiente byte en el paquete, tal como se muestra en cada caso):

- Dirección 00000000 (0): Dirección especial de difusión (broadcast).
- Direcciones 00000001 a 01111111 (1 a 127): decodificadores multifunción con direcciones de 7 bits.

$0A_6A_5A_4A_3A_2A_1A_0$

Total de direcciones disponibles: 127

- Direcciones 10000000 a 10111111 (128 a 191): decodificadores de accesorios básicos con direcciones de 9 bits y decodificadores de accesorios extendidos con direcciones de 11 bits.

$10A_5A_4A_3A_2A_1A_0$	0	$1A_8A_7A_6$CDDD

Total de direcciones disponibles: 512 (0 a 511)

$10A_5A_4A_3A_2A_1A_0$	0	$0A_{10}A_9A_80A_7A_61$

Total de direcciones disponibles: 2048 (0 a 2047)

- Direcciones 11000000 a 11100111 (192 a 231): decodificadores multifunción con direcciones de 14 bits.

$11A_5A_4A_3A_2A_1A_0$	0	$A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6$

Total de direcciones disponibles: 16383 (1 a 16383)

- Direcciones 11101000 a 11111110 (232 a 254): reservado para uso futuro.
- Dirección 11111111 (255): paquete vacío.

El último byte de todo paquete en formato extendido es el de detección de errores, que se calcula mediante la operación de OR exclusivo bit a bit sobre todos los bytes de dirección y datos. Se aplica lo indicado en el apartado de paquetes básicos DCC, para el caso de no concordancia entre el valor calculado por un decodificador y el recibido. Se entiende por decodificador multifunción al que se usa para controlar uno o más motores y/o funciones. Por su parte, un decodificador de accesorios es un dispositivo capaz de controlar un determinado número de funciones simples (desvíos, luces, etc) si es del tipo básico o capaz de controlar señales luminosas u otros accesorios complejos, si es del tipo extendido.

- *Paquetes con formato extendido para decodificadores multifunción*
  - Comando de difusión para decodificadores multifunción:

111111111111	0	00000000	1, 2 ó 3 bytes de instrucción	0	EEEEEEEE	1

La instrucción contenida en los bytes centrales del paquete debe ser ejecutada por todos los decodificadores que lo reciban. Aquellos que no implementen la instrucción correspondiente, pueden ignorarla.

  - Paquetes de instrucción para decodificadores multifunción:

    Tienen la siguiente estructura, que varía según la extensión del campo de dirección:

111111111111	0	$0A_6A_5A_4A_3A_2A_1A_0$	1, 2 ó 3 bytes de instrucción	0	EEEEEEEE	1

    Decodificadores con direcciones de 7 bits:

111111111111	0	11$A_5A_4A_3A_2A_1A_0$	0	$A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6$	1, 2 ó 3 bytes de instrucción	0	EEEEEEEE	1

Decodificadores con direcciones de 14 bits.

La instrucción contenida en el paquete tiene, según sea de longitud 1, 2 ó 3 bytes, la siguiente estructura:

`CCCDDDDD` `CCCDDDDD,0,DDDDDDDD` `CCCDDDDD,0,DDDDDDDD,0,DDDDDDDD`

Es decir, siempre hay un campo inicial de 3 bits que codifica el tipo de instrucción y 5 bits adicionales de datos, a los que pueden añadirse 8 bits o 16 bits de datos adicionales. La tabla siguiente muestra los tipos de instrucción disponibles en el estándar:

	CCC	**Tipo de instrucción**
A	000	Control de decodificador y de multitracción
B	001	Instrucciones de operación avanzada
C	010	Instrucción de velocidad y dirección para operación marcha hacia delante
D	011	Instrucción de velocidad y dirección para operación marcha hacia atrás
E	100	Instrucción de funciones grupo 1
F	101	Instrucción de funciones grupo 2
G	110	Expansión futura
H	111	Instrucción de acceso a variable de configuración (CV)

A continuación, se detallan todos ellos:

- Control de decodificador y de multitracción (000):

  Con la excepción de la función de acuse de recibo (acknowledgment) del decodificador (00001111), en un paquete sólo puede incluirse una sola instrucción de control de decodificador y de multitracción.

  - Control del decodificador (0000)

    Tiene los dos posibles formatos siguientes: 0000CCCD y 0000CCCD,0,dddddddd

    En cualquiera de los casos, esta instrucción se usa para fijar o modificar la configuración interna de los decodificadores, de manera que algunas de sus características, en función del valor de D, se activen o desactiven.

CCC	**Acción**
000	Con D='0', reset "en caliente" del decodificador: borrado de su memoria volátil (incluyendo datos de dirección y velocidad) y retorno a los valores de encendido. Con D='1', reset "en frío" del decodificador: las CVs 29, 31 y 32 se devuelven a sus valores de fábrica, la CV19 se pone a '00000000' y se hace un reset "en caliente".
001	Instrucción de prueba de fábrica (*Factory Test*). No debe usarse en operación normal.

010	Reservada para uso futuro
011	Activar *flags* de decodificador o grupo de decodificadores. Su formato es 0000011D 0 cccc0aaa En función de los valores de cccc y de D, puede deshabilitarse/habilitarse la petición de acuse de recibo, activarse y fijar la comunicación bidireccional y aceptar o no las instrucciones de acceso a CV. aaa es la subdirección del decodificador (su valor se fija en la CV31)
100	Reservada para uso futuro
101	Activar direccionamiento avanzado (bit 5 de CV29)
110	Reservada para uso futuro
111	Con D='1' petición de acuse de recibo del decodificador

- Control de multitracción (0001)

  Sirve para establecer multitracciones, activarlas y desactivarlas y tiene el formato 0001CCCC,0,0AAAAAAA

  El valor ‘1’ del segundo byte se reserva para uso futuro, CCCC es la (sub)instrucción de control de multitracción y AAAAAAA es la dirección de la multitracción. Si se usa el valor ‘0000000’, la multitracción se desactiva y si se usan valores entre 1 y 127, la multitracción se activa. Al activarse una multitracción su dirección se almacena en los bits 0 a 6 de la CV19 y el bit 7 de la CV19 se pone a ‘1’.

  Cuando una multitracción está activada, el decodificador ignorará cualquier instrucción de dirección y velocidad dirigida a su dirección base (salvo que ésta coincida con la de multitracción). Las funciones controladas por las instrucciones ‘100’ y ‘101’ continuarán respondiendo en la dirección base del decodificador y responderán a la dirección de la multitracción si están activados los valores apropiados en las CV21 y CV22.

  Las (sub)instrucciones de control de multitracción son:

  1) CCCC=’0010’ (00010010). Fija la dirección de la multitracción con el valor del siguiente byte y se activa la multitracción con dirección de marcha hacia delante (o la desactiva si la dirección es ‘0000000’).
  2) CCCC=’0011’ (00010011). Fija la dirección de la multitracción con el valor del siguiente byte y activa la multitracción con dirección de marcha atrás (o la desactiva si la dirección es ‘0000000’).
  3) El resto de los valores CCCC se reservan para uso futuro.

- Instrucción de operación avanzada (001)

  Esta instrucción, que no puede repetirse dentro de un mismo paquete y que permite el acceso a funciones avanzadas del decodificador, tiene la estructura 001CCCCC,0,DDDDDDDD. De las 32 (sub)instrucciones posibles a las que los 5 bits pueden dar lugar, sólo están definidas las siguientes, quedando el resto reservadas para uso futuro:

  - CCCCC=’11111’. Control de velocidad con 128 pasos. Sirve para enviar al decodificador la orden de variación de velocidad con el formato 00111111,0,d$s_6s_5s_4s_3s_2s_1s_0$, donde d es la dirección (‘0’ hacia delante, ‘1’ hacia atrás) y

$s_6s_5s_4s_3s_2s_1s_0$ el valor en binario del paso de velocidad (126 valores posibles, pues U0000000 se utiliza para indicar parada y U0000001 para parada de emergencia).

- CCCCC=’11110’. Instrucción de pasos de velocidad restringidos. Sirve para restringir la velocidad máxima del decodificador y tiene el formato 00111110,0, d$U_{s_6s_5s_4s_3s_2s_1s_0}$, donde d=’0’ indica habilitar operación restringida de velocidad, d=’1’ deshabilitarla y $s_5s_4s_3s_2s_1s_0$ son los pasos de velocidad tal como se definen en el caso de los paquetes básicos (para el modo de 128 pasos, se escala a 28 pasos).

- Instrucción de dirección y velocidad marcha adelante (010)

  Tiene el formato 010DDDDD y se utiliza para enviar información de control a los motores conectados a los decodificadores multifunción en modo marcha hacia delante. Los bits 0 a 3 codifican el paso de velocidad, según lo definido en los paquetes básicos. Si el bit 1 de la CV29 tiene el valor ‘1’, el bit 4 de la instrucción sirve para codificar un paso de velocidad intermedio (tal como se define en los paquetes básicos). Si el bit 1 de la CV29 tiene el valor ‘0’, el bit 4 de la instrucción sirve para controlar la función F0 (luces). En este modo, U0000 significa parada, U0001, parada de emergencia, U0010 es el primer paso de velocidad y U1111 el de velocidad máxima.

- Instrucción de dirección y velocidad marcha atrás (011)

  Esta instrucción, con formato 010DDDDD, es idéntica a la anterior, pero para el caso de marcha atrás.

- Instrucción de grupo de funciones 1 (100)

  Tiene el formato 100DDDDD y permite controlar hasta 5 funciones (F0 y F1-F4). Los bits 0 a 3 controlan respectivamente las funciones F1 a F4, de manera que un valor ‘1’ significa función activada y un valor ‘0’ función desactivada. Si el bit 1 de la CV29 está a ‘1’, entonces el bit 4 de la instrucción controla la función F0 (luces). Si el bit indicado de la CV29 está a ‘0’, entonces el bit 4 de la instrucción no tiene significado y las luces se controlan con las instrucciones de velocidad y dirección (010 y 011).

- Instrucción de grupo de funciones 2 (101)

  Tiene el formato 101SDDDD y permite controlar hasta 8 funciones adicionales (F5-F12). El bit 4 (S) define el uso de los bits 0 a 3 (DDD). Si S=’1’, los bits 0 a 3 definen, respectivamente, el estado activado (valor ‘1’) o desactivado (valor ‘0’) de las funciones F5 a F8. Si S=’0’, los bits 0 a 3 definen, respectivamente, el estado activado o desactivado de las funciones F9 a F12. Los bits 0 a 3 controlan respectivamente las funciones F1 a F4, de manera que un valor ‘1’ significa función activada y un valor ‘0’ función desactivada. Si el bit 1 de la CV29 está a ‘1’, entonces el bit 4 de la instrucción controla la función F0 (luces). Si el bit indicado de la CV29 está a ‘0’, entonces el bit de la instrucción no tiene significado y las luces se controlan con las instrucciones de velocidad y dirección (010 y 011).

- Instrucción para expansión futura (110).

  Tiene el formato 110DDDDD,0,DDDDDDDD y su uso se reserva para el futuro.

- Instrucción de acceso a CV (111)

  Sirve para fijar o modificar CVs del decodificador ya sea desde la vía de programación o desde la vía normal. Hay dos formas de instrucción (corta y larga), según se desee acceder a CVs seleccionadas de acceso frecuente o verificar/modificar cualquier CV. En cualquiera de los casos, sólo puede haber una instrucción de acceso a CV por paquete.

  - Instrucción de acceso a CV corta

Tiene la estructura 1111CCCC,0,DDDDDDDD, donde DDDDDDDD es el valor que se coloca en la CV identificada por CCCC. Por ahora sólo están definidos los valores siguientes, quedando el resto reservados para uso futuro.

CCCC	
0000	Uso no permitido
0010	Valor de aceleración (CV23)
0011	Valor de deceleración (CV24)

- Instrucción de acceso a CV larga

  Sirve para la manipulación directa de cualquier CV y tiene la estructura:

  1110CCA$_9$A$_8$,0,A$_7$A$_6$A$_5$A$_4$A$_3$A$_2$A$_1$A$_0$,0,DDDDDDDD.
  La CV destino de la instrucción es el valor indicado por A$_9$A$_8$A$_7$A$_6$A$_5$A$_4$A$_3$A$_2$A$_1$A$_0$ más 1 (es decir, la CV3 se codifica 0000000010). La operación para realizar se codifica con los valores de CC, tal como se indica en la tabla:

CC	Operación
00	Reservado para uso futuro
01	Verificar byte Se compara la CV destino con el valor DDDDDDDD
10	Escribir byte Se escribe en la CV destino el valor DDDDDDDD. Para que la escritura tenga efecto, el decodificador debe recibir dos paquetes de escritura idénticos
11	Manipulación de bit Tiene el formato 111011A$_9$A$_8$,0,A$_7$A$_6$A$_5$A$_4$A$_3$A$_2$A$_1$A$_0$,0,111DCAAA, donde AAA indica la posición del bit dentro de la CV y C el tipo de operación ('0' verificar y '1' escribir)

- *Paquetes con formato extendido para decodificadores de accesorios*

  Como ya se ha indicado, de acuerdo con la NMRA, un decodificador de accesorios es un dispositivo capaz de controlar un determinado número de funciones simples (desvíos, luces, etc). Para posibilitar el manejo de un gran número de dispositivos, se permite que un decodificador de accesorios pueda responder a una o varias direcciones. Cada dirección de decodificador controla 4 pares de salidas (equivalentes a 8 salidas individuales), cada una de las cuales puede activarse permanentemente o durante un periodo de tiempo configurable (CV515 a CV518). La desactivación puede realizarse en cualquier momento.

  - Formato de paquete para decodificadores de accesorios básicos (activación y desactivación de accesorios)

111111111111	0	10A$_5$A$_4$A$_3$A$_2$A$_1$A$_0$	0	1~~A$_8$A$_7$A$_6$~~CDDD	0	EEEEEEEE	1

  Los 6 bits menos significativos de la dirección del decodificador destinatario del paquete (A$_5$A$_4$A$_3$A$_2$A$_1$A$_0$) se codifican en el segundo byte; los tres más significativos, en complemento a uno (A$_8$A$_7$A$_6$) y en el tercero. El bit C del tercer byte indica si la salida debe activarse (“1”) o desactivarse (“0”). El tiempo de activación de cada par de salidas

se configura en las CV515 a CV518 (0 indica activación permanente). Los bits DDD del tercer byte identifican la salida (los dos primeros DD identifican el par (1-4) y el tercero la salida dentro del par). Ejemplo: paquete de activación de la salida 1 del tercer par del decodificador de dirección 157.

`111111111111 0 10011101 0 11011100 0 01000001 1`

(CDDD=1100 y 157 es 010011101, de donde $A_5A_4A_3A_2A_1A_0$=011101 y $A_8A_7A_6$=101)

- Formato de paquete para decodificadores de accesorios extendidos

  Tiene la estructura:

111111111111	0	10$A_5A_4A_3A_2A_1A_0$	0	0~~$A_{10}A_9A_8$~~0$A_7A_6$1	0	000XXXXX	0	EEEEEEEE	1

  Permite la transmisión de posiciones a señales luminosas o de datos a decodificadores de funciones complejos. Cada señal luminosa puede mostrar una posición cada vez. XXXXX indica la posición que debe mostrarse, siendo '00000' la de parada absoluta y el resto dependiente del sistema de señalización empleado. La dirección del decodificador se codifica en 11 bits, con los bits 8 a 10 en complemento a uno.

- Difusión (para todos los decodificadores de accesorios básicos)

  Tiene la estructura y significado del paquete de 'activación/desactivación de accesorios, pero utilizando la dirección especial de difusión 511.

111111111111	0	10111111	0	1000CDDD	0	0011~~CDDD~~	1

  En este caso, los 4 bits menos significativos del byte de control de errores es el complemento a uno de los bits CDDD del tercer byte.

- Difusión (para todos los decodificadores de accesorios extendidos)

  Tiene la estructura:

111111111111	0	10111111	0	00000111	000XXXXX	0	101~~XX~~XXX	1

  Utiliza la dirección 2047 con el mismo significado del paquete general para decodificadores extendidos.

- Instrucción de acceso a CV de decodificador de accesorios

  Los valores de las CVs de los decodificadores de accesorios pueden mejorarse como en el caso de los decodificadores multifunción, mediante el uso de la forma larga de la instrucción de acceso a CV definida más arriba. En el caso de los decodificadores de accesorios, la dirección se expande en dos bytes, según el procedimiento siguiente:

  1) Decodificadores básicos: 10$A_5A_4A_3A_2A_1A_0$,0,1~~$A_8A_7A_6$~~1DDD (DDD indica la salida a cuya CV se accede para modificación).

     Con ello, el formato del paquete queda:

111111111111	0	10$A_5A_4A_3A_2A_1A_0$	0	1~~$A_8A_7A_6$~~1DDD	0	1110CCAA	0	AAAAAAAA	0	DDDDDDDD	0	EEEEEEEE	1

  2) Decodificadores extendidos: 10$A_5A_4A_3A_2A_1A_0$,0,0~~$A_{10}A_9A_8$~~0 $A_7A_6$1

     Con ello, el formato del paquete queda:

111111111111	0	10$A_5A_4A_3A_2A_1A_0$	0	0~~$A_{10}A_9A_8$~~0$A_7A_6$1	0	1110CCAA	0	AAAAAAAA	0	DDDDDDDD	0	EEEEEEEE	1

La secuencia de órdenes (instruction data byte) estándar DCC permite controlar una serie de parámetros de las locomotoras y accesorios, cuya descripción general es la siguiente:

- Órdenes de sentido y velocidad: Sirven para ordenar el sentido de marcha y velocidad de la locomotora. Se admiten diferentes formatos según el número mayor o menor de pasos de control admitidos por el sistema y que habitualmente son 14, 28 o 128. Cada uno de estos pasos equivale a un nivel de tensión que debe recibir el motor entre 0 V y la tensión máxima, por lo tanto, cuanto mayor sea el número de pasos menor será el "salto" de tensión y más fina la regulación de velocidad. Existen, además, una serie de valores especiales que definen el sentido de marcha y la parada de emergencia.
- Órdenes de función: Sirven para indicar la activación/desactivación de las diferentes funciones de las que sea capaz el decodificador y que sirven para gobernar sonidos, luces, etc.
- Variables de configuración: Es una de las más importantes. Se trata de una serie de instrucciones que permiten modificar la programación de los decodificadores de las locomotoras para adecuarlos a cada locomotora en particular o a los gustos del usuario.
- Instrucciones de accesorios: Se utilizan para el control de los decodificadores de accesorios (desvíos, desenganches, etc).
- Reset: Activa la parada de emergencia.

Por su parte, los bytes de dirección permiten determinar a qué decodificador concreto van destinadas las órdenes anteriores. Existen diferentes rangos de direcciones según se destinen a decodificadores de locomotoras o de accesorios, además de direcciones especiales reservadas para funciones especiales añadidas (que pueden ser utilizadas o no). Entre estas direcciones se encuentra la "0000000", que es reconocida por todos los decodificadores y que se utiliza para la parada de emergencia del sistema.

También hay direcciones que reconocen decodificadores específicos y que se utilizan, por ejemplo, para la creación de zonas de frenado automáticas.

Todas estas órdenes recibidas por el decodificador son mantenidas en la memoria interna del mismo hasta que son modificadas o anuladas por otra orden o hasta que se corta la corriente del decodificador. Así, teóricamente será suficiente con transmitir la orden una vez para que el decodificador siguiera cumpliéndola indefinidamente. Sin embargo, en la realidad, la alimentación del decodificador es un tanto precaria. La suciedad, deficiencias en el trazado, la falta de amortiguación en las ruedas, etc, genera irregularidades y microcortes (algunos no tan "micro") de corriente que pueden hacer perder la información al decodificador y por tanto su programación.

Para evitar esto, la señal enviada a la vía no es única, sino que la central, de forma cíclica y cada pocos milisegundos, repite las últimas órdenes generadas, permitiendo de esta manera "restaurar" las posibles pérdidas de información del decodificador. Dado que las órdenes deben repetirse para TODOS los decodificadores, la central posee una memoria interna en la que va almacenando todas las direcciones de decodificadores usadas por nosotros, casi siempre de forma permanente, hasta que el usuario las borre.

Esto tiene una serie de implicaciones prácticas, positivas y negativas, que deberán tenerse en cuenta.

Puesto que la central no sabe que locomotoras están realmente en vía y que la memoria de la central no es volátil, las órdenes son emitidas para todas las direcciones memorizadas. Como consecuencia negativa, si existen 30 direcciones memorizadas se repetirán cíclicamente las instrucciones para las 30 con independencia de si la locomotora correspondiente está en uso o no. Dado que cada transmisión requiere un tiempo conforme aumenta el número de direcciones disminuye la frecuencia con que las órdenes son reenviadas, con lo que si existen muchas locomotoras en memoria el tiempo desde que damos una orden hasta que esta es recibida por el decodificador aumenta, y, a veces, de forma significativa. Por ello, es conveniente eliminar periódicamente direcciones de la memoria de la central y especialmente en sistemas

de control por software, dado que en este caso se suman los tiempos necesarios para la transmisión de datos entre el ordenador, el software y la central.

Por otra parte, y como consecuencia positiva, esta repetición de órdenes permite que cuando una locomotora digital se encuentre en un tramo sin corriente, al restablecer ésta, el decoder recupere su última programación y puede continuar la marcha de forma normal.

El inconveniente de la repetición de órdenes se debe al sistema de información unidireccional, es decir, que la información se produce en la central y es transmitida a los decodificadores, pero no a la inversa.

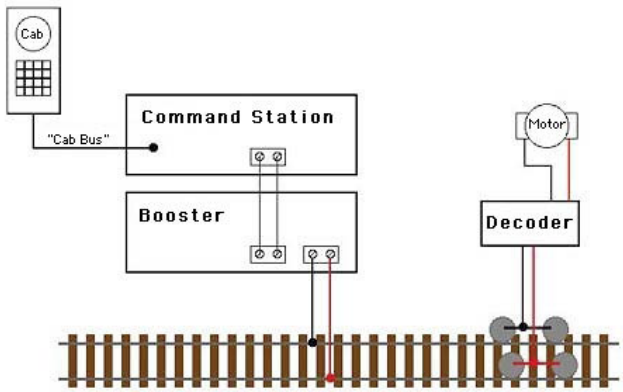
Visto el problema de la posible pérdida de paquetes DCC enviados a los decodificadores por problemas en la transmisión, las centrales digitales deben repetir su envío tan frecuentemente como sea posible. Debe ser posible configurar las centrales digitales para que transmitan un paquete completo al menos cada 30 mseg (tiempo medio entre los bits iniciales de cada paquete). Los decodificadores deben ser capaces de reaccionar correctamente frente a la recepción múltiple de paquetes, siempre que éstos se reciban separados al menos 5 mseg en el tiempo. Debe tenerse especial cuidado de no enviar dos paquetes con direcciones idénticas en el rango 112-127 con separación inferior a 5 mseg, pues los decodificadores antiguos pueden interpretarlos como paquetes de modo de servicio. Si un decodificador recibe un paquete no válido, debe de ser capaz en cualquier caso de reconocer el siguiente preámbulo válido como el comienzo de un nuevo paquete. Según el estándar DCC, se anima a los constructores de decodificadores a que incluyan mecanismos de conversión automática para diferentes tipos de señales de potencia y formatos de control adicionales a DCC, siempre que dicha conversión automática pueda deshabilitarse por el usuario. Si la conversión automática está habilitada por el decodificador, éste debe mantenerse en modo DCC mientras reciba paquetes DCC válidos con tiempos entre bits iniciales menores o iguales a 30 mseg. Si está deshabilitada, el decodificador debe mantenerse en modo DCC independientemente de la temporización en la recepción de paquetes. Como caso particular de conversión automática puede citarse el paso de modo digital a modo convencional (analógico). Éste tiene asignado el código 00000001 de la NMRA y permite utilizar locomotoras con decodificador DCC en entornos no digitales en los que se utiliza tensión continua variable. Otras conversiones con código asignado por la NMRA son Radio (00000010), Zero-1 (00000100), TRIX (00001000) y CTC-16/RailCommand (00010000). Hay que indicar que el uso simultáneo de protocolo DCC y Motorola, por utilizar la misma señal base, no necesita de conversión automática.

Existe una única excepción, común a todos los decodificadores, que es la capacidad de éstos para generar pequeñas sobretensiones, aprovechándola para informar a la central de que se ha cambiado su programación (cuando pasa esto se puede apreciar un pequeño parpadeo o una pequeña actividad del motor). No obstante, existen ya sistemas bidireccionales, como el sistema propietario de “Zimo” (requiere que todo el sistema sea Zimo para poder funcionar), o el sistema RailCom de Lenz (que aspira a convertirse al igual que el DCC en estándar). Esta bidireccionalidad, al poder transmitir información desde el decoder de la locomotora, permitirá conocer no sólo las vías que están ocupadas, sino también saber por cuál de las locomotoras y el estado de dicha locomotora. En este punto hay que hacer una aclaración, y es que tanto el sistema RailCom como el Zimo, utilizan el propio decodificador como receptor/emisor (de hecho, la serie de decodificadores Lenz Gold ya llevan el software y varios de los decoders Zimo pueden programarse para que sean compatibles), permitiendo la comunicación bidireccional entre la central y el decoder de forma continua y en cualquier parte de la vía.

Esto los diferencia de sistemas como Lissy, que requieren de un elemento emisor en la locomotora que transmita la información del decoder hacia los receptores, situados en posiciones fijas del trayecto, que son los encargados de transmitir la información a la central cuando el vehículo pasa sobre ellos.

Dos de los componentes principales (incluso se puede decir que los más relevantes) en un sistema DCC son la unidad de control (command station) y los decodificadores (decoders). Es por este motivo por el que se extiende la explicación de estos componentes más de lo habitual en esta exposición del sistema DCC.

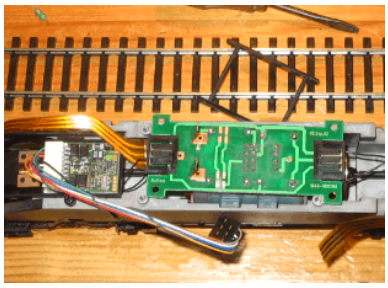
El siguiente es un esquema básico de los elementos que intervienen en la instalación de un sistema DCC:



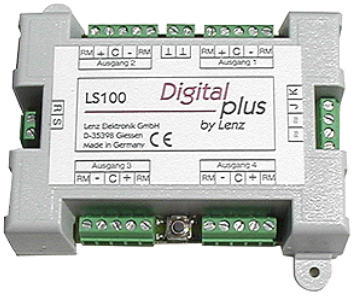
- "Mando de control" (cab): Su función principal es controlar al tren, gestionando el control de velocidad, la dirección, luces, etc. Suele disponer de una pequeña pantalla donde podemos ver los ajustes programados. También podemos configurar el sistema con el mando de control. En el caso de las locomotoras: número de dirección asignada, pasos de velocidad configurados, etc, y, en el caso de funciones varias: configuración de funciones especiales a través de los botones de función, posición de desvíos, etc. Con un solo mando se pueden programar y controlar varios trenes. El mando de control constituye el interfaz entre el sistema y el usuario.
- Central de control (command station): Tiene varias funciones muy importantes:
    a) Implementa las comunicaciones con el mando de control, es decir, lee los comandos del mando y actualiza el status de la pantalla del mismo.
    b) Traduce los comandos del mando en comandos DCC.
    c) Mantiene el estado de las locomotoras: Velocidad, funciones especiales, luces, etc.
    d) Proporciona, opcionalmente, interfaz de comunicación con un ordenador.

  Se considera que la central de control es el cerebro del sistema DCC, puesto que controla el conjunto de periféricos conectados. Cuando la central de control se encarga de codificar las órdenes que han de recibir los diferentes componentes y procesar los datos que llegan de los mismos lo hace conforme al estándar DCC S-9.1. La forma de comunicación se establece en el estándar DCC S-9.2.
- Amplificador (booster): Amplifica la señal DCC proveniente de la central de control, convirtiendo la señal de datos en una señal eléctrica de potencia. Esta señal convertida se transmite a la vía. El amplificador puede estar agrupado con la unidad de control en una sola caja. La mayoría de los aparatos llevan acoplado (externamente) un transformador de potencia que reduce la corriente de 220 voltios a 16 voltios en corriente alterna. Además, proporciona tanto voltaje regulado y dependiente de la escala, como una corriente de salida fijada y dependiente del número de trenes en funcionamiento. Además, también proporciona protección contra cortocircuitos (ignorará los cortocircuitos "cortos" y ejecutará un "shutdown" en caso de cortocircuitos largos o persistentes. Actúa de protección contra los problemas de los bucles de retorno en la vía.
- Decodificadores (decoders): Le asigna a cada locomotora una única dirección y permite detectar los decoders DCC y programarlos cuando son accedidos. Este elemento es el que decodifica los comandos para el control del motor (tanto en dirección como en velocidad). Además, decodifica los comandos para controlar las funciones especiales de salida, como luces, sonido, etc. Los decoders pueden ser de dos tipos: embarcados o estáticos. Los primeros se instalan en el interior

de las locomotoras y vagones, mientras que los segundos se utilizan para controlar los desvíos, semáforos, accesorios, etc. En el interior del decodificador hay un microcontrolador que es el que procesa la información y reparte el trabajo entre los distintos amplificadores del mismo. Se pueden ver los dos tipos de decodificadores en las siguientes figuras:



Decoder embarcado

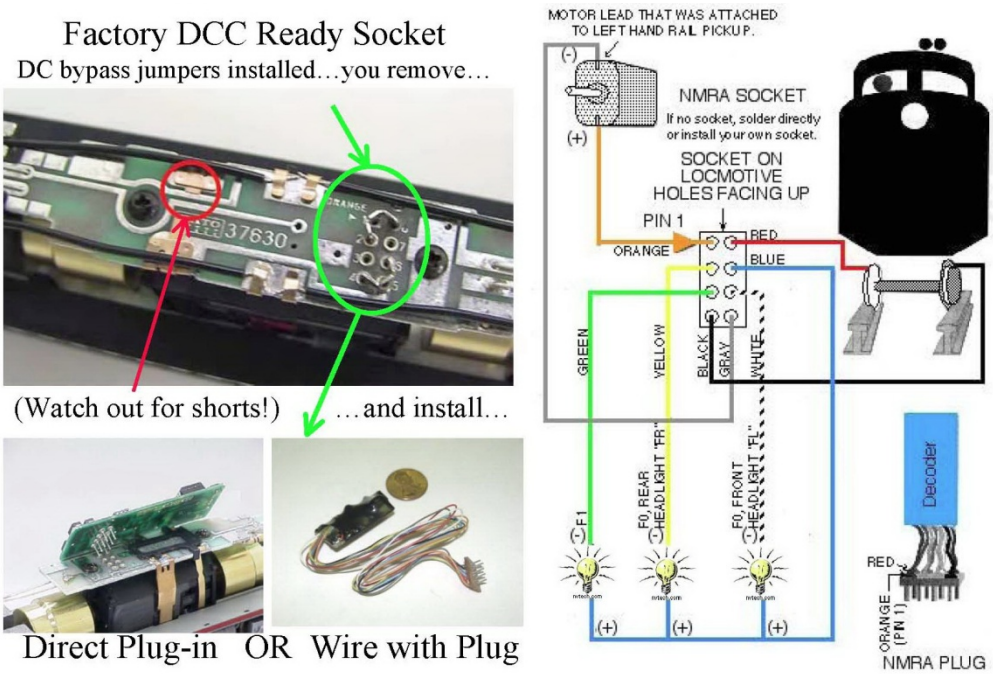


Decoder estático

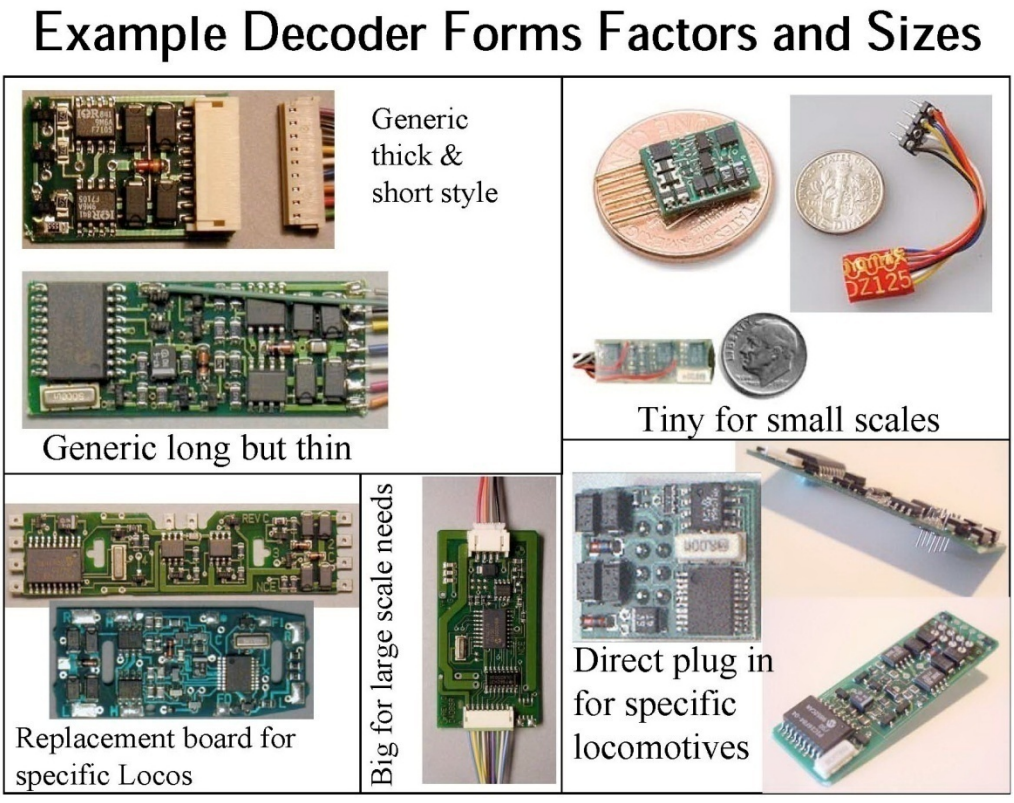
El decodificador quizá es el dispositivo más importante después de la central. En realidad, el descodificador es el que "traduce" las informaciones provenientes de la central, y las ejecuta, ya sea en una locomotora, en un desvío, etc. Los descodificadores para escalas grandes son de proporciones generosas porque no tienen peligro de no caber en la locomotora, eso sí, como se les pide potencia, han de entregar cifras de amperaje más que respetables. Suelen utilizar tornillos para la fijación de los cables. El resto de decodificadores embarcados dependen de sus medidas para entrar en la locomotora, por eso están fabricados con tamaños muy pequeños y componentes electrónicos del tipo SMD. Los más generosos en medidas son los de escala H0, que pueden llegar a entregar 1A al motor, y los más pequeños son los de escala N. De momento, la escala que se queda sin posibilidad de embarcar decodificadores es la Z, aunque se han inventado soluciones de acantonamiento a base de decodificadores exteriores a la locomotora. No se pueden activar las funciones, pero el funcionamiento en digital es impecable. La mayoría de fabricantes tiene ya unos estándares de fabricación para que poner un decodificador sea lo más fácil posible y no haya que soldar cables a la locomotora. Así surgieron las normas NEM 651 y 652 con respecto del digital, y que vienen a estandarizar los conectores de los decodificadores para que puedan ser introducidos en una locomotora con el mínimo esfuerzo (sacar un conector ciego y enchufar el conector del decodificador) para tener la locomotora funcionando en digital. En los últimos años se ha progresado mucho en el mundo de los decodificadores, pasando a tener compensación de carga, control de motor por frecuencia, autoprotección contra cortocircuitos y calentamiento.

Los decodificadores estáticos son aquellos que están dispuestos en la maqueta y no sirven para la tracción. La gran diferencia con los embarcados es que los estáticos solo controlan funciones y en ningún caso controlan la velocidad de motores. Hay dos tipos de decodificadores, los que no dan retroinformación a la central y los que sí que la entregan. La diferencia más notable entre estos y los anteriores es que éstos suelen recurrir a fuente de alimentación externa para no gastar la corriente digital. La mayoría están opto acoplados para que no haya parásitos entre las señales de entrada y salida y pueden servir para varios usos.

A continuación, se muestra el conexionado típico de un decodificador insertado en una locomotora:



Y en el siguiente gráfico se pueden observar los distintos formatos en los que se pueden encontrar los decodificadores, dependiendo de la escala y uso al que vayan destinados:

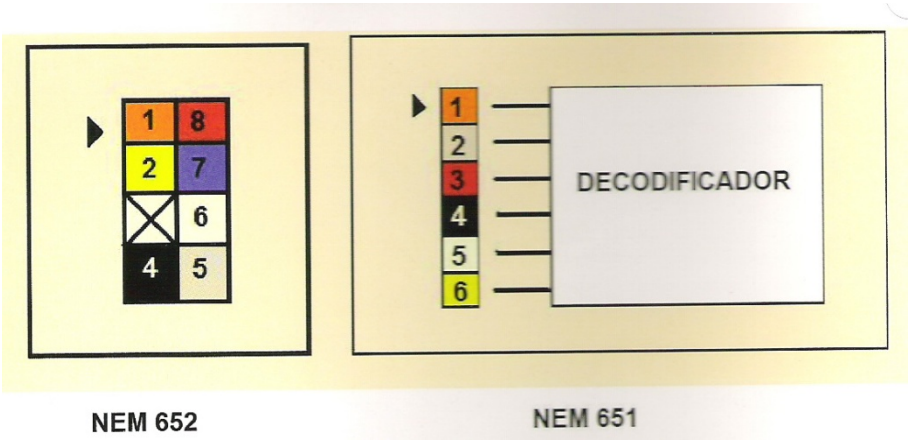


¿Qué modelo de decodificador elegir?

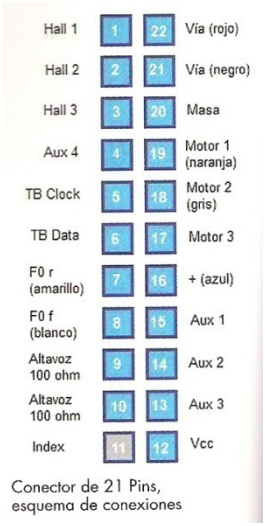
Esta es una pregunta importante cuando lo que se pretende es digitalizar una locomotora. Lo primero que hay que hacer es desmontarla para examinar su interior con detalle. Para evitar problemas, es conveniente echar un vistazo a las instrucciones o manual del fabricante para ver cómo hacerlo.

Modelos con conector digital

Si el modelo que se quiere digitalizar ya viene previsto de lo necesario para digitalizarlo sólo queda ver qué tipo de conector lleva y buscar el decodificador adecuado. Existen dos tipos de conectores estandarizados: NEM 651 y NEM 652. Se pueden observar en la siguiente figura:



El modelo NEM 651 es el habitual que se puede encontrar en las locomotoras a escala N, mientras que el NEM 652 se utiliza en las locomotoras a escala H0. Últimamente ha aparecido en el mercado un conector de 21 pines, y aunque aún no está normalizado, ya se está utilizando por varios fabricantes. Las locomotoras previstas para digitalizar ya vienen con una pequeña plaquita enchufada en el conector. Esta plaquita, conocida como “dummy”, y de aspecto en algunos casos similar a un decodificador, permite que la locomotora funcione correctamente en un sistema analógico, pero obviamente no en un sistema digital. Sólo hay que desenchufar esta plaquita y conectar el decodificador en su lugar para digitalizar la locomotora. En el caso del conector NEM 651 no hay que hacer nada más, puesto que el decodificador queda sujeto por los pines, pero en el caso del NEM 652 no es así, puesto que el decodificador está unido a los pines mediante cables. Hay que fijar el decodificador en el lugar previsto por el fabricante mediante cinta adhesiva de doble cara (los fabricantes suelen incluir en la caja del decodificador un trozo de esta cinta del tamaño adecuado). En caso de disponer de un conector de 21 pines, el decodificador queda igualmente sujeto.



Conector de 21 Pins, esquema de conexiones

En algunos casos, sobre todo con conectores NEM 651 y NEM 652, el decodificador puede enchufarse en dos posiciones distintas. ¿Cómo saber cuál es la posición correcta? En muchos casos el fabricante indica en el circuito impreso de la locomotora o bien en el manual, cual es la patilla 1. Sólo hay que consultar el manual del decodificador para averiguar cuál es la referida patilla y proceder. En caso de no disponer del manual o no ver claro cuál es la posición adecuada, es fácil averiguarla. En cualquier caso, la posición no debe preocupar al usuario, puesto que el diseño de las patillas está estudiado de manera que no ocurra ningún desastre si se conecta el decodificador al revés. Si el dispositivo se conecta de forma incorrecta, suele ocurrir que solo funcionan las luces, o bien que se encienden nada más colocar el tren en la vía.

Modelos con pre-instalación digital

Con cierta frecuencia, hay modelos relativamente recientes que no disponen de conector digital. Pero también hay algunos casos en el que el fabricante ha previsto espacio para el decodificador y, además, da unas instrucciones lo suficientemente precisas de cómo conectarlo. Casi siempre hay que cortar una o varias pistas, retirar algún componente, etc. En este caso, se suele aplicar todo lo que se contempla en los modelos sin conector digital, aunque el proceso se simplifica enormemente, pues ya se tienen una guía o manual de cómo conectarlo y que, además, ya está resuelto el problema del espacio, porque el fabricante ya ha reservado un lugar para el decodificador.

Modelos sin conector digital

En este caso hay una entretenida tarea por delante, aunque según el modelo puede costar más o menos trabajo. Hoy día se puede afirmar que no hay ninguna locomotora a la que no se le puede instalar un decodificador. Lo primero de todo, se impone un estudio de la locomotora. Para ello, hay que encontrar un hueco adecuado para el decodificador. Si no, habrá que “fabricarlo”, bien retirando alguna parte no imprescindible, o bien rebajando el chasis lo suficiente como para que nos quepa sin problemas, etc.

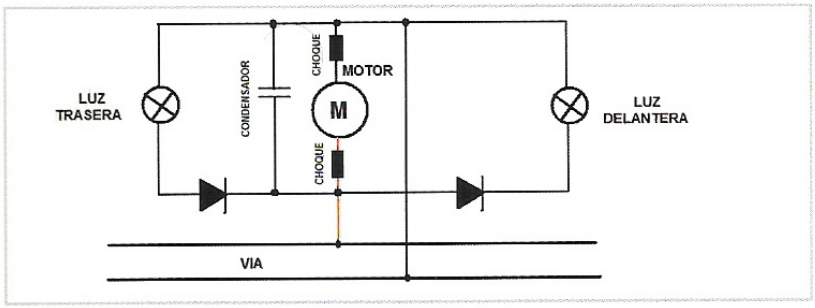
¿Qué características debe tener este espacio para el decodificador? Se debe intentar seguir, en la medida de lo posible, las siguientes recomendaciones:

- Buscar un lugar en el que quepa el decodificador sin problemas. Es recomendable que quede espacio alrededor del decodificador, pues mejorará su refrigeración.
- Si hay que rebajar el chasis, hay que prever espacio adicional para un aislante, a fin de evitar cortocircuitos. El decodificador debe estar completamente aislado para evitar problemas.
- Si hay disponibilidad de escoger más de un lugar para ubicarlo, es preferible que no esté cerca de la carcasa, ya que, si por alguna razón el decodificador se sobrecalienta, el calor generado puede dañarla.

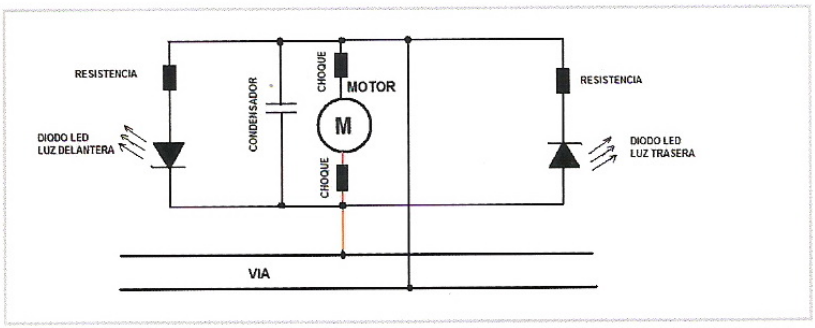
- Prever espacio para pasar los cables desde la ubicación prevista del decodificador a los puntos de conexión.

Normalmente un decodificador de N se suele calentar algo más que uno de H0, ya que estos últimos son más grandes y sus componentes electrónicos de mayor tamaño disipan mejor el calor.

Una vez resuelto el tema del espacio y ubicación, y antes de pasar a conectar los cables del decodificador, hay que ojear los típicos esquemas de conexión para entender mejor cómo se debe conexionar. El esquema de conexionado de una locomotora analógica sería el siguiente:



Esquema eléctrico de una locomotora con iluminación con lámparas



Esquema eléctrico de una locomotora con iluminación por LED

En este esquema se ven diferenciados varios componentes. Por un lado, está el motor y las luces delantera y trasera. Además, se encuentran una serie de componentes electrónicos: un par de diodos, un condensador y dos choques (bobinas con núcleo de ferrita). Cada uno de estos componentes realiza una función distinta pero necesaria.

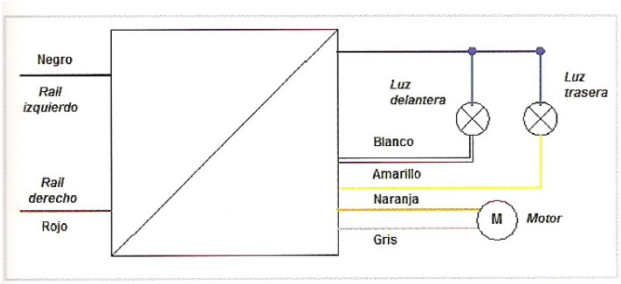
Los diodos colocados en serie dejan pasar la corriente continua en un solo sentido. Están colocados de manera que solo se enciende una lámpara en función del sentido de la corriente, para conseguir que se ilumine solo la luz según el sentido de la marcha. Las locomotoras más modernas suelen llevar la iluminación por diodos LED en lugar de las clásicas lámparas incandescentes. En el segundo esquema de la figura anterior se puede ver esto.

Como los LED funcionan con voltajes entre 1,5 y 3 voltios aproximadamente (según modelos), existen un par de resistencias para cada uno a fin de rebajar la tensión al valor que necesitan, pues de otra manera se fundirían. Además, los diodos que hacen que solo funcionen según el sentido de la marcha se han eliminado, pues los LED son a fin de cuentas “diodos” y solo dejan pasar la corriente en un solo sentido. Colocados, por tanto, de forma correcta se obtiene el mismo efecto que con la pareja lámpara convencional-diodo.

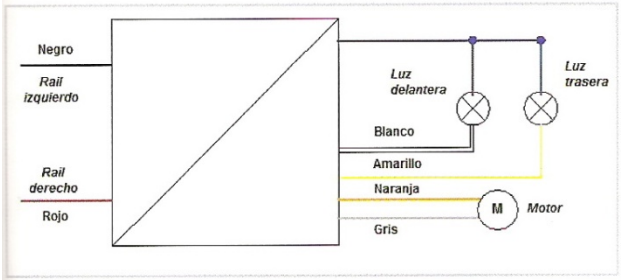
En los dos esquemas se puede ver también un par de choques o bobinas conectados en serie con el motor. Estas bobinas dejan pasar la corriente continua, pero al mismo tiempo actúan como filtros y eliminan las posibles interferencias generadas por el motor en otros aparatos (televisores, radios, etc). El condensador, colocado en paralelo, no deja pasar la corriente continua, pero actúa igualmente de filtro. Su misión suele

ser minimizar las interferencias generadas por el chisporroteo de las escobillas en su roce con el colector del motor. Estos componentes electrónicos se comportan de manera muy distinta en presencia de la tensión digital. Para entender sus efectos se puede considerar que la tensión digital es similar a una corriente alterna, por cuanto su polaridad cambia con una determinada frecuencia. Las bobinas colocadas en serie no dejan pasar la totalidad de una tensión alterna, y los condensadores, sí dejan pasar una pequeña porción. Por tanto, hay que eliminar todos estos componentes a fin de evitar efectos indeseados que puedan interferir en el funcionamiento del decodificador.

A continuación, se expondrán algunos esquemas de conexión de un decodificador en una locomotora. Hay dos posibles esquemas de conexión. El primero de ellos muestra cómo la alimentación de las luces de la locomotora utiliza el cable azul como común. Sin embargo, en el segundo se utiliza como común para este fin el mismo cable negro que toma la alimentación de la vía.



Esquema de conexión con cable común independiente



Esquema de conexión sin cable común independiente

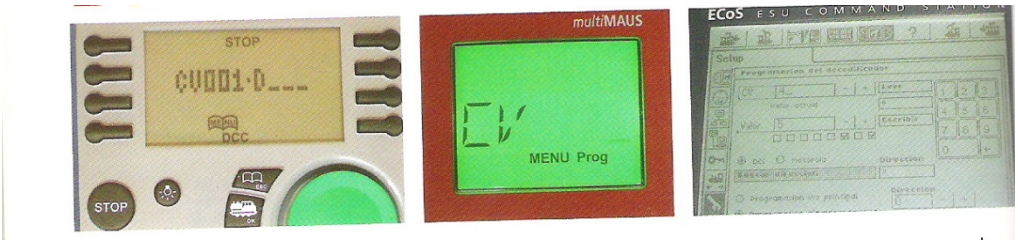
¿Qué diferencia hay entre uno y otro?

Pues bien, lo primero que hay que comentar es que el cable azul no es imprescindible utilizarlo. De hecho, en algunos modelos de decodificadores no existe y lo usual es no utilizarlo, muchas veces simplemente por comodidad. La explicación es la siguiente: la salida de alimentación de las luces de la locomotora (cables blanco y amarillo) es de corriente continua. A través del cable azul, la tensión obtenida utiliza todas las “semiondas” completas de la tensión digital. Por el contrario, si utilizamos el cable negro sólo se utiliza la mitad.

¿Qué efectos o consecuencias puede tener esto en el funcionamiento de la locomotora? Pues en la práctica ninguna. Lo único que se aprecia es que si se utiliza el cable negro en vez del azul hay menos intensidad en las luces. Otro efecto apreciable es que al hacer funcionar una locomotora digitalizada en un sistema analógico convencional es probable que una de las luces no funcione, aunque esto no ocurre en todos los modelos de decodificadores. Si una locomotora está digitalizada, lo más usual es que se utilice principalmente en un sistema digital. La razón principal de que haya dos opciones de conexión para las luces parece ser que viene motivada porque hay muchas locomotoras que utilizan lámparas de casquillo metálico embutidas en el chasis. Al mismo tiempo, este chasis suele utilizarse también para tomar la corriente de la vía. En estos casos es realmente complicado aislar la toma de corriente o las luces del chasis, por lo que se puede utilizar el cable negro como común. En resumen, no hay que preocuparse si no se puede usar el cable azul. Lo que no se debe hacer en ningún caso es unir el cable negro y el cable azul, que provocaría la destrucción del decodificador. Se debe utilizar uno u otro según interese.

Programación de decodificadores

Cuando ya se ha instalado el decodificador en la locomotora hay que asegurarse (normalmente colocándola en la vía) que responde a las órdenes de la central digital y que todo funciona correctamente. Sin embargo, el trabajo de digitalización no termina con esto. Para el usuario que no quiera complicarse más de la cuenta, seguramente cambiará la dirección del decodificador de la locomotora y dará por concluido el trabajo. Sin embargo, mediante los ajustes adecuados se puede mejorar el funcionamiento de la locomotora y "personalizarla" de manera que funcione de manera muy parecida a como lo hacen las reales.



Este aparato de programación resulta quizás aún más gratificante que el de la instalación del decodificador, pues se obtendrán resultados aún más visibles y espectaculares en el funcionamiento del tren respecto al funcionamiento analógico, mejorándolo en casi todos los casos.

Cada central digital tiene sus propios menús y opciones de programación. No obstante, al final el resultado obtenido es el mismo. A fin de manejar correctamente cada central, lo más adecuado es consultar las instrucciones del fabricante.

Las variables de configuración (CV)

En realidad, se puede afirmar que un decodificador es como un pequeño ordenador en miniatura. Contiene su chip o procesador y también utiliza un programa (software). Lo que interesa conocer de este dispositivo es que contiene unas celdas de memorias en las que se almacenan los valores. Estas celdas o memorias son las llamadas abreviadamente CV, y están numeradas. Cada una de estas celdas contiene un valor que influye en un determinado aspecto en el funcionamiento del decodificador. Los posibles valores de las CV van de 0 a 255. En casi todas se puede escribir, aunque hay algunas que solo se pueden leer (como por ejemplo el número de la versión del software que trae el decodificador o el código del fabricante). También es importante resaltar que el valor almacenado en cada una de las CV no es volátil, es decir, no se pierde cuando el decodificador no recibe tensión. Muchas de estas CV están reguladas por la NMRA y la mayoría de los decodificadores las cumplen. Lo usual es que los fabricantes utilicen un grupo de CVs más o menos estandarizadas, aunque también otras muchas no lo están. La NMRA, que dicta las normas sobre sistemas digitales, tan solo marca como obligatorias el valor de cuatro CVs (CV1, CV7, CV8 y CV29) dejando otras como recomendadas, y el resto a total disposición de los fabricantes que pueden destinarlas al uso que deseen.

Existen dos clases de CV. Las llamadas "estándar", cuyo valor implica un solo cambio en el funcionamiento del decodificador, y las que son de tipo complejo, pues almacenan simultáneamente varios valores. En estas CV complejas, lo que se utiliza es el valor que tenga almacenado tratándolo en formato binario, por lo que su interpretación implica un cambio en varios aspectos a la vez. Cada bit resultante de convertir el número a formato binario se utiliza por separado. Como en el formato binario solo existen dos valores posibles (0 y 1) éstos actúan como "interruptores" que activan (1) o desactivan (0) determinadas funciones (se ha visto esto de forma ampliada anteriormente en esta exposición). Hay centrales que permitirán programar estos bits separadamente. En las demás habrá que realizar una pequeña conversión para introducir el valor correcto. Las variables más comunes y que siguen la norma DCC son las siguientes:

CV	DESCRIPCIÓN	VALOR POR DEFECTO	RANGO DE VALORES
1	Dirección: este número identifica la locomotora (formato corto). Valores entre 1 y 127)	3	1 – 127
2	Velocidad mínima: velocidad de la locomotora en el paso 1 del mando.	1	0 – 255
3	Aceleración: nivel de aceleración.	1	0 – 255
4	Deceleración: nivel de deceleración	1	0 – 255
5	Velocidad Máxima: define la velocidad máxima, cuando el mando regulador está al máximo.	255	0 – 255
6	Velocidad Media: define un punto central en la curva de velocidad del modelo.	Según modelo	Según modelo
7	Versión: (solo lectura). Contiene la versión del firmware del decodificador.	-	Varias
8	ID Fabricante: (solo lectura) Contiene el código del fabricante.	-	117
9	PWM: Ajuste de los pulsos del motor.	Según modelo	Según modelo
17 + 18	Dirección Extendida: también llamada dirección larga (128 - 9999).	0	128-10240
19	Dirección para Mando Múltiple: normalmente está a 0. En ella se almacena el valor de la locomotora "maestra". Permite el mando múltiple de varias locomotoras.	0	1-128
29	Bits de configuración: controla varias características. Bit 0 – Dirección de marcha: 0 = normal 1 = invertida Bit 1 – pasos de velocidad: 0 = 14, 1 = 28 ó 128 Bit 2 – alimentación: 0 = solo digital 1 = digital y analógico Bit 3 – No usado Bit 4 – curva de velocidad: 0 = Por defecto, controlada por CV 2, 5, 6 1 = tabla libre definición CV 67 - 94 Bit 5 – Dirección extendida de la loco: 0 = 1-128 almacenada en CV 1 1 = 128 - 10240 almacenada en CV 17 + 18	2	0, 1 (bits)
67-94	Curva de velocidad configurable: se activa cuando el bit 4=1 en CV 29	–	0-252

Aunque éstas son las más usuales, existen muchas otras que los decodificadores utilizan. En cualquier caso, es imprescindible consultar el manual de cada decodificador, pues incluso las CV estandarizadas permiten distinguir rangos de valores.

CV1. Dirección de locomotora

El rango de direcciones de locomotoras permitido en DCC es de la 1 a la 9999. Hay que distinguir entre las llamadas "direcciones cortas" (de la 1 a la 127) y las llamadas "direcciones largas" (de la 128 a la 9999). La dirección corta se almacena en la CV1, mientras que las largas lo hacen en las CV 17 y 18. Mediante el bit 5 de la CV29 se le indica al decodificador que utilice una u otra (visto anteriormente en esta exposición). Las centrales que soportan 9999 direcciones suelen disponer de un menú separado para programar las direcciones largas, lo que ahorra tener que calcular y descomponer la dirección para almacenarla en las CV17 y CV18. Incluso algunas realizan el cambio del bit 5 de la CV29, aunque otras no. Si la central no dispone de esa opción, habrá que almacenar a mano la dirección en estas dos variables. El cálculo se puede hacer fácilmente con la calculadora de Windows:

a) Se convierte la dirección decimal a formato hexadecimal.

b) Si salen menos de 4 cifras, completar hasta las 4 cifras con ceros a la izquierda.

c) Separar el número hexadecimal por la mitad y convertir estos dos números a decimal.

d) El número en decimal de la parte izquierda hay que almacenarlo en la CV17 y el número de la derecha en la CV18.

e) Ya se tiene la dirección introducida. Para que el decodificador use la dirección larga tenemos que activar el bit5 de la CV29 (ver cálculo de variables complejas). Por defecto el bit 5 normalmente viene desactivado (por defecto el decodificador viene de fábrica ajustado en la dirección 3, que entra dentro del rango de direcciones cortas), con lo que se usaría la dirección corta (de 0 a 127) y, si se pone a 1, se usaría la dirección larga.

CV2. Velocidad mínima

En esta CV se almacena la velocidad a la que la locomotora comenzará a moverse en el primer paso de velocidad. Lo correcto es almacenar un valor tan bajo como se pueda de manera que en el primer paso de velocidad la locomotora comience a moverse a la mínima velocidad posible, pero de forma constante, segura y sin detenerse.

CV3. Aceleración

Esta es una de las CV que permitirán acercar el funcionamiento del modelo al de una locomotora real. En el sistema tradicional analógico, la locomotora responde inmediatamente a los cambios de velocidad marcados por el regulador. Pero en la realidad esto no ocurre así. Cuando se pisa el acelerador de un coche, éste tarda en responder un tiempo determinado intentando alcanzar la velocidad deseada. Igual ocurre en las locomotoras. El maquinista ajusta el regulador hasta una posición determinada, y a continuación la locomotora comienza a moverse, intentando coger velocidad. Mediante el valor introducido en esta CV se ajusta la aceleración. Cuanto mayor sea este valor, más tiempo tardará en acelerar la locomotora y viceversa.

CV4. Deceleración (o frenado)

Ocurre lo contrario que en la CV3 de aceleración. Cuando se pisa el freno de un coche o el maquinista de una locomotora acciona los frenos, el vehículo no se detiene en seco, sino que conserva una cierta inercia de frenado, recorriendo una determinada distancia hasta quedar detenido. El ajuste de la CV4 permite regular el frenado de la locomotora. A mayor valor, más distancia recorrerá la locomotora y más tiempo tardará en detenerse.

CV5. Velocidad máxima

Con frecuencia ocurre que los trenes en miniatura corren “más de la cuenta”. Los diferentes tipos de motores empleados en modelismo, y las relaciones de engranajes no están con frecuencia ajustados al tipo de locomotora. Actuando sobre el valor de esta variable se consigue adecuar su velocidad máxima real.

CV6. Velocidad media

Los decodificadores trabajar internamente con una curva de velocidad. Esta curva marca el funcionamiento del motor en función de los pasos de velocidad. Por defecto, esta curva de velocidad viene regulada por los valores contenidos en las CV2, CV5 y CV6. Alterando el valor de la CV6, se puede ajustar esta curva, de manera que no sea lineal, aunque lo normal es que cada fabricante prefije los valores de estas CV de forma que las velocidades sean menores en los primeros pasos y luego aumenten progresivamente, a fin de conseguir una regulación precisa en los primeros puntos del regulador. Si se activa el bit 4 de la CV29, se pueden utilizar los valores contenidos en las CV67 a CV94, con lo cual se puede ajustar una curva de velocidad personalizada con toda precisión (visto anteriormente en esta exposición).

Las **CV7** y **CV8** solo se pueden leer: no se puede escribir en ellas. Son valores que indicarán la versión del software (programa interno) del decodificador y un número identificador del fabricante (ver tabla de fabricantes más abajo).

CV9. Ajuste de los pulsos del motor

Regula la anchura y frecuencia de corriente pulsante que el decodificador envía al motor. Dependiendo del modelo y marca estos valores cambian ostensiblemente, por lo que se impone consultar el manual del fabricante. Mediante el ajuste de este valor, se consigue que la locomotora arranque o vaya más suave a bajas velocidades. Si los pulsos son más continuos, el movimiento será más suave y viceversa. Si la anchura de los pulsos es mayor, el modelo tenderá a moverse a tironcitos, aunque el efecto se notará más o menos dependiendo de su mecánica. Donde este valor influye más es a bajas velocidades.

CV29. Ajustes de configuración (CV compleja)

Esta CV controla varios parámetros a la vez. El valor en binario que contenga, activará o desactivará ciertas funciones del decodificador.

- Bit 0 → Dirección de marcha: 0 = normal; 1 = invertida
- Bit 1 → Pasos de velocidad: 0 = 14; 1 = 28/128
- Bit 2 → Sistema de alimentación: 0 = sólo digital; 1 = digital y analógico
- Bit 3 → No usado
- Bit 4 → Curva de velocidad: 0 = Por defecto, controlada por CV2, CV5 y CV6; 1 = tabla libre definición mediante las CV67-94
- Bit 5 → Dirección extendida de la locomotora: 0 = 1-128 almacenada en CV1; 1 = 128-9999 almacenada en CV17+18

Si la central permite el modo de programación bit a bit, se puede activar o desactivar cada uno de los parámetros de manera muy sencilla. En cualquier caso, el cálculo para obtener el valor equivalente es fácil: se van sumando valores correspondientes a los bit activos y el valor resultante se programa como cualquier otra CV:

Bit 0 es 1, sumar 1; Bit 1 es 1, sumar 2; Bit 2 es 1, sumar 4; Bit 3 es 1, sumar 8; Bit 4 es 1, sumar 16; Bit 5 es 1, sumar 32; Bit 6 es 1, sumar 64, Bit 7 es 1, sumar 128

En la siguiente página se muestra una tabla de identificación de fabricantes, en la que se muestra el nombre del fabricante, su número de identificación y su país:

Advance IC Engineering	17	US
AMW	19	AT
Arnold – Rivarossi	173	DE
Atlas Model Railroad Products	127	US
AuroTrains	170	US/IT
Bachmann Trains	101	US
BRAWA Modellspielwaren GmbH & Co.	186	DE
CML Electronics Limited	1	UK
Computer Dialysis France	105	FR
Con-Com GmbH	204	AT
cT Elektronik	117	AT
CVP Products	135	US
Dietz Modellbahntechnik	115	DE
Digitrax	129	US
Doehler & Haas	97	DE
Electronic Solutions Ulm GmbH	151	DE
Gebr. Fleischmann GmbH & Co.	155	DE
Haber & Koenig Electronics GmbH (HKE)	111	AT
Intelligent Command Control	133	US
JMRI	18	US
KAM Industries	22	US
Kreischer Datentechnik	21	DE
Kuehn Ing.	157	DE
Lenz Elektronik GmbH	99	DE
LGB (Ernst Paul Lehmann Patentwerk)	159	DE
Massoth Elektronik, GmbH	123	DE
MAWE Elektronik	68	CH
MBTronik – PiN GITmBH	26	DE
MoBaTron.de	24	DE
Model Electronic Railway Group	165	UK
Model Rectifier Corp.	143	US
Modelleisenbahn GmbH (formerly Roco)	161	AT
MTH Electric Trains, Inc.	27	US
Nagasue System Design Office	103	JP
NCE Corporation (North Coast Engineering)	11	US
New Your Byano Limited	71	HK
NMRA Reserved	238	US
ProfiLok Modellbahntechnik GmbH	125	DE
PSI –Dynatrol	14	US
Public Domain & Do-It-Yourself Decodificadores	13	-
QS Industries (QSI)	113	US
Railnet Solutions, LLC	66	US
Ramfixx Technologies (Wangrow)	15	CA/US

RealRail Effects	139	US
Rock Junction Controls	149	US
RR-Cirkits	87	US
S Helper Service	23	-
Sanda Kan Industrial, Ltd.	95	-
T4T – Technology for Trains GmbH	20	DE
Tams Elektronik GmbH	62	DE
Team Digital, LLC	25	US
The Electric Railroad Company	73	US
Throttle-Up (Soundtraxx)	141	US
Train Control Systems	153	US
Train Technology	2	BE
Trix Modelleisenbahn	131	DE
Uhlenbrock GmbH	85	DE
Umelec Ing. Buero	147	CH
Viessmann Modellspielwaren	109	DE
W. S. Ataras Engineering	119	US
Wangrow Electronics	12	US
WP Railshops	163	CA
Zimo Elektronik	145	AT
ZTC	132	UK

Modos de programación

Existen cuatro modos de programación:

Modo directo (modo CV)

Es el más utilizado. En este modo es como se suelen programar los CV, asignando a cada una de ellas el valor correspondiente.

Modo bit a bit

Mediante este modo podemos programar por separado cada uno de los bits del valor binario de una CV.

Modo paginado (modo PM)

Es un modo similar al anterior. No todos los decodificadores y centrales soportan este modo de programación.

Modo registro (modo RM)

Utilizado por los decodificadores antiguos. Estos decodificadores utilizaban registros de memoria (usualmente hasta 8) en vez del sistema de CV utilizado actualmente. No todas las centrales disponen de todos los modos de programación. Lo habitual es utilizar el modo directo CV, aunque también podría utilizarse el modo paginado. Si el decodificador es antiguo (por ejemplo algunos decodificadores de Arnold con 14 pasos de velocidad), probablemente no soporte CV y habrá que programarlos mediante el modo registro.

Trucos y consejos de programación

El voltaje de arranque (CV2) se debe ajustar de manera que la locomotora empiece justo a andar en el primer punto del mando. Es cuestión de probar, hasta dar con el valor adecuado. La velocidad máxima

también es conveniente ajustarla de manera que la locomotora circule lo más parecido al modelo real, aunque sobre gustos no hay nada escrito.

Las CV67-94 permiten una curva de velocidad definida por el usuario. Esto es: cómo se comporta realmente una locomotora durante el recorrido del regulador. Esto permite un ajuste muy fino, aunque es un poco pesado ir introduciendo uno a uno todos los valores y además es difícil calcularlos. Para activar la curva de valores personalizada, hay que activar el bit 4 de la CV29. Hay programas en Internet que pueden ayudar con esta tarea (se dibuja la curva y devuelven los valores a programar).

¿Modo de 28 ó 128 pasos? A 28 pasos de velocidad se puede regular la velocidad de una locomotora con comodidad. No obstante, se puede optar por el modo de 128 pasos que permite un ajuste más preciso, aunque en las centrales con mando giratorio del tipo sinfín algunas veces es algo incómodo el tener que girar en exceso el mando para alcanzar las velocidades superiores. Los decodificadores actuales soportan reconocimiento automático entre los modos de 28 y 128 pasos, por lo que solamente hay que cambiar este ajuste en la central. En cualquier caso, siempre debe haber una concordancia entre el número de pasos configurado en la central y el decodificador.

Ajuste de intensidad de las luces: Hay decodificadores que permiten (al igual que se puede ajustar la velocidad máxima de la locomotora) ajustar la intensidad de las luces (lo que se llama “dimming”). En algunas locomotoras que utilizan lámparas para las luces puede ocurrir que funcionando a máxima tensión el calor generado sea excesivo, con peligro de dañar la carcasa. En este caso se pueden atenuar las luces disminuyendo el calor generado. En algunas locomotoras, el brillo de las lámparas (o incluso de los diodos LED) puede hacer que la luz traspase la carcasa, generando un efecto bastante antiestético, lo que puede mitigarse en parte bajando la intensidad. Hay decodificadores, como por ejemplo Uhlenbrock, que por defecto vienen de fábrica con la intensidad de las luces ajustada aproximadamente a la mitad.

También hay posibilidades de configuración más avanzadas de las salidas de las luces. Existe la posibilidad, por ejemplo, para una locomotora que disponga de generador de humo, activar la tecla de las luces para que se enciendan las dos luces (trasera y delantera) y con otra tecla de función activar el generador de humo.

Si las luces no se encienden, o parpadean a cada paso de velocidad, lo más probable es que los pasos de velocidad de la central y el decodificador no concuerden. Verificar los dos dispositivos y contrastar que los dos estén configurados de la misma manera.

Funcionamiento en modo analógico: Todos los decodificadores suelen tener un reconocimiento de funcionamiento en analógico. Bajo ciertas circunstancias, los reguladores analógicos de corriente pulsante pueden hacer que el decodificador “piense” que está funcionando bajo un sistema digital, con lo que puede bloquearse y hacer que la locomotora se detenga. Es un problema sin fácil solución, pues normalmente no hay una opción de ajuste en los decodificadores de “sólo funcionamiento analógico”. Lo más frecuente es que los ajustes sean “sólo digital” y “analógico y digital”. Algunos decodificadores vienen con la primera opción predeterminada.

Cuando hay problemas de funcionamiento y no se sabe si son debidos a una incorrecta configuración de alguna de las CVs, es interesante hacer un “reset” (puesta a cero) al decodificador, con lo que todas las variables volverán a su estado predeterminado (y por tanto la dirección a la 3, que es la que traen por defecto en la configuración de fábrica). Algunos modelos soportan dos tipos de reset: hard y soft. En éstos, si se hace un reset soft no se altera el valor de determinadas variables (por ejemplo, la tabla de velocidad). Esto vendrá indicado en su manual. Lo habitual es que el reset se realice introduciendo un valor en la CV8, aunque esto no está estandarizado. En los decodificadores Lenz y de otras marcas, se realiza programando la CV8 con el valor 33 (otros mediante CV8=8). En los CT Electronic, además de este reset, se permite realizar uno por hard poniendo la CV1=0. Los Uhlenbrock usan la CV59=1 para el reset.

Si la central sólo permite 99 direcciones, probablemente será posible programar en las CVs valores superiores a esta cifra. Hay algunos modelos (por ejemplo, los del fabricante austriaco CT-Elektronik) que disponen de una CV (CV53), que permite "sumar" 100 ó 200 al valor introducido si previamente se programa con los valores 1 ó 2. De esta manera se pueden programar valores por encima de 99. No olvidar volver a programar esta CV especial con el valor estándar una vez finalizado el proceso a fin de evitar introducir valores incorrectos en otras CV.

Ajustes del motor

La CV9 es una de las variables más interesantes en cuanto al ajuste del motor. Regula la manera en cómo se envían los pulsos al motor, con lo que retocando este valor se consigue que la locomotora arranque o vaya más suave a bajas velocidades. Consejos:

a) Casi todas las locomotoras de motores antiguos de 3 polos (como por ejemplo de Trix) suelen ser bastante bruscos en el arranque y a bajas velocidades, pegando "tirones". En estos casos se puede suavizar este efecto mediante el ajuste de la CV9, a fin de obtener unos pulsos más cortos y seguidos. No obstante, esto aumentará ligeramente el zumbido del motor en las marchas lentas. Casi todos los decodificadores actuales suelen controlar bien los motores a baja velocidad. Consultar el manual del decodificador, ya que no todos utilizan la CV9 para este ajuste (Uhlenbrock, por ejemplo, utiliza la CV53).

b) La compensación de carga (BEMF en inglés) influye notablemente en el arranque de las locomotoras. Al tender a controlar el movimiento del motor, se consiguen arranques y paradas realmente precisas. La compensación de carga en muchos decodificadores es regulable:

   - Ajuste global de la intensidad. Alterando el valor de una CV se puede regular la intensidad de la BEMF. Esto puede ayudar en muchos casos a suavizar el funcionamiento. Se trata de buscar un compromiso entre el efecto de la compensación y una marcha adecuada.

   - Voltaje de referencia: Permite un ajuste aún más preciso de la BEMF. Es el voltaje para realizar la compensación. En el manual de cada fabricante se especifica cómo ajustarlo.

   - Ajuste de los parámetros P e I. Estos dos valores regulan la forma en que actúa la BEMF. Este ajuste varía de una marca a otra, y es cuestión de experimentar con ellos. Por norma general, hay que procurar mantener la proporción entre los dos valores. Si aumentamos uno, hay que aumentar también de forma proporcional el otro, en función de los valores de fábrica. Por ejemplo, los decodificadores de la marca CT-Elektronik suelen venir por defecto con las variables P e I ajustadas de la siguiente manera: CV51=80, CV52=40. La regulación entre una y otra es el doble. Si se aumenta manualmente, por ejemplo, la CV51=120, ajustar la CV52=60, manteniendo la proporción. No todos los decodificadores disponen de todos estos parámetros, ni los denomina de la misma manera, pero suelen actuar de manera similar, aunque su ajuste varía mucho de una locomotora a otra y del tipo de motor utilizado. Por ejemplo, los decodificadores ESU denominan a la intensidad de la compensación de carga como parámetro K. Algunas marcas comercializan algún modelo sin compensación de carga. Suelen ofrecer algunas variables con ajustes adicionales para suplir la ausencia de compensación de carga en el arranque. El inconveniente de estos decodificadores es que una vez ajustado el decodificador para conseguir un buen arranque, el funcionamiento difiere si la locomotora circula aislada o remolcando una composición.

En ocasiones suele haber problemas de comunicación entre la central y el decodificador a la hora de la programación. Hay veces en que, aunque la pantalla de la central indique “error” el valor realmente se almacena en la CV. Suele deberse a un voltaje de programación ligeramente más elevado de lo normal. La solución pasa por conectar en serie con la vía de programación (o la vía principal) una resistencia de aproximadamente 100 ohm y 1W. Esta resistencia debemos retirarla para el funcionamiento normal de la central. Solo sirve como solución momentánea.

Programas de ayuda a la programación

A pesar de que todas las prestaciones y opciones de un decodificador se pueden configurar a través de las CV del decodificador, hay otra opción que puede facilitar la tarea: los programas de ayuda a la programación. Estos programas son realmente útiles, porque, además, simplifican tareas como la de la construcción de manera gráfica de la curva de velocidad del decodificador. Además, se puede almacenar en el ordenador toda la configuración completa, disponiendo así de una base de datos de los parámetros de cada decodificador. Lógicamente, para utilizar estos programas se necesita que la central digital disponga de interfaz de conexión a un ordenador personal. Hay algunos fabricantes, caso de ESU, que cómo veremos más adelante ofrecen un programa para cargar los sonidos. Dichos programas permiten también su utilización con decodificadores convencionales, a fin de ajustar cómodamente los parámetros del decodificador. Obviamente, sólo sirve para los decodificadores de la propia marca. Zimo también dispone de programa propio que realiza la misma función. En cualquier caso, en Internet se encuentran disponibles programas que funcionan con decodificadores de manera genérica. Uno de los más conocidos es el Decoder-Pro, del proyecto JMRI (http://jmri.sourceforge.net/DecoderPro) que permite la programación de muchos modelos de decodificadores. Si un modelo no se encuentra en la base de datos del programa, se puede añadir de forma manual, o bien cargar los ficheros de definición (llamados “roster”) que van apareciendo en la web.

4. Patrón MVC

Algunas de las características más importantes de este software son:

El patrón de arquitectura **MVC** (Modelo Vista Controlador) es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del workflow de la aplicación). Es decir, separa la lógica de negocio de la interfaz de usuario, incrementando la reutilización y la flexibilidad.

De esta forma, dividimos el sistema en tres capas donde, tenemos la encapsulación de los datos, la interfaz o vista por otro y por último la lógica interna o controlador.

El patrón de arquitectura "modelo vista controlador", es una filosofía de diseño de aplicaciones, compuesta por:

- **Modelo**

  Contiene el núcleo de la funcionalidad (dominio) de la aplicación.

  Encapsula el estado de la aplicación.

  No sabe nada / independiente del Controlador y la Vista.

- **Vista**

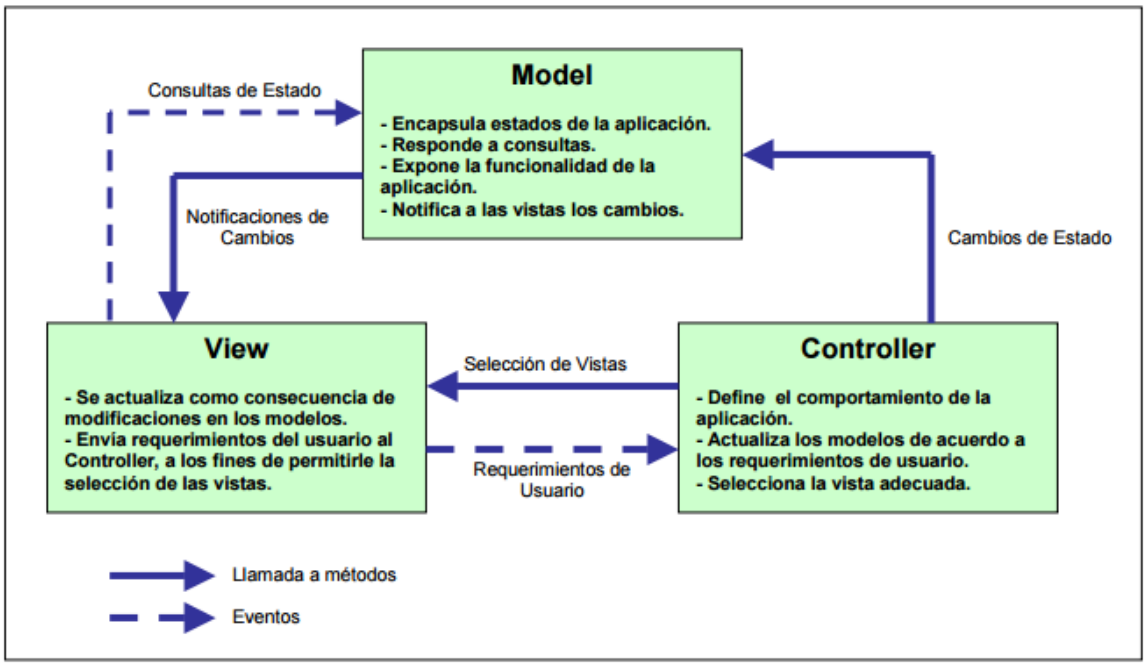
  Es la presentación del Modelo.

  Puede acceder al Modelo, pero nunca cambiar su estado.

  Puede ser notificada cuando hay un cambio de estado en el Modelo.

- **Controlador**

  Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente.



MVC provee muchos beneficios en el diseño de nuestra aplicación.

- Separando el modelo de la vista permite incorporar múltiples presentaciones para los mismos datos, facilitando incorporación de nuevas tecnologías para las presentaciones.
- Separando el controlador de la vista, permite una selección en tiempo de ejecución de la vista apropiada basada en workflow, comportamientos del usuario o estado del modelo.
- Separar el controlador del modelo brinda la posibilidad de poder convertir acciones del usuario en el controlador, a funciones de la aplicación en el modelo.

La comunicación entre el modelo, la vista y el controlador se debe hacer de una manera estable, de forma que sea coherente con las iteraciones que el usuario realizara. Como es lógico la comunicación entre la vista y el controlador es bastante básica pues están diseñados para operar juntos, pero los modelos se comunican de una manera diferente, un poco más sutil.

El modelo puede estar asociado a múltiples vistas y controladores, pero cada vista solo puede ser asociada a un único controlador, por lo que deberán tener una variable de tipo controler que notificara a la vista cuál es su controlador o modelo asignado. De igual manera, el controlador tiene una variable llamada View que apunta a la vista. De esta manera, pueden enviarse mensajes directos el uno al otro y al mismo tiempo, a su modelo.

La vista es quien tiene la responsabilidad de establecer la comunicación entre los elementos del patrón MVC. Cuando la vista recibe un mensaje que concierne al modelo o al controlador, lo deja registrado como el modelo con el cual se comunicara y apunta con la variable controller al controlador asignado, enviándole al mismo su identificación para que el controlador establezca en su variable view el identificador de la vista y así puedan operar conjuntamente. El responsable de deshacer estas conexiones, seguirá siendo la vista, quitándose a sí misma como dependiente del modelo y liberando al controlador.

El patrón MVC es utilizado en múltiples frameworks:

- Java Swing
- Java Enterprise Edition (J2EE)
- XForms (Formato XML estándar del W3C para la especificación de un modelo de proceso de datos XML e interfaces de usuario como formularios web)
- GTK+ (escrito en C, toolkit creado por Gnome para construir aplicaciones gráficas, inicialmente para el sistema X Window)
- ASP.NET MVC Framework (Microsoft)
- Google Web Toolkit (GWT, para crear aplicaciones Ajax con Java)
- Apache Struts (framework para aplicaciones web J2EE)
- Ruby on Rails (framework para aplicaciones web con Ruby)

Flujo de control del patrón MVC

1) El usuario realiza una acción en la interfaz.
2) El controlador trata el evento de entrada.
3) Previamente se ha registrado.
4) El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo (si no es una mera consulta).
5) Se genera una nueva vista. La vista toma los datos del modelo.
6) El modelo no tiene conocimiento directo de la vista.
7) La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo.

MVC es un patrón de diseño enfocado a separar las responsabilidades dentro de nuestra aplicación y es muy utilizado en la web por su enfoque y las ventajas que ofrece con respecto a algunas otras formas o patrones de desarrollo de aplicaciones web.

Porque usar MVC:

- Se puede dividir la lógica de negocio del diseño, haciendo nuestro proyecto más escalable.
- Facilita el uso de URL amigables, importantes para el SEO (Posicionamiento web), la mayoría de frameworks MVC lo controlan.
- Muchos frameworks MVC ya incluyen librerías de Javascript como Jquery, lo que facilita la validación de formularios (Ej. Jquery.Validate) en el cliente y en el servidor.
- Se puede utilizar abstracción de datos, facilitando la realización de consultas a la base de datos.
- La mayoría de frameworks controlan el uso de la memoria Caché, hoy en día muy importante para el posicionamiento web, ya que buscadores como google dan prioridad a las webs que tengan menor tiempo de descarga.
- Un Framework MVC ayuda a controlar los recursos del servidor, evitando Bugs que puedan repercutir en el rendimiento.

5. Implementación de la base de datos

A continuación, se detallan todas las tablas de la base de datos (las claves primarias están subrayadas).

Tabla estación

- **idestacion:** es de tipo 'Integer
- **nombre:** es de tipo 'String' guarda el nombre de la estación, y va a ser el identificador de la estación.
- **población:** es de tipo 'String', se guardará la población donde este situada la estación.
- **direccion:** es de tipo 'String', se guardará la dirección donde este situada la estación.
- **provincia:** es de tipo 'String', se guardará la provincia donde este situada la estación.
- **tlfnoinfo:** es de tipo 'String', se guardará el teléfono de información que tiene la estación.

Tabla Flota

- **idtren:** es de tipo 'Integer
- **fabricante:** es de tipo 'String' y va a guardar el fabricante del tren.
- **fechainicioservicio:** es de tipo 'Date, se guardará el año de inicio del servicio del tren.
- **nombrecomercial:** es de tipo 'String', se guardará el nombre comercial que se le asigna al tren.
- **nombretecnico:** es de tipo 'String', se guardará el nombre técnico que tiene el tren y va a ser el identificador del tren.
- **nplazas:** es de tipo 'Integer, se guardará el número de plazas que tenga el tren.
- **tipotren:** es de tipo 'String', se guardará el tipo de tren, es decir si es de pasajeros, mercancías, etc.
- **trenescomprados**: es de tipo 'Integer, se guardará el número de unidades que han sido compradas de este modelo de tren.
- **trenesenservicio**: es de tipo 'Integer, se guardará el número de unidades de las que han sido compradas que han sido puestas en servicio del tren en cuestión.
- **Velocidadmax**: es de tipo 'Integer, se guardará la velocidad máxima en Km/hora a la que puede ir el tren.

Tabla Ruta

- **idruta:** es de tipo 'Integer'.
- **nombreruta:** es de tipo 'String', se guardará el identificador de la ruta.
- **tipotren:** es de tipo 'String', se guardará el tipo de tren que circula (pasajeros, mercancías, etc.).
- **nombretrencomercial:** es de tipo 'String', se guardará el nombre comercial del tren.
- **nombretecnico:** es de tipo 'String', se guardará el nombre técnico del tren. Es clave foránea.

- **estacionsalida:** es de tipo ‘String’, se guardará el nombre de la estación de salida del tren. Es clave foránea.
- **estacionllegada:** es de tipo ‘String’, se guardará el nombre de la estación de llegada del tren. Es clave foránea.
- **notas:** es de tipo ‘String’, se guardará la fecha y la hora.

Tabla Scada

- **idscada:** es de tipo ‘Integer’. se guardará el identificador del dato Scada.
- **fechayHora:** es de tipo ‘Date’, se guardará la fecha.
- **sensor0est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 0 de la estación 1, si está a ‘false’, indica que el sensor no está activo y no está detectando al tren, si está a ‘true’, indica que el sensor está activo, por lo que está detectando el paso del tren.
- **sensor1est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 1 de la estación 1.
- **sensor2est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 2 de la estación 1.
- **sensor3est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 3 de la estación 1.
- **sensor4est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 4 de la estación 1.
- **sensor5est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 5 de la estación 1.
- **sensor6est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 6 de la estación 1.
- **sensor7est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 7 de la estación 1.
- **sensor8est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 8 de la estación 1.
- **sensor9est1:** es de tipo ‘Boolean’, se guardará el estado del sensor 9 de la estación 1.
- **sensor0est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 0 de la estación 2.
- **sensor1est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 1 de la estación 2.
- **sensor2est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 2 de la estación 2.
- **sensor3est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 3 de la estación 2.
- **sensor4est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 4 de la estación 2.
- **sensor5est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 5 de la estación 2.
- **sensor6est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 6 de la estación 2.
- **sensor7est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 7 de la estación 2.
- **sensor8est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 8 de la estación 2.
- **sensor9est2:** es de tipo ‘Boolean’, se guardará el estado del sensor 9 de la estación 2.
- **velPasajeros:** de tipo ‘Integer’, se guarda la salida analógica de la velocidad pasajeros.
- **velMercancias:** ‘Integer’, se guarda la salida analógica de la velocidad mercancías.
- **marcadorVeloPasaje1:** es de tipo ‘Integer’, se guardará la salida analógica escalada en una dirección del tren de pasajeros.

- **marcadorVeloPasaje2:** es de tipo 'Integer', se guardará la salida analógica escalada en la otra dirección del tren de pasajeros.
- **marcadorVeloMercan1:** es de tipo 'Integer', se guardará la salida analógica escalada en una dirección del tren de mercancías.
- **marcadorVeloMercan2:** es de tipo 'Integer', se guardará la salida analógica escalada en la otra dirección del tren de mercancías.
- **semaf1Est1Rojo:** es de tipo 'Boolean', se guardará el estado del semáforo 1 de la estación 1 Rojo, si está a 'true', el led rojo del semáforo estará activo, se está a 'false' el led rojo del semáforo estará apagado.
- **semaf1Est1Verde:** de tipo 'Boolean', guarda el estado del semáforo 1 de la estación 1.
- **Semaf2Est1Rojo:** de tipo 'Boolean', se guarda el estado del semáforo 2 de la estación 1
- **Semaf2Est1Verde:** de tipo 'Boolean', guarda el estado del semáforo 2 de la estación 1.
- **Semaf3Est1Rojo:** de tipo 'Boolean', guarda el estado del semáforo 3 de la estación 1.
- **Semaf3Est1Verde:** de tipo 'Boolean', guarda el estado del semáforo 3 de la estación 1.
- **Semaf4Est1Rojo:** de tipo 'Boolean', guarda el estado del semáforo 4 de la estación 1.
- **Semaf4Est1Verde:** de tipo 'Boolean', guarda el estado del semáforo 4 de la estación 1.
- **Semaf5Est1Rojo:** de tipo 'Boolean', guarda el estado del semáforo 5 de la estación 1.
- **Semaf5Est1Verde:** de tipo 'Boolean', guarda el estado del semáforo 5 de la estación 1.
- **Semaf6Est1Rojo:** de tipo 'Boolean', guarda el estado del semáforo 6 de la estación 1.
- **Semaf6Est1Verde:** de tipo 'Boolean', guarda el estado del semáforo 6 de la estación 1.
- **Semaf7Est1Rojo:** de tipo 'Boolean', guarda el estado del semáforo 7 de la estación 1.
- **Semaf7Est1Verde:** de tipo 'Boolean', guarda el estado del semáforo 7 de la estación 1.
- **Semaf8Est1Rojo:** de tipo 'Boolean', guarda el estado del semáforo 8 de la estación 1.
- **Semaf8Est1Verde:** de tipo 'Boolean', guarda el estado del semáforo 8 de la estación 1.
- **allSemafEst1:** 'Integer', se guarda los 2 bytes de salida de los semáforos de la estac. 1.
- **byte1SemafEst1:** 'Integer', guarda el byte de menor peso de salida de los semáf esta. 1.
- **byte2SemafEst1:** 'Integer', guarda el byte de mayor peso de salida de los semáf esta. 1.
- **semaf1Est2Rojo:** de tipo 'Boolean', guarda el estado del semáforo 1 de la estación 2.
- **semaf1Est2Verde:** de tipo 'Boolean', guarda el estado del semáforo 1 de la estación 2.
- **Semaf2Est2Rojo:** de tipo 'Boolean', guarda el estado del semáforo 2 de la estación 2.
- **Semaf2Est2Verde:** de tipo 'Boolean', guarda el estado del semáforo 2 de la estación 2.
- **Semaf3Est2Rojo:** de tipo 'Boolean', guarda el estado del semáforo 3 de la estación 2.
- **Semaf3Est2Verde:** de tipo 'Boolean', guarda el estado del semáforo 3 de la estación 2.
- **Semaf4Est2Rojo:** de tipo 'Boolean', guarda el estado del semáforo 4 de la estación 2.
- **Semaf4Est2Verde:** de tipo 'Boolean', guarda el estado del semáforo 4 de la estación 2.

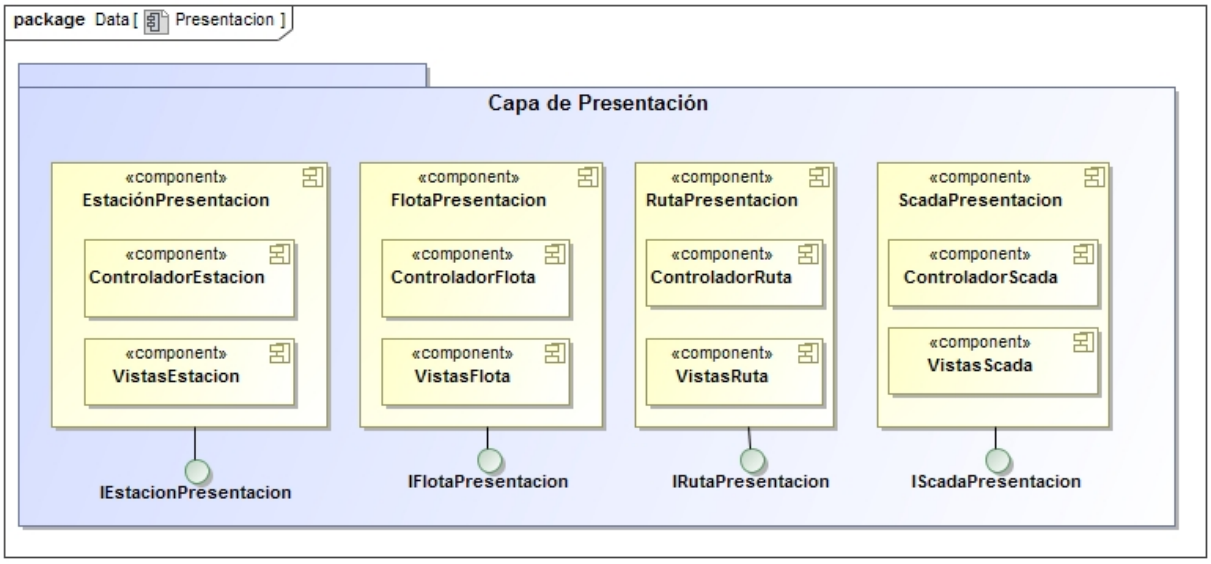
- **Semaf5Est2Rojo:** de tipo ‘Boolean’, guarda el estado del semáforo 5 de la estación 2.
- **Semaf5Est2Verde:** de tipo ‘Boolean’, guarda el estado del semáforo 5 de la estación 2.
- **Semaf6Est2Rojo:** de tipo ‘Boolean’, guarda el estado del semáforo 6 de la estación 2.
- **Semaf6Est2Verde:** de tipo ‘Boolean’, guarda el estado del semáforo 6 de la estación 2.
- **Semaf7Est2Rojo:** de tipo ‘Boolean’, guarda el estado del semáforo 7 de la estación 2.
- **Semaf7Est2Verde:** de tipo ‘Boolean’, guarda el estado del semáforo 7 de la estación 2.
- **Semaf8Est2Rojo:** de tipo ‘Boolean’, guarda el estado del semáforo 8 de la estación 2.
- **Semaf8Est2Verde:** de tipo ‘Boolean’, guarda el estado del semáforo 8 de la estación 2.
- **Semaf9Est2Rojo:** de tipo ‘Boolean’, guarda el estado del semáforo 9 de la estación 2.
- **Semaf9Est2Verde:** de tipo ‘Boolean’, guarda el estado del semáforo 9 de la estación 2.
- **semaf10Est2Rojo:** de tipo ‘Boolean’, guarda el estado del semáforo 10 de la estación 2.
- **semaf10Est2Verde:** tipo ‘Boolean’, guarda el estado del semáforo 10 de la estación 2.
- **allSemafEst2:** ‘Integer’, se guarda los 2 bytes de salida de los semáforos de la estac. 2.
- **byte1SemafEst2:** ‘Integer’, guarda el byte de menor peso de salida de los semáf esta. 2.
- **byte2SemafEst2:** ‘Integer’, guarda el byte de mayor peso de salida de los semáf esta. 2.
- **desvio1:** es de tipo ‘Boolean’, guarda el estado del desvío 1, si está a ‘false’, indica que el desvío no está activo y por lo tanto está en su posición inicial, es decir en recto, si está a ‘true’, indica que el desvío está activo, por lo que el desvío está curvo.
- **desvio2:** es de tipo ‘Boolean’, se guardará el estado del desvío 2.
- **desvio3:** es de tipo ‘Boolean’, se guardará el estado del desvío 3.
- **desvio4:** es de tipo ‘Boolean’, se guardará el estado del desvío 4.
- **desvio5:** es de tipo ‘Boolean’, se guardará el estado del desvío 5.
- **desvio6:** es de tipo ‘Boolean’, se guardará el estado del desvío 6.
- **sistema Automático:** de tipo ‘Boolean’, a ‘false’, indica que el sistema Automático no está activo, si está a ‘true’, indica que el sistema Automático estará activo.
- **sistManualLibre:** es de tipo ‘Boolean’, si está a ‘false’, indica que el sistema Manual Libre no está activo, si está a ‘true’, indica que el sistema Manual Libre estará activo.
- **sistManual:** es de tipo ‘Boolean’, si está a ‘false’, indica que el sistema Manual no está activo, si está a ‘true’, indica que el sistema Manual estará activo.
- **sistemaOn:** de tipo ‘Boolean’, a ‘false’, indica que el sistema no está activo, es decir está a OFF, y si está a ‘true’, indica que el sistema estará activo, es decir, estará a ON.
- **emergencia:** Boolean’, a ‘false’, la emergencia no activa, a ‘true’, la emergencia activa.
- **trenPasajeActivo:** ‘Boolean’, a ‘false’, tren de Pasajeros no está activo, a ‘true’, activo.
- **trenMercanActivo:** ‘Boolean’, a ‘false’, tren de Mercancías no activo, a ‘true’, activo.
- **condInicialesAuto:** es de tipo ‘Boolean’, se guardará las condiciones iniciales.

- **liberarFrenoPasaje:** de tipo ‘Boolean’, a ‘false’, liberar Freno Pasajeros no está activo, a ‘true’, indica que liberar Freno Pasajeros está activo.
- **liberarFrenoMercan:** de tipo ‘Boolean’, a ‘false’, liberar Freno Mercancías no está activo, a ‘true’, indica que liberar Freno Mercancías está activo.
- **rutaSeleccionada:** es de tipo ‘Integer’, se guardará en la ruta seleccionada.
- **permisoRuta1:** es de tipo ‘Boolean’, indica el permiso para la ruta 1.
- **ruta1seleccionada:** es de tipo ‘Boolean’, indica el si la ruta 1 está seleccionada.
- **permisoTodasRutas:** es de tipo ‘Integer’, indica si hay permiso para todas las rutas.
- **ruta2seleccionada:** es de tipo ‘Boolean’, indica el si la ruta 2 esta seleccionada.
- **permisoRuta2:** es de tipo ‘Boolean’, indica el permiso para la ruta 2.
- **bucleRutasSeleccionado:** es de tipo ‘Boolean’, indica el bucle de rutas seleccionado.
- **permisoBucle1:** es de tipo ‘Boolean’, indica si hay permiso para el bucle 1.
- **permisoTodosBucles**: es de tipo ‘Integer’, indica si hay permiso para todos los bucles.
- **bucleRuta1seleccionado**: ‘Boolean’, indica si esta seleccionado el bucle de la ruta1.
- **voltaje319316**: es de tipo ‘BigDecimal’, el voltaje que la salida analógica del autómata está dando al mando de control del sistema digital que controla a este tren (pasajeros).
- **voltaje319402**: es de tipo ‘BigDecimal’, el voltaje que la salida analógica del autómata está dando al mando de control del sistema digital que controla a este tren (mercancías).

6. Diagramas

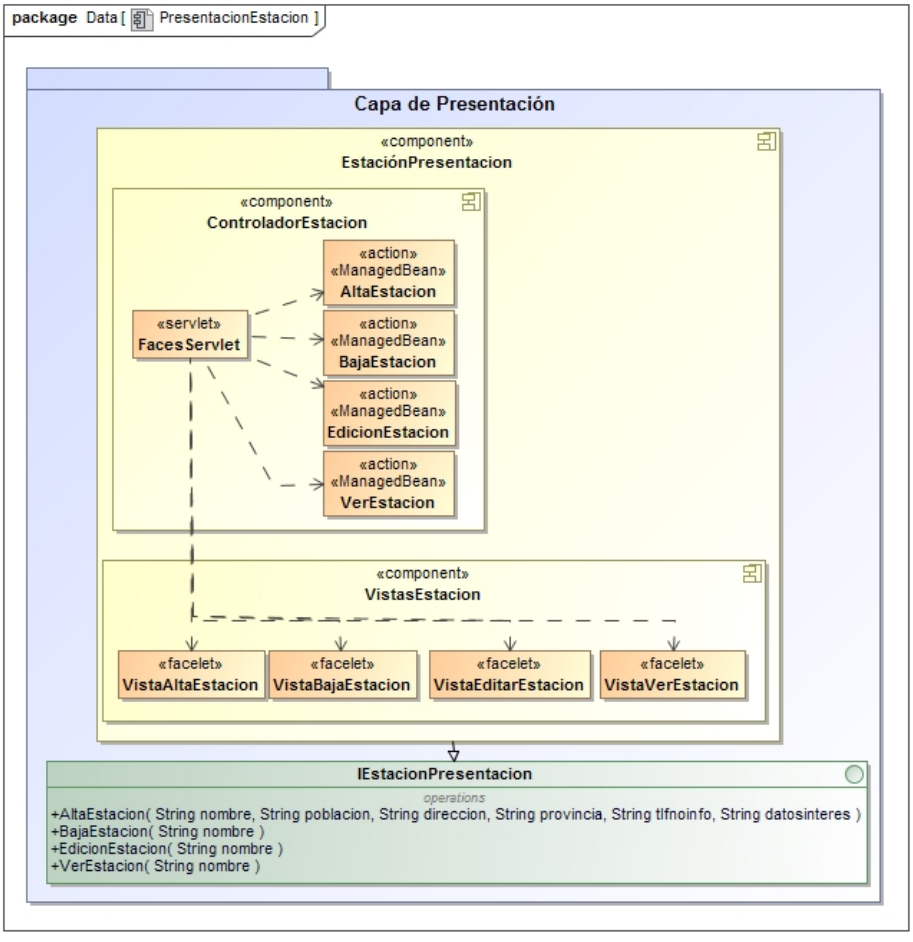
6.1. Diagramas de componentes

Siguiendo el patrón MVC, y suponiendo que el modelo esta implementado en el componente de negocio, se puede dividir el componente de presentación en el componente que hará de controlador y el componente que hará de vista, como queda en la siguiente figura:

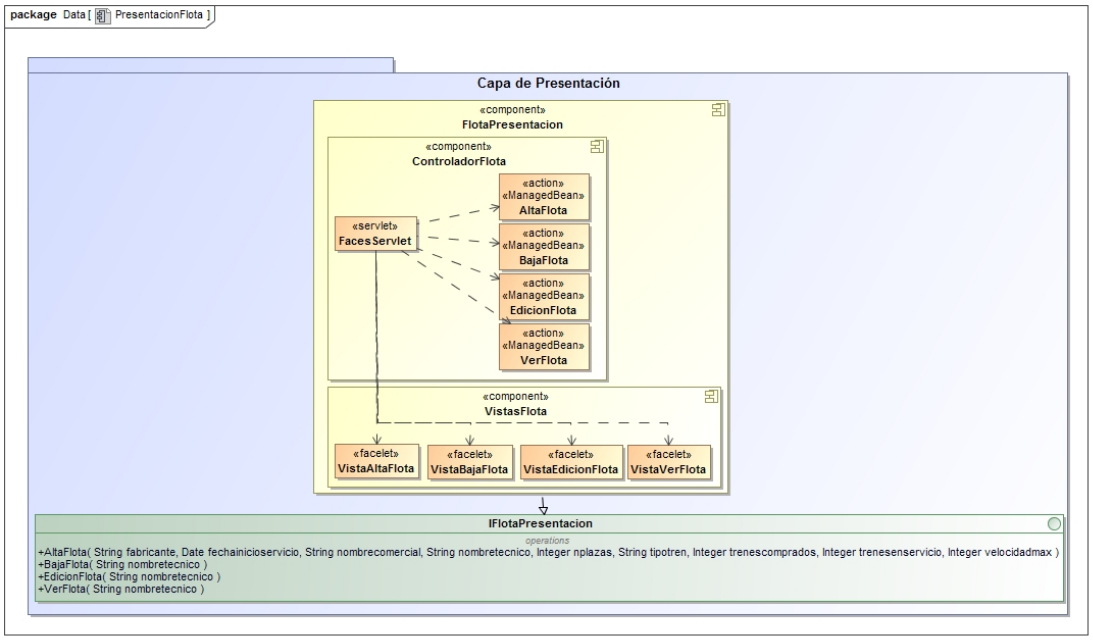


En el siguiente gráfico se puede ver, el diagrama de componentes software refinado para la capa de presentación, se puede observar el controlador de la capa (el servlet Faces Servlet), las acciones definidas con Managed Bean y las vistas implementadas mediante Facelets.

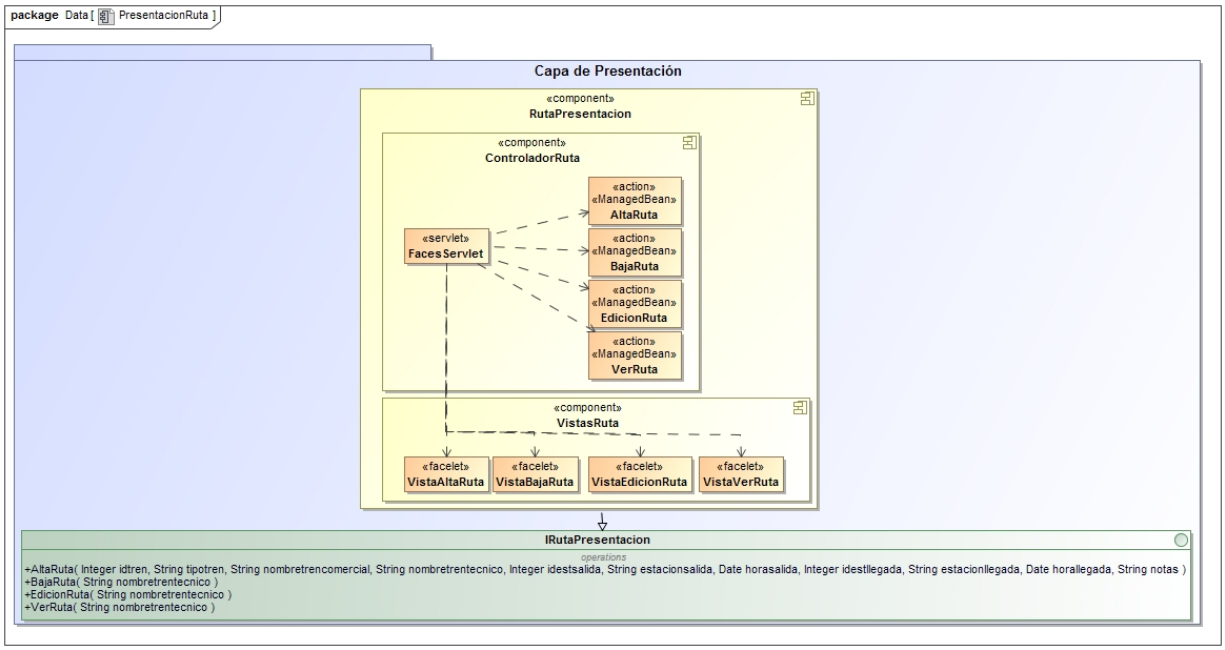
Estación



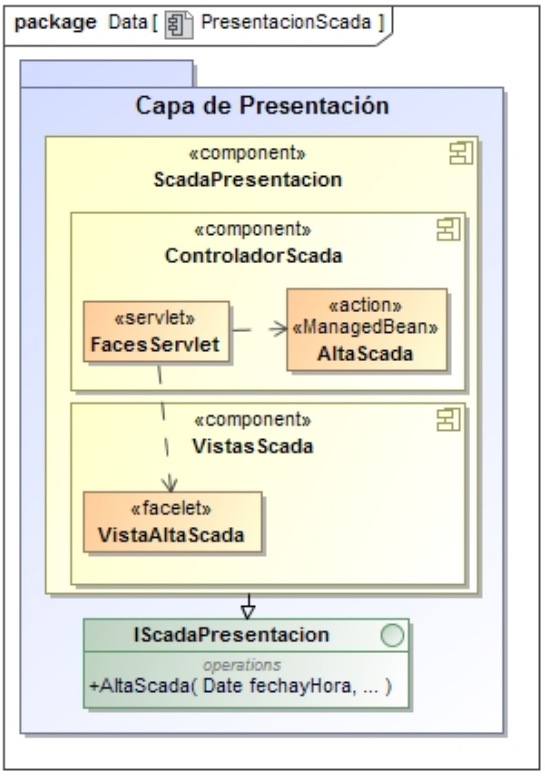
Flota


package Data [PresentacionFlota]
Capa de Presentación
«component»
FlotaPresentacion
«component»
ControladorFlota
«servlet»
FacesServlet
«action»
«ManagedBean»
AltaFlota
«action»
«ManagedBean»
BajaFlota
«action»
«ManagedBean»
EdicionFlota
«action»
«ManagedBean»
VerFlota
«component»
VistasFlota
«facelet»
VistaAltaFlota
«facelet»
VistaBajaFlota
«facelet»
VistaEdicionFlota
«facelet»
VistaVerFlota
IFlotaPresentacion
operations
+AltaFlota(String fabricante, Date fechainicioservicio, String nombrecomercial, String nombretecnico, Integer nplazas, String tipotren, Integer trenescomprados, Integer trenesenservicio, Integer velocidadmax)
+BajaFlota(String nombretecnico)
+EdicionFlota(String nombretecnico)
+VerFlota(String nombretecnico)

Ruta


package Data [PresentacionRuta]
Capa de Presentación
«component»
RutaPresentacion
«component»
ControladorRuta
«servlet»
FacesServlet
«action»
«ManagedBean»
AltaRuta
«action»
«ManagedBean»
BajaRuta
«action»
«ManagedBean»
EdicionRuta
«action»
«ManagedBean»
VerRuta
«component»
VistasRuta
«facelet»
VistaAltaRuta
«facelet»
VistaBajaRuta
«facelet»
VistaEdicionRuta
«facelet»
VistaVerRuta
IRutaPresentacion
operations
+AltaRuta(Integer idtren, String tipotren, String nombretrencomercial, String nombretrentecnico, Integer idestsalida, String estacionsalida, Date horasalida, Integer idestllegada, String estacionllegada, Date horallegada, String notas)
+BajaRuta(String nombretrentecnico)
+EdicionRuta(String nombretrentecnico)
+VerRuta(String nombretrentecnico)

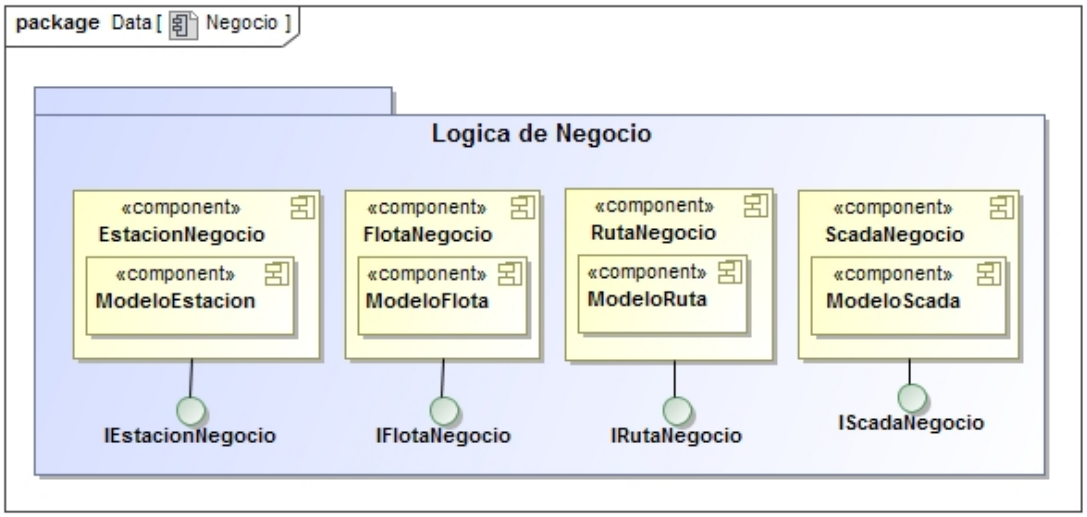
Scada



Nota: Las operaciones del Interfaz 'IScadaPresentacion', se han reducido para poder mostrar la figura. La totalidad de los datos necesarios para la operación se detalla en la explicación de la base de datos.

Componente de Negocio.

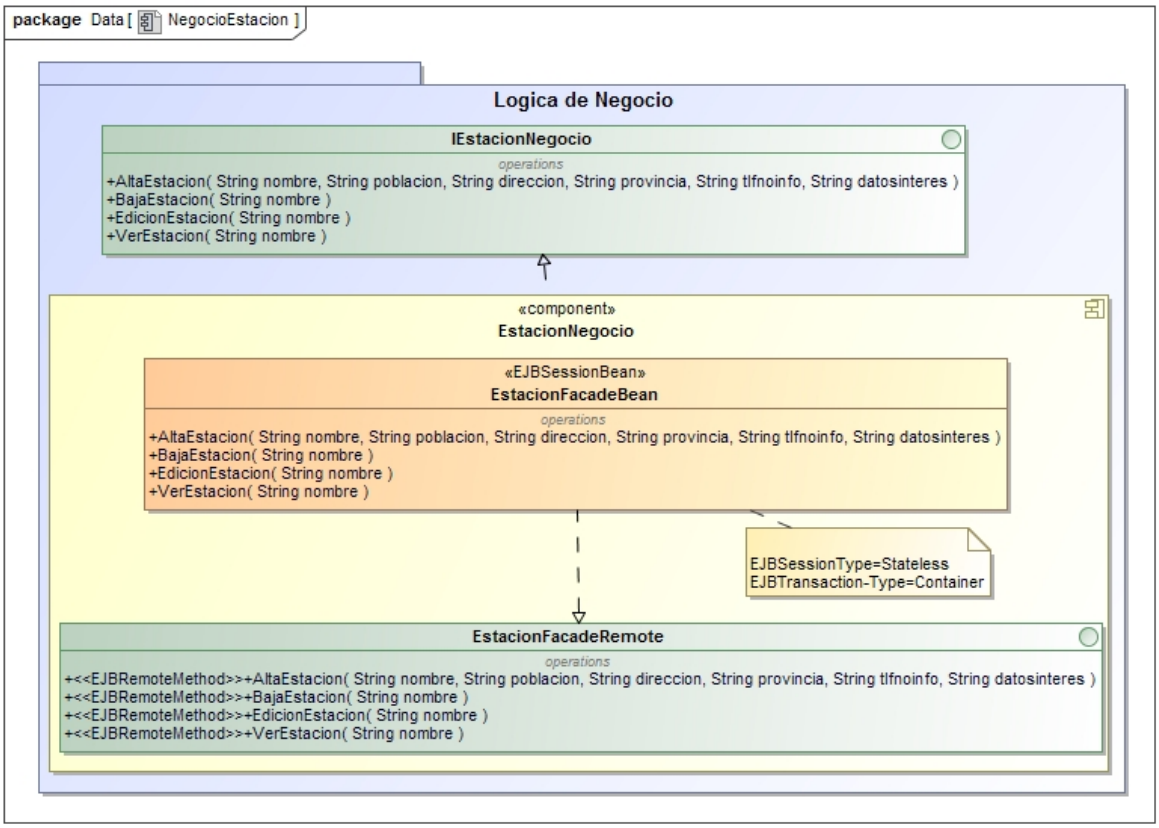
Contiene la lógica de negocio de las operaciones que los usuarios pueden efectuar, alta, baja, edición y ver.
Siguiendo con el patrón MVC, en la capa de negocio está implementado el modelo, como queda en la siguiente figura:



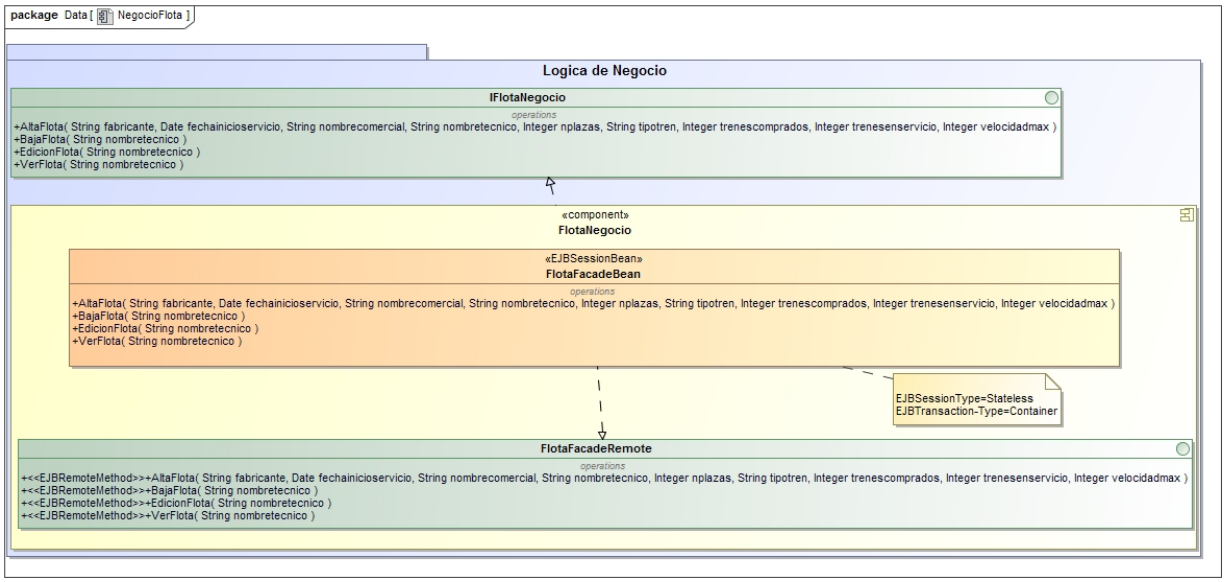
El modelo es el encargado de acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento. También es el responsable de definir las reglas de negocio (la funcionalidad del sistema) y de llevar un registro de las vistas y controladores del sistema. El modelo se puede asociar a múltiples vistas y controladores.

En el siguiente gráfico se muestra el diagrama de componentes software refinado de la capa de negocio, que sigue un patrón Fachada (Facade). Las operaciones que realiza son síncronas y no tienen estado y será el contenedor JavaEE quien gestiones las transacciones. De esta forma se emplea un EJB de sesión sin estado para que el contenedor JavaEE resuelva la complejidad de las comunicaciones remotas, el comportamiento transaccional y la gestión de la seguridad.

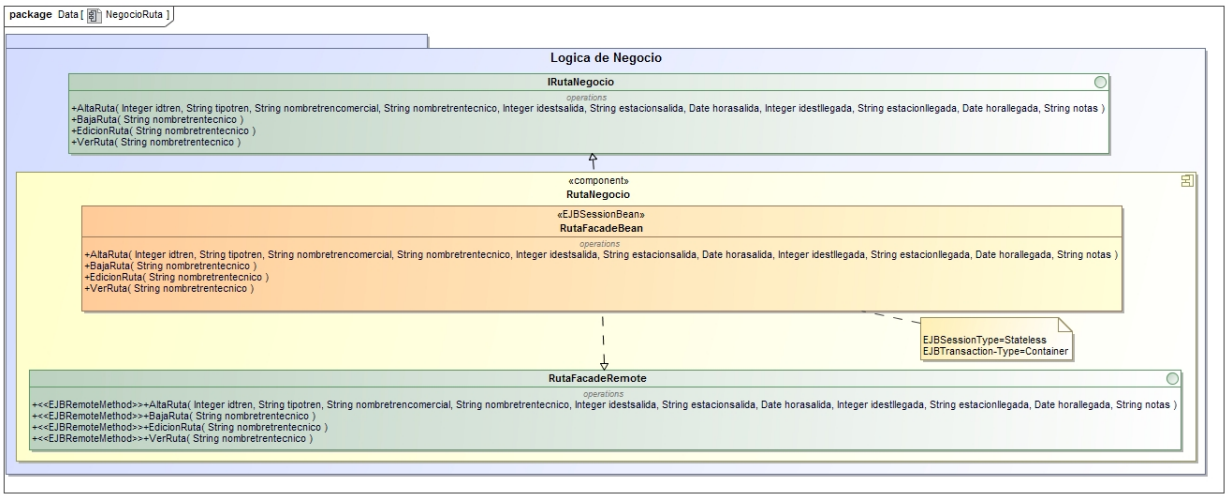
Estación


package Data [NegocioEstacion]
Logica de Negocio
IEstacionNegocio
operations
+AltaEstacion(String nombre, String poblacion, String direccion, String provincia, String tlfnoinfo, String datosinteres)
+BajaEstacion(String nombre)
+EdicionEstacion(String nombre)
+VerEstacion(String nombre)
«component»
EstacionNegocio
«EJBSessionBean»
EstacionFacadeBean
operations
+AltaEstacion(String nombre, String poblacion, String direccion, String provincia, String tlfnoinfo, String datosinteres)
+BajaEstacion(String nombre)
+EdicionEstacion(String nombre)
+VerEstacion(String nombre)
EJBSessionType=Stateless
EJBTransaction-Type=Container
EstacionFacadeRemote
operations
+<<EJBRemoteMethod>>+AltaEstacion(String nombre, String poblacion, String direccion, String provincia, String tlfnoinfo, String datosinteres)
+<<EJBRemoteMethod>>+BajaEstacion(String nombre)
+<<EJBRemoteMethod>>+EdicionEstacion(String nombre)
+<<EJBRemoteMethod>>+VerEstacion(String nombre)

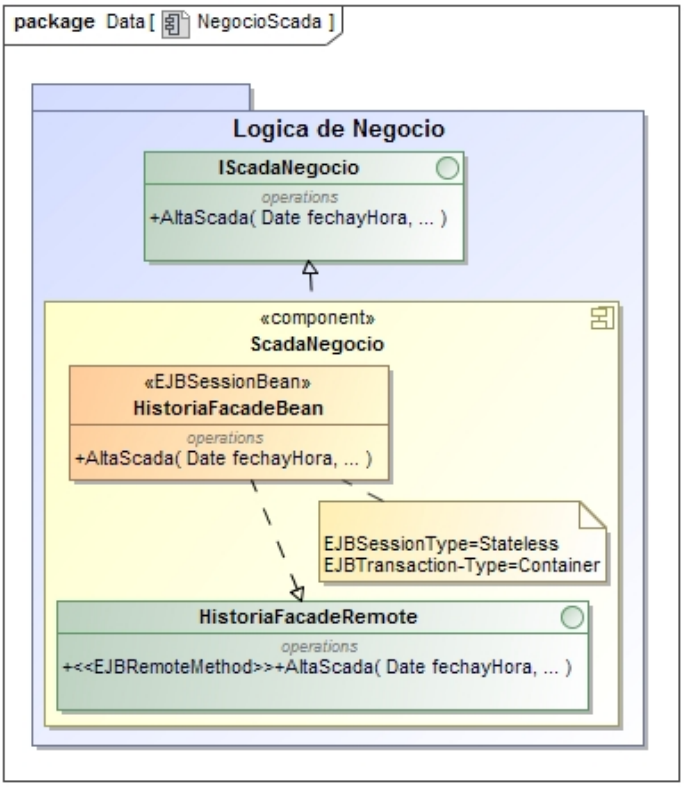
Flota


package Data [NegocioFlota]
Logica de Negocio
IFlotaNegocio
operations
+AltaFlota(String fabricante, Date fechainicioservicio, String nombrecomercial, String nombretecnico, Integer nplazas, String tipotren, Integer trenescomprados, Integer trenesenservicio, Integer velocidadmax)
+BajaFlota(String nombretecnico)
+EdicionFlota(String nombretecnico)
+VerFlota(String nombretecnico)
«component»
FlotaNegocio
«EJBSessionBean»
FlotaFacadeBean
operations
+AltaFlota(String fabricante, Date fechainicioservicio, String nombrecomercial, String nombretecnico, Integer nplazas, String tipotren, Integer trenescomprados, Integer trenesenservicio, Integer velocidadmax)
+BajaFlota(String nombretecnico)
+EdicionFlota(String nombretecnico)
+VerFlota(String nombretecnico)
EJBSessionType=Stateless
EJBTransaction-Type=Container
FlotaFacadeRemote
operations
+<<EJBRemoteMethod>>+AltaFlota(String fabricante, Date fechainicioservicio, String nombrecomercial, String nombretecnico, Integer nplazas, String tipotren, Integer trenescomprados, Integer trenesenservicio, Integer velocidadmax)
+<<EJBRemoteMethod>>+BajaFlota(String nombretecnico)
+<<EJBRemoteMethod>>+EdicionFlota(String nombretecnico)
+<<EJBRemoteMethod>>+VerFlota(String nombretecnico)

Ruta


package Data [NegocioRuta]
Logica de Negocio
IRutaNegocio
operations
+AltaRuta(Integer idtren, String tipotren, String nombretrencomercial, String nombretrentecnico, Integer idestsalida, String estacionsalida, Date horasalida, Integer idestllegada, String estacionllegada, Date horallegada, String notas)
+BajaRuta(String nombretrentecnico)
+EdicionRuta(String nombretrentecnico)
+VerRuta(String nombretrentecnico)
«component»
RutaNegocio
«EJBSessionBean»
RutaFacadeBean
operations
+AltaRuta(Integer idtren, String tipotren, String nombretrencomercial, String nombretrentecnico, Integer idestsalida, String estacionsalida, Date horasalida, Integer idestllegada, String estacionllegada, Date horallegada, String notas)
+BajaRuta(String nombretrentecnico)
+EdicionRuta(String nombretrentecnico)
+VerRuta(String nombretrentecnico)
EJBSessionType=Stateless
EJBTransaction-Type=Container
RutaFacadeRemote
operations
+<<EJBRemoteMethod>>+AltaRuta(Integer idtren, String tipotren, String nombretrencomercial, String nombretrentecnico, Integer idestsalida, String estacionsalida, Date horasalida, Integer idestllegada, String estacionllegada, Date horallegada, String notas)
+<<EJBRemoteMethod>>+BajaRuta(String nombretrentecnico)
+<<EJBRemoteMethod>>+EdicionRuta(String nombretrentecnico)
+<<EJBRemoteMethod>>+VerRuta(String nombretrentecnico)

Scada



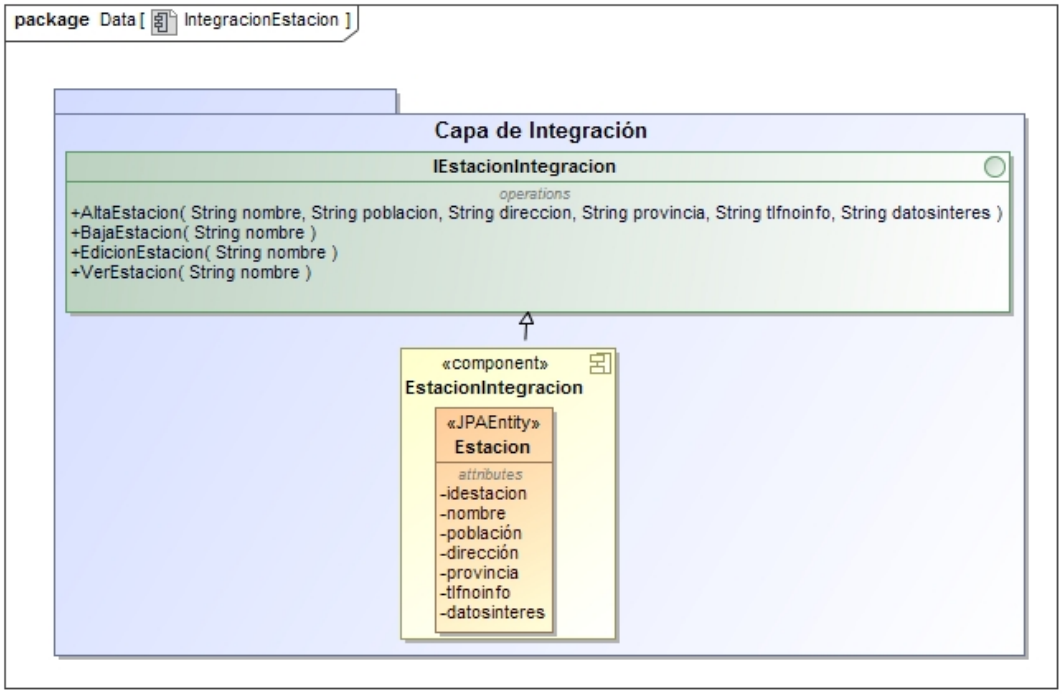
Componente de Integración.

Contiene la lógica que permite al componente de negocio interactuar con los almacenes de datos.
Los datos que va a utilizar la aplicación, se encuentran en la base de datos y el sistema ofrece la funcionalidad de alta, baja, edición y ver sobre estación, flota y ruta, y la opción de alta para Scada.

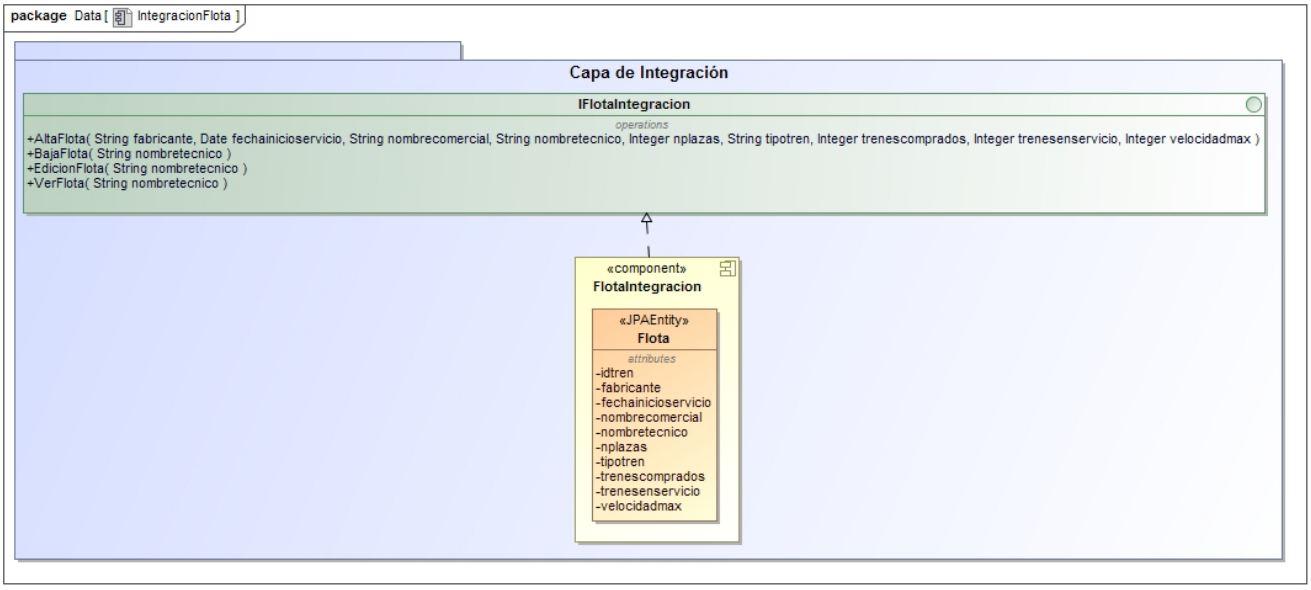
En el siguiente gráfico se muestra el diagrama de componentes software refinado de la capa de integración. La capa de integración va a ser la que resuelva la persistencia del sistema. Se va a representar los datos con entidades JPA que han de ser almacenados en la base de datos.

En las siguientes figuras se muestra como queda la capa de integración para Estación, Flota, Ruta y Scada.

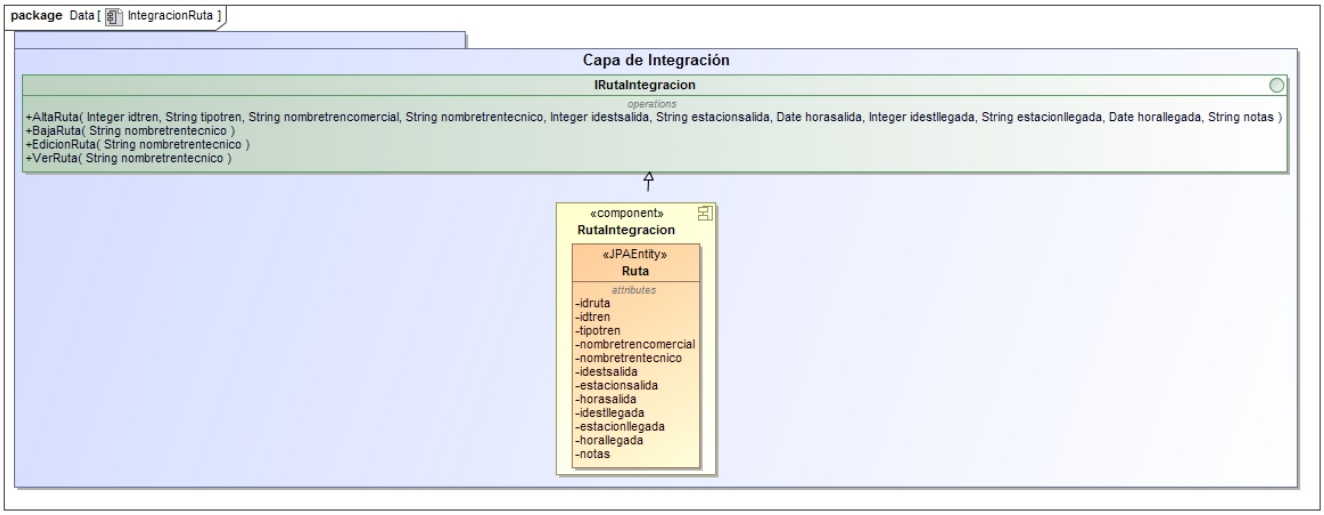
Estación



Flota



Ruta


package Data [IntegracionRuta]
Capa de Integración
IRutaIntegracion
operations
+AltaRuta(Integer idtren, String tipotren, String nombretrencomercial, String nombretrentecnico, Integer idestsalida, String estacionsalida, Date horasalida, Integer idestllegada, String estacionllegada, Date horallegada, String notas)
+BajaRuta(String nombretrentecnico)
+EdicionRuta(String nombretrentecnico)
+VerRuta(String nombretrentecnico)
«component»
RutaIntegracion
«JPAEntity»
Ruta
attributes
-idruta
-idtren
-tipotren
-nombretrencomercial
-nombretrentecnico
-idestsalida
-estacionsalida
-horasalida
-idestllegada
-estacionllegada
-horallegada
-notas

Scada

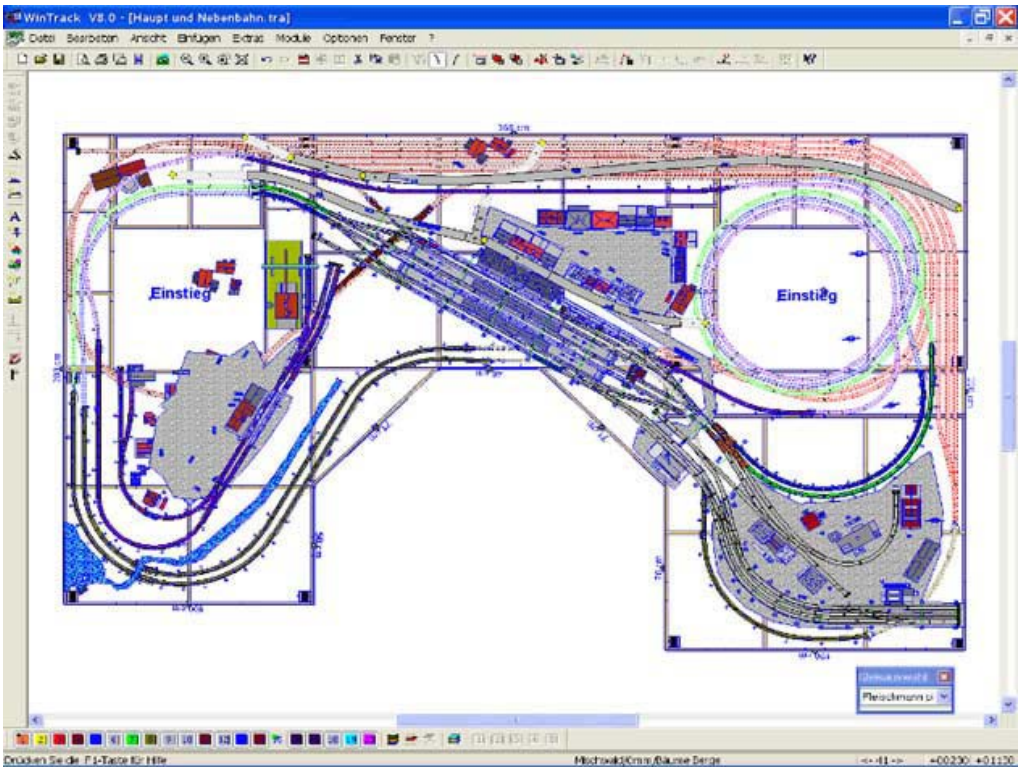
Nota: Los campos se han reducido para poder mostrar la figura. Habría un campo para cada sensor de la estación1 y lo mismo para la estación2. Solo se ha indicado un sensor para la estación 1 (sensor3est1) y otro para la 2 (sensor3est2). Lo mismo ocurre con los desvíos, solamente se ha indicado 'desvio1', cuando en realidad hay 6 desvíos. Para los semáforos, solamente se ha indicado uno para color rojo (semaf1Est1Rojo) y otro para color verde (semaf1Est1Verde) de la estación 1 y lo mismo para la estación 2 (semaf1Est2Rojo, semaf1Est2Verde). En total hay 18 semáforos, (18 rojos y 18 verdes).
La totalidad de los datos se detalla en la explicación de la base de datos.

7. Software de diseño de maqueta ferroviaria. Wintrack

Algunas de las características más importantes de este software son:

Diseño en 2D

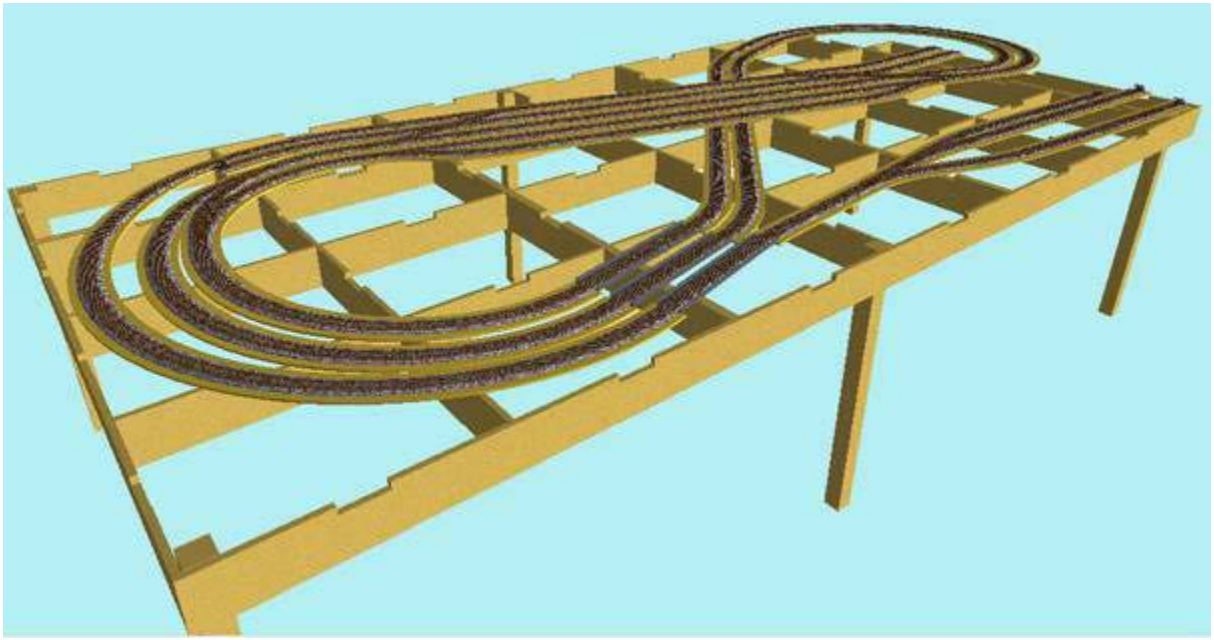
- Diseño de superficies hasta 15x15 metros.
- Librerías para Märklin (H0, Z, 1), Fleischmann (H0, N), Trix (H0, N), Roco (H0, H0e, N), Tillig (TT, H0), Peco (H0, N), Lima (H0), Arnold (N), Bemo (H0e/m) y LGB con todos los accesorios y elementos de vía (rotondas, señales, etc).
- Símbolos adicionales para señalización, túneles, tramos aislados, cotas, pendientes, alturas, árboles, etc.
- Edificios, carreteras y calles, ríos, etc.
- Uso fácil de la vía flexible adaptable con el mouse o mediante cuadro de diálogo si se desea calcular su radio de forma exacta.
- Diseño del tablero (base) directa con el ratón o mediante asistente.
- Selección rápida de los elementos de vía a colocar mediante una lista flotante que puede ser situada en cualquier posición de la pantalla.
- Posibilidad de mostrar números en los elementos de vía de forma automática o manual y de mostrar/ocultar las referencias de los elementos de vía.
- Permite copiar/pegar con múltiples elementos de vía. También permite guardar estos bloques si se usan de forma habitual para tener estructuras predefinidas.
- Hasta 12 capas que permiten colorear los elementos del diseño para separar elementos o secciones de vía.
- Unión automática de los elementos de vía tanto en rectas como en curvas.
- Los símbolos pueden moverse o rotar simplemente usando el mouse.
- Permite indicar los sentidos de circulación de las vías.
- Permite el diseño del cableado de la maqueta.
- Exporta los planos a WMF para poder cargarlos en otras aplicaciones como Microsoft Office.
- Genera una lista de compra con las referencias, cantidades y precios.
- Impresión a cualquier escala, incluyendo 1:1 para su fácil implantación en el tablero.
- Las pendientes de las vías se pueden indicar fácilmente y son calculables mediante asistentes para la correcta representación en el módulo 3D.



Ejemplo de diseño de maqueta ferroviaria (layout)

Diseño de la estructura

WinTrack permite el diseño de la base de la maqueta para su posterior construcción, permitiendo el diseño por cuadernas y facilitando su construcción.

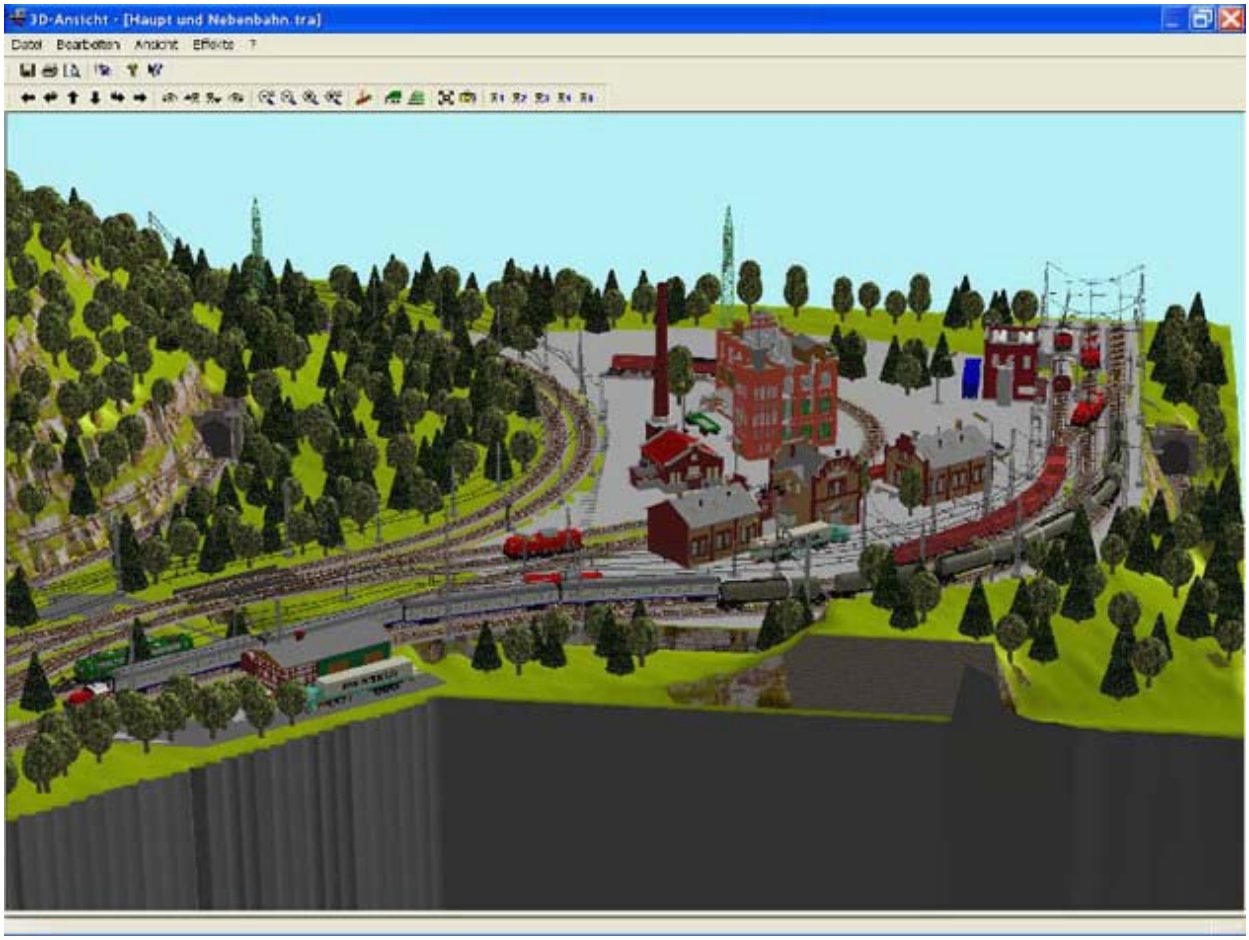


Ejemplo de vista 3D del soporte de la maqueta ferroviaria.

Diseño 3D

Un sólo clic y WinTrack genera la vista 3D a partir del esquema de vías y la información de cotas y alturas. No es preciso tener ninguna noción de CAD.

- Se puede generar la vista 3D y ser vista desde diferentes ángulos y lados.
- La posición de la cámara puede ser guardada para cada layout realizado.



Ejemplo de diseño en 3D de maqueta ferroviaria.

8. Kits de inicio de modelismo ferroviario Roco. Detalle del material utilizado para crear el layout.

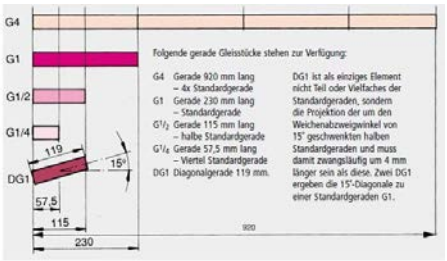
Se expone a continuación una breve descripción del material utilizado en el layout:

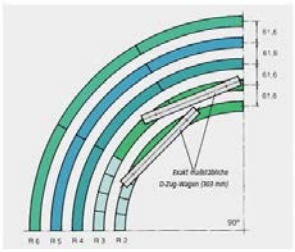
Los tramos de vía pertenecen al modelo geoLine de Roco. Este modelo de vías es perfecto para los aficionados al mundo del modelismo ferroviario. Entre sus cualidades destaca que ya viene con un lecho de vía estable acoplado, y, de esta forma, es posible ensayar el modelo ferroviario rápidamente sin necesidad de complicados retoques. Además, Roco ha integrado algunos nervios transversales en la parte inferior del balasto de vía, a distancias irregulares, cuya misión es amortiguar en la medida de lo posible el ruido producido por el paso del tren.

La unión de las vías entre sí se realiza a través de los clásicos elementos de unión de vías que se alojan adicionalmente en la cubierta protectora. Las juntas de vía ladeadas forman ya parte del pasado y los perfiles de vía forman ahora juntas de calidad impecable.

La vía recta estándar tiene un largo de 200 mm y éste es también el largo de todas las formas de cambios de vía. La vía levemente más corta con su 185 mm sirve como vía compensadora en los tramos de cambio de vía. La distancia de vía es de 76,5 mm, asegurando que los vagones no se pueden cruzar incluso en caso de radios muy reducidos, sin correr riesgo de chocar entre sí. Junto a las vías dobladas en tres radios diferentes se dispone también de las formas básicas de cambios de vía: cambios de vía normales, cambio de vía de doble cruce, cambio de vía de tres itinerarios y una travesía simple, todos con un ángulo de bifurcación de 22,5º. Esto posibilita la construcción de tramos de cambios de vía bastante entrelazados incluso en espacios reducidos. Los cambios de vía en arco se adaptan a los radios 3 y 4, garantizando así que también las locomotoras grandes a vapor puedan circular sobre estos elementos de vía.

En las siguientes figuras se muestran los radios de giro posibles:

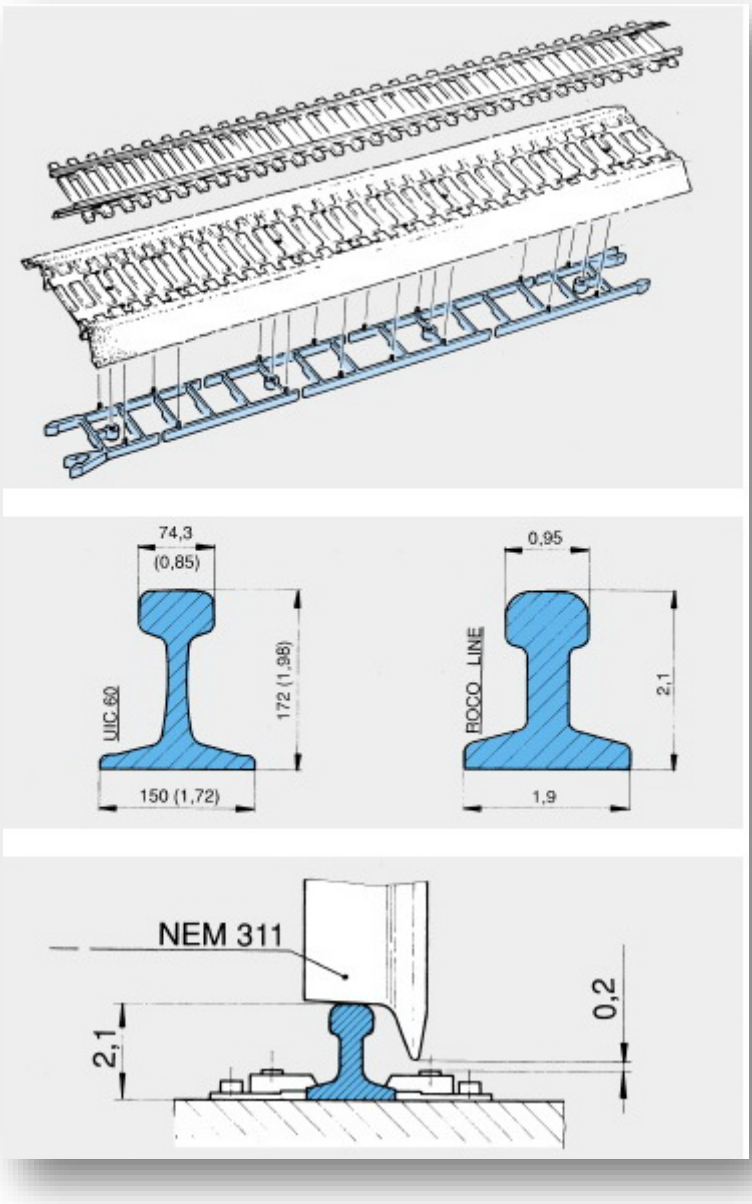


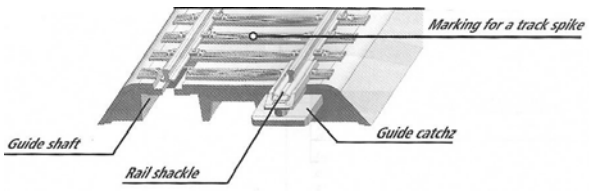


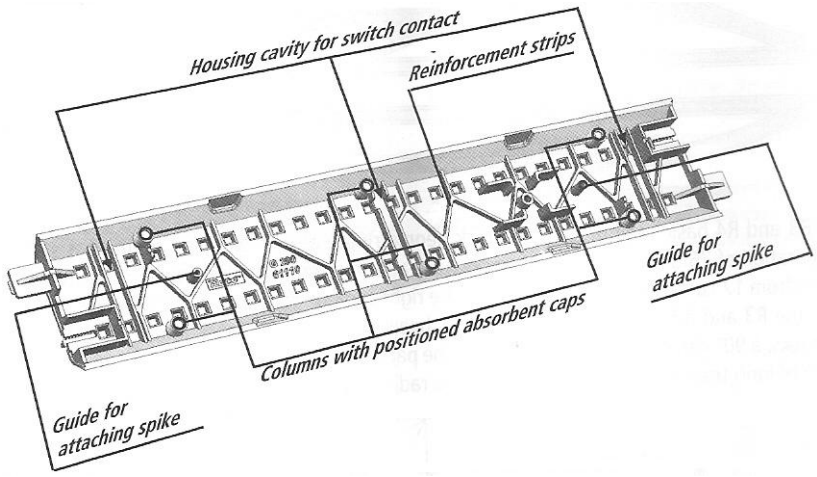
Este modelo de vía es perfecta para todos los motores de cambios de vía, ya que las vías pueden equiparse con un accionamiento eléctrico o con el clásico mecanismo de mando. Los cambios de vía cuentan con

agujas de vía que se han fresado en los perfiles, con lo que se garantiza una tracción segura a largo plazo. La articulación de las agujas de vía se realiza mediante articulaciones estables con un resultado preciso.

El perfil de vía geoLine tiene una altura de sólo 2,1 mm y genera por lo tanto una impresión visual muy fiel a la original. El croquis de la vía es el siguiente:

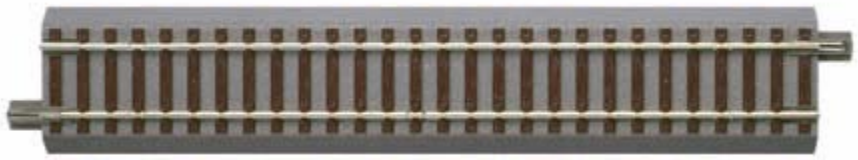






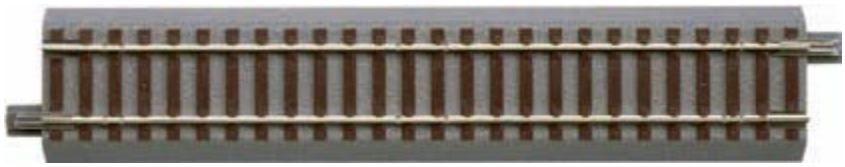
A continuación, se detalla el material de vía empleado en la realización del trayecto:

- Ref. 61110 (rectas G200) → 17 unidades utilizadas.



Largo 200 mm (largo estándar)

- Ref. 61111 (rectas G185) → 2 unidades utilizadas



Largo 185 mm. Vía de compensación para tipos de cambios de vía 22,5º

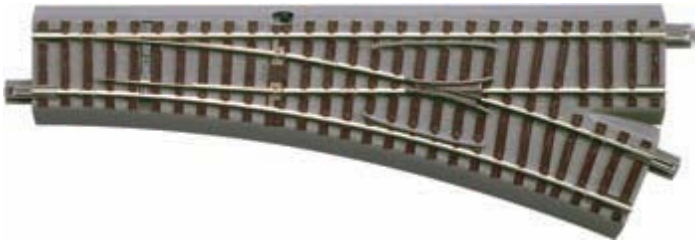
- Ref. 61123 (Arco R3) →16 unidades utilizadas.

434,5 mm/30º

- Ref. 61128 (Arco invertido 2GB)→ 2 unidades utilizadas.

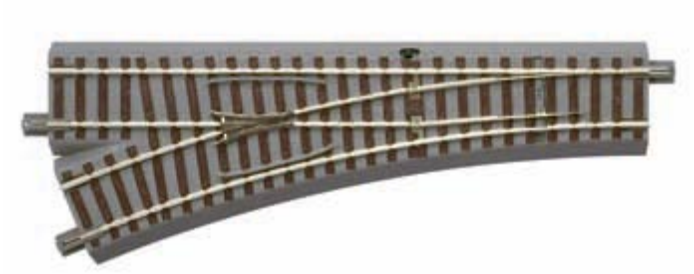
Vía de compensación para cambio de vía de 22,5º. Radio 502,7 mm

- Ref. 61141 (cambio de vía derecha) → 2 unidades utilizadas.



Largo 200 mm, radio de bifurcación 502, 7 mm, ángulo de bifurcación 22,5º

- Ref. 61140 (cambio de vía izquierda) → 2 unidades utilizadas.



Largo 200 mm, radio de bifurcación 502, 7 mm, ángulo de bifurcación 22,5º

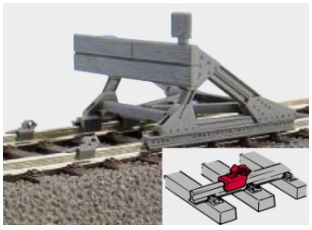
- Ref. 61155 (Cambio de vía de desvío derecha ¾) → 2 unidades utilizadas.

Radio de vía directa y de vía de bifurcación 434,5 mm/30º

- Ref. 61112 (Rectas G76,5) → 3 unidades utilizadas.

Largo 76,5 mm

- Ref. 42608 (Tope de fin de vía) → 1 unidad utilizada.



- Ref. 42267 (Tope de fin de vía) → 1 unidad utilizada.

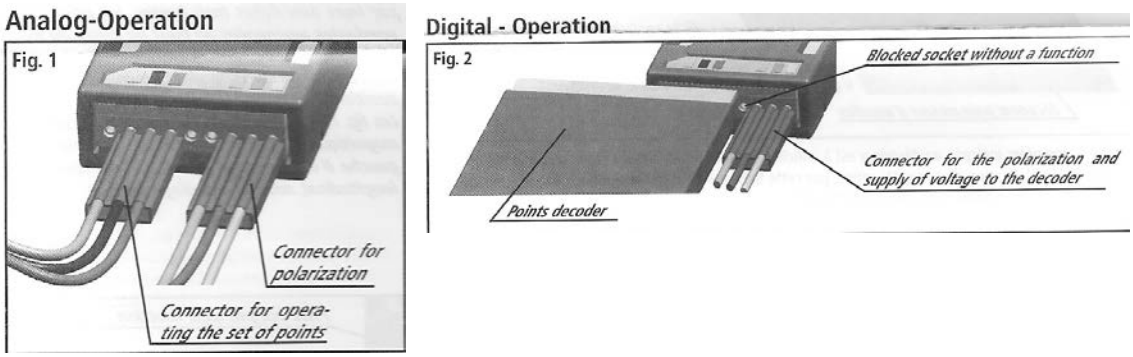
- Ref. 42651 (Fin de vía) → 2 unidades utilizadas.

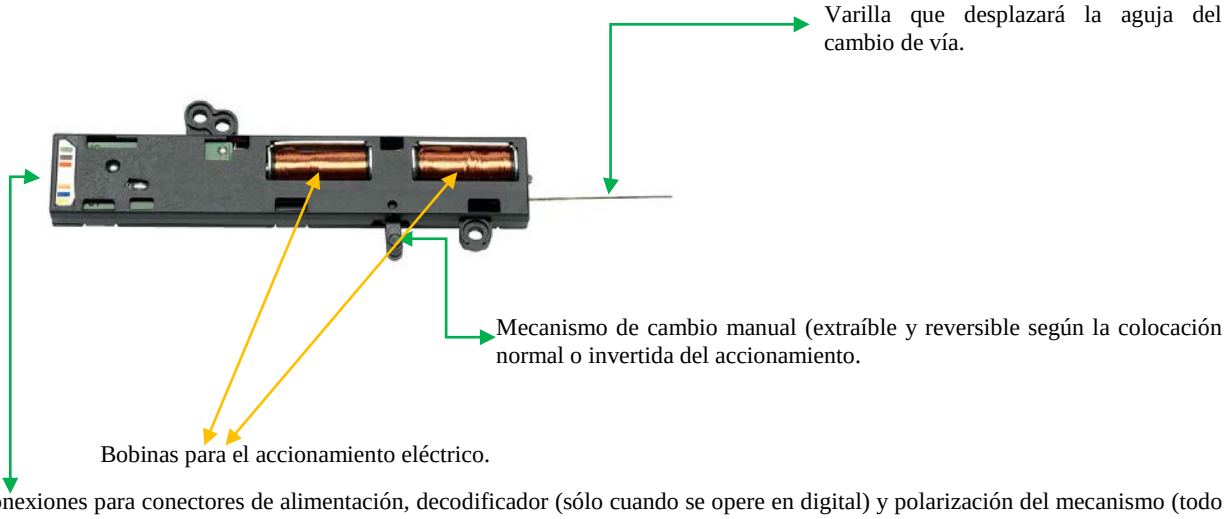
A continuación, se detallan también los dispositivos de los kits de Roco que se han utilizado.

- Ref. 61195 (Accionamiento de vía universal) → 6 unidades utilizadas.

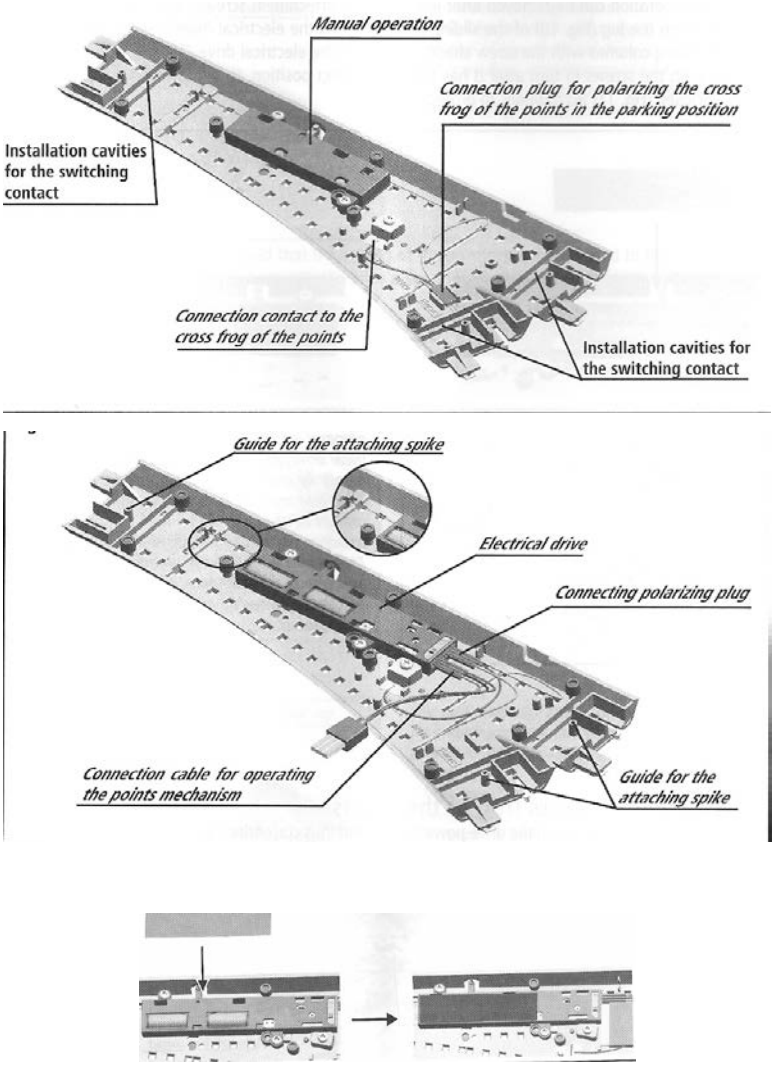
El accionamiento eléctrico 61195 se utiliza en los cambios de aguja de vía en el trayecto. Puede operar tanto en un sistema analógico como en un sistema digital. El accionamiento se sitúa en la parte interior del desvío y se fija con tornillos al mismo. La palanca de accionamiento manual (ver el siguiente gráfico) se debe colocar acorde a la forma de instalación del accionamiento, llevando un capuchón de plástico extraíble que se puede colocar en un lado o en el otro de la manilla que lo lleva. Este accionamiento eléctrico lleva un sistema de doble bobina que posibilita el cambio de posición de la varilla que se ve en la parte frontal del accionamiento (ver el siguiente gráfico). Esta varilla se introduce por un canal guiado en la parte interior del desvío y provoca el movimiento mecánico de la aguja del mismo. Hay varias posibilidades de conexión de este dispositivo, atendiendo como se ha comentado anteriormente, a si el dispositivo opera en modo analógico o en modo digital. Siempre, de una forma o de otra, habrá que tener en cuenta que el accionamiento tiene una etiqueta con colores marcados, indicando la posición en la que se deben colocar los cables eléctricos de conexión a la vía, al decodificador asignado al desvío (si se opera en modo digital) y polarización del mecanismo. A continuación, también se muestra la conexión del accionamiento, tanto para modo analógico como para modo digital. Sin embargo, en este

proyecto no se utiliza ninguno de estos dos modos de conexión, ya que los cambios de vía son controlados externamente al sistema de control digital de Roco, como se verá en apartados posteriores de esta documentación.

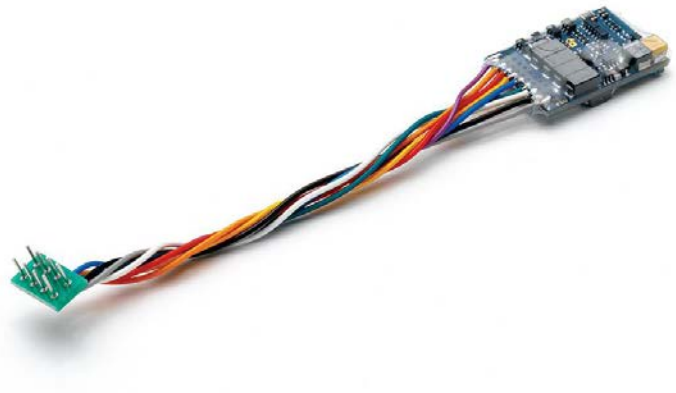




En las siguientes figuras se puede observar el cambio de vía por la parte interior. En principio, los kits de inicio ferroviarios de Roco van equipados con un mecanismo manual de cambio de aguja en la vía. Es necesario quitar este mecanismo manual para instalar el accionamiento eléctrico 61195. Los cambios de vía ya disponen de una cavidad adecuada para instalar el accionamiento, anclándolo a su base con unos tornillos. Los accionamientos incluyen una pegatina con la que se pueden tapar las bobinas para su protección.



- Ref. 10746 (Decodificador digital DCC con regulador de carga para locomotora) → 2 unidades utilizadas.



Las características más importantes de este decodificador son:

- Detección automática de los pasos de velocidad (14/28/128).

- Protección contra sobrecarga en todas las salidas.
- Detección automática de modo digital-analógico.
- Algunas funciones especiales que pueden ser activadas:
    - F1: Función especial (= cable verde, por ejemplo, generador de humo).
    - F3: Cambios de vía a media velocidad.

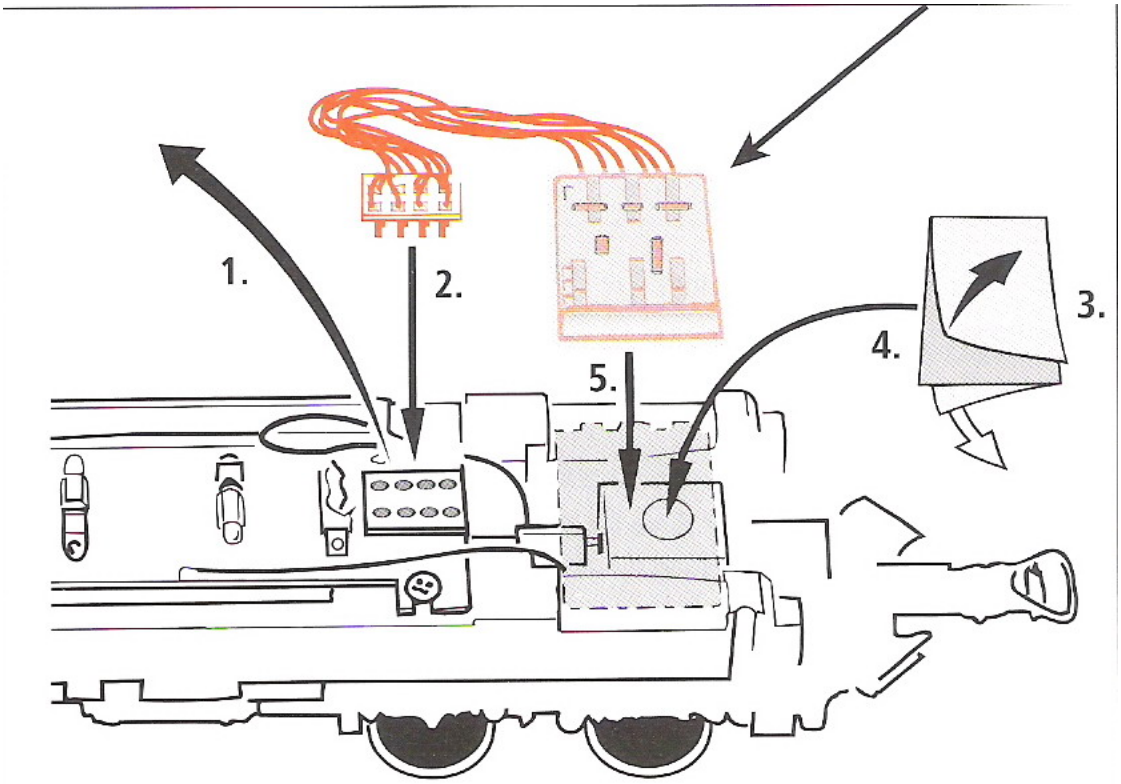
Los modos de operación en los que puede funcionar:

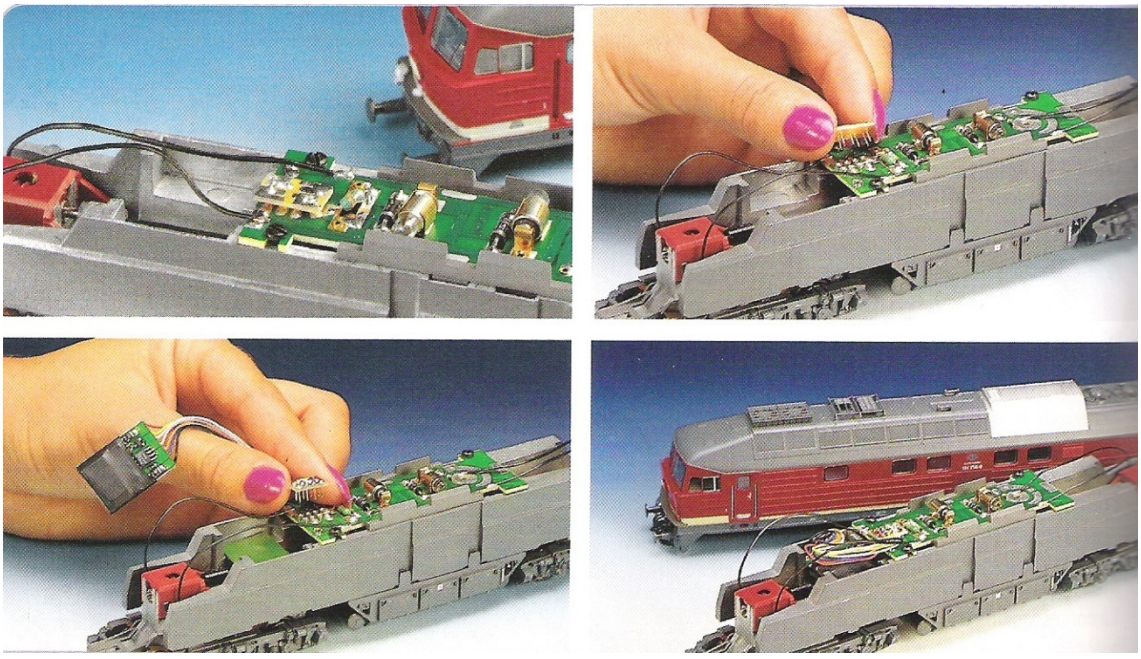
- Operación digital multi-tren en sistemas compatibles NMRA como.
    - Lokmaus 1.
    - Lokmaus 2/PowerMouse, Lokmaus R3.
    - MultiMAUS.
    - ROCOMOTION.
- Operación en locomotoras H0 DC que posean una interface para decodificador acorde a las normas NMRA S 9.1/9.2 y NEM 650/652.

Como norma de seguridad para instalar el decodificador en la locomotora es preciso que la misma se encuentre fuera de la vía en el momento de la instalación. Se pueden seguir las siguientes instrucciones para su instalación:

- Quitar la carcasa de la locomotora y retirar el conector “dummy” o, si ya hay un decodificador instalado, retirarlo igualmente.
- Insertar el conector del decodificador en el interface de forma que el cable rojo/naranja del conector se encuentre en el lado marcado con + o * (de acuerdo a las normas de polaridad NMRA/NEM).
- Posicionar el decodificador en una zona apropiada en la locomotora, teniendo especial cuidado de que ninguna zona del decodificador toque partes metálicas de la locomotora. Si es necesario, habrá que aislar la zona donde se vaya a instalar el decodificador para asegurar su funcionamiento. También hay que poner especial cuidado en que el decodificador pueda ventilarse correctamente y no se sobrecaliente.

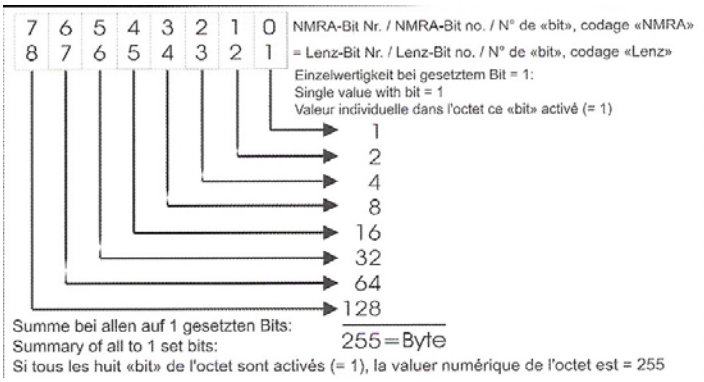
A continuación, se muestra una imagen explicativa de la inserción del decodificador en la locomotora:



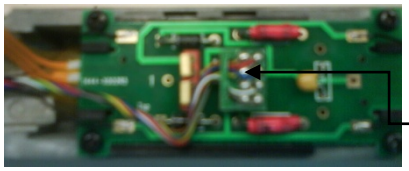


La dirección asignada de fábrica para el decodificador es la dirección 03. Para programar una dirección nueva se recomienda revisar el manual del MultiMAUS (consultar en sección de documentación de anexos el documento pdf “MultiMAUS”).

Para comprobar el funcionamiento correcto del decodificador simplemente hay que fijarse en las luces de la locomotora. Si las luces no se iluminan es un claro síntoma de que el decodificador se ha insertado de forma incorrecta (normalmente girándolo 180º el decodificador funcionará correctamente). Si el problema se produce sólo con la luz del frente de la locomotora, significa que el decodificador se está usando en el modo de 28 pasos de velocidad. Para solucionar esto se propone modificar la programación del CV29.



Esta es una imagen en la que se puede ver el decodificador de una de las locomotoras utilizadas en este proyecto:



El decodificador se encuentra insertado en su zócalo correspondiente (según norma NEM 652) en el interior de la locomotora.

Decodificador digital

A continuación, se muestra una tabla con las CVs más importantes en la programación de este decodificador:

CV	USO	CONFIG. POR DEFECTO	RANGO ADMITIDO
01	Dirección de locomotora	03	01-99
02	Velocidad mínima	03	01-63
03	Tiempo de aceleración	08	00-63
04	Tiempo de frenado	06	00-63
05	Velocidad máxima	64	00-63
07	Versión del decodificador		¡SÓLO LECTURA!
08	Identificación del fabricante o Volver a configuración por defecto	151	Sólo lectura/Reset para valores de fábrica =08
29	Pasos de velocidad, modo analógico, dirección de conducción:NMRA	06	Bit 0→Dirección de conducción: 0 =normal; 1=invertida; Bit 1→Pasos de velocidad: 0=14;1=28/128 Bit 2→Modo analógico: 0=Off;1=On
49	Regulación de carga	01	Bit 0: 0=Off;1=On
63	Regulador para la salidas de las funciones	07	00-07

- Ref 10810 (Mando de control digital) → 2 unidades utilizadas.

MultiMAUS es el mando de control digital de la marca Roco con la que se controlan todas las funciones de los trenes y de cualquier accesorio de la maqueta conectado al sistema digital gestionado por la central digital. Algunas de las características más importantes de este mando son las siguientes:

Características generales:

- Ergonomía para su empleo en la mano.
- Gran pantalla de CL (cristal líquido) retroiluminada.
- Fácil control de la velocidad y de la dirección de las locomotoras con el regulador giratorio.
- Posición neutral (de parada) marcada en el regulador giratorio.
- Multilenguaje.
- Compatibilidad con otros mandos NMRA-DCC.
- El sistema digital ROCO permite conectar hasta un máximo de 31 componentes tales como Lokmaus, controlador de rutas, etc.
- Actualizaciones desde la interfaz RS485 y ROCOMOTION (X-BUS).

Características en cuanto a capacidad:

- Hasta 9.999 direcciones de locomotoras, alternativamente activadas o memorizadas.
- Nombres alfanuméricos para las locomotoras, pueden memorizarse hasta una máximo de 64 locomotoras.
- Control de la velocidad de las locomotoras en 14, 28 y 128 pasos, ajustable individualmente para cada una de ellas.
- Control de las luces direccionales y de 20 funciones adicionales más, en cada locomotora.
- Control de hasta 1.024 direcciones de desvíos.
- Modificación de las variables de configuración (DCC-CVs).

Características en cuanto a seguridad:

- Parada de emergencia de toda la instalación.
- Parada de emergencia solamente de la locomotora activa.
- Bloqueo del acceso a determinadas operaciones, para evitar utilizaciones no deseadas por parte de niños, por ejemplo, para programar.

En las siguientes imágenes se puede observar este mando digital de control, incluyendo las funciones más importantes de los botones del mismo:

Con este mando de control digital se pueden controlar tanto las locomotoras (en todas sus funciones) como los desvíos y señalizaciones. La multimaus envía sus órdenes en paquetes digitales a las vías. Locomotoras y desvíos "escuchan" por así decirlo mediante sus propios decodificadores. Cada componente reaccionará sólo cuando la señal de control le afecte, no interfiriendo en el funcionamiento de otros dispositivos que formen parte del sistema digital.

El cable plano de datos suministrado con cada mando de control es suficientemente largo como para tener una cierta libertad de movimiento y poder seguir la marcha de los trenes, pudiéndose prolongar más usando adaptadores y más cable si es necesario.

La pantalla posee brillo ajustable, iluminándose en color verde.

Con un sólo mando se pueden controlar simultáneamente hasta cuatro locomotoras en un solo circuito de corriente. No obstante, si se necesitase más potencia se puede incluir en la instalación un "booster" o amplificador. Además, se pueden activar hasta 9.999 direcciones de locomotoras por ingreso directo, 64 de ellas pueden ser memorizadas en una base de datos y ser provistas de nombres (por ejemplo, 103RB).

Si las locomotoras se equipan también para ello, se pueden activar hasta 21 funciones por cada una de ellas (por ejemplo, iluminación frontal, sonido, salida de humo, etc).

En el modo de funcionamiento del mando llamado "desvíos" se pueden controlar hasta 1.024 accesorios digitales (los principales son los desvíos). Para ello, estos desvíos deben poseer un accionamiento eléctrico y estar conectados a un decodificador preparado exclusivamente para estos dispositivos (estos decodificadores de accesorios se programan por impulsos eléctricos a través de la vía).

En posteriores secciones de este documento se explican las modificaciones que se han llevado a cabo en los mandos de control para su utilización en este proyecto. Comentar también que se ha decidido incluir en la sección de la documentación de anexos el manual de funcionamiento íntegro de este dispositivo, ya que es un documento extenso y esta sección sólo pretende ser una introducción a la tecnología digital utilizada en el proyecto.

- Ref. 10725 (Transformador universal) → 2 unidades utilizadas.

Características técnicas principales:

- Salida → 16 voltios, 50VA.
- Utilización → Para la alimentación eléctrica de los siguientes componentes: Amplificador 10764, booster 10765, interfaz 10785, regulador analógico 10727.
- Protecciones → Fusible de sobrecarga, a prueba de cortocircuitos. Protección contra retorno, homologado por LGA.

Este transformador va, por un lado, directamente conectado a la tensión de alimentación de red (~230V), y por otro lado, al amplificador 10764 (suministrándole de esta forma la alimentación eléctrica necesaria).

El motivo por el cual se han utilizado dos unidades se especifica en una sección posterior de la documentación, en la que se explican las modificaciones realizadas en los dispositivos para una utilización externa de los mismos.

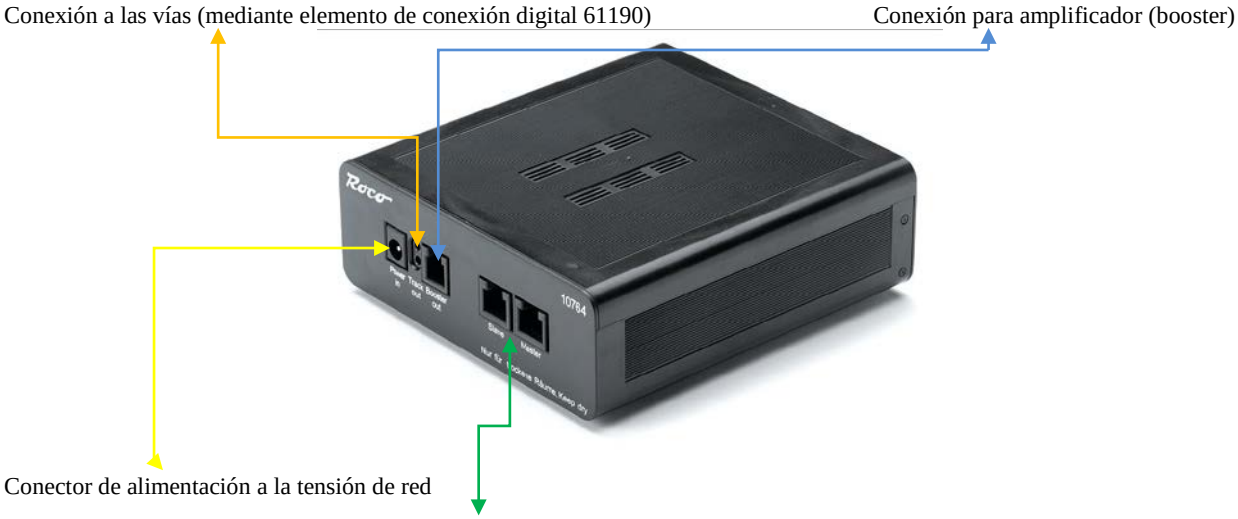
Se puede ver el transformador en la siguiente imagen:

- Ref. 10764 (Amplificador digital) → 1 unidad utilizada.

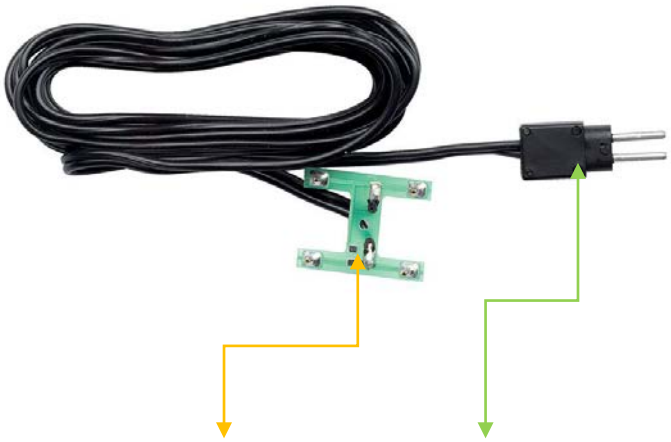
Este amplificador ofrece dos conexiones (Master y Slave). En cada una de ellas puede conectarse un mando digital (o varios si se utilizan derivadores o adaptadores con ese propósito). Evidentemente debe existir siempre un mando actuando de Master en la instalación, sin embargo, los dispositivos Slave son opcionales. A través de las conexiones con estos mandos de control, el amplificador prepara los datos suministrados por los mismos para enviarlos a la vía, previa conexión del amplificador a la tensión de alimentación de la red. Este amplificador realiza básicamente estas dos funciones: por una parte, recoge los datos suministrados por los mandos de control y, por otra, suministra la tensión correspondiente a la vía para el funcionamiento de los dispositivos, ya sean las locomotoras o cualquier tipo de accesorio digital conectado al sistema. Para ello, este amplificador digital (o también llamado central digital), prepara estos datos en forma de tensión eléctrica. Esta transformación realizada por el amplificador es el resultado de la modificación en modulación (para introducir los datos proporcionados por los mandos de control) de la tensión eléctrica suministrada por el transformador visto anteriormente, acatando una normativa específica creada por la NMRA para el protocolo de transmisión. Este protocolo de transmisión de datos se denomina DCC (Digital Command Control) y ha sido estudiado en profundidad en el capítulo 2 sección 2.2 de esta documentación.

Cuando el amplificador o central digital ha transformado la señal ajustándola al protocolo DCC la transmite a la vía. Para ello, el amplificador digital dispone de un conector en el que se inserta La diferencia de este sistema digital con respecto al sistema analógico tradicional radica fundamentalmente en que en este último el objetivo es que a través de las variaciones de tensión en las vías el motor de una locomotora variase su velocidad (relación tensión-velocidad), mientras que en el sistema digital la señal de control es enviada a través de la vía, pero solo es reconocida por el dispositivo concreto al que se refiere la señal en cada momento. Gracias a los decodificadores, estudiados anteriormente, es posible que la central digital haga que un dispositivo concreto reaccione y otros no, dependiendo del patrón de datos enviado en cada trama de la señal DCC. Por lo tanto, la secuencia de información es la siguiente: Los mandos de control generan y envían los datos de control al amplificador digital. Este amplificador, alimentado al mismo tiempo con la tensión de red proporcionada por el transformador, es capaz de mezclar la tensión alterna “normal” con los datos del mando, creando una señal de tensión cuadrada y adaptada a normativa DCC. El amplificador envía constantemente este tipo de señal modificada a las vías y, los decodificadores de cualquier tipo de dispositivo conectado al sistema se queda a la escucha. Todos ellos evalúan la señal que envía la central digital a las vías, actuando en función de las órdenes recibidas sólo cuando reconocen que ese paquete de datos se refiere a él, y no haciendo nada cuando no sea así, siguiendo constantemente a la escucha.

En la siguiente imagen se muestra el amplificador digital:



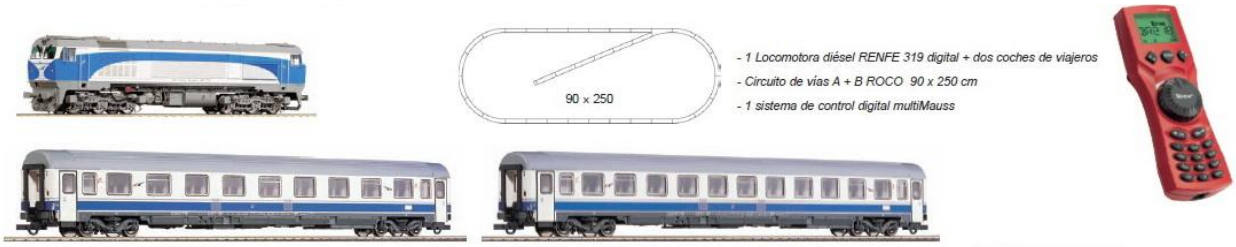
- Ref. 61190 (Elemento de conexión a la vía) → 1 unidad utilizada.



Conexión a la vía Conexión al amplificador digital

A continuación, se muestra un breve resumen de los kits de inicio utilizados en este proyecto:

- Ref. Roco 41257: Digital Starset Renfe Pasajeros (1 unidad utilizada).



La locomotora es un modelo a escala H0 del modelo real Renfe 319-316

- Ref. Roco 41256: Digital Starset Renfe Mercancías (1 unidad utilizada).

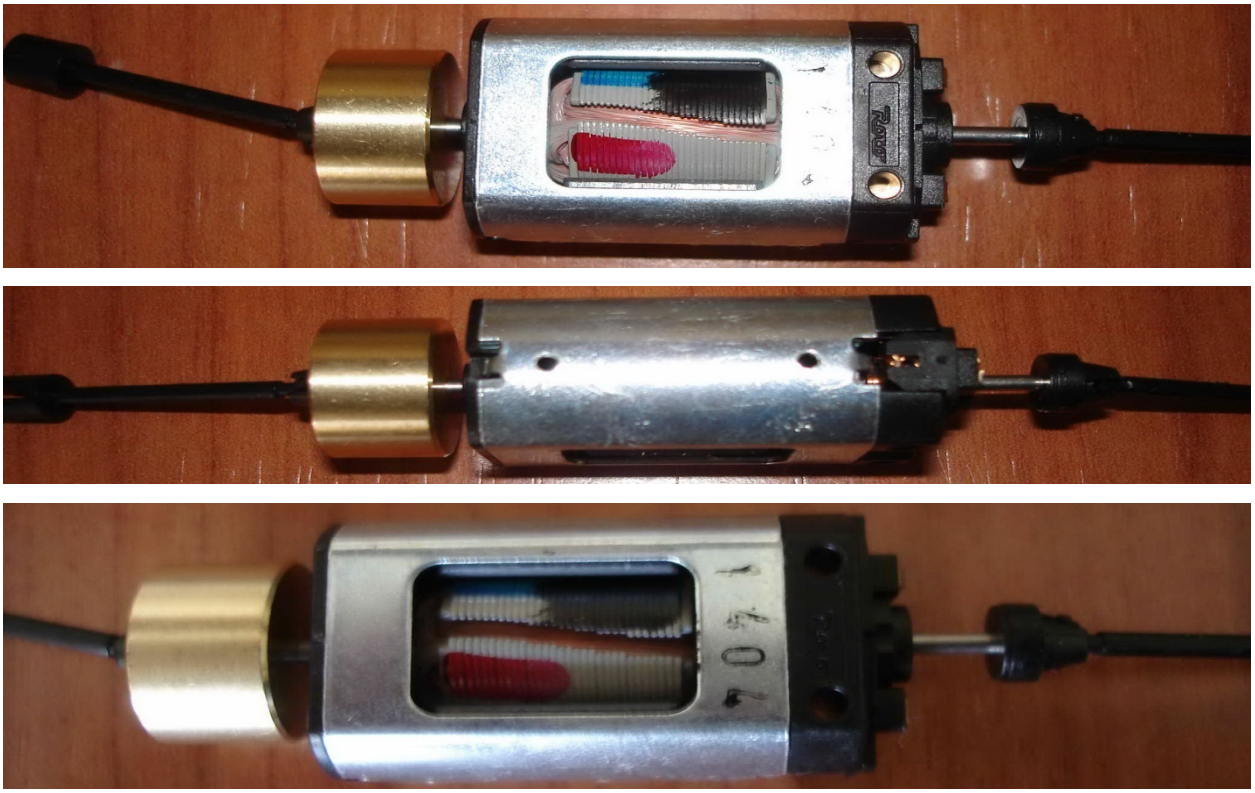
La locomotora es un modelo a escala H0 del modelo real Renfe 319-402

El motivo de adquirir dos kits de inicio es principalmente debido a que uno de los objetivos del proyecto es controlar varios trenes (en este caso dos, uno incluido en cada kit) sobre la maqueta. Por otra parte, el circuito a construir debe tener ciertos requisitos, como la existencia de dos estaciones, con sus correspondientes entradas al interior de la estación, la existencia de varias vías muertas (en este caso dos), etc, y para ello es necesario la utilización de componentes de dos kits de inicio.

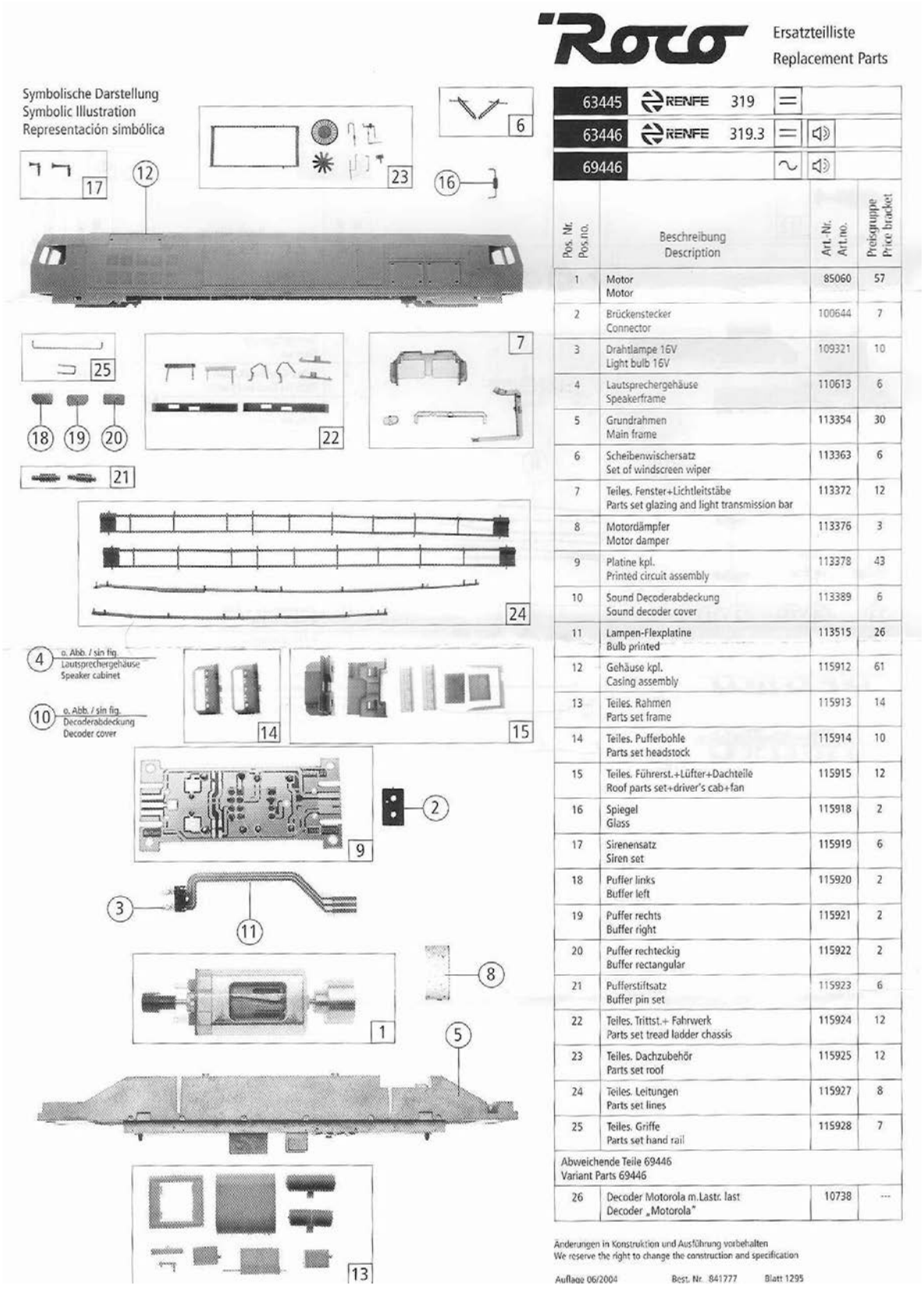
A pesar de que se podrían haber adquirido los componentes de forma independiente, se ha tomado la decisión de adquirir dos kits de inicio porque de esta forma resulta bastante más económico. Curiosamente, en el mundo del modelismo ferroviario, es más económico adquirir (siempre teniendo en cuenta que se trate de aficionados que se inician al mundo del modelismo) kits de inicio que comprar todos los componentes de forma independiente y por separado. Por lo tanto, y a pesar de que no se han utilizado, por ejemplo, los vagones de tren que, integrados en los kits o la totalidad de las vías, ha resultado más económico adquirir el material de esta forma y no comprando el material justo y necesario por separado. Esto además proporciona la ventaja de disponer de material de reserva (vías y componentes de control digital y alimentación) por si el primer material instalado da algún problema.

En cuanto a las locomotoras, destacar que son modelos muy parecidos, puesto que las dos pertenecen al modelo real de Renfe 319. Sin embargo, una de ellas se trata del modelo 319 serie 316, y la otra se trata del modelo 319 serie 402 (ambas bajo la ref.63445 de Roco). En el kit 41257 utilizado, la locomotora 319-316 es utilizada para el transporte de pasajeros, mientras que en el kit 41256 la locomotora 319-402 es utilizada para el transporte de mercancías. Externamente, y en cuanto a los modelos a escala del kit de inicio de modelismo ferroviario se refiere (en las locomotoras reales de Renfe hay varías diferencias técnicas importantes, como la potencia del motor, etc), son prácticamente idénticas, exceptuando algún que otro detalle en su decoración y color. Destacar que el exterior decorado de las locomotoras es de plástico, mientras que en el interior tienen un armazón metálico en el cuál se distribuyen todos los componentes necesarios tales como motor, decodificador, luces, cables de conexión, engranajes de las ruedas, etc. Anteriormente, y al igual que anteriormente se ha mostrado una imagen del decodificador integrado dentro de una de las locomotoras, se muestra una imagen de uno de los motores de la

locomotora (en concreto de la locomotora 319-316 de pasajeros), aunque, como se ha comentado anteriormente, el otro motor correspondiente a la locomotora 319-402 es exactamente el mismo modelo:



El interior de las locomotoras de los kits de inicio Roco (de estos dos kits en concreto) son exactamente iguales, por lo que a continuación se muestran unas imágenes de los componentes que integran las locomotoras (tener en cuenta que los componentes que la referencia utilizada es 63445):

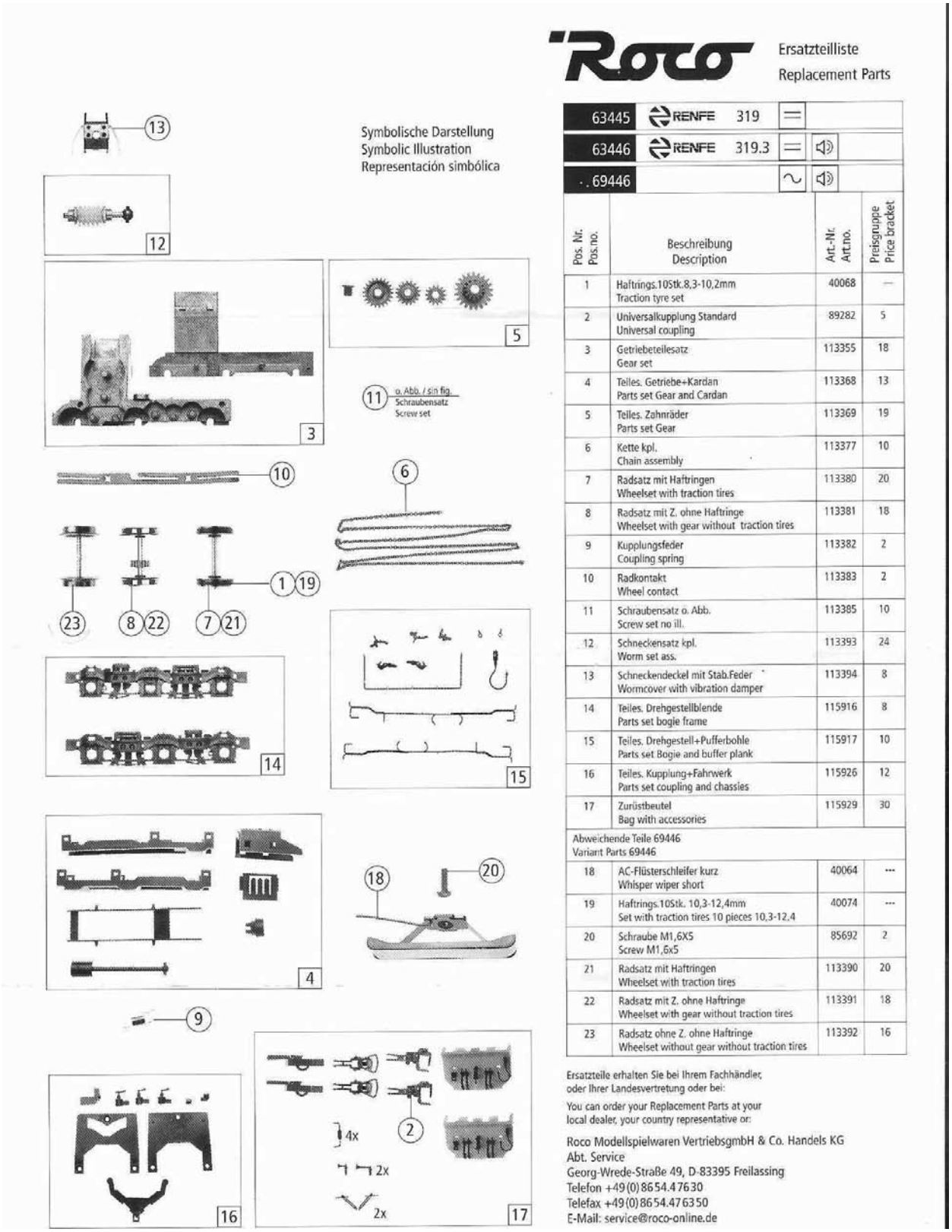


63445	RENFE	319	=	
63446	RENFE	319.3	=	🔊
69446			~	🔊

Pos. Nr. Pos.no.	Beschreibung Description	Art.-Nr. Art.no.	Preisgruppe Price bracket
1	Motor Motor	85060	57
2	Brückenstecker Connector	100644	7
3	Drahtlampe 16V Light bulb 16V	109321	10
4	Lautsprechergehäuse Speakerframe	110613	6
5	Grundrahmen Main frame	113354	30
6	Scheibenwischersatz Set of windscreen wiper	113363	6
7	Teiles. Fenster+Lichtleitstäbe Parts set glazing and light transmission bar	113372	12
8	Motordämpfer Motor damper	113376	3
9	Platine kpl. Printed circuit assembly	113378	43
10	Sound Decoderabdeckung Sound decoder cover	113389	6
11	Lampen-Flexplatine Bulb printed	113515	26
12	Gehäuse kpl. Casing assembly	115912	61
13	Teiles. Rahmen Parts set frame	115913	14
14	Teiles. Pufferbohle Parts set headstock	115914	10
15	Teiles. Führerst.+Lüfter+Dachteile Roof parts set+driver's cab+fan	115915	12
16	Spiegel Glass	115918	2
17	Sirenensatz Siren set	115919	6
18	Puffer links Buffer left	115920	2
19	Puffer rechts Buffer right	115921	2
20	Puffer rechteckig Buffer rectangular	115922	2
21	Pufferstiftsatz Buffer pin set	115923	6
22	Teiles. Trittst.+ Fahrwerk Parts set tread ladder chassis	115924	12
23	Teiles. Dachzubehör Parts set roof	115925	12
24	Teiles. Leitungen Parts set lines	115927	8
25	Teiles. Griffe Parts set hand rail	115928	7
Abweichende Teile 69446 Variant Parts 69446			
26	Decoder Motorola m.Lastr. last Decoder „Motorola"	10738	---

Änderungen in Konstruktion und Ausführung vorbehalten
We reserve the right to change the construction and specification

Auflage 06/2004 Best. Nr. 841777 Blatt 1295



Roco

Ersatzteilliste
Replacement Parts

63445	RENFE 319	=	
63446	RENFE 319.3	=	🔊
. 69446		~	🔊

Pos. Nr. Pos.no.	Beschreibung Description	Art.-Nr. Art.no.	Preisgruppe Price bracket
1	Haftrings.10Stk.8,3-10,2mm Traction tyre set	40068	—
2	Universalkupplung Standard Universal coupling	89282	5
3	Getriebeteilesatz Gear set	113355	18
4	Teiles. Getriebe+Kardan Parts set Gear and Cardan	113368	13
5	Teiles. Zahnräder Parts set Gear	113369	19
6	Kette kpl. Chain assembly	113377	10
7	Radsatz mit Haftringen Wheelset with traction tires	113380	20
8	Radsatz mit Z. ohne Haftringe Wheelset with gear without traction tires	113381	18
9	Kupplungsfeder Coupling spring	113382	2
10	Radkontakt Wheel contact	113383	2
11	Schraubensatz o. Abb. Screw set no ill.	113385	10
12	Schneckensatz kpl. Worm set ass.	113393	24
13	Schneckendeckel mit Stab.Feder Wormcover with vibration damper	113394	8
14	Teiles. Drehgestellblende Parts set bogie frame	115916	8
15	Teiles. Drehgestell+Pufferbohle Parts set Bogie and buffer plank	115917	10
16	Teiles. Kupplung+Fahrwerk Parts set coupling and chassies	115926	12
17	Zurüstbeutel Bag with accessories	115929	30
Abweichende Teile 69446 Variant Parts 69446			
18	AC-Flüsterschleifer kurz Whisper wiper short	40064	---
19	Haftrings.10Stk. 10,3-12,4mm Set with traction tires 10 pieces 10,3-12,4	40074	---
20	Schraube M1,6X5 Screw M1,6x5	85692	2
21	Radsatz mit Haftringen Wheelset with traction tires	113390	20
22	Radsatz mit Z. ohne Haftringe Wheelset with gear without traction tires	113391	18
23	Radsatz ohne Z. ohne Haftringe Wheelset without gear without traction tires	113392	16

Ersatzteile erhalten Sie bei Ihrem Fachhändler, oder Ihrer Landesvertretung oder bei:

You can order your Replacement Parts at your local dealer, your country representative or:

Roco Modellspielwaren VertriebsgmbH & Co. Handels KG
Abt. Service
Georg-Wrede-Straße 49, D-83395 Freilassing
Telefon +49 (0) 8654.47630
Telefax +49 (0) 8654.476350
E-Mail: service@roco-online.de

9. Descripción del material utilizado en el sistema de control.

El sistema de control para la maqueta ferroviaria está integrado por varios elementos utilizados normalmente en la automatización de procesos, como son controladores para automatización (denominados "Programable Logic Controller" o, de forma abreviada, autómata o PLC), diversos dispositivos electrónicos como relés de control, sensores, pulsadores y señalización luminosa, etc. La estructura principal del sistema de control está basada en tres PLCs cuyas entradas y salidas digitales y analógicas interaccionan directamente con el hardware de control y los elementos de la maqueta ferroviaria y un ordenador desde el cual se puede controlar y monitorizar todo el sistema, tanto a nivel de hardware de control como los distintos componentes de la maqueta ferroviaria que sean automatizados.

El hardware de control debe ser capaz de manejar lo que se pretende automatizar, disponiendo de todas las entradas y salidas de control necesarias para ello. Además, se debe considerar la necesidad de determinados requisitos de velocidad en la adquisición de datos y ejecución de las órdenes de control sobre el sistema ferroviario (sistema de control crítico en tiempo y condiciones de seguridad). Hay que destacar que el sistema ferroviario necesita ser controlado por un sistema automatizado que sea capaz de ejecutar las acciones de control en, prácticamente, "tiempo real", además de asegurar un altísimo porcentaje de seguridad y fiabilidad en las transacciones de datos que se lleven a cabo con el sistema. Al ser el transporte ferroviario un transporte público y bastante utilizado en la actualidad, es indispensable mantener al sistema en un estado estable y con una alta condición de seguridad global, que evitará cualquier tipo de error crítico que acarree como consecuencia una catástrofe. El hardware para realizar el control automatizado que se ha seleccionado cumple todas las propiedades necesarias para este proyecto en particular. Se trata de varios autómatas de la marca comercial Siemens con la siguiente distribución:

- **Siemens S7 314 2-DP**: Este autómata realizará las labores de control central de todo el sistema. Es decir, cuando en la documentación se hace referencia al autómata CTC, es a éste autómata S7-314 al que nos estamos refiriendo. Tiene la función de ejercer el control sobre el resto de autómatas de la instalación automatizada y de mandar o recibir las órdenes de control del software Scada diseñado para el Jefe de Control. Este autómata cumple unos tiempos de funcionamiento que cumplen sobradamente los requisitos de velocidad y fiabilidad de los datos que tiene que manejar.
- **Siemens S7 226 (Estaciones 1 y 2):** Son dos autómatas (uno para cada estación del trayecto ferroviario), que controlaran a cada estación y su zona de influencia (respectivamente). Estos autómatas controlan directamente (físicamente cableados) todos los componentes de detección, seguridad y mecanismos controlables del sistema ferroviario. A su vez, estos autómatas son controlados directamente por el autómata que ejerce de maestro en el CTC (directamente o a través del software Scada). El autómata del CTC da órdenes directas a estos dos autómatas a través de una comunicación PROFIBUS DP, y estos autómatas deben cumplirlas, además de enviar los datos de estado de todos los dispositivos cuando el CTC se lo ordena. Sin embargo, y debido a este modo de funcionamiento establecido (no es el único que se podría haber adoptado, pero si el más eficiente desde un punto de vista de control jerárquico), estos autómatas no dan ordenes al CTC, ya que actúan de "esclavos" del mismo. Se limitan a cumplir las órdenes del CTC (mediante su autómata "maestro"), y a suministrar los datos necesarios cuando aquel lo requiere.

También se ha tenido en cuenta el siguiente hardware:

- Ordenador de control: Este ordenador lo maneja el Jefe de Control y el software Scada/HMI que lleva instalado es el interface de comunicación humano/máquina por el cual se puede monitorizar y controlar el sistema de automatización y, en consecuencia, el sistema ferroviario. Este equipo informático debe disponer de unos requisitos mínimos de hardware común, además de disponer de una tarjeta de comunicaciones PROFIBUS DP específica para su comunicación con el autómata "maestro" de la instalación.

- Electrónica para los cambios de aguja en la vía: Se dispone de tecnología de relés para controlar los motores que llevan incorporados los cambios de agujas en las vías. Con estos relés se controla la posición exacta de los cambios de vía y se asegura una rapidez suficiente y segura de cambio de estado. El autómata del CTC, encargado del control de todo el sistema en modo automático, ordena estos cambios de aguja cuando sea necesario, o también, en los modos semi-automático y manual, cuando el Jefe de Control lo ordene. En cualquier caso, la jerarquía de control "maestro-esclavo" siempre se cumple en cualquiera de los tres modos de funcionamiento, ya sea el Jefe de Control quien dé las ordenes de cambio de aguja o el propio autómata del CTC cuando el sistema se encuentre en modo automático.

- Pulsadores, seta de emergencia y selectores de modo para realizar el control del sistema, que son accionados por el Jefe de Control cuando sea necesario.

- Señalización luminosa, indicando estados de funcionamiento del sistema de control, como, por ejemplo, funcionamiento correcto, advertencias, errores o situaciones de emergencia.

- Los sensores, en este proyecto de tipo inductivos y fotoeléctricos, suministran información al autómata del CTC, y por tanto, al Jefe de Control, de la posición de los trenes. Además, esta información será utilizada por el autómata de forma bidireccional, ya que el que un tren pase por un sensor desencadenará las acciones de control previstas en la programación del sistema. Es decir, los sensores suministrarán información y ejercerán de "disparadores" de acciones en consecuencia a la programación de control.

- La tecnología de diseño de los semáforos se basa en electrónica de leds, y se han adaptado al escenario ferroviario diseñado. Disponen de un led de color verde y otro led de color rojo (que son los dos colores más habituales en los semáforos de los trayectos ferroviarios reales, dependiendo, claro está, del tipo de zona y trayecto ferroviario a cubrir). En este proyecto sólo se contempla este tipo de semáforo porque es suficiente para controlar el recorrido diseñado, pero es importante destacar que existen más variedades de semáforos en las normas de control ferroviario real. El material empleado en su construcción se detallará en posteriores secciones de esta documentación.

- Cámara de red PT (Pan/Tilt) de la compañía LevelOne. Dispone de funcionalidad de tracking o seguimiento de objetos en movimiento. Gracias al SDK que incluye, esta cámara puede programarse para que detecte los convoys en movimiento y tome fotografías de los objetivos y zonas que se establezcan (targets).

La lista completa de material relativo a esta parte de automatización y control se detalla más adelante en este documento. En cuanto al posicionamiento de los autómatas y electrónica de control se ha pensado en un soporte de tablero que permite su desplazamiento parcial, donde se distribuyen los autómatas y la electrónica necesaria para realizar el control. El ordenador de control se considera descentralizado y se puede colocar a una distancia considerable gracias a la comunicación PROFIBUS (RS 485 en este caso), que admite, entre otras, el siguiente rango de distancias según la velocidad de transmisión deseada:

- A 12 Mbit/s, distancia máxima de 100 metros (un solo segmento de cable sin repetidor y con una carga máxima de 32 estaciones)

- A 1,5 Mbit/s, distancia máxima de 400 metros (un solo segmento de cable sin repetidor y con una carga máxima de 32 estaciones)

- A 187,5 kBit/s, distancia máxima de 1000 metros (un solo segmento de cable sin repetidor y con una carga máxima de 32 estaciones)
- Gracias al uso de repetidores, la distancia máxima puede extenderse hasta 10 Km de distancia

9.1. CPU Serie S7-300

En el proyecto se ha decidido utilizar 1 Uds. de la CPU de la serie S7-300, concretamente una CPU 314C-2DP, con número de referencia Siemens 6ES7314-6CG03-0AB0. Esta CPU, es una CPU compacta con entradas y salidas digitales y analógicas integradas y un puerto PROFIBUS DP maestro/esclavo integrado. Esta CPU se puede utilizar para tareas con funciones especiales y para conectar periferia distribuida si es necesario, además de disponer de funciones tecnológicas. Para utilizar esta CPU se requiere una micro memory card.

CPU S7 314C-2DP

Elección de la CPU y justificación técnica para el proyecto

Para seguir la estructura "maestro-esclavo" se ha pensado en una CPU de la serie S7-300 controlando a dos CPUs de la serie S7-200, siendo el S7-300 el maestro de la red de autómatas y los dos S7-200 los esclavos del primero, los cuales sólo tienen que obedecer las órdenes de su maestro y suministrar los datos de estado de todos los dispositivos a su cargo cuando el maestro se lo ordene.

La comunicación entre PLC-PLC y PLC-PC va a ser PROFIBUS DP, por lo que se ha decidido adquirir un autómata de la serie 300, puesto que los autómatas de la serie S7-200 no pueden actuar como maestros de la red PROFIBUS DP. Sin embargo, los PLCs de la serie S7-300 pueden actuar tanto de maestros (incluso en dos niveles) como de esclavos en la red. Este comportamiento lo determina el propio fabricante, por lo que la red de control en bus PROFIBUS DP queda compuesta por un PLC de la serie S7-300 con función maestro de la red y dos PLCs de la serie S7-200 con función esclavo de la red.

Con respecto a la comunicación PROFIBUS DP, hay dos formas de conseguir poder comunicar con este protocolo usando una CPU S7-300 para ello:

- Adquiriendo una CP (o módulo de comunicaciones) externo a la CPU y conectándolo como módulo de ampliación, gestionando gran parte de la comunicación la CP y liberando a la CPU de parte de esta tarea para que pueda ejecutar otros cometidos.
- Adquiriendo una CPU que ya lleve integrado el puerto de comunicaciones PROFIBUS DP, dependiendo la gestión de la comunicación de la propia CPU.

Un puerto de comunicaciones PROFIBUS DP integrado en el autómata sería suficiente, puesto que la red de autómatas no es extensa y las tareas a realizar no justificarían las ventajas que aporta un módulo CP específico.

Por otra parte, y comparando económicamente las dos propuestas, resulta más asequible la adquisición de una CPU con puerto de comunicaciones PROFIBUS DP integrado, que adquirir un procesador de comunicaciones CP específico.

En cuanto a E/S digitales se determina que son necesarias más de 8 entradas digitales y unas 16 salidas digitales, por lo tanto, hay que seleccionar una CPU que lleve a bordo como mínimo 16 entradas digitales y 16 salidas digitales para conseguir el mayor ahorro económico posible. Las CPUs compactas de la serie S7 300 suelen llevar este número de entradas y salidas digitales a bordo.

En cuanto a E/S analógicas, no es necesario que el S7-300 las tenga, puesto que, según la distribución automatizada para este proyecto, ésta CPU no debe gestionar ninguna.

También será necesario que la CPU lleve integrado un puerto de comunicaciones MPI, ya que habrá que utilizarlo para cargar o salvaguardar el programa a/de la CPU y realizar otras operaciones entre PC-PLC.

En cuanto a memoria interna que debe integrar la CPU y velocidad de procesamiento no hay restricciones críticas, puesto que cualquiera de las CPUs de la serie S7-300 tiene estas cualidades sobredimensionadas a lo que es necesario en este proyecto.

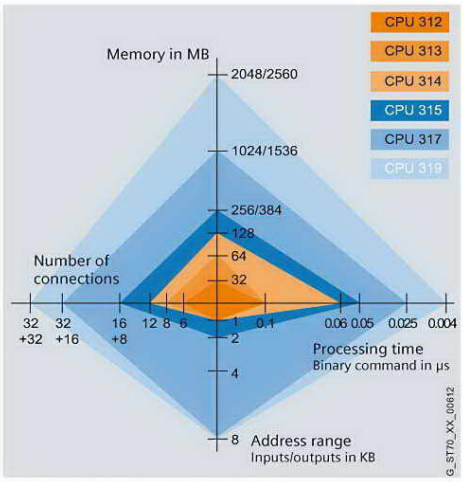
Por lo tanto, las restricciones más importantes y en las que fundamentalmente hay que basar la decisión de compra son, el puerto de comunicaciones PROFIBUS DP integrado y el número de E/S digitales suficientes para el proyecto sin tener que ampliar con más módulos (a ser posible para no incrementar el coste económico total del equipo).

La decisión final ha sido la adquirir de una CPU S7 314C-2DP (ref. Siemens 6ES7314-6CG03-0AB0). Esta CPU, tiene, una relación calidad/prestaciones/precio bastante equilibrada, lo que hace que su adquisición sea más recomendable que otras CPU situadas por debajo o por encima de ésta. En cuanto a características técnicas, dispone de puerto de comunicaciones MPI y PROFIBUS DP, solventando el tema de las comunicaciones a través de protocolo PROFIBUS DP. En cuanto a E/S digitales, posee ya integradas 24 entradas digitales y 16 salidas digitales, siendo este número, en principio, suficiente para el desarrollo de todo el proyecto según la carga y distribución de automatización prevista en el mismo. Las CPUs situadas en la gama por encima de la seleccionada para este proyecto tienen un precio más elevado, puesto que también integran cualidades no utilizadas en este proyecto y que elevan su precio (por ejemplo, capacidad de comunicación Ethernet o PROFINET, elevada cantidad de E/S digitales a bordo, etc). Las CPU situadas en la gama por debajo de la seleccionada no poseen alguna de las características necesarias para este proyecto, por ejemplo, no disponen de puerto PROFIBUS DP o no tienen el número de E/S digitales necesarias a bordo. Cualquiera de estos problemas se solventaría adquiriendo módulos de ampliación especializados, pero, en ese momento, la justificación económica de su compra quedaría sin validez, porque aún saldría económicamente menos viable que adquirir la CPU finalmente seleccionada.

De este modo se puede concluir que la CPU S7 314C-2DP seleccionada finalmente es la más adecuada y equilibrada en prestaciones para el desarrollo de este proyecto.

En la siguiente figura se puede observar el rendimiento comparado de las diferentes CPUs de la serie S7-300:

Performance overview S7-300 CPUs



Six performance classes of the S7-300 CPUs

1) Connections stand for internal resources of the CPU for the communication with PGs/OPs and over blocks. The standard bus communication and the PtP coupling do not require connections. The PN-CPUs offer 8 resp. 16 resp. 32 additional connections for TCP/IP, UDP, and ISO-on-TCP.

La siguiente tabla muestra las características generales de la serie S7-300:

Datos técnicos

Datos técnicos generales S7-300, S7-300F	
Grado de protección	IP20 según IEC 60 529
Temperatura ambiente	
• para montaje horizontal	0 a 60 °C
• para montaje vertical	0 a 40 °C
Humedad relativa	5 a 95%, sin condensación (grado de solicitación RH 2 según IEC 61131-2)
Presión atmosférica	795 a 1080 hPa
Aislamiento	
• Circuitos 24 V DC	Tensión de ensayo 500 V DC
• Circuitos 230 V AC	Tensión de ensayo 1460 V AC
Compatibilidad electromagnética	Requerimientos de la Directiva CEM;
	Inmunidad a las descargas electroestáticas conforme a IEC 61000-6-2, ensayo según : IEC 61000-4-2, 61000-4-3, IEC 61000-4-4, IEC 61000-4-5, IEC 61000-4-6
	Emisión de perturbaciones según EN 50081-2, comprobación según EN 55011, clase A, grupo 1
Cargas mecánicas	
• Vibraciones, ensayo según / ensayadas con	IEC 60068, parte 2-6/10 a 58 Hz; amplitud constante 0,075 mm; 58 a 150 Hz; aceleración constante 1 g; duración de la vibración: 10 barridos de frecuencia por eje en cada sentido de los tres ejes perpendiculares entre sí
• Choques ensayados según/con	IEC 60068, Parte 2-27/semi-sinusoidal: intensidad del choque 15 g (valor cresta), duración 11 ms

La tabla completa de características técnicas de la CPU seleccionada para el proyecto (S7 314C-2DP) se expone a continuación:

Datos técnicos

	6ES7 314-6CG03-0AB0
Versión	
Paquete de programas asociado	STEP 7 a partir de 5.2 + SP1 + actualización de HW
Tensiones de alimentación	
Valor nominal	
• 24 V DC	Sí
• Rango admisible, límite inferior (DC)	20,4 V
• Rango admisible, límite superior (DC)	28,8 V
Tensiones e intensidades	
Protección externa para líneas de alimentación (recomendación)	Interruptor automático magnetotérmico tipo C mín. 2 A; interruptor automático magnetotérmico tipo B mín. 4 A
Consumo	
Intensidad de cierre, típ.	11 A
I^2t	0,7 A^2s
Consumo (en marcha en vacío), típ.	150 mA
Consumo (valor nominal)	1.000 mA
de la tensión de alimentación L+, máx.	1.000 mA
Pérdidas, típ.	14 W
Memoria	
Tipo de memoria	
• Memoria de trabajo	
- integrada	96 Kbyte(s); para programa y datos
- ampliable	No

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
• Memoria de carga	
- Enchufable (MMC)	Sí
- Enchufable (MMC), máx.	8 Mbyte(s)
Respaldo	
• existente	Sí; garantizado por la MMC (sin mantenimiento)
• sin pila	Sí; Programa y datos
CPU/bloques	
DB	
• Cantidad, máx.	511; Banda de números: 1 a 511
• Tamaño, máx.	16 Kbyte(s)
FB	
• Cantidad, máx.	1.024; Banda numérica: 0 a 2047
• Tamaño, máx.	16 Kbyte(s)
FC	
• Cantidad, máx.	1.024; Banda numérica: 0 a 2047
• Tamaño, máx.	16 Kbyte(s)
OB	
• Cantidad, máx.	Ver Lista de operaciones
• Tamaño, máx.	16 Kbyte(s)
Profundidad de anidamiento	
• por cada prioridad	8
• adicional, dentro de un OB de error	4
CPU/tiempos de ejecución	
para operaciones de bits, mín.	0,1 µs
para operaciones de palabras, mín.	0,2 µs
para aritmética en coma fija, mín.	2 µs
para aritmética en coma flotante, mín.	3 µs
Temporizadores/contadores y su remanencia	
Contadores S7	
• Cantidad	256
• de ellos, remanentes sin pila	
- configurable	Sí
- Límite inferior	0
- Límite superior	255
• Remanencia	
- configurable	Sí
- Límite inferior	0
- Límite superior	255
• Rango de contaje	
- Límite inferior	0
- Límite superior	999

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Contadores IEC	
• existente	Si
• Clase	SFB
Temporizadores S7	
• Cantidad	256
• Remanencia	
- configurable	Si
- Límite inferior	0
- Límite superior	255
- predeterminado	sin remanencia
• Rango de tiempo	
- Límite inferior	10 ms
- Límite superior	9.990 s
Temporizadores IEC	
• existente	Si
• Clase	SFB
Áreas de datos y su remanencia	
Marcas	
• Cantidad, máx.	256 byte(s)
• Remanencia disponible	Si; MB 0 a MB 255
• Nº de marcas de ciclo	8; 1 byte de marcas
Bloques de datos	
• Cantidad, máx.	511
• Tamaño, máx.	16 Kbyte(s)
• Remanencia configurable	Si; a través de la propiedad de volatibilidad del DB
• Remanencia predeterminada	Si
Datos locales	
• por cada prioridad, máx.	510 byte(s)
Área de direcciones	
Área de direcciones de periferia	
• Entradas	1 Kbyte(s)
• Salidas	1 Kbyte(s)
• de ellas, descentralizadas	
- Entradas	1.000 byte(s)
- Salidas	1.008 byte(s)
Imagen del proceso	
• Entradas	128 byte(s)
• Salidas	128 byte(s)
Canales digitales	
• Entradas	8.192
• Salidas	8.192
• Entradas, de ellas centralizadas	1.016
• Salidas, de ellas centralizadas	1.008

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Canales analógicos	
• Entradas	512
• Salidas	512
• Entradas, de ellas centralizadas	253
• Salidas, de ellas centralizadas	250
Configuración del hardware	
Aparatos centrales, máx.	1
Aparatos de ampliación, máx.	3
Bastidores, máx.	4
Módulos por bastidor, máx.	8; en el bastidor 3 máx. 7
Nº de maestros DP	
• integrada	1
• Vía CP	4
Nº de FM y CP utilizables (recomendación)	
• FM	8
• CP, punto a punto	8
• CP, LAN	10
Hora	
Reloj	
• Reloj por hardware (reloj tiempo real)	Si
• Reloj por software	
• respaldado y sincronizable	Si
• Desviación diaria, máx.	10 s
Contador de horas de funcionamiento	
• Cantidad	1
• Número/banda numérica	0
• Rango de valores	2^31 horas (utilizando el SFC 101)
• Granularidad	1 hora
• remanente	Si; tiene que reiniciarse en cada rearranque
Sincronización de la hora	
• soportada	Si
• en MPI, maestro	Si
• en MPI, esclavo	Si
• en el autómata, maestro	Si
Funciones de aviso S7	
Cantidad de equipos que pueden conectarse para funciones de aviso, máx.	12; depende de las conexiones configuradas para la comunicación PG/OP y S7 básica
Avisos de diagnóstico de proceso	Si
Bloques Alarm-S activos simultáneamente, máx.	40

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Funciones de test y puesta en marcha	
Estado/forzado	
• Variable Estado/Forzado	Sí
• Variables	Entradas, salidas, marcas, DB, temporizadores, contadores
• Nº de variables, máx.	30
• De ellas, variables de estado, máx.	30
• De ellas, variables de forzado, máx.	14
Forzado permanente	
• Forzado permanente	Sí
• Forzado permanente, variables	Entradas, salidas
• Nº de variables, máx.	10
Bloque Estado	Sí
Paso individual	Sí
Nº de puntos de parada	2
Búfer de diagnóstico	
• existente	Sí
• Nº de entradas, máx.	100
• configurable	
Funciones de comunicación	
Comunicación PG/OP	Sí
Enrutado	Sí
Comunicación de datos globales	
• soportada	Sí
• Tamaño de paquetes GD, máx.	22 byte(s)
Comunicación básica S7	
• soportada	Sí
Comunicación S7	
• soportada	Sí
Comunicación compatible con S5	
• soportada	Sí; a través de CP y FC cargables
Nº de conexiones	
• Total	12
• usable para comunicación PG	11
• usable para comunicación OP	11
• usables para comunicación básica S7	8
• usables para enrutado	4; máx.
Sistema de conexión	
Conector frontal requerido	2 x 40 polos
MPI	
Longitud del cable, máx.	50 m; sin repetidor

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Punto a punto	
Longitud del cable, máx.	
Drivers de protocolo integrados	
• 3964 (R)	
• ASCII	
• RK512	
Velocidad de transferencia RS 422/485	
• con protocolo 3964(R), máx.	
• con protocolo ASCII, máx.	
• con protocolo RK 512, máx.	
1. interfaz	
Tipo de interfaz	Interfaz RS485 integrada
Norma física	RS 485
Con aislamiento galvánico	No
Alimentación en interfaz (15 a 30 V DC), máx.	200 mA
Funcionalidad	
• MPI	Sí
• Maestro DP	No
• Esclavo DP	No
• Acoplamiento punto a punto	No
MPI	
• Nº de conexiones	12
• Servicios	
- Comunicación PG/OP	Sí
- Enrutado	Sí
- Comunicación de datos globales	Sí
- Comunicación básica S7	Sí
- Comunicación S7	Sí
- Comunicación S7, como client	No
- Comunicación S7, como servidor	Sí
• Velocidades de transmisión, máx.	187,5 kBit/s
2. interfaz	
Tipo de interfaz	Interfaz RS 485 integrada
Norma física	RS 485
Con aislamiento galvánico	Sí
Alimentación en interfaz (15 a 30 V DC), máx.	200 mA

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Funcionalidad	
• MPI	No
• Maestro DP	Sí
• Esclavo DP	Sí
• Acoplamiento punto a punto	No
• PROFINET CBA	No
• PROFINET IO-Controller	No
Maestro DP	
• Número de conexiones, máx.	12; para comunicación PG/OP
• Nº de conexiones (de ellas, reservadas), máx.	1 para PG, 1 para OP
• Servicios	
- Comunicación PG/OP	Sí
- Enrutado	Sí
- Comunicación de datos globales	No
- Comunicación básica S7	Sí
- Comunicación S7	Sí
- Comunicación S7, como client	No
- Comunicación S7, como servidor	Sí
- Soporte de equidistancia	Sí
- SYNC/FREEZE	Sí
- Activar/desactivar esclavos DP	Sí
- Intercambio directo de datos (comunicación directa)	Sí
- DPV1	Sí
• Velocidades de transmisión, máx.	12 Mbits/s
• Nº de esclavos DP, máx.	32
• Área de direcciones	
- Entradas, máx.	1 Kbyte(s)
- Salidas, máx.	1 Kbyte(s)
• Datos útiles por esclavo DP	
- Entradas, máx.	244 byte(s)
- Salidas, máx.	244 byte(s)
Esclavo DP	
• Nº de conexiones	12
• Servicios	
- Comunicación PG/OP	Sí
- Enrutado	Sí; sólo con interfaz activa
- Comunicación de datos globales	No
- Comunicación básica S7	Sí
- Comunicación S7, como client	No
- Comunicación S7, como servidor	Sí
- Intercambio directo de datos (comunicación directa)	Sí
- DPV1	No

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
• Archivo GSD	Encontrará el archivo GSD actual en: http://www.ad.siemens.de/support en el área Product Support
• Velocidades de transmisión, máx.	12 kBit/s
• Búsqueda automática de velocidad de transferencia	Sí; sólo con interfaz pasiva
• Memoria de transferencia	
- Entradas	244 byte(s)
- Salidas	244 byte(s)
• Área de direcciones, máx.	32
• Datos útiles por área de direcciones, máx.	32 byte(s)
Acoplamiento punto a punto	
• Velocidad de transferencia, máx.	
• Longitud del cable, máx.	
• Interfaz controlable desde el programa de usuario	
• Interfaz puede disparar alarmas/interrupciones en el programa de usuario	
• Driver de protocolo	
CPU/programación	
Lenguaje de programación	
• STEP 7	Sí; V5.2 SP1 con actualización de HW
• KOP	Sí
• FUP	Sí
• AWL	Sí
• SCL	Sí
• CFC	Sí
• GRAPH	Sí
• HiGraph	Sí
Librerías de software	
Juego de operaciones	ver Lista de operaciones
Niveles de paréntesis	8
Protección de programas de usuario/Protección por contraseña	Sí
Funciones de sistema (SFC)	ver Lista de operaciones
Bloques de función de sistema (SFB)	ver Lista de operaciones

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Módulos de E digitales	
Nº de entradas digitales	24
• De ellas, entradas usable para funciones tecnológicas	16
Número de entradas atacables simultáneamente	
• Posición de montaje vertical	
- hasta 40 °C, máx.	12
• Posición de montaje horizontal	
- hasta 40 °C, máx.	24
- hasta 60 °C, máx.	12
Longitud del cable	
• Longitud del cable apantallado, máx.	1.000 m; 50 m para funciones tecnológicas
• Longitud del cable no apantallado, máx.	600 m; Para funciones tecnológicas: no
• Funciones tecnológicas	
- apantallado, máx.	50 m
- no apantallado, máx.	no permitido
• DI estándar	
- apantallado, máx.	1.000 m
- no apantallado, máx.	600 m
Característica de entrada según IEC 1131, tipo 1	Si
Tensión de entrada	
• Valor nominal, DC	24 V
• para señal "0"	-3 a 5 V
• para señal "1"	15 a 30 V
Intensidad de entrada	
• para señal "1", tip.	9 mA
Retardo de entrada (a tensión nominal de entrada)	
• para entradas estándar	
- parametrizable	Si; 0,1/0,3/3/15 ms
- Valor nominal	3 ms
• para contadores/funciones tecnológicas:	
- en transición "0" a "1", máx.	8 µs
Módulos de S digitales	
Número de salidas digitales	16
• De ellas, salidas rápidas	4
Longitud del cable apantallado, máx.	1.000 m
Longitud del cable no apantallado, máx.	600 m
Protección contra cortocircuitos en salida	Si; por pulsación electrónica
• Umbral de respuesta, tip.	1 A
Limitación de la sobretensión inductiva de corte a	L+ (-48 V)
Carga tipo lámpara, máx.	5 W
Ataque de una entrada digital	Si

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Tensión de salida	
• para señal "1", mín.	L+ (-0,8 V)
Intensidad de salida	
• para señal "1" valor nominal	500 mA
• rango admisible para señal "1", mín.	5 mA
• rango admisible para señal "1", máx.	0,6 A
• para señal "1" intensidad de carga mínima	5 mA
• para señal "0" Intensidad residual, máx.	0,5 mA
Conexión en paralelo de 2 salidas	
• para aumentar la potencia	No
• para control redundante de una carga	Si
Frecuencia de conmutación	
• con carga resistiva, máx.	100 Hz
• con carga inductiva, máx.	0,5 Hz
• con carga tipo lámpara, máx.	100 Hz
• de las salidas de impulsos, con carga óhmica, máx.	2,5 kHz
Intensidad suma de las salidas (por grupo)	
• Posición de montaje vertical	
- hasta 40 °C, máx.	2 A
• Posición de montaje horizontal	
- hasta 40 °C, máx.	3 A
- hasta 60 °C, máx.	2 A
Rango de resistencia de carga	
• Límite inferior	48 Ω
• Límite superior	4 kΩ
Entradas analógicas	
Nº de entradas analógicas para medida de tensión/intensidad	4
Nº de entradas analógicas para medida de resistencia/temperatura	1
Longitud del cable apantallado, máx.	100 m
Tensión de entrada admisible para entrada de tensión (límite de destrucción), máx.	30 V; permanente
Tensión de entrada admisible para entrada de intensidad (límite de destrucción), máx.	2,5 V; permanente
Intensidad de entrada admisible para entrada de tensión (límite de destrucción). máx.	0,5 mA; permanente
Intensidad de entrada admisible para entrada de intensidad (límite de destrucción), máx.	50 mA; permanente
Unidad para medida de temperatura ajustable	Si; Grados Celsius/grados Fahrenheit/Kelvin

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Rangos de entrada (valores nominales), tensiones	
• 0 a +10 V	Sí
• -10 V a +10 V	Sí
Rangos de entrada (valores nominales), intensidades	
• 0 a 20 mA	Sí
• -20 a +20 mA	Sí
• 4 a 20 mA	Sí
Rangos de entrada (valores nominales), resistencias	
• Tensión en vacío, tip.	2,5 V
• Intensidad de medida, tip.	1,8 mA a 3,3 mA
• 0 a 600 óhmios	Sí
Rangos de entrada (valores nominales), termorresistencias	
• Pt 100	Sí
Linealización de característica	
• parametrizable	Sí; software
• para termorresistencias	Pt 100
Compensación de temperatur	
• parametrizable	No
Salidas analógicas	
Nº de salidas analógicas	2
Longitud del cable apantallado, máx.	200 m
Salida de tensión, protección de cortocircuito	Sí
Salida de tensión, intensidad de cortocircuito, máx.	55 mA
Salida de intensidad, tensión en vacío, máx.	17 V
Rangos de salida, tensión	
• 0 a 10 V	Sí
• -10 a +10 V	Sí
Rangos de salida, intensidad	
• 0 a 20 mA	Sí
• -20 a +20 mA	Sí
• 4 a 20 mA	Sí
Conexión de actuadores	
• Para salidas de tensión, conexión a 2 hilos	Sí; sin compensación de la resistencia de los cables
• Para salidas de tensión, conexión a 4 hilos	No
• Para salidas de intensidad, conexión a 2 hilos	Sí
Resistencia de carga (en rango nominal de la salida)	
• con salidas de tensión, mín	1 kΩ

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
• con salidas de tensión, carga capacitiva, máx.	0,1 µF
• con salidas de intensidad, máx.	300 Ω
• con salidas de intensidad, carga inductiva, máx.	0,1 mH
Límite de destrucción por tensiones y intensidades aplicadas desde el exterior	
• Tensiones en las salidas con respecto a MANA	16 V; ermanente
• Intensidad, máx.	50 mA; permanente
Formación de valores analógicos	
Principio de medición	Codificación instantánea (aproximación sucesiva)
Tiempo de integración y conversión/resolución por canal	
• Resolución con rango de rebase (bits incl. signo), máx.	12 bits
• Tiempo de integración parametrizable	Sí; 2,5/16,6/20 ms
• Frecuencia de entrada permitida, máx.	400 Hz
• Supresión de perturbaciones de tensión para frecuencia perturbadora f1 en Hz	400/60/50 Hz
• Tiempo de conversión (por canal)	1 ms
• Constante del filtro de entrada	0,38 ms
• Tiempo de ejecución básico del módulo (todos los canales habilitados)	1 ms
Tiempo de estabilización	
• para carga resistiva	0,6 ms
• para carga capacitiva	1 ms
• para carga inductiva	0,5 ms
Sensor	
Conexión de los sensores	
• para medida de tensión	Sí
• para medición de intensidad como transductor a 2 hilos	Sí; con alimentación externa
• para medición de intensidad como transductor a 4 hilos	Sí
• para medición de resistencia con conexión a 2 hilos	Sí; sin compensación de la resistencia de los cables
• para medición de resistencia con conexión a 3 hilos	No
• para medición de resistencia con conexión a 4 hilos	No
Sensores compatibles	
• BERO a 2 hilos	Sí
• Intensidad permitida en reposo (BERO a 2 hilos), máx.	1,5 mA

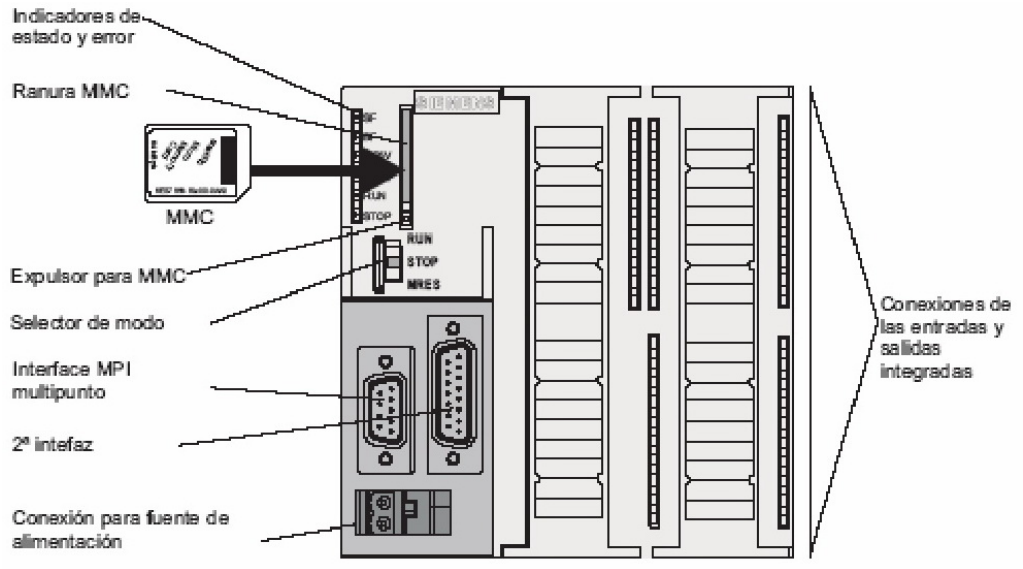
Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Error/precisiones	
Error de temperatura (referido al rango de entrada)	± 0,006 %/K
Diafonía entre entradas, mín.	60 dB
Precisión de repetición en estado estacionario a 25 °C (referido al rango de entrada)	± 0,06 %
Ondulación de salida (referida al rango de salida, ancho de banda 0 a 50 kHz)	± 0,1 %
Error de linealidad (referido al rango de salida)	± 0,15 %
Error por temperatura (referido al rango de salida)	± 0,01 %/K
Diafonía entre las salidas, mín.	60 dB
Precisión de repetición en estado estacionario a 25 °C (referido al rango de salida)	± 0,06 %
Límite de error práctico en todo el rango de temperatura	
• Tensión, referida al rango de entrada	± 1 %
• Intensidad, referida al rango de entrada	± 1 %
• Resistencia, referida al rango de entrada	± 5 %
• Tensión, referida al rango de salida	± 1 %
• Intensidad, referida al rango de salida	± 1 %
Límite de error básico (límite de error práctico a 25 °C)	
• Tensión, referida al rango de entrada	± 0,7 %; Error de linealidad ±0,06%
• Intensidad, referida al rango de entrada	± 0,7 %; Error de linealidad ±0,06%
• Resistencia, referida al rango de entrada	± 3 %; Error de linealidad ±0,2%
• Termorresistencia, referida al rango de entrada	± 3 %
• Tensión, referida al rango de salida	± 0,7 %
• Intensidad, referida al rango de salida	± 0,7 %
Supresión de frecuencias perturbadoras para f = n x (fl ± 1 %), fl = frecuencia perturbadora	
• Perturbación en modo serie (pico de la perturbación < valor nominal del rango de entrada), mín.	30 dB
• Perturbación en modo común, mín.	40 dB

Datos técnicos (continuación)

	6ES7 314-6CG03-0AB0
Funciones integradas	
Nº de contadores	4; Ver manual "Funciones tecnológicas"
Frecuencia de contaje (contadores), máx.	60 kHz
Medición de frecuencia	Sí
Posicionamiento en lazo abierto	Sí
Regulador PID	Sí
Nº de salidas de impulsos	4; Modulación de ancho de impulso hasta máx. 2,5 kHz (ver manual "Funciones tecnológicas")
Frecuencia límite (impulsos)	2,5 kHz
Aislamiento galvánico	
Aislamiento galvánico módulos de E digitales	
• Aislamiento galvánico módulos de E digitales	Sí
• entre los canales	No
• entre los canales, en grupos de	
• entre los canales y el bus de fondo	Sí
Aislamiento galvánico módulos de S digitales	
• Aislamiento galvánico módulos de S digitales	Sí
• entre los canales	Sí
• entre los canales, en grupos de	8
• entre los canales y el bus de fondo	Sí
Aislamiento galvánico módulos E analógicas	
• Aislamiento galvánico módulos E analógicas	Sí
• entre los canales	No
• entre los canales y el bus de fondo	Sí
Aislamiento galvánico módulos de S analógicas	
• Aislamiento galvánico módulos de S analógicas	Sí
• entre los canales	No
• entre los canales y el bus de fondo	Sí
Dimensiones	
Ancho	120 mm
Alto	125 mm
Profundidad	130 mm
Pesos	
Peso, aprox.	676 g

En la siguiente imagen se representa la vista general de una CPU 314C-2DP:



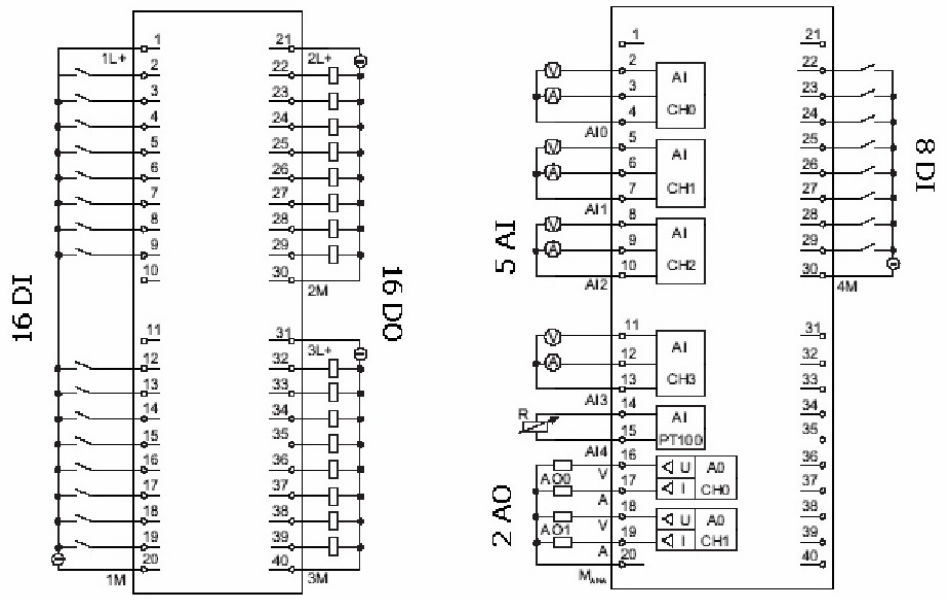
Posee varios indicadores en su frontal:

Indicadores de la CPU 314C-2DP	
SF (rojo)	Fallo de hardware o software
BF (rojo)	Fallo de bus
DC5V (verde)	Alimentación de 5 V para CPU y bus S7-300 funciona correctamente
FRCE (amarillo)	Petición de forzado activa
RUN (verde)	CPU en RUN; Led parpadea en ARRANQUE a 2 Hz; en PARADA a 0.5 Hz
STOP (amarillo)	CPU en STOP o en PARADA o en ARRANQUE. Led parpadea a 0.5 Hz al solicitar borrado, durante el borrado a 2 Hz

- En el selector de modo de operación son posibles tres posiciones:
  - RUN: la CPU procesa el programa de usuario.
  - STOP: la CPU no procesa ningún programa de usuario.
  - MRES: borrado total: Posición no enclavable del selector para el borrado total de la CPU.
- Dispone de ranura para el cartucho de memoria. El tipo de memoria es Micro Memory Card (MMC). Es necesaria para el funcionamiento de la CPU, pues este modelo no dispone de memoria de carga integrada.
- Esta CPU posee dos interfaces de comunicación integrados, que son:
  - Inteface MPI (Multi Point Inteface).
  - Interface PROFIBUS DP (Periferia Descentralizada).

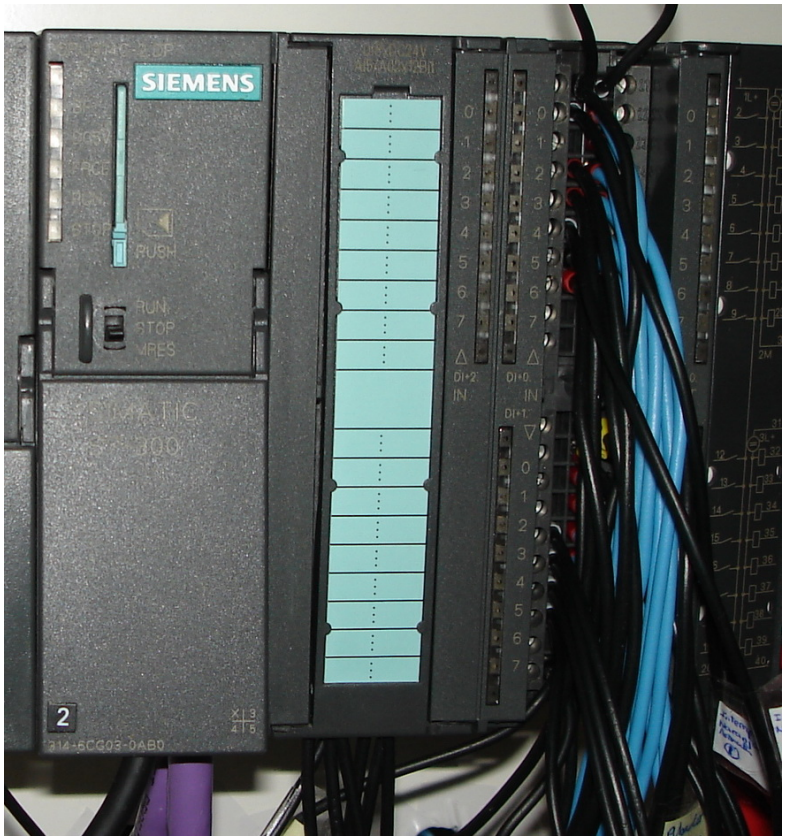
- Posee 24 entradas digitales, 16 salidas digitales. 5 entradas analógicas y 2 salidas analógicas integradas.

La disposición de las entradas/salidas de esta CPU es:



Para información más ampliada de esta CPU se adjuntan como anexos los datasheets de Siemens con la totalidad de datos técnicos sobre el material.

En las siguientes imágenes se puede ver la CPU utilizada en el proyecto:


SIEMENS
RUN
STOP
MRES
2


RUN
STOP
MRES
V2.6.6
X1 MPI
X2 DP

9.2. Perfil soporte para CPU S7-300

Es el soporte mecánico de los módulos SIMATIC S7-300. Los módulos, incluida la CPU, se anclan físicamente al perfil o bastidor, que a su vez es atornillable a pared o, en el caso de este proyecto, al tablero de madera donde se han distribuido todos los dispositivos de automatización.

Se encuentra disponible en varias dimensiones en cuanto a longitud se refiere, siendo el perfil 482 mm el que se va a utilizar en el proyecto, debido a que no es necesaria mayor longitud de soporte. Su referencia Siemens es 6ES7 390-1AE80-0AA0. En el proyecto se va a utilizar 1 Uds de esta referencia.

Perfil soporte para CPU S7 300

9.3. Conector frontal

Este componente es utilizado para conectar de forma simple y cómoda sensores y actuadores. Como el conector es extraíble (lleva un tornillo de fijación al módulo), no es necesario tocar el cableado cuando se cambia de módulo. Lleva elementos codificadores para evitar errores al sustituir el módulo.

La referencia que se va a emplear en el proyecto es Siemens 6ES7 392-1AM00-0AA0, que se corresponde con un borne de 40 polos con bornes de tornillo. Serán necesarios dos bornes de este tipo para los dos módulos de E/S integrados en la CPU.

Conector frontal

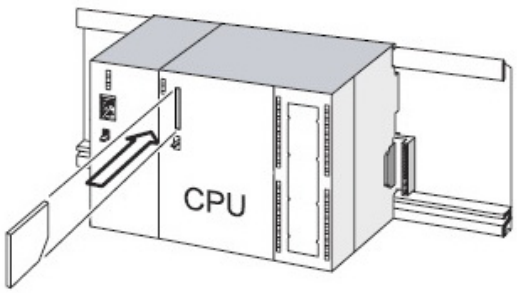
9.4. Micro Memory Card 2MB

En el proyecto se va a utilizar 1 Uds de memory card de 2 MB (referencia Siemens 6ES7 953-8LL20-0AA0), que corresponde a una nueva generación de micro memory card del fabricante. La memory card debe seleccionarse en base a su tamaño, ya que todas disponen de la misma tecnología, cambiando sólo el tamaño disponible. En este caso se ha seleccionado una memoria de 2 MB, que es más que suficiente para hacer un backup del programa de usuario completo, además de datos de usuario si es necesario. Para hacer un backup de todo el sistema operativo de la CPU haría falta, no obstante, una micro memory card de 4 MB como mínimo. En este proyecto no se ha contemplado esta opción ya que la carga de trabajo, ambiente de trabajo y demás factores que condicionan el trabajo de la CPU hacen improbable la necesidad de almacenar todo el sistema operativo pensando en fallos críticos de la CPU. Sin embargo, en ambientes de más riesgo es una práctica aconsejable.

La micro memory card de 2MB, tiene una relación calidad/precio aceptable. A partir de 2 MB de tamaño, la capacidad aumenta a 4 MB o a 8 MB, disparando también su precio.

Micro Memory Card

En la siguiente imagen se puede ver el lugar donde se debe insertar la memory card en la CPU.



9.5. Fuente de alimentación S7 5A

Esta es la fuente de alimentación de carga para el S7-300. Su función es la de convertir la tensión de red en tensión de 24V DC, que es la empleada por el autómata y sensores, actuadores y dispositivos conectados al mismo.

En el proyecto se va a utilizar 1 Uds de la fuente de alimentación de 5 A de intensidad de salida.

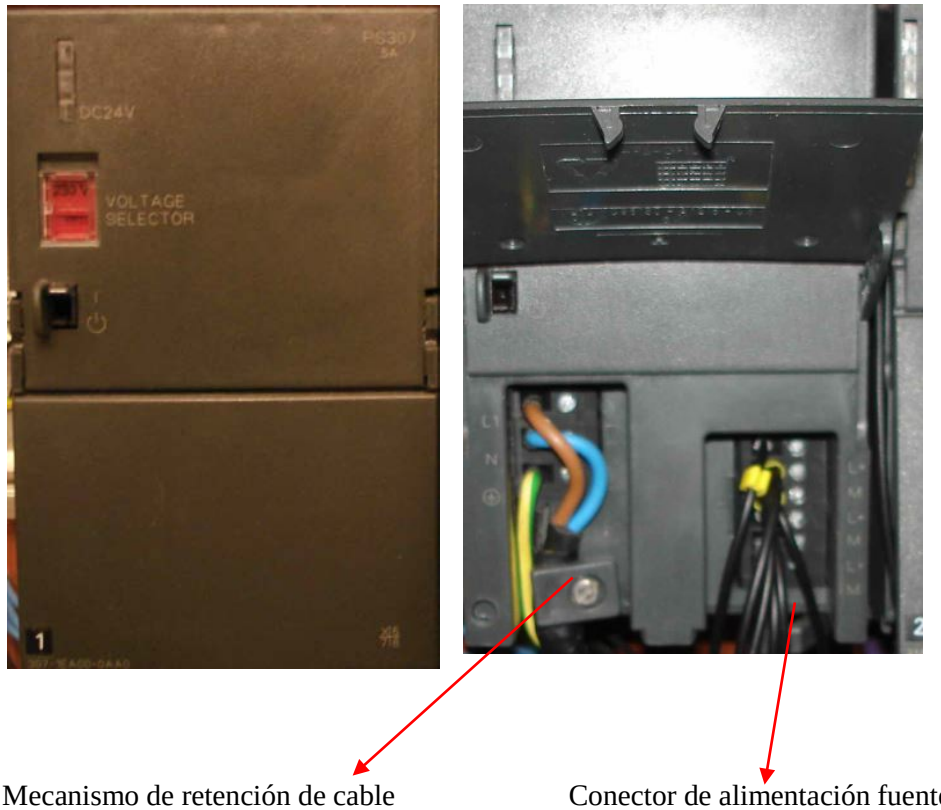
Aunque la fuente de alimentación que se va a utilizar no tiene por qué ser de la marca Siemens, se ha decidido adquirir ésta porque Siemens fabrica estas fuentes pensando en el acoplamiento directo de la misma al conjunto del autómata. De esta forma, se puede anclar perfectamente al perfil soporte del conjunto (visto anteriormente). Además, dispone de un conector concebido especialmente para conectar la salida de la fuente de 24V DC a la alimentación de la CPU de 24V DC. Este conector es muy útil porque evita tener que cablearlo manualmente y además es muy seguro, puesto que es compacto y no hay riesgo de que se corte ningún cable.

La fuente seleccionada para el proyecto ha sido la referencia 6ES7 307-1EA00-0AA0, de 5A de intensidad de salida.

Fuente de alimentación S7 5A

Conector

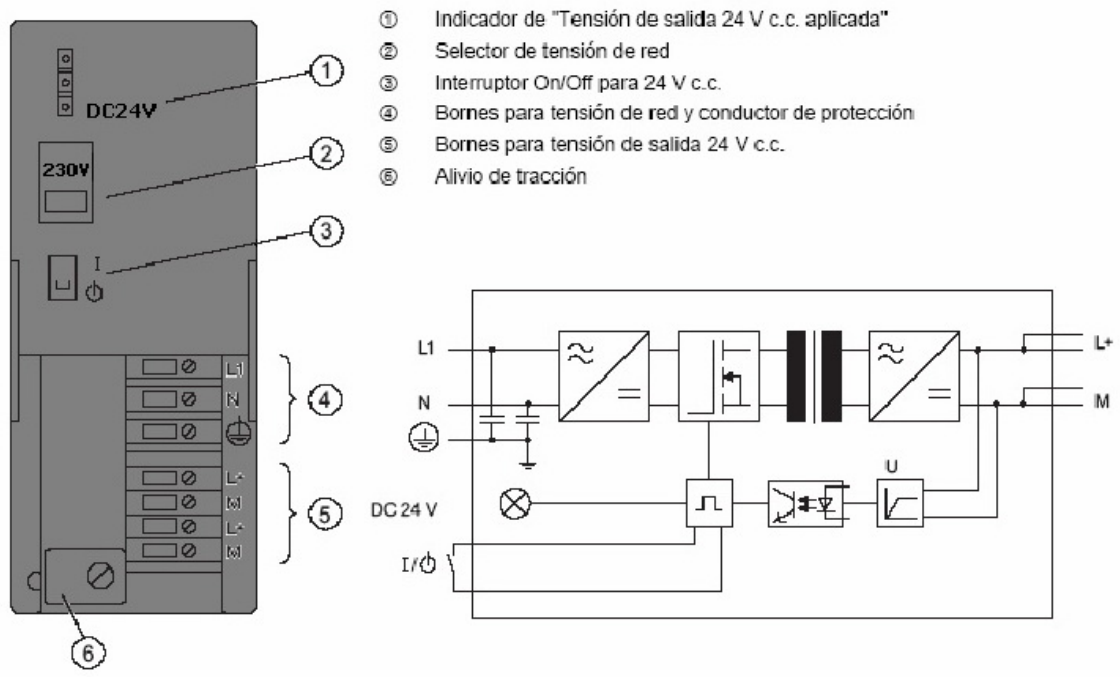
En la siguiente imagen se puede observar la fuente de alimentación utilizada en el proyecto:



Mecanismo de retención de cable

Conector de alimentación fuente/CPU

Aquí se muestra una vista general de la fuente de alimentación PS307-5A:



9.6. SIMATIC PC ADAPTER USB

El SIMATIC PC ADAPTER USB conecta un PC a través de interfaz USB con la interfaz MPI/DP de un sistema S7/C7. No es necesario un slot en el PC, es decir, el adaptador es también adecuado para los PCs no ampliables, como pueden ser los portátiles. El PC ADAPTER USB sustituye al PC ADAPTER con puerto serie. Con él, es posible programar todas las CPUs de las gamas S7-300 y S7-400 a través de su puerto MPI/DP. También se puede conectar al IM 151-7 de la ET 200S. Las CPUs del WinAC Slot son excepciones, ya que no suministran los 24V de alimentación necesarios.

La siguiente imagen muestra el PC Adapter USB que se va a utilizar en el proyecto:

PC Adapter USB

Las características y funciones más importantes son:

- Búsqueda de la velocidad y del perfil automática.
- Hasta 16 enlaces de comunicación, de los cuales hasta 4 pueden ser para esclavos.
- Posibilidad de trabajar con la función de routing.
- El adaptador se alimenta a través del puerto MPI (24 V). El anterior PC Adapter precisaba de una alimentación de 5V independiente.
- El firmware del nuevo PC Adapter se puede actualizar a su último estado para funciones adicionales o corrección de errores.
- Compatible con USB v1.1.
- Prestaciones considerablemente aumentadas con respecto al PC Adapter anterior (serie).

Sus velocidades de transmisión son:

Velocidad	MPI	Profibus DP
9.600 (bits/s)		x
19.200 (bits/s)	x	x
45.450 (bits/s)		x
93.750 (bits/s)		x
187.500 (bits/s)	x	x
500 (Kbits/s)		x
1.500 (Kbits/s)	x	x

El PC Adapter USB incluye los siguientes componentes:

- CD ROM “SIMATIC Software PC Adapter USB”, con software (drivers) y documentación.

- Cable USB de 5 m de longitud.
- Cable MPI de 0,3 m de longitud (limitado por razones técnicas).

El PC Adapter USB necesita un PC con uno de los siguientes sistemas operativos:

- Windows 2000
- Windows XP Profesional
- Windows XP Home

También es necesario un paquete de software SIMATIC que permita la comunicación MPI (por ejemplo, STEP 7).

9.7. CPUs SERIE S7-200

En la estructura de PLCs diseñada para el proyecto, se deben integrar 2 Uds de PLCs de la serie S7-200, que actuarán de esclavos en la red PROFIBUS DP, dependiendo directamente de las órdenes del S7 314C-2DP, su maestro en la red. Cada uno de los S7-200 controla las áreas de influencia de cada estación (Estación 1 y Estación 2), de ahí que se requieran dos CPUs, ya que en el diseño de la maqueta intervienen dos estaciones ferroviarias.

Elección de la CPU y justificación técnica para el proyecto

El SIMATIC S7-200 es un micro-PLC compacto y potente, particularmente en lo que atañe a respuesta en tiempo real. Es rápido, ofrece una conectividad excelente y todo tipo de facilidades en el manejo del software y del hardware. El SIMATIC-S7 200 está orientado a maximizar la rentabilidad. Toda la gama de CPUs ofrece un alto nivel de prestaciones, una modularidad óptima y una alta conectividad, por lo que es aplicable tanto para los controles más simples como también para tareas complejas de automatización. El SIMATIC S7-200 se puede conectar aislado, en red o en configuraciones descentralizadas.

Entre las principales características de esta gama de PLCs, se pueden destacar tres aspectos:

- Comunicación abierta:
    - Puerto estándar RS-485 con velocidad de transferencia de datos comprendida entre 0,3 y 187,5 kbits/s.
    - Protocolo PPI en calidad de bus del sistema para interconexión sin problemas.
    - Modo libremente programable con protocolos personalizados para comunicación con cualquier equipo.
    - Rápido en la comunicación por PROFIBUS vía módulo dedicado, operando como esclavo. Todas las CPUs a partir de la 222 son aptas para comunicación PROFIBUS vía módulo esclavo PROFIBUS DP.
    - Potente en la comunicación por bus AS-Interface, operando como maestro (a partir de la CPU 222).
    - Accesibilidad desde cualquier punto gracias a comunicación por modem (para telemantenimiento, teleservice o telecontrol).
    - Conexión a Industrial Ethernet vía módulo dedicado.
    - Con conexión a Internet mediante módulo correspondiente.
    - S7-200 PC ACCESS, que es un servidor OPC para simplificar la conexión al mundo del PC.
- Altas prestaciones:

  - Pequeño y compacto, ideal para aplicaciones donde se cuenta con reducido espacio
  - Extensa funcionalidad básica uniforme en todos los tipos de CPU.
  - Alta capacidad de memoria.
  - Extraordinaria respuesta en tiempo real; la posibilidad de dominar en cualquier instante todo el proceso permite aumentar la calidad, la eficiencia y la seguridad.
  - Manejo simplificado gracias a software de fácil uso STEP 7-Micro/WIN, ideal tanto para novatos como para expertos.
- Software
  - Gran facilidad de uso
  - Estándar Windows.
  - Parametrizar en lugar de programar: los asistentes.
  - Gran repertorio de instrucciones, aplicables por simple “arrastrar y colocar”.
  - Función de visualización de estado para lenguajes AWL, KOP y FUP.

Atendiendo a estas características, y evaluando las tareas que se tienen que llevar a cabo en el proyecto, se deduce que en cuestión de rapidez de la CPU todas podrían ser aptas, ya que la serie completa tiene capacidades muy uniformes en este sentido. Por otro lado, y a diferencia de lo que se ha visto con la CPU de la serie S7-300, las CPUs de la serie S7-200 no llevan integrado el puerto de comunicaciones PROFIBUS DP (en ningún modelo). Por este motivo se hace imprescindible adquirir dos módulos con interfaz PROFIBUS DP dedicado, uno para cada CPU.

Otro de los puntos importantes a determinar es el número de entradas/salidas digitales y analógicas necesarias. En cuanto a las entradas digitales, destacar que únicamente van a ser necesarias ocho entradas en una CPU y nueve entradas en la otra CPU. Este número se corresponde con los sensores que serán conectados a cada CPU. En base a esto se deduce que no habría problema alguno en seleccionar cualquier CPU de la gama.

Sin embargo, son las salidas digitales las que tienen relevancia en esta decisión, puesto que a ellas se van a conectar la totalidad de los semáforos de la maqueta ferroviaria a controlar. Más adelante se verá que son necesarios 18 semáforos para todo el trayecto, teniendo cada semáforo dos leds (dos señales independientes para el autómata por semáforo). Por lo tanto, deben ser 36 las señales individuales controladas por las salidas digitales disponibles en los autómatas.

Una vez más, y al igual que pasaba con la serie S7-300, resulta económicamente más viable la adquisición de una CPU con las E/S necesarias integradas, que adquirir módulos de ampliación de E/S dedicados. Por poner un ejemplo, si se necesitasen 16 salidas digitales en una CPU, resulta más económico adquirir la CPU más alta de la gama, CPU S7 226, que ya las lleva a bordo, que adquirir la CPU inmediatamente inferior, CPU 224, que sólo integra 10 salidas digitales. En ese caso, para la CPU 224 sería necesario adquirir un módulo de ampliación de al menos 8 salidas digitales más, siendo el coste económico más elevado que adquirir la CPU 226 de partida. Además, la CPU 226 integra alguna cualidad más que es superior a CPUs inferiores.

De esta forma, se deduce que se necesitan bastantes salidas digitales en cada CPU, en concreto, 36 salidas digitales a controlar solamente en cuestión de semáforos. En cuanto a la repartición de las competencias de control sobre los mismos, el autómata perteneciente a la estación 1 debería controlar los semáforos que se encuentren dentro del entorno de la estación 1, mientras que el autómata perteneciente a la estación 2 debería controlar los semáforos del entorno de la estación 2.

En cuanto a la relación calidad/precio, la gama S7-200 tiene una relación más equilibrada que la serie S7-300. Esto es debido a que, tecnológicamente hablando, las CPUs de toda la gama tienen unas

características más uniformes que las de la serie 300, y esto hace que los precios varíen también de forma más uniforme y controlada de una CPU a la siguiente. Esto, unido a que los precios de esta gama en sí mismos son más bajos que los de la serie 300, hacen de estos autómatas la elección perfecta para colocar uno de ellos en cada estación ferroviaria.

Debido a que la capacidad máxima de salidas digitales de la gama S7-200 es de 16 salidas en su CPU S7 226, y a la uniformidad en la escala de precios en la gama de CPUs, se decide comprar dos CPUs S7 226 con capacidad total de controlar 16 semáforos entre las dos (32 señales en total). Esto es insuficiente, puesto que se necesitaban controlar 36 señales, por lo que faltan 4 señales que no pueden ser controladas con los S7 226. La solución final adoptada, y teniendo presente el objetivo de no incrementar el coste económico por la compra de un módulo de ampliación de varias salidas digitales para uno de los S7 226, ha sido establecer el control de esas 4 señales (2 semáforos) con el S7 314C-2DP, que tiene salidas digitales libres que no van a ser utilizadas para otros propósitos (es decir, esos dos semáforos van a depender directamente del centro de control de la línea ferroviaria).

De esta forma, todos los semáforos pueden ser controlados por los tres autómatas que componen el sistema de control.

La decisión final ha sido la adquisición de dos CPUs S7-226, referencia Siemens 6ES7 216-2AD23-0XB0.

En la siguiente imagen se muestra una CPU S7-226 general:

CPU S7 226

A continuación, se muestra una de las dos CPUs S7-226 conectada ya al sistema de automatización del proyecto:

9.8. Fuente de alimentación SITOP POWER 3,5A

SITOP Power 24/3,5A es la fuente de alimentación óptima cuando la CPU SIMATIC S7-200 estándar no tiene capacidad suficiente para alimentar todas las cargas. Esta fuente conmutada está totalmente adaptada, tanto en diseño como en funcionalidad, al micro-PLC S7-200. Puede integrarse en el conjunto como cualquier otro módulo S7-200, ya que incluso posee el anclaje de carril DIN propio del SIMATIC S7-200.

Esta fuente de alimentación ha sido seleccionada en primer lugar, porque posee unas características eléctricas aceptables y la fuente se adapta al entorno del sistema de automatización diseñado. Esta fuente se adapta perfectamente al carril DIN donde se colocan los componentes de los S7-226. Sin embargo, y aunque en este diseño automatizado no supone ningún problema, en caso de detectar falta de espacio para colocar una fuente de este estilo, se podría recurrir a fuentes tipo SITOP SMART, puesto que estos modelos son bastante más reducidos de tamaño con unas características eléctricas similares.

Finalmente se han adquirido 2 Uds de la fuente SITOP POWER 3,5A referencia Siemens 6EP1 332-1SH31.

En la siguiente imagen puede verse esta fuente de alimentación:

SITOP POWER 3,5A

9.9. Módulo de dos salidas analógicas EM232

Este es el módulo apropiado de ampliación de salidas analógicas para el S7-200. Existen también módulos mixtos de salidas/entradas analógicas, pero en este proyecto no se van a utilizar entradas analógicas, por lo que lo lógico es adquirir módulos únicamente de salidas analógicas.

En este proyecto las salidas analógicas, en concreto de tensión, son fundamentales para el desarrollo del sistema de automatización. A través de las salidas analógicas se va a enviar la tensión apropiada a los mandos de control de los trenes del sistema ferroviario. Por lo tanto, gracias a la regulación de esta tensión en las salidas analógicas, se obtiene control total sobre la velocidad de los trenes.

El módulo de salidas analógicas seleccionado es el EM232, un módulo que posee dos salidas analógicas que se pueden conexionar como salidas en tensión o en intensidad, dependiendo de las necesidades del proyecto que se realice. En concreto, en este sistema de automatización se conexiona la salida en tensión, no siendo necesaria la de intensidad.

Uno sólo de estos módulos hubiese sido suficiente para regular la velocidad de los dos trenes que se tendrán recorriendo la maqueta simultáneamente, ya que cada módulo posee precisamente dos salidas analógicas. Sin embargo, se ha decidido adquirir dos de éstos módulos analógicos, disponiendo de cuatro

salidas analógicas en total (evaluando sólo los S7-226 utilizados). De esta forma, uno de los trenes será controlado por una salida analógica de un PLC y el otro tren será controlado por la salida analógica del otro PLC. Los motivos de esta decisión se han tomado en base a las siguientes consideraciones:

En una situación real (aunque esto no significa que realmente tenga que funcionar de esta forma), parece más lógico que cada tren tenga su propio sistema de control independiente, puesto que cada tren se puede considerar como una entidad independiente. Por lo tanto, lo lógico es que la regulación de velocidad de cada tren sea independiente y gobernada por un PLC dedicado a cada tren.

Por lo tanto, se ha tomado la decisión de adquirir 2 Uds del módulo de salidas analógicas EM232 con referencia Siemens 6ES7 232-0HB22-0XA0

EM232

9.10. Módulo PROFIBUS DP EM 277

Este es el módulo para conectar el S7-22x (en este caso, los S7-226) a la red PROFIBUS-DP, como esclavos en la red. También se puede utilizar para conectar los S7-22x a una red MPI, funcionando de igual forma como dispositivo esclavo.

Es capaz de proporcionar una velocidad de transmisión máxima de 12Mbits/s, y es aplicable en CPUs a partir de la versión 6ES7 2xx-xxx21-xxxx.

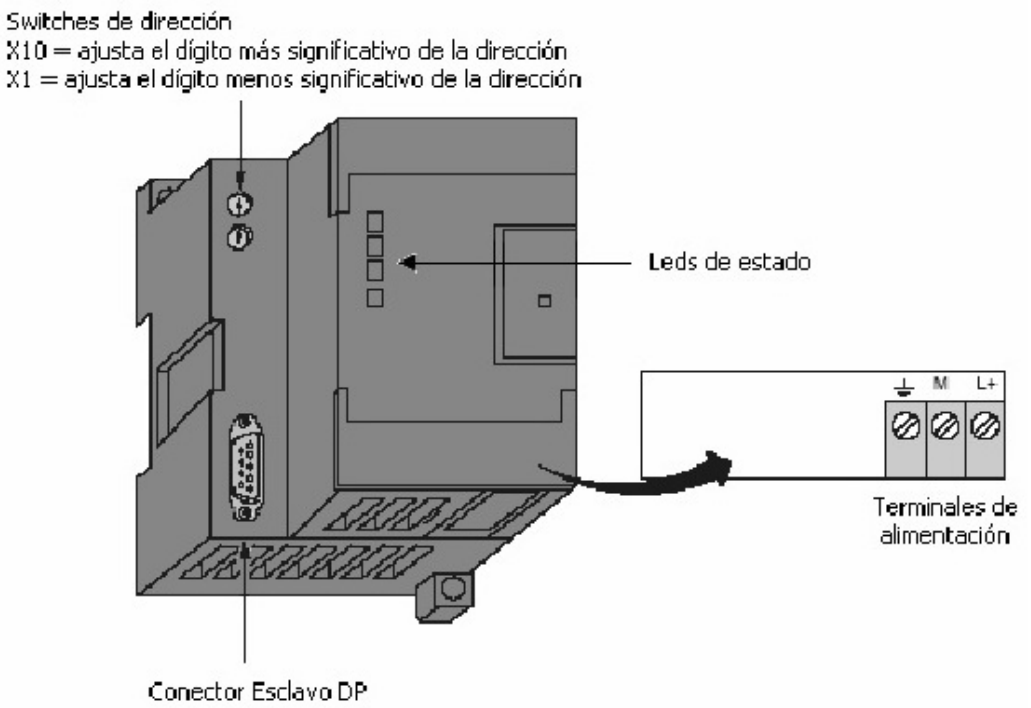
Se han adquirido dos módulos (cada CPU debe tener el suyo propio para que se pueda identificar en la red PROFIBUS), con referencia Siemens 6ES7 277-0AA22-0XA0.

Estos módulos, necesitan ser configurados para establecer una dirección válida en la red PROFIBUS con la que poder identificarse y gestionar la comunicación con el resto de la red PROFIBUS.

El puerto de comunicaciones se adapta al estándar RS-485. Es un conector del tipo 9-Pin Sub D I/O hembra.

PROFIBUS DP EM 277

La vista general de un módulo de este tipo es:



El módulo EM 277 cuenta con 4 leds de estado en el panel frontal, los cuales indican:

LED	OFF	ROJO	PARPADEO ROJO	VERDE
CPU FAULT	No fallo	Fallo interno	-	-
POWER	No 24VDC	-	-	Alimentación OK
DP ERROR	No error	Datos perdidos	Error parametrización	-
DX MODE	No hay intercambio de datos	-	-	En modo intercambio de datos

En la siguiente imagen se muestra uno de los dos módulos EM 277 utilizados en el proyecto:

9.11. Tarjeta de comunicaciones CP 5622

La tarjeta de comunicaciones CP 5622 permite conectar programadoras y PCs a PROFIBUS y a la interfaz multipunto MPI de SIMATIC S7. El principal requisito es que el PG o PC disponga de 1 slot PCI-EXPRESS libre.

CP 5622

9.12. Cartucho de memoria 256Kb para CPUs S7-22x

El módulo de memoria externo es una memoria conectable no volátil, en la que se guardan los módulos de programa, los módulos de datos, los módulos de datos del sistema, las recetas, las configuraciones de datos LOG y los valores forzados. Con el módulo de memoria se puede transferir un programa, los datos y los datos del sistema a una CPU del S7-200, sin utilizar un equipo de programación.

Con respecto a la adquisición del módulo de memoria hay dos opciones, un módulo de 64 Kbytes de capacidad u otro de 256 Kbytes de capacidad. En este caso, y siendo la diferencia de precio no relevante para el presupuesto económico del proyecto, se ha decidido la adquisición de dos módulos de memoria de 256 Kbytes de capacidad (cada una de las dos CPUs S7-226 del proyecto se equipará con un módulo de memoria). La referencia Siemens es 6ES7 291-8GH23-0XA0.

Cartucho de memoria 256Kb

En la siguiente imagen se muestra el cartucho de memoria insertado en uno de los S7-226 que se utiliza en el proyecto:

Cartucho de memoria instalado en CPU S7-226.

9.13. Cable USB/PPI para conexión de S7-200 a PC

Con este cable va a ser posible la conexión del PLC al PC, de tal forma que se pueda programar, transferir información, etc. Con la ayuda de este cable se establece una conexión entre el puerto USB del PC y el SIMATIC S7-200 o una red PPI (RS 485). Se puede utilizar como maestro en una red Multi-Master-PPI.

Para el proyecto se va a adquirir 1 Uds del cable USB/PPI, referencia Siemens 6ES7 901-3DB30-0XA0.

No es imprescindible adquirir un cable para cada autómata, puesto que el mismo cable puede intercambiarse entre las dos CPUs cuando se está programando. Por otra parte, este tipo de comunicación, en este proyecto en concreto, no se va a utilizar normalmente, por lo que sólo va a ser preciso para tareas de programación y diagnóstico. Por este motivo, no es preciso adquirir dos cables.

Cable USB/PPI para conexión de S7-200 a PC

9.14. Cable de bus PROFIBUS FAST CONNECT (2 hilos apantallado)

En este proyecto se han utilizado aproximadamente 10 metros de cable PROFIBUS FC Standard, ya que este tipo de cable permite una conexión rápida y no hay ningún requerimiento especial para utilizar otro tipo de cable más específico. La referencia Siemens es 6XV1 830-0EH10.

En la siguiente imagen se puede ver el cable PROFIBUS FC Standard:

Cable PROFIBUS FAST CONNECT

9.15. Conectores de bus PROFIBUS (Salida de cable a 90º y 180º)

El conector de bus se enchufa directamente en el interface PROFIBUS (conector hembra Sub-D de 9 polos) de la estación PROFIBUS o de un componente de red PROFIBUS. Sirve para conectar estaciones PROFIBUS al cable de bus PROFIBUS.

Tienen la ventaja de asegurar un montaje sencillo, ya que la conexión es de tipo desplazamiento de aislamiento.

En este proyecto se van a utilizar 3 conectores referencia Siemens 6ES7 972-0BB50-0XA0 con salida de cable a 90º, y 1 conector referencia Siemens 6GK1 500-0FC00 con salida de cable a 180º.

Los 3 conectores con salida a 90º se van a utilizar en la conexión PROFIBUS de los tres PLCs, con la posibilidad incluso de conectar el PC al mismo conector para tareas de mantenimiento, por ejemplo, si

fuera preciso. Para la conexión del PC (a través de la tarjeta interna), se ha preferido un conector con salida axial a 180º. En el caso del PC, es deseable que sea a 180º para una mayor comodidad de cableado y posición del PC en la infraestructura de la sala de control, además de no ser necesario que el conector PROFIBUS lleve un puerto adicional para conectar otro PC o PG.

En la siguiente imagen se muestran los dos tipos de conectores:

Conectores de bus PROFIBUS

9.16. Fast Connect Stripping Tool

El sistema de pelado Fast Connect permite conectar rápida y simplemente conectores PROFIBUS a los cables de bus PROFIBUS. La configuración especial de los cables de bus Fast Connect permite utilizar la herramienta peladora Fast Connect Stripping, con la cual se separa en una única operación la cubierta exterior y la pantalla de malla con las medidas exactas. Un cable así preparado se conecta a los conectores de bus Fast Connect por el sistema de desplazamiento de aislamiento.

La utilización de esta herramienta queda justificada por el trabajo manual que ahorra, la realización de un montaje sin posibilidad de errores y por un pelado de cable ajustado y preciso.

En la siguiente figura se muestra la herramienta de pelado de cable PROFIBUS Fast Connect:

Fast Connect Stripping Tool

9.17. Fuente de alimentación SITOP SMART 5A

Siguiendo con los requisitos del proyecto, se integrarán nuevos dispositivos, tales como sensores, semáforos, etc, que necesitarán de alimentación estable y fiable. Únicamente con las fuentes de alimentación de los PLCs no es aconsejable empezar a alimentar dispositivos externos, puesto que las fuentes de las CPUs pueden perder eficacia o rebasar su capacidad de salida máxima si no se tiene cuidado o no se calcula bien lo que se va a conectar a las mismas. Por ello es aconsejable disponer de fuentes de alimentación externas (o diferentes) a la de la CPU para alimentar cualquier dispositivo ajeno al autómata. Pensando en ello, se han sopesado varias opciones, seleccionando las fuentes SITOP SMART de Siemens. Esta gama de fuentes de alimentación tiene muy buenas cualidades eléctricas, entre ellas, la protección contra sobrecargas y la estabilización de la señal de salida, y ya van preparadas para ser integradas en carril DIN, al igual que todo el sistema de automatización. Un requisito imprescindible es que la salida sea a 24 VDC, puesto que los dispositivos que conectaremos a las fuentes precisaran de esa alimentación para su funcionamiento, además de ser el estándar de tensión de salida en

automatización. En cuanto a la intensidad, se ha sopesado la relación calidad/precio y se ha optado por adquirir dos fuentes de 5 A cada una, con seguridad plena de que va a ser suficiente para alimentar los dispositivos previstos. Por lo tanto, la decisión final ha sido adquirir dos fuentes de alimentación SITOP SMART 5A referencia Siemens 6EP1333-2AA01.

Fuente de alimentación SITOP SMART 5A

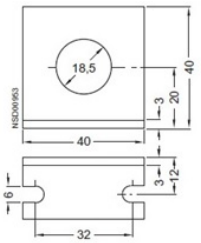
9.18. Detectores inductivos de proximidad SIMATIC PXI-200

La forma de controlar, en diferentes puntos del trayecto ferroviario, el paso de los trenes, requiere de la integración en el sistema automatizado de dispositivos que permitan “avisar” al autómata de que un tren pasa por un punto determinado.

Se decide adquirir 9 sensores inductivos con distancia de trabajo de 8 mm, referencia Siemens 3RG4023-0AG01.

Para la fijación de los sensores se necesita la escuadra de fijación de referencia Siemens 3RX7 301. Sirve para fijar detectores de proximidad con cuerpo cilíndrico M18. Está fabricado en acero galvanizado. Se deben adquirir una por sensor.

Detectores inductivos de proximidad



Escuadra de fijación

9.19. Sensores inductivos de proximidad Scheneider Electric

Se decide adquirir 6 sensores inductivos con distancia de trabajo de 8 mm, salida PNP, la referencia Scheneider Electric Siemens XS118B3PAL2. Estos sensores son más cortos que los anteriores y se pueden colocar en zonas del trayecto en las que haya menor espacio.

Para la fijación de los sensores se necesita el soporte de fijación de referencia Balluff 9027593.

Detectores inductivos de proximidad

Soporte de fijación

9.20. Sensor fotoeléctrico Sick sistema difuso led

Se decide adquirir 1 sensor fotoeléctrico con alcance de trabajo de 4 mm a 150 mm, salida PNP, la referencia Sick WTB4-3P1361. Este tipo de sensores permiten regular la distancia de alcance con un potenciómetro.

Detector fotoeléctrico sistema difuso

9.21. Sensor fotoeléctrico Sick sistema de supresión de fondo led

Se decide adquirir 1 sensor fotoeléctrico con sistema de supresión de fondo, con alcance de trabajo de 20 mm a 950 mm, salida PNP, la referencia Sick GTB10-P1212. Este tipo de sensores permiten regular la distancia de alcance con un potenciómetro.

Detector fotoeléctrico sistema de supresión de fondo

9.22. LEDS (Iluminación semáforos)

El tipo de iluminación seleccionada para los semáforos del trayecto ferroviario son diodos led. Los diodos led tienen un bajo consumo y una gran durabilidad. De hecho, este tipo de tecnología está siendo aplicada a los semáforos y todo tipo de señalización real en vías terrestres (ferroviaria, carreteras, etc). Unido a esto, los leds tienen un coste muy bajo, por lo que se hace rentable su compra. Además, debido a su tamaño pequeño, son fácilmente adaptables a la maqueta ferroviaria.

En concreto, y tomando como referencia semáforos ferroviarios reales, se han adquirido leds de color rojo y verde para componer la señalización de los semáforos.

Los leds de color verde han sido fabricados por la compañía LEDT, con referencia número L02R5000H1D1, se necesitan 18 Uds.

Los leds de color rojo han sido fabricados por la compañía LEDT, con referencia número L4RR5000H1D1, se necesitan 18 Uds.

Leds

9.23. Resistencias 1,1kΩ, 0,6W

Para no quemar ni dañar los leds, se necesitan unas resistencias. Esto se explica porque hay que conseguir disminuir la tensión de partida de las fuentes de alimentación utilizadas (que suministran tensión a 24VDC), a la tensión adecuada de funcionamiento del led, que debe ser lo más próxima posible a la V_F típica de sus especificaciones técnicas (frecuentemente unos 2,1 V). Esta tarea la realizarán las resistencias, que absorberán esa tensión de más y la disiparán en forma de calor.

En este caso se han seleccionado resistencias de 0,6 W, que serán suficientes para el trabajo que deben realizar en los semáforos.

Resistencias 1,1kΩ, 0,6W

9.24. Terminal de conexión para PCB

Es un terminal de conexión por tornillo para montaje en PCB con cuerpos apilables que permiten elaborar grandes sistemas de conexionado múltiple.

Los terminales son de bronce chapado en níquel con protectores de cable en níquel-plata en cuerpos de poliamida 6 reforzados de fibra de vidrio. Tiene un paso de 5 mm.

La opción estándar acepta cable de 2,5 mm2 y la opción de perfil bajo acepta cable de 1,5 mm2.

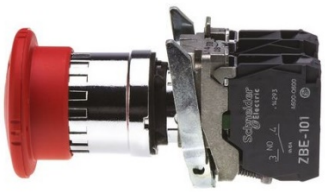
Para el proyecto se van a comprar 18 terminales de perfil bajo de 3 posiciones.

Terminal de conexión para PCB

9.25. "Seta" de emergencia

En cualquier proyecto de automatización es imprescindible un pulsador que permita parar el sistema en caso de emergencia. Los dispositivos típicos para esta tarea son los pulsadores en forma de seta de

emergencia. En este proyecto se va a utilizar un pulsador de seta de emergencia del fabricante Telemecanique con referencia XB4BS8445.

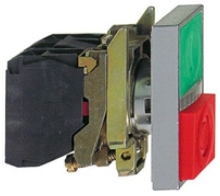


"Seta" de emergencia

9.26. Pulsador luminoso doble

En el proyecto se van a integrar 3 pulsadores luminosos dobles con piloto led integrado. Uno de ellos se va a utilizar para arrancar o parar el sistema de automatización. Los otros dos pulsadores se utilizarán para activar o desactivar los dos trenes, de forma que cuando el pulsador de un tren esté en off, ese tren no se ponga en marcha, aunque se incremente la velocidad (simulando al encendido real de un tren).

El pulsador integra bloque de contactos (1 NA + 1 NC). El fabricante es Telemecanique y su referencia es XB4BW84B5.



Pulsador luminoso doble

9.27. Interruptor de conmutación de 2 posiciones

En el proyecto se va a integrar un interruptor de conmutación de 2 posiciones, que responderán a establecer el sistema en modo semi-automático o modo automático. El modelo seleccionado ha sido Telemecanique referencia K1B011UCH. Es un interruptor compuesto por cuerpo de montaje frontal y un cabezal de fijación de 22 mm de Ø con una palanca de 34 mm de longitud y un escudete de 45x45 mm marcado para el funcionamiento del interruptor.

Interruptor de conmutación de 2 posiciones

9.28. Indicadores de estación de control led

Para indicar el estado de funcionamiento del sistema automatizado, se necesitan unos indicadores luminosos.

Se emplean:

- 1 Uds indicador de estación de control, led, de color verde. El proveedor es Schneider Electric y la referencia XB4BVB3.
- 1 Uds indicador de estación de control, led, de color amarillo. El proveedor es Schneider Electric y la referencia XB4BVB5.

- 1 Uds. indicador de estación de control, led, de color rojo. El proveedor es Schneider Electric y la referencia XB4BVB4.

9.29. Relé electrónico

En el proyecto se va a hacer uso de relés electrónicos con el objetivo de accionar los cambios de aguja de las vías o desvíos del trayecto ferroviario en un sentido o en el otro. Puesto que en el total del trayecto ferroviario hay distribuidos 6 cambios de aguja, se necesitarán 6 relés para cubrir el funcionamiento de todos ellos.

El relé seleccionando es del fabricante Finder, el modelo es el 40.52.7024.0000.

La selección de este modelo de relé y fabricante se debe principalmente a la disposición de los contactos que tiene, ya que es adecuada para accionar el desvío en un sentido o en otro (posee contactos NA y NC, y que se adapta muy bien a su distribución en carril DIN de 35 mm (concretamente se pueden distribuir los relés en una sección de carril de idénticas dimensiones al carril utilizado en la distribución de los autómatas S7-200, por lo que se puede realizar una instalación optimizada de cableado y simplificación de conexiones). Se ha seleccionado una tensión de bobina de 24 VDC, que es la tensión de salida de las fuentes de alimentación que se van a usar en el proyecto.

Relé

Zócalo para el relé (referencia 95.05)

Zócalo para el relé

Un accesorio de mayor importancia es el diodo de protección CEM tipo 99.02. La desactivación del relé provoca una corriente de descarga en la bobina en sentido inverso que pone en peligro el elemento electrónico utilizado para su activación. En este caso, se protege al transistor de la salida digital del autómata, que es quien ordena la activación y desactivación del relé. Colocando un diodo polarizado inversamente cortocircuita esa corriente y elimina el problema. El inconveniente es que la descarga de la bobina es más lenta, por lo que la frecuencia a la que puede ser activado el relé es más baja. A este tipo de diodo se le llama comúnmente diodo volante.

Para el proyecto se han seleccionado 6 diodos de protección que se colocarán en el zócalo de conexión de los relés. Los diodos son específicos para este modelo de relé y zócalo.

Diodo

Otro accesorio que se va a utilizar es un puente de 8 terminales para el zócalo (ref: 095.18). Este accesorio no es imprescindible, pero ayuda bastante simplificando el cableado. Gracias a este peine de conexión, sólo hace falta llevar un cable de alimentación a uno de los bornes y automáticamente ya se tiene con la misma señal el resto de bornes. Por ejemplo, en los 6 relés que se van a utilizar, se va a simplificar la conexión de masa en la alimentación de la bobina de cada uno de ellos, realizando esta conexión a través del peine.

Peine

A continuación, se exponen las características técnicas más importantes de este modelo de relé

Características

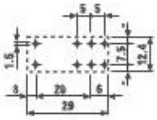
Relé con 1 o 2 contactos
40.31 - 1 contacto 10 A (pas 3.5 mm)
40.51 - 1 contacto 10 A (pas 5 mm)
40.52 - 2 contactos 8 A (pas 5 mm)

Montaje en circuito impreso
- directo o en zócalo
Montaje en carril de 35 mm (EN 60715)
- en zócalos con bornes a pletina o de conexión rápida

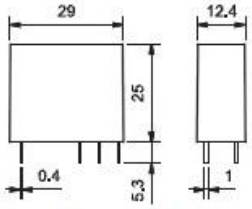
- Bobina DC (estándar o sensible) y bobina AC
- Contactos sin Cadmio
- 8 mm, 6 kV (1.2/50 µs) entre bobina y contactos
- UL Listing (combinaciones relé/zócalo)
- Estanco al flux: RT II estándar, (disponible en versión RT III)
- Zócalos serie 95
- Modulos de señalización y protección CEM
- Modulos temporizados serie 86

40.52

- Reticulado 5 mm
- 2 contactos 8 A
- Montaje en circuito impreso o en zócalo serie 95



Vista parte inferior



Para Cargas de Motores y "Pilot Duty" Homologadas por UL ver "Información Técnica General" página V

Características de los contactos		
Configuración de contactos		2 contactos conmutados
Corriente nominal/Máx. corriente instantánea	A	8/15
Tensión nominal/Máx. tensión de conmutación	V AC	250/400
Carga nominal en AC1	VA	2000
Carga nominal en AC15 (230 V AC)	VA	400
Motor monofásico (230 V AC)	kW	0.3
Capacidad de ruptura en DC1: 30/110/220	VA	8/0.3/0.12
Carga mínima conmutable	mW (V/mA)	300 (5/5)
Material estándar de los contactos		AgNi
Características de la bobina		
Tensión nominal	V AC (50/60 Hz)	6 - 12 - 24 - 48 - 60 - 110 - 120 - 230 - 240
de alimentación (U_N)	V DC	5 - 6 - 7 - 9 - 12 - 14 - 18 - 21 - 24 - 28 - 36 - 48 - 60 - 90 - 110 - 125
Potencia nominal en AC/DC/DC sens.	VA (50 Hz)/W/W	1.2/0.65/0.5
Campo de funcionamiento	AC	$(0.8...1.1)U_N$
	DC/DC sensible	$(0.73...1.5)U_N/(0.73...1.75)U_N$
Tensión de mantenimiento	AC/DC	0.8 U_N /0.4 U_N
Tensión de desconexión	AC/DC	0.2 U_N /0.1 U_N
Características generales		
Vida útil mecánica AC/DC	ciclos	$10 \cdot 10^6/20 \cdot 10^6$
Vida útil eléctrica con carga nominal AC1	ciclos	$100 \cdot 10^3$
Tiempo de respuesta: conexión/desconexión	ms	7/3 - (12/4 sensible)
Aislamiento entre bobina y contactos (1.2/50 µs)	kV	6 (8 mm)
Rigidez dieléctrica entre contactos abiertos	V AC	1000
Temperatura ambiente	°C	–40...+85
Categoría de protección		RT II**
Homologaciones (según los tipos)		RINA

En cuanto al zócalo para el relé (referencia 95.05), las características técnicas son las siguientes:

95.05

Homologaciones (según los tipos):

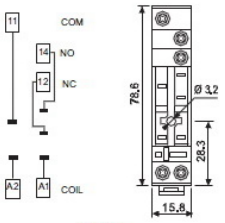
Combinación relé/zócalo

095.01

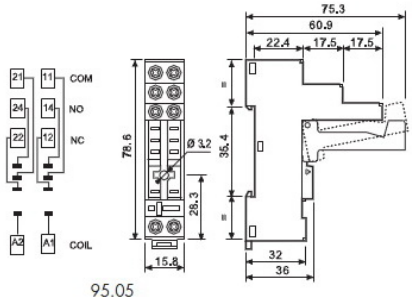
060.72

Zócalo con bornes de jaula montaje en panel o carril 35 mm (EN 60715)		**95.03 Azul**	**95.03.0 Negro**	**95.05 Azul**	**95.05.0 Negro**
Tipo de relé		40.31		40.51, 40.52, 40.61	
Accesorios					
Brida de retención metálica		095.71			
Palanca de retención y extracción de plástico (suministrada con el zócalo - código de embalaje SPA)		095.01	095.01.0	095.01	095.01.0
Puente de 8 terminales		095.18	095.18.0	095.18	095.18.0
Etiqueta de identificación		095.00.4			
Modulos (ver tabla abajo)		99.02			
Modulos temporizados (ver tabla abajo)		86.30			
Juego de etiquetas de identificación para palanca de retención y extracción de plástico 095.01, 72 unidades, 6x12 mm		060.72			
Características generales					
Valor nominal		10 A - 250 V *			
Rigidez dieléctrica		6 kV (1.2/50 µs) entre bobina y contactos			
Grado de protección		IP 20			
Temperatura ambiente	°C	–40...+70			
Par de apriete	Nm	0.5			
Largo de pelado del cable	mm	8			
Capacidad de conexión de los bornes para zócalos 95.03 y 95.05		hilo rígido		hilo flexible	
	mm²	1x6 / 2x2.5		1x4 / 2x2.5	
	AWG	1x10 / 2x14		1x12 / 2x14	

* Con corrientes >10 A, los bornes de los contactos deben conectarse en paralelo (21 con 11, 24 con 14, 22 con 12). Con relés 40.51 utilizar los bornes 21, 12 y 14.

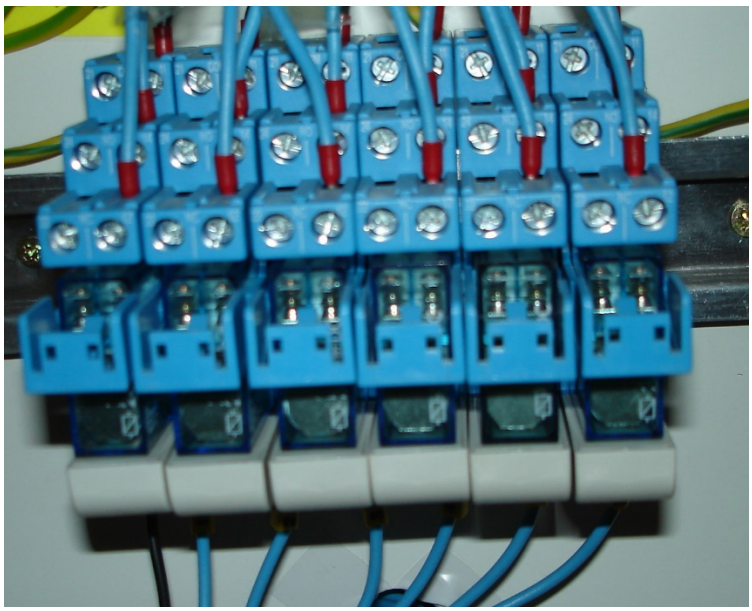


95.03



95.05

El conjunto de relé, soporte, y diodo queda de la siguiente forma en el proyecto:



9.30. Material diverso

En este grupo se engloban el cable eléctrico, punteras, terminales de conexión (fichas), herramienta necesaria a nivel eléctrico, material necesario para construir la maqueta (tableros de madera, tubo de hierro, etc), herramienta necesaria para la construcción de la maqueta (radial de corte, soldadora, etc). Es decir, todo el material no específico del mundo ferroviario o del sistema de automatización, y herramientas utilizadas en la realización del montaje e instalación del sistema.

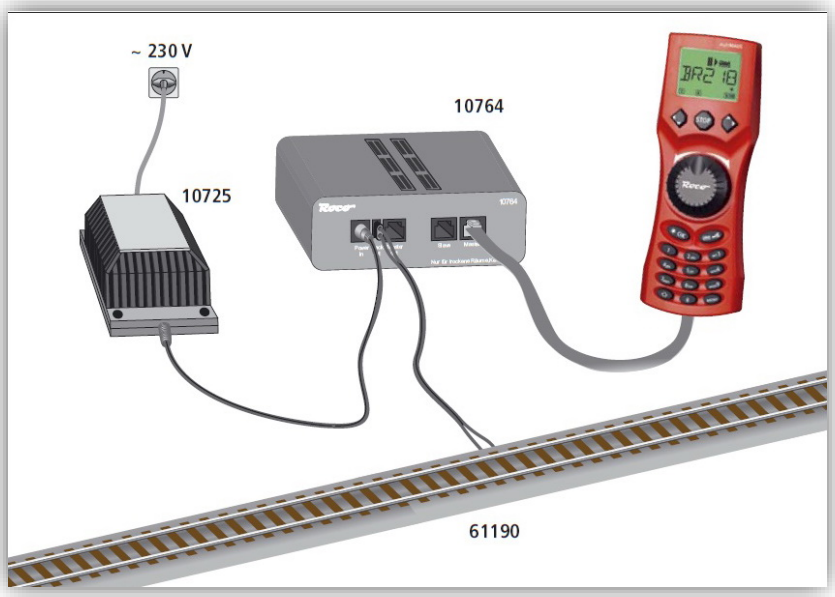
El cable eléctrico que se va a utilizar en el conexionado del sistema de automatización es de color azul y negro, además de cable de tierra en su color característico. Para la realización de las conexiones del sistema de automatización se va a emplear cable de 1 mm^2 fundamentalmente, y de 0,5 mm^2 cuando sea necesario para simplificar las conexiones por el aglomeramiento de cableado. La terminación del cable en los autómatas y en los terminales se realizará con la ayuda de punteras, para evitar roturas de los filamentos del cable y que de esta forma queden las conexiones reforzadas y limpias.

Cables y punteras

10. Montaje de los dispositivos de señalización de la maqueta (semáforos, desvíos, sensores).

10.1. Acciones en el material ferroviario

El material ferroviario utilizado en el proyecto ya ha sido presentado anteriormente en esta documentación. Como ya se ha comentado, el sistema de control utilizado por el fabricante es DCC (Digital Command Control). Gráficamente, el sistema de control propuesto por el fabricante en su kit de modelismo ferroviario es el siguiente (que es el modelo utilizado para este proyecto):



En este sistema, la central (o amplificador) digital (referencia del fabricante 10764) recibe las señales de dos partes diferentes. Por una parte, el mando, en este caso, el MultiMaus de Roco (referencia del fabricante 10810), genera los datos de control para las locomotoras y los accesorios como desvíos, semáforos, etc, y las envía a través del cable plano de datos acorde con el protocolo estándar de control DCC. Por otra parte, el transformador (referencia del fabricante 10725), que será conectado directamente a la tensión de red (típicamente 230V AC), convertirá ésta en 16V AC a su salida, que será la tensión de alimentación que enviará a la central o amplificador digital. La tarea del amplificador es mezclar los dos tipos de señales recibidas y enviarlas a la vía en forma de señal eléctrica de alimentación. Es decir, cogerá la forma de onda típica sinusoidal correspondiente a la tensión alterna y la transformará en un tipo de señal de forma de onda cuadrada. La explicación es que una forma de onda cuadrada se puede modular en frecuencia, por lo que el protocolo DCC provee normas en las que se dicta que los datos provenientes del mando de control sean introducidos en la señal eléctrica modulando la misma en una frecuencia determinada. De esta forma, las distintas modulaciones podrán ser interpretadas por un dispositivo que llevarán todos los componentes que se deseen controlar (locomotoras, desvíos, semáforos, etc). Estos decodificadores, y mediante datos introducidos en los paquetes de información enviada en esa señal modulada reconocerán si esa información les concierne (a través de direcciones de dispositivo determinadas) o no, y si les concierne, decodificarán la información y responderán en consecuencia (más o menos velocidad para la locomotora, encendido o apagado de luces de la locomotora, cambio de posición de los desvíos, etc). Básicamente este es el sistema propuesto por Roco. En el caso de este proyecto, hay que añadir un segundo mando de control, que actuará de slave del principal, conectándose al puerto slave de la central digital. En esta estructura, cada mando de control se hará cargo de controlar un tren. Sólo habrá que tener en cuenta que para que el slave funcione, es necesario que el master esté conectado también.

En el proyecto, lo que se pretende es automatizar el control de velocidad de las locomotoras, sustituyendo a la ruleta de giro que integra cada mando de control.

Mando de control giratorio regulador de velocidad (eliminada su funcionalidad en el proyecto)

Cada locomotora tendrá una dirección de dispositivo determinada, resultado de programar la CV1 del decodificador. Se puede seguir el manual de instrucciones incluido en la sección de anexos para realizar los pasos necesarios en esta programación. Allí se verá también las instrucciones para programar un registro de la locomotora en un archivo incorporado como una especie de base de datos en el mando. El requisito principal es que la dirección del decodificador físico de la locomotora programado con la CV1 coincida con la dirección registrada en la base de datos del mando de control, ya que hay una relación directa entre el registro de cada locomotora en la base de datos y la dirección física del decodificador de la misma. No obstante, se muestra a continuación un extracto del manual citado en el que se programa una locomotora por primera vez:

FUNCIONAMIENTO DEL *multi*MAUS

E

A pesar de sus muchas posibilidades, el ***multi*MAUS** tiene un manejo fácil e intuitivo de acuerdo con el concepto ya introducido por ROCO en los Lokmaus de primera y segunda generación. Ahora veremos el funcionamiento del ***multi*MAUS** mediante varios ejemplos.

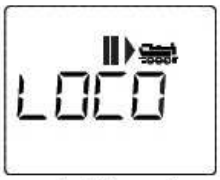
En la página 28 dentro de "CONSEJOS, TRUCOS Y AYUDA BÁSICA" podrá encontrar las soluciones a los pequeños problemas que pueden presentarse durante el funcionamiento del ***multi*MAUS**.

1. Puesta en marcha

Después de conectar la alimentación la pantalla del ***multi*MAUS** muestra la palabra "***multi*MAUS**". Seguidamente entra en el modo Locomotora y muestra la dirección de la primera locomotora.

1.1. La primera vez que se utiliza

Cuando conectamos por primera vez el ***multi*MAUS**, solamente está memorizada por defecto una sola locomotora, (la que tiene la dirección 3). En la pantalla se muestra el símbolo de locomotora y el símbolo de parada "II", (es decir la locomotora no se mueve) junto con las últimas funciones utilizadas y la palabra "LOCO" pudiendo conducirla de forma inmediata.



Si el ***multi*MAUS** forma parte de un conjunto de iniciación, la locomotora de dicho conjunto está ya completamente programada en el mando, por lo que podemos pasar directamente a conducirla (figura 3).

1.2. Las veces sucesivas (ya se ha creado el Archivo)

Haben Sie die ***multi*MAUS** schon in Betrieb gehabt, wird nach dem Einschalten immer die zuletzt gesteuerte Lok im jeweiligen Modus – Bibliothek oder Lokadresse – angezeigt.

2. Activación de las locomotoras

El ***multi*MAUS** puede acceder a las locomotoras de 2 maneras diferentes:
– desde la página del modo Archivo: ver apartado 2.1.
– desde el modo Direcciones, entrando directamente la dirección: ver apartado 2.2.
Podemos cambiar entre estos dos modos pulsando simultáneamente las teclas "Shift" y "Loco/ Desvíos".

2.1. El modo "Archivo"

El archivo de locomotoras, es una base de datos que ofrece la posibilidad de memorizar las direcciones de hasta un máximo de 64 locomotoras distintas, junto con sus nombres correspondientes (5 caracteres) y los pasos de velocidad con los que trabaja. Estos datos solo se guardan en el ***multi*MAUS**, no se guardan en el descodificador de la locomotora.

⚠ Siempre se puede modificar la dirección del descodificador mediante la CV 1, (ver el aparatado programación rápida de la página 13). Esta modificación no se traslada automáticamente al archivo, debe ser realizada manualmente.

La pantalla (en el ejemplo la locomotora "S 3/6"):
– el nombre de la locomotora (aquí "S 3/6") y el símbolo de locomotora
– La dirección de marcha (detenida desde marcha hacia adelante)
– Luces direccionales (luces activadas)
– F1 y F4 (las funciones activas de la locomotora).

(E)

Recuperación de las locomotoras guardadas en el archivo ("páginas del archivo")

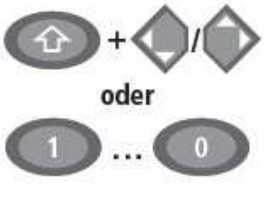
Las locomotoras quedan en el archivo en el mismo orden en que se han entrado. De todas formas este orden puede ser modificado:

locomotora seleccionada → + o +

Estas combinaciones con las teclas (pulsadas simultáneamente) desplazan la locomotora una posición hacia arriba o hacia abajo dentro del archivo. Podemos repetir esta operación tantas veces como queramos. Cuando mediante las "teclas de flechas" llegamos al final del archivo en la pantalla se nos preguntara si queremos entrar una nueva locomotora, veamos:

Añadir una locomotora al archivo (en el ejemplo la locomotora diesel DR-120 "Taiga")

Entrada	*Pantalla*	*Observaciones*
/	NUEV'?	Con "¿NUEVA?" se nos pregunta si queremos añadir una locomotora, recuerde que podemos desplazarnos por las diversas páginas del archivo mediante las "teclas de flechas". Confirmaremos pulsando "OK".
OK	ABC	Ahora entraremos el nombre de la locomotora. Dicho nombre no debe sobrepasar los 5 caracteres. En el ejemplo escribimos "Taiga". La posición actual del cursor se muestra mediante una pequeña barra horizontal que luce de forma intermitente.
1 … 0	TAI ABC	Los caracteres los escribimos con la ayuda de las teclas numéricas, que en este caso trabajan como en un teléfono móvil: tan pronto como pulsamos una tecla aparece en la pantalla el carácter correspondiente. El cursor se ilumina y después hace una pausa antes de desplazarse a la posición siguiente.
1 … 0	TAIGA ABC	La tecla "0" introduce un espacio cada vez que la pulsamos. Los caracteres especiales (/, –, \,* , [,], ‹, ›) están en la tecla "1". Para corregir una entrada errónea basta con pulsar la tecla "flecha izquierda" que nos desplazará una posición hacia la izquierda cada vez.
OK	L 3	Confirmaremos con "OK" El *multi*MAUS cambia después de esto a la entrada de la dirección de la locomotora, se muestra la letra "L" al principio. Aquí se nos sugiere el valor "3 " de forma intermitente.



Cambiaremos el valor sugerido por el que nos interese, (que será la dirección del descodificador de la locomotora) mediante la tecla "Shift" y una de las "teclas de flechas", o bien escribiendo directamente su valor con las teclas numéricas.

Este valor solo se memoriza en el Archivo. Si todavía no lo hemos hecho deberemos programarla en la CV1 del descodificador, vea el apartado 6 de la página 13.

Confirmaremos con "OK"

La selección de los pasos de →velocidad, (en la pantalla se muestran las letras "FS"), se realiza con las "teclas de flechas". Se debe seleccionar un valor de entre los 3 disponibles: 14, 28 o 128 pasos de velocidad. El valor seleccionado por defecto es el de 28 pasos. En el menú "REGLAJES" podemos cambiar este valor por defecto.

Si pasa a 128 pasos de velocidad, conseguirá regular la velocidad de la locomotora de una forma muy precisa. Todos los descodificadores modernos admiten este número de pasos de velocidad.

Si no desea realizar ninguna modificación basta con que pulse "OK".

La última pulsación de la tecla "OK" guardará los datos de la locomotora en el archivo. Compruebe que la dirección de la locomotora guardada en el archivo es la misma que la del descodificador de la locomotora que queremos. Si no es así modifique el valor guardado o reprograme la dirección del descodificador, encontrará la forma de hacerlo en el apartado 6 de la página 13.

Ahora ya puede conducir normalmente la locomotora.

2.2. El modo de Entrada de Direcciones

El *multi*MAUS también tiene la posibilidad de reconocer las locomotoras por la dirección de su descodificador. La pantalla muestra el sentido de marcha, la dirección precedida por la letra "L", (en el ejemplo la 36), el símbolo de locomotora y las funciones activas.

Podemos seleccionar otra dirección de locomotora de dos maneras distintas:

– mediante las "teclas de flecha"

– o entrando directamente su número mediante las teclas numéricas después de haber pulsado simultáneamente las teclas "Shift" y "Luces/OK".

Podemos también modificar los pasos de velocidad desde el menú "LOCOMOTORA" > "MODIFICAR", (ver en la segunda parte "Funciones de los Menús"), o mediante la tecla "MENU" junto con una de las "teclas de flecha". Para salir del modo de "Entrada de Direcciones" pulsaremos la tecla "STOP".

Encontrará más información sobre el empleo de las "teclas de flecha" en el apartado "Búsqueda rápida" que figura en el glosario de la tercera parte. Si mantenemos presionada la tecla de flechas se nos mostrará durante breves instantes cada dirección antes de pasar a la siguiente.

Encontrará la manera de programar una nueva dirección en el descodificador de la locomotora, mediante la CV1, en el apartado 6 de la página 13.

En el sistema ferroviario habrá dos locomotoras en funcionamiento. Una de ellas se identificará como locomotora de pasajeros, puesto que en el kit de Roco venía con vagones de pasajeros, y la otra se

identificará como tren de mercancías, ya que en el kit de Roco venía con vagones de mercancías. La locomotora de pasajeros ha sido programada en el canal Master de la central (o amplificador) digital con la dirección de decodificador 3, mientras que la locomotora de mercancías se ha programado en el canal Slave con la dirección de decodificador 4. Igualmente se han introducido en la base de datos de archivo que posee el mando con los nombres “PASAJE” y “MERCAN” respectivamente.

Otra de las modificaciones que se ha hecho en la programación de las locomotoras es la regulación de los pasos de velocidad. Concretamente, se han programado 128 pasos (mediante programación de la CV29, bit 1=1). Esto permitirá un control más suave en los incrementos de velocidad, aumentando la capacidad de control de la misma y posibilitando que los trenes no “funcionen a tirones”.

El resto de los valores de programación de las CV se ha dejado con los valores de fábrica, ya que no tienen una repercusión importante para el proyecto.

Hasta aquí ya está el sistema montado con las especificaciones del fabricante y preparado para funcionar. En este punto es en el que se deben hacer modificaciones para poder automatizar de forma externa los trenes y complementos de la maqueta ferroviaria sin utilizar para ello el sistema del fabricante.

El protocolo DCC de comunicaciones se debe mantener activo, puesto que, si no se hiciese esto, habría que tomar dos soluciones:

- Desarrollar un protocolo de comunicaciones propio (similar a DCC) para generar órdenes de control propias. Esto excedería en gran medida el objetivo de este proyecto, teniendo en cuenta que implicaría crear una central digital propia que obedeciese al nuevo protocolo de comunicaciones generado. Por otra parte, no tendría sentido crear prácticamente un “clon” de un sistema que se considera un estándar desde hace varios años y en el que trabaja multitud de gente para mejorarlo constantemente.
- Otra solución sería eliminar el decodificador de las locomotoras, lo que implicaría una vuelta al sistema analógico que había existido hasta que el sistema digital lo sustituyó. El sistema analógico tiene varios problemas, entre los que se incluyen como más importantes la posibilidad de generar cortocircuitos en bucles de retorno de vías o la imposibilidad de manejar dos o más locomotoras en el trayecto con velocidades independientes cada una de ellas. El sistema de control analógico no se basa en una modulación de la señal eléctrica que se envía a la vía, sino en la regulación de la tensión enviada a la vía, y, en función de esa tensión, el motor de la locomotora hará que vaya más deprisa o más despacio. Esta solución tampoco es viable para conseguir los objetivos del proyecto, ya que no se puede controlar las locomotoras de forma independiente y, en vez de conseguir mejoras en el sistema del fabricante, se empeoraría.

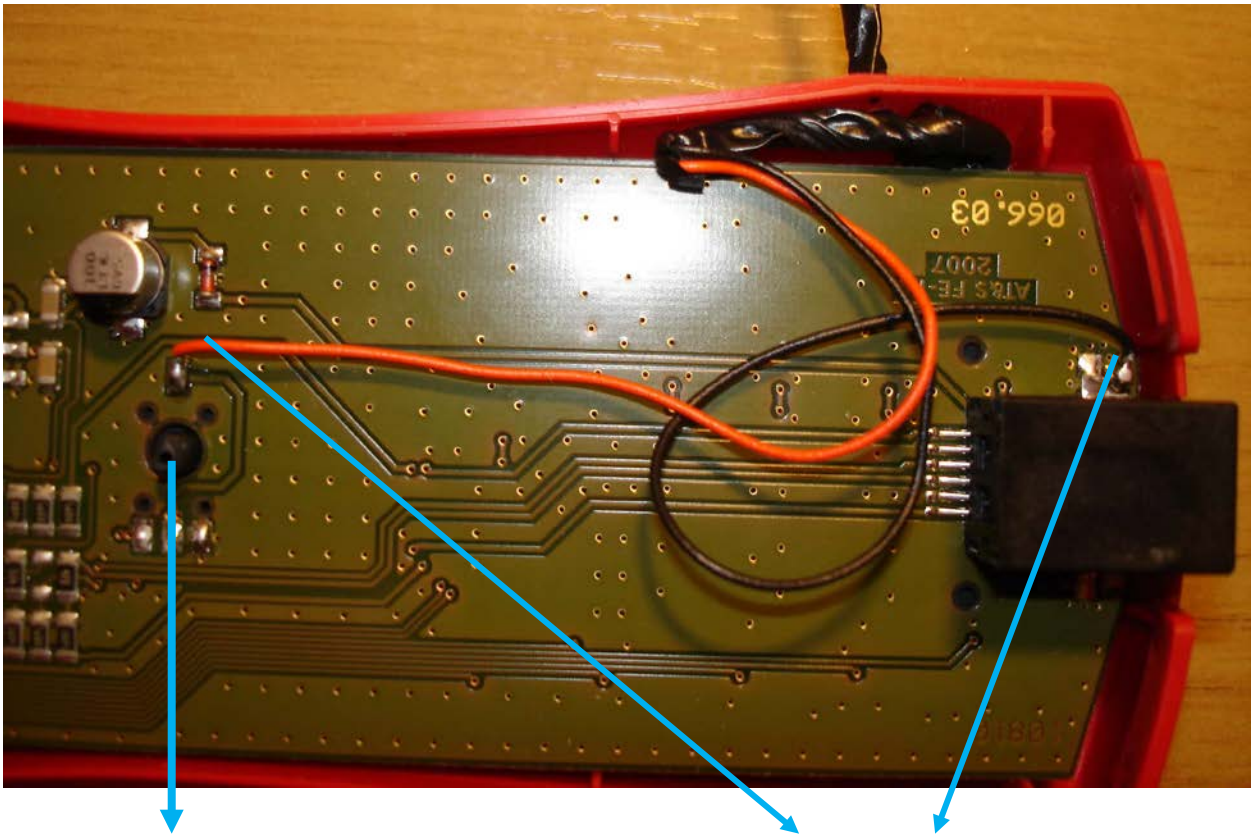
Por estos motivos, el protocolo DCC va a seguir activo y se va a hacer uso de él. La central digital va a recibir el mismo tipo de señal DCC de su sistema. El cambio que se propone en este proyecto es modificar el mando de control, de tal forma que no siga las órdenes de la ruleta de regulación de velocidad de su sistema. La regulación de velocidad se realizará utilizando la señal de salida en tensión de las salidas analógicas de los autómatas, procedimiento que se verá más adelante.

Para ello, ha sido necesario eliminar de los mandos de control el dispositivo que integran para regular la velocidad. El mando de control de Roco incluye un potenciómetro, que es el dispositivo electrónico que realmente se regula cuando el usuario gira en un sentido u otro el mando giratorio. Destacar que el potenciómetro que el fabricante incluye en su mando es de tipo sinfín, por lo que gira 360º sin tope de ningún tipo (el tope del giro en ambos sentidos va incorporado en lo que es el mando giratorio de plástico). La posición superior del mando giratorio significa tren parado o velocidad 0, uno de los lados de giro es para un sentido de la marcha y el otro lado de giro es para el otro sentido de la marcha. El tren adquirirá más velocidad conforme se mueva el mando giratorio hacia abajo (quedando el marcador que lleva dibujado en la parte superficial más alejada de la parte superior), e irá más despacio cuando el marcador dibujado en el mando giratorio se encuentre más próximo a la parte superior. Para asegurar que se consigue una velocidad 0, el mando giratorio de plástico también tiene unos topecitos en los que se

queda encajado un saliente que va girando conforme el usuario gira el mando, y que tiene su punto de encaje cuando el marcador dibujado en el mando giratorio está justo en la parte superior del mismo. Esto ofrece una pequeña resistencia al giro del mando con el objetivo de que cuando se quiera parar el o los trenes, sepamos cuando el mando está en la posición correcta para ello.

El mando de control debe ser modificado si se quiere regular la velocidad con las salidas analógicas de los autómatas, porque no es factible que la central digital atienda a dos órdenes de mando de velocidad diferentes. Para solucionar este problema, se ha eliminado el potenciómetro incluido por el fabricante en cada mando de control. Para ello, hay que desmontar el mando quitando un tornillo de fijación que lleva en la parte trasera y abriendo el mando, quedando dos partes (la de abajo, que integra la placa electrónica del mando, y la de arriba que integra el plástico transparente de la pantalla y los botones del mando).

Con la ayuda de una fuente de alimentación, un osciloscopio y un multímetro digital ha sido posible la identificación de la señal correspondiente al potenciómetro, haciendo diversas pruebas para identificar los terminales de alimentación y masa correspondientes. Después, se ha desoldado el potenciómetro de la placa electrónica, y, en su lugar, se han soldado dos cables para sustituir esa señal por las órdenes de mando de la salida analógica del autómata. En la siguiente imagen se puede ver uno de los dos mandos desmontado y con la modificación realizada:



Lugar donde estaba soldado el potenciómetro.

Sustitución de la señal de salida del potenciómetro por las salidas analógicas de los autómatas de control.

Haciendo diversas pruebas se ha determinado que un tren se encuentra parado cuando el voltaje suministrado por la salida analógica es de 1,50 V aproximadamente. El rango de tensión suministrado por la salida analógica de 0 V a 1,49V es el rango de velocidades en un sentido de la marcha del tren, y el rango de voltaje de 1,51V a 3,2 V aproximadamente es el rango de velocidades del otro sentido de marcha del tren. Ha sido necesario ajustar un poco más el rango de voltaje que las salidas analógicas

suministran a la central digital, puesto que el trazado diseñado en la maqueta ferroviaria incluye curvas y no es lo suficientemente grande como para que los trenes funcionen con seguridad en el trayecto, existiendo peligro de descarrilamiento, etc. El rango de tensión de las salidas analógicas se ha ajustado a un rango de 1,03V a 1,49 V en un sentido de la marcha y a un rango de 1,51 V a 2,78 V en el otro sentido de la marcha, de esta forma, aún con la tensión máxima suministrada, los trenes funcionan a una velocidad segura para su circulación por la maqueta.

De esta forma ya se controla la velocidad de las locomotoras desde los autómatas.

En concreto, la velocidad de la locomotora de pasajeros es controlada por el S7-226 que controla el entorno de la estación 1, mientras que la velocidad de la locomotora de mercancías es controlada por el S7-226 que controla el entorno de la estación 2, por supuesto, a través de sus salidas analógicas. Éstas a su vez, serán controladas y monitorizadas por el centro de control ferroviario (representado en la instalación mediante el S7 314C-2DP). A su vez, el Jefe de Control ha de ser capaz de monitorizar visualmente la potencia infringida a la locomotora y el valor de tensión equivalente suministrado por la salida analógica del autómata, teniendo pleno control de su velocidad. Esto sólo lo podrá realizar cuando el sistema se encuentre en modo manual-libre, ya que, si el sistema se encuentra en modo manual o automático, serán los autómatas (mediante programación), quienes controlarán la velocidad de los trenes.

A continuación, se van a explicar las modificaciones realizadas en los cambios de aguja o desvíos para poder realizar su control. En principio, los kits de inicio de modelismo ferroviario de Roco incluyen un mecanismo manual de accionamiento del desvío o cambio de aguja. La alternativa eléctrica del fabricante consta de un accesorio compuesto por dos bobinas eléctricas, accionadas a través de la conexión de un decodificador especial diseñado por el fabricante (comúnmente denominado decodificador de accesorios). Esto no es sino la repetición del sistema utilizado en las locomotoras, pero ahora para los desvíos. De hecho, los mandos MultiMaus de Roco llevan un botón de función para cambiar de función control de locomotora a función control desvíos. El funcionamiento del decodificador del desvío es bastante parecido al decodificador de la locomotora, obviamente con tareas diferentes. En la siguiente imagen se puede ver un accionamiento de vía universal:

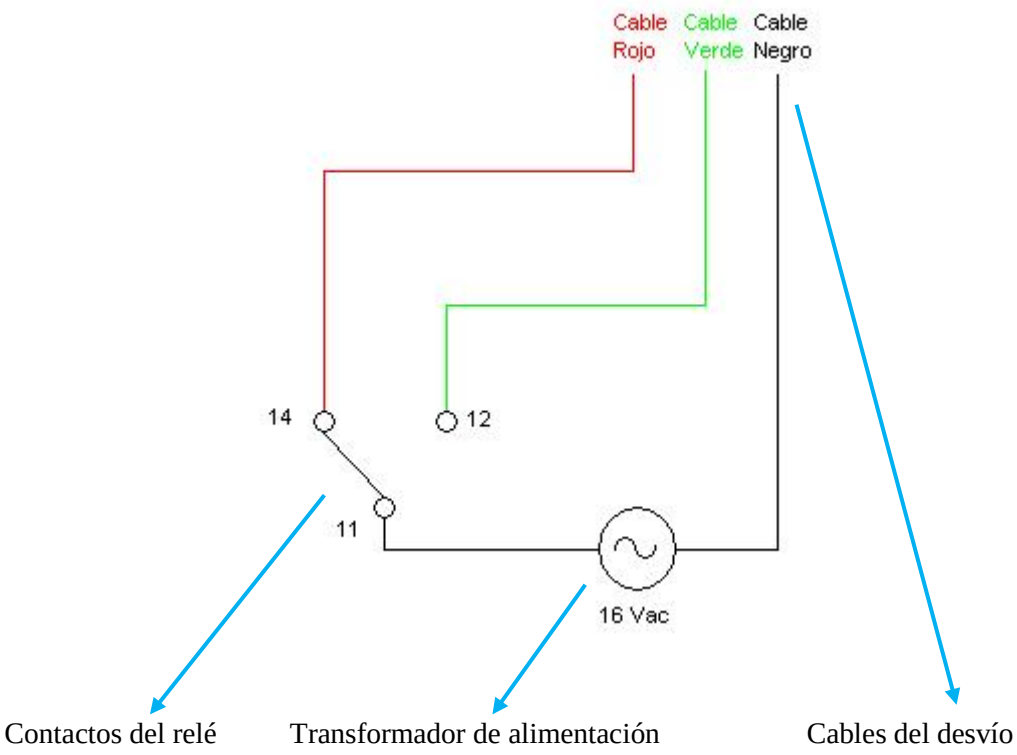
La función del cambio eléctrico ordenado por el decodificador hará que la varilla de la parte delantera del dispositivo se mueva. Esta varilla pasará por un paso especial del interior de la vía, y su movimiento producirá el cambio de aguja de vía visto desde la parte exterior de la misma.

En este caso, si lo que se pretende es automatizar externamente este cambio de aguja, se debe eliminar el decodificador que lo controla, si es que lo lleva ya incorporado. No es el caso, puesto que Roco no comercializa sus kits de inicio con esta opción de partida y, por lo tanto, no es necesario retirar ningún decodificador.

Por otra parte, es necesario averiguar cómo se alimentan estas bobinas del accionamiento eléctrico, que sí es necesario adquirir de antemano. Para ello, hay que tener en cuenta que a la vía llega la señal suministrada por el transformador de corriente (16 V AC), modificada por la central digital. Entonces, si se ha comentado que no hay decodificador que controle el desvío, es obvio pensar que hay que integrar un dispositivo externo de control que haga las veces de decodificador, controlando los cambios eléctricamente. El dispositivo que más se ajusta a esta función es el relé electrónico. El relé se instalará

entre el autómata y el desvío, siendo el relé el accionador final del cambio de aguja y el autómata el que dará la orden de cambio cuando sea preciso.

El esquema de conexión es el siguiente:



Uno de los cables de color del desvío (por ejemplo, el de color rojo) va al contacto normalmente cerrado del relé, y el otro cable de color del desvío (color verde) va al contacto normalmente abierto del relé. El cable negro del desvío va conectado directamente a uno de los cables del transformador de alimentación.

El otro cable del transformador de alimentación va conectado al común de cada relé (identificado por el número 11).

Este es el sentido de emplear dos transformadores de alimentación, ya que uno de ellos se centra en la alimentación del sistema digital y el otro va a alimentar el sistema de accionamiento de los desvíos. La explicación es que se está utilizando el dispositivo accionador de cambio de vía que lleva integradas dos bobinas, las cuales deben ser alimentadas a 16 V AC. La diferencia de alimentación radica en que, en este caso, no es necesario ni recomendable que la señal sea modificada por la central digital antes de llegar a las bobinas, puesto que no hay decodificador y, por lo tanto, no hay que modular la señal eléctrica.

Al segundo transformador van a ir conectados los 6 dispositivos de cambio de aguja, de la misma forma que en el esquema anterior.

En este momento se tiene ya desviado el control de todos los desvíos a los relés. Para cerrar el conexionado, es necesario alimentar los relés desde el autómata. Para ello, se conecta la salida del autómata al terminal A1 del relé, mientras que el terminal A2 del relé se conecta a masa. El transistor que lleva integrado la salida del autómata provocará el accionamiento o no de la bobina del relé, siendo para ello, alimentada a 24 V DC o no alimentada, respectivamente. Ya se ha explicado con anterioridad el concepto de contacto normalmente abierto y normalmente cerrado, por lo que ahora simplemente se resumirá el hecho de que cuando la bobina se encuentre sin alimentación, será la posición del desvío que esté conectada al contacto normalmente cerrado la que estará activa (y el contacto NA seguirá abierto, y

cuando se vuelva a alimentar la bobina del relé, la posición del desvío activa cambiará a la ordenada por el contacto normalmente abierto, que se cerrará y dejará abierto el otro contacto. Este es el concepto de contacto de conmutación, que es lo que hace el tipo de relé utilizado en el proyecto, conmutar entre una posición y otra.

10.2. Fabricación de los semáforos

Para la señalización visual de la línea ferroviaria en la maqueta se han diseñado y fabricado semáforos personalizados, utilizando electrónica específica. Se han fabricado los semáforos en vez de comprar los que se comercializan para este tipo de maquetas por dos motivos principalmente:

- Son demasiado costosos económicamente. Cada semáforo puede llegar a tener un precio de unos 25 €, por lo que, al necesitar 18 semáforos, el presupuesto final del proyecto se dispara. Se ha comparado este coste con el de realizar los semáforos de forma independiente utilizando electrónica específica, y el resultado ha sido que resulta más económico realizar los semáforos de forma personalizada.
- Uno de los objetivos principales del proyecto es realizar la automatización de los componentes ferroviarios, por lo que se ha decidido externalizar los semáforos para poder realizar el control de los mismos a través del sistema de automatización.

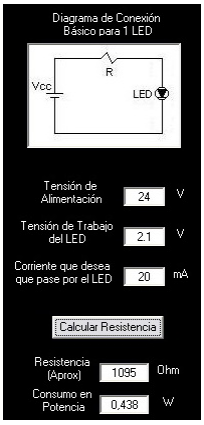
El proceso de fabricación de los semáforos empieza teniendo en cuenta el material necesario para que funcionen correctamente. Se ha decidido que la iluminación de los semáforos se realice con leds. Los led poseen varias ventajas con respecto a otros tipos de iluminación, puesto que tienen bajo consumo y gran durabilidad. Además, su tamaño es apropiado para instalarlos en una maqueta como la que se ha construido para este proyecto. Los leds que se utilizan para la fabricación de los semáforos deben ser alimentados a 2,1 V. Teniendo en cuenta que las fuentes de alimentación utilizadas tienen la salida a 24 V DC, se necesitan componentes que reduzcan obligatoriamente la tensión de entrada a los led, ya que si no serán dañados sin remedio. Estos dispositivos son las resistencias. Ahora bien, hay que determinar de alguna forma el valor de esas resistencias y la potencia (W) que deben soportar (las resistencias absorberán esa tensión de más para que no llegue al led, pero por ello disipará calor, y la resistencia debe ser capaz de disipar todo el calor producido en esa absorción con la seguridad de que el calor no la destruya). Por eso es aconsejable calcular el valor en Ω de la resistencia, pero también se debe tener en cuenta la potencia que debe soportar. El valor de la resistencia se puede calcular mediante la siguiente fórmula:

$$Rs = (Vdd - Vf)/If$$

Siendo Vdd la tensión de alimentación, Vf la tensión de funcionamiento típica del led, If la corriente de funcionamiento típica del led

De esta forma, la resistencia calculada para el led rojo utilizado en el proyecto debería ser de 1095 Ω, recordando que tiene una Vf = 2,1 y una If de 20 mA, siendo la tensión de alimentación de 24 V DC. El mismo valor para el led verde, puesto que las características eléctricas de funcionamiento son idénticas.

Sin embargo, buscando mayor facilidad en el cálculo, existen aplicaciones software muy sencillas de manejar que, introduciendo determinados datos, devuelven el valor de la resistencia adecuado y la potencia que deben soportar. Se puede ver esto en la siguiente imagen:



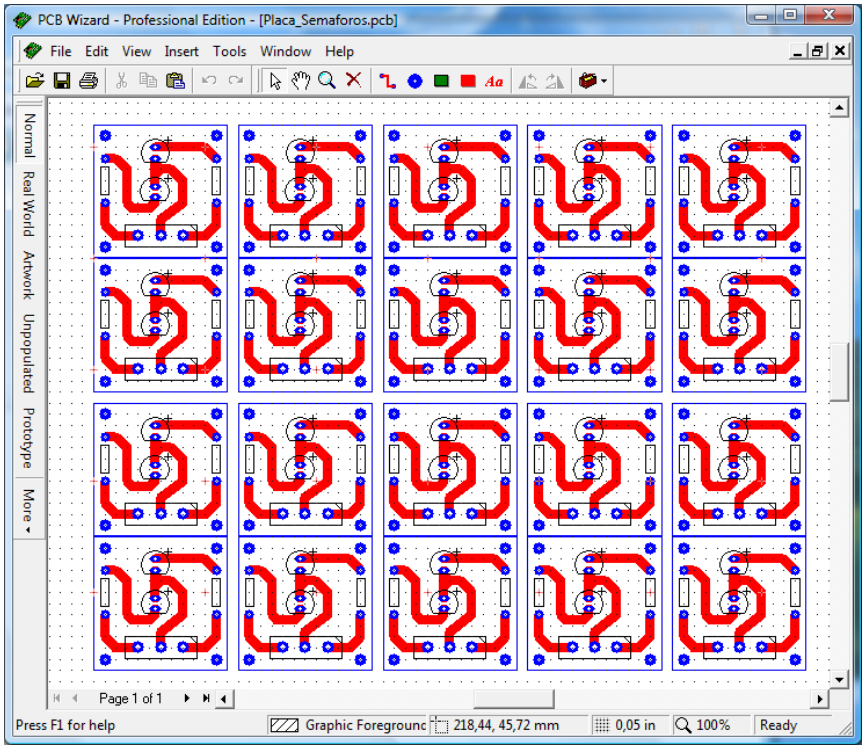
Como se puede observar, se verifica que el valor de la resistencia debe ser de 1095 Ω aproximadamente, recordando que las resistencias que se fabrican tienen valores predeterminados. De esta forma, lo que se aconseja siempre es escoger una resistencia de valor lo más aproximado posible al resultado del cálculo, siempre ajustando al valor de resistencia inmediatamente superior al calculado. Si se escoge una resistencia de un valor inferior se corre el riesgo de dañar la resistencia y el dispositivo protegido, en este caso, el led. Para el caso concreto de este proyecto, el valor seleccionado de resistencia es 1100 Ω, que es el valor estándar de fabricación inmediatamente superior al cálculo teórico. El programa en cuestión también informa que el consumo en potencia que la resistencia debe ser capaz de soportar es de 0,438 W.

Finalmente, para el proyecto se han seleccionado resistencias de 1100 Ω, con una potencia de 0,6W y de película metálica, con mejores propiedades que la película de carbón típica de las resistencias normales.

Por último, se hace necesario disponer de un terminal de conexión de los cables de alimentación por cada semáforo, lo que reportará facilidad de cableado y limpieza visual en la presentación del semáforo.

Una vez seleccionados los componentes, queda diseñar el circuito de cada semáforo. Para que los 18 semáforos se puedan situar en la maqueta ferroviaria en los puntos estratégicos requeridos, es necesario que tengan un tamaño lo más discreto posible. La implementación de los semáforos se ha hecho en placa de circuito impreso (PCB), diseñando primero el circuito y después usando técnicas de revelado de placas de circuito impreso para la fabricación física de los mismos.

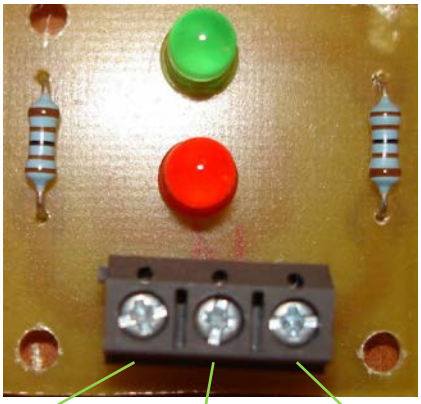
Para el diseño de los circuitos de semáforos se ha utilizado software especializado en el diseño de PCBs, en concreto, se ha utilizado el software PCB Wizard, un software sencillo y potente que permite el dimensionado real de todos los componentes electrónicos y pistas del circuito, con lo cual se puede calcular el espacio y diseño necesarios de construcción sobre el mismo software. Este programa tiene integrado un conjunto de librerías de componentes electrónicos diversos con sus dimensiones reales, de forma que se puede diseñar el circuito PCB con total exactitud:



Como se puede observar, finalmente se ha diseñado una placa que contiene 20 circuitos de semáforo, empleando para el proyecto 18 de ellos. Después de la fase de diseño es necesaria una placa de fibra de vidrio y técnicas de revelado para imprimir el diseño sobre la placa, para después atacar con ácido a la misma y dejar al descubierto el cobre de las pistas y componentes electrónicos, donde se soldarán más tarde los mismos. Una vez fabricada la placa, se debe taladrar los orificios para introducir los componentes electrónicos. A continuación, hay que proceder a soldar con estaño los componentes. En cuanto al conexionado de las resistencias, no tienen polaridad, por lo que se pueden colocar en el sentido que se prefiera, y a diferencia de éstas, con el conexionado de los led hay que tener en cuenta que tienen polaridad (ánodo y cátodo). Esto es muy importante, puesto que, si se colocan al revés, el diodo quedará polarizado inversamente, y no funcionará. En la siguiente imagen se puede ver un led típico:

En el diseño realizado del circuito de semáforo, el terminal corto (cátodo) va conectado a masa, es decir, al borne central del terminal de conexión, mientras que el terminal largo (ánodo) va conectado a la salida digital del autómata (+24V DC).

Un semáforo construido completamente quedaría de la siguiente forma:

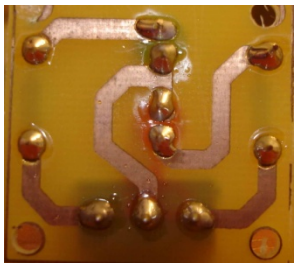


Conexión a salida digital autómata para controlar led rojo

Conexión a salida digital autómata para controlar led verde

Conexión a masa común

En la siguiente imagen se puede ver el semáforo por el lado de las pistas:

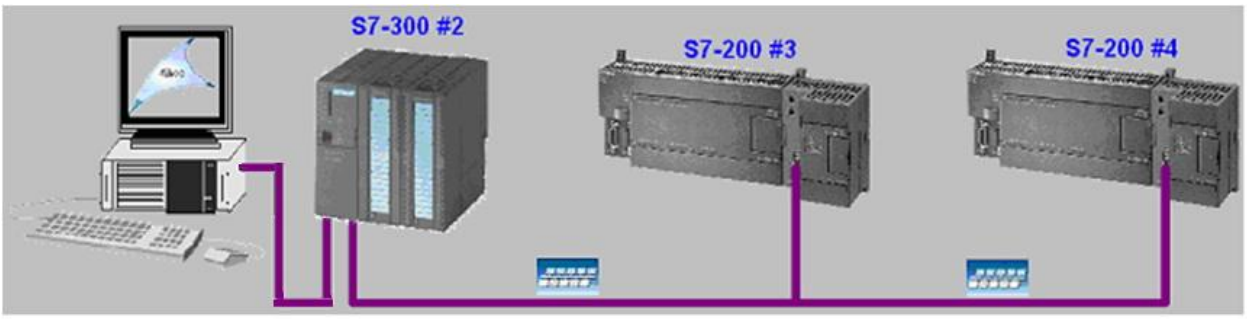


A continuación, se han construido soportes realizados en madera para los 18 semáforos situados en la maqueta:

10.3. Acciones hardware en el sistema de automatización

A continuación, se va a explicar la estructura hardware del sistema de automatización.

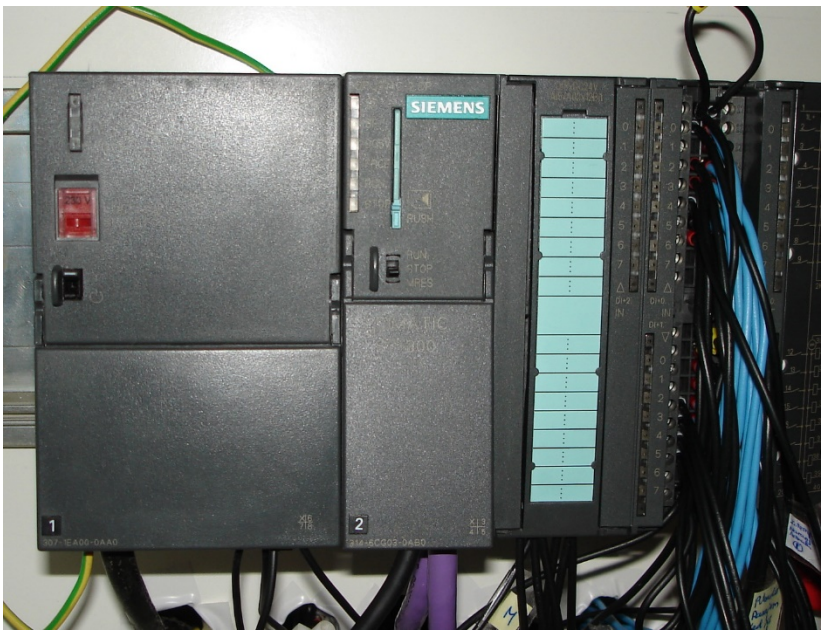
El sistema de automatización va a estar compuesto de tres autómatas o PLCs, además de un PC de control y monitorización, principalmente. El diagrama de bloques del sistema a grosso modo es el siguiente:



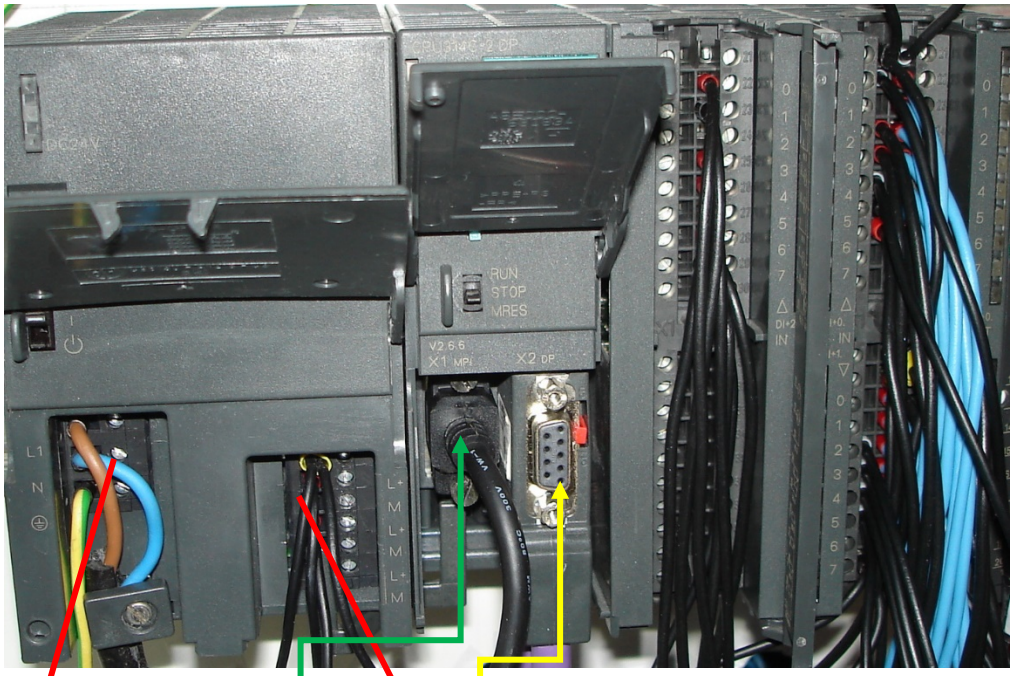
El número que aparece al lado de cada autómata es la dirección del dispositivo asignado en la red PROFIBUS DP.

Para las instrucciones de conexionado de cada autómata es aconsejable la lectura del manual de sistema que Siemens suministra de cada autómata, donde vienen explicadas con detalle todas las conexiones. No obstante, se va a ir analizando brevemente las conexiones efectuadas en cada autómata.

En cuanto al S7 314C-2DP, en la siguiente imagen se muestra la estructura completa de los componentes que lo forman:



A la izquierda de la imagen se puede observar la fuente de alimentación, y justo al lado por la derecha, se encuentra la CPU y los racks de E/S digitales y analógicas. Como es una CPU compacta, estas E/S van integradas en la CPU. Estas E/S integradas han sido suficientes para el desarrollo del proyecto, por lo que no han hecho falta módulos de ampliación adicionales. En la siguiente imagen se puede ver la misma estructura pero con las tapas internas de conexiones abiertas:



Conexiones de alimentación a tensión de red (230 V AC)

Conexión de alimentación para CPU y conexiones auxiliares de 24 VDC

Conexión puerto MPI (programación PLC a través de PC)

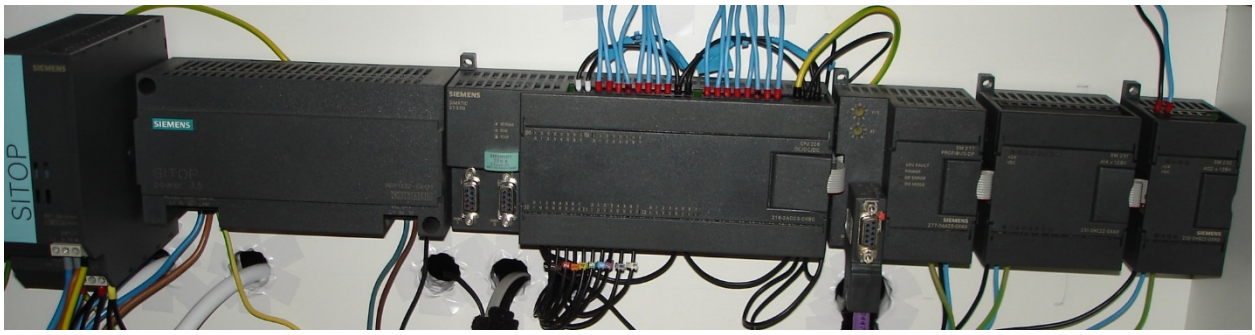
Conexión puerto PROFIBUS DP (Master)

Como se puede observar, la fuente de alimentación transforma la tensión de red de 230V AC en 24 V DC, ajustando esta tensión para el funcionamiento de la CPU, E/S y dispositivos que se quieran conectar compatibles con esa tensión de funcionamiento. Se puede observar que esta CPU viene provista de un pequeño conector de plástico que tiene la forma adecuada para realizar el puente de conexión entre la fuente de alimentación y la CPU. Esto disminuye el cableado manual y permite un conexionado rápido y robusto entre los dos dispositivos. Por otra parte, en lo que es el bloque de la CPU se pueden apreciar los puertos de comunicaciones. Esta CPU dispone de puerto MPI, un tipo de comunicación muy utilizado en automatización y que en este caso es utilizado para conectar el PLC al PC y poder de esta forma introducir el programa en el PLC, salvaguardarlo en el PC, conexión PC/PLC para diagnósticos, etc. El adaptador USB utilizado para esto se puede ver en la parte inferior del autómata. Esta conexión no es necesaria cuando no se esté programando o diagnosticando el autómata, puesto que en este proyecto se utiliza exclusivamente la conexión PROFIBUS DP durante el funcionamiento del sistema. Justo al lado del puerto MPI se encuentra el puerto PROFIBUS DP (en la imagen el conector PROFIBUS está conectado al mismo). Con este puerto de comunicaciones se puede habilitar el S7 314 como un dispositivo master o slave en la red PROFIBUS, dependiendo de la estructura y diseño del sistema de automatización deseado.

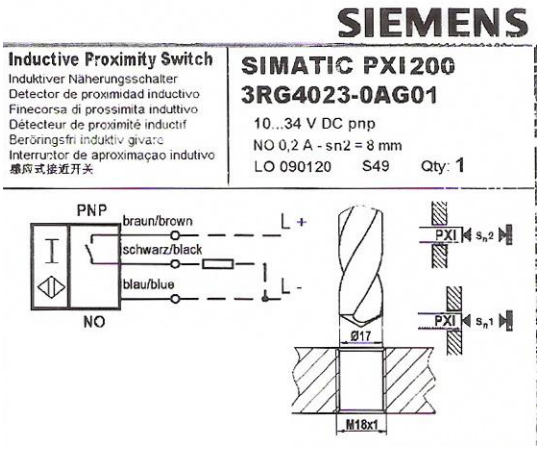
El esquema de conexionado de las E/S digitales se encuentra en la puerta del rack, en la que además aparecen las E/S numeradas. En los bornes marcados con letra +L se deberá llevar tensión de +24V DC, y a los terminales marcados con la letra M se deberá llevar masa. Estos bornes específicos están diseñados para, una vez conectados, ya sea a +24V DC o a masa, repliquen esa conexión al resto del bornero.

En cuanto al direccionamiento de las E/S digitales utilizadas, se pueden direccionar libremente si se quiere, pero se ha decidido utilizar las direcciones marcadas por defecto (bytes 124, 125 y 126 para las entradas digitales y bytes 124, 125 para las salidas digitales).

En cuanto a los S7-226, los dos utilizados en este proyecto tienen la misma estructura. Es la siguiente:



A la izquierda de la imagen se puede ver la fuente de alimentación (conectada a tensión de red de 230V AC), que suministra tensión de 24V DC al autómata y dispositivos que se conecten auxiliarmente compatibles con esta tensión de funcionamiento. En el centro se puede ver el PLC S7-226. En el rack de entradas digitales (parte inferior del autómata), se han conectado los sensores inductivos de una de las estaciones, con el siguiente esquema de conexión (responde a un sensor de tipo pnp):



Cable azul: conectado a L-

Cable marrón: conectado a L+

Cable negro: conectado a la entrada digital del PLC.

En la parte superior del autómata se han conectado las señales correspondientes a 8 semáforos (cada semáforo tendrá dos señales, una para el led rojo y otra para el led verde). En el frontal se pueden apreciar los dos puertos de comunicaciones de los que dispone. Normalmente se ha utilizado el puerto 0 para la conexión con el PC para tareas de programación y diagnóstico.

A continuación, se puede apreciar el puerto de comunicaciones PROFIBUS DP (módulo EM277 dedicado). Este módulo ha sido configurado específicamente para la conexión a la red Profibus.

A continuación, se puede ver el módulo de entradas analógicas EM 231. Y por último, el módulo de salidas analógicas EM232, el cual dispone de dos salidas de tipo analógico que se pueden configurar bien en tensión o en corriente. En este proyecto se utilizan las salidas analógicas en tensión, por lo, tanto, sólo se conexionará la salida analógica en tensión V0. Aunque se podrían haber conectado las dos salidas analógicas controlando con un solo autómata la velocidad de las dos locomotoras, se ha preferido que cada autómata S7-226 controle sólo una locomotora. El borne V0 va directamente conectado al cable de alimentación del mando de control de la locomotora, mientras que el borne M va directamente conectado al cable de masa del mando de control de la locomotora. Establecida esta conexión, la salida analógica está dispuesta ya para controlar mediante voltaje la velocidad de la locomotora.

En cuanto al otro autómata S7-226, las conexiones son idénticas, exceptuando que hay un sensor más conectado y, por lo tanto, una entrada digital más utilizada.

En cuanto al PC de control, destacar que hay que insertar la tarjeta de comunicaciones CP 5622 en un bus PCI libre de la placa base del PC. Los drivers para Windows de esta tarjeta de comunicaciones se encuentran implícitos en el software de programación STEP 7, por lo que basta con tener el software instalado ya en el PC para que al poner la tarjeta y arrancar el PC el sistema operativo la reconozca y pueda utilizarse ya.

En cuanto a la comunicación PROFIBUS DP, en la siguiente imagen puede verse el procedimiento de conexionado de los conectores PROFIBUS con el sistema Fast Connect:

Los conectores poseen cuatro conexiones diferentes (dos de ellas son los dos cables de entrada, y otras dos los dos cables de salida si es necesario utilizarla). Los cables están diferenciados en color verde y rojo

y el conector también está marcado con los mismos colores. Además, es prácticamente imposible equivocarse en la conexión, puesto que está todo marcado y la herramienta de pelado deja ya el cable preparado para conectar. Incluso la herramienta de pelado deja la distancia correcta para la malla externa, que se posiciona perfectamente en el conector.

Los conectores que se han utilizado tienen además una resistencia terminadora. Es muy importante, puesto que gracias a ella se puede determinar el final de la red de dispositivos PROFIBUS.

En los dispositivos que no sean fin de la red PROFIBUS (S7 314C-2DP y S7 226 Estación 1), y, por tanto, tendrán que tener los dos canales del conector (entrada y salida) utilizados, la resistencia terminadora hay que ajustarla en posición off, quedando de la siguiente manera:

En los dispositivos que sean fin de la red PROFIBUS (S7 226 Estación 2 por un extremo y el PC por el otro extremo) y, por tanto, sólo tendrán utilizado el canal de entrada del conector, hay que ajustar la resistencia terminadora en posición on. Esto indica que ese dispositivo es el último de la red, y por lo tanto, no hay canal de salida conectado por el que seguir transmitiendo información.

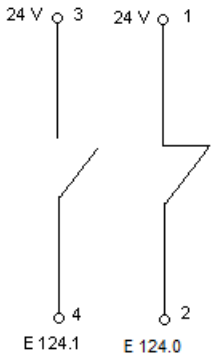
10.3.1. Conexión de los pulsadores de puesta en marcha/paro

En el sistema se van a instalar tres pulsadores de puesta en marcha/paro. La utilización de los mismos es la siguiente:

- 1 para la puesta en marcha/paro del sistema en general.
- 1 para encender/apagar la locomotora de pasajeros.
- 1 para encender/apagar la locomotora de mercancías.

Al ser pulsador doble, tiene un contacto NA y otro contacto NC. La posición por defecto es siempre stop o parado, con lo cual esta posición será la que se conexionará con el contacto NC. Por lo tanto, al contacto NA se conecta al botón de puesta en marcha. Los botones o pulsadores no tienen enclavamiento, por lo que retornan a su posición cada vez que se pulsan. De esta forma, en la programación de los pulsadores hay que tener en cuenta la autoretención, ya que, si no es así, se tendría que estar con el pulsador constantemente presionado para generar el comportamiento marcha/paro deseado.

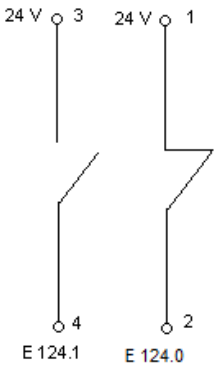
Este es un ejemplo del conexionado de un pulsador doble.



Estos pulsadores van provistos de un led preparado para ser alimentado a 24V DC directamente, que se puede utilizar para saber a simple vista cuando se encuentra activo el sistema y las locomotoras.

10.3.2. Conexión de la seta de emergencia

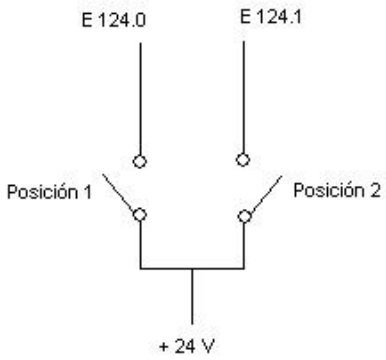
En el sistema se va a instalar una seta de emergencia. Este dispositivo es imprescindible en cualquier instalación automatizada, puesto que es un elemento de seguridad previsto para poder parar la actividad de la instalación en un momento de emergencia. Destacar que no es lo mismo apagar el sistema que parar al sistema por alguna emergencia. Apagar el sistema desde el pulsador de parada significa desconectar el sistema, de forma que no se pueda ejercer ninguna acción sobre el mismo a no ser que sea volverlo a poner en funcionamiento. Sin embargo, parar al sistema por una emergencia implica que se pare toda actividad que pudiese estar haciendo el mismo, en este caso, pararía las locomotoras si estuviesen circulando, pero el sistema seguiría conectado. En esa condición, liberando la seta de emergencia se podría reanudar el normal funcionamiento del sistema, sin necesidad de volver a activarlo (ojo, si se deben reanudar las actividades que pudiese estar ejecutando antes de ser parado por una emergencia). El esquema de conexiones es igual al anterior (contacto NA+ contacto NC).



La seta de emergencia se libera girando levemente el pulsador en forma de seta. Esto es así porque lleva incorporado un sistema de retención. El sentido de giro viene impreso en el pulsador de la seta.

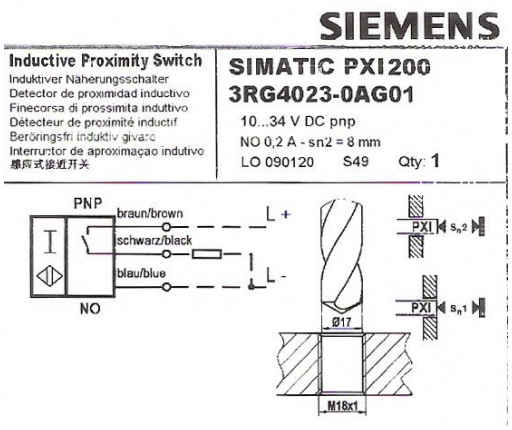
10.3.3. Conexión del interruptor de conmutación

El interruptor de conmutación se va a utilizar para cambiar el modo de funcionamiento del sistema de control automatizado, de forma que se pueda elegir entre modo manual y modo automático. El interruptor de conmutación es, por lo tanto, de dos posiciones, y un ejemplo de cómo puede ser su esquema de conexión es el siguiente:



10.3.4. Conexión de los sensores inductivos

Los sensores inductivos son cableados a las entradas digitales de las CPU S7 226. El cableado de los sensores es el siguiente:



Cable azul: conectado a L- (FA)

Cable marrón: conectado a L+ (FA)

Cable negro: conectado a la entrada digital del PLC.

El esquema de conexión responde a conexión pnp ó de lógica positiva.

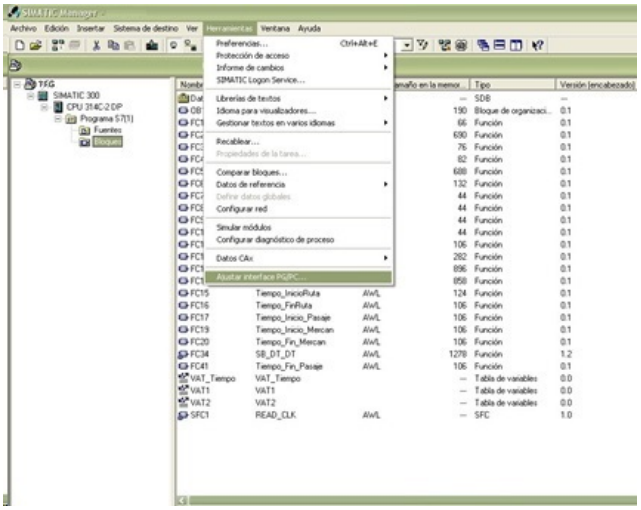
11. Instalación y configuración del software necesario.

Después de instalar el software necesario y antes de empezar a realizar la programación, hay que realizar varias configuraciones.

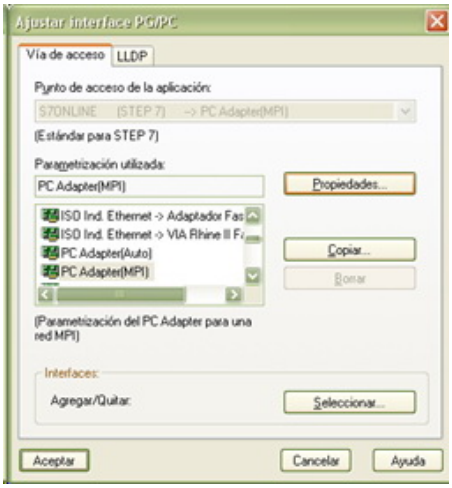
11.1. Configuración de comunicaciones MPI S7 314C-2DP/PC

Para realizar la configuración y programación del PLC S7 314 es necesario conectarlo con el PC y tener el software STEP 7 instalado. Se debe conectar para ello el adaptador MPI/PC (visto en la descripción del material descrito en la fase de análisis y diseño). Una vez conectado, ha habido que configurar la comunicación PG/PC por software (Administrador SIMATIC del STEP 7), de la siguiente forma:

- Acceder al menú Herramientas → Ajustar interface PG/PC



Aparecerá la siguiente ventana:



Entre todas las opciones que muestra la ventana, hay que seleccionar PCAdapter (MPI) como parametrización utilizada. A continuación, hacer clic sobre el botón Propiedades para visualizar las propiedades de esa parametrización concreta. Aparecerá la siguiente ventana:

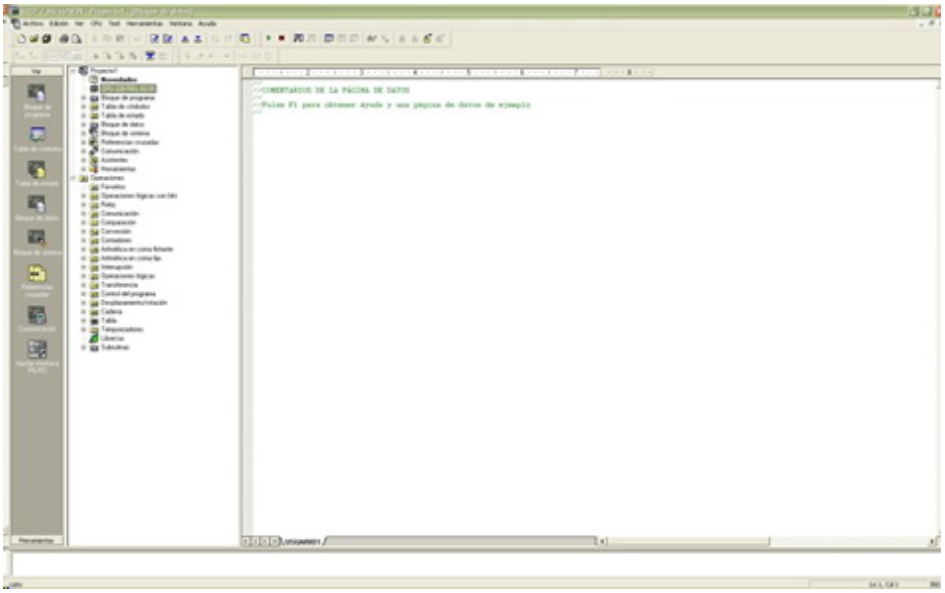
Para realizar el proyecto se ha tenido en cuenta el hardware de automatización para la configuración MPI con el S7 314, se ha configurado la dirección 0 con un timeout de 30 segundos. La velocidad de transferencia es 187,5Kbit/s, siendo la dirección de estación más alta permitida 31. En cuanto a la opción "PG/PC es el único maestro del bus", en este caso, es indiferente marcarla o no, funcionando igualmente si hubiese sido marcada, puesto que este PLC es el único equipo conectado a la red MPI. Si se selecciona la pestaña "Conexión local" aparece la siguiente ventana:

En esta ventana se ha seleccionado el tipo de conexión existente, en este caso, el adaptador MPI lleva conexión USB, por lo tanto, se selecciona "Conexión a" USB. Se marca la casilla "Aplicar configuración a todas las parametrizaciones de la tarjeta".

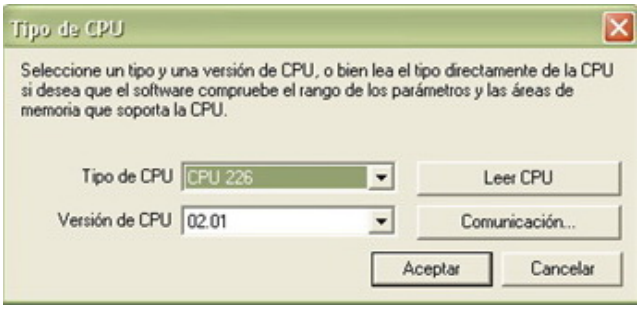
Con la realización de estos pasos, y conectando físicamente el adaptador al puerto MPI del PLC y a un puerto USB del PC, se obtiene la conexión satisfactoria del PLC con el PC mediante el Administrador SIMATIC del STEP 7.

11.2. Configuración de comunicaciones PPI S7 226/PC

Para realizar la configuración y programación de los PLC S7 226 ha sido necesario conectarlos con el PC y tener el software STEP 7 Micro/WIN instalado. Se debe conectar para ello el adaptador PPI/PC (visto en la descripción del material en la fase de análisis y diseño). Una vez conectado, la configuración de la comunicación PG/PC por software (desde el STEP 7 Micro/WIN), se puede realizar de varias formas. Una de ellas es la siguiente:



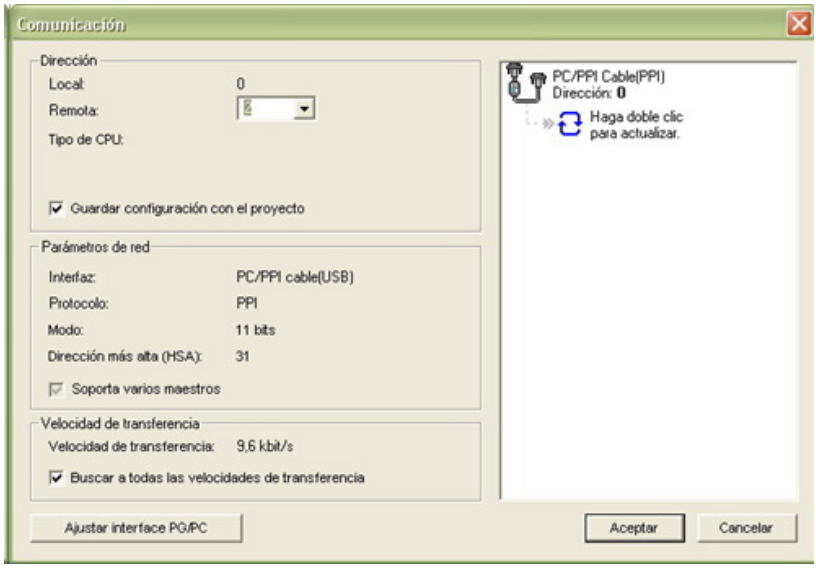
Una vez abierto Micro/WIN, se hace doble clic sobre la CPU que muestre por defecto el programa una vez abierto (en la imagen sería la CPU 226 REL 02.01). Al hacer doble clic se abre la siguiente ventana:



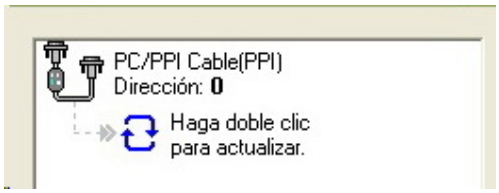
Si ya se ha configurado anteriormente la comunicación correcta alguna vez que el programa hubiese estado abierto, directamente seleccionar la CPU correcta y la versión de firmware de la misma. Si es la primera vez que se va a realizar la configuración de comunicaciones, se tienen dos opciones:

1) Clic sobre el botón “Leer CPU” → El programa intentará leer las características de la CPU.
2) Clic sobre el botón “Comunicación” → Esta es la opción más adecuada para configurar la comunicación PLC/PC vía PPI por primera vez, ya que permite parametrizar las comunicaciones manualmente.

Escogiendo la opción “Comunicación”, aparecerá la siguiente ventana:

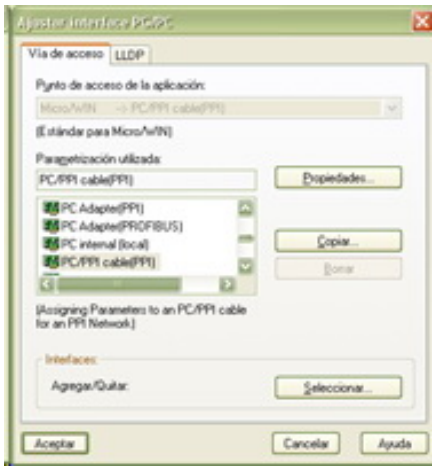


En esta pantalla también se pueden adoptar diversas opciones. Si ya se ha configurado alguna vez anteriormente el interface PG/PC, bastará con hacer doble clic sobre el siguiente dibujo:



Con ello, el programa testeará todas las direcciones y velocidades compatibles con el adaptador PPI/PC en busca de estaciones conectadas al bus PPI. Cuando las encuentre, el programa mostrará todas ellas.

Por otra parte, si el programa no es capaz de encontrar ninguna estación conectada o si nunca antes se había configurado el interface PG/PC, se debe hacer clic sobre la opción “Ajustar interface PG/PC” de la ventana. Haciéndolo, aparecerá la siguiente ventana:



La opción adecuada, teniendo en cuenta de que el cable de comunicación es PPI, es PC/PPI cable (PPI). Para configurar la parametrización escogida, hay que hacer clic sobre el botón “Propiedades”. Al hacerlo, aparece la siguiente ventana:

La dirección de la estación es 0, puesto que además la CPU S7 226 que es conectada, será la única estación conectada al bus PPI, con un timeout de comunicación establecido en 10 segundos.

La velocidad de transferencia, ha sido configurada en 9,6 kbit/s, aunque el cable PPI y la CPU admiten más posibilidades de velocidad de transferencia. La dirección más alta configurable es la 31.

Seleccionando la pestaña “Conexión local”, se accederá a la siguiente ventana, donde hay que configurar el tipo de conexión y, puesto que el cable de comunicaciones PPI es USB, se configura esa opción:

11.3. Configuración física de los PLCs S7-226 para comunicación PROFIBUS DP

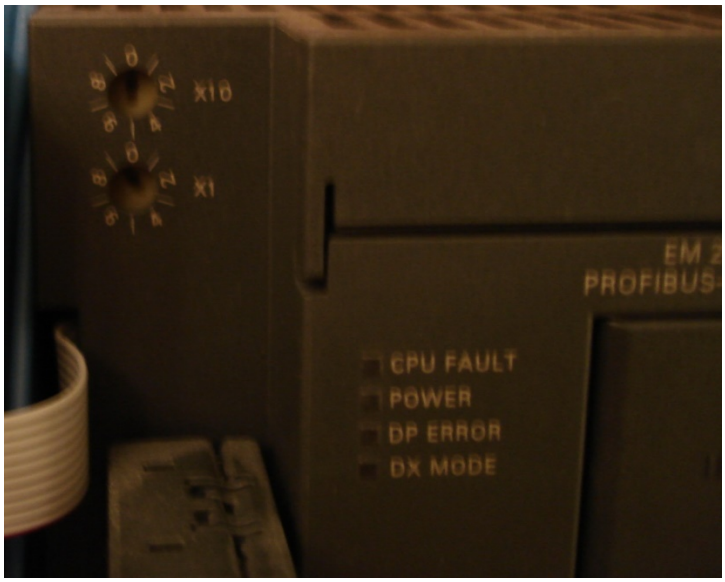
Se ha tenido que configurar los PLC S7-226 para su incorporación en la red PROFIBUS DP. Para ello, lo primero que ha habido que comprobar ha sido que el módulo EM 277 de cada CPU se haya acoplado al bus externo de la CPU. Después se ha tenido que configurar una dirección de PROFIBUS DP. Para ello se giran los conmutadores rotativos que el módulo EM 277 incorpora en su parte frontal, colocándolos de tal forma que indique el número de estación asignado a cada uno. En el proyecto es:

- Esclavo con la dirección #3:
  - $10^1 = 0$ (X10)

- 10^0 = 3 (X1)

Esclavo con la dirección #3

- Esclavo con la dirección #4:
  - 10^1 = 0 (X10)
  - 10^0 = 4 (X1)



Esclavo con la dirección #4

11.4. Implementación de la red PROFIBUS DP en Step7

Para la implementación de la red se han tenido que seguir los siguientes pasos:

- Crear nuevo proyecto en STEP 7 (será el proyecto de programación completo del S7-314).

- Insertar equipo → SIMATIC S7-300
- Configuración del hardware de la CPU S7-300

Entrar en HW-Config para configurar el hardware del PLC S7-300:

- Insertar bastidor. Catálogo de componentes → SIMATIC 300 →BASTIDOR 300 → Perfil soporte.
- Insertar fuente de alimentación. Catálogo de componentes → SIMATIC 300 → PS 300 → PS 307 5ª
- Insertar CPU. Catálogo de componentes → SIMATIC 300 → CPU 300 → CPU 314C-2DP
- Instalar archivo GSD del módulo EM277.

Se debe disponer de los archivos:

siem089d.gsd siem 089d.bmp

Estos se pueden encontrar en Internet en las siguientes páginas:

www.siemens.com

www.profibus.com

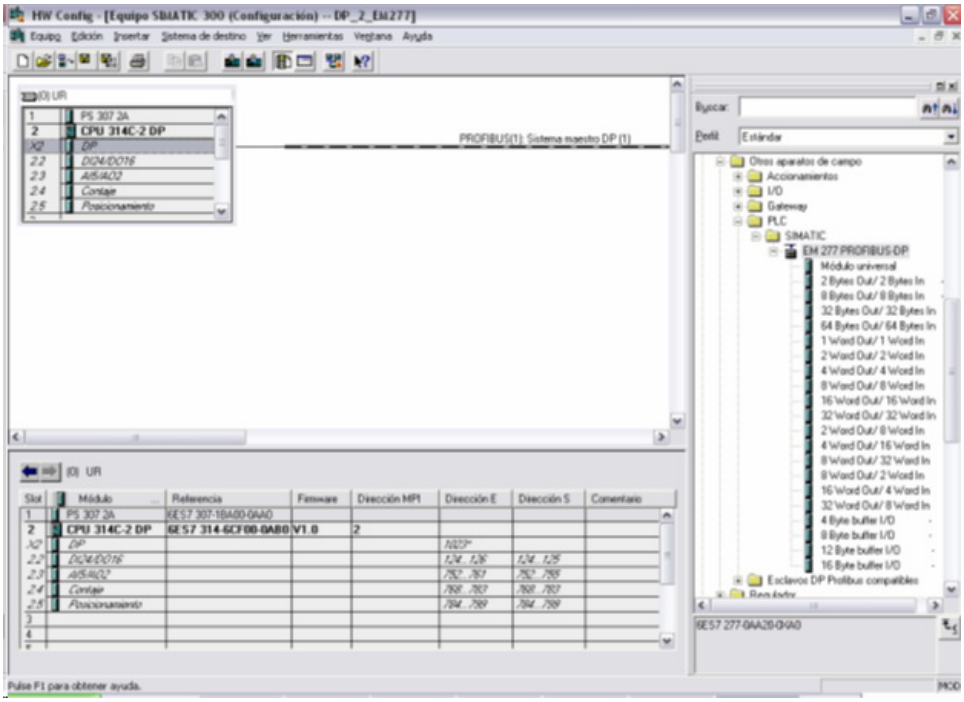
Desde el HW-Config, elegir menú Herramientas→Instalar archivos GSD→Buscar el lugar donde se encuentran los archivos anteriormente mencionados, seleccionar el archivo "siem089.gsd" y pulsar el botón "Instalar". Con esto se ha conseguido que aparezca en el catálogo de componentes el equipo EM277. Éste se encuentra dentro de:

PROFIBUS DP→OTROS APARATOS DE CAMPO→PLC→SIMATIC→EM 277 PROFIBUS DP

Dentro de HW-Config, hacer un doble clic sobre la conexión "X2-DP", que corresponde al puerto DP integrado en la CPU. Aparece una ventana de "Propiedades", en la que se debe pulsar en el botón "Propiedades…" para configurar la red PROFIBUS DP en la CPU S7-314C-2DP. Hay que hacer los siguientes pasos:

a) Crear una nueva red PROFIBUS.
b) Asignarle la dirección #2 al S7 314C-2DP.
c) Conectarlo a la red.

Una vez comprobado que todo es correcto, aparece una ventana, en la que se encontrará la CPU, de donde nace una línea que corresponde al bus PROFIBUS-DP, al cual se deberá ir insertando los diferentes esclavos.



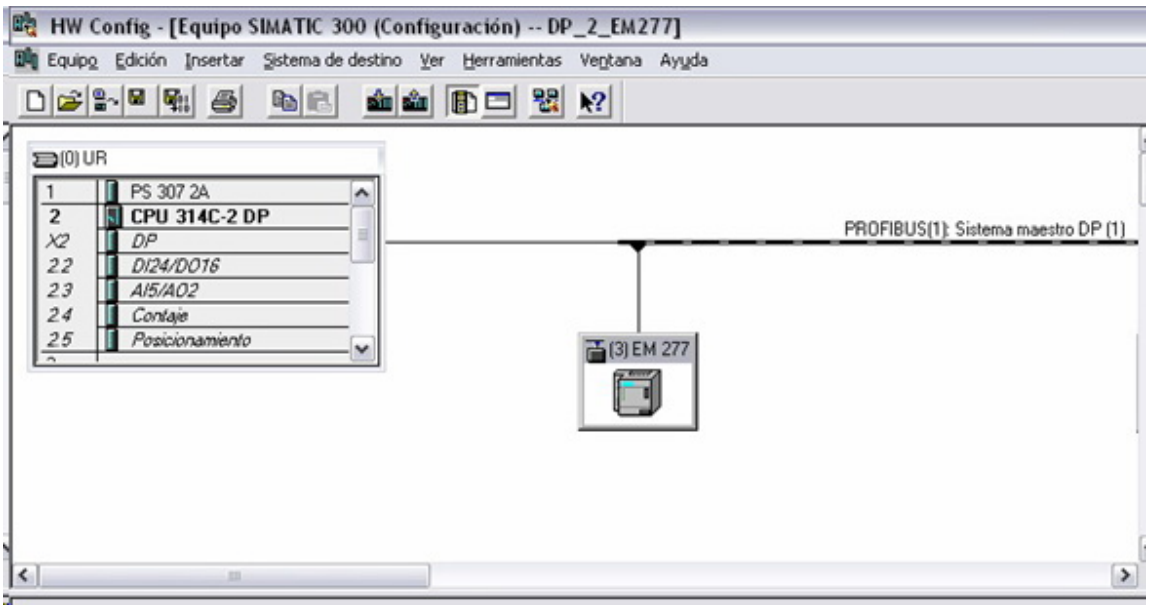
Dentro de HW-Config, hay que elegir el equipo EM277 del catálogo. Se encuentra en la siguiente ruta (del catálogo):

PROFIBUS DP→OTROS APARATOS DE CAMPO→PLC→SIMATIC→EM 277 PROFIBUS-DP

Allí se ha seleccionado el componente EM277 y hay que arrastrarlo hasta conectarlo a la red PROFIBUS-DP.

Se ha tenido que modificar la dirección que el equipo tiene en la red. Para ello, desde la pestaña "Parámetros", se ha tenido que asignar la dirección #3.

Con esta acción ya se tendrá un equipo EM277 conectado a la red PROFIBUS DP.



Haciendo doble clic sobre el equipo EM277 se abre una ventana, donde hay que acceder a la pestaña "Parametrizar", se despliega el apartado "Parámetros específicos del aparato", apareciendo un campo que se debe modificar, que es "I/O Offset in the V-memory", y que inicialmente tendrá valor 0. En lugar de 0, en el proyecto se ha determinado valor 2000. De esta forma se da casi por finalizada la configuración del PLC S7-226 a través del módulo EM277 en la red PROFIBUS DP trabajando como esclavo.

El otro PLC S7-226 tendrá una configuración idéntica a éste que se acaba de ver, con la excepción de que el segundo PLC tendrá la dirección #4 en la red PROFIBUS DP.

El siguiente paso que se ha dado, ha sido determinar el tamaño del buffer de comunicación entre el master y los esclavos. Esto viene determinado por la relación de datos a intercambiar.

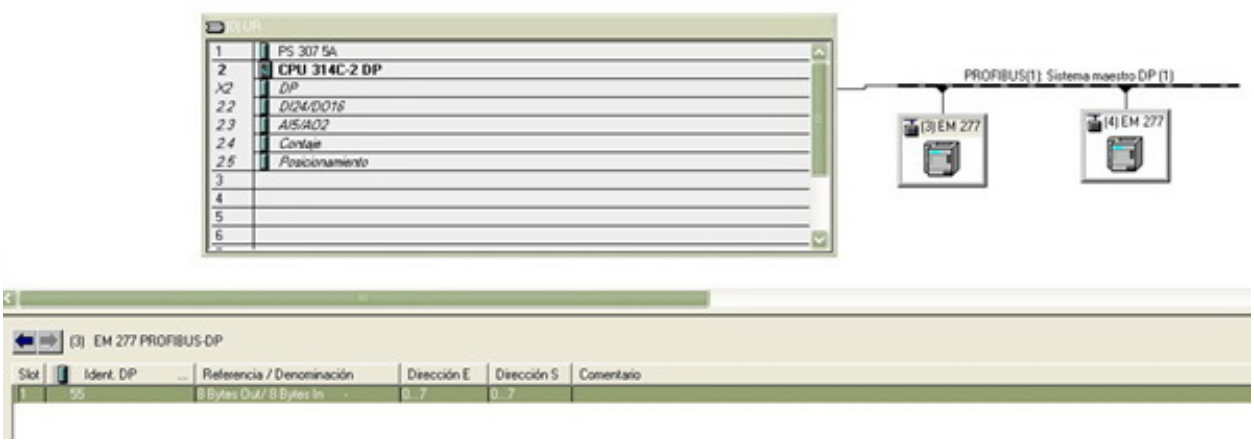
Entre las diversas opciones que permite el EM277, se ha seleccionado 8 bytes Out/8 bytes In, puesto que la opción inferior de 2 bytes Out/2 bytes In es insuficiente. De esta forma, la configuración de intercambio de datos entre el S7 314C-2DP y el S7 226 #3, queda:

CPU S7 226 (área de memoria V)			**CPU S7 314C-2DP**	
VW2008/ VW2014	8 bytes en donde pondrá el S7 226 los datos para que el master lea.	→	8 bytes en donde se colocarán los datos que el S7-314 lee de los bytes de salida del buffer de comunicaciones del S7-226.	EW0/EW6
VW 2000/ VW 2006	8 bytes en donde se desea que el S7-226 deposite los datos escritos por el master.	←	8 bytes en donde se colocarán los datos que el S7-314 quiere escribir en los bytes de entrada del buffer de comunicaciones del S7-226	AW0/AW6

De la misma forma, la configuración del intercambio de datos entre el S7 314C-2DP y el S7-226 #4 queda:

CPU S7 226 (área de memoria V)			**CPU S7 314C-2DP**	
VW2008/ VW2014	8 bytes en donde pondrá el S7 226 los datos para que el master lea.	→	8 bytes en donde se colocarán los datos que el S7-314 lee de los bytes de salida del buffer de comunicaciones del S7-226.	EW8/EW14
VW 2000/ VW 2006	8 bytes en donde se desea que el S7-226 deposite los datos escritos por el master.	←	8 bytes en donde se colocarán los datos que el S7-314 quiere escribir en los bytes de entrada del buffer de comunicaciones del S7-226	AW8/AW14

Esto se ha realizado en HW-Config seleccionando la opción "8 Bytes Out/8 Bytes In" y arrastrándola a la rejilla inferior de la configuración hardware del PLC-S7 314.



En la configuración anterior se ha parametrizado el valor 2000 como “I/O Offset in the V-memory”. Este parámetro indica en qué zona de memoria el master de la red, en este caso, el S7 314C-2DP debe escribir y/o leer los datos del esclavo (en este caso el S7 226). La zona de memoria destinada debe ser el área de datos V del PLC S7 226, área que es inalterable por el usuario. Lo que sí es configurable por el usuario es el valor a partir del cual asignamos el área de intercambio, valor que en este caso es 2000, queriendo esto decir que a partir del VB2000 del PLC S7-226 (en ambos dos), el master de la red (S7 314C-2DP) utilizará para el intercambio de información.

12. Programación de los autómatas de control.

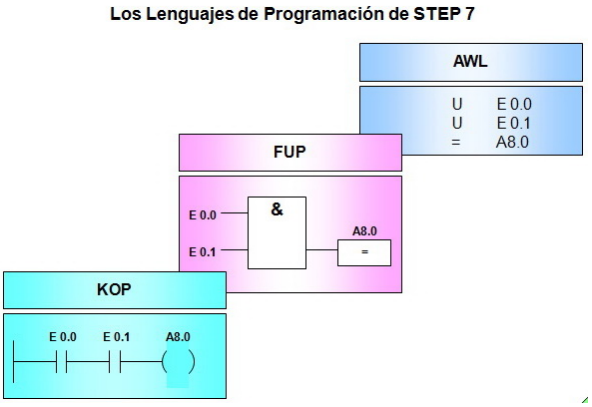
12.1. Programación del S7 314C-2DP

LISTA DE INSTRUCCIONES UTILIZADAS EN LA PROGRAMACIÓN

STEP 7 dispone de varios lenguajes de programación que puede utilizar el programador, dependiendo de la preferencia y conocimientos de los que disponga. No obstante, y siempre basándose en unas normas específicas, el programa puede crearse en Lista de Instrucciones y ser convertido a otro lenguaje de programación. Los lenguajes de programación disponibles en STEP 7 son los siguientes:

- **KOP**: También llamado Esquema de Contactos. Es muy similar a un esquema eléctrico. Se usan símbolos como contactos y bobinas. Este lenguaje de programación está indicado para aquellos que "crecieron" con contactores.
- **AWL**: También llamado Lista de Instrucciones. Se compone de instrucciones de STEP 7. Este tipo de lenguaje permite bastante libertad a la hora de programar, siendo el preferido por los programadores que ya están familiarizados con otros lenguajes de programación.
- **FUP**: También llamado Diagrama de Funciones. Utiliza "cajas" para las funciones individuales. El carácter que aparece en la caja indica la función (por ejemplo, &→Operación lógica AND). Este lenguaje de programación tiene la ventaja de que incluso un "no programador", como por ejemplo un ingeniero de proceso, puede trabajar con él.

Este es un ejemplo de la representación de instrucciones en estos tres lenguajes de programación:



En el proyecto se ha utilizado el lenguaje AWL, que permite facilidad de programación y una visión global del código en forma de programación en lista de instrucciones.

A continuación, se muestra una lista con las instrucciones utilizadas en lenguaje AWL. Se adjunta el formato de la instrucción con información ampliada y ejemplos obtenidos de los documentos de información técnica de Siemens:

- INSTRUCCIÓN U (Y)

Formato

U <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descripción de la operación

U consulta el bit direccionado para saber si tiene el estado de señal "1", y combina el resultado de la consulta con el RLO realizando una Y lógica.

Consultar el estado de los bits de la palabra de estado:

Utilizando la operación Y también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo

Programa AWL		Esquema de conexiones de relé
		Barra de alimentación
U	E 1.0	E 1.0 Estado de señal 1 — Contacto normal. abierto
U	E 1.1	E 1.1 Estado de señal 1 — Contacto normal. abierto
=	A 4.0	A 4.0 Estado de señal 1 — Bobina
Indica un contacto cerrado.		

- INSTRUCCIÓN UN (Y-No)

Formato

UN <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descripción de la operación

UN consulta el bit direccionado para saber si tiene el estado de señal "0" y combina el resultado de la consulta con el RLO realizando una Y lógica.

Consultar el estado de los bits de la palabra de estado:

Con la operación Y-No también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo

Programa AWL		Esquema de conexiones de relé
		Barra de alimentación
U	E 1.0	E 1.0 Estado de señal 0 — Contacto normal. abierto
UN	E 1.1	E 1.1 Estado de señal 1 — Contacto normal. cerrado
=	A 4.0	A 4.0 Estado de señal 0 — Bobina

- INSTRUCCIÓN O

Formato

O <bit>

Operando	Tipo de datos	Area de memoria
<Bit>	BOOL	E, A, M, L, D, T, Z

Descripción de la operación

O consulta el bit direccionado para saber si tiene el estado de señal "1", y combina el resultado de la consulta con el RLO realizando una O lógica.

Consultar el estado de los bits de la palabra de estado:

Con la operación O también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	x	1

Ejemplo

Programa AWL		Esquema conexiones de relé
		Barra de alimentación
O	E 1.0	E 1.0 Estado de señal 1 Contacto — E 1.1 Estado de señal 0 Contacto
O	E 1.1	
=	A 4.0	A 4.0 Estado de señal 1 — Bobina
		Indica un contacto cerrado.

- INSTRUCCIÓN ON (O-No)

Formato

ON <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

Descripción de la operación

ON consulta el bit direccionado para saber si tiene el estado de señal "0", y combina el resultado de la consulta con el RLO realizando una O lógica.

Consultar el estado de los bits de la palabra de estado:

Con la operación O-No también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	x	1

Ejemplo

Programa AWL		Esquema de conexiones de relé
		Barra de alimentación — Contacto
O	E 1.0	E 1.0 Estado de señal 0 — normalm. abierto
ON	E 1.1	E 1.1 Estado de señal 1 — Contacto normalm. cerrado
=	A 4.0	A 4.0 Estado de señal 1 — Bobina

- INSTRUCCIÓN CC (LLAMADA CONDICIONADA)

Formato

CC <identificación del bloque lógico>

Descripción de la operación

CC <identificación del bloque lógico> (Llamada condicionada) llama a un bloque lógico del tipo FC o SFC sin parámetros cuando el RLO es 1. La operación CC es prácticamente igual que la operación CALL, con la diferencia de que aquí no se pueden transferir parámetros. La operación almacena en la pila BSTACK la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos actuales y el bit MA; desactiva la dependencia con respecto al MCR; crea el área de datos locales del bloque a llamar y empieza a ejecutar el bloque lógico que se ha llamado. La identificación del bloque lógico puede indicarse tanto de forma absoluta como simbólica.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	0	0	1	1	0

Ejemplo 1

```
AWL             Explicación
U       E 2.0   //Consultar el estado de señal en la entrada E 2.0.
CC      FC6     //Llamar a la función FC6, si E 2.0 es 1.
U       M3.0    //Se ejecuta: tras regresar de la función llamada, si E  2.0  = 1, ó
                inmediatamente después de la instrucción U E 2.0,         si E  2.0  =  0.
```

Ejemplo 2

```
AWL             Explicación
U       E 2.0   //Consultar el estado de señal en la entrada E 2.0.
CC      SFC43   //Llamar a la función de sistema SFC43 (sin parámetros), si E 2.0 es1.
```

Nota

Si se utiliza la operación CALL para llamar a un bloque de función (FB) o a un bloque de función de sistema (SFB), en la instrucción hay que especificar un bloque de datos de instancia (n.° de DB). Los bloques de datos no se pueden asignar en el operando de la instrucción.

Dependiendo de cuál sea el segmento con el que se esté trabajando, durante la compilación del lenguaje de programación Esquema de contactos al lenguaje de programación Lista de instrucciones "KOP/AWL: Programar bloques" genera en parte la operación UC y en parte la operación CC. Por ello, se recomienda utilizar por regla general la instrucción CALL, con el fin de que no se produzcan errores en los programas que el usuario haya creado.

- INSTRUCCIÓN UC (LLAMADA INCONDICIONADA)

Formato

UC <identificación del bloque lógico>

Descripción de la operación

UC <identificación del bloque lógico> (Llamada incondicionada) llama a un bloque lógico del tipo FC o SFC. La operación UC es prácticamente igual a la operación CALL, con la diferencia de que no se pueden transmitir parámetros. La operación almacena en la pila BSTACK la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos actuales y el bit MA; desactiva la dependencia con respecto al MCR, crea el área de datos locales del bloque a llamar y empieza a ejecutar el bloque lógico que se ha llamado.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	0	0	1	-	0

Ejemplo 1

```
AWL             Explicación
UC      FC6     //Llamar a la función FC6 (sin parámetros).
```

Ejemplo 2

```
AWL             Explicación
UC      SFC43   //Llamar a la función de sistema SFC43 (sin parámetros).
```

Nota

Si se utiliza la operación CALL para llamar a un bloque de función (FB) o a un bloque de función de sistema (SFB), en la instrucción hay que especificar un bloque de datos de instancia (n.° de DB). Los bloques de datos no se pueden asignar en el operando de la instrucción.

Según cuál sea el segmento con el que se trabaja, "KOP/AWL: Programar bloques" genera, durante la compilación del lenguaje de programación Esquema de contactos al lenguaje de programación Lista de instrucciones, en parte la operación UC y en parte la operación CC. Por lo general, utilice la operación CALL para que no se produzcan errores en los programas que usted haya creado.

- INSTRUCCIÓN = (ASIGNACIÓN)

Formato

= <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

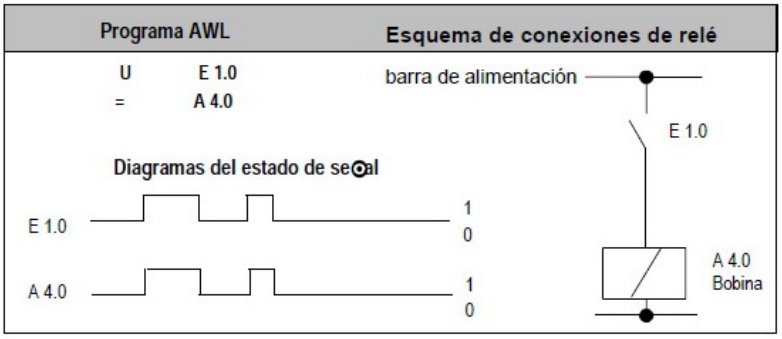
Descripción de la operación

= <bit> escribe el RLO en el bit direccionado si el Master Control Relay está conectado (MCR = 1). Si el MCR es 0, en el bit direccionado se escribe el valor "0" en vez del RLO.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	-	0

Ejemplo



- INSTRUCCIÓN R (DESACTIVAR ó RESET)

Formato

R <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D

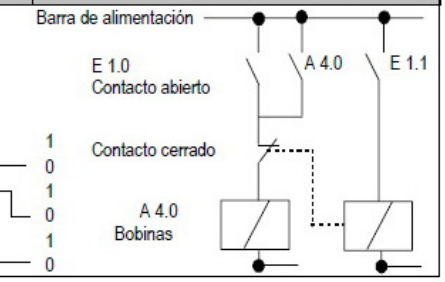
Descripción de la operación

R (Desactivar bit) escribe el valor "0" en el bit direccionado si el RLO es 1 y si el Master Control Relay (MCR = 1) está conectado. Si el MCR es 0, el bit direccionado no varía.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	-	0

Ejemplo

Programa AWL	Esquema de conexiones de relé
U E 1.0 S A 4.0 U E 1.1 R A 4.0 Diagramas del estado de señal E 1.0 E 1.1 A 4.0	

- INSTRUCCIÓN S (ACTIVAR ó SET)

Formato

S <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D

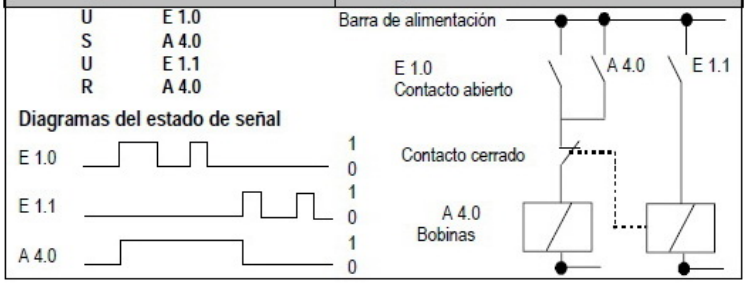
Descripción de la operación

S (Activar bit) escribe el valor "1" en el bit direccionado si el RLO es 1 y si el Master Control Relay (MCR = 1) está conectado. Si el MCR es 0, el bit direccionado no varía.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	-	0

Ejemplo

Programa AWL	Esquema de conexiones de relé
U E 1.0 S A 4.0 U E 1.1 R A 4.0 Diagramas del estado de señal	

- INSTRUCCIÓN L (CARGAR)

Formato

L <operando>

Operando	Tipo de datos	Area de memoria	Dirección fuente
<operando>	BYTE WORD DWORD	E, A, PE, M, L, D, puntero, parámetro	0...65535 0...65534 0...65532

Descripción de la operación

L <operando> carga en el ACU 1 el contenido del byte, de la palabra o de la doble palabra direccionado, después de haberse almacenado el anterior contenido del ACU 1 en el ACU 2 y de haber puesto el ACU 1 a "0".

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	-	-	-	-

Ejemplo

```
AWL             Explicación
L     EB10      //Cargar byte de entrada EB10 en el ACU1-L-L.
L     MB120     //Cargar byte de marcas MB120 en el ACU1-L-L.
L     DBB12     //Cargar byte de datos DBB12 en el ACU1-L-L.
L     DIW15     //Cargar palabra de datos de instancia DIW15 en el ACU1-L.
L     LD252     //Cargar doble palabra de datos locales LD252 en el ACU 1.
L     P# E 8.7  //Cargar puntero en ACU1
L     OTTO      //Cargar parámetro "OTTO" en ACU1
L     P# ANNA   //Cargar puntero en el parámetro indicado en el ACU1 (Este comando carga el
                offset de direcciones relativo del parámetro indicado. Para calcular en FBs
                aptos para multiinstancia el offset absoluto en el bloque de datos de instancia,
                se tiene que sumar a este valor el contenido del registro AR2.
```

- INSTRUCCIÓN T (TRANSFERIR)

Formato

T <operando>

Operando	Tipo de datos	Area de memoria	Dirección fuente
<operando>	BYTE WORD DWORD	E, A, PA, M, L, D	0...65535 0...65534 0...65532

Descripción de la operación

T <operando> transfiere (copia) el contenido del ACU 1 a la dirección de destino si está conectado el Master Control Relay (MCR = 1). Si el MCR es 0, en la dirección de destino se escribe el valor "0". El número de bytes que se copia del ACU 1 dependerá del tamaño indicado en la dirección de destino. El ACU 1 también almacena los datos después de la operación de transferencia. La operación de transferencia a un área de periferia directa (área de memoria PA) también transfiere el contenido del ACU 1 ó "0" (si el MCR es 0) a la dirección correspondiente en la imagen del proceso de las salidas (área de memoria A). La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	-	-	-	-

Ejemplo

```
AWL             Explicación
T     AB10      //Transferir el contenido del ACU1-L-L al byte de salida AB10.
T     MW14      //Transferir el contenido del ACU1-L a la palabra de marcas MW14.
T     DBD2      //Transferir el contenido del ACU 1 a la doble palabra de datos DBD2.
```

- INSTRUCCIÓN SI (TEMPORIZADOR COMO IMPULSO)

Formato

SI <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

Descripción de la operación

SI <temporizador> arranca el temporizador direccionado si el RLO cambia de "0" a "1". El intervalo programado transcurre mientras el RLO sea 1. Si el RLO cambia a "0" antes de que haya transcurrido el intervalo programado, el temporizador se para. Para esta operación (Arrancar temporizador) tienen que estar almacenados el valor de temporización y la base de tiempo en formato BCD en el ACU1-L.

Consulte también Area de memoria y componentes de un temporizador.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo

```
AWL             Explicación
U     E 2.0
FR    T1        //Habilitar el temporizador T1.
U     E 2.1
L     S5T#10s   //Ajustar una preselección de 10 segundos en el ACU 1.
SI    T1        //Arrancar el temporizador T1 como impulso.
U     E 2.2
R     T1        //Poner el temporizador T1 a 0.
U     T1        //Consultar el estado de señal del temporizador T1.
=     A 4.0
L     T1        //Cargar el valor de temporización actual del temporizador T1 como número
                binario.
T     MW10
LC    T1        //Cargar el valor de temporización actual del temporizador T1 en formato BCD.
T     MW12
```

- INSTRUCCIÓN SPA (SALTO INCONDICIONADO)

Formato

SPA <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

Descripción de la operación

SPA <meta> interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado, independientemente de cuál sea el contenido de la palabra de estado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	-	-	-	-

Ejemplo

```
AWL                 Explicación
      U     E 1.0
      U     E 1.2
      SPB   DELE   //Si el RLO es 1 saltar a la meta DELE.
      L     MB10
      INC   1
      T     MB10
      SPA   FORW   //Salto incondicionado a la meta FORW.
DELE: L     0
      T     MB10
FORW: U     E 2.1  //La ejecución del programa continúa aquí después de haber saltado a la meta
                   FORW.
```

- INSTRUCCIÓN SPB (SALTO CONDICIONADO A RLO=1)

Formato

SPB <meta>

Operando	Descripción
<meta >	Nombre simbólico de la meta del salto

Descripción de la operación

Si el RLO es 1, la operación **SPB <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

Si el RLO es 0 no se ejecuta el salto. El RLO se pone a "1" y la ejecución del programa continúa con la instrucción siguiente.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	1	1	0

Ejemplo

```
AWL                 Explicación
      U     E 1.0
      U     E 1.2
      SPB   JOVR   //Saltar a la meta JOVR si RLO = 1
      L     EW8    //La ejecución del programa continúa aquí en caso de que no se ejecute el
                   salto.
      T     MW22
JOVR: U     E 2.1  //La ejecución del programa continúa aquí después de haber saltado a la meta
                   JOVR.
```

- INSTRUCCIÓN SV (TEMPORIZADOR COMO IMPULSO PROLONGADO)

Formato

SV <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

Descripción de la operación

SV <temporizador> arranca el temporizador direccionado si el RLO cambia de "0" a "1". El intervalo programado transcurre aunque el RLO cambie mientras tanto a "0". Si el RLO cambia de "0" a "1" antes de que haya transcurrido el intervalo programado, se vuelve a arrancar el intervalo programado. Para que se ejecute esta orden de arrancar el temporizador tienen que estar almacenados en el ACU1-L el valor de temporización y la base de tiempo en formato BCD.

Consulte también Area de memoria y componentes de un temporizador.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo

```
AWL             Explicación
U     E 2.0
FR    T1        //Habilitar el temporizador T1.
U     E 2.1
L     S5T#10s   //Ajustar una preselección de 10 segundos en el ACU 1.
SV    T1        //Arrancar el temporizador T1 como impulso prolongado.
U     E 2.2
R     T1        //Poner el temporizador T1 a 0.
U     T1        //Consultar el estado de señal del temporizador T1.
=     A 4.0
L     T1        //Cargar el valor de temporización actual del temporizador T1 como número
                binario.
T     MW10
LC    T1        //Cargar el valor de temporización actual del temporizador T1 en formato BCD.
T     MW12
```

- INSTRUCCIÓN CALL (LLAMADA)

Formato

CALL <identificación del bloque lógico>

Descripción de la operación

CALL <identificación del bloque lógico> sirve para llamar tanto a funciones (FC) y a bloques de función (FB), como para llamar a funciones de sistema (SFC) y a bloques de función del sistema (SFB) suministrados por Siemens. La operación CALL llama a la FC/SFC o al FB/SFB que se indica como operando, independientemente de cuál seal el RLO y de condiciones de cualquier otro tipo. Cuando se utiliza CALL para llamar a un FB o a un SFB hay que asignar un bloque de datos de instancia al FB/SFB. Tras editar el bloque llamado el programa continúa la ejecución con el programa del bloque que efectúa la llamada. La identificación del bloque lógico puede indicarse tanto de forma absoluta como simbólica. El contenido de los registros se vuelven a restaurar después de ejecutar la llamada al SFB/a la SFC.

Ejemplo: CALL FB1, DB1 ó CALL FILLVAT1, RECIPE1

Bloque lógico	Tipo de bloque	Sintaxis para la llamada (dirección absoluta)
FC	Función	CALL FCn
SFC	Función de sistema	CALL SFCn
FB	Bloque de función	CALL FBn1,DBn2
SFB	Bloque de función de sistema	CALL SFBn1,DBn2

Nota

Si se está utilizando el editor de AWL, los datos de la tabla anterior, n, n1 ó n2, deben indicar bloques válidos que ya existan. Si se quieren emplear nombres simbólicos hay que definirlos previamente.

Transferir parámetros (utilícese para ello el editor incremental)

El bloque que efectúa la llamada puede intercambiar parámetros con el bloque llamado a través de la lista de variables. La lista de variables se agrega automáticamente al programa AWL en cuestión en cuanto se introduzca correctamente una instrucción CALL.

Si se llama a un FB/SFB o a una FC/SFC, y la tabla de declaración de variables del bloque que se ha llamado tiene declaraciones del tipo IN, OUT e IN_OUT, dichas variables se añaden al programa del bloque que efectúa la llamada en forma de lista de parámetros formales.

Al llamar a las FC y SFC, a los parámetros formales se les tiene que asignar los parámetros actuales del bloque lógico que efectúa la llamada.

Al llamar a los FB y SFB sólo tienen que indicarse los parámetros actuales que vayan a modificarse con respecto a la última llamada, ya que los parámetros actuales se almacenan en el DB de instancia después de que se haya ejecutado el FB. Si el parámetro actual es un DB, se tiene que indicar siempre la dirección absoluta y completa, p.ej. DB1, DBW2.

- FUNCIÓN DE SISTEMA SFC1

Descripción

Con la SFC 1 "READ_CLK" (read system clock) se lee el reloj en la CPU. Se obtienen la fecha y hora actuales.

Parámetro	Declaración	Tipo de datos	Area de memoria	Descripción
RET_VAL	OUTPUT	INT	E, A, M, D, L	Si durante el proceso de la función ocurre un error, el valor de retorno contiene un código de error.
CDT	OUTPUT	DT	D, L	En la salida CDT se emiten la fecha y la hora actuales.

- LLAMAR A UNA FUNCIÓN DE SISTEMA (CALL SFC n)

Formato

CALL SFC n

Nota

Si trabaja con el editor de AWL, la indicación (n) tendrá que refererirse a bloques válidos ya existentes. Los nombres simbólicos también se deberán haber definido previamente.

Descripción

Esta operación permite llamar funciones estándar creadas por el usuario (SFCs). La operación CALL llama la SFC indicada como operando, independientemente del RLO o de cualquier otra condición. Una vez procesado el bloque invocado, el programa del bloque invocante seguirá procesándose. La identificación del bloque lógico puee indicarse de forma absoluta o simbólica.

Transferir parámetros (para ello trabaje con el modo incremental)

El bloque invocante puede intercambiar parámetros con el bloque invocado mediante la tabla de variables. Dicha tabla de variables se actualiza de forma automática en su programa AWL al introducir una instrucción CALL válida.

Si llama una SFC y la tabla de declaración de variables del bloque invocado dispone de declaraciones del tipo IN, OUT e IN_OUT, estas variables se actualizarán en el programa del bloque invocante como tabla de parámetros formales.

Al llamar las SFCs tiene que asignar parámetros actuales del bloque lógico invocante a los parámetros formales.

Los parámetros IN se pueden introducir como constantes o direcciones absolutas o simbólicas. Los parámetros OUT e IN_OUT tienen que introducirse como direcciones absolutas o simbólicas. Vigile que todas las direcciones y constantes sean compatibles con los tipos de datos que se vayan a transferir.

La operación CALL guarda la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos abiertos y el bit MA en la pila BSTACK. Además la operación desactiva la dependencia MCR y crea el área de datos locales del bloque que debe ser llamado.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	0	0	1	-	0

Ejemplo: Llamar a una SFC sin parámetros

AWL		Explicación
CALL	SFC43	//Llama la SFC43 para arrancar de nuevo la supervisión del tiempo (sin parámetros).

CONTROL DE LA VELOCIDAD DE LAS LOCOMOTORAS

La velocidad de los trenes es controlada mediante las salidas analógicas de los dos esclavos de la red PROFIBUS, las CPU S7 226. Esa señal analógica en forma de voltaje será enviada a los mandos de control de los trenes, llevando ya el voltaje deseado que el sistema digital DCC convertirá en una señal eléctrica adecuada (enviándola a través de la vía), para ser decodificada por los decodificadores instalados en las locomotoras y, en consecuencia, ordenado a sus motores que muevan las locomotoras a mayor o menor velocidad. A su vez, las salidas analógicas de los esclavos PROFIBUS serán controladas por el master de la red, el S7 314C-2DP, el que, a través de su programación (o atendiendo a las órdenes generadas en el Scada del PC de control con programación específica para ello), enviará el valor adecuado a la salida analógica para obtener el voltaje de salida deseado.

El rango de salida de tensión de los mandos de control de Roco originales son aproximadamente un valor entre 1V a 3,5V (aprox.), teniendo en cuenta que deben ser acotados para que los trenes no vayan a demasiada velocidad y haya riesgo de descarrilamiento o daños en ellos mismos o en las vías. Según las especificaciones técnicas de los módulos de salidas analógicas EM232 (1 en cada CPU S7 226), pueden sacar valores de tensión de -10V a +10V, pudiendo utilizarlos para obtener señales de tensión bipolar. Sin embargo, en este proyecto no hace falta esa bipolaridad, puesto que los rangos de tensión manejados son pequeños y siempre positivos. De cualquier forma, las señales de salida analógica no se manejan directamente en voltaje utilizando programación. Para programar los valores de salida analógica se utilizan valores numéricos de 16 bits, que pueden ir de -32.000 a +32.000 en el caso de señales bipolares o de 0 a +32.000 en caso de señales unibipolares. En este proyecto, cualquier programación que se haga sobre las salidas analógicas deberá estar comprendido en un rango de valores de 0 a +32.000.

Los mandos de control de los trenes tienen la particularidad de que el valor de tensión en el que las locomotoras se encuentran en velocidad 0 (parados) es de 1,5V. De 1,5 V a 0 V se puede regular la velocidad de las locomotoras en un sentido de la marcha, mientras que de 1,5 V a 3,5 V aproximadamente es el rango de velocidad de las locomotoras en el otro sentido de la marcha. Por motivos de seguridad, y debido a que los rangos máximos de tensión provocan velocidades demasiado altas para el trayecto diseñado, se ha establecido un margen de tensión en un sentido de la marcha de 1,5 V a 1,03 V aproximadamente, y en el otro sentido de la marcha un rango de 1,5 V a 1,97 V aproximadamente.

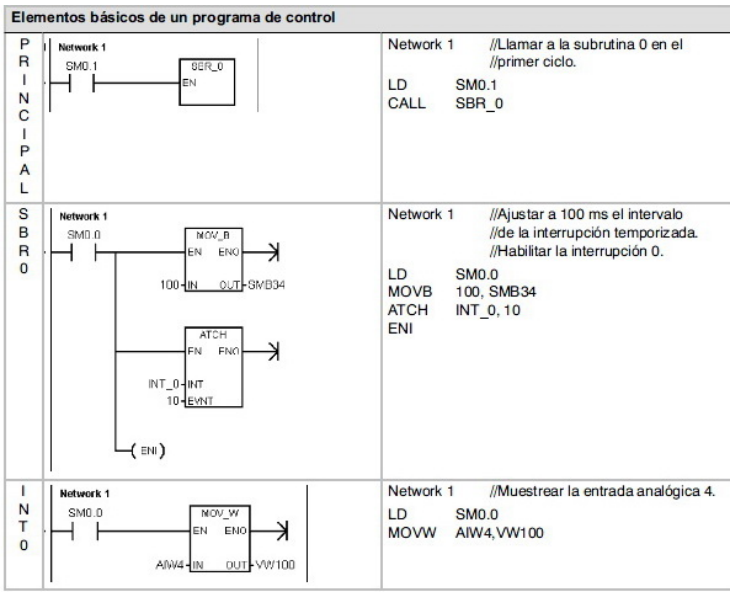
Con todos estos datos, se puede establecer un cálculo mediante una simple regla de tres para obtener el valor de programación adecuado. Por ejemplo, si el valor 32.000 es +10 V de salida en el módulo EM 232, ¿qué valor de programación es el adecuado para obtener +1,5V en la salida? Con esta regla de tres se calcula que para poder dejar parado un tren se necesita un valor de programación de 4800. De la misma forma, el valor de programación 3300 significaría velocidad máxima en un sentido de la marcha y 6300 velocidad máxima en el otro sentido de la marcha.

El S7 314C-2DP, como master de la red PROFIBUS que es, enviará por el buffer de comunicaciones de la red el valor de programación a los dos S7 226 (direcciones #3 y #4), siguiendo la consistencia de datos vista anteriormente. De esta forma, cuando el master quiera ordenar un valor de salida analógica para regular el tren de pasajeros del S7 226 #3, cargará el dato en la marca de palabra MW10 para después transferirlo a la AW0 (buffer de comunicación), que será recibido por el S7 226 #3 en su zona de memoria VW2000. De la misma forma, cuando el master quiera ordenar un valor de salida analógica para regular el tren de mercancías del S7 226 #4, cargará el dato en la marca de palabra MW12 para después transferirlo a la AW8 (buffer de comunicación), que será recibido por el S7 226 #4 en su zona de memoria VW2000.

12.2. Programación de los S7 226

Para la programación de los S7 226 se ha utilizado STEP 7/MicroWIN. En este lenguaje, un bloque de programa se compone del código ejecutable y los comentarios. El código ejecutable comprende el programa principal, así como subrutinas y/o rutinas de interrupción opcionales. El código se compila y se carga en el S7 226, a excepción de los comentarios del programa. Las unidades de organización (programa principal, subrutinas y rutinas de interrupción) sirven para estructurar el programa de control.

La siguiente imagen muestra un programa de ejemplo que incluye una subrutina y una rutina de interrupción. Este programa utiliza una interrupción temporizada para leer el valor de una entrada analógica cada 100 ms:



PROGRAMA PRINCIPAL

El programa principal contiene las operaciones que controlan la aplicación. El S7 226 ejecuta estas operaciones en orden secuencial una vez por ciclo. El programa principal se denomina también OB1, al igual que en el STEP 7.

SUBRUTINAS (OPCIONALES)

Las subrutinas opcionales del programa se ejecutan sólo cuando se llaman desde el programa principal, desde una rutina de interrupción, o bien desde otra subrutina. Las subrutinas son elementos opcionales del programa, adecuándose para funciones que se deban ejecutar repetidamente. Así, en vez de tener que escribir la lógica en cada posición del programa principal donde se deba ejecutar una función, basta con escribirla sólo una vez en una subrutina y llamar a la subrutina desde el programa principal cada vez que sea necesario. Las subrutinas tienen varias ventajas:

- La utilización de subrutinas permite reducir el tamaño total del programa.
- La utilización de subrutinas acorta el tiempo de ciclo, puesto que el código se ha extraído del programa principal. El S7 226 evalúa el código del programa principal en cada ciclo, sin importar si el código se ejecuta o no. Sin embargo, el S7 226 evalúa el código en la subrutina sólo si se llama a ésta. En cambio, no lo evalúa en los ciclos en los que no se llame a la subrutina.
- La utilización de subrutinas crea códigos portátiles. Es posible aislar el código de una función en una subrutina y copiar ésta a otros programas sin necesidad de efectuar cambios o con solo pocas modificaciones.

Como consejo se puede decir que la utilización de direcciones de la memoria V limita la portabilidad de las subrutinas, ya que la asignación de direcciones de un programa en la memoria V puede estar en conflicto con la asignación en un programa diferente. En cambio, las subrutinas que utilizan la tabla de variables locales (memoria L) para todas las asignaciones de direcciones se pueden transportar muy fácilmente, puesto que no presentan el riesgo de conflictos de direcciones entre la subrutina y otra parte del programa.

RUTINAS DE INTERRUPCIÓN (OPCIONALES)

Las rutinas de interrupción opcionales del programa reaccionan a determinados eventos de interrupción. Las rutinas de interrupción se pueden programar para gestionar eventos de interrupción predefinidos. El S7 226 ejecuta una rutina de interrupción cuando ocurre el evento asociado.

El programa principal no llama a las rutinas de interrupción. Una rutina de interrupción se asocia a un evento de interrupción y el S7 226 ejecuta las operaciones contenidas en esa rutina sólo cada vez que ocurra el evento en cuestión. Puesto que no es posible saber con anterioridad cuando el S7 226 generará una interrupción, es recomendable limitar el número de variables utilizadas tanto por la rutina de interrupción como en otra parte del programa.

Se recomienda utilizar la tabla de variables locales de la rutina de interrupción para garantizar que ésta utilice únicamente la memoria temporal, de manera que no se sobrescriban los datos utilizados en ninguna parte del programa.

OTROS ELEMENTOS DEL PROGRAMA

Hay otros bloques que contienen información para el S7 226. A la hora de cargar el programa en la CPU, es posible indicar qué bloques se deben cargar.

Bloques de sistema

El bloque de sistema permite configurar diversas opciones de hardware para el S7 226.

Bloques de datos

En el bloque de datos se almacenan los valores de las diferentes variables (memoria V) utilizadas en el programa. Este bloque se puede usar para introducir los valores iniciales de los datos.

UTILIZAR STEP 7-Micro/WIN PARA CREAR PROGRAMAS

Una vez abierto el programa se puede ver la barra de herramientas, que contiene botones de método abreviado para los comandos de menú de uso frecuente. Estas barras se pueden mostrar u ocultar.

También se puede ver la barra de navegación, que contiene iconos que permiten acceder a las diversas funciones de programación de STEP 7/MicroWIN.

En el árbol de operaciones se visualizan todos los objetos del proyecto y las operaciones para crear el programa de control. Para insertar operaciones en el programa, se puede utilizar el método de arrastrar y soltar desde el árbol de operaciones, o bien hacer doble clic en una operación con objeto de insertarla en el editor de programas.

El editor de programas contiene el programa y una tabla de variables locales donde se pueden asignar nombres simbólicos a las variables locales temporales. Las subrutinas y las rutinas de interrupción se visualizan en forma de fichas en el borde inferior del editor de programas. Para acceder a las subrutinas, a las rutinas de interrupción o al programa principal, hay que hacer doble clic en la ficha en cuestión.

STEP 7-Micro/WIN ofrece tres editores para crear programas: Esquema de contactos (KOP), Lista de instrucciones (AWL) y Diagrama de funciones (FUP). Con algunas restricciones, los programas creados con uno de estos editores se pueden visualizar y editar con los demás.

Funciones del editor AWL

El editor AWL visualiza el programa textualmente. Permite crear programas de control introduciendo la nemotécnica de las operaciones. El editor AWL sirve para crear ciertos programas que, de otra forma, no se podrían programar con los editores KOP ni FUP. Ello se debe a que AWL es el lenguaje nativo del S7 226, a diferencia de los editores gráficos, sujetos a ciertas restricciones para poder dibujar los diagramas correctamente. En la siguiente imagen se puede ver un ejemplo de programación en AWL:

```
LD    I0.0    //Leer una entrada
A     I0.1    //AND con otra entrada
=     Q1.0    //Escribir el valor en la salida 1
```

El S7 226 ejecuta cada operación en el orden determinado por el programa, de arriba abajo, reiniciando después arriba.

AWL utiliza una pila lógica para resolver la lógica de control. El usuario inserta las operaciones AWL para procesar las operaciones de pila.

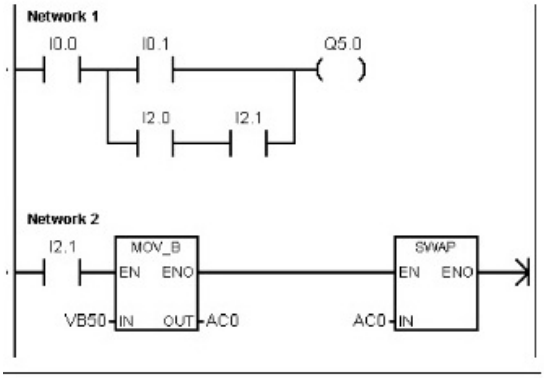
Cuando se elige AWL para programar hay que tener en cuenta lo siguiente:

- El lenguaje AWL es más apropiado para los programadores expertos.
- En algunos casos, AWL permite solucionar problemas que no se podrían resolver fácilmente con los editores KOP o FUP.
- El editor AWL soporta sólo el juego de operaciones SIMATIC.
- En tanto que el editor AWL se puede utilizar siempre para ver o editar programas creados con los editores KOP o FUP, lo contrario no es posible en todos los casos. Los editores KOP o FUP no siempre permiten visualizar un programa que se haya creado en AWL.

Funciones del editor KOP

El editor KOP visualiza el programa gráficamente, de forma similar a un esquema de contactos. Los programas KOP hacen que el programa emule la circulación de corriente eléctrica desde una fuente de alimentación, a través de una serie de condiciones lógicas de entrada que, a su vez, habilitan condiciones lógicas de salida. Los programas KOP incluyen una barra de alimentación izquierda que está energizada. Los contactos cerrados permiten que la corriente circule por ellos hasta el siguiente elemento, en tanto que los contactos abiertos bloquean el flujo de energía.

La lógica se divide en segmentos (“networks”). El programa se ejecuta un segmento tras otro, de izquierda a derecha y luego de arriba abajo. La siguiente figura muestra un ejemplo de un programa KOP:



Las operaciones se representan mediante símbolos gráficos que incluyen tres formas básicas.

Los contactos representan condiciones lógicas de entrada, tales como interruptores, botones o condiciones internas.

Las bobinas representan condiciones lógicas de salida, tales como lámparas, arrancadores de motor, relés interpuestos o condiciones internas de salida.

Los cuadros representan operaciones adicionales, tales como temporizadores, contadores u operaciones aritméticas.

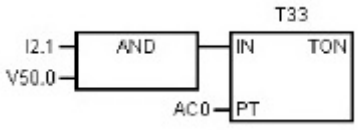
Cuando se elige el editor KOP para programar hay que tener en cuenta lo siguiente:

- El lenguaje KOP facilita el trabajo a los programadores principiantes.
- La representación gráfica es fácil de comprender, siendo popular en el mundo entero.
- El editor KOP se puede utilizar con los juegos de operaciones SIMATIC e IEC 1131-3.
- El editor AWL se puede utilizar siempre para visualizar un programa creado con el editor SIMATIC KOP.

Funciones del editor FUP

El editor FUP visualiza el programa gráficamente, de forma similar a los circuitos de puertas lógicas. En FUP no existen contactos ni bobinas como en el editor KOP, pero sí que hay operaciones equivalentes que se representan en forma de cuadros.

La siguiente figura muestra un ejemplo de un programa FUP:



El lenguaje de programación FUP no utiliza las barras de alimentación izquierda ni derecha. Sin embargo, el término “circulación de corriente” se utiliza para expresar el concepto análogo del flujo de señales por los bloques lógicos FUP.

La ruta “1” lógica por los elementos FUP se denomina circulación de corriente. El origen de una entrada de circulación de corriente y el destino de una salida de circulación de corriente se pueden asignar directamente a un operando.

La lógica del programa se deriva de las conexiones entre las operaciones de cuadro. Así pues, la salida de una operación (por ejemplo, un cuadro AND) se puede utilizar para habilitar otra operación (por ejemplo, un temporizador), con objeto de crear la lógica de control necesaria. Estas conexiones permiten solucionar numerosos problemas lógicos.

Cuando se elige FUP para programar hay que tener en cuenta lo siguiente:

- El estilo de representación en forma de puertas gráficas es apropiado para observar el flujo del programa.
- El editor FUP soporta los juegos de operaciones SIMATIC e IEC 1131-3.
- El editor AWL se puede utilizar siempre para visualizar un programa creado con el editor SIMATIC FUP.

Elección del editor de programación para el proyecto

Como se puede comprobar, el editor AWL proporciona más funcionalidad a la hora de programar, al igual que pasaba en el STEP 7. Por este motivo y porque ya se está familiarizado con el editor AWL, es la opción seleccionada como editor de programación de los dos S7 226 del proyecto.

13. Programación de la aplicación Scada.

La programación de la aplicación Scada se ha realizado con el paquete de software WinCC (Windows Control Center para Windows), que constituye el entorno de desarrollo de Siemens en el marco de los Scadas para visualización y control de procesos industriales.

WinCC es una aplicación IHMI (Integrated Human Machine Interface) que integra el software de controlador de planta en el proceso de automatización. Los componentes de WinCC permiten integrar sin problemas aplicaciones nuevas o ya existentes.

Entre sus características más importantes se pueden resumir en:

- Arquitectura de desarrollo abierta (programación en C).
- Soporte de tecnologías Active X.
- Comunicación con otras aplicaciones vía OPC.
- Comunicación sencilla mediante drivers (código que implementa el protocolo de comunicaciones con un determinado equipo inteligente) implementados.

Programación online: no es necesaria detener la runtime del desarrollo para poder actualizar las modificaciones en la misma.

WinCC combina la arquitectura de las aplicaciones de Windows con la programación de entornos gráficos, e incluye elementos destinados al control y supervisión de procesos.

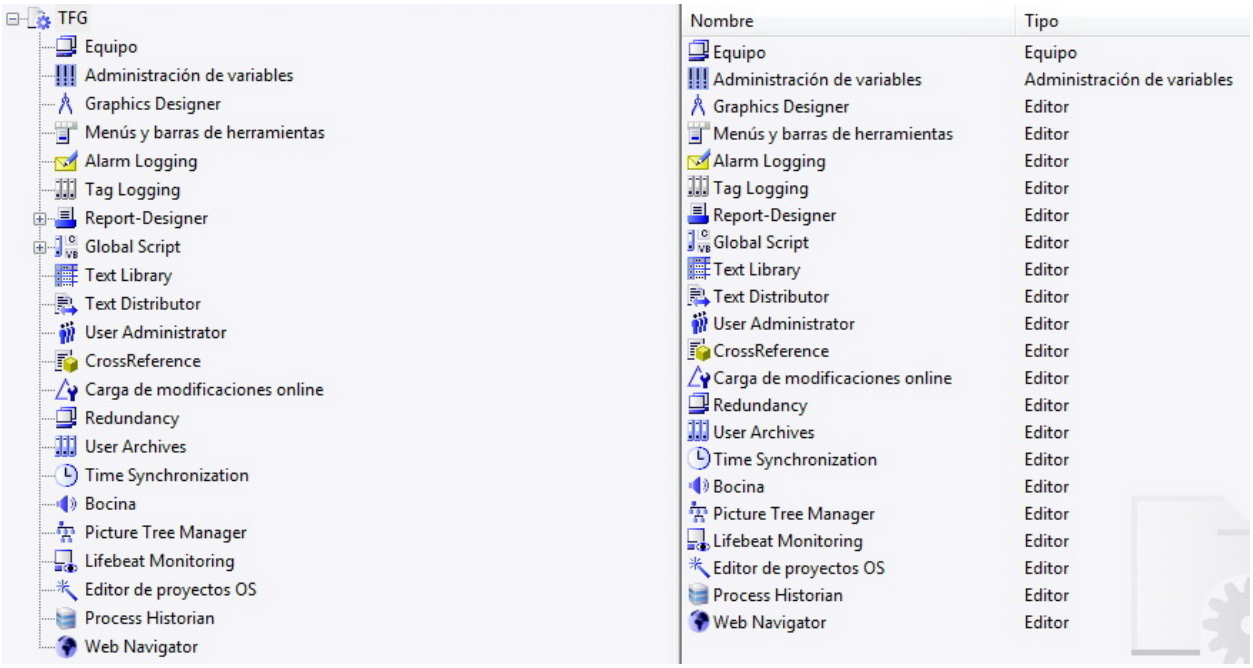
El entorno de ingeniería de proyectos de WinCC engloba:

- Dibujos → Para diseñar representaciones de planta.
- Estructura de archivos → Para guardar datos/eventos marcados con fecha y hora en una base de datos SQL Server.
- Generador de informes → Para generar informes sobre los datos solicitados.
- Administración de datos → Para definir y recopilar datos de toda la planta.
- Tiempo de ejecución de WinCC.

Permite a los operarios interactuar con la aplicación directamente en la máquina o desde un centro de control.

13.1. Crear un nuevo proyecto en Wincc

Para realizar el TFG se ha creado un nuevo proyecto en Wincc que se llama “TFG.MCP” (la extensión “.mcp” (Master Control Program)), dispone de los siguientes elementos una vez creado:



13.2. El explorador de WinCC

El explorador de WinCC representa el acceso a todas las opciones del WinCC como sistema de desarrollo para visualización de procesos industriales. Desde el explorador de WinCC se dispone de todos los módulos software para la creación de ventanas gráficas, archivos de procesos, ventanas de alarmas y generación de documentos a impresora.
La estructura del proyecto, visto desde el explorador de WinCC, contiene las siguientes partes (las más importantes):

- El nombre del proyecto, en este caso, TFG.
- Equipo: Todos los parámetros relacionados con el entorno de trabajo de la aplicación en general.
- Administración de variables: Se administran todas las variables (tags) del proyecto, tanto las internas como las de proceso, por lo que en este apartado se administran también las comunicaciones (utilizadas para los tags de proceso).
- Estructuras de variables: Estructuras de datos o tags de diferentes formatos, tamaños o procedencias (de comunicaciones o internas), cuya relación entre ellos viene definida por una funcionalidad común de cara al proceso.
- Graphics Designer: Editor gráfico que permite dibujar las pantallas que componen la aplicación Scada realizada.
- Alarm Logging: Editor de alarmas que permite configurar las ventanas y tratamiento de alarmas del proceso.
- Report Designer: Editor de informes a impresora. Se encarga de configurar todo lo referente al envío a impresora de informes.
- Global Script: Compilador en C que permite programar acciones propias y ejecutarlas de manera periódica o mediante eventos de cambio de variables.
- Text Library: Editor de texto que permite asignar diferentes configuraciones de textos según el idioma seleccionado en el WinCC.
- User Administrator: Administrador de usuarios que permite activar o desactivar usuarios mediante activación de passwords.

- Cross Reference: Referencias cruzadas de los diferentes componentes de la runtime del proyecto. Mediante este módulo se puede conocer en qué pantallas o funciones de proyecto se utiliza una determinada variable.

13.3. Agregar un driver de PLC

Hay que definir qué dispositivo se va a utilizar como interfaz de comunicación con la instalación ferroviaria. Un driver es un interfaz entre el PLC (Controlador Lógico Programable) y WinCC. El driver seleccionado depende del PLC utilizado. La familia de PLCs Siemens es SIMATIC y abarca entre unas pocas y varios miles de entradas/salidas.

¿Qué es un driver o canal de comunicaciones?

Un driver de comunicaciones es una dll con extensión *.CHN que posibilita al WinCC comunicarse mediante un determinado protocolo con un tipo determinado de PLC industrial o aplicación software. Un canal de comunicaciones puede soportar varios enlaces de comunicaciones a la vez o no, dependiendo del tipo de canal. Existen canales de comunicaciones que, pese a encontrarse dentro del CD de WinCC, necesitan para funcionar una licencia independiente. Es posible generar un nuevo canal de comunicaciones si se dispone de la herramienta CDK, paquete de desarrollo de WinCC para canales de comunicaciones.

Para realizar el proyecto se necesita agregar el driver de PLC Siemens S7 314C-2DP. A continuación, se debe seleccionar la unidad de canal PROFIBUS y crear una “Nueva conexión”. Con las “Propiedades del enlace”, se cambia el nombre por “ProfibusTREN”, hay que establecer las propiedades.

El parámetro más importante de las propiedades, es la dirección de estación. La dirección de la estación debe ser la 2, que se corresponderá con el PLC S7 314C-2DP, maestro de la red PROFIBUS. WinCC se comunicará directamente con esta CPU, disponiendo entonces de conexión en toda la red PROFIBUS integrada por el master y los dos esclavos S7 226 (direcciones #3 y #4).

13.4. Creación de tags

En el proyecto Scada pueden utilizarse dos tipos de tags:

- **Tags internos:** son asignaciones de memoria dentro de WinCC que cumplen la misma funcionalidad que un PLC real, aunque sin comunicación real. Pueden calcularse y modificarse dentro de WinCC, aunque pueden almacenar el resultado de una operación matemática obtenida a partir de variables de comunicaciones. Los tags internos son ilimitados, por los que se pueden crear los que se precise (en función de la memoria RAM del PC) sin coste económico adicional.
- **Tags de proceso:** son asignaciones de memoria dentro del PLC conectado al proceso (o de un dispositivo similar). Por lo tanto, son aquellas cuyo valor se obtiene de la comunicación entre el WinCC y cualquier red de PLC’s o aplicación. La licencia de WinCC varía económicamente con este tipo de tags, puesto que Siemens pone un precio a cierta cantidad de tags que se pueden utilizar. Por ejemplo, Siemens tiene un precio definido para la utilización como máximo de 128 tags, sin embargo, en este proyecto se han utilizado 220 tags, por lo que esa licencia sería insuficiente. Para este proyecto se ha adquirido el siguiente nivel de licencia que Siemens comercializa, en este caso, 512 tags.

Realmente un tag es el elemento de enlace entre la base de datos del WinCC, las variables del PLC y los objetos del runtime de la aplicación. Los valores de los tag son almacenados en la base de datos del proyecto. Cuando arranca el WinCC, carga estos valores de la base de datos. A partir de ese momento se pueden modificar dichos valores, pero hay que tener en cuenta que dichas dinamizaciones no se almacenan en la base de datos, por lo que, si se utilizan variables internas para realizar una receta de valores, y se sale de WinCC, cargará al arrancar de nuevo los valores de las variables internas que tenga asignados en propiedades estáticas, y no el último valor que se hubiese introducido.

Los tags internos se pueden crear fácilmente y luego reasignarse a un PLC real. Los de proceso están monitorizando los datos del proceso de automatización. Para que las variables de comunicaciones puedan ser almacenadas es imprescindible añadir en primer lugar el driver de comunicaciones correspondiente, en este caso, PROFIBUS, mientras que para las variables internas no es necesario implementar driver de comunicaciones.

En el proyecto diseñado, el canal de comunicación PROFIBUS transferirá (bidireccionalmente) el valor de las variables vía la conexión PROFIBUS (utilizando el recurso de comunicación CP5622, tarjeta de comunicaciones PROFIBUS DP instalada en el PC).

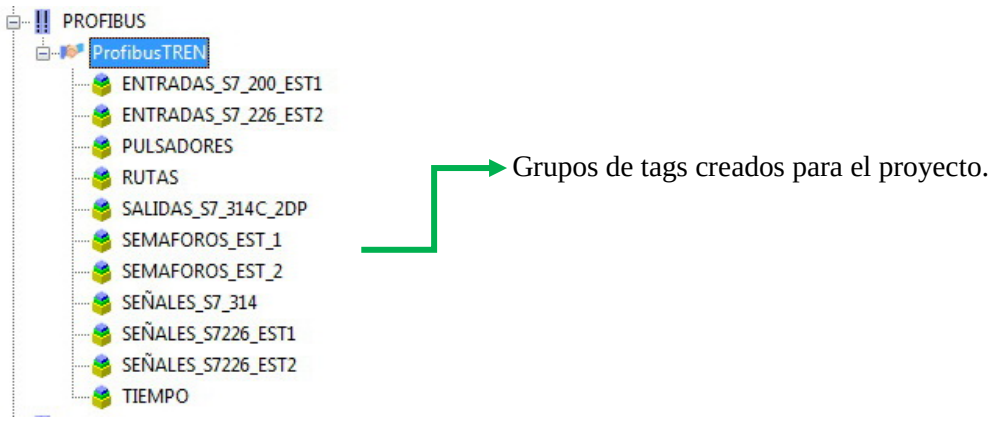
Los tags se pueden organizar en grupos o crearse individualmente. Los grupos sirven para estructurar los tags y así obtener una mayor claridad y lograr acceder a las variables de una forma más intuitiva. Aunque no es imprescindible crear grupos de tags, en el proyecto se han creado varios grupos:

13.4.1. Creación de un grupo de tags

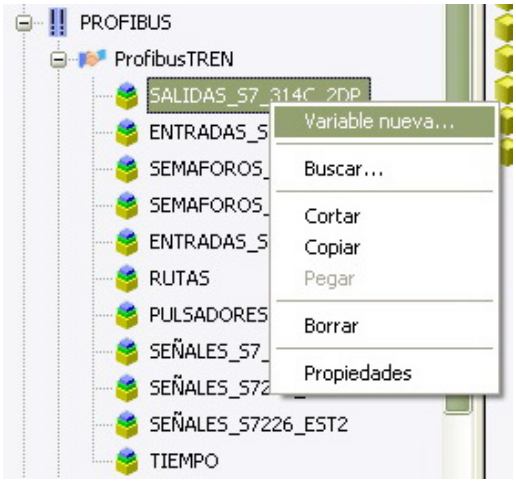
Se pueden crear tantos grupos de tags como se quiera, y cada uno de ellos puede estar formado por todos los tags que sean precisos.

Para crear un nuevo grupo, hacer clic mediante el botón derecho del ratón en la conexión creada del PLC, en este caso, ProfibusTREN. En el menú contextual que aparece, hacer clic en “Grupo nuevo”.

Los grupos de tags creados de esta forma aparecen en la subventana izquierda bajo la conexión del PLC ProfibusTREN:



Para añadir un nuevo tag al grupo sólo hay que hacer clic con el botón derecho del ratón en el icono del grupo y seleccionar “Variable nueva…”:



13.4.2. Creación de tags internos

En primer lugar, habrá que desplegar el nodo “Administración de variables” en el explorador de WinCC si se tiene cerrado. Después hay que pinchar con el botón derecho del ratón sobre “Variables internas” y seleccionar “Variable nueva”, como se muestra en la siguiente imagen:

A continuación, en el cuadro de diálogo “Propiedades de variable”, hay que asignar un nombre al tag elegir el tipo de datos que se necesite.

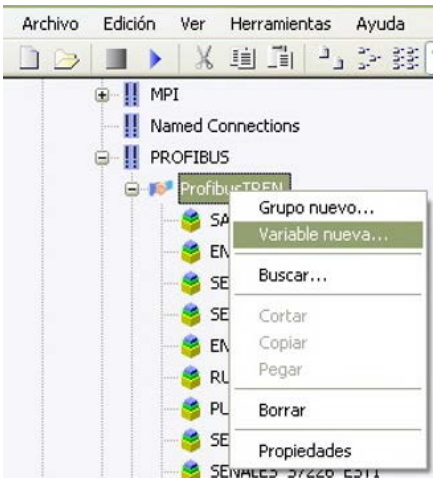
Todos los tags internos creados aparecen en la subventana derecha de la ventana del explorador de WinCC. Con esto, ya se pueden crear los tags internos que se necesiten repitiendo este proceso. También es posible copiar, cortar y pegar tags

13.4.3. Creación de tags de proceso

Para poder crear una variable de proceso es necesario, instalar antes un driver y crear una conexión.

Se pueden copiar y pegar en la conexión creada los tags utilizando los mismos métodos que con los tags internos (teniendo en cuenta que solamente se pueden asignar tags a una conexión mediante los comandos “Copiar” y “Pegar”, y no se puede hacer clic en el icono de un tag y arrastrarlo a una conexión.

Para crear un tag de proceso, hacer clic con el botón derecho del ratón sobre la conexión de PLC creada, en este caso, ProfibusTREN, y seleccionar después "Variable nueva…". Esto también se puede hacer sobre cualquier grupo de tags creado, teniendo en cuenta que, si se hace sobre un grupo de tags, la variable creada estará dentro de ese grupo:



En el cuadro de diálogo que aparecerá, "Propiedades de variable", hay que ponerle un nombre ("demo" en la siguiente imagen). A continuación, se selecciona el tipo de datos adecuado en la opción "Ajuste de formato".

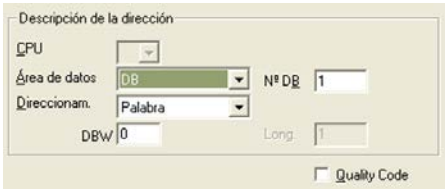
El tamaño de la variable a utilizar puede ser:

- Binary tag: Un bit.
- Signed 8 bit value: Un byte con signo (-128 a 127).
- Unsigned 8 bit value: Un byte sin signo (0 a 255).
- Signed 16 bit value: Una palabra con signo (-32768 a 32767).
- Unsigned 16 bit value: Una palabra sin signo (0 a 65535).
- Signed 32 bit value: Una doble palabra con signo (-2147483647 a 2147483647)
- Unsigned 32 bit value: Una doble palabra sin signo (0 a 4294967295)
- Floating Point 32 bits: Una doble palabra en coma flotante de 32 bits de resolución.

- Floating Point 64 bits: Una doble palabra en coma flotante de 64 bits de resolución.
- Text tag 16 bit Character Set: Una cadena de texto de la longitud que se requiera de caracteres de 8 bits (caracteres ASCII).
- Text tag 16 bit Character Set: Una cadena de texto de la longitud que se requiera de caracteres de 16 bits (caracteres Unicode).
- Raw data type: Telegrama de datos que no es tratado por el procesador del PLC.
- Text Reference: Un puntero a una cadena de texto que se encuentra en el Text Library. Asociándole a la variable el número identificador del Text Library, soporta el valor de la cadena de texto que se defina allí.
- Structure Types: Una estructura es un conjunto de variables de igual o diferentes tamaños agrupadas según una determinada propiedad que las relaciona. Para poder seleccionar una propiedad en esta pestaña es necesario haber generado anteriormente la estructura en Data Types.

Especificación de la dirección de enlace con el PLC:

En el cuadro de diálogo de la imagen anterior hay un botón que pone “Seleccionar”, al lado derecho de “Dirección”, que se debe presionar para establecer la dirección en el PLC (en este caso el S7 314C-2DP) que será “asociada” con el tag que se está creando. Cuando se presiona el botón “Seleccionar”, aparece el siguiente cuadro de diálogo:

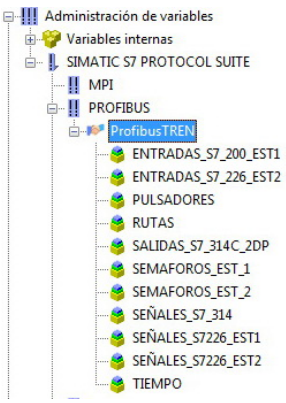


Aquí será necesario ajustar el tipo de área de datos, cuyas opciones (aparecerán desplegando el menú de selección) dependerán del formato de dato seleccionado anteriormente. Los más usuales son: DB, Marca, Entrada, Salida, Contador, Temporizadores.

Lo mismo ocurre con el dato “Direccionamiento”, cuyas opciones dependerán del tipo de formato seleccionado y también del área de datos seleccionada.

El poder seleccionar estos mapas de memoria permite realizar controles de instalaciones que ya estuvieran funcionando con su programa de PLC siempre y cuando las zonas a las que se acceda para escribir no estén a su vez siendo escritas desde el programa de PLC. Con respecto a la lectura no existe evidentemente ninguna limitación, accediendo a todas ellas libremente.

A continuación, se expone la lista completa de grupos de tags de variables de proceso que se han utilizado en el proyecto:



- **SALIDAS _S7_314C-2DP** → En este grupo se han creado las variables que están conectadas a las salidas digitales del S7 314C-2DP, master de la red PROFIBUS, y que van a ser monitorizadas y controladas por el Scada. La lista de variables de este grupo es la siguiente:

Variables [SALIDAS_S7_314C_2DP]

	Nombre	Tipo datos	Dirección
1	Desvio1	Variable binaria	A124.0
2	Desvio2	Variable binaria	A124.1
3	Desvio3	Variable binaria	A124.2
4	Desvio4	Variable binaria	A124.3
5	Desvio5	Variable binaria	A124.4
6	Desvio6	Variable binaria	A124.5
7	Semaforo1_Rojo	Variable binaria	A125.1
8	Semaforo1_Verde	Variable binaria	A125.2
9	Semaforo2_Rojo	Variable binaria	A125.3
10	Semaforo2_Verde	Variable binaria	A125.4
11			

El S7-314 tiene dos semáforos a su cargo, que son el semáforo 9 y 10 de la estación 2 en la red general de semáforos de la instalación. En la nomenclatura de salidas del S7 314 el semáforo 1 se corresponde con el semáforo 9 de la red general y el semáforo 2 del S7 314 se corresponde con el semáforo 2 de la red general. Todos los semáforos están distribuidos en dos señales que se corresponden con las salidas digitales que activan o desactivan los leds de cada uno de ellos (rojo y verde).

En cuanto a los desvíos, que son controlados a través de los relés, se puede asociar la salida digital correspondiente (todos ellos dependen directamente del S7 314). Sólo se necesita de una salida digital del autómata (aunque el relé tenga conectados dos contactos), puesto que se evalúa con la salida del autómata si la salida digital está a 1 o a 0 (estando el desvío recto cuando la salida digital esté a 0 o el desvío curvo cuando la salida digital esté a 1).

Por supuesto, se muestra el valor de dirección absoluta del PLC con la que se ha asociado la variable.

- **ENTRADAS_S7_200_EST1**→ En este grupo se han creado las variables que están conectadas a las entradas digitales del S7 226 EST1(#3), un esclavo de la red PROFIBUS, y que van a ser monitorizadas por el Scada. La lista de variables de este grupo es la siguiente:

Variables [ENTRADAS_S7_200_EST1]

	Nombre	Tipo datos	Dirección
1	Sensor0Est1	Variable binaria	M18.0
2	Sensor1Est1	Variable binaria	M18.1
3	Sensor2Est1	Variable binaria	M18.2
4	Sensor3Est1	Variable binaria	M18.3
5	Sensor4Est1	Variable binaria	M18.4
6	Sensor5Est1	Variable binaria	M18.5
7	Sensor6Est1	Variable binaria	M18.6
8	Sensor7Est1	Variable binaria	M18.7
9	Sensor8Est1	Variable binaria	M19.0
10	Sensor9Est1	Variable binaria	M19.1
11			

Los sensores de todo el trayecto ferroviario son controlados por los S7 226. El S7 226 #3 (EST1) tiene conectados a sus entradas digitales los sensores inductivos que controlan el entorno de la estación 1. Como se puede ver, no se declaran las variables correspondiéndose directamente con la entrada digital del S7 226 #3, puesto que éste es un esclavo de la red PROFIBUS y el Scada no se comunica directamente con esta estación. Las direcciones asignadas a los sensores son marcas pertenecientes al S7 314 (master), el cual utilizará el buffer de comunicaciones existente con la estación esclava para intercambiar los datos del estado de los sensores (entradas digitales) con ella. Cada vez que el S7 314 requiera el estado de los sensores, se pondrá en comunicación con las estaciones esclavas y éstas le proporcionaran el estado de todas las entradas digitales, las cuales el S7 314 dejará disponibles en las marcas que se pueden ver en la imagen. Finalmente, estas variables del Scada son asociadas a esas marcas, por lo que en todo momento se puede saber si cualquiera de los sensores ha sido activado o no. Las salidas digitales sí pueden ser leídas y escritas desde el Scada, sin embargo, los sensores, al tratarse de entradas digitales, sólo pueden

ser leídas, ya que la activación de un sensor o no depende de si eléctricamente se activa o no, y esto sólo ocurre cuando los trenes pasan lo suficientemente cerca del sensor como para activarlo.

- **SEMÁFOROS_EST_1→** En este grupo se han creado las variables que están conectadas a las salidas digitales del S7 226 EST1(#3), un esclavo de la red PROFIBUS, y que van a ser monitorizadas y controladas por el Scada. La lista de variables de este grupo es la siguiente:

Variables [SEMAFOROS_EST_1]

	Nombre	Tipo datos	Dirección
1	ALL_SEMAF_EST1	Valor de 16 bits sin signo	MW16
2	BUFFER_BYTE1_SEMAF_EST1_1	Valor de 8 bits sin signo	AB6
3	BUFFER_BYTE2_SEMAF_EST1_1	Valor de 8 bits sin signo	AB7
4	BYTE1_SEMAF_EST1	Valor de 8 bits sin signo	MB16
5	BYTE2_SEMAF_EST1	Valor de 8 bits sin signo	MB17
6	Semaforo1_Rojo_Est1	Variable binaria	M16.0
7	Semaforo1_Verde_Est1	Variable binaria	M16.1
8	Semaforo2_Rojo_Est1	Variable binaria	M16.2
9	Semaforo2_Verde_Est1	Variable binaria	M16.3
10	Semaforo3_Rojo_Est1	Variable binaria	M16.4
11	Semaforo3_Verde_Est1	Variable binaria	M16.5
12	Semaforo4_Rojo_Est1	Variable binaria	M16.6
13	Semaforo4_Verde_Est1	Variable binaria	M16.7
14	Semaforo5_Rojo_Est1	Variable binaria	M17.0
15	Semaforo5_Verde_Est1	Variable binaria	M17.1
16	Semaforo6_Rojo_Est1	Variable binaria	M17.2
17	Semaforo6_Verde_Est1	Variable binaria	M17.3
18	Semaforo7_Rojo_Est1	Variable binaria	M17.4
19	Semaforo7_Verde_Est1	Variable binaria	M17.5
20	Semaforo8_Rojo_Est1	Variable binaria	M17.6
21	Semaforo8_Verde_Est1	Variable binaria	M17.7
22			

Las salidas digitales del S7 226 EST1 (#3) están ocupadas con las señales de los semáforos que influyen en el entorno de la estación 1. Como se puede ver nada más observar los nombres de las variables, es fácilmente identificable a qué semáforo y color corresponde, siendo todas las variables de tipo variable binaria, excepto las cinco últimas. La variable “ALL_SEMAF_EST1” corresponde a la Word (palabra) donde se almacena el valor que toman los 16 bits de las salidas digitales del S7 226 #3, por lo que esta variable es del tipo valor de 16 bits sin signo. La variable “BYTE1_SEMAF_EST1” corresponde al byte donde se almacena el valor que toman los 8 bits de las salidas digitales del byte numerado 0 del S7 226 #3, por lo que esta variable es del tipo valor de 8 bits sin signo. La variable “BYTE2_SEMAF_EST1” corresponde al byte donde se almacena el valor que toman los 8 bits de las salidas digitales del byte numerado 1 del S7 226 #3, por lo que esta variable es del tipo valor de 8 bits sin signo. La variable “BUFFER_BYTE1_SEMAF_EST1_1” se utiliza para acceder directamente al buffer de comunicación con el S7 226 #3 para intercambiar los datos del byte 0 numerado de salidas digitales del S7 226. Lo mismo con la variable “BUFFER_BYTE2_SEMAF_EST1_1” pero con el byte 1 numerado de salidas digitales del S7 226.

Todas las variables se corresponden con marcas de memoria del S7 314, y no con las salidas digitales de los S7 226 directamente. La explicación es la existencia del buffer de comunicaciones PROFIBUS existente entre el S7 314 y el S7 226 #3.

- **SEMÁFOROS_EST_2→** En este grupo se han creado las variables que están conectadas a las salidas digitales del S7 226 EST2(#4), un esclavo de la red PROFIBUS, y que van a ser monitorizadas y controladas por el Scada. La lista de variables de este grupo es la siguiente:

Variables [SEMAFOROS_EST_2]

	Nombre	Tipo datos	Dirección
1	ALL_SEMAF_EST2	Valor de 16 bits sin signo	MW24
2	BUFFER_BYTE1_SEMAF_EST2	Valor de 8 bits sin signo	AB14
3	BUFFER_BYTE2_SEMAF_EST2	Valor de 8 bits sin signo	AB15
4	BYTE1_SEMAF_EST2	Valor de 8 bits sin signo	MB24
5	BYTE2_SEMAF_EST2	Valor de 8 bits sin signo	MB25
6	Semaforo1_Rojo_Est2	Variable binaria	M24.0
7	Semaforo1_Verde_Est2	Variable binaria	M24.1
8	Semaforo2_Rojo_Est2	Variable binaria	M24.2
9	Semaforo2_Verde_Est2	Variable binaria	M24.3
10	Semaforo3_Rojo_Est2	Variable binaria	M24.4
11	Semaforo3_Verde_Est2	Variable binaria	M24.5
12	Semaforo4_Rojo_Est2	Variable binaria	M24.6
13	Semaforo4_Verde_Est2	Variable binaria	M24.7
14	Semaforo5_Rojo_Est2	Variable binaria	M25.0
15	Semaforo5_Verde_Est2	Variable binaria	M25.1
16	Semaforo6_Rojo_Est2	Variable binaria	M25.2
17	Semaforo6_Verde_Est2	Variable binaria	M25.3
18	Semaforo7_Rojo_Est2	Variable binaria	M25.4
19	Semaforo7_Verde_Est2	Variable binaria	M25.5
20	Semaforo8_Rojo_Est2	Variable binaria	M25.6
21	Semaforo8_Verde_Est2	Variable binaria	M25.7
22			

Las salidas digitales del S7 226 EST2 (#4) están ocupadas con las señales de los semáforos que influyen en el entorno de la estación 2. Como se puede ver, es fácilmente identificable a qué semáforo y color corresponde, siendo todas las variables de tipo variable binaria, excepto las cinco últimas. La variable "ALL_SEMAF_EST2" corresponde a la Word (palabra) donde se almacena el valor que toman los 16 bits de las salidas digitales del S7 226 #4, por lo que esta variable es del tipo valor de 16 bits sin signo. La variable "BYTE1_SEMAF_EST2" corresponde al byte donde se almacena el valor que toman los 8 bits de las salidas digitales del byte numerado 0 del S7 226 #4, por lo que esta variable es del tipo valor de 8 bits sin signo. La variable "BYTE2_SEMAF_EST2" corresponde al byte donde se almacena el valor que toman los 8 bits de las salidas digitales del byte numerado 1 del S7 226 #4, por lo que esta variable es del tipo valor de 8 bits sin signo. La variable "BUFFER_BYTE1_SEMAF_EST2" se utiliza para acceder directamente al buffer de comunicación con el S7 226 #4 para intercambiar los datos del byte 0 numerado de salidas digitales del S7 226. Lo mismo con la variable "BUFFER_BYTE2_SEMAF_EST2" pero con el byte 1 numerado de salidas digitales del S7 226.

Todas las variables se corresponden con marcas de memoria del S7 314, y no con las salidas digitales de los S7 226 directamente. La explicación es la existencia del buffer de comunicaciones PROFIBUS existente entre el S7 314 y el S7 226 #4.

- **ENTRADAS_S7_200_EST2→** En este grupo se han creado las variables que están conectadas a las entradas digitales del S7 226 EST2(#4), un esclavo de la red PROFIBUS, y que van a ser monitorizadas por el Scada. La lista de variables de este grupo es la siguiente:

Variables [ENTRADAS_S7_226_EST2]

	Nombre	Tipo datos	Dirección
1	Sensor0Est2	Variable binaria	M14.0
2	Sensor1Est2	Variable binaria	M14.1
3	Sensor2Est2	Variable binaria	M14.2
4	Sensor3Est2	Variable binaria	M14.3
5	Sensor4Est2	Variable binaria	M14.4
6	Sensor5Est2	Variable binaria	M14.5
7	Sensor6Est2	Variable binaria	M14.6
8	Sensor7Est2	Variable binaria	M14.7
9	Sensor8Est2	Variable binaria	M15.0
10	Sensor9Est2	Variable binaria	M15.1
11			

Se ha comentado anteriormente que los sensores de todo el trayecto ferroviario son controlados por los S7 226. El S7 226 #4 (EST2) tiene conectados a sus entradas digitales los sensores inductivos que controlan el entorno de la estación 2. Como se puede ver, no se declaran las variables correspondiéndose directamente con la entrada digital del S7 226 #4, puesto que éste es un esclavo de la red PROFIBUS y el

Scada no se comunica directamente con esta estación. Las direcciones asignadas a los sensores son marcas pertenecientes al S7 314 (master), el cual utilizará el buffer de comunicaciones existente con la estación esclava para intercambiar los datos del estado de los sensores (entradas digitales) con ella. Cada vez que el S7 314 requiera el estado de los sensores, se pondrá en comunicación con las estaciones esclavas y éstas le proporcionaran el estado de todas las entradas digitales, las cuales el S7 314 dejará disponibles en las marcas que se pueden ver en la imagen. Finalmente, estas variables del Scada son asociadas a esas marcas, por lo que en todo momento se puede saber si cualquiera de los sensores ha sido activado o no. Las salidas digitales sí pueden ser leídas y escritas desde el Scada, sin embargo, los sensores, al tratarse de entradas digitales, sólo pueden ser leídas, ya que la activación de un sensor o no depende de si eléctricamente se activa o no, y esto sólo ocurre cuando los trenes pasan lo suficientemente cerca del sensor como para activarlo.

- **RUTAS→** En este grupo se han creado las variables que regularán las rutas, tanto su permiso como su selección. Estos permisos y selecciones de ruta las efectuará el Jefe de Control del CTC desde las pantallas de la aplicación Scada, que transmitirá esta información al master de la red Profibus para que actúe en consecuencia. De esta forma:
    - La variable “RutaSeleccionada” es el byte que almacena la ruta que se ha seleccionado.
    - La variable “PermisoTodasRutas” es el byte que almacena si el CTC da su permiso para efectuar la ruta o no.
    - La variable “BucleRutasSeleccionado” es el byte que almacena que bucle de rutas se ha seleccionado para ser efectuado.
    - La variable “PermisoTodosBucles” es el byte que almacena los permisos del CTC para efectuar un bucle de rutas o no.
    - Las variables “PermisoRuta1”, “PermisoRuta2” y “PermisoBucle1” son los bits (de los bytes respectivos vistos anteriormente) que almacenan si el CTC ha dado su permiso o no para efectuar la ruta o el bucle de rutas en cuestión.
    - Las variables “Ruta1Seleccionada”, “Ruta2Seleccionada” y “BucleRuta1Seleccionado” son los bits (de los respectivos bytes vistos anteriormente) que almacenan si la ruta o el bucle de rutas en cuestión ha sido seleccionado para ser efectuado.

Variables [RUTAS]

	Nombre	Tipo datos	Dirección
1	BucleRuta1Seleccionadc	Variable binaria	M53.0
2	BucleRutasSeleccionado	Valor de 8 bits sin signo	MB53
3	PermisoBucle1	Variable binaria	M51.0
4	PermisoRuta1	Variable binaria	M50.1
5	PermisoRuta2	Variable binaria	M50.2
6	PermisoTodasRutas	Valor de 8 bits sin signo	MB50
7	PermisoTodosBucles	Valor de 8 bits sin signo	MB51
8	Ruta1Seleccionada	Variable binaria	M52.0
9	Ruta2Seleccionada	Variable binaria	M52.1
10	RutaSeleccionada	Valor de 8 bits sin signo	MB52
11			

- **PULSADORES →** Este grupo ha sido creado para tratar todos los pulsadores del sistema, por lo tanto, los modos de funcionamiento del sistema (sistemaOn, modo automático, manual, manual libre, seta de emergencia, tren de mercancías activo y tren de pasajeros activo). Las siguientes figuras pertenecen a la configuración de la variable SistemaOn, siendo el resto de variables del grupo de configuración similar (cambiando sólo el nombre y direccionamiento):

Variables [PULSADORES]

	Nombre	Tipo datos	Dirección
1	Emergencia	Variable binaria	E126.7
2	SistemaAuto	Variable binaria	M2.0
3	SistemaAutomatico	Variable binaria	E126.4
4	SistemaManual	Variable binaria	M2.1
5	SistemaManualLibre	Variable binaria	M2.3
6	SistemaON	Variable binaria	M1.0
7	SistManual	Variable binaria	E126.3
8	TrenMercanciasActivo	Variable binaria	M1.2
9	TrenPasajerosActivo	Variable binaria	M1.1
10			

- **SEÑALES_S7_314 →** Este grupo se ha creado para crear las variables asociadas a todas las entradas y salidas digitales del S7 314C-2DP (todas en conjunto). Todas las variables son de tipo variable binaria, poniendo los nombres y direccionamiento adecuado.

Variables [SEÑALES_S7_314]

	Nombre	Tipo datos	Dirección
1	Entrada124_0	Variable binaria	E124.0
2	Entrada124_1	Variable binaria	E124.1
3	Entrada124_2	Variable binaria	E124.2
4	Entrada124_3	Variable binaria	E124.3
5	Entrada124_4	Variable binaria	E124.4
6	Entrada124_5	Variable binaria	E124.5
7	Entrada124_6	Variable binaria	E124.6
8	Entrada124_7	Variable binaria	E124.7
9	Entrada125_0	Variable binaria	E125.0
10	Entrada125_1	Variable binaria	E125.1
11	Entrada125_2	Variable binaria	E125.2
12	Entrada125_3	Variable binaria	E125.3
13	Entrada125_4	Variable binaria	E125.4
14	Entrada125_5	Variable binaria	E125.5
15	Entrada125_6	Variable binaria	E125.6
16	Entrada125_7	Variable binaria	E125.7
17	Entrada126_0	Variable binaria	E126.0
18	Entrada126_1	Variable binaria	E126.1
19	Entrada126_2	Variable binaria	E126.2
20	Entrada126_3	Variable binaria	E126.3
21	Entrada126_4	Variable binaria	E126.4
22	Entrada126_5	Variable binaria	E126.5
23	Entrada126_6	Variable binaria	E126.6
24	Entrada126_7	Variable binaria	E126.7
25	Salida124_0	Variable binaria	A124.0
26	Salida124_1	Variable binaria	A124.1
27	Salida124_2	Variable binaria	A124.2
28	Salida124_3	Variable binaria	A124.3
29	Salida124_4	Variable binaria	A124.4
30	Salida124_5	Variable binaria	A124.5
31	Salida124_6	Variable binaria	A124.6
32	Salida124_7	Variable binaria	A124.7
33	Salida125_0	Variable binaria	A125.0
34	Salida125_1	Variable binaria	A125.1
35	Salida125_2	Variable binaria	A125.2
36	Salida125_3	Variable binaria	A125.3
37	Salida125_4	Variable binaria	A125.4
38	Salida125_5	Variable binaria	A125.5
39	Salida125_6	Variable binaria	A125.6
40	Salida125_7	Variable binaria	A125.7
41			

- **SEÑALES_S7226_EST1→** Este grupo se ha creado para crear las variables asociadas a todas las entradas y salidas digitales del S7 226 #3 (EST1) (todas en conjunto). Todas las variables son de tipo variable binaria, poniendo los nombres y direccionamiento adecuado.

Variables [SEÑALES_S7226_EST1]

	Nombre	Tipo datos	Dirección
1	Entrada0_0	Variable binaria	M18.0
2	Entrada0_1	Variable binaria	M18.1
3	Entrada0_2	Variable binaria	M18.2
4	Entrada0_3	Variable binaria	M18.3
5	Entrada0_4	Variable binaria	M18.4
6	Entrada0_5	Variable binaria	M18.5
7	Entrada0_6	Variable binaria	M18.6
8	Entrada0_7	Variable binaria	M18.7
9	Entrada1_0	Variable binaria	M19.0
10	Entrada1_1	Variable binaria	M19.1
11	Entrada1_2	Variable binaria	M19.2
12	Entrada1_3	Variable binaria	M19.3
13	Entrada1_4	Variable binaria	M19.4
14	Entrada1_5	Variable binaria	M19.5
15	Entrada1_6	Variable binaria	M19.6
16	Entrada1_7	Variable binaria	M19.7
17	Salida0_0	Variable binaria	M16.0
18	Salida0_1	Variable binaria	M16.1
19	Salida0_2	Variable binaria	M16.2
20	Salida0_3	Variable binaria	M16.3
21	Salida0_4	Variable binaria	M16.4
22	Salida0_5	Variable binaria	M16.5
23	Salida0_6	Variable binaria	M16.6
24	Salida0_7	Variable binaria	M16.7
25	Salida1_0	Variable binaria	M17.0
26	Salida1_1	Variable binaria	M17.1
27	Salida1_2	Variable binaria	M17.2
28	Salida1_3	Variable binaria	M17.3
29	Salida1_4	Variable binaria	M17.4
30	Salida1_5	Variable binaria	M17.5
31	Salida1_6	Variable binaria	M17.6
32	Salida1_7	Variable binaria	M17.7
33			

- **SEÑALES_S7226_EST2→** Este grupo se ha creado para crear las variables asociadas a todas las entradas y salidas digitales del S7 226 #4 (EST2) (todas en conjunto). Todas las variables son de tipo variable binaria, poniendo los nombres y direccionamiento adecuado.

Variables [SEÑALES_S7226_EST2]

	Nombre	Tipo datos	Dirección
1	Entradab_0_0	Variable binaria	M14.0
2	Entradab_0_1	Variable binaria	M14.1
3	Entradab_0_2	Variable binaria	M14.2
4	Entradab_0_3	Variable binaria	M14.3
5	Entradab_0_4	Variable binaria	M14.4
6	Entradab_0_5	Variable binaria	M14.5
7	Entradab_0_6	Variable binaria	M14.6
8	Entradab_0_7	Variable binaria	M14.7
9	Entradab_1_0	Variable binaria	M15.0
10	Entradab_1_1	Variable binaria	M15.1
11	Entradab_1_2	Variable binaria	M15.2
12	Entradab_1_3	Variable binaria	M15.3
13	Entradab_1_4	Variable binaria	M15.4
14	Entradab_1_5	Variable binaria	M15.5
15	Entradab_1_6	Variable binaria	M15.6
16	Entradab_1_7	Variable binaria	M15.7
17	Salidab_0_0	Variable binaria	M24.0
18	Salidab_0_1	Variable binaria	M24.1
19	Salidab_0_2	Variable binaria	M24.2
20	Salidab_0_3	Variable binaria	M24.3
21	Salidab_0_4	Variable binaria	M24.4
22	Salidab_0_5	Variable binaria	M24.5
23	Salidab_0_6	Variable binaria	M24.6
24	Salidab_0_7	Variable binaria	M24.7
25	Salidab_1_0	Variable binaria	M25.0
26	Salidab_1_1	Variable binaria	M25.1
27	Salidab_1_2	Variable binaria	M25.2
28	Salidab_1_3	Variable binaria	M25.3
29	Salidab_1_4	Variable binaria	M25.4
30	Salidab_1_5	Variable binaria	M25.5
31	Salidab_1_6	Variable binaria	M25.6
32	Salidab_1_7	Variable binaria	M25.7
33			

- **TIEMPO→** En este grupo se crean las variables asociadas a todas las marcas de byte donde se almacenan el año, mes, día, hora, minuto y segundo de los diferentes registros horarios utilizados en el programa del PLC para tener un control horario del tiempo de ruta general y del tiempo que les cuesta a los trenes llegar a su destino. Como estos valores se almacenan en las marcas correspondientes, el Scada leerá estos datos para mostrarlos por pantalla (se leen con la SFC1 del PLC master PROFIBUS s7 314).

Variables [TIEMPO]

	Nombre	Tipo datos	Dirección
1	Año_Actual	Valor de 8 bits sin signo	MB44
2	Dia_Actual	Valor de 8 bits sin signo	MB46
3	Hora_Actual	Valor de 8 bits sin signo	MB47
4	Hora_Actual_2	Valor de 8 bits sin signo	MB41
5	Hora_Fin_Mercan	Valor de 8 bits sin signo	MB29
6	Hora_Fin_Pasaje	Valor de 8 bits sin signo	MB35
7	Hora_Inicio_Mercan	Valor de 8 bits sin signo	MB32
8	Hora_Inicio_Pasaje	Valor de 8 bits sin signo	MB38
9	Mes_Actual	Valor de 8 bits sin signo	MB45
10	Minuto_Actual	Valor de 8 bits sin signo	MB48
11	Minuto_Actual_2	Valor de 8 bits sin signo	MB42
12	Minuto_Fin_Mercan	Valor de 8 bits sin signo	MB30
13	Minuto_Fin_Pasaje	Valor de 8 bits sin signo	MB36
14	Minuto_Inicio_Mercan	Valor de 8 bits sin signo	MB33
15	Minuto_Inicio_Pasaje	Valor de 8 bits sin signo	MB39
16	Segundo_Actual	Valor de 8 bits sin signo	MB49
17	Segundo_Actual_2	Valor de 8 bits sin signo	MB43
18	Segundo_Fin_Mercan	Valor de 8 bits sin signo	MB31
19	Segundo_Fin_Pasaje	Valor de 8 bits sin signo	MB37
20	Segundo_Inicio_Mercan	Valor de 8 bits sin signo	MB34
21	Segundo_Inicio_Pasaje	Valor de 8 bits sin signo	MB40
22	Tempo_Estimado	Valor de 16 bits sin signo	MW21
23	Tempo_Retraso	Valor de 16 bits sin signo	MW27
24	Tiempo_Transcurrido	Valor de 16 bits sin signo	T3
25			

- **TAGS ProfibusTREN (sin grupo asignado)→** En este grupo se encuentran las variables que gestionan la comunicación con las señales de salida analógicas de los S7 226 para el control de velocidad de las dos locomotoras. También se encuentran unas variables especiales que se les ha llamado Marcador…, con los que mediante escalado lineal por software se crean los marcadores de velocidad de las locomotoras. También se encuentran en este grupo las variables que recogen el voltaje que les llega a las locomotoras (escalado lineal). Por último, una variable de tipo valor de 8 bits con signo (“SISTEMA_MODOS”), que recoge el modo de funcionamiento en que se encuentra el sistema y una variable “Trenes_Activos”, de tipo valor de 8 bits sin signo para almacenar los trenes que están en funcionamiento en cada momento.

	Variable de proceso	Nombre de variable
1	MARCADOR_VELO_MERCAN1	MARCADOR_VELO_MERCAN1
2	MARCADOR_VELO_MERCAN2	MARCADOR_VELO_MERCAN2
3	MARCADOR_VELO_PASAJE1	MARCADOR_VELO_PASAJE1
4	MARCADOR_VELO_PASAJE2	MARCADOR_VELO_PASAJE2
5	REG_TREN_MERCANCIAS	REG_TREN_MERCANCIAS
6	TREN_PASAJEROS	TREN_PASAJEROS
7	VOLTAJE_319_316	VOLTAJE_319_316
8	VOLTAJE_319_402	VOLTAJE_319_402
9		

Todas las variables son de tipo valor de 16 bits sin signo y valor de 8 bits sin signo, excepto las dos variables de voltajes, “VOLTAJE_319_316” y “VOLTAJE_319_402”.

Variables internas

Se han creado también variables internas para funciones simuladas que no necesitan de comunicación con los PLC. Estas variables son las siguientes:

	Nombre	Tipo datos
1	Condiciones_Iniciales_Automatico	Variable binaria
2	Condiciones_Iniciales_Manual	Variable binaria
3	Liberar_freno	Variable binaria
4	Liberar_frenoMercan	Variable binaria
5	TiempoDif	Valor de 8 bits sin signo
6	TiempoEstimado	Valor de 8 bits sin signo
7	TiempoReal	Valor de 8 bits sin signo
8		

Definición de una graduación lineal (Escala lineal)

La graduación lineal solamente se puede realizar para los tags de proceso. En este caso, se han escalado varias variables:

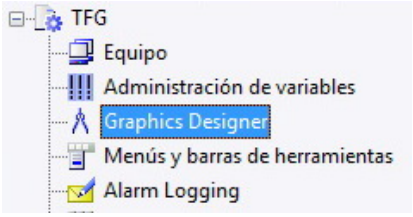
- "MARCADOR_VELO_MERCAN1", "MARCADOR_VELO_MERCAN2", "MARCADOR_VELO_PASAJE1", "MARCADOR_VELO_PASAJE2" → Estas variables representan los marcadores de velocidad de las dos locomotoras. Como los valores de programación de las señales de salidas analógicas no dan ninguna información en cuanto a velocidad en Km/h, que sería lo adecuado en una situación real de un maquinista controlando una locomotora, se ha utilizado el escalado lineal por software para convertir este valor.

- "VOLTAJE_319_316" y "VOLTAJE_319_402" → Estas dos variables representan en voltaje el valor escalado de las salidas analógicas de control de velocidad de los trenes. De esta forma, se puede establecer una escala lineal entre el valor de programación que toma la variable de proceso y el valor de voltaje escalado, ya que se puede averiguar utilizando un multímetro para medir qué incrementos o decrementos de voltaje se producen con respecto a la variable de programación. Después, atendiendo a los valores críticos calculados, se puede establecer una escala lineal de valores.

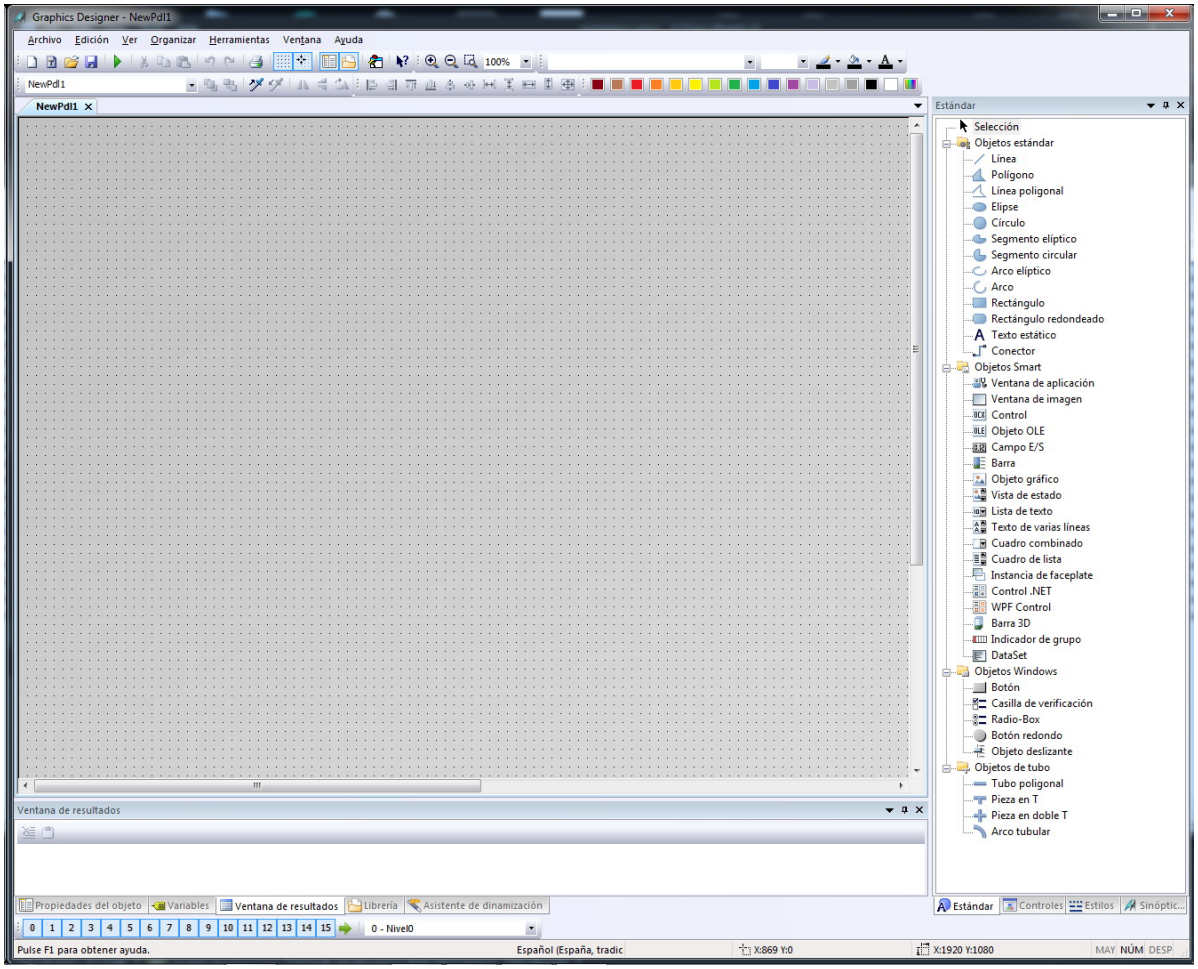
13.5. Graphics Designer

El editor Graphics Designer se encarga de la confección de las pantallas del WinCC. Básicamente es un entorno de dibujo con la característica de que los objetos poseen la capacidad de asociar sus propiedades a variables de comunicaciones que son proporcionadas por el Tag Management. Desde el propio Graphics Designer se pueden crear nuevas variables de comunicaciones, por lo que no será necesario salirse del mismo si la variable a usar aún no ha sido generada.

Para abrir el graphics designer hay que hacer doble clic mediante el botón derecho del ratón sobre "Graphics Designer", situado en la subventana izquierda del explorador de WinCC.



Una vez abierto el entorno aparecerá una ventana similar a la siguiente:



El entorno es personalizable mediante varias opciones, incluyendo las barras de menús, herramientas, etc, que pueden verse. Existen varias secciones importantes:

- Gama de colores: Asignar colores a los objetos seleccionados. Además de los 16 colores estándar, también se pueden utilizar los colores personalizados definidos por el usuario.
- Gama de objetos: Contiene los objetos estándar (polígono, elipse, rectángulo, etc), objetos inteligentes (control de OLE, elemento OLE, campos de E/S, etc), así como los objetos de ventana (botones, casillas de verificación, etc).
- Asistente dinámico: Ayuda a crear objetos dinámicos, por ejemplo, objetos que se muevan, arrancar otras aplicaciones o cambiar de idioma online.
- Funciones de alineamiento: Permite cambiar la posición absoluta de uno o varios objetos, cambiar la posición de los objetos seleccionados entre sí o estandarizar la altura y el ancho de varios objetos.
- Funciones de zoom: Define el factor de zoom (en porcentaje) para la ventana activa. Los factores de zoom estándar son: 8, 4, 1, ½ y ¼.
- Barra de menús: Contiene todos los comandos del menú para el diseñador gráfico. Los comandos no disponibles actualmente se visualizan en gris.
- Barra de herramientas estándar: Contiene los botones para realizar rápidamente los comandos más frecuentes.
- Barra de capas: Se utiliza para visualizar una de las 32 capas (0 a 31). Por defecto se selecciona la capa 0.

Para el desarrollo del proyecto se han realizado varias pantallas utilizando el graphics designer. Además, se han utilizado objetos gráficos de las librerías de WinCC. Externamente se han utilizado algunas imágenes de fondo para alguna de las pantallas realizadas.

14. Implementación de la página web

14.1. Implementación de la capa de integración (persistencia)

Especificación de las entidades JPA

Las entidades JPA que se emplean en la aplicación son:

EstacionJPA: Representa toda la información relativa a las estaciones.

```
/** @author Rafael Bello Ferrer, 2016*/
package jpa;
import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

/*** JPA Class EstacionJPA*/
@Entity
@Table(name="web_tfg.estacion")
public class EstacionJPA implements Serializable {
	private static final long serialVersionUID = 1L;

	@GeneratedValue
	private Integer idestacion;
	@Id
	private String nombre;
	private String poblacion;
	private String direccion;
	private String provincia;
	private String tlfnoinfo;
	private String datosinteres;

	/*** Class constructor methods*/
	public EstacionJPA() {
		super();
	}
	public EstacionJPA(Integer idestacion, String nombre, String poblacion, String
direccion, String provincia, String tlfnoinfo, String datosinteres){
		this.idestacion = idestacion;
		this.nombre = nombre;
		this.poblacion = poblacion;
		this.direccion = direccion;
		this.provincia = provincia;
		this.tlfnoinfo = tlfnoinfo;
		this.datosinteres= datosinteres;
	}

	/***  Methods get/set the fields of database*/

	@Id
	public String getNombre() {
		return nombre;
	}
	public  void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getPoblacion() {
```

```
		return poblacion;
	}
	public void setPoblacion(String poblacion) {
		this.poblacion = poblacion;
	}
	public String getDireccion() {
		return direccion;
	}
	public  void setDireccion(String direccion) {
		this.direccion = direccion;
	}
	public String getProvincia() {
		return provincia;
	}
	public  void setProvincia(String provincia) {
		this.provincia = provincia;
	}
	public String getTlfnoinfo() {
		return tlfnoinfo;
	}
	public  void setTlfnoinfo(String tlfnoinfo) {
		this.tlfnoinfo = tlfnoinfo;
	}
	public String getDatosinteres() {
		return datosinteres;
	}
	public  void setDatosinteres(String datosinteres) {
		this.datosinteres = datosinteres;
	}
}
```

FlotaJPA: Representa toda la información relativa a los trenes de la flota de la empresa.

```
/** @author Rafael Bello Ferrer, 2016*/
package jpa;
import java.io.Serializable;
import javax.persistence.*;

/*** JPA Class FlotaJPA*/
@Entity
@Table(name="web_tfg.flota")
public class FlotaJPA implements Serializable {
	private static final long serialVersionUID = 1L;

	@GeneratedValue
	private Integer idtren;

	private String fabricante;
	private Integer fechainicioservicio;
	private String nombrecomercial;
	@Id
	private String nombretecnico;
	private Integer nplazas;
	private String tipotren;
	private Integer trenescomprados;
	private Integer trenesenservicio;
	private Integer velocidadmax;

	/*** Class constructor methods */
	public FlotaJPA() {
		super();
	}
```

```
	public FlotaJPA(Integer idtren, String fabricante, Integer fechainicioservicio,
String nombrecomercial, String nombretecnico, Integer nplazas, String tipotren,
Integer trenescomprados,Integer trenesenservicio, Integer velocidadmax) {
		this.idtren=idtren;
		this.fabricante=fabricante;
		this.fechainicioservicio=fechainicioservicio;
		this.nombrecomercial=nombrecomercial;
		this.nombretecnico=nombretecnico;
		this.nplazas=nplazas;
		this.tipotren=tipotren;
		this.trenescomprados=trenescomprados;
		this.trenesenservicio=trenesenservicio;
		this.velocidadmax=velocidadmax;
	}

	/***  Methods get/set the fields of database*/

	public String getFabricante() {
		return fabricante;
	}
	public  void setFabricante(String fabricante) {
		this.fabricante = fabricante;
	}
	public Integer getFechainicioservicio() {
		return fechainicioservicio;
	}
	public void setFechainicioservicio(Integer fechainicioservicio) {
		this.fechainicioservicio = fechainicioservicio;
	}
	public String getNombrecomercial() {
		return nombrecomercial;
	}
	public  void setNombrecomercial(String nombrecomercial) {
		this.nombrecomercial = nombrecomercial;
	}
	@Id
	public String getNombretecnico() {
		return nombretecnico;
	}
	public void setNombretecnico(String nombretecnico) {
		this.nombretecnico = nombretecnico;
	}
	public Integer getNplazas() {
		return nplazas;
	}
	public  void setNplazas(Integer nplazas) {
		this.nplazas = nplazas;
	}
	public String getTipotren() {
		return tipotren;
	}
	public void setTipotren(String tipotren) {
		this.tipotren = tipotren;
	}
	public Integer getTrenescomprados() {
		return trenescomprados;
	}
	public void setTrenescomprados(Integer trenescomprados) {
		this.trenescomprados = trenescomprados;
	}
	public Integer getTrenesenservicio() {
		return trenesenservicio;
	}
	public void setTrenesenservicio(Integer trenesenservicio) {
```

```
		this.trenesenservicio = trenesenservicio;
	}
	public Integer getvelocidadmax() {
		return velocidadmax;
	}
	public void setVelocidadmax(Integer velocidadmax) {
		this.velocidadmax = velocidadmax;
	}
}
```

RutaJPA: Representa toda la información relativa a las rutas establecidas.

```
/** @author Rafael Bello Ferrer, 2016 */
package jpa;
import java.io.Serializable;
import javax.persistence.*;

/*** JPA Class RutaJPA */
@Entity
@Table(name="web_tfg.ruta")
public class RutaJPA implements Serializable {
	private static final long serialVersionUID = 1L;

	@GeneratedValue
	private Integer idruta;
	@Id
	private String nombreruta;
	private String tipotren;
	private String nombretrencomercial;
	private String nombretecnico;
	private String estacionsalida;
	private String estacionllegada;
	private String notas;

	/*** Class constructor methods */
	public RutaJPA() {
		super();
	}
	public RutaJPA(Integer idruta, String nombreruta, String tipotren, String
nombretrencomercial, String nombretecnico, String estacionsalida, String
estacionllegada, String notas)
	{
	this.idruta=idruta;
	this.nombreruta=nombreruta;
	this.tipotren=tipotren;
	this.nombretrencomercial=nombretrencomercial;
	this.nombretecnico=nombretecnico;
	this.estacionsalida=estacionsalida;
	this.estacionllegada=estacionllegada;
	this.notas=notas;
	}

	/***  Methods get/set the fields of database */
	@Id
	public String getNombreruta() {
		return nombreruta;
	}
	public void setNombreruta(String nombreruta) {
		this.nombreruta = nombreruta;
	}
	public String getTipotren() {
		return tipotren;
	}
```

```
	public void setTipotren(String tipotren) {
		this.tipotren = tipotren;
	}
	public String getNombretrencomercial() {
		return nombretrencomercial;
	}
	public  void setNombretrencomercial(String nombretrencomercial) {
		this.nombretrencomercial = nombretrencomercial;
	}
	public String getNombretecnico() {
		return nombretecnico;
	}
	public  void setNombretecnico(String nombretecnico) {
		this.nombretecnico = nombretecnico;
	}
	public String getEstacionsalida() {
		return estacionsalida;
	}
	public void setEstacionsalida(String estacionsalida) {
		this.estacionsalida = estacionsalida;
	}
	public String getEstacionllegada() {
		return estacionllegada;
	}
	public void setEstacionllegada(String estacionllegada) {
		this.estacionllegada = estacionllegada;
	}
	public String getNotas() {
		return notas;
	}
	public void setNotas(String notas) {
		this.notas = notas;
	}
}
```

ScadaJPA: Representa toda la información relativa a los datos Scada.

```
/** @author Rafael Bello Ferrer, 2016 */
package jpa;
import java.io.Serializable;
import java.sql.*;
import javax.persistence.*;

/*** JPA Class ScadaJPA */
@Entity
@Table(name="web_tfg.scada")
public class ScadaJPA implements Serializable {
	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue(strategy=GenerationType.AUTO)
	private Integer idscada;
	private Date fechayHora;
	private Boolean sensor0est1;
	private Boolean sensor1est1;
	private Boolean sensor2est1;
	private Boolean sensor3est1;
	private Boolean sensor4est1;
	private Boolean sensor5est1;
	private Boolean sensor6est1;
	private Boolean sensor7est1;
	private Boolean sensor8est1;
	private Boolean sensor9est1;
	private Boolean sensor0est2;
	private Boolean sensor1est2;
	private Boolean sensor2est2;
	private Boolean sensor3est2;
```

```
        private Boolean sensor4est2;
        private Boolean sensor5est2;
        private Boolean sensor6est2;
        private Boolean sensor7est2;
        private Boolean sensor8est2;
        private Boolean sensor9est2;
        private Integer velPasajeros;
        private Integer velMercancias;
        private Integer marcadorVeloPasaje1;
        private Integer marcadorVeloPasaje2;
        private Integer marcadorVeloMercan1;
        private Integer marcadorVeloMercan2;
        private Boolean semaf1Est1Rojo;
        private Boolean semaf1Est1Verde;
        private Boolean semaf2Est1Rojo;
        private Boolean semaf2Est1Verde;
        private Boolean semaf3Est1Rojo;
        private Boolean semaf3Est1Verde;
        private Boolean semaf4Est1Rojo;
        private Boolean semaf4Est1Verde;
        private Boolean semaf5Est1Rojo;
        private Boolean semaf5Est1Verde;
        private Boolean semaf6Est1Rojo;
        private Boolean semaf6Est1Verde;
        private Boolean semaf7Est1Rojo;
        private Boolean semaf7Est1Verde;
        private Boolean semaf8Est1Rojo;
        private Boolean semaf8Est1Verde;
        private Integer allSemafEst1;
        private Integer byte1SemafEst1;
        private Integer byte2SemafEst1;
        private Boolean semaf1Est2Rojo;
        private Boolean semaf1Est2Verde;
        private Boolean semaf2Est2Rojo;
        private Boolean semaf2Est2Verde;
        private Boolean semaf3Est2Rojo;
        private Boolean semaf3Est2Verde;
        private Boolean semaf4Est2Rojo;
        private Boolean semaf4Est2Verde;
        private Boolean semaf5Est2Rojo;
        private Boolean semaf5Est2Verde;
        private Boolean semaf6Est2Rojo;
        private Boolean semaf6Est2Verde;
        private Boolean semaf7Est2Rojo;
        private Boolean semaf7Est2Verde;
        private Boolean semaf8Est2Rojo;
        private Boolean semaf8Est2Verde;
        private Boolean semaf9Est2Rojo;
        private Boolean semaf9Est2Verde;
        private Boolean semaf10Est2Rojo;
        private Boolean semaf10Est2Verde;
        private Integer allSemafEst2;
        private Integer byte1SemafEst2;
        private Integer byte2SemafEst2;
        private Boolean desvio1;
        private Boolean desvio2;
        private Boolean desvio3;
        private Boolean desvio4;
        private Boolean desvio5;
        private Boolean desvio6;
        private Boolean sistemaAutomatico;
        private Boolean sistManualLibre;
        private Boolean sistManual;
        private Boolean sistemaOn;
        private Boolean emergencia;
        private Boolean trenPasajeActivo;
        private Boolean trenMercanActivo;
        private Boolean condInicialesAuto;
        private Boolean liberarFrenoPasaje;
        private Boolean liberarFrenoMercan;
        private Integer rutaSeleccionada;
        private Boolean permisoRuta1;
        private Boolean ruta1seleccionada;
```

```
        private Integer permisoTodasRutas;
        private Boolean ruta2seleccionada;
        private Boolean permisoRuta2;
        private Boolean bucleRutasSeleccionado;
        private Boolean permisoBucle1;
        private Integer permisoTodosBucles;
        private Boolean bucleRuta1seleccionado;
        private Double voltaje319316;
        private Double voltaje319402;

        /*** Class constructor methods */
        public ScadaJPA() {
                super();
        }
        public ScadaJPA(Integer idscada, Date fechayHora, Boolean sensor0est1, Boolean
sensor1est1, Boolean sensor2est1,Boolean sensor3est1, Boolean sensor4est1, Boolean sensor5est1,
Boolean sensor6est1, Boolean sensor7est1, Boolean sensor8est1, Boolean sensor9est1, Boolean
sensor0est2, Boolean sensor1est2, Boolean sensor2est2, Boolean sensor3est2, Boolean sensor4est2,
Boolean sensor5est2, Boolean sensor6est2, Boolean sensor7est2, Boolean sensor8est2, Boolean
sensor9est2, Integer velPasajeros, Integer velMercancias, Integer marcadorVeloPasaje1, Integer
marcadorVeloPasaje2, Integer marcadorVeloMercan1, Integer marcadorVeloMercan2 , Boolean
semaf1Est1Rojo, Boolean semaf1Est1Verde, Boolean semaf2Est1Rojo, Boolean semaf2Est1Verde, Boolean
semaf3Est1Rojo, Boolean semaf3Est1Verde, Boolean semaf4Est1Rojo, Boolean semaf4Est1Verde, Boolean
semaf5Est1Rojo, Boolean semaf5Est1Verde, Boolean semaf6Est1Rojo, Boolean semaf6Est1Verde, Boolean
semaf7Est1Rojo, Boolean semaf7Est1Verde, Boolean semaf8Est1Rojo, Boolean semaf8Est1Verde, Integer
allSemafEst1, Integer byte1SemafEst1, Integer byte2SemafEst1, Boolean semaf1Est2Rojo,
Boolean semaf1Est2Verde, Boolean semaf2Est2Rojo,Boolean semaf2Est2Verde, Boolean semaf3Est2Rojo,
Boolean semaf3Est2Verde, Boolean semaf4Est2Rojo, Boolean semaf4Est2Verde, Boolean semaf5Est2Rojo,
Boolean semaf5Est2Verde, Boolean semaf6Est2Rojo, Boolean semaf6Est2Verde, Boolean semaf7Est2Rojo,
Boolean semaf7Est2Verde, Boolean semaf8Est2Rojo, Boolean semaf8Est2Verde,Boolean semaf9Est2Rojo,
Boolean semaf9Est2Verde, Boolean semaf10Est2Rojo, Boolean semaf10Est2Verde, Integer allSemafEst2,
Integer byte1SemafEst2, Integer byte2SemafEst2, Boolean desvio1, Boolean desvio2, Boolean
desvio3, Boolean desvio4, Boolean desvio5, Boolean desvio6, Boolean sistemaAutomatico, Boolean
sistManualLibre, Boolean sistManual, Boolean sistemaOn, Boolean emergencia, Boolean
trenPasajeActivo, Boolean trenMercanActivo, Boolean condInicialesAuto, Boolean
liberarFrenoPasaje, Boolean liberarFrenoMercan, Integer rutaSeleccionada, Boolean
permisoRuta1,Boolean ruta1seleccionada, Integer     permisoTodasRutas, Boolean ruta2seleccionada,
Boolean permisoRuta2, Boolean bucleRutasSeleccionado, Boolean permisoBucle1, Integer
permisoTodosBucles, Boolean bucleRuta1seleccionado, Double voltaje319316, Double voltaje319402) {

                this.idscada=idscada;
                this.fechayHora=fechayHora;
                this.sensor0est1=sensor0est1;
                this.sensor1est1=sensor1est1;
                this.sensor2est1=sensor2est1;
                this.sensor3est1=sensor3est1;
                this.sensor4est1=sensor4est1;
                this.sensor5est1=sensor5est1;
                this.sensor6est1=sensor6est1;
                this.sensor7est1=sensor7est1;
                this.sensor8est1=sensor8est1;
                this.sensor9est1=sensor9est1;
                this.sensor0est2=sensor0est2;
                this.sensor1est2=sensor1est2;
                this.sensor2est2=sensor2est2;
                this.sensor3est2=sensor3est2;
                this.sensor4est2=sensor4est2;
                this.sensor5est2=sensor5est2;
                this.sensor6est2=sensor6est2;
                this.sensor7est2=sensor7est2;
                this.sensor8est2=sensor8est2;
                this.sensor9est2=sensor9est2;
                this.velPasajeros=velPasajeros;
                this.velMercancias=velMercancias;
                this.semaf1Est1Rojo=semaf1Est1Rojo;
                this.semaf1Est1Verde=semaf1Est1Verde;
                this.semaf2Est1Rojo=semaf2Est1Rojo;
                this.semaf2Est1Verde=semaf2Est1Verde;
                this.semaf3Est1Rojo=semaf3Est1Rojo;
                this.semaf3Est1Verde=semaf3Est1Verde;
                this.semaf4Est1Rojo=semaf4Est1Rojo;
                this.semaf4Est1Verde=semaf4Est1Verde;
                this.semaf5Est1Rojo=semaf5Est1Rojo;
```

```
		this.semaf5Est1Verde=semaf5Est1Verde;
		this.semaf6Est1Rojo=semaf6Est1Rojo;
		this.semaf6Est1Verde=semaf6Est1Verde;
		this.semaf7Est1Rojo=semaf7Est1Rojo;
		this.semaf7Est1Verde=semaf7Est1Verde;
		this.semaf8Est1Rojo=semaf8Est1Rojo;
		this.semaf8Est1Verde=semaf8Est1Verde;
		this.allSemafEst1=allSemafEst1;
		this.byte1SemafEst1=byte1SemafEst1;
		this.byte2SemafEst1=byte2SemafEst1;
		this.semaf1Est2Rojo=semaf1Est2Rojo;
		this.semaf1Est2Verde=semaf1Est2Verde;
		this.semaf2Est2Rojo=semaf2Est2Rojo;
		this.semaf2Est2Verde=semaf2Est2Verde;
		this.semaf3Est2Rojo=semaf3Est2Rojo;
		this.semaf3Est2Verde=semaf3Est2Verde;
		this.semaf4Est2Rojo=semaf4Est2Rojo;
		this.semaf4Est2Verde=semaf4Est2Verde;
		this.semaf5Est2Rojo=semaf5Est2Rojo;
		this.semaf5Est2Verde=semaf5Est2Verde;
		this.semaf6Est2Rojo=semaf6Est2Rojo;
		this.semaf6Est2Verde=semaf6Est2Verde;
		this.semaf7Est2Rojo=semaf7Est2Rojo;
		this.semaf7Est2Verde=semaf7Est2Verde;
		this.semaf8Est2Rojo=semaf8Est2Rojo;
		this.semaf8Est2Verde=semaf8Est2Verde;
		this.semaf9Est2Rojo=semaf9Est2Rojo;
		this.semaf9Est2Verde=semaf9Est2Verde;
		this.semaf10Est2Rojo=semaf10Est2Rojo;
		this.semaf10Est2Verde=semaf10Est2Verde;
		this.allSemafEst2=allSemafEst2;
		this.byte1SemafEst2=byte1SemafEst2;
		this.byte2SemafEst2=byte2SemafEst2;
		this.desvio1=desvio1;
		this.desvio2=desvio2;
		this.desvio3=desvio3;
		this.desvio4=desvio4;
		this.desvio5=desvio5;
		this.desvio6=desvio6;
		this.sistemaAutomatico=sistemaAutomatico;
		this.sistManualLibre=sistManualLibre;
		this.sistManual=sistManual;
		this.sistemaOn=sistemaOn;
		this.emergencia=emergencia;
		this.trenPasajeActivo=trenPasajeActivo;
		this.trenMercanActivo=trenMercanActivo;
		this.condInicialesAuto=condInicialesAuto;
		this.liberarFrenoPasaje=liberarFrenoPasaje;
		this.liberarFrenoMercan=liberarFrenoMercan;
		this.rutaSeleccionada=rutaSeleccionada;
		this.permisoRuta1=permisoRuta1;
		this.ruta1seleccionada=ruta1seleccionada;
		this.permisoTodasRutas=permisoTodasRutas;
		this.ruta2seleccionada=ruta2seleccionada;
		this.permisoRuta2=permisoRuta2;
		this.bucleRutasSeleccionado=bucleRutasSeleccionado;
		this.permisoBucle1=permisoBucle1;
		this.permisoTodosBucles=permisoTodosBucles;
		this.bucleRuta1seleccionado=bucleRuta1seleccionado;
		this.voltaje319316=voltaje319316;
		this.voltaje319402=voltaje319402;
	}

	/***  Methods get/set the fields of database */

	public Date getFechayHora() {
		return fechayHora;
	}
	public void setFechayHora(Date fechayHora) {
		this.fechayHora = fechayHora;
	}
	public Boolean getSensor0est1() {
		return sensor0est1;
```

```
	}
	public void setSensor0est1(Boolean sensor0est1) {
		this.sensor0est1 = sensor0est1;
	}
	public Boolean getSensor1est1() {
		return sensor1est1;
	}
	public void setSensor1est1(Boolean sensor1est1) {
		this.sensor1est1 = sensor1est1;
	}
	public Boolean getSensor2est1() {
		return sensor2est1;
	}
	public void setSensor2est1(Boolean sensor2est1) {
		this.sensor2est1 = sensor2est1;
	}
	public Boolean getSensor3est1() {
		return sensor3est1;
	}
	public void setSensor3est1(Boolean sensor3est1) {
		this.sensor3est1 = sensor3est1;
	}
	public Boolean getSensor4est1() {
		return sensor4est1;
	}
	public void setSensor4est1(Boolean sensor4est1) {
		this.sensor4est1 = sensor4est1;
	}
	public Boolean getSensor5est1() {
		return sensor5est1;
	}
	public void setSensor5est1(Boolean sensor5est1) {
		this.sensor5est1 = sensor5est1;
	}
	public Boolean getSensor6est1() {
		return sensor6est1;
	}
	public void setSensor6est1(Boolean sensor6est1) {
		this.sensor6est1 = sensor6est1;
	}
	public Boolean getSensor7est1() {
		return sensor7est1;
	}
	public void setSensor7est1(Boolean sensor7est1) {
		this.sensor7est1 = sensor7est1;
	}
	public Boolean getSensor8est1() {
		return sensor8est1;
	}
	public void setSensor8est1(Boolean sensor8est1) {
		this.sensor8est1 = sensor8est1;
	}
	public Boolean getSensor9est1() {
		return sensor9est1;
	}
	public void setSensor9est1(Boolean sensor9est1) {
		this.sensor9est1 = sensor9est1;
	}
	public Boolean getSensor0est2() {
		return sensor0est2;
	}
	public void setSensor0est2(Boolean sensor0est2) {
		this.sensor0est2 = sensor0est2;
	}
	public Boolean getSensor1est2() {
		return sensor1est2;
	}
	public void setSensor1est2(Boolean sensor1est2) {
		this.sensor1est2 = sensor1est2;
	}
	public Boolean getSensor2est2() {
		return sensor2est2;
	}
}
```

```
	public void setSensor2est2(Boolean sensor2est2) {
		this.sensor2est2 = sensor2est2;
	}
	public Boolean getSensor3est2() {
		return sensor3est2;
	}
	public void setSensor3est2(Boolean sensor3est2) {
		this.sensor3est2 = sensor3est2;
	}
	public Boolean getSensor4est2() {
		return sensor4est2;
	}
	public void setSensor4est2(Boolean sensor4est2) {
		this.sensor4est2 = sensor4est2;
	}
	public Boolean getSensor5est2() {
		return sensor5est2;
	}
	public void setSensor5est2(Boolean sensor5est2) {
		this.sensor5est2 = sensor5est2;
	}
	public Boolean getSensor6est2() {
		return sensor6est2;
	}
	public void setSensor6est2(Boolean sensor6est2) {
		this.sensor6est2 = sensor6est2;
	}
	public Boolean getSensor7est2() {
		return sensor7est2;
	}
	public void setSensor7est2(Boolean sensor7est2) {
		this.sensor7est2 = sensor7est2;
	}
	public Boolean getSensor8est2() {
		return sensor8est2;
	}
	public void setSensor8est2(Boolean sensor8est2) {
		this.sensor8est2 = sensor8est2;
	}
	public Boolean getSensor9est2() {
		return sensor9est2;
	}
	public void setSensor9est2(Boolean sensor9est2) {
		this.sensor9est2 = sensor9est2;
	}
	public Integer getVelPasajeros() {
		return velPasajeros;
	}
	public void setVelPasajeros(Integer velPasajeros) {
		this.velPasajeros = velPasajeros;
	}
	public Integer getVelMercancias() {
		return velMercancias;
	}
	public void setVelMercancias(Integer velMercancias) {
		this.velMercancias = velMercancias;
	}
	public Integer getMarcadorVeloPasaje1() {
		return marcadorVeloPasaje1;
	}
	public void setMarcadorVeloPasaje1(Integer marcadorVeloPasaje1) {
		this.marcadorVeloPasaje1 = marcadorVeloPasaje1;
	}
	public Integer getMarcadorVeloPasaje2() {
		return marcadorVeloPasaje2;
	}
	public void setMarcadorVeloPasaje2(Integer marcadorVeloPasaje2) {
		this.marcadorVeloPasaje2 = marcadorVeloPasaje2;
	}
	public Integer getMarcadorVeloMercan1() {
		return marcadorVeloMercan1;
	}
	public void setMarcadorVeloMercan1(Integer marcadorVeloMercan1) {
```

```
		this.marcadorVeloMercan1 = marcadorVeloMercan1;
	}
	public Integer getMarcadorVeloMercan2() {
		return marcadorVeloMercan2;
	}
	public void setMarcadorVeloMercan2(Integer marcadorVeloMercan2) {
		this.marcadorVeloMercan2 = marcadorVeloMercan2;
	}
	public Boolean getSemaf1Est1Rojo() {
		return semaf1Est1Rojo;
	}
	public void setSemaf1Est1Rojo(Boolean semaf1Est1Rojo) {
		this.semaf1Est1Rojo = semaf1Est1Rojo;
	}
	public Boolean getSemaf1Est1Verde() {
		return semaf1Est1Verde;
	}
	public void setSemaf1Est1Verde(Boolean semaf1Est1Verde) {
		this.semaf1Est1Verde = semaf1Est1Verde;
	}
	public Boolean getSemaf2Est1Rojo() {
		return semaf2Est1Rojo;
	}
	public void setSemaf2Est1Rojo(Boolean semaf2Est1Rojo) {
		this.semaf2Est1Rojo = semaf2Est1Rojo;
	}
	public Boolean getSemaf2Est1Verde() {
		return semaf2Est1Verde;
	}
	public void setSemaf2Est1Verde(Boolean semaf2Est1Verde) {
		this.semaf2Est1Verde = semaf2Est1Verde;
	}
	public Boolean getSemaf3Est1Rojo() {
		return semaf3Est1Rojo;
	}
	public void setSemaf3Est1Rojo(Boolean semaf3Est1Rojo) {
		this.semaf3Est1Rojo = semaf3Est1Rojo;
	}
	public Boolean getSemaf3Est1Verde() {
		return semaf3Est1Verde;
	}
	public void setSemaf3Est1Verde(Boolean semaf3Est1Verde) {
		this.semaf3Est1Verde = semaf3Est1Verde;
	}
	public Boolean getSemaf4Est1Rojo() {
		return semaf4Est1Rojo;
	}
	public void setSemaf4Est1Rojo(Boolean semaf4Est1Rojo) {
		this.semaf4Est1Rojo = semaf4Est1Rojo;
	}
	public Boolean getSemaf4Est1Verde() {
		return semaf4Est1Verde;
	}
	public void setSemaf4Est1Verde(Boolean semaf4Est1Verde) {
		this.semaf4Est1Verde = semaf4Est1Verde;
	}
	public Boolean getSemaf5Est1Rojo() {
		return semaf5Est1Rojo;
	}
	public void setSemaf5Est1Rojo(Boolean semaf5Est1Rojo) {
		this.semaf5Est1Rojo = semaf5Est1Rojo;
	}
	public Boolean getSemaf5Est1Verde() {
		return semaf5Est1Verde;
	}
	public void setSemaf5Est1Verde(Boolean semaf5Est1Verde) {
		this.semaf5Est1Verde = semaf5Est1Verde;
	}
	public Boolean getSemaf6Est1Rojo() {
		return semaf6Est1Rojo;
	}
	public void setSemaf6Est1Rojo(Boolean semaf6Est1Rojo) {
		this.semaf6Est1Rojo = semaf6Est1Rojo;
```

```
	}
	public Boolean getSemaf6Est1Verde() {
		return semaf6Est1Verde;
	}
	public void setSemaf6Est1Verde(Boolean semaf6Est1Verde) {
		this.semaf6Est1Verde = semaf6Est1Verde;
	}
	public Boolean getSemaf7Est1Rojo() {
		return semaf7Est1Rojo;
	}
	public void setSemaf7Est1Rojo(Boolean semaf7Est1Rojo) {
		this.semaf7Est1Rojo = semaf7Est1Rojo;
	}
	public Boolean getSemaf7Est1Verde() {
		return semaf7Est1Verde;
	}
	public void setSemaf7Est1Verde(Boolean semaf7Est1Verde) {
		this.semaf7Est1Verde = semaf7Est1Verde;
	}
	public Boolean getSemaf8Est1Rojo() {
		return semaf8Est1Rojo;
	}
	public void setSemaf8Est1Rojo(Boolean semaf8Est1Rojo) {
		this.semaf8Est1Rojo = semaf8Est1Rojo;
	}
	public Boolean getSemaf8Est1Verde() {
		return semaf8Est1Verde;
	}
	public void setSemaf8Est1Verde(Boolean semaf8Est1Verde) {
		this.semaf8Est1Verde = semaf8Est1Verde;
	}
	public Integer getAllSemafEst1() {
		return allSemafEst1;
	}
	public void setAllSemafEst1(Integer allSemafEst1) {
		this.allSemafEst1 = allSemafEst1;
	}
	public Integer getByte1SemafEst1() {
		return byte1SemafEst1;
	}
	public void setByte1SemafEst1(Integer byte1SemafEst1) {
		this.byte1SemafEst1 = byte1SemafEst1;
	}
	public Integer getByte2SemafEst1() {
		return byte2SemafEst1;
	}
	public void setByte2SemafEst1(Integer byte2SemafEst1) {
		this.byte2SemafEst1 = byte2SemafEst1;
	}
	public Boolean getSemaf1Est2Rojo() {
		return semaf1Est2Rojo;
	}
	public void setSemaf1Est2Rojo(Boolean semaf1Est2Rojo) {
		this.semaf1Est2Rojo = semaf1Est2Rojo;
	}
	public Boolean getSemaf1Est2Verde() {
		return semaf1Est2Verde;
	}
	public void setSemaf1Est2Verde(Boolean semaf1Est2Verde) {
		this.semaf1Est2Verde = semaf1Est2Verde;
	}
	public Boolean getSemaf2Est2Rojo() {
		return semaf2Est2Rojo;
	}
	public void setSemaf2Est2Rojo(Boolean semaf2Est2Rojo) {
		this.semaf2Est2Rojo = semaf2Est2Rojo;
	}
	public Boolean getSemaf2Est2Verde() {
		return semaf2Est2Verde;
	}
	public void setSemaf2Est2Verde(Boolean semaf2Est2Verde) {
		this.semaf2Est2Verde = semaf2Est2Verde;
	}
```

```
	public Boolean getSemaf3Est2Rojo() {
		return semaf3Est2Rojo;
	}
	public void setSemaf3Est2Rojo(Boolean semaf3Est2Rojo) {
		this.semaf3Est2Rojo = semaf3Est2Rojo;
	}
	public Boolean getSemaf3Est2Verde() {
		return semaf3Est2Verde;
	}
	public void setSemaf3Est2Verde(Boolean semaf3Est2Verde) {
		this.semaf3Est2Verde = semaf3Est2Verde;
	}
	public Boolean getSemaf4Est2Rojo() {
		return semaf4Est2Rojo;
	}
	public void setSemaf4Est2Rojo(Boolean semaf4Est2Rojo) {
		this.semaf4Est2Rojo = semaf4Est2Rojo;
	}
	public Boolean getSemaf4Est2Verde() {
		return semaf4Est2Verde;
	}
	public void setSemaf4Est2Verde(Boolean semaf4Est2Verde) {
		this.semaf4Est2Verde = semaf4Est2Verde;
	}
	public Boolean getSemaf5Est2Rojo() {
		return semaf5Est2Rojo;
	}
	public void setSemaf5Est2Rojo(Boolean semaf5Est2Rojo) {
		this.semaf5Est2Rojo = semaf5Est2Rojo;
	}
	public Boolean getSemaf5Est2Verde() {
		return semaf5Est2Verde;
	}
	public void setSemaf5Est2Verde(Boolean semaf5Est2Verde) {
		this.semaf5Est2Verde = semaf5Est2Verde;
	}
	public Boolean getSemaf6Est2Rojo() {
		return semaf6Est2Rojo;
	}
	public void setSemaf6Est2Rojo(Boolean semaf6Est2Rojo) {
		this.semaf6Est2Rojo = semaf6Est2Rojo;
	}
	public Boolean getSemaf6Est2Verde() {
		return semaf6Est2Verde;
	}
	public void setSemaf6Est2Verde(Boolean semaf6Est2Verde) {
		this.semaf6Est2Verde = semaf6Est2Verde;
	}
	public Boolean getSemaf7Est2Rojo() {
		return semaf7Est2Rojo;
	}
	public void setSemaf7Est2Rojo(Boolean semaf7Est2Rojo) {
		this.semaf7Est2Rojo = semaf7Est2Rojo;
	}
	public Boolean getSemaf7Est2Verde() {
		return semaf7Est2Verde;
	}
	public void setSemaf7Est2Verde(Boolean semaf7Est2Verde) {
		this.semaf7Est2Verde = semaf7Est2Verde;
	}
	public Boolean getSemaf8Est2Rojo() {
		return semaf8Est2Rojo;
	}
	public void setSemaf8Est2Rojo(Boolean semaf8Est2Rojo) {
		this.semaf8Est2Rojo = semaf8Est2Rojo;
	}
	public Boolean getSemaf8Est2Verde() {
		return semaf8Est2Verde;
	}
	public void setSemaf8Est2Verde(Boolean semaf8Est2Verde) {
		this.semaf8Est2Verde = semaf8Est2Verde;
	}
	public Boolean getSemaf9Est2Rojo() {
```

```
		return semaf9Est2Rojo;
	}
	public void setSemaf9Est2Rojo(Boolean semaf9Est2Rojo) {
		this.semaf9Est2Rojo = semaf9Est2Rojo;
	}
	public Boolean getSemaf9Est2Verde() {
		return semaf9Est2Verde;
	}
	public void setSemaf9Est2Verde(Boolean semaf9Est2Verde) {
		this.semaf9Est2Verde = semaf9Est2Verde;
	}
	public Boolean getSemaf10Est2Rojo() {
		return semaf10Est2Rojo;
	}
	public void setSemaf10Est2Rojo(Boolean semaf10Est2Rojo) {
		this.semaf10Est2Rojo = semaf10Est2Rojo;
	}
	public Boolean getSemaf10Est2Verde() {
		return semaf10Est2Verde;
	}
	public void setSemaf10Est2Verde(Boolean semaf10Est2Verde) {
		this.semaf10Est2Verde = semaf10Est2Verde;
	}
	public Integer getAllSemafEst2() {
		return allSemafEst2;
	}
	public void setAllSemafEst2(Integer allSemafEst2) {
		this.allSemafEst2 = allSemafEst2;
	}
	public Integer getByte1SemafEst2() {
		return byte1SemafEst2;
	}
	public void setByte1SemafEst2(Integer byte1SemafEst2) {
		this.byte1SemafEst2 = byte1SemafEst2;
	}
	public Integer getByte2SemafEst2() {
		return byte2SemafEst2;
	}
	public void setByte2SemafEst2(Integer byte2SemafEst2) {
		this.byte2SemafEst2 = byte2SemafEst2;
	}
	public Boolean getDesvio1() {
		return desvio1;
	}
	public void setDesvio1(Boolean desvio1) {
		this.desvio1 = desvio1;
	}
	public Boolean getDesvio2() {
		return desvio2;
	}
	public void setDesvio2(Boolean desvio2) {
		this.desvio2 = desvio2;
	}
	public Boolean getDesvio3() {
		return desvio3;
	}
	public void setDesvio3(Boolean desvio3) {
		this.desvio3 = desvio3;
	}
	public Boolean getDesvio4() {
		return desvio4;
	}
	public void setDesvio4(Boolean desvio4) {
		this.desvio4 = desvio4;
	}
	public Boolean getDesvio5() {
		return desvio5;
	}
	public void setDesvio5(Boolean desvio5) {
		this.desvio5 = desvio5;
	}
	public Boolean getDesvio6() {
		return desvio6;
```

```
	}
	public void setDesvio6(Boolean desvio6) {
		this.desvio6 = desvio6;
	}
	public Boolean getSistemaAutomatico() {
		return sistemaAutomatico;
	}
	public void setSistemaAutomatico(Boolean sistemaAutomatico) {
		this.sistemaAutomatico = sistemaAutomatico;
	}
	public Boolean getSistManualLibre() {
		return sistManualLibre;
	}
	public void setSistManualLibre(Boolean sistManualLibre) {
		this.sistManualLibre = sistManualLibre;
	}
	public Boolean getSistManual() {
		return sistManual;
	}
	public void setSistManual(Boolean sistManual) {
		this.sistManual = sistManual;
	}
	public Boolean getSistemaOn() {
		return sistemaOn;
	}
	public void setSistemaOn(Boolean sistemaOn) {
		this.sistemaOn = sistemaOn;
	}
	public Boolean getEmergencia() {
		return emergencia;
	}
	public void setEmergencia(Boolean emergencia) {
		this.emergencia = emergencia;
	}
	public Boolean getTrenPasajeActivo() {
		return trenPasajeActivo;
	}
	public void setTrenPasajeActivo(Boolean trenPasajeActivo) {
		this.trenPasajeActivo = trenPasajeActivo;
	}
	public Boolean getTrenMercanActivo() {
		return trenMercanActivo;
	}
	public void setTrenMercanActivo(Boolean trenMercanActivo) {
		this.trenMercanActivo = trenMercanActivo;
	}
	public Boolean getCondInicialesAuto() {
		return condInicialesAuto;
	}
	public void setCondInicialesAuto(Boolean condInicialesAuto) {
		this.condInicialesAuto = condInicialesAuto;
	}
	public Boolean getLiberarFrenoPasaje() {
		return liberarFrenoPasaje;
	}
	public void setLiberarFrenoPasaje(Boolean liberarFrenoPasaje) {
		this.liberarFrenoPasaje = liberarFrenoPasaje;
	}
	public Boolean getLiberarFrenoMercan() {
		return liberarFrenoMercan;
	}
	public void setLiberarFrenoMercan(Boolean liberarFrenoMercan) {
		this.liberarFrenoMercan = liberarFrenoMercan;
	}
	public Integer getRutaSeleccionada() {
		return rutaSeleccionada;
	}
	public void setRutaSeleccionada(Integer rutaSeleccionada) {
		this.rutaSeleccionada = rutaSeleccionada;
	}
	public Boolean getPermisoRuta1() {
		return permisoRuta1;
	}
}
```

```
	public void setPermisoRuta1(Boolean permisoRuta1) {
		this.permisoRuta1 = permisoRuta1;
	}
	public Boolean getRuta1seleccionada() {
		return ruta1seleccionada;
	}
	public void setRuta1seleccionada(Boolean ruta1seleccionada) {
		this.ruta1seleccionada = ruta1seleccionada;
	}
	public Integer getPermisoTodasRutas() {
		return permisoTodasRutas;
	}
	public void setPermisoTodasRutas(Integer permisoTodasRutas) {
		this.permisoTodasRutas = permisoTodasRutas;
	}
	public Boolean getRuta2seleccionada() {
		return ruta2seleccionada;
	}
	public void setRuta2seleccionada(Boolean ruta2seleccionada) {
		this.ruta2seleccionada = ruta2seleccionada;
	}
	public Boolean getPermisoRuta2() {
		return permisoRuta2;
	}
	public void setPermisoRuta2(Boolean permisoRuta2) {
		this.permisoRuta2 = permisoRuta2;
	}
	public Boolean getBucleRutasSeleccionado() {
		return bucleRutasSeleccionado;
	}
	public void setBucleRutasSeleccionado(Boolean bucleRutasSeleccionado) {
		this.bucleRutasSeleccionado = bucleRutasSeleccionado;
	}
	public Boolean getPermisoBucle1() {
		return permisoBucle1;
	}
	public void setPermisoBucle1(Boolean permisoBucle1) {
		this.permisoBucle1 = permisoBucle1;
	}
	public Integer getPermisoTodosBucles() {
		return permisoTodosBucles;
	}
	public void setPermisoTodosBucles(Integer permisoTodosBucles) {
		this.permisoTodosBucles = permisoTodosBucles;
	}
	public Boolean getBucleRuta1seleccionado() {
		return bucleRuta1seleccionado;
	}
	public void setBucleRuta1seleccionado(Boolean bucleRuta1seleccionado) {
		this.bucleRuta1seleccionado = bucleRuta1seleccionado;
	}
	public Double getVoltaje319316() {
		return voltaje319316;
	}
	public void setVoltaje319316(Double voltaje319316) {
		this.voltaje319316 = voltaje319316;
	}
	public Double getVoltaje319402() {
		return voltaje319402;
	}
	public void setVoltaje319402(Double voltaje319402) {
		this.voltaje319402 = voltaje319402;
	}
}
```

14.2. Implementación de la base de datos

Creación de tablas e inserción de registros en la base de datos

A continuación, se muestran los script utilizados para la creación de las tablas en la base de datos y la inserción de datos en las tablas.

- Creación de la tabla Estación

```
CREATE TABLE web_tfg.estacion
(idestacion serial NOT NULL,
nombre text NOT NULL,
poblacion text,
direccion text,
provincia text,
tlfnoinfo text,
datosinteres text,
CONSTRAINT estacion_pkey PRIMARY KEY (nombre))
WITH (OIDS=FALSE);
ALTER TABLE web_tfg.estacion owner TO "USER";
```

- Inserción de registros en la tabla Estación

```
insert into web_tfg.estacion(nombre, poblacion, direccion, provincia, tlfnoinfo, datosinteres)
values('Estacion_1','Población_1','Dirección1','Provincia_1','978123456','Estación rehabilitada');

insert into web_tfg.estacion(nombre, poblacion, direccion, provincia, tlfnoinfo, datosinteres)
values('Estacion_2','Población_2',' Dirección2','Provincia_2','974789456','Estación de nueva construcción');

insert into web_tfg.estacion(nombre, poblacion, direccion, provincia, tlfnoinfo, datosinteres)
values('Estacion_3','Población_3',' Dirección3','Provincia_3','Telf_3','Datos_3');

insert into web_tfg.estacion(nombre, poblacion, direccion, provincia, tlfnoinfo, datosinteres)
values('Estacion_4','Población_4',' Dirección4','Provincia_4','Telf_4','Datos_4');

insert into web_tfg.estacion(nombre, poblacion, direccion, provincia, tlfnoinfo, datosinteres)
values('Estacion_5','Población_5',' Dirección5','Provincia_5','Telf_1','Datos_5');

insert into web_tfg.estacion(nombre, poblacion, direccion, provincia, tlfnoinfo, datosinteres)
values('Estacion_6','Población_6',' Dirección6','Provincia_6','Telf_2','Datos_6');
```

- Creación de la tabla Flota

```
CREATE TABLE web_tfg.flota
(idtren serial NOT NULL,
fabricante text,
fechainicioservicio integer,
nombrecomercial text,
nombretecnico text NOT NULL,
nplazas integer,
tipotren text,
```

```
trenescomprados integer,
trenesenservicio integer,
velocidadmax integer,
CONSTRAINT "flota_pkey" PRIMARY KEY (nombretecnico))
WITH (OIDS=FALSE);
ALTER TABLE web_tfg.flota owner TO "USER";
```

- Inserción de registros en la tabla Flota

```
insert into web_tfg.flota(fabricante, fechainicioservicio, nombrecomercial, nombretecnico, nplazas, tipotren, trenescomprados, trenesenservicio, velocidadmax) values('CAF',2009,'MEDIA DISTANCIA','DIESEL S-599',184,'Pasajeros',10,6,160);

insert into web_ tfg.flota(fabricante, fechainicioservicio, nombrecomercial, nombretecnico, nplazas, tipotren, trenescomprados, trenesenservicio, velocidadmax) values('SIEMENS',2009,'MEDIA DISTANCIA','ELECTRICO S-449',260,'Pasajeros',50,42,140);

insert into web_ tfg.flota(fabricante, fechainicioservicio, nombrecomercial, nombretecnico, nplazas, tipotren, trenescomprados, trenesenservicio, velocidadmax) values('BOMBARDIER',2008,'LARGA DISTANCIA','ELECTRICO S-253',0,'Mercancias',100,15,140);

insert into web_ tfg.flota(fabricante, fechainicioservicio, nombrecomercial, nombretecnico, nplazas, tipotren, trenescomprados, trenesenservicio, velocidadmax) values('ALSTON',1992,'AVE','S-100',329,'Pasajeros',24,24,300);

insert into web_ tfg.flota(fabricante, fechainicioservicio, nombrecomercial, nombretecnico, nplazas, tipotren, trenescomprados, trenesenservicio, velocidadmax) values('TALGO-BOMBARDIER',2005,'AVE','S-102/s-112',346,'Pasajeros',46,16,330);

insert into web_ tfg.flota(fabricante, fechainicioservicio, nombrecomercial, nombretecnico, nplazas, tipotren, trenescomprados, trenesenservicio, velocidadmax) values('SIEMENS',2007,'AVE','S-103',402,'Pasajeros',26,12,350);
```

- Creación de la tabla Ruta

```
CREATE TABLE web_tfg.ruta
(idruta serial NOT NULL,
nombreruta text NOT NULL,
tipotren text,
nombretrencomercial text,
nombretecnico text,
estacionsalida text,
estacionllegada text,
notas text,
CONSTRAINT "ruta_pkey" PRIMARY KEY ("nombreruta"),
CONSTRAINT tren_fkey FOREIGN KEY (nombretecnico)
   REFERENCES web_tfg.flota (nombretecnico) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
```

```
CONSTRAINT estsalida_fkey FOREIGN KEY (estacionsalida)
   REFERENCES web_tfg.estacion (nombre) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT estllegada_fkey FOREIGN KEY (estacionllegada)
   REFERENCES web_tfg.estacion (nombre) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION)
WITH (OIDS=FALSE);
ALTER TABLE web_tfg.ruta owner TO "USER";
```

- Inserción de registros en la tabla Ruta

```
insert into web_tfg.ruta(nombreruta, tipotren, nombretrencomercial, nombretecnico, estacionsalida,
estacionllegada,  notas) values('Ruta-1','Pasajeros','AVE','S-102/s-112','Estacion_5','Estacion_1','notas1');

insert into web_tfg.ruta(nombreruta, tipotren, nombretrencomercial, nombretecnico, estacionsalida,
estacionllegada,  notas) values('Ruta-2','Mercancias','LARGA DISTANCIA','ELECTRICO S-253',
'Estacion_2','Estacion_3','notas2');
```

- Creación de la tabla Scada

```
CREATE TABLE web_tfg.scada
(idscada serial NOT NULL, fechayHora date, sensor0est1 boolean, sensor1est1 boolean, sensor2est1 boolean,
sensor3est1 boolean, sensor4est1 boolean,sensor5est1 boolean, sensor6est1 boolean, sensor7est1 boolean,
sensor8est1 boolean, sensor9est1 boolean,sensor0est2 boolean, sensor1est2 boolean, sensor2est2 boolean,
sensor3est2 boolean, sensor4est2 boolean,sensor5est2 boolean, sensor6est2 boolean, sensor7est2 boolean,
sensor8est2 boolean, sensor9est2 boolean, velPasajeros integer, velMercancias integer, marcadorVeloPasaje1
integer, marcadorVeloPasaje2 integer, marcadorVeloMercan1 integer, marcadorVeloMercan2 integer,
semaf1Est1Rojo boolean, semaf1Est1Verde boolean, semaf2Est1Rojo boolean, semaf2Est1Verde boolean,
semaf3Est1Rojo boolean, semaf3Est1Verde boolean, semaf4Est1Rojo boolean, semaf4Est1Verde boolean,
semaf5Est1Rojo boolean, semaf5Est1Verde boolean, semaf6Est1Rojo boolean, semaf6Est1Verde boolean,
semaf7Est1Rojo boolean, semaf7Est1Verde boolean, semaf8Est1Rojo boolean, semaf8Est1Verde boolean,
allSemafEst1 integer, byte1SemafEst1 integer, byte2SemafEst1 integer, semaf1Est2Rojo boolean,
semaf1Est2Verde boolean, semaf2Est2Rojo boolean, semaf2Est2Verde boolean, semaf3Est2Rojo boolean,
semaf3Est2Verde boolean, semaf4Est2Rojo boolean, semaf4Est2Verde boolean, semaf5Est2Rojo boolean,
semaf5Est2Verde boolean, semaf6Est2Rojo boolean, semaf6Est2Verde boolean, semaf7Est2Rojo boolean,
semaf7Est2Verde boolean, semaf8Est2Rojo boolean, semaf8Est2Verde boolean, semaf9Est2Rojo boolean,
semaf9Est2Verde boolean, semaf10Est2Rojo boolean, semaf10Est2Verde boolean, allSemafEst2 integer,
byte1SemafEst2 integer, byte2SemafEst2 integer, desvio1 boolean, desvio2 boolean, desvio3 boolean, desvio4
boolean, desvio5 boolean, desvio6 boolean, sistemaAutomatico boolean, sistManualLibre boolean, sistManual
boolean, sistemaOn boolean, emergencia boolean, trenPasajeActivo boolean, trenMercanActivo boolean,
condInicialesAuto boolean, liberarFrenoPasaje boolean, liberarFrenoMercan boolean, rutaSeleccionada integer,
permisoRuta1 boolean, ruta1seleccionada boolean, permisoTodasRutas integer, ruta2seleccionada boolean,
permisoRuta2 boolean, bucleRutasSeleccionado boolean, permisoBucle1 boolean, permisoTodosBucles integer,
bucleRuta1seleccionado boolean, voltaje319316 real, voltaje319402 real,
CONSTRAINT "scada_pkey" PRIMARY KEY (idscada))
WITH (OIDS=FALSE);
ALTER TABLE web_tfg.scada owner TO "USER";
```

- Inserción de registros en la tabla Scada

```
insert into web_tfg.scada(fechayHora, sensor0est1, sensor1est1, sensor2est1, sensor3est1, sensor4est1,
sensor5est1, sensor6est1, sensor7est1, sensor8est1, sensor9est1, sensor0est2, sensor1est2, sensor2est2,
sensor3est2, sensor4est2, sensor5est2, sensor6est2, sensor7est2, sensor8est2, sensor9est2,
velPasajeros,velMercancias,marcadorVeloPasaje1, marcadorVeloPasaje2, marcadorVeloMercan1,
marcadorVeloMercan2, semaf1Est1Rojo, semaf1Est1Verde, semaf2Est1Rojo, semaf2Est1Verde,
semaf3Est1Rojo, semaf3Est1Verde, semaf4Est1Rojo, semaf4Est1Verde, semaf5Est1Rojo, semaf5Est1Verde,
semaf6Est1Rojo, semaf6Est1Verde, semaf7Est1Rojo, semaf7Est1Verde,
semaf8Est1Rojo,semaf8Est1Verde,allSemafEst1,byte1SemafEst1,byte2SemafEst1,semaf1Est2Rojo,semaf1Est2V
erde, semaf2Est2Rojo, semaf2Est2Verde, semaf3Est2Rojo, semaf3Est2Verde, semaf4Est2Rojo,
semaf4Est2Verde, semaf5Est2Rojo, semaf5Est2Verde, semaf6Est2Rojo, semaf6Est2Verde, semaf7Est2Rojo,
semaf7Est2Verde, semaf8Est2Rojo, semaf8Est2Verde, semaf9Est2Rojo, semaf9Est2Verde, semaf10Est2Rojo,
semaf10Est2Verde, allSemafEst2, byte1SemafEst2, byte2SemafEst2, desvio1, desvio2, desvio3,desvio4, desvio5,
desvio6, sistemaAutomatico, sistManualLibre, sistManual, sistemaOn, emergencia, trenPasajeActivo,
trenMercanActivo, condInicialesAuto, liberarFrenoPasaje, liberarFrenoMercan, rutaSeleccionada, permisoRuta1,
ruta1seleccionada, permisoTodasRutas, ruta2seleccionada, permisoRuta2, bucleRutasSeleccionado,
permisoBucle1, permisoTodosBucles, bucleRuta1seleccionado,voltaje319316,voltaje319402)

values('26/12/2016','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','FALSE',
'FALSE','FALSE','FALSE','FALSE','FALSE','FALSE','FALSE','FALSE','FALSE',20,30,2,3,4,5,'TRUE','FALSE',
'TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE',
'FALSE',1,1,1,'TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE',
'FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE',0,0,0,'TRUE','TRUE','TRUE','TRUE
','TRUE','TRUE','TRUE','FALSE','FALSE','TRUE','FALSE','TRUE','TRUE','TRUE','TRUE','TRUE',1,'TRUE',
'TRUE',1,'TRUE','TRUE','TRUE','TRUE',1,'TRUE',2.5,1.2);


insert into web_tfg.scada(fechayHora, sensor0est1, sensor1est1, sensor2est1, sensor3est1, sensor4est1,
sensor5est1, sensor6est1, sensor7est1, sensor8est1, sensor9est1, sensor0est2, sensor1est2, sensor2est2,
sensor3est2, sensor4est2, sensor5est2, sensor6est2, sensor7est2, sensor8est2, sensor9est2,
velPasajeros,velMercancias,marcadorVeloPasaje1, marcadorVeloPasaje2, marcadorVeloMercan1,
marcadorVeloMercan2, semaf1Est1Rojo, semaf1Est1Verde, semaf2Est1Rojo, semaf2Est1Verde,
semaf3Est1Rojo, semaf3Est1Verde, semaf4Est1Rojo, semaf4Est1Verde, semaf5Est1Rojo, semaf5Est1Verde,
semaf6Est1Rojo, semaf6Est1Verde, semaf7Est1Rojo, semaf7Est1Verde,
semaf8Est1Rojo,semaf8Est1Verde,allSemafEst1,byte1SemafEst1,byte2SemafEst1,semaf1Est2Rojo,semaf1Est2V
erde, semaf2Est2Rojo, semaf2Est2Verde, semaf3Est2Rojo, semaf3Est2Verde, semaf4Est2Rojo,
semaf4Est2Verde, semaf5Est2Rojo, semaf5Est2Verde, semaf6Est2Rojo, semaf6Est2Verde, semaf7Est2Rojo,
semaf7Est2Verde, semaf8Est2Rojo, semaf8Est2Verde, semaf9Est2Rojo, semaf9Est2Verde, semaf10Est2Rojo,
semaf10Est2Verde, allSemafEst2, byte1SemafEst2, byte2SemafEst2, desvio1, desvio2, desvio3,desvio4, desvio5,
desvio6, sistemaAutomatico, sistManualLibre, sistManual, sistemaOn, emergencia, trenPasajeActivo,
trenMercanActivo, condInicialesAuto, liberarFrenoPasaje, liberarFrenoMercan, rutaSeleccionada, permisoRuta1,
ruta1seleccionada, permisoTodasRutas, ruta2seleccionada, permisoRuta2, bucleRutasSeleccionado,
permisoBucle1, permisoTodosBucles, bucleRuta1seleccionado,voltaje319316,voltaje319402)

values('27/12/2016','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE', 'FALSE',
'FALSE','FALSE','FALSE','FALSE','FALSE','FALSE','FALSE','FALSE','FALSE',20,30,2,3,4,5,'TRUE','FALSE',
'TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE',
'FALSE',1,1,1,'TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE',
'FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE','TRUE','FALSE',0,0,0,'TRUE','TRUE','TRUE','TRUE
','TRUE','TRUE','TRUE','FALSE','FALSE','TRUE','FALSE','TRUE','TRUE','TRUE','TRUE','TRUE',1,'TRUE',
'TRUE',1,'TRUE','TRUE','TRUE','TRUE',1,'TRUE',3.5,8.2);
```

14.3. Ficheros de configuración del proyecto

Fichero “persistence.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
            http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
      version="1.0">
   <persistence-unit name="web_tfg">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:jboss/postgresDS</jta-data-source>
     <properties>
         <property name="hibernate.dialect"
                   value="org.hibernate.dialect.HSQLDialect"/>
         <property name="hibernate.hbm2ddl.auto" value="create-drop"/>
      </properties>
   </persistence-unit>
</persistence>
```

Fichero “application.xml”

```
<application>
  <display-name>WebTFG</display-name>
  <module>
    <web>
      <web-uri>WebTFG.war</web-uri>
      <context-root>/WebTFG</context-root>
    </web>
  </module>
  <module>
    <ejb>WebTFG.jar</ejb>
  </module>
</application >
```

Fichero “build.xml”

```
<project name="TFG Web service" default="all" basedir=".">
  <description> UOC TFG </description>
  <!-- definition of global property -->
  <property environment="env"/>
  <property name="jboss.home" value="${env.JBOSS_HOME}"/>
  <property name="source" value="."/>
  <property name="sourcesrc" value="${source}/src"/>
  <property name="build" value="${source}/build"/>
  <property name="buildjar" value="${build}/jar"/>
  <property name="buildwar" value="${build}/war"/>
  <property name="dist" value="${source}/dist"/>
  <property name="jboss-config" value="default"/>
  <property name="deploy" value="${jboss.home}\standalone\deployments"/>
  <property name="jboss.module.dir" value="${jboss.home}/modules" />

  <path id="jboss.classpath">
      <fileset dir="${jboss.module.dir}">
            <include name="**/*.jar"/>
      </fileset>
  </path>

  <target name="all" depends="clean, init, compileEjb, compileWar, jarEjb,
deployClient,
      ear, deployear"/>
```

```
  <target name="init"
            description = "inicialitzacions is relevant: the structure created
            copy files and directories there. xml ">
    <!-- Crea el time-stamp -->
    <tstamp/>
    <!-- It creates the directory structure -->
    <mkdir dir="${buildjar}"/>
    <mkdir dir="${buildwar}"/>
    <mkdir dir="${buildjar}/META-INF"/>
    <mkdir dir="${buildwar}/WEB-INF"/>
       <mkdir dir="${buildwar}/WEB-INF/classes"/>
       <mkdir dir="${dist}"/>
   </target>

  <!--Compiling the EJB classes and makes the build directory -->
  <target name="compileEjb" depends="init">
       <copy file="${sourcesrc}/META-INF/persistence.xml" todir="${buildjar}/META-
INF"/>
       <copy file="${sourcesrc}/log4j.properties" todir="${buildjar}"/>
   <javac srcdir="${sourcesrc}" destdir="${buildjar}"
            includes="ejb/*.java, jpa/*.java"
         classpathref="jboss.classpath"
       includeantruntime="true"
    />
  </target>

  <!-- Compile the client application, creating the structure buildwar -->
  <target name="compileWar" depends="compileEjb">
       <copy todir="${buildwar}">
             <fileset dir="${source}/docroot"/>
       </copy>
       <javac srcdir="${sourcesrc}" destdir="${buildwar}/WEB-INF/classes"
includes="/managedbean/*.java"
        classpathref="jboss.classpath"
       includeantruntime="true"
       />
  </target>

  <!-- Update the EJB jar file and create if not exist -->
  <target name="jarEjb" depends="compileEjb">
    <jar jarfile="${dist}/WebTFG.jar"
      basedir="${buildjar}"
      update="yes">
    </jar>
  </target>

  <!-- Update the WAR file and create if not exist -->
  <target name="deployClient" depends="compileWar">
    <jar jarfile="${dist}/WebTFG.war"
      basedir="${buildwar}"
      excludes="/WEB-INF/classes/ejb/*.*, /WEB-INF/classes/jpa/*.*"
      update="yes">
    </jar>
  </target>

  <!-- Update the application ear file and created if not exist -->
  <target name="ear" depends="clean, jarEjb, deployClient">
    <copy file="${sourcesrc}/META-INF/application.xml" todir="${dist}/META-INF"/>
       <jar jarfile="${dist}/WebTFG.ear"
          basedir="${dist}" update="yes">
       </jar>
  </target>

  <!-- Deploy the ear. Copy the ear of the JBoss deployment directory -->
```

```
  <target name="deployear" depends="ear">
    <copy file="${dist}/WebTFG.ear" todir="${deploy}"/>
  </target>

  <!-- Clean the build directory -->
  <target name="clean">
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```

15. Implementación de la aplicación software de detección de obstáculos en la vía.

El objetivo de esta aplicación consiste en diseñar un sistema de búsqueda de patrones mediante operaciones de correlación. Este método debe ser capaz de detectar posibles obstáculos en una región específica de la vía. Se aplicará este método a las imágenes obtenidas mediante la cámara de vigilancia, debiendo sintonizar adecuadamente el intervalo de correlación para que las coincidencias con el patrón (matching) sean lo más fiables posibles.

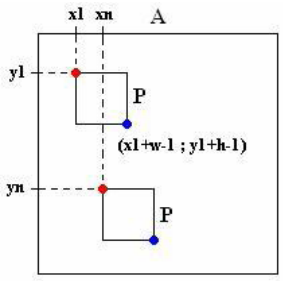
A continuación, se realiza una breve introducción sobre el sistema de búsqueda de patrones por correlación en una imagen.

Búsqueda de patrones

La búsqueda de patrones es una técnica de análisis que se puede aplicar en detección de objetos, reconocimiento, seguimiento y correspondencia. La idea es: dada una imagen (patrón) encontrar sus apariciones dentro de otra imagen mayor. No se buscan sólo las apariciones “exactas”, sino que se permite cierto grado de variación respecto al patrón.

Uno de los métodos más sencillos de búsqueda de patrones es lo que se denomina “template matching” (comparación de plantillas).

Template matching: Sea A una imagen (de tamaño W x H), y sea P un patrón (de w x h), el resultado es una imagen M (de tamaño (W-w+1) x (H-h+1)), donde cada píxel M(x,y) indica la “verosimilitud” (probabilidad) de que el rectángulo [x,y]-[x+w-1,y+h-1] de A contenga el patrón P. Gráficamente esto se expresa de la siguiente forma:



La imagen M se define usando alguna función de similitud entre dos trozos de imagen.

Se debe tener en cuenta que para poder cuantificar estas posibles similitudes es necesario encontrar una forma de adjudicar y restringir los resultados. Esto se consigue mediante la normalización de los resultados. Como veremos más adelante con la función de correlación cruzada normalizada de Matlab, los

resultados se encontrarán en un rango de -1 a 1. Cuanto más alto sea el resultado (más cercano a 1) más será la probabilidad de que el patrón buscado se encuentre en la posición comprobada. Esto se interpreta como una correlación entre imágenes, siendo esta la forma de encontrar los patrones aplicados en esta práctica.

Los métodos de búsqueda de patrones por correlación se basan en la comparación de la imagen a clasificar con una o varias imágenes patrón que caracterizan a zonas de la imagen. Una forma de comparación entre la imagen buscada y las zonas en la imagen principal es ver cuál es la diferencia píxel a píxel y realizar la suma cuadrática de esas diferencias. El problema de utilizar este método es que esta comparación es errónea si la imagen ha sufrido un cambio de iluminación. Para solucionar esto se utiliza la correlación normalizada. Este método se basa en eliminar las variaciones globales de iluminación en la

imagen y en evaluar las diferencias en desviación respecto de la media en una ventana con la desviación respecto de la media en la otra ventana. La fórmula que modela este método es la siguiente:

$$\rho = \frac{\sum_{ij}\left(W_{ij}^{a} - \overline{W}_{ij}^{a}\right)\left(W_{ij}^{b} - \overline{W}_{ij}^{b}\right)}{\sqrt{\sum_{ij}\left(W_{ij}^{b} - \overline{W}_{ij}^{b}\right)^{2}}\sqrt{\sum_{ij}\left(W_{ij}^{a} - \overline{W}_{ij}^{a}\right)^{2}}} \qquad \rho \in [-1,1]$$

Esto es lo que se denomina coeficiente de correlación, que será 1 cuando las dos subventanas sean iguales (salvo en todo caso por una relación lineal, por ejemplo, que una de las imágenes sea igual que la otra, pero multiplicada por una constante) y 0 o con valor negativo hasta -1 cuando no se encuentre relación entre los píxeles de una subventana con respecto a la otra.

El proceso que normalmente se sigue cuando se aplica este método es el siguiente:

Conseguir un patrón, P, representativo de la clase de objetos a buscar.

Aplicar la búsqueda de patrones (template matching) a la imagen, obteniendo M.

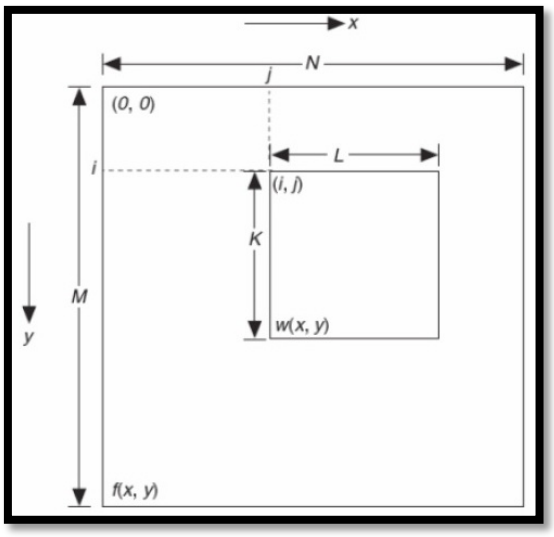
Buscar los máximos locales de M para encontrar los matches correspondientes. Este proceso se debe repetir hasta que se haya realizado la búsqueda en toda la imagen. En este paso también se puede introducir un cierto umbral de búsqueda, de forma que si M(x,y) es menor que el umbral, se pasa directamente al siguiente punto de búsqueda.

Debido a esta forma de búsqueda, esta técnica es muy sensible a cambios de escala y rotaciones. Para solucionar esto se podría realizar búsqueda en la imagen a diferentes escalas y probar con un rango definido de rotaciones posibles, pero esto supone un coste computacional elevado.

El método de correlación cruzada utilizado es uno de los métodos más utilizados en el reconocimiento de patrones. Este método se basa en una serie de multiplicaciones según la siguiente ecuación:

$$C(i,j) = \sum_{x=0}^{L-1}\sum_{y=0}^{K-1} w(x,y) f(x+i, y+j)$$

Considerando una subimagen w(x,y) de tamaño K x L dentro de una imagen f(x,y) de tamaño M x N, en donde K≤M y L≤N. La correlación entre w(x,y) y f(x,y) es C(i,j). De forma gráfica, el proceso de correlación se realiza de la siguiente manera:



La subimagen se recorre por toda la región de búsqueda. El máximo valor de C(i,j) indica la posición que mejor se ajusta a la imagen patrón. Se puede reducir el tiempo de procesamiento si se reduce el tamaño de la imagen patrón o el tamaño de búsqueda dentro de la imagen.

En la correlación se asume que añadir una constante a la señal o multiplicar una señal de media 0 por una constante no cambia la puntuación de la correlación, de forma que se minimiza la sensibilidad al cambio de iluminación. Sin embargo, como luego veremos gráficamente, la correlación si es sensible a cambios de escala y rotaciones.

16. Implementación de la aplicación software para la ayuda al mantenimiento preventivo de los componentes del sistema ferroviario.

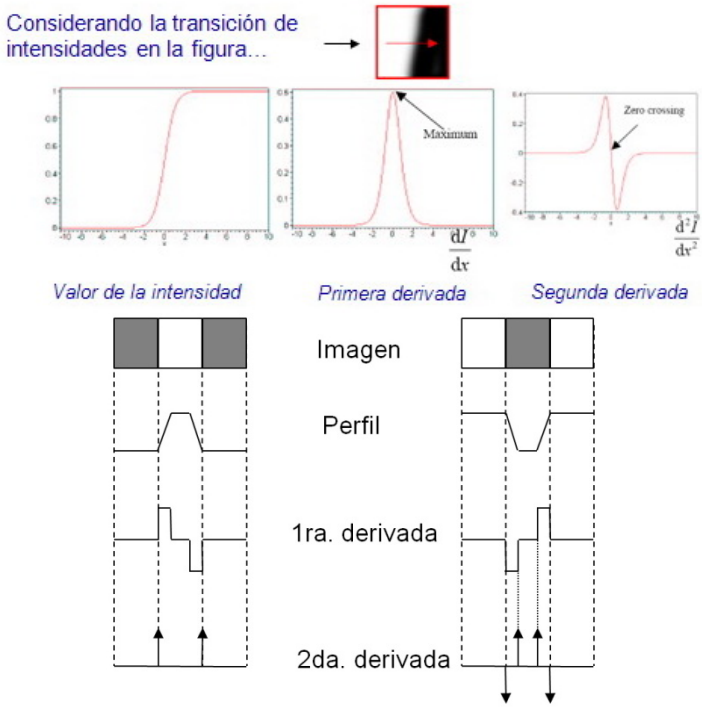
El objetivo de esta aplicación consiste en diseñar un sistema de detección de contornos mediante el operador de Sobel y el operador de Canny capaz de detectar posibles desperfectos en las piezas mecánicas y chasis del que se componen los convoys. Además, esta aplicación también se puede utilizar para detectar obstáculos en una región específica de la vía. Se aplicarán los dos métodos a las imágenes obtenidas mediante la cámara de vigilancia, debiendo sintonizar adecuadamente los dos detectores para que los contornos sean lo más fiables posibles.

A continuación, se realiza una breve introducción sobre los sistemas de detección de contornos.

Detectores de contornos

Un método detector de bordes o contornos está diseñado para ‘mejorar’ la imagen donde se presentan bordes, es decir, transiciones de un valor a otro en las intensidades de la imagen. Estos métodos se basan

en el cálculo de la primera o segunda derivada de los valores de intensidad de los píxeles de la imagen. Por ejemplo:



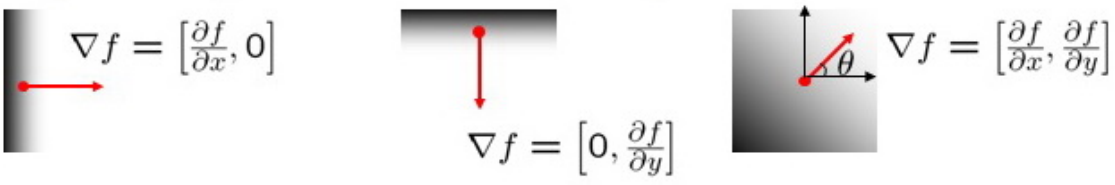
La primera derivada indica la presencia de un borde, mientras que la segunda derivada indica si un píxel pertenece a la parte "oscura" o "clara" del borde. Hay que tener en cuenta que las derivadas amplifican el ruido, por lo que será necesario filtrar previamente la imagen con la idea de suavizar en una dirección y derivar en la dirección perpendicular.

Algunos de los conceptos que se utilizan en el cálculo de contornos de una imagen son los siguientes:

- El gradiente de una imagen se expresa mediante:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- El gradiente apunta a la dirección de cambio más rápido en intensidades:



- La dirección del gradiente está dada por:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- La fuerza del gradiente está dada por su magnitud:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

- En una imagen discretizada, el diferencial de intensidades se puede aproximar mediante la diferencia de intensidades vecinas: I(x)-I(x-1). Por ejemplo, para una sección de 2x2: $df/dx = I_{1,2} - I_{1,1}$; $df/dy = I_{2,1} - I_{1,1}$

Detección de contornos mediante Canny

El método de Canny para la detección de bordes o contornos es considerado como uno de los mejores algoritmos para este trabajo. La detección de contornos es una de las tareas clave en el procesamiento de imágenes, puesto que facilita otras tareas como el reconocimiento de objetos, segmentación de regiones de una imagen, etc. El método de Canny está basado en la primera derivada y emplea máscaras de convolución para realizar aproximaciones con diferencias finitas entre puntos del contorno, siendo estos puntos zonas de píxeles en las que existe un cambio brusco de nivel de gris.

En 1986, Canny propuso este método para la detección de contornos, basándose en los siguientes criterios:

- Un criterio de detección expresa el hecho de evitar la eliminación de contornos importantes y no suministrar falsos contornos.
- Un criterio de localización establece que la distancia entre la posición real y la localizada del contorno se debe minimizar.
- Un criterio de una respuesta que integre las respuestas múltiples correspondientes a un único contorno.

El método de Canny utiliza la primera derivada para detectar un contorno, de forma que toma valor 0 en las regiones donde no varía la intensidad y valor constante en toda la transición de intensidad. Por tanto, cuando se produce un cambio brusco de intensidad se manifiesta en un cambio brusco en la primera derivada, y esto se detecta como un contorno (borde). El método consiste en tres pasos fundamentales:

- Obtención del gradiente: en este paso se calcula la magnitud y orientación del vector gradiente en cada píxel.
- Supresión no máxima: El objetivo es “adelgazar” el ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.
- Histéresis de umbral: Se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos.

En cuanto a la orientación del gradiente, lo primero que se hace es aplicar un filtro gaussiano a la imagen original con el objetivo de suavizar la imagen y tratar de eliminar el posible ruido existente en la misma. Es importante seleccionar adecuadamente los valores del filtro, puesto que si se suaviza demasiado se pueden perder detalles importantes en la imagen, como se ya se comentó en la práctica 2 de filtrado de imágenes. El suavizado se obtiene promediando los valores de intensidad de los píxeles en el entorno de vecindad con una máscara de convolución de media 0 y desviación típica σ. Una vez que se obtiene la imagen suavizada mediante el filtro, después se obtiene la magnitud y modulo (orientación) del gradiente. El algoritmo que se utiliza para obtener el gradiente es el siguiente:

- Entrada: Imagen I

  Máscara de convolución H, con media 0 y desviación típica σ.

- Salida: Imagen Em de la magnitud del gradiente.

  Imagen Eo de la orientación del gradiente.

1. Suavizar la imagen I con H mediante un filtro gaussiano y obtener J como imagen de salida.
2. Para cada píxel (i,j) en J, obtener la magnitud y orientación del gradiente basándose en lo siguiente:
   - El gradiente de una imagen f(x,y) en un punto (x,y) se define como un vector bidimensional dada la siguiente ecuación:

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix}$$

Esto constituye un vector perpendicular al borde, donde el vector G apunta en la dirección de variación máxima de f en el punto (x,y) por unidad de distancia, con la magnitud y orientación dadas por:

$$|G| = \sqrt{G_x^2 + G_y^2} = |G_x| + |G_y|$$

$$\phi(x,y) = \tan^{-1} \frac{G_y}{G_x}$$

3. Obtener Em a partir de la magnitud de gradiente y E0 a partir de la orientación, de acuerdo a las expresiones anteriores.

El resultado de este proceso son dos imágenes que se utilizan como entrada para generar una imagen con los bordes delgados. Para conseguir esto se deben considerar cuatro direcciones identificadas por las orientaciones de 0º, 45º, 90º y 135º con respecto al eje horizontal. Para cada píxel hay que encontrar la dirección que mejor se aproxime a la dirección del ángulo del gradiente. El algoritmo necesario sería el siguiente:

- Entrada: Imagen Em de la magnitud del gradiente.

  Imagen Eo de la orientación del gradiente.
- Salida: Imagen I_n
- Considerar: Cuatro direcciones d1, d2, d3, d4 identificadas por las direcciones 0º, 45º, 90º y 135º con respecto al eje horizontal.

1. Para cada píxel (i,j):
   1.1. Encontrar la dirección d_k que mejor se aproxima a la dirección $E_o(i,j)$ (perpendicular al borde).
   1.2. Si $E_m(i,j)$ es más pequeño que al menos uno de sus dos vecinos en la dirección d_k, al píxel (i,j) de I_n se le asigna el valor 0, $I_n(i,j)=0$ (supresión), de otro modo $I_n(i,j)=E_m(i,j)$.
2. Devolver I_n.

A continuación, se debe observar si el valor de la magnitud de gradiente es más pequeño que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en el paso anterior. Si es así, se asigna el valor 0 a dicho píxel. Si no, se asigna el valor que tenga la magnitud del gradiente.
La salida de este algoritmo es la imagen In con los bordes "adelgazados", es decir, Em(i,j) después de la supresión no máxima de puntos de borde. La imagen que se obtiene suele contener máximos locales creados por el ruido. Una solución para eliminar este ruido es la histéresis del umbral.
El proceso consiste en tomar la imagen obtenida en el paso anterior, la orientación de los puntos de borde de la imagen y dos umbrales, el primero más pequeño que el segundo. Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de este punto, hay que seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayores al primer umbral. De esta forma se marcan los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado. El algoritmo necesario sería el siguiente:

- Entrada: Imagen In obtenida en el paso anterior.

Imagen E_o de la orientación del gradiente.
Umbral t_1
Umbral t_2, donde $t_1<t_2$

- Salida: Imagen G con los bordes conectados de contornos.

1. Para todos los puntos de I_n y explorando I_n en orden fijo:
   1.1. Localizar el siguiente punto de borde no explorado previamente, $I_n(i,j)$, tal que $I_n(i,j)>t_2$
   1.2. Comenzar a partir de $I_n(i,j)$, seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde, siempre que $I_n>t_1$.
   1.3. Marcar todos los puntos explorados y, salvar la lista de todos los puntos en el entorno conectado encontrado.
2. Devolver G formada por el conjunto de bordes conectados de contornos de la imagen, así como la magnitud y orientación, describiendo las propiedades de los puntos de borde.

Aunque el algoritmo de Canny contempla estos pasos como los principales del proceso, se puede añadir un último paso que consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de ruido. Esto se debe a que el método de Canny tiene la desventaja de generar una gran cantidad de bordes abiertos.
Un método muy utilizado es el algoritmo de Deriche y Cocquerez. Este algoritmo utiliza como entrada una imagen binarizada de contornos de un píxel de ancho. El algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente hasta cerrarlos con otro extremo abierto.
El procedimiento, a grandes rasgos, consiste en buscar para cada píxel uno de los ocho patrones posibles que delimitan la continuación del contorno en tres posibles direcciones. Esto se logra con la convolución de cada píxel con una máscara específica. Cuando alguno de los tres puntos es ya un píxel de borde se entiende que el borde se ha cerrado. De lo contrario, se elige el píxel con el valor máximo de gradiente y se marca como nuevo píxel de borde, aplicándose nuevamente la convolución. Estos pasos se repiten para todo extremo abierto hasta encontrar su cierre o hasta llegar a un número determinado de iteraciones.

En resumen, el detector de Canny sigue estos criterios:

1) Buena detección: El detector óptimo debe minimizar la probabilidad de falsos positivos, así como de falsos negativos. Es decir, baja probabilidad de no detectar un punto de borde y baja probabilidad de marcar como borde un punto que no lo es.
2) Buena Localización: Los contornos detectados deben estar tan cerca como sea posible de los contornos reales (en concreto, lo más cerca posible del centro del verdadero contorno o borde).
3) Restricción de respuesta simple: El detector debe regresar a un punto solamente para cada contorno. Es decir, el detector debe dar una sola respuesta a un borde único.

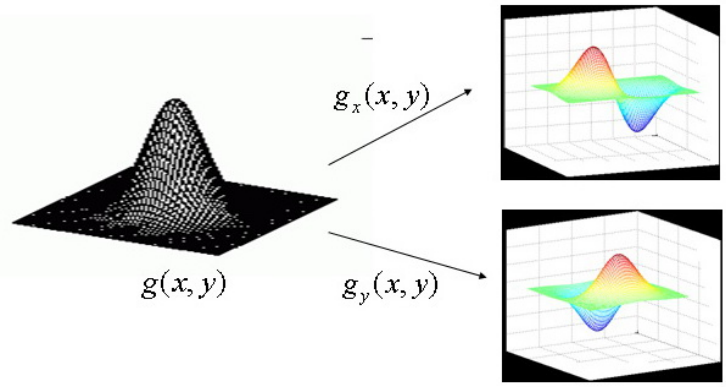
Los pasos que hemos visto anteriormente para implementar el detector de Canny se resumen de la siguiente forma:

1) Suavizar la imagen con un filtro Gaussiano y calcular la primera derivada de la imagen filtrada.

SUAVIZADO $\Rightarrow$ $S = I * g(x,y) = g(x,y) * I$ $\quad g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$

DERIVADA $\Rightarrow$ $\nabla S = \nabla(g * I) = (\nabla g) * I$

$$\nabla S = \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I = \begin{bmatrix} g_x * I \\ g_y * I \end{bmatrix} \qquad \nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$



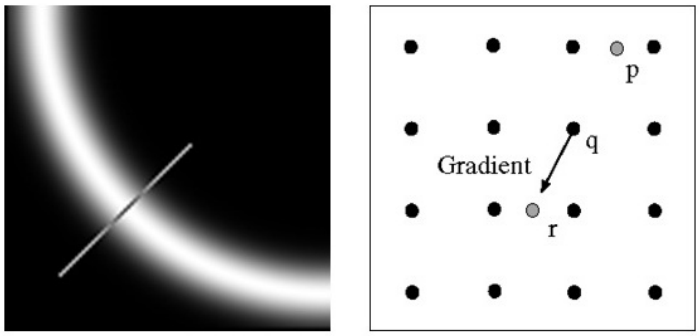
1) Encontrar la magnitud y orientación del gradiente.

(S_x, S_y) Vector gradiente

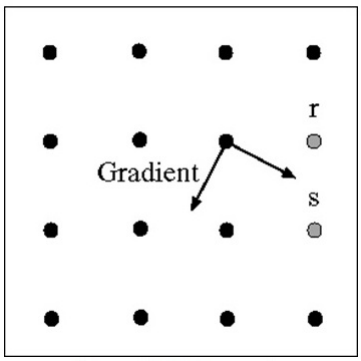
$$\text{magnitud} = \sqrt{(S_x^2 + S_y^2)}$$

$$\text{dirección} = \theta = \tan^{-1}\frac{S_y}{S_x}$$

2) Aplicar "Supresión de No-máximos.



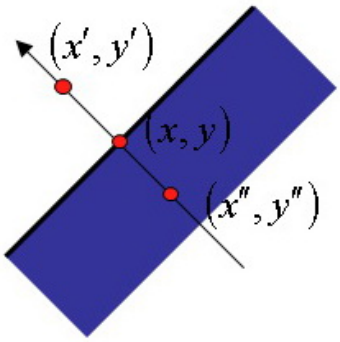
Comprobar si el píxel es un máximo local sobre la dirección del gradiente requiere verificar los píxeles interpolados 'p' y 'r'.
Se hace una predicción del siguiente punto de borde:



Suponiendo que el punto marcado es un punto de borde, se construye la tangente a la curva del borde (la cual es normal al gradiente en ese punto) y se utiliza para predecir el siguiente punto (ya sea 'r' o 's').

A continuación, se suprimen los píxeles en $|\nabla S|$ que no sean máximos locales.

$$M(x,y)=\begin{cases}|\nabla S|(x,y) & \text{if } |\Delta S|(x,y)>|\Delta S|(x',y') \\ & \&\,|\Delta S|(x,y)>|\Delta S|(x'',y'') \\ 0 & \text{en cualquier otro caso}\end{cases}$$



x' y x'' son los vecinos de x a lo largo de la dirección normal al borde.

Los vectores normales son cuantificados:

$$\tan\theta=\frac{S_y}{S_x}$$

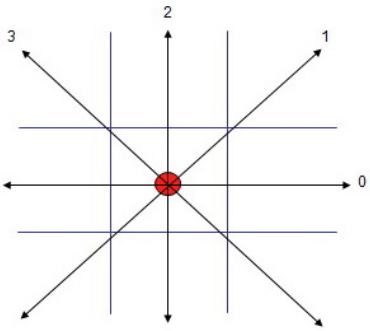
Quantizations:

0: $-0.4142<\tan\theta\le 0.4142$

1: $0.4142<\tan\theta<2.4142$

2: $|\tan\theta|\ge 2.4142$

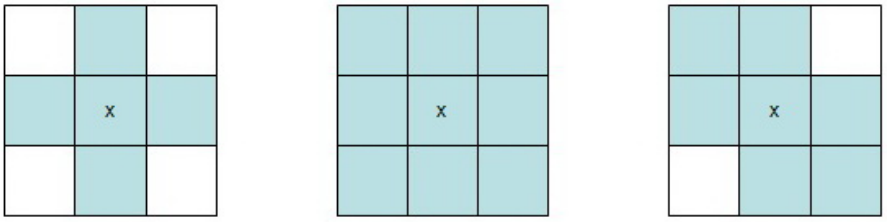
3: $-2.4142<\tan\theta\le -0.4142$



3) Aplicar "Umbral de Histéresis".

Si el gradiente en un píxel es:

- Mayor que "Alto", declararlo como píxel contorno.
- Menor que "Bajo", declararlo como píxel fuera de contorno.
- Entre "Bajo" y "Alto": Considerar sus vecinos de forma iterativa y entonces declararlo como un píxel contorno si está conectado a un píxel contorno ya sea directamente o vía píxeles entre "Bajo" y "Alto".



Conectividad-4 Conectividad-8 Conectividad-6

- La elección de σ depende del comportamiento que queramos asignar al detector:
  - ❖ σ grande: Detecta bordes a gran escala.
  - ❖ σ pequeña: Detecta características finas.

El método de detección de bordes de Canny dispone de gran adaptabilidad para el procesado de varios tipos de imagen, aunque una de sus principales desventajas es la etapa de suavizamiento, donde al aumentar σ demasiado se puede perder calidad en el suavizamiento y perjudicar el cálculo de la orientación.

Función "edge" de Matlab para la detección de contornos mediante Canny

En Matlab, la función que permite detectar contornos se denomina "edge". Esta función busca lugares en la imagen donde la intensidad cambia de forma repentina, utilizando para ello uno de estos dos criterios:

a) Lugares donde la primera derivada de la intensidad es mayor en magnitud a un determinado umbral.
b) Lugares donde la segunda derivada de la intensidad tiene un cruce por 0.

La función "edge" proporciona un número determinado de estimadores de la derivada, cada uno de los cuales implementa una de los dos criterios anteriores. Para algunos de estos estimadores es posible especificar si la operación debe realizarse sobre los contornos horizontales, verticales o ambos. La función "edge" devuelve una imagen binaria que contiene 1 en los bordes o contornos detectados y 0 en el resto de lugares.
El método más potente de detección de contornos que proporciona la función "edge" es el método de Canny, que difiere del resto en que utiliza dos umbrales diferentes (detección de bordes fuertes y débiles). Además, el método de Canny sólo incluye los bordes débiles en la salida si están conectados con bordes fuertes. Este método es menos propenso a ser afectado por el ruido y proporciona buenos resultados en la detección de bordes débiles de la imagen.
El método de Canny aplica dos umbrales al gradiente: un umbral alto para una sensibilidad de borde baja y un umbral bajo para una sensibilidad de borde alta. La función "edge" comienza con el resultado de baja sensibilidad y después lo hace crecer incluyendo los píxeles de borde conectados desde el resultado de alta sensibilidad. Esto permite rellenar huecos en los bordes detectados.
La sintaxis de la función es la siguiente:

a) **BW=edge(I,'canny')** → Método de Canny automático, ya que no se especifican manualmente los valores de umbral para el método y los selecciona automáticamente la función "edge".
b) **BW=edge(I,'canny',thresh)** → Permite especificar los valores de umbral para el método de Canny. 'Thresh' es un vector de dos elementos, donde el primer elemento es el umbral bajo y el segundo es el umbral alto. Si en vez de estos dos valores se especifica un único valor, la función "edge" ajusta el umbral bajo mediante la siguiente operación: 0.4*thresh. Si no se especifica thresh o si se especifica vacío ([]), la función "edge" ajusta los dos valores de umbral de forma automática (mismo caso que a)). El valor de thresh es relativo al mayor valor de la magnitud del gradiente de la imagen.
c) **BW=edge(I,'canny',thresh, sigma)** → Permite especificar, además de los valores de umbral para el detector, el valor de sigma como valor de la desviación típica del filtro Gaussiano. Si no se especifica sigma, la función "edge" ajusta automáticamente su valor en 1. El tamaño del filtro se selecciona automáticamente en función del valor de sigma.

d) **[BW,thresh]=edge(I,'canny',…)** → Devuelve los valores de umbral como un vector de dos elementos.

Detección de contornos mediante Sobel

Se define como un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen.
Se encuentra los bordes con la aproximación Sobel a la derivada espacial y retorna el borde en los puntos donde el gradiente I es máximo.
El operador de Sobel es uno de los denominados operadores de gradiente.
El operador gradiente aplicado sobre un píxel (x,y) de una imagen devuelve un vector que indica la dirección de máxima variabilidad de la intensidad luminosa y su nivel de variación:

$$\nabla f(x,y)=\begin{bmatrix}\frac{\partial f(x,y)}{\partial x}\\ \frac{\partial f(x,y)}{\partial y}\end{bmatrix} \Rightarrow \begin{cases}|\nabla f(x,y)|=\sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2+\left(\frac{\partial f(x,y)}{\partial y}\right)^2}\\ \angle\nabla f(x,y)=\arctan\left(\frac{\frac{\partial f(x,y)}{\partial y}}{\frac{\partial f(x,y)}{\partial x}}\right)\end{cases}$$

Un píxel se considerará que pertenece a un contorno si el módulo del gradiente supera un cierto umbral T:

$$g(x,y)=\begin{cases}1 & |\nabla f(x,y)|>T\\ 0 & |\nabla f(x,y)|<T\end{cases}$$

En la práctica, generalmente se establece como umbral, T, cuando el módulo del gradiente está por encima del 70% al 80% del valor máximo detectado en la imagen.
Para el cálculo del módulo del gradiente se suele emplear la aproximación de la suma de sus derivadas parciales, mejorando el coste computacional:

$$\left|\nabla f(x,y)\right|\cong\left|\frac{\partial f(x,y)}{\partial x}\right|+\left|\frac{\partial f(x,y)}{\partial y}\right|$$

La discretización del operador gradiente se basa en las diferencias de los niveles de grises en el entorno de vecindad. En el escenario numérico de aproximación de las derivadas por las diferencias, éstas suelen emplear el truncamiento del desarrollo de
Taylor obteniendo las diferencias progresivas y regresivas:

$$\frac{\partial f^+(x,y)}{\partial x}\approx\frac{f(x+\Delta x,y)-f(x,y)}{\Delta x}\quad \frac{\partial f^+(x,y)}{\partial y}\approx\frac{f(x,y+\Delta y)-f(x,y)}{\Delta y}$$

$$\frac{\partial f^-(x,y)}{\partial x}\approx\frac{f(x,y)-f(x-\Delta x,y)}{\Delta x}\quad \frac{\partial f^-(x,y)}{\partial y}\approx\frac{f(x,y)-f(x,y+\Delta y)}{\Delta y}$$

Un promediado de ambas se consigue las diferencias centradas:

$$\frac{\partial f(x,y)}{\partial x}\approx\frac{f(x+\Delta x,y)-f(x-\Delta x,y)}{2\Delta x}\quad \frac{\partial f(x,y)}{\partial y}\approx\frac{f(x,y+\Delta y)-f(x,y-\Delta y)}{2\Delta y}$$

Existen diferentes máscaras de convolución que implementan estas diferenciaciones. En un entorno de 2x2 se usa el operador de Roberts y en una máscara de 3x3 se aproximan por el operador de Prewitt, Sobel y el de Fre-Chen. Considerando el caso de un entorno de vecindad 3x3, las derivadas parciales serían:

$$\begin{pmatrix}f_{-1,-1} & f_{-1,0} & f_{-1,1}\\ f_{0,-1} & f_{0,0} & f_{0,1}\\ f_{1,-1} & f_{1,0} & f_{1,1}\end{pmatrix}\rightarrow\begin{cases}\left.\frac{\partial f}{\partial x}\right|_{0,0}\cong\frac{(f_{1,-1}+f_{1,0}+f_{1,1})-(f_{-1,-1}+f_{-1,0}+f_{-1,1})}{\Delta x}\\ \left.\frac{\partial f}{\partial y}\right|_{0,0}\cong\frac{(f_{-1,1}+f_{0,1}+f_{1,1})-(f_{-1,-1}+f_{0,-1}+f_{1,-1})}{\Delta y}\end{cases}$$

Los incrementos de las variables independientes, Dx e Dy, no se consideran ya que son valores constantes correspondientes a la discretización espacial y en la detección de borde sólo importa el valor de módulo relativizado.

De la expresión anterior se deducen las máscaras de Sobel:

∇_x

−1	0	1
−2	0	2
−1	0	1

∇_y

1	2	1
0	0	0
−1	−2	−1

Las matrices que se usan para su aproximación, como acabamos de ver, son dos kernels de [3x3]; el primero Gx se aplica para los cambios horizontales y el segundo Gy para las verticales mediante la convolución con la imagen original.
En cada punto de la imagen, los resultados de las aproximaciones de los gradientes horizontal y vertical pueden ser combinados para obtener la magnitud del gradiente, mediante:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Con esta información, podemos calcular también la dirección del gradiente:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

De forma experimental se ha observado que todos los operadores de gradiente ofrecen resultados similares, y que la respuesta se va deteriorando de acuerdo a la cantidad de ruido presente en una imagen. Debido a un menor coste computacional, el operador de Sobel suele ser el más utilizado de ellos.

Función “edge” de Matlab para la detección de contornos mediante Sobel

Otro método de detección de contornos que proporciona la función “edge” es el método de Sobel, que aplica un umbral para la detección.
La sintaxis de la función es la siguiente:

a) **BW=edge(I,’sobel’)** → Método de Sobel automático, ya que no se especifica manualmente el valor de umbral para el método y lo selecciona automáticamente la función “edge”.
b) **BW=edge(I,’sobel’,thresh)** → Permite especificar el valor de umbral para el método de Sobel. ‘Thresh’ especifica el umbral de sensibilidad para el método de Sobel, ignorando los contornos que sean menores que el umbral especificado.
   Si no se especifica thresh o si se especifica vacío ([]), la función “edge” ajusta el valor de umbral de forma automática (mismo caso que a)).
c) **BW=edge(I,’sobel’,thresh, direction)** → Permite especificar, además del valor de umbral para el detector, la dirección de detección que va a utilizar el método. El valor de “direction” es una cadena de caracteres que especifican si se van a buscar contornos horizontales (‘horizontal’), verticales (‘vertical’) o ambos (‘both’). El valor de “direction” por defecto para el método de Sobel es ‘both’.
d) **BW=edge(I,’sobel’,thresh,…, options)** → Permite especificar unas cadenas de caracteres opcionales que determinaran el rendimiento del propio detector (“thinning” y “nothinning”). La cadena ‘nothinning’ aumenta la velocidad de detección del algoritmo debido a que omite la etapa adicional de “adelgazamiento” de los contornos obtenidos. Cuando se especifica la cadena “thinning” el algoritmo aplica una técnica de “adelgazamiento” de los contornos, siendo éste también el comportamiento por defecto de la función edge con el método Sobel.
e) **[BW,thresh]=edge(I,’sobel’,…)** → Devuelve el valor de umbral.

f) **[BW,thresh,gv,gh]=edge(I,'sobel',…)** → Devuelve los contornos verticales y horizontales en función de los operadores de gradiente de Sobel. Además de esta expresión, puede utilizarse también la siguiente para obtener las respuestas de estos gradientes:

```
if ~(isa(I,'double') || isa(I,'single')); I = im2single(I); end
gh = imfilter(I,fspecial('sobel') /8,'replicate');
gv = imfilter(I,fspecial('sobel')'/8,'replicate');
```

17. Valoración económica.

17.1. Hardware.

17.1.1. Autómatas

Nombre del Material	NºUnidades	Precio / Unidad	Total
CPU314C-2DP 24ED/16SD	1	1.623,97 €	1.623,97 €
Micro Memory Card 2MB	1	245,70 €	245,70 €
Conector Tornillo 40 P	2	33,97 €	67,93 €
Perfil Soporte 480mm	1	27,48 €	27,48 €
Fuente Aliment. S7 5A	1	117,68 €	117,68 €
CPU226 NUEVA 24E/16S	2	487,06 €	974,11 €
Modulo Esclavo DP/MPI	2	193,60 €	387,19 €
PC Adapter USB Win2000	1	299,33 €	299,33 €
Cable USB/PPI	1	121,76 €	121,76 €
Cable PROFIBUS FAST 2h Apan	10	1,31 €	13,14 €
Tarjeta PROFIBUS CP5622	1	609,84 €	609,84 €
Módulo 2S Analógicas	2	169,94 €	339,89 €
Conector profibus 90º ES7972-0BB50-0XA0	3	48,62 €	145,85 €
Conector Profibus 180º axial	1	45,99 €	45,99 €
Fuente SITOP power 3,5A	2	96,74 €	193,48 €
Memoria 256Kb (Nueva)	2	58,69 €	117,38 €
fast connect stripping tool	1	57,23 €	57,23 €
Modem GSM TC35T	1	272,25 €	272,25 €
Kit para TC35T	1	27,60 €	27,60 €
Cable PC/PPI	1	104,24 €	104,24 €
Adaptador DB9M-DB9M	1	2,06 €	2,06 €
		Total:	**5.794,10 €**

17.1.2. Maquetas de tren

Nombre del Material	NºUnidades	Precio / Unidad	Total
Maqueta tren Mercancías 41256	1	260,00 €	260,00 €
Maqueta tren Pasajeros 41257	1	250,00 €	250,00 €
Motor accionamiento de vía universal para desvíos 61195	6	16,40 €	98,40 €
Decodificador de vía para desvíos 61196	6	20,40 €	122,40 €
Vía recta de 76,5 mm 61112	3	76.20 €	76.20 €
Vía recta de 185 mm 61111	2		
Cambio de vía derecha largo 200mm 61141	2		
Cambio de vía de desvío der. ¾ 434,5 mm 61155	2		
Vía arco invertido 2GB radio 502,7 mm 61128	2		
Tableros de madera, caballetes y soportes de hierro.	1	100,00 €	100,00 €
		Total:	**830,80 €**

17.1.3. Sensores

Nombre del Material	NºUnidades	Precio / Unidad	Total
SIEMENS 3RG4023-0AG01 Detector de Proximidad Bero	9	18,51 €	166,61 €
SIEMENS 3RX7301 Soporte de Montaje Ajustable P	9	7,44 €	66,98 €
Schneider Electric XS118B3PAL2 Sensor inductivo	9	31,39 €	282,51 €
BALLUFF 9027593 Soporte de montaje para BOS 18	9	3,20 €	28,80 €
Sick WTB4-3P1361 Sensor Fotoeléctrico, Sistema Difuso, LED	1	169,00 €	169,00 €
Sick GTB10-P1212 sensor Fotoeléctrico, Sistema Supresión de Fondo, LED	1	102,00 €	102,00 €
SIEMENS 6EP1333-2AA01 Fuente SITOP Smart 5A	2	94,38 €	188,76 €
		Total:	**1.004,66 €**

17.1.4. Punteras y cables

Nombre del Material	NºUnidades	Precio / Unidad	Total
Cable H05V-K(0,5 mm) Negro	300 metros	9,28 €/100m	27,84 €
Cable H05V-K(1 mm) Azul	200 metros	9,51 €/100 m	19,02 €
Cable H05V-K(1 mm) Negro	200 metros	9,51 €/100 m	19,02 €
Puntera 0,50 mm Blanca	100	0,03 €	2,97 €
Puntera 1 mm Roja	100	0,03 €	3,01 €
Puntera 1 mm Roja larga	500	30,11 €	30,11 €
Puntera 1,50 mm Negra	100	0,03 €	3,11 €
		Total:	**105,08 €**

17.1.5. Materiales diversos

Nombre del Material	NºUnidades	Precio / Unidad	Total
Led verde difuso L02500H1D1 5 mA 5mm	18	0,24 €	4,28 €
Led rojo difuso alta eficacia 5mA	18	0,24 €	4,28 €
Resistencia de película metálica MBB 1K1 0.6W	36	0,02 €	0.792 €
Terminal macho vert. Perfil bajo PCB 3 vías	18	0,76 €	13,61 €
Relé min montaje en PCB DPCO 8A 24Vdc	6	2,65 €	15,90 €
Zócalo para relé DPCO serie 40 carril Din	6	3,69 €	22,14 €
Diodo module 99.02.300000 6 220Vdc	6	2,18 €	13,05 €
Barra de conexión para relé	1	1,69 €	1,69 €
Seta parada emergencia disparo giratoria	1	29,91 €	29,91 €
Botón pulsador de doble cabezal 24Vac/dc	3	25,82 €	77,46 €
Interruptor conmutable 1P c/2 posiciones	1	15,77 €	15,77 €
Indicador de estación de control, led, de color verde.	1	11,72 €	11,72 €
Indicador de estación de control, led, de color naranja.	1	18,26 €	18,26 €
Indicador de estación de control, led, de color rojo.	1	11,72 €	11,72 €
		Total:	**239,79 €**

18. Lista de Riesgos

18.1. Propósito.

Se ha creado una lista de riesgos con la intención de poder aclarar y evitar todos los posibles contratiempos que pueda tener el proyecto, para lo cual se proponen distintas soluciones.

Lo que se realizará será una cuantificación de los riesgos mediante su probabilidad de que ocurran, así como el impacto hacia el proyecto.

18.2. Descripción.

La lista de posibles riesgos viene definida por las siguientes valoraciones, que permite hacer una clasificación de cada riesgo:

- Cada riesgo tiene asociado una probabilidad de que ocurra dicho riesgo siendo: ‘0’ poco probable de que ocurra el riesgo, y ‘1’ la probabilidad de que ocurra seguro.
- Además, cada riesgo produce un cierto impacto cuyo valor va definido entre 0 y 10, siendo ‘0’ el valor que representa un riesgo poco perjudicial y ‘10’ informa que el riego es muy perjudicial para el proyecto.
- Otro valor representado es la exposición al riesgo que es el producto entre la probabilidad y el impacto. Toma valores entre 0, mínima exposición, y 10, máxima exposición al riesgo.
- También se adjuntará la información para la reducción del riesgo.

Uno de los objetivos en el desarrollo del proyecto es identificar, controlar y eliminar las fuentes de riesgo antes de que empiecen a afectar al cumplimiento de los objetivos del proyecto. En primer lugar, se debe identificar estos riesgos potenciales, por lo que se van a enumerar los que se prevén a simple vista:

A) Riesgos por el tamaño del hardware desarrollado.

Riesgos en cuanto a tamaño del producto: Hay que controlar el tamaño que va adquiriendo el producto (hardware utilizado). Este punto es muy importante, ya que aquí se controla el tamaño que adquiere la maqueta ferroviaria y su sistema de automatización. La idea es construir un prototipo fácilmente transportable y manejable al mismo tiempo.

Otro factor de riesgo importante es el tamaño que adquieren, en cuanto a software, las bases de datos creadas para el sistema. Hay que optimizar al máximo tanto los diferentes datos incluidos en el sistema como el almacenamiento de los mismos, ya que de otra forma habría que enfrentarse a bases de datos inmensas que podrían perjudicar el análisis de los datos obtenidos.

Uno de los riesgos más altos que se deben abordar es el número de cambios que habrá que hacer en la aplicación en la fase de desarrollo. Es muy peligroso, puesto que obligaría a considerar de nuevo variables tan importantes como el tiempo de ejecución y plazos de entrega del proyecto (desbordando completamente estos plazos e incumpliendo con los objetivos del proyecto). Además, se debe de tener cuidado con los cambios durante la programación del sistema automatizado, puesto que un cambio de filosofía del producto cuando ya se lleva la programación muy avanzada ocasionaría una reestructuración general del producto y el proyecto podría no tener viabilidad bajo esas condiciones.

B) Riesgos asociados al ambiente de desarrollo.

Hay que asegurarse de la disponibilidad de las herramientas adecuadas durante el desarrollo. Hay que tener en cuenta la utilización de herramientas de trabajo mecánico/eléctricas para la construcción de la

maqueta ferroviaria. Este tipo de herramientas (véase radial de corte, soldadora, etc), requiere un grado de preparación en su uso y conocer los riesgos derivados del mismo. Por otra parte, el sistema automatizado requiere la conexión de distintos equipos electrónicos, siendo imprescindible conocer su modo de conexión y funcionamiento.

C) Riesgos tecnológicos.

¿Tecnología de automatización/tarea a automatizar conocida o nueva? Hay que tener especial cuidado en la elección que se determine, ya que, si el entorno de automatización o el entorno de trabajo es nuevo y no hay familiaridad con el mismo, hay que prever formación adicional para el desarrollador del proyecto y tener en cuenta la merma de rendimiento que ello produciría por no disponer de tiempo de "entrenamiento de habilidades" de la persona que lo lleva a cabo.

D) Riesgos del negocio

Riesgo de aceptación por parte de cliente. Creación de una solución que asegure los requerimientos del cliente, presentando la información de una forma clara y con un manejo sencillo para los usuarios, pero que disponga de características avanzadas para el aprovechamiento del sistema con fines de diagnóstico y mantenimiento del mismo. Se deben planificar los tiempos de decisión y revisión de las diferentes versiones que llevarán a la versión final. Es el cliente quien debe certificar si el producto que le estamos ofreciendo en base a sus propios requerimientos es el adecuado y se permite la continuación del trabajo.

Riesgo de presupuesto. Si se gestiona mal el presupuesto (tanto a la hora de estimarlo como en la fase de desarrollo), se puede llegar a la situación caótica de haber destinado prácticamente todo el presupuesto del proyecto y no haber llegado a la solución final acordada con el cliente. Es muy importante controlar el gasto y la evolución del proyecto y contemplar posibles problemas de desarrollo con el consiguiente consumo económico que llevaría solucionarlos (es decir, imprevistos, problemas "no visibles" hasta la fase de desarrollo, etc.).

Para que estos riesgos no aparezcan o si ya han aparecido darles solución habrá que:

Ante los posibles cambios en la estructura del sistema de automatización en la fase de desarrollo, la solución más recomendable es fijar claramente las necesidades de la empresa, y los requisitos que el usuario final, que es quien lo va a manejar, estima absolutamente necesarios para desarrollar bien el trabajo.

En los riesgos del ambiente de desarrollo, existe la posibilidad que se necesiten herramientas para desarrollar el proyecto, esto se podría solucionar, cuando una vez que se han planteado las necesidades del cliente, se haga la planificación del proyecto, y una vez que estén especificadas las herramientas que se van a emplear, habrá que asegurarse que se dispone de dichas herramientas, o en caso de no disponer de ellas, habría que asegurarse de conseguirlas.

Los riesgos que van asociados a la elección de la tecnología, están en parte asociados a los riesgos antes comentados de formación necesaria para el desarrollo del proyecto. Si la elección para desarrollar el proyecto es muy nueva, se necesitará formación, mientras que, si la tecnología no es tan nueva, en la mayoría de los casos se puede evitar el gasto en formación. Aunque en la elección de la tecnología se tendrá en cuenta, las necesidades del proyecto.

Otro riesgo asociado a la tecnología, es la coordinación entre el software y el hardware, habrá que tener en cuenta a la hora de elegir la tecnología para desarrollar el proyecto, si se va a reutilizar equipos ya existentes, porque allí habrá que comprobar que el proyecto puede funcionar correctamente. Si los equipos, una vez hechas las comprobaciones se ve que no van a poder sacar el rendimiento necesario para el funcionamiento de la aplicación, conviene saberlo lo antes posible, para exponerlo en el presupuesto del proyecto.

Existe el riesgo de que una vez acabada la aplicación y preparada para funcionar, no sea lo que el cliente esperaba, o no cumpla con sus necesidades. Para solucionar lo primero, que no sea lo que esperaba, se van a presentar al cliente para su revisión, varias partes del proyecto funcionando y por separado, para que una vez acabado, el cliente no se lleve sorpresa y diga que eso no es lo que esperaba. Para darle solución a lo segundo, como se ha explicado antes, se tendrán las necesidades y requerimientos del cliente bien especificados antes de empezar a desarrollar la aplicación.

Otro posible riesgo es el de presupuesto, es decir, que se tenga todo el presupuesto asignado y el proyecto no esté concluido, lo cual lleva a tener un control riguroso de los gastos, prever posibles imprevistos, y explicar claramente al cliente, que, si las especificaciones del proyecto no están claras desde el principio y quiere hacer modificaciones más adelante, seguramente van a modificar el presupuesto.

18.3. Lista de riesgos y su impacto.

Nº	Categoría del riesgo	Probabilidad	Impacto	Exposición al riesgo	Pasos de mitigación del riesgo
1	Formación del desarrollador del proyecto.	0,2	0,4	0,8	- Incluir tiempo extra en las estimaciones para la formación inicial del desarrollador del proyecto. - Recibir sesiones de formación.
2	Insuficiente conocimiento acerca del negocio (sistema ferroviario).	0,5	5	2,5	- Buscar mayor asesoramiento por parte del cliente o expertos en el tema en las etapas más tempranas posibles.
3	Planificaciones poco realistas	0,2	0,5	1,0	- Establecer claramente los objetivos del proyecto con el cliente. - Preparar los recursos que van a ser utilizados con anticipación.
4	Cambios de requisitos.	0,5	8	4,0	- Acordar las especificaciones iniciales de requisitos con el cliente y obtener su firma. - Convencer al cliente de que los cambios afectan a la planificación del proyecto. - Establecer unos procedimientos estándar para manejar los cambios de requisitos.
5	Requisitos poco claros.	0,7	4,0	2,8	- Servirse de la lógica y la experiencia para suponer que es lo que el cliente busca; mantener a este informado y buscar su visto bueno. - Construir un prototipo que sirva de demostración al cliente para que pueda comprobar si se cumplen los requisitos que esperaba.
6	Baja del autor del proyecto.	0,2	10	2,0	- Se podría emplear tiempo que no estaba estimado en la planificación, en caso de que la enfermedad sea temporal y en caso de que fuese de larga duración, habría que plantear al cliente una bajada de requisitos a desarrollar para la entrega.

7	Interfaz poco intuitivo para el cliente y sistema inestable en funcionamiento hardware/software	0,4	3	1,2	- Ayuda y opinión del cliente en cuanto al diseño del interfaz y a la fiabilidad del sistema de gestión en su forma de operar. - Creación de prototipos - Presentación de varios diseños de interfaz al cliente para su elección. - Test de funcionamiento de todos los componentes hardware/software
8	Retrasos en la entrega de documentación o de prototipos.	0,7	7,5	5,25	- Mantener un margen de seguridad en el proyecto para evitar retrasos. - Tener documentación adecuada del trabajo realizado por el autor del proyecto.
9	Mala gestión de la Financiación (presupuesto agotado antes del cumplimiento)	0,3	10	3,0	- Prevenir al cliente sobre un posible aumento del presupuesto necesario por imprevistos y negociarlo.

Bibliografía consultada

[1] Riley Edwards, J., M. Hart, John, Todorovic, Sinisa et al., "Development of Machine Vision Technology for Railcar Safety Appliance Inspection", In Proceedings of the International Heavy Haul Conference Specialist Technical Session-High Tech in Heavy Haul, Kiruna, Sweden (pp. 745-752)

- Railwaymania magazine (modelismo ferroviario), "Normas NEM de modelismo ferroviario", nº 13, Marzo 2007. http://www.railwaymania.com/
- Rodríguez, José Ramón; García Mínguez, Jordi; Lamarca Orozco, Ignacio; "Gestión de proyectos informáticos: métodos, herramientas y casos", ISBN: 978-84-9788-568-3, Editorial UOC, 2000
- José Manuel Díez Aznar, Iniciación Al WinCC. Visualización, Monitorización Y Control De Procesos Industriales. Universidad Politécnica de Valencia, 2003.
- Ramón Piedrafita Moreno, Ingeniería De La Automatización Industrial. 2ª Edición Ampliada Y Actualizada. Ra-ma, 2004.
- Christian Diedrich, Thomas Bangemann, Profibus PA: Instrumentation Technology for the Process Industry. Oldenbourg Industrieverla.
- AQUILINO RODRIGUEZ PENIN, Comunicaciones Industriales. MARCOMBO, 2008.
- Mandado Pérez,Enrique/Marcos Acevedo, Jorge Fernandez Silva, Celoso/Armesto J, Autómatas Programables Y Sistemas De Automatización. MARCOMBO, 2009.
- Documentación oficial de cursos de Siemens en programación de autómatas y hardware.
- Documentación oficial del software de programación SIMATIC STEP 7 y Micro/WIN
- Documentación de normativa ferroviaria actualizada y sistemas de control ferroviario.
- Documentación oficial del software Scada WinCC v7.0
- Información recogida de numerosas fuentes en Internet, ya sean de fuentes oficiales de los fabricantes de los productos utilizados como de fuentes diversas.
- Irueste Lobo, José; "Mantenimiento de material rodante ferroviario". ISBN 9788496437326, CIE INVERSIONES EDITORIALES DOSSAT-2000, 2008
- "Nociones básicas ferroviarias 2ª ED" ISBN: 9788426714626, S.A. MARCOMBO, 2007
- Arques Patón, José Luis; "Ingeniería y gestión del mantenimiento en el sector ferroviario" ISBN: 9788479789169, DIAZ DE SANTOS, 2009
- González Fernández, Francisco Javier; Fuentes Losa, Julio; "Ingeniería Ferroviaria 2ª ED"UNED

- González Fernández, Francisco Javier; "Señalización y seguridad ferroviaria" Colegio de ingenieros de caminos, canales y puertos, ED Garceta,2016.
- Montes Ponce de León, Fernando;" Los sistemas de control de tráfico y señalización en el ferrocarril"; Universidad de Comillas,2011
- Villaronte Fernández-Villa, Juan Antonio; "Tecnología e ingeniería Ferroviaria. Tecnología de la vía 4ª ED", DELTA publicaciones, 2012
- Información técnica del material ferroviario utilizado de la compañía Roco
- Stan Ames, Rutger Friberg, Ed Loizeaux. Digital Commando Control –the comprehensive guide to DCC-, NMRA 1998
- IRIDA DA CUNHA;"El trabajo de fin de grado y de master, redacción, defensa y publicación", ISBN: 9788490643907, Editorial UOC, 2016
- Jiménez de Parga, Carlos; UML Aplicaciones en Java y C++; Ed RAMA, 2015
- Ttevens, Perdita; Pooley, Rob; UTILIZACION DE UML en ingeniería del software con objetos y componentes, 2ª Ed, Addison Wesley, 2010
- Sznajdleder, Pablo Augusto; Java a Fondo, 2ª Ed, Marcombo, 2013
- Pérez Martínez, Eugenia; Hibernate Persistencia de objetos en JEE, Ra-Ma, 2015
- Ceballos, Fco. Javier; Java, Interfaces gráficas y aplicaciones para internet 4ª ED, Ra-Ma, 2015
- Rodríguez Penin, Aquilino; Sistemas SCADA Guía Practica, Marcombo
- Orbegozo Arana, Borja; Curso práctico avanzado de PostgreSQL La base de datos más potente, Altaria, 2014
- "Human Machine Interface (HMI) Design: The Good, The Bad, and The Ugly (and what makes them so)", 66th Annual Instrumentation Symposium for the Process Industries January 27-29, 2011
- ISA_101_HMI_Workshop.pdf
- "ISA_101_HMI_standard_nears_completion", Publication of the International Society of Automation, July-August 2014.
- "High Performance Graphics to Maximize Operator Effectiveness", PAS 2012