Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Máster en Informática Industrial

Título del TFM

Gestión de un motor lineal LinMot mediante interfaz EtherCat

Autor:

Gordo Martín, Juan Manuel

Tutor:

González Sánchez, José Luis

Ingeniería de sistemas y automática

Valladolid, septiembre de 2018.

Resumen

Se pretende desarrollar la capa de aplicación para una interfaz EtherCat.
Los objetivos de este Trabajo fin de máster son:

- Gestión desde Windows: en el que se parametrizará y configurará el controlador del motor, se generarán los movimientos y se hará el protocolo de nivel de operación, todo esto con una conexión RS-232 entre el ordenador de Windows y el controlador del motor LinMot.
- Gestión desde Linux: En el se instalará y configurará el driver del maestro EtherCat, de la compañía Hilscher, y se crearán las funciones, en lenguaje de programación C, para poder establecer conexión enviar y recibir datos por EtherCat, para realizar esta parte se utilizará un ordenador Linux con el controlador del motor Ethercat todo ello conectado con el bus EtherCat.

Palabras claves

- Sistema operativo Linux.
- EtherCat.
- Lenguaje de programación C.
- Maestro Hilscher para EtherCat.
- Ingeniería Software.

Índice

1. Introducción

En este trabajo se pretende controlar en tiempo real un motor lineal, para en el futuro en el departamento de Ingeniería de sistemas y automática poder hacer el control en tiempo real, pero de varios motores conectado por un bus de campo.

El motor cuenta con un encoder con ello el controlador sabe en qué posición se encuentra y con el podemos saber si ya ha llegado a la posición comandada o no.

El motor y el controlador utilizados son de la empresa LinMot, a continuación, se explica en más detalle los motores que ofrece está empresa y el que disponemos en el laboratorio.

Tipos de motores que ofrece LinMot:

- Motores lineales:

  Los motores lineales son usados cuando los movimientos de translación tengan que ser ejecutados de forma dinámica, con baja fricción y un alto grado de flexibilidad.
  La parte principal de los sistemas de motorización LinMot consiste en la electrónica de control, el servo controlador y los motores lineales.
  Los motores LinMot son motores electromagnéticos directos de formato tubular. El movimiento lineal es generado de manera totalmente eléctrica y sin desgaste, sin ningún tipo de transmisión intermedia de reductoras mecánicas, husillos, engranajes o correas. El motor lineal consta sólo de dos partes: el vástago y el estator. El vástago está compuesto de imanes de neodimio encapsulados en un tubo de acero inoxidable de alta precisión. El estator contiene los bobinados del motor, el casquillo de rozamiento para el vástago, los sensores de lectura de posición y un circuito microprocesador para la monitorización del motor. El sensor de posición interno mide y controla la posición actual del motor lineal no sólo cuando éste está parado, sino también mientras está en movimiento. Las variaciones de posición se detectan inmediatamente y se comunica al controlador. Los motores lineales LinMot pueden posicionar libremente dentro de todo el rango del recorrido del actuador. Además, se pueden controlar con precisión la velocidad del recorrido y la aceleración. Para movimientos más complejos se pueden generar y almacenar perfiles discretos tales como curvas personalizadas en el servo controlador y, posteriormente, ser ejecutados en el motor a la

velocidad deseada. En la figura 1.1 podemos observar una fotografía de las partes que se compone el motor lineal.

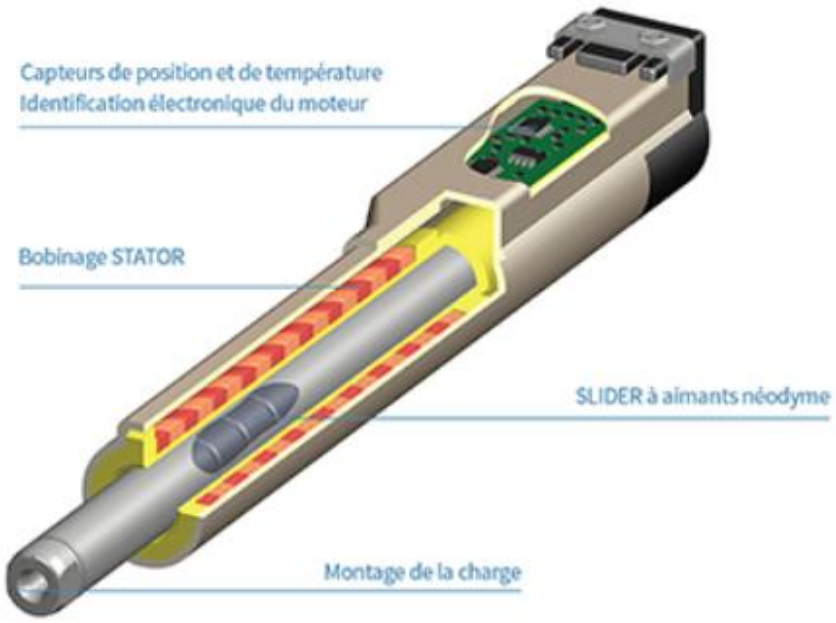

Figura 1.1: Partes del motor lineal.

Dentro de motores lineales tenemos los siguientes tipos:

- Motores estándares
  La versión estándar del motor lineal se puede usar de manera universal, abarcando una amplia gama de aplicaciones. Un gran número de diferentes tareas de posicionamiento se pueden realizar en el rango de baja tensión. La familia de motores incluye 3 modelos de estator y recorrido de diferentes longitudes. El usuario puede seleccionar un máximo de longitud de movimiento de 1830 mm y un máximo de fuerza de 1024 N. Debido a que los componentes del motor están encapsulados, el motor está protegido de forma óptima, incluso para aplicaciones de uso intensivo.
  En la figura 1.2 observamos una imagen de este tipo de motor lineal.

Figura 1.2: Motor lineal estándar.

- Motores HP

  La línea de alto rendimiento de motores lineales posee un mayor rendimiento que los motores estándar, con componentes del mismo tamaño e idénticas dimensiones. Esto se debe, en gran medida, al desarrollo continuo del bobinado del motor y del circuito magnético, así como el uso de materiales de alto rendimiento.

  Gracias a los motores de alto rendimiento, se tiene acceso a prácticamente el doble de la potencia nominal. Este mayor rendimiento con el mismo formato permite incrementar la energía en máquinas e instalaciones existentes, sin necesidad de un rediseño mecánico.

Figura 1.3: Motor lineal HP.

- Motores de corta longitud

  Este tipo de motor es el formato más corto para los motores lineales LinMot. Los motores de corta longitud han sido desarrollados especialmente para aplicaciones con restricciones espaciales. El formato corto también facilita las aplicaciones multiaxiales, donde se implementan varios estatores en un mismo vástago dentro de un espacio muy pequeño. Con el fin de proporcionar al diseñador tantas opciones como sean posibles

para el recorrido de cables, estos motores están provistos de tres coberturas para la salida del cable a la izquierda, a la derecha o en el lado frontal. El cable del motor puede conectarse bajo la cobertura para una instalación rápida y fácil.
En la figura 1.4 se muestra una imagen de estos motores.

Figura 1.4: Motor lineal de corta longitud.

- Motores lineales de 3x400VAC

Los motores de la serie P10-70 son las unidades más potentes que fabrica LinMot. Con su mayor rendimiento y máximos de fuerzas de hasta 2700 N, los motores imponen mayores exigencias para las unidades de drive y servo controladores. Estos motores funcionan 3x400VAC, así como servo controladores con fuente de alimentación directa de la red. Dependiendo de la aplicación, los motores también pueden funcionar con 1x230VAC rectificado.
Los motores pueden ser controlados por los controladores de cualquier fabricante.
El grupo de motores P10-54 consiste en motores lineales de tipo tubular, compactos, de potencia media, para tareas de posicionamiento dinámico o para reemplazar las soluciones neumáticas. Opcionalmente equipado con encoder 1Vpp sin/cos o encoder incremental A/B, esta serie de motores lineales puede ser controlada por controladores de otros fabricantes. En la figura 1.5 se muestra una imagen de estos motores.

Figura 1.5: Motor lineal 3x400VAC.

- Motores con guía integrada

  Los motores lineales del tipo P04 tienen todas las ventajas de los motores lineales de tipo tubular y además ofrecen la ventaja de una guía integrada.
  El actuador P04 puede ser equipado con accesorios mecánicos conocidos en la industria de neumática. Gracias a esta característica, el reemplazo de las partes neumáticas es más fácil, ya que se pueden realizar todas las opciones de montaje conocidas de neumática. Los movimientos dinámicos y precisos son la principal especialidad de este tipo de motor. Además, las cargas laterales son amortiguadas por el cojinete integrado para que las aplicaciones simples como el "empujador" puedan realizarse sin componentes adicionales. En la figura 1.6 se muestra una imagen de estos motores.

Figura 1.: Motor lineal con guía integrada.

- **Motores ATEX:**

  En un entorno donde se pueden generar gases explosivos, mezclas de aire-vapor o polvo inflamable, se requieren accionamientos eléctricos especiales.
  LinMot ATEX, una variante de motores lineales, ha sido desarrollada para esas condiciones especiales.
  Los motores están completamente encapsulados en acero inoxidable y no requieren juntas. Todas las uniones están soldadas. Para un cierre completo del motor, los bobinados y el resto de los estatores del interior han sido encapsulados con resina epoxi. De este modo, se garantiza la protección óptima del motor y se elimina el riesgo de chispas hacia el exterior.
  Dado que en una atmósfera explosiva la temperatura de la superficie es de crucial importancia, se ha integrado en el motor un control adicional

de temperatura. Los motores pueden ser equipados, de forma opcional, con un sistema de refrigeración por fluido.
La gama de motores ATEX ofrece dos tamaños de potencia con un amplio rango de longitudes para el vástago. El desplazamiento varía hasta un recorrido de 980 mm.
En la figura 1.2 podemos observar este tipo de motores.

Figura 1.2: Motores ATEX.

- Motores de acero inoxidable:

  Diseñados para entornos exigentes, estos motores lineales compactos están desarrollados completamente de acero inoxidable. Respecto al diseño del motor, la atención se centra en un estilo higiénico. Para evitar los depósitos de suciedad, el motor ha sido fabricado sin bordes innecesarios, esquinas u orificios. Además, todas las juntas han sido soldadas, por lo que no se necesitan componentes de unión.
  Los bobinados del estator están completamente encapsulados con resina epoxi, de modo que el relleno de cobre y el estator están protegidos de la condensación o la corrosión. Gracias a este tipo de encapsulado, los motores también están protegidos contra la penetración de polvo y agua (limpieza por chorro con presión/vapor). Esta impermeabilidad significa que cumplen con la protección IP69K según la norma DIN EN 60529. Los cojinetes de material sintético han sido especialmente desarrollados para facilitar la limpieza y ahorrar tiempo en los procesos de lavado.
  Para el control de la temperatura todos los motores lineales están equipados con sensores que transfieren los datos al controlador. De manera opcional, LinMot también ofrece una refrigeración por fluido integrada para temperaturas superficiales bajas, lo que conduce a un mayor rendimiento al tiempo que reduce el efecto de nucleación.

  En la figura 1.3 y 1.4 podemos observar este tipo de motores.

Figura 1.3: Motores de acero inoxidable.

Figura 1.4: Motores de acero inoxidable.

- <u>Motores lineales con driver integrado:</u>

  La unidad de drive PD03 consiste en un motor lineal compacto y un drive integrado. Tiene la posibilidad de acoplar eficientemente varios dispositivos en conexiones en cadena. Esta nueva generación de motores se puede utilizar para implementar fácilmente conceptos de máquinas modulares. La unidad del PD03 contiene un motor lineal de serie 37Sx120F-HP. Este motor de tipo corto puede producir un máximo de fuerza de 255 N y un máximo de velocidad de 3.8 m/s. Junto con el programa del vástago de alta resolución, el rango de recorrido es de hasta 1480 mm. Como todos los motores LinMot, este modelo también se puede colocar libremente, tiene una alta estabilidad de proceso y una larga vida útil, sin componentes de desgaste.
  El drive suministra al motor la corriente de fase máxima de hasta 25 A y tiene una interfaz Ethernet industrial. EtherCAT, ProfiNet, Ethernet IP, Sercos III, y Powerlink están disponibles. Los perfiles de dispositivo CoE, CiA402, SoE y PROFIdrive también son compatibles. La interfaz de configuración RS232 y el interruptor DIP están instalados detrás de una cubierta protectora lo que garantiza que se cumplan los requisitos de estanqueidad. El motor completo con drive integrado cumple entonces con la clase de protección IP 65 para evitar la infiltración de polvo y de agua.

En la figura 1.5 podemos observar el tipo de controlador para estos motores.

Figura 1.5: Motores lineales con driver integrado.

- <u>Motores rotolineales:</u>

La serie de motores rotolineales PR01 se puede utilizar para implementar cualquier combinación de movimientos lineales y rotativos. El cuerpo compacto contiene tanto el accionamiento lineal electromagnético como el accionamiento directo rotativo, cada uno controlado por su servo controlador. Esto significa que el controlador de nivel superior puede implementar secuencias de movimiento lineal-rotativo altamente dinámicas que pueden ser programadas para ser síncronas o completamente independientes entre sí.
Dependiendo de los requerimientos, el usuario puede seleccionar entre varios tamaños que generan un par de giro máximo que va desde 1.5 hasta 8.9 Nm. Las longitudes de recorrido lineal de hasta 300 mm garantizan la suficiente flexibilidad en cada aplicación concreta. Además de la versión estándar, la gama de productos incluye la variante con reductora, ejes huecos y componentes de acero inoxidable.

En la figura 1.6 podemos observar este tipo de motores.

Figura 1.6: Motores rotolineales.

Dentro de los motores rotolineales tenemos los siguientes tipos:

- Motor rotolineal estándar

  La versión estándar del motor rotolineal proporciona al usuario una amplia variedad de longitudes de recorrido disponibles, fuerza de empuje y par de giro.

  En la imagen 1.7 se puede observar este tipo de motor rotolineal.

Figura 1.7: Motores rotolineales estándar.

- Motor rotolineal versión con gama reductora

  El motor rotolineal con reductora permite un posicionamiento preciso y dinámico incluso para cargas con alto momento de inercia. La recogida y descarga de cargas fuera del eje de rotación es una aplicación clásica para este tipo de motor. También se pueden implementar con facilidad movimientos dinámicos y libremente programables con recorridos rotativos y lineales simultáneos gracias a este elemento de nuevo diseño.

  En la imagen 1.8 se puede observar este tipo de motor rotolineal.

Figura 1.8: Motores rotolineales versión con gama reductora.

- Motor rotolineal versión de eje hueco

  El motor rotolineal con reductora permite un posicionamiento preciso y dinámico incluso para cargas con alto momento de inercia. La recogida y descarga de cargas fuera del eje de rotación es una aplicación clásica para este tipo de motor. También se pueden implementar con facilidad movimientos dinámicos y libremente

programables con recorridos rotativos y lineales simultáneos gracias a este elemento de nuevo diseño.
En la imagen 1.9 se puede observar este tipo de motor rotolineal.

Figura 1.9: Motores rotolineales versión de eje de hueco.

- Motor rotolineal versión inoxidable

Además de los motores rotolineares estándar, la serie PR01-84 incluye versiones de acero al cromo. La brida frontal y el eje rotativo están desarrollados en acero cromado. El eje rotativo está sellado frente al estator. Esto significa que la tecnología rotolineal también puede utilizarse de manera óptima en las industrias alimentarias y químicas.
En la imagen 1.10 se puede observar este tipo de motor rotolineal.

Figura 1.10: Motores rotolineales versión inoxidable.

En nuestro caso el motor del que se dispone es un motor lineal, a este motor lineal le conectaremos un servo controlador, con bus de comunicación Ethercat.
Los servo controladores LinMot son controladores de posicionado compactos, con uno o más elementos de potencia, para motores actuadores, así como controlador inteligente para el control de posición de lazo cerrado integrado. El elemento de control se encarga de todas las funciones de control y de monitorización relacionadas con el motor. Se pueden utilizar posiciones definidas por el regulador general del

sistema, o ejecutar perfiles de movimiento almacenados internamente usando simples señales analógicas o digitales. La amplia variedad de servo controladores propuestos hace que sea posible implementar rápidamente aplicaciones simples con dos posiciones finales, así como aplicaciones multieje complejas, de alta precisión, sincronizadas a un eje electrónico maestro. Los controladores LinMot abarcan toda la gama de potencias para accionar, desde mini-motores compactos de baja corriente y niveles de baja tensión 24-72 Vdc, hasta servo-motores de alta potencia con alimentación directa por una red trifásica de hasta 3x480 Vac.

Figura 1.11: Servo controladores de LinMot.

En la imagen 1.11 podemos observar los distintos servo controladores de LinMot disponibles, el disponible para realizar este trabajo es con bus de comunicación EtherCat, este controlador se explicará más adelante.

2. Objetivos

En este trabajo se aborda el control de un motor lineal mediante un ordenador como maestro de bus de comunicación, luego en el departamento de Ingeniería de sistemas y Automática lo que se quiere hacer es el control de varios motores lineales conectados a un controlador.
El control de estos motores se realiza en la primera parte por un controlador comercial configurándolo con su propio software y parametrizándolo, en la segunda parte del trabajo el controlador pasa a ser esclavo de un bus de comunicación y solo hace lo que le envía el maestro del bus.
El maestro del bus es un ordenador con un sistema operativo Linux, en el que mediante un programa escrito en c le manda por el bus de campo, EtherCat, las tramas de envío con lo que tiene que hacer el motor, el esclavo también envía para saber el estado del motor.

3. Configuración del controlador y del motor lineal

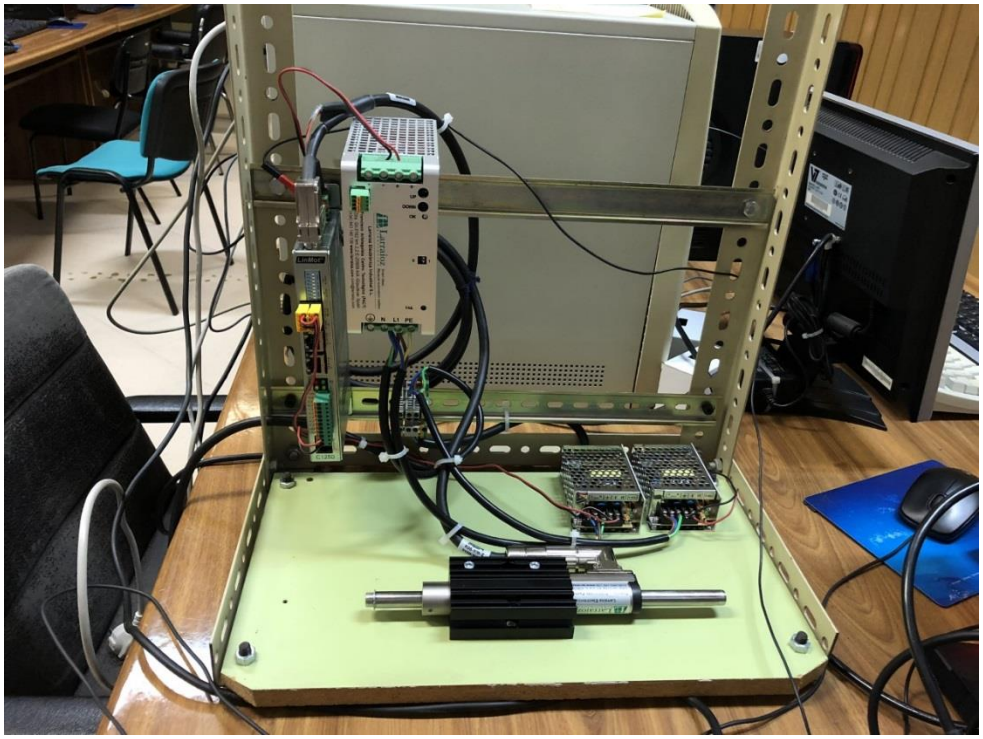

Figura 3.1: Hardware del equipo.

En la figura 3.1 tenemos la instalación de todos los componentes, el Hardware del equipo. En este Hardware podemos encontrar las siguientes partes:

- Motor lineal P01.
- Controlador C1250.
- Fuente de alimentación de 72 V.
- Convertidor de 240 VAC a 24 Vcc.

El controlador que se va a utilizar en este proyecto es C1250, el cual tiene protocolo de comunicación Ethercat, dicho protocolo se explica más adelante en el punto 4.

Figura 3.2: Controlador C1250.

En la figura 3.2 tenemos una imagen de este controlador, a continuación, se va a explicar cada conexión que hay que realizarle.

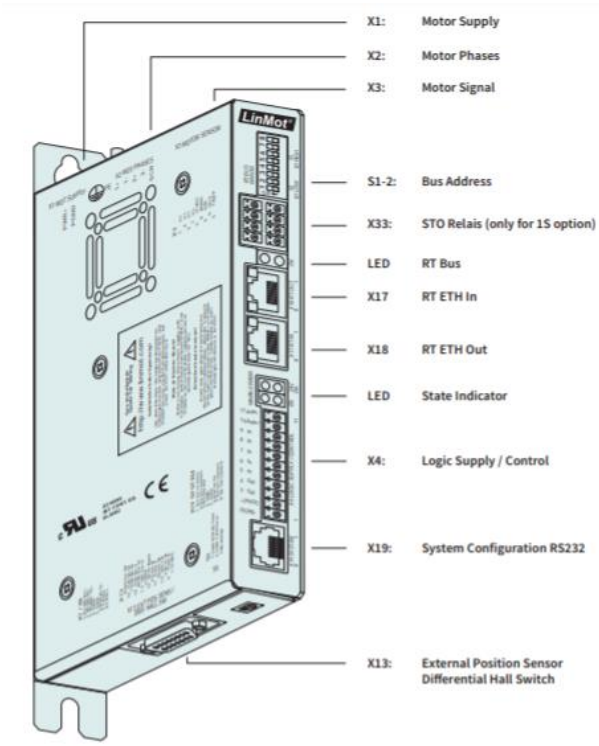

Figura 3.3: Controlador C1250.

En las figuras 3.3 y 3.4 podemos observar los distintos puertos del que se compone el controlador. X1 conectamos la tensión de la fuente de alimentación, fuente de 24V..72V, X2 es la alimentación al motor y X3 conectamos la señal del encoder que nos envía el motor, X33 se conecta dos cables uno es 24V y otro 0V. En los puertos X7 y X8 son para EtherCat, X17 la entrada y X18 la salida del bus. En X4 le tenemos que conectar 24V y 0V, tanto X4 como X33 viene de fuentes distintas no puede proceder de la misma fuente de alimentación, por eso en la instalación se dispone de dos convertidores, para tener dos fuentes de 24 Vcc. X19 es para conectarse mediante el protocolo de comunicación RS232, protocolo que se utilizará para conectarse a un ordenador con sistema operativo Windows y el software LinMot Talk, que se explicará más adelante.

Por ultimo respecto al controlador mencionar el puerto S1-2, es un switch en el que se introduce la dirección del bus del controlador.

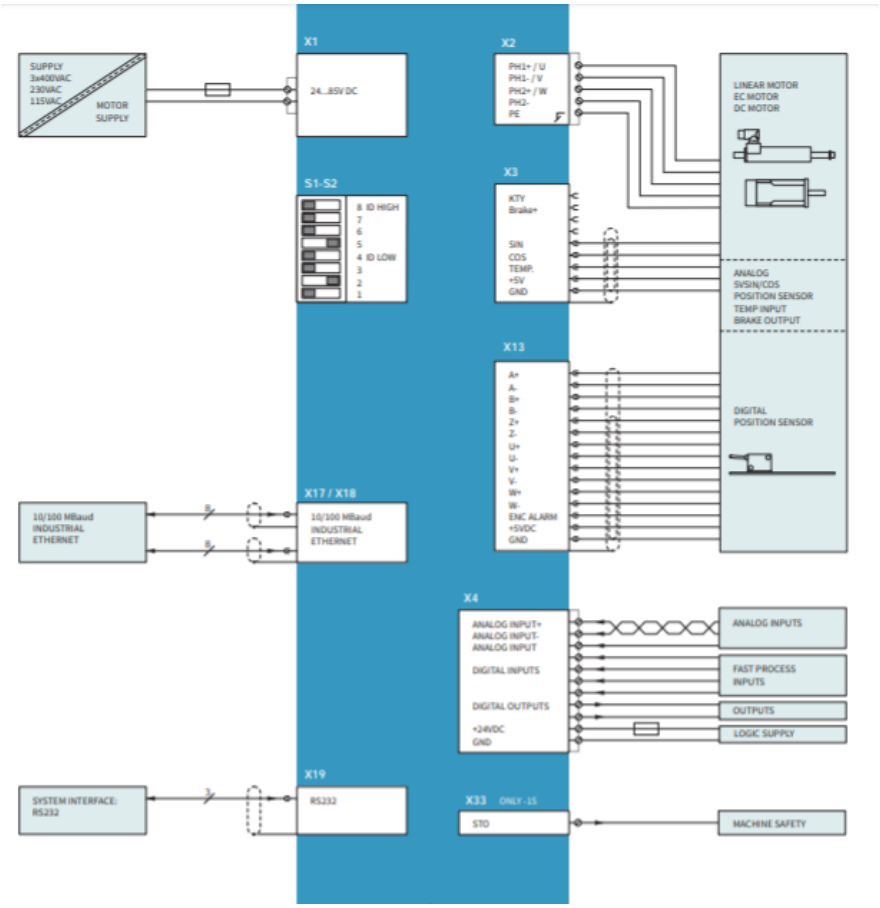

Figura 3.4: Conexiones del controlador C1250.

En la figura 3.4 se observa las conexiones que se pueden realizar a este controlador, en la instalación no se disponen de todas las conexiones, de la X4 no se dispone ni de entradas ni de salidas analógicas ni digitales, y X13 tampoco puesto que no se dispone de ningún sensor de posición.

3.1. Funcionamiento del software LinMot Talk

En este apartado se va a explicar cómo utilizar el software LinMot Tal, lo primero que hay que saber es para poder usar el software y que el controlador y motor funcionen tenemos que conectar el motor y el controlador como se ha mencionado en el punto anterior, y además en el ordenador hay que conectar el convertidor de RS232 a USB, figura 3.1.1, y el cable, RJ45, se conecta a ese convertidor y al puerto X19 del controlador.

La conexión entre el ordenador y el controlador se usa con el protocolo RS232, es una conexión punto a punto.

Figura 3.1.1: Convertidor RS232-USB.

En la figura 3.1.2 se observa la pantalla inicial del software LinMot Talk, lo primero que se tiene que hacer es o cargar una conexión o meter una conexión nueva, para ello desde la pantalla principal se pulsa File y después Login/Open Offline, como se puede observar en la figura 3.1.3. Después de pulsar Login/Open Offline se nos abre la ventana de la figura 3.1.4.

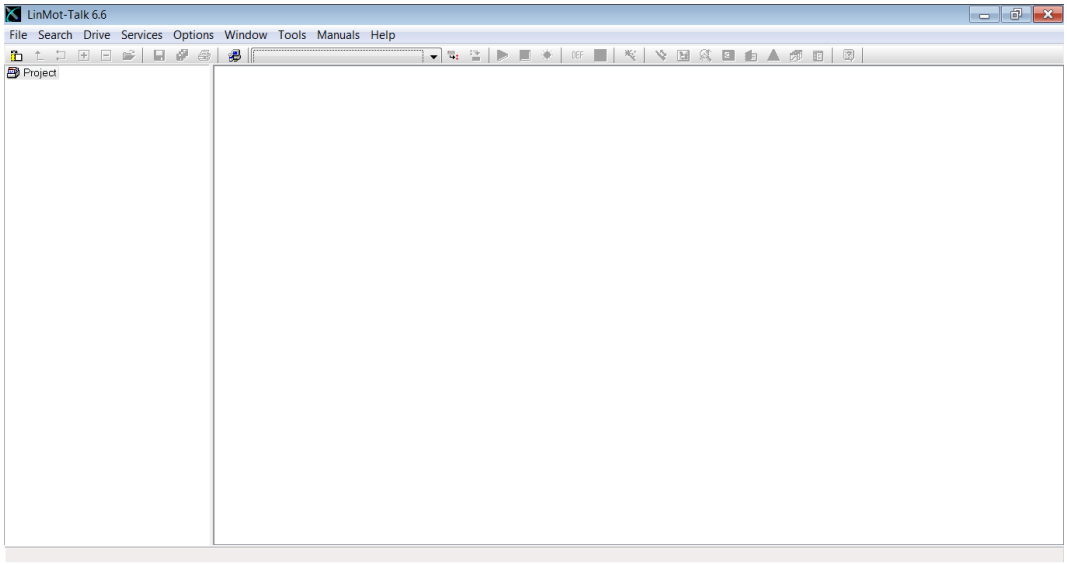

Figura 3.1.2: Pagina inicial software LinMot Talk.

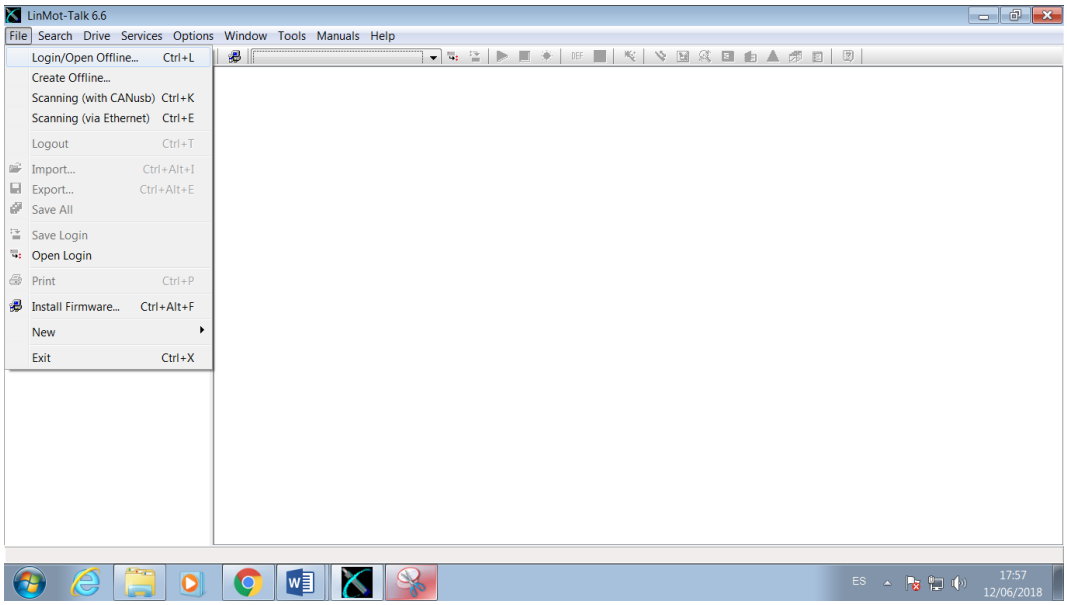

Figura 3.1.3: Creación de conexión.

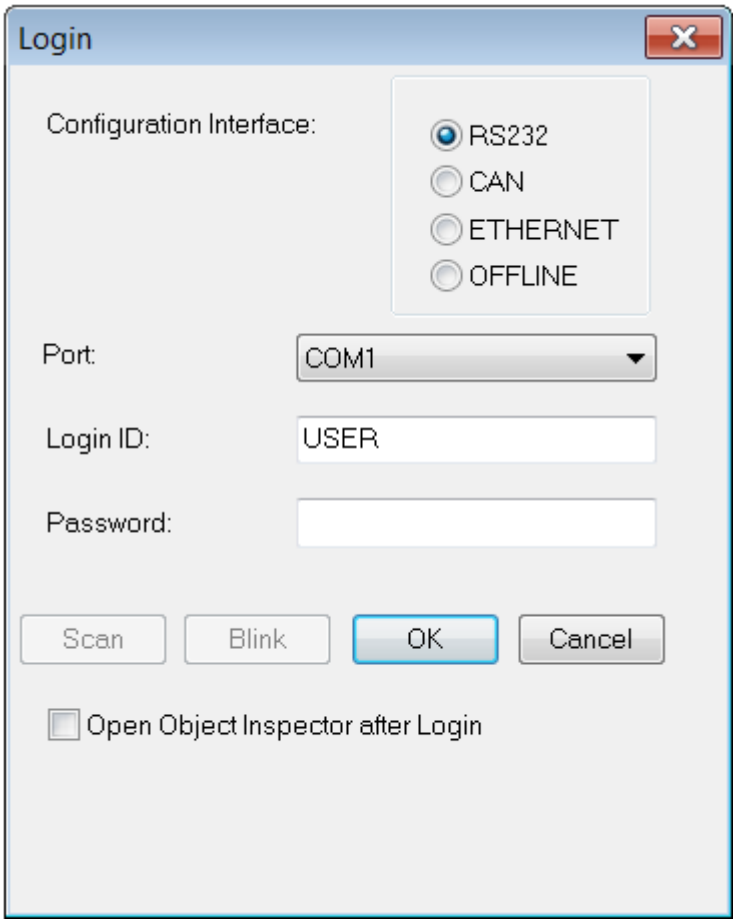

Figura 3.1.4: Creación de conexión.

En la figura 3.1.4 se observa la ventana que tenemos que configurar para la creación de una conexión, para ello se marca que la configuración interface es RS232, el puerto elegido tiene que ser donde conectemos el convertidor de RS232-USB, el Login ID puede ser el que uno quiera, se ha elegido USER en este caso, y la contraseña es optativa se puede poner una o no según se desee.

Una vez que hemos realizado esto pulsamos OK y ya tenemos configurada la conexión.

Una vez configurado la conexión nos aparece la pantalla de la figura 3.1.5, una vez que se ha creado la conexión se guarda, y la próxima vez que se quiera acceder no se crearía otra conexión si no se cargaría la conexión guardada.

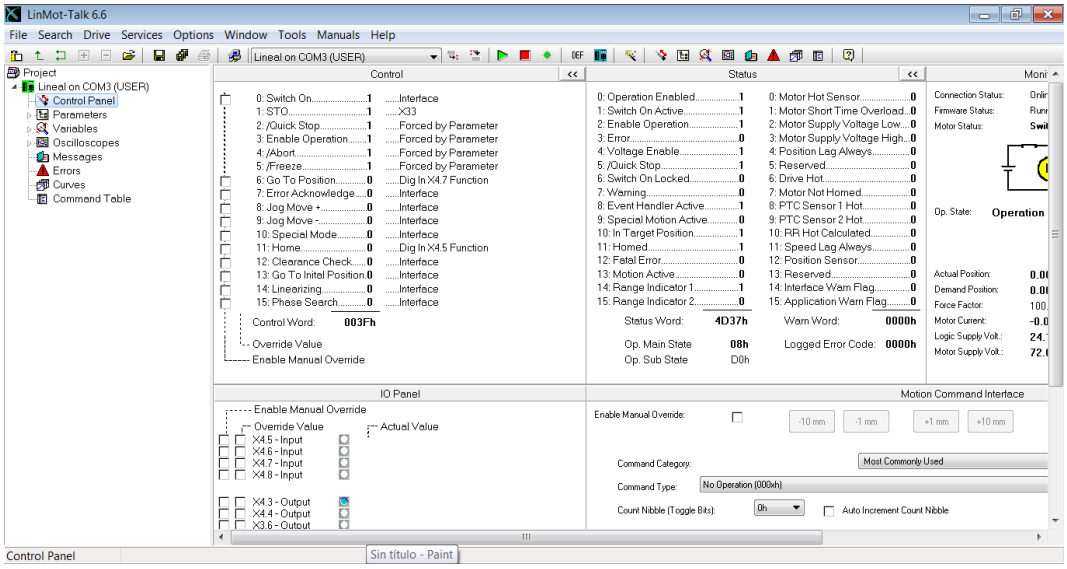

Figura 3.1.5: Creación de conexión.

Después de hacer esto se pulsa en la página principal pulsando la pestaña File se pulsa Instalar Firmware, como se indica en la figura 3.1.6.

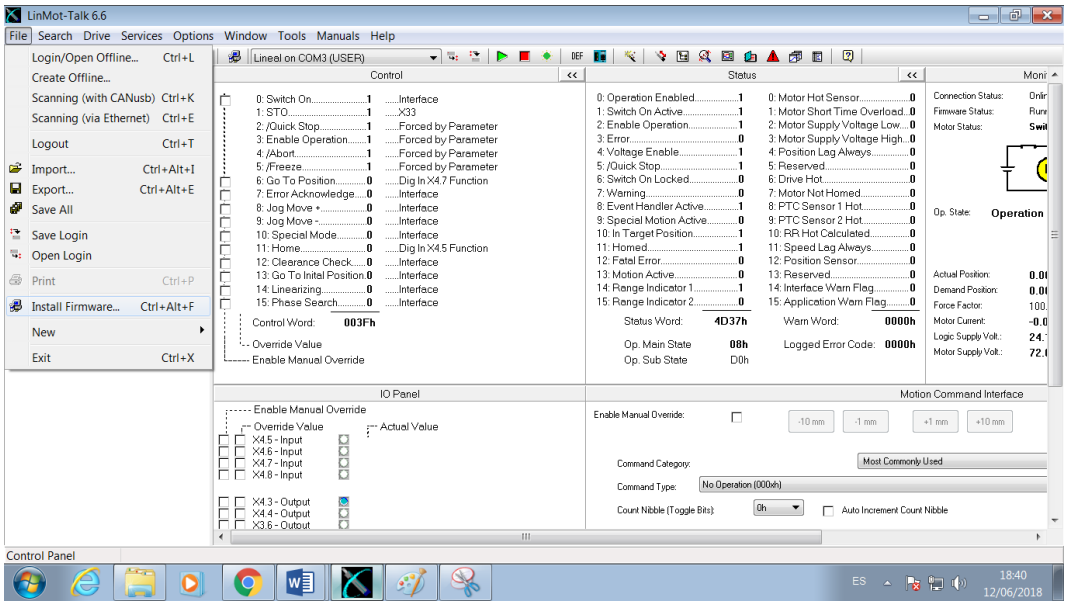

Figura 3.1.6: Instalación Firmware.

Una vez que pulsamos instalar Firmware tenemos que seleccionar donde está el archivo y pulsar instalar, como se muestra en la figura 3.1.7.

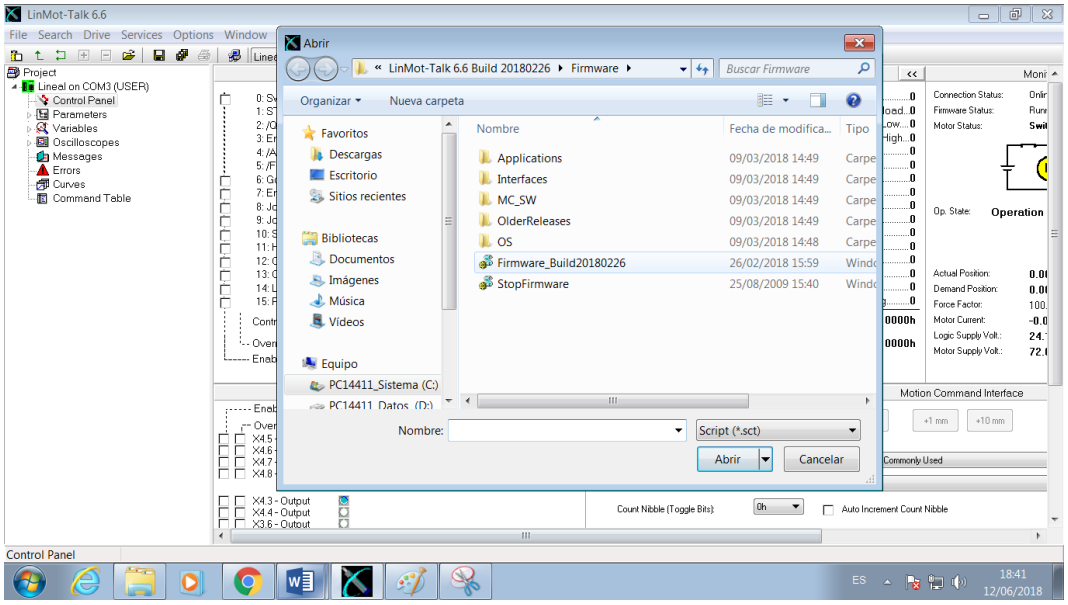

Figura 3.1.7: Instalación Firmware.

Después de dar abrir tenemos que indicar el puerto para instalar el firmware, se muestra en la figura 3.1.8, pulsamos ok y se nos instala el firmware.

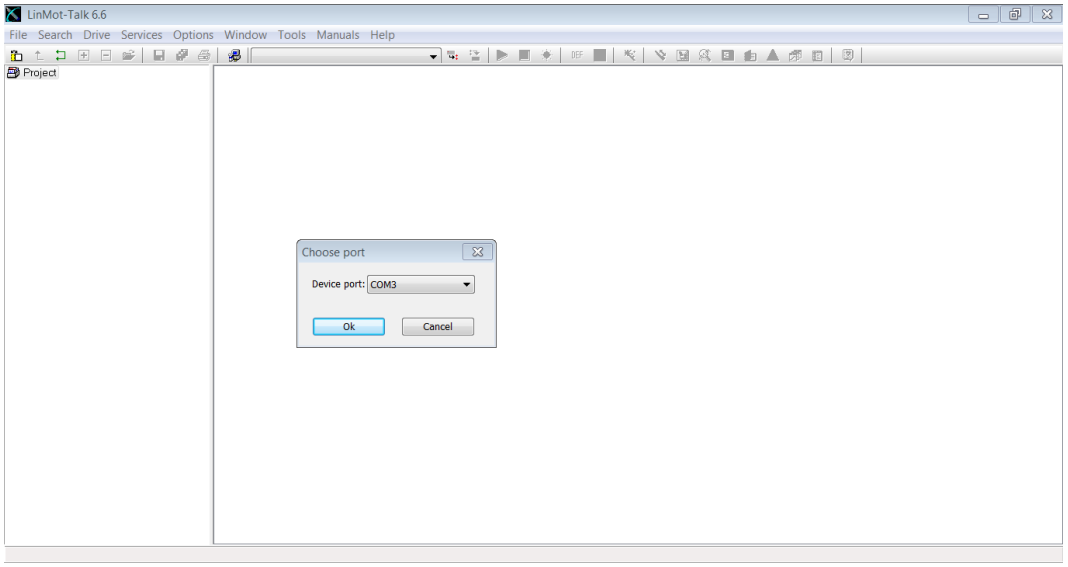

Figura 3.1.8: Instalación Firmware.

A continuación, en el siguiente punto se va a explicar cómo parametrizar el motor desde el software.

3.2. Parametrización del motor lineal

Después de explicar el funcionamiento del software de LinMot Talk, en este punto nos vamos a centrar en la parametrización del motor y su funcionamiento con este software.

3.2.1. Configuración Motor Wizard

En este apartado vamos a configurar todos los parámetros que tiene el motor Wizard.

Para acceder al menú del Motor Wizard, se va a la pestaña Drive y ahí se pulsa motor wizard, como se muestra en la imagen 3.2.1.1, o pulsando “ctrl+w”.

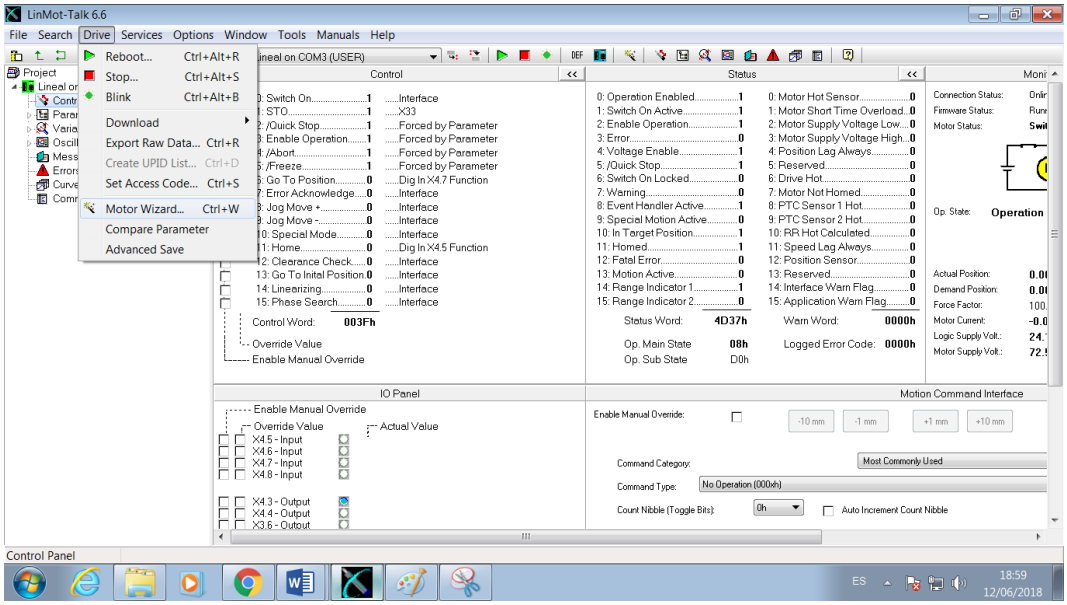

Figura 3.2.1.1: Configuración Motor Wizard.

Una vez que se pulsa el botón motor wizard se tienen que realizar nueve pasos de configuración.

- **Step 1**

  En la figura 3.2.1.2 se puede observar la pantalla de este paso, en el hay que elegir el estator y el slider que se tiene, en la imagen se observa cual es el estator y cuál es el slider del que se dispone para realizar este proyecto.

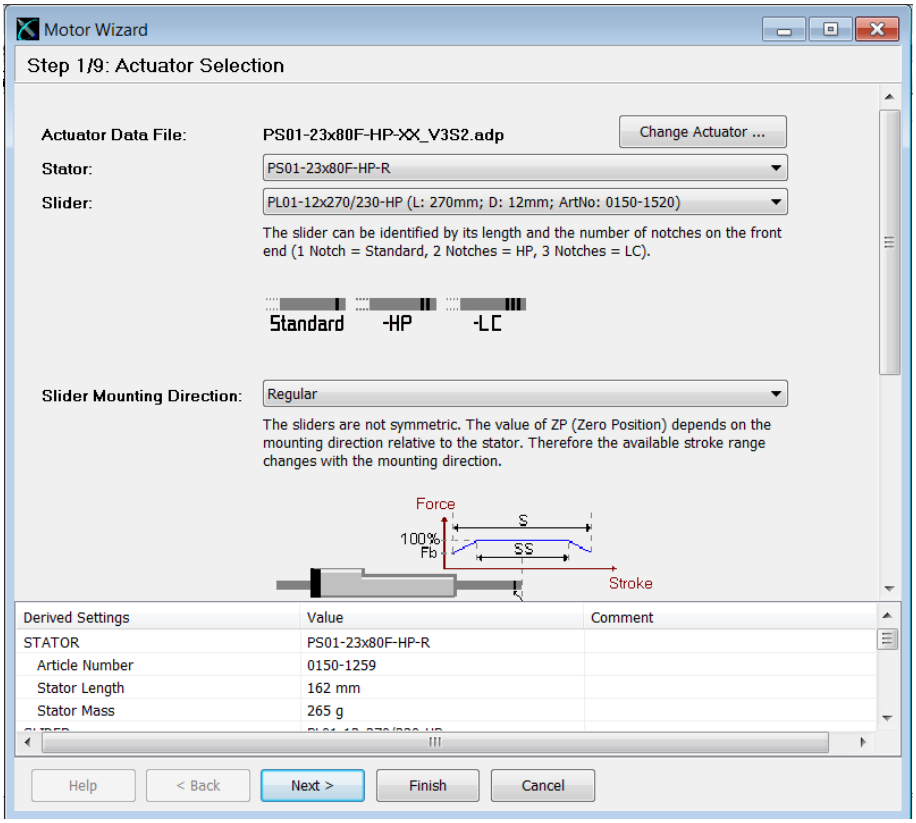

Figura 3.2.1.2: Configuración Motor Wizard, step 1.

Una vez que en la figura 3.2.1.2 hemos configurado todos los parámetros pulsamos next.

- **Step 2**

Este paso se puede observar en la imagen 3.2.1.3, aquí solo hay que configurar el movimiento del slider, en nuestro caso es lineal.

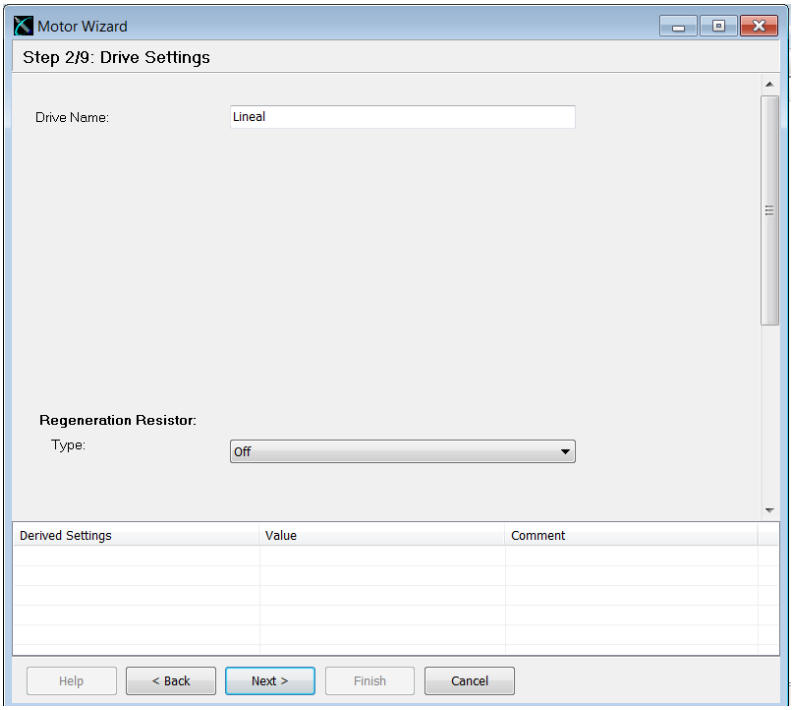

Figura 3.2.1.3: Configuración Motor Wizard, step 2.

- **Step 3**
  En este paso se configura la resistencia del cable, se puede observar este paso y los parámetros que se han puesto en la figura 3.2.1.4.

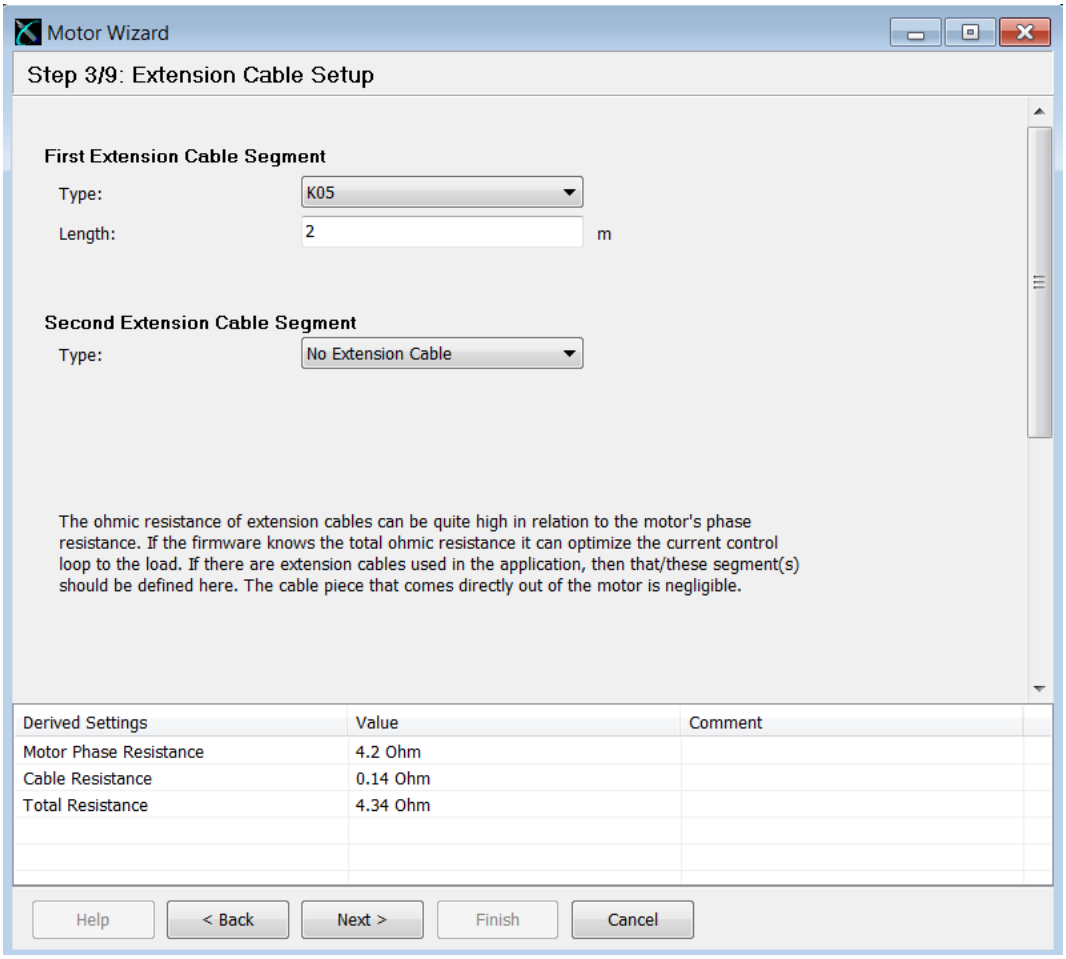

Figura 3.2.1.3: Configuración Motor Wizard, step 3.

- **Step 4**
  En este paso solo hay que configurar si tenemos algún sensor, en nuestro caso no tenemos ningún sensor conectado, este paso se observa en la figura 3.2.1.5.

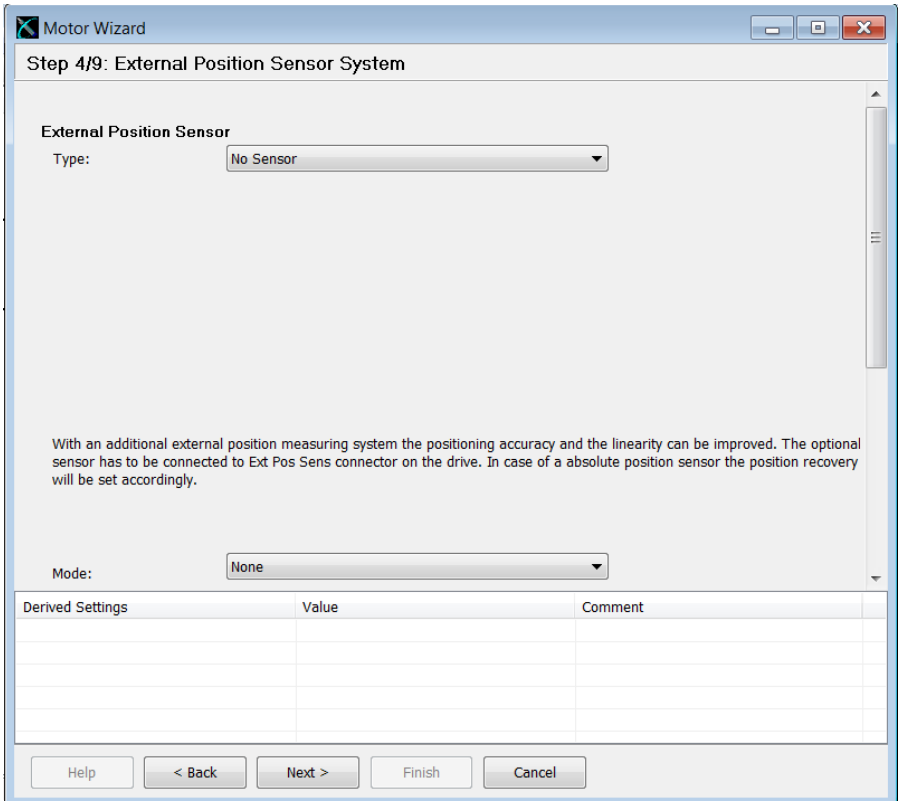

Figura 3.2.1.5: Configuración Motor Wizard, step 4.

- **Step 5**
  En este paso se configura en el eje en que se mueve el slider y si hay alguna fuerza externa, en la figura 3.2.1.6 se puede observar los parámetros configurados en este paso.

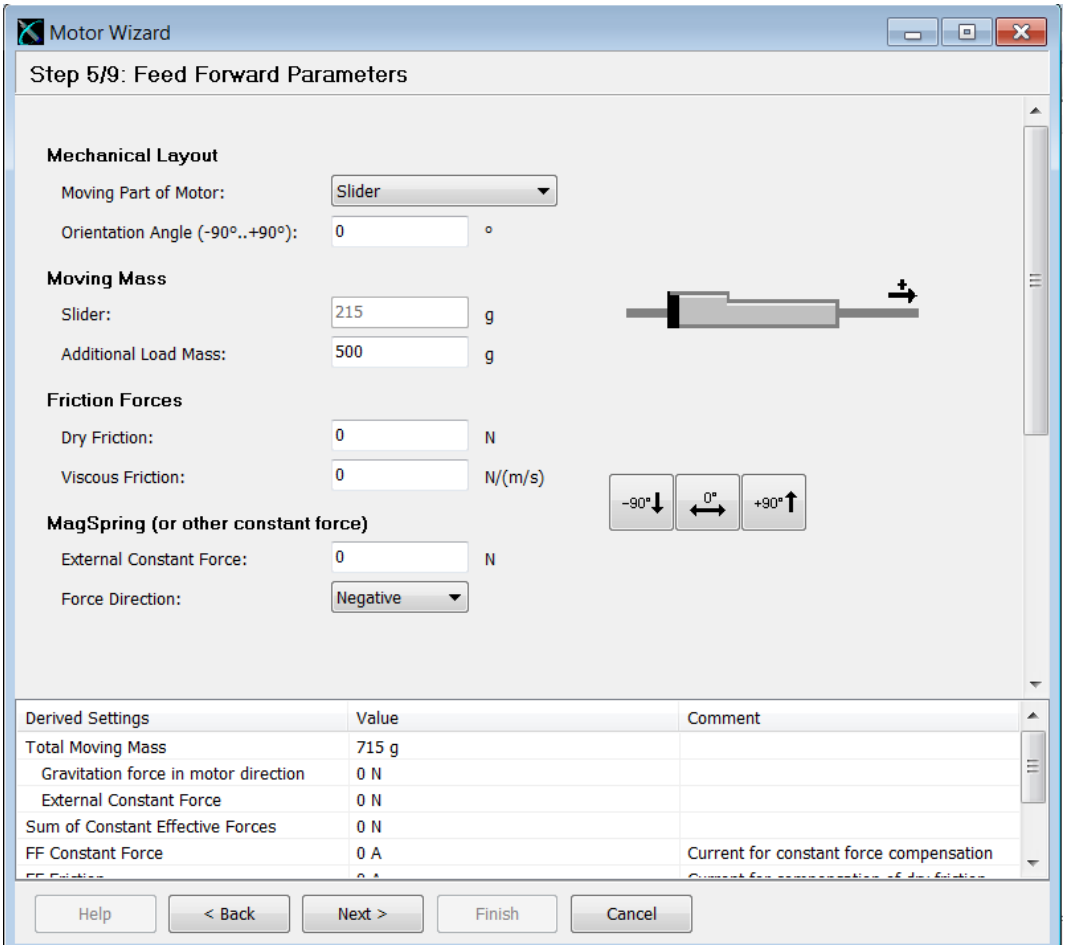

Figura 3.2.1.6: Configuración Motor Wizard, step 5.

- **Step 6**
  En este paso lo que se configura son los parámetros de PID de posición del controlador, los parámetros configurados en este paso se muestran en la figura 3.2.1.7.

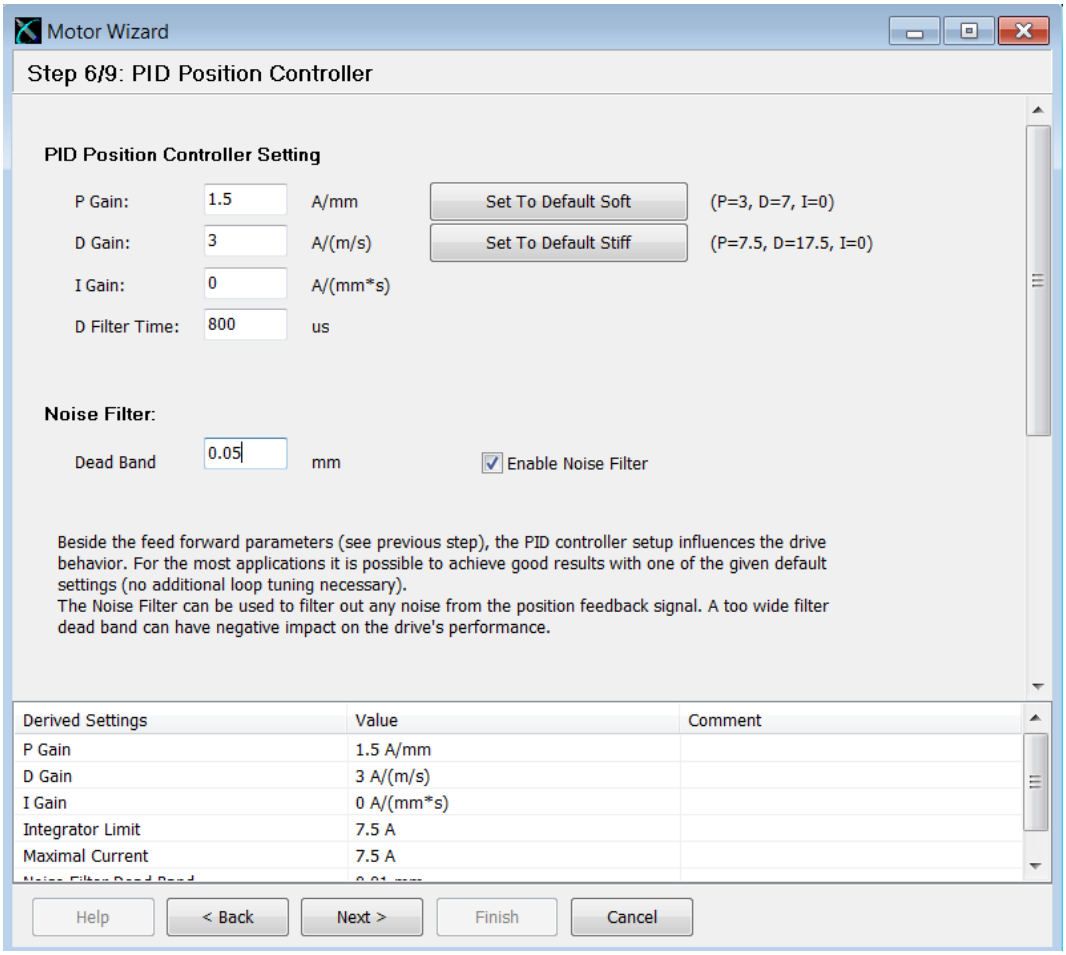

Figura 3.2.1.7: Configuración Motor Wizard, step 6.

- **Step 7**
  En este paso lo que se configura es la velocidad a la que el slider se mueve cuando se le manda ir a la posición Home, la configuración de este parámetro se puede observar en la figura 3.2.1.8.

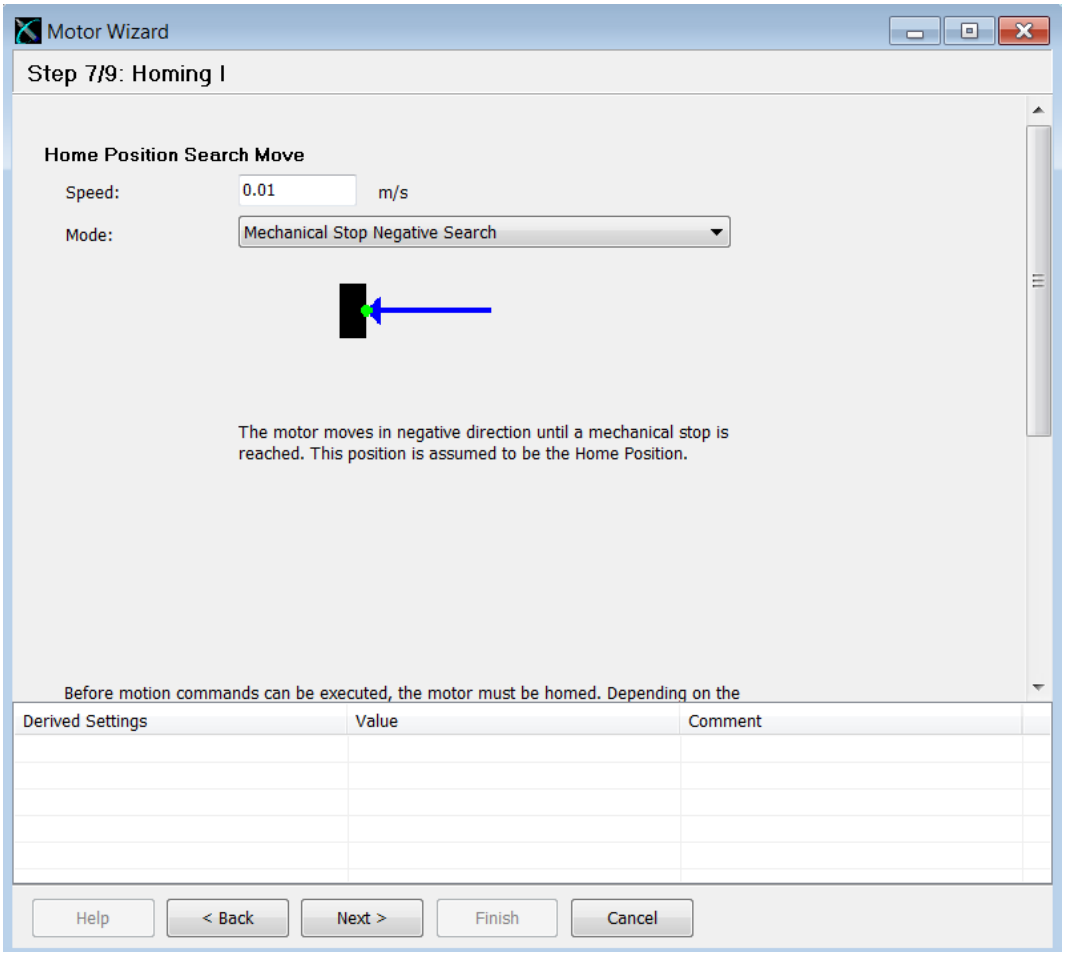

Figura 3.2.1.7: Configuración Motor Wizard, step 7.

- **Step 8**
  En este paso lo que se va a configurar es la distancia A y B, que son la distancia desde el fin del estator a fin del slider y a la posición Home, se puede observar esta configuración en la figura 3.2.1.8.

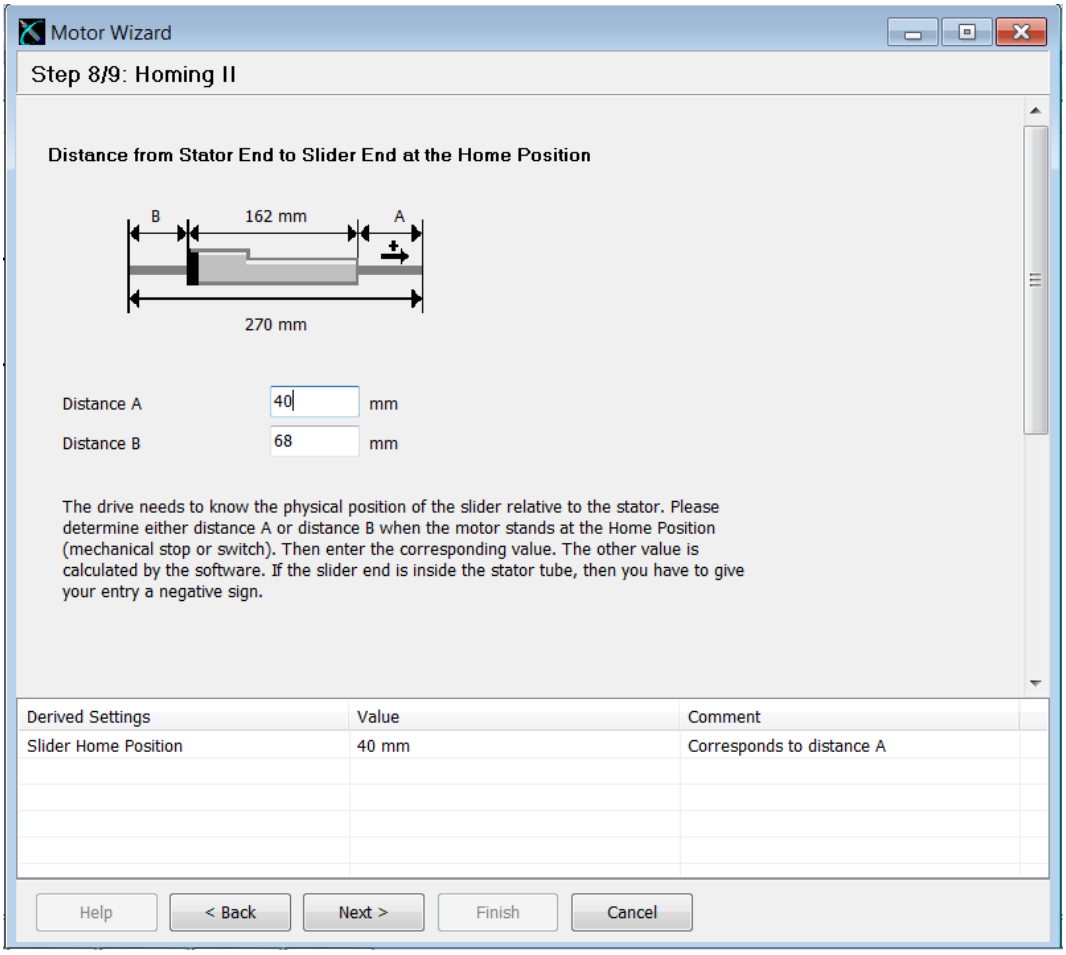

Figura 3.2.1.8: Configuración Motor Wizard, step 8.

- **Step 9**
  En este paso lo que se configura es cuál es la posición Home del estator, se puede observar esta configuración en la figura 3.2.1.9.

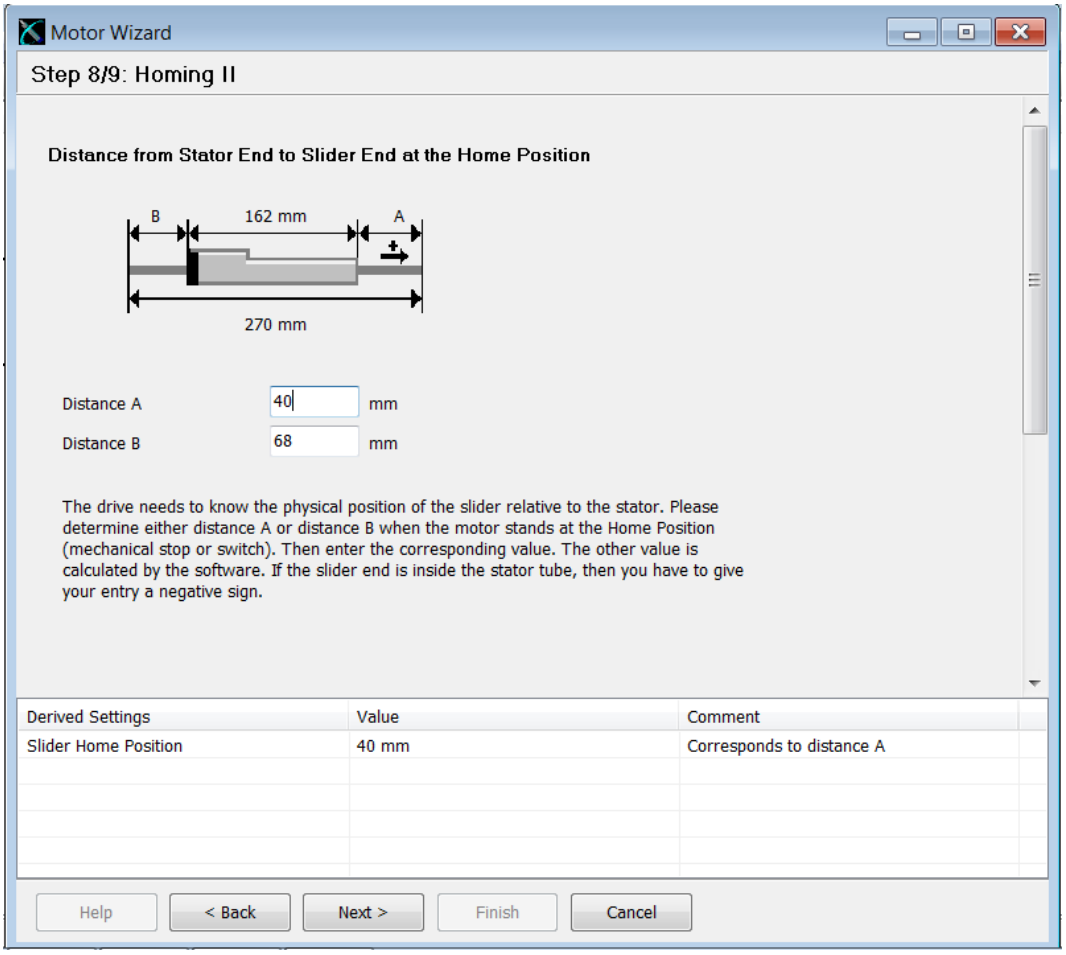

Figura 3.2.1.9: Configuración Motor Wizard, step 9.

3.2.2. Parámetros

En este apartado se va hablar sobre la configuración de la pestaña parámetros, en la figura 3.2.2.1 se muestra como ir a esta ventana, desde aquí principalmente se puede configurar que queremos que hagan las entradas analógicas y las entradas digitales que tiene el controlador, en nuestro caso no le vamos a poner ninguna entrada al controlador, ni analógica ni digital, pero se va a comentar como se podría hacer.

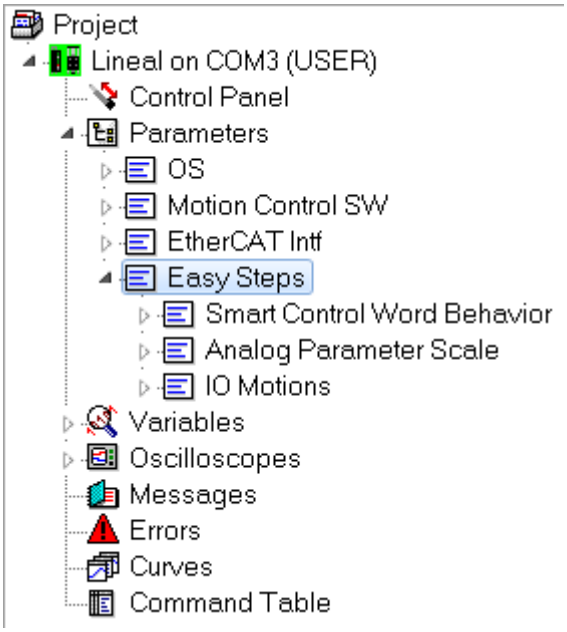

Figura 3.2.2.1: Configuración parámetros.

Como observamos en la figura 3.2.2.1 se observa que dentro del menú parámetros tenemos a su vez cuatro desplegables que son: OS, Motion Control SW, EtherCat y Easy Step.

De aquí mencionar que los parámetros que se definan en Eassy Steps no pueden ser contradictorios con los que hemos configurado en el apartado motor wizard, si no nos daría error. En el apartado de Easy step es donde se configura que hace el controlador ante una entrada analógica o una entrada digital, para ello pulsamos IO Motions para las entradas digitales o Analog Parameter Scale para las entradas analógicas.

En la figura 3.2.2.2 se observa las entradas digitales disponibles, como observamos solo tiene 4, de la entrada 4.5 a la entrada 4.8.

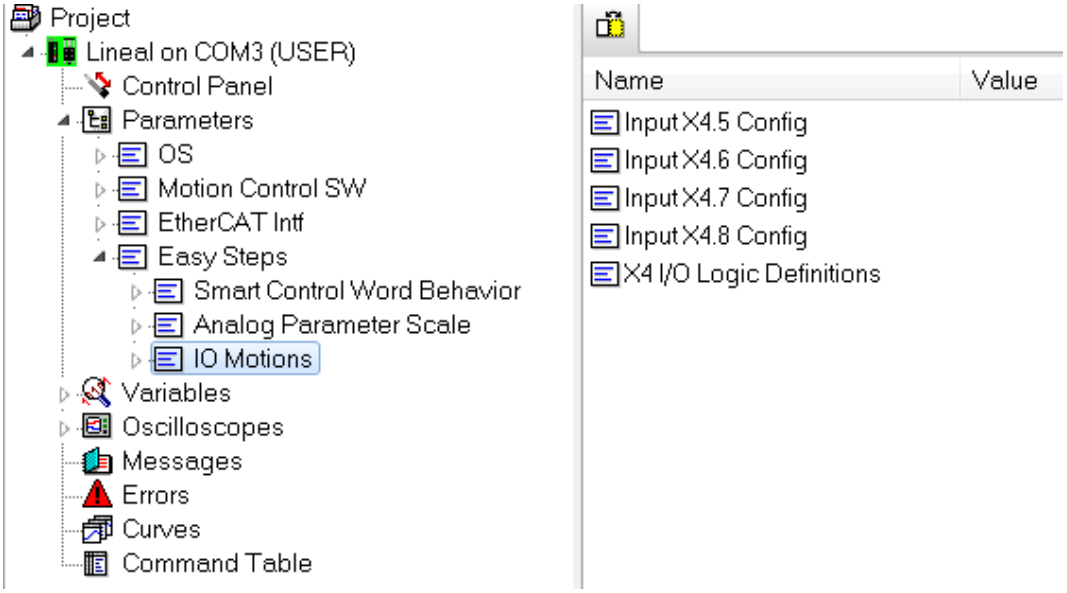

Figura 3.2.2.2: Configuración entradas digitales.

Las opciones que se pueden elegir para que el controlador interprete al llegar esa entrada son las que se muestran en la figura 3.2.2.3, para que se observe un ejemplo en la entrada 4.6 se ha elegido que ejecute las curvas 1 y 2, dichas curvas se explicarán más adelante.

none	Off	0000h	3700h	UInt16
Goto Abs Position	Off	0001h	3700h	UInt16
Increment Target Position	Off	0002h	3700h	UInt16
Increment Demand Position	Off	0003h	3700h	UInt16
Goto Abs Position From Act...	Off	0004h	3700h	UInt16
Increment Actual Position	Off	0005h	3700h	UInt16
Go To Analog Position	Off	0006h	3700h	UInt16
Inc Actual Position Between...	Off	0007h	3700h	UInt16
Start Curve From Actual Po...	Off	0008h	3700h	UInt16
Goto Abs Position With Max...	Off	0009h	3700h	UInt16
Eval Command Table Com...	On	000Ch	3700h	UInt16
VAI Stop	Off	000Dh	3700h	UInt16
VAI Infinite Motion Positive ...	Off	000Eh	3700h	UInt16
VAI Infinite Motion Negative ...	Off	000Fh	3700h	UInt16
Master Homing	Off	001Ah	3700h	UInt16

Figura 3.2.2.3: Configuración entradas digitales.

Al activar esa entrada el controlador haría eso mandando al motor su correspondiente palabra de trabajo para que realice eso.

3.2.3. Curvas

En este apartado se va a explicar cómo acceder a las curvas que tiene configuradas el controlador y como cambiarlas.

Para acceder al apartado de lar curvas se indica en las figuras 3.2.3.1 y 3.2.3.2, como podemos observar aquí disponemos de dos curvas, pero podríamos tener más configuradas, puedes crear una curva nueva, pinchando en la pestaña New Curve en la que lo que habría que hacer es configurar la curva, como se muestra en la figura 3.2.3.3. También aparte de crear una curva nueva se puede modificar una ya existente para ello tienes que pinchar encima de la curva que se desea modificar y pinchar sobre la flecha ^, con ello se desplaza la curva a la ventana de editar, se muestra en la figura 3.2.3.3, una vez que está en esta ventana se puede editar las propiedades o editar el valor de la curva, si editamos el valor de la curva lo que nos aparece es lo que se muestra en la figura 3.2.3.4, y con ello cambiaríamos los puntos que quisiéramos de esa curva. Otra forma de cambiar los valores de la curva es pulsando dos veces sobre la curva que se desea cambiar y después desplazar la curva como queramos, se muestra en la figura 3.2.3.5.

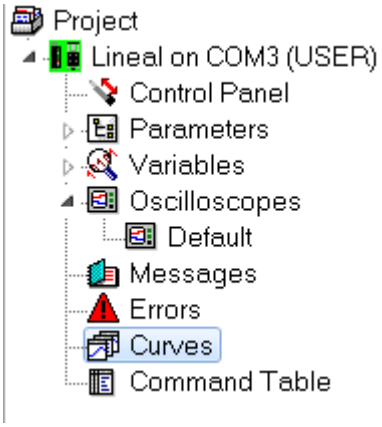

Figura 3.2.3.1: Curvas.

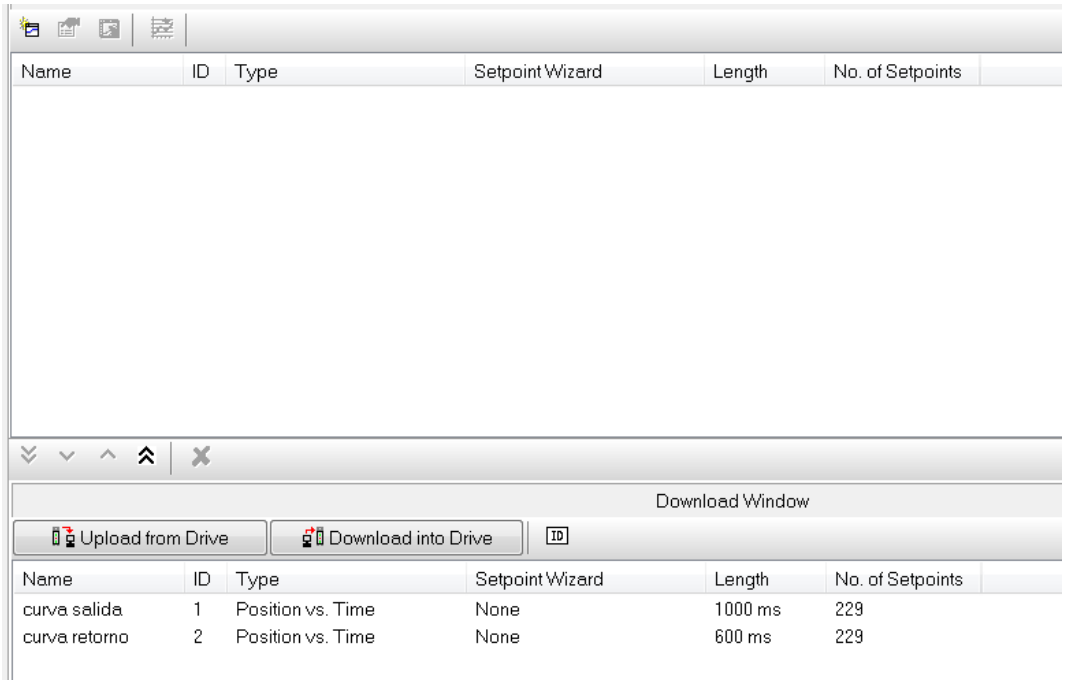

Figura 3.2.3.2: Curvas.

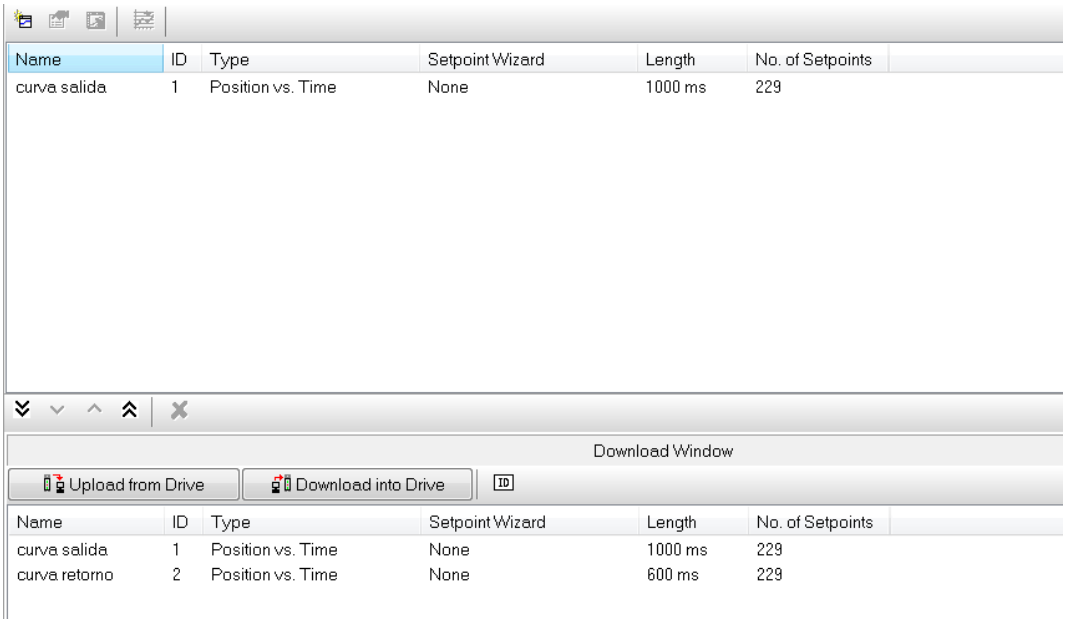

Figura 3.2.3.3: Modificación de Curvas.

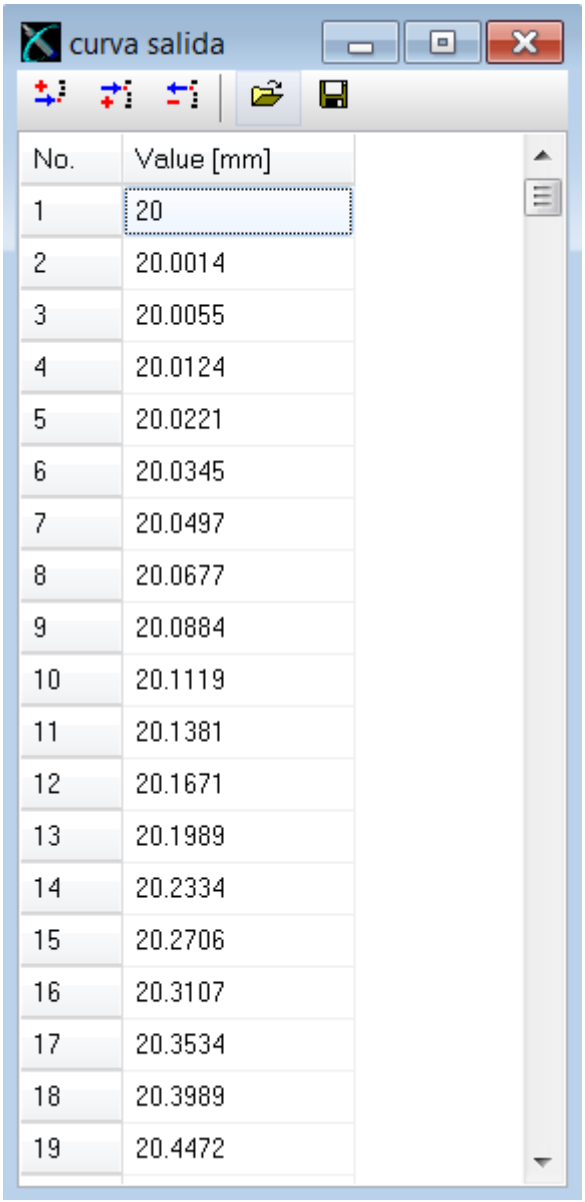

Figura 3.2.3.4: Editar valores curva.

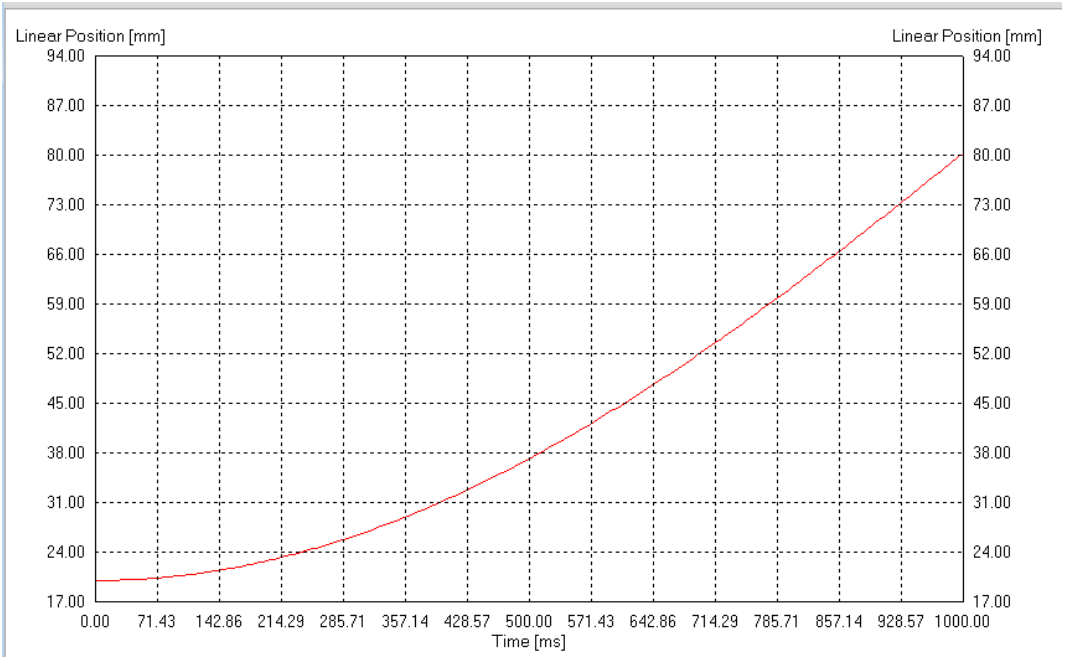

Figura 3.2.3.5: Editar valores curva.

3.3. Gestión del motor lineal mediante el software LinTalk

En este apartado se va hablar sobre las palabras de intercambio entre el ordenador y el controlador, y también como se gestiona el motor con la aplicación para Windows LinMot-Talks.

3.3.1. Palabras de intercambio

Una vez que hemos establecido conexión con el controlador, en la pantalla tenemos lo que se muestra en la figura 3.3.1.1, en ella podemos ver las palabras de intercambio entre el controlador y el ordenador, como podemos observar tenemos 3 palabras, la palabra de control, la palabra de estado y la palabra de alarma.

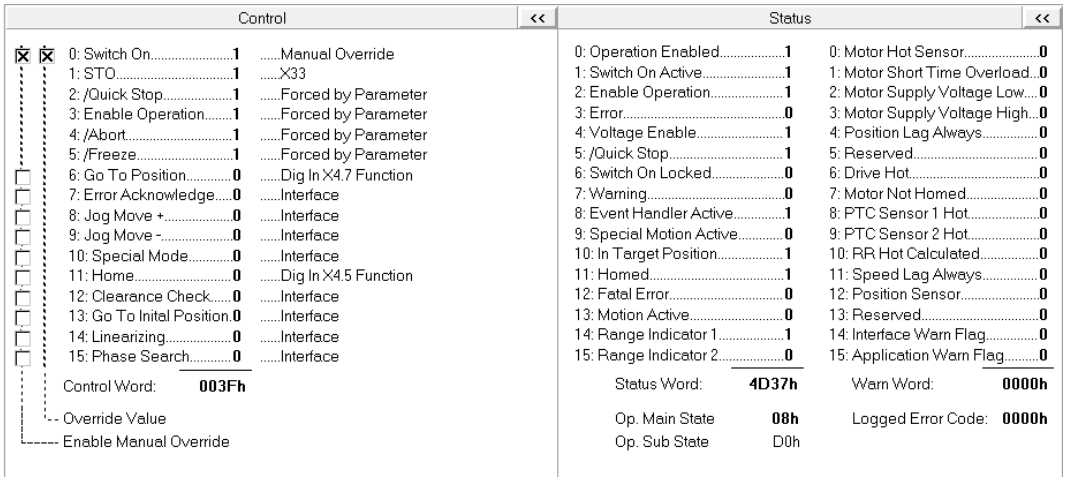

Figura 3.3.1.1: Palabras de intercambio.

La palabra que recibe el controlador es la palabra de control, esa palabra es la que se envía desde el ordenador, la palabra de estado y de alarma son las que envía el controlador al ordenador.

Cada palabra son 2 byte que a su vez son 16 bits, por lo cual cada palabra son 16 variables booleanas.

- **Palabra de control**

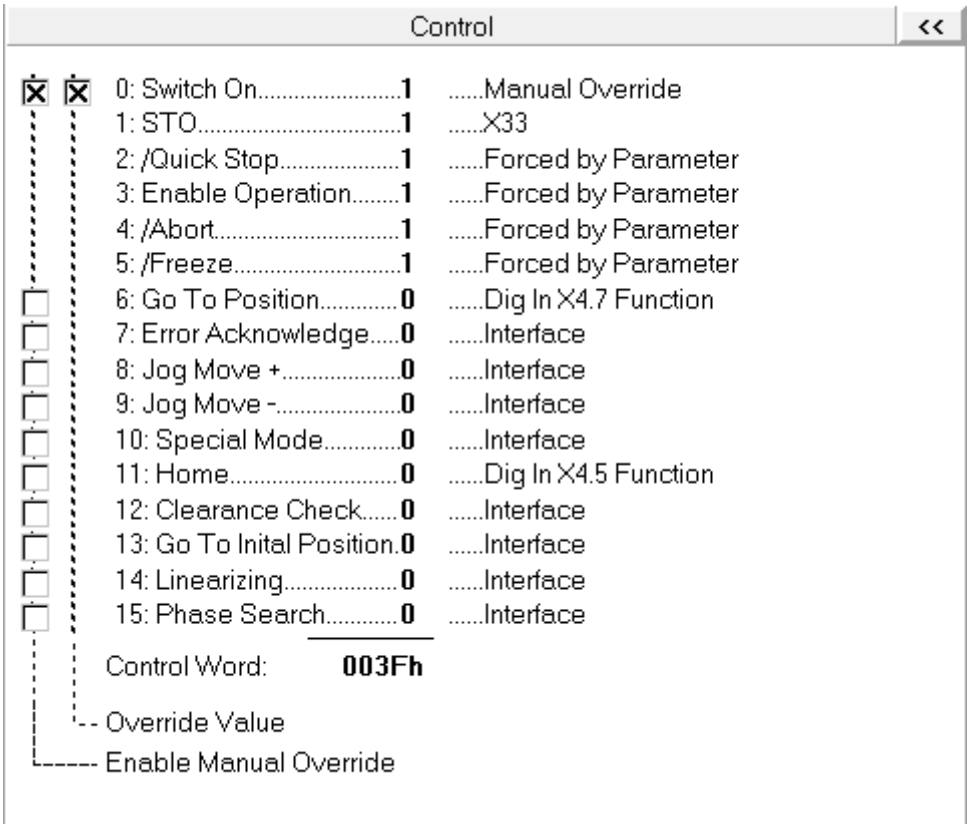

Figura 3.3.1.2: Palabra de control.

En la figura 3.3.1.2 se observa esta palabra desde el software LinMot Talks. Esta palabra se compone de 16 bit, y dependiendo de que bit se active cambia el valor de la palabra. Como se observa en la imagen 3.3.1.2 cada bit de esa palabra tiene un efecto en el controlador, por ejemplo al activar el bit 11 le decimos al controlador que es motor vaya a la posición de Home que hemos definido en el Motor Wizard, al activar el bit 6 le decimos que vaya a la posición definida en el motor Wizard, en este caso se definió esa posición como x=100mm, pero el primer bit que debemos activar y si no está activo el motor no va a funcionar es el bit 0, que lo que hace el controlador es dar tensión al motor, esas son los bit que más se van a utilizar para el control del motor. Otros bits que también pueden ser interesantes son los bits 8 y 9, en ellos se manda al controlador que el motor se mueva 0.1 mm, este dato se configura en el Motor Wizard, y se mueve en sentido positivo, si se activa el bit 8 o en sentido negativo si se activa el bit 9. El bit 13 es también interesante lo que se le dice al controlador es que envié al motor desde la posición que se encuentra a la posición inicial, esta posición se configura también como ya se explicó en el apartado 3.2.1 en el Motor Wizard.

- **Palabra de estado**

En la figura 3.3.1.3, en la parte de la izquierda, se observa esta palabra desde el software LinMot Talks. Esta palabra contiene 16 bits, y dependiendo de que bit active cambia el valor de la palabra, con esta palabra se puede saber el estado del motor, si el motor ha llegado a la posición demandada, si el motor está con tensión, si el motor está funcionando y se está moviendo a la posición demandada, eso es lo que más nos va a interesar del estado del motor

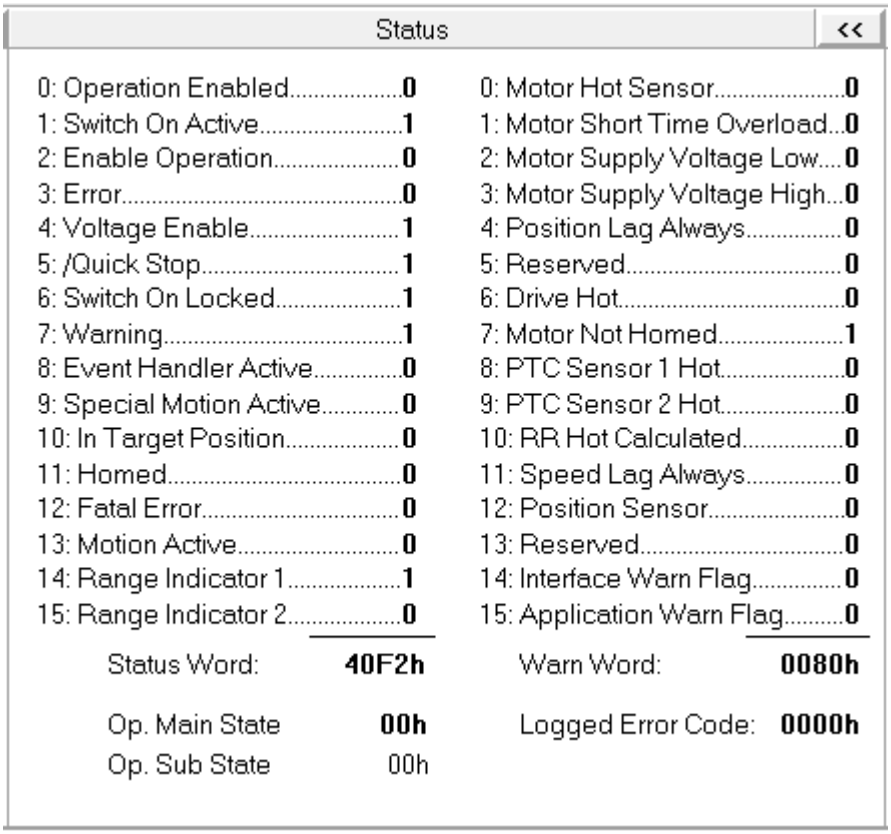

Figura 3.3.1.3: Palabras de estado y de alarma.

- **Palabra de alarma**

En la figura 3.3.1.3, en la parte de la derecha, se observa esta palabra desde el software LinMot Talks. Esta palabra contiene 16 bits, y dependiendo de que bit active cambia el valor de la palabra, esta

palabra si no hay ninguna alarma o error grave vale cero, si tenemos alguna alarma el valor ya no es cero, pero el motor se va a mover, en cambio sí tenemos algún error grave el motor no se va a mover, en este caso lo que se requiere es hacer un reboot al controlador desde el ordenador y con ello se resetea el error y ya podemos volver a mandar una orden al controlador para que la ejecute el motor.

3.3.2. Control del motor

Una vez que se ha explicado como configurar el motor, el manejo del software LinMot Talks y el intercambio de palabras entre el ordenador y el controlador se está ya en disposición de controlar al motor y ver cómo responde las ordenes que se envían desde el software de LinMot Talks.

En este apartado lo que se va a mostrar son unos ejemplos para ver cómo funciona el motor desde esta aplicación, para ello se le va a ir activando diversos bits de la palabra de control, se va a observar la palabra de estado y la de alarma.

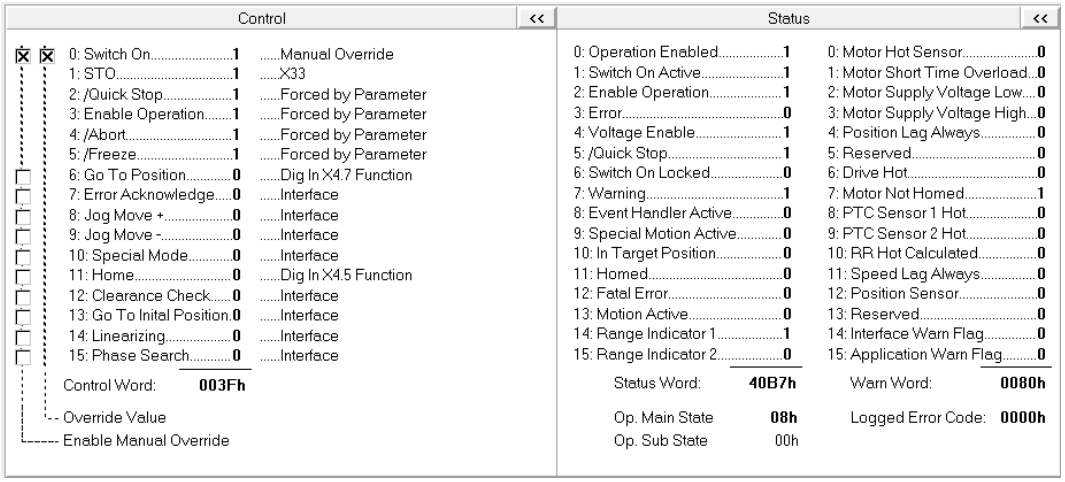

Figura 3.3.2.1: Ejemplo de control del motor.

En la figura 3.3.2.1 tenemos el motor con tensión, para ello se activa el bit 0 de la palabra de control, y el controlador nos devuelve la palabra de estado y de alarma que se ve en dicha figura. También se observa la palabra de control que se le envía al controlador.

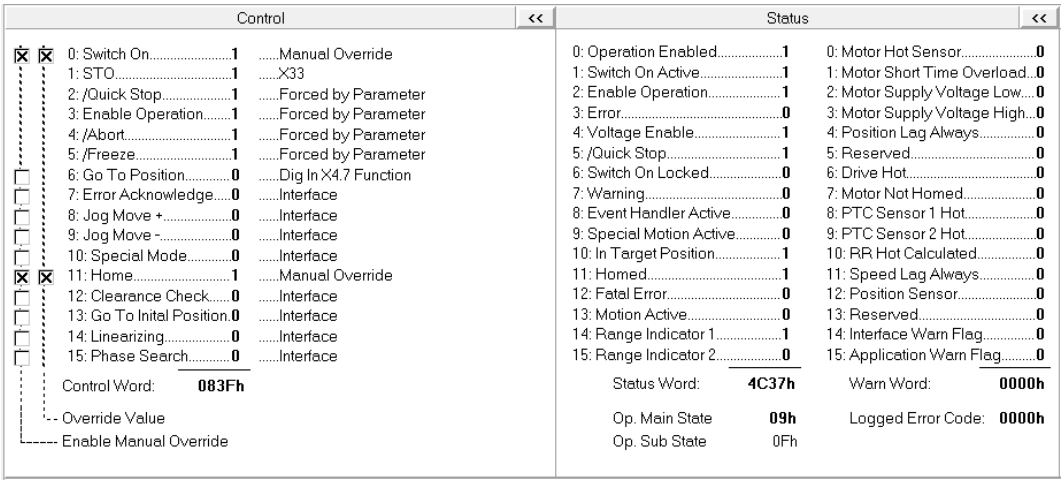

Figura 3.3.2.2: Ejemplo de control del motor.

En la figura 3.3.2.2 tenemos el motor en la posición HOME, para ello se activa el bit 0 de la palabra de control, para que el motor se pueda mover, y el bit 11 para que el motor vaya a la posición HOME, el controlador nos devuelve la palabra de control y de estado que se observa en dicha imagen. También se observa la palabra de control que se le envía al controlador.

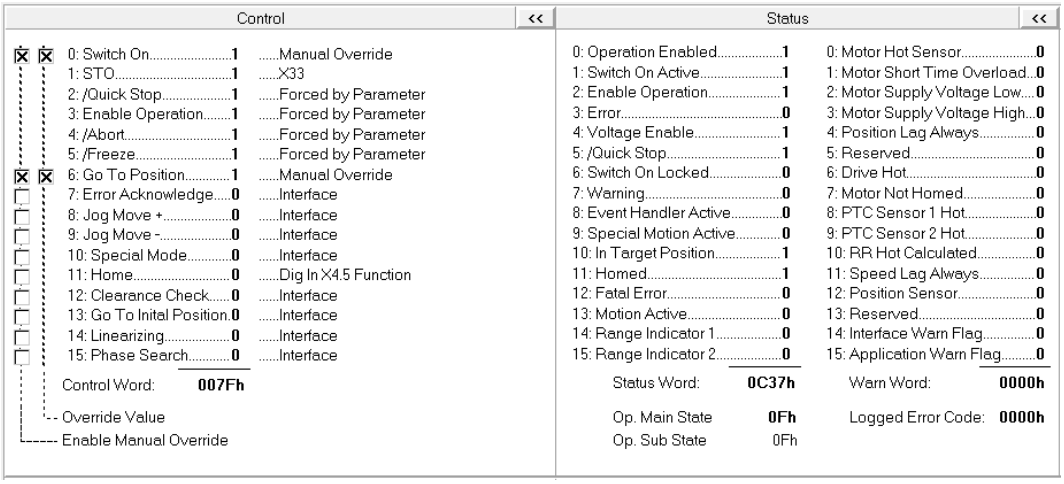

Figura 3.3.2.3: Ejemplo de control del motor.

En la figura 3.3.2.3 tenemos el motor en la posición definida anteriormente, 90mm, para ello se activa el bit 0 de la palabra de control, para que el motor se pueda mover, y el bit 6 para que el motor vaya a la posición definida, el controlador nos devuelve la palabra de control y de estado que se observa en dicha imagen. También se observa la palabra de control que se le envía al controlador.

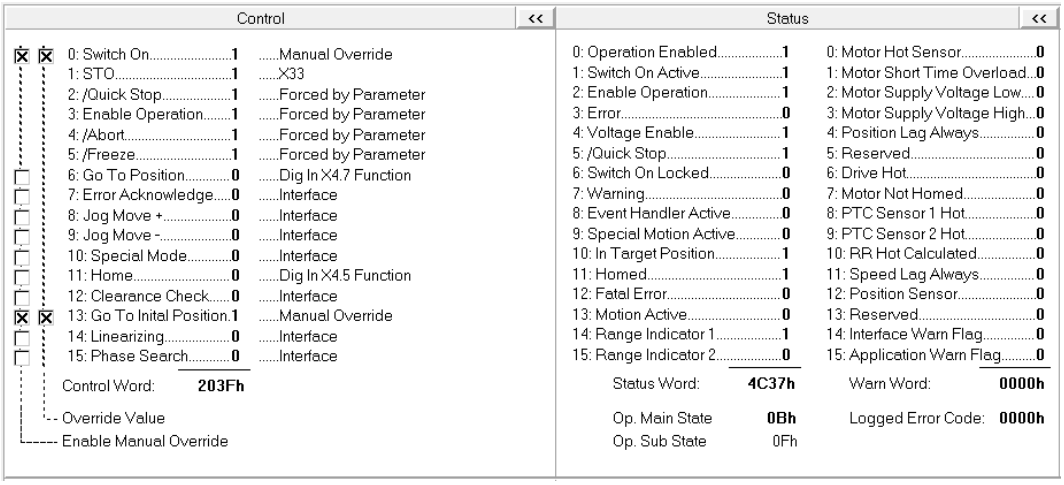

Figura 3.3.2.4: Ejemplo de control del motor.

En la figura 3.3.2.4 tenemos el motor en la posición inicial para ello se activa el bit 0 de la palabra de control, para que el motor se pueda mover, y el bit 13 para que el motor vaya a la posición inicial, el controlador nos devuelve la palabra de control y de estado que se observa en dicha imagen. También se observa la palabra de control que se le envía al controlador.

4. Buses de campo

En este apartado vamos analizar los distintos buses de campo que permite el controlador de la marca de LinMot, explicando en que consiste cada bus de campo y su protocolo de comunicación, para centrarnos en última instancia el bus de campo elegido para este trabajo.

4.1. Profibus

En este apartado se va a explicar el bus de campo Profibus.

Profibus surgió como una propuesta alemana liderada por Siemens y Bosch, dos fabricantes alemanes, se convirtió en estándar nacional definida en la norma DIN 19245.

La arquitectura Profibus está basada en el modelo de referencia OSI eliminando aquellos niveles cuyas funciones no se consideran necesarias, y consiguen así una arquitectura más ágil y simple.

Profibus usa un método de acceso al medio hibrido, que combina paso de testigo entre estaciones principales con acceso sondeo-selección en cada grupo de esclavas que dependen de una principal.

Profibus admite hasta 127 dispositivos, de los cuales hasta 32 se pueden configurar como activos.

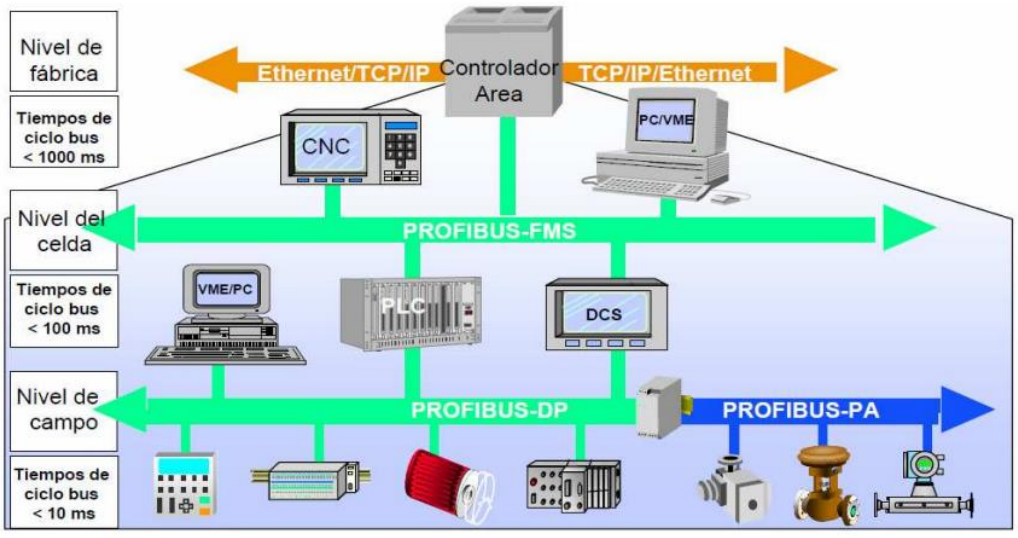

Figura 4.1.1: Perfiles de Profibus.

En la figura 4.1.1 se puede observar los distintos perfiles que se han definido para Profibus, como observamos son 3 y se describen a continuación:

- **FMS:** Orientado al intercambio de información estructurada entre el PLC y controladores de área, etc.
- **DP:** Orientado a la conexión con dispositivos de campo, esta optimizado para la transmisión cíclica de datos a alta velocidad.
- **PA:** Está orientada a su utilización en control de procesos.

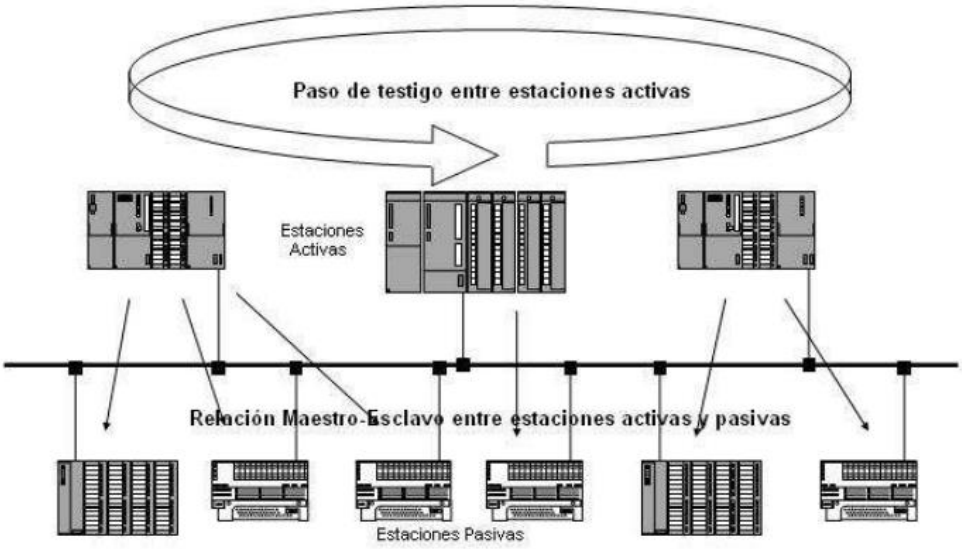

Figura 4.1.2: Métodos de acceso al medio para Profibus.

En la figura 4.1.2 se observa los distintos métodos de acceso al medio que podemos encontrar para este bus de campo, depende de si es maestro o esclavo encontramos dos tipos distintos de acceso al medio.

Para estaciones activas, maestros, el acceso al medio es mediante paso de testigo. Todos los maestros constituyen en un orden definido un anillo con paso de testigo. Cada usuario activo conoce los otros usuarios activos, así como su orden para recibir el testigo. Cada maestro va pasando el anillo al maestro que le corresponde. Cuando el maestro recibe el testigo es autorizado para actuar de maestro durante un tiempo limitado que se denomina tiempo de retención de testigo.

Los usuarios pasivos, o esclavos, nunca reciben este testigo, su método de acceso al medio es mediante Maestro-Esclavo, cuando su maestro recibe el testigo pasa a ser maestro y les consulta para que envíen la información correspondiente.

En este apartado, hemos empezado por una estrategia de control muy simple para observar el comportamiento del sistema.

4.2. DeviceNet

En este apartado lo que se explicará es el bus de campo DeviceNet, bus de campo que se puede utilizar con el controlador de la marca LinMot.

DeviceNet es una implementación del protocolo Common Industrial Protocol (CIP) para redes de comunicación industriales. Fue desarrollado por el fabricante Allan-Bradley, fabricante estadounidense.

DeviceNet es una de las tres tecnologías de redes abiertas y estandarizadas, cuya capa de aplicación usa el CIP (Common Application Layer – Capa de Aplicación Común). Al lado de ControlNet y EtherNet/IP, posee una estructura común de objetos. Es decir, es independiente del medio físico y de la capa de enlace de datos. Esta capa es una capa de aplicación estándar, integrada a interfaces de hardware y software abiertas, constituye una plataforma de conexión universal entre componentes en un sistema de automatización, desde la fábrica hasta el nivel de internet. La Figura 4.2.1 muestra una arquitectura del CIP dentro del modelo OSI, como se puede observar en esa imagen la capa de enlace de DeviceNet requiero el protocolo de comunicación CAN que es el que se explicara.

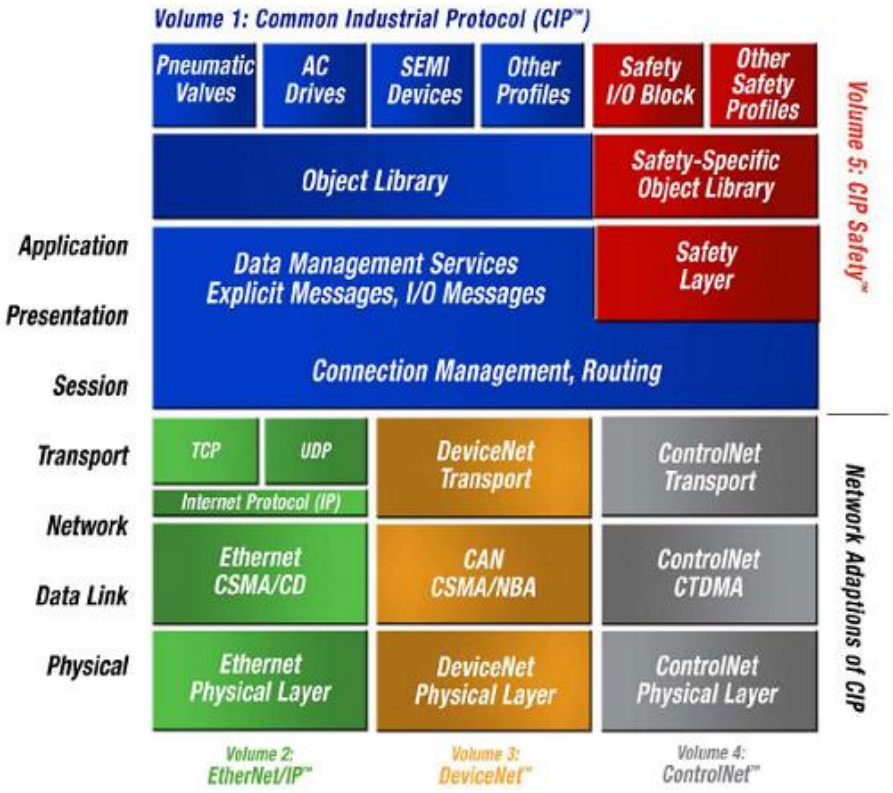

Figura 4.2.1: Arquitectura CIP.

Soporta comunicaciones de casi todos los tipos, ya sea Maestro-Esclavo, Punto a Punto, Multicast, Broadcast, etc.

El medio de transmisión para utilizar este bus de campo generalmente es a través de un cable hibrido para la transmisión de datos y para la alimentación de corriente. El cable está compuesto por dos pares trenzados y apantallados.

En este bus de campo se permiten hasta 64 estaciones, incluido el maestro.

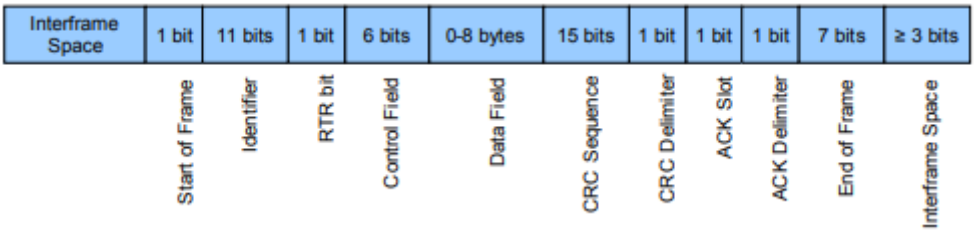

Figura 4.2.2: Frame de datos DeviceNet.

En la figura 4.2.2 se observa la trama para el mensaje que se envía por este bus de campo, para el identificador tenemos 11 bits, para indicar que empezamos la trama tenemos 1 bit, para los datos que queremos enviar tenemos hasta 8 bytes, y para indicar el fin de la trama tenemos 7 bits.

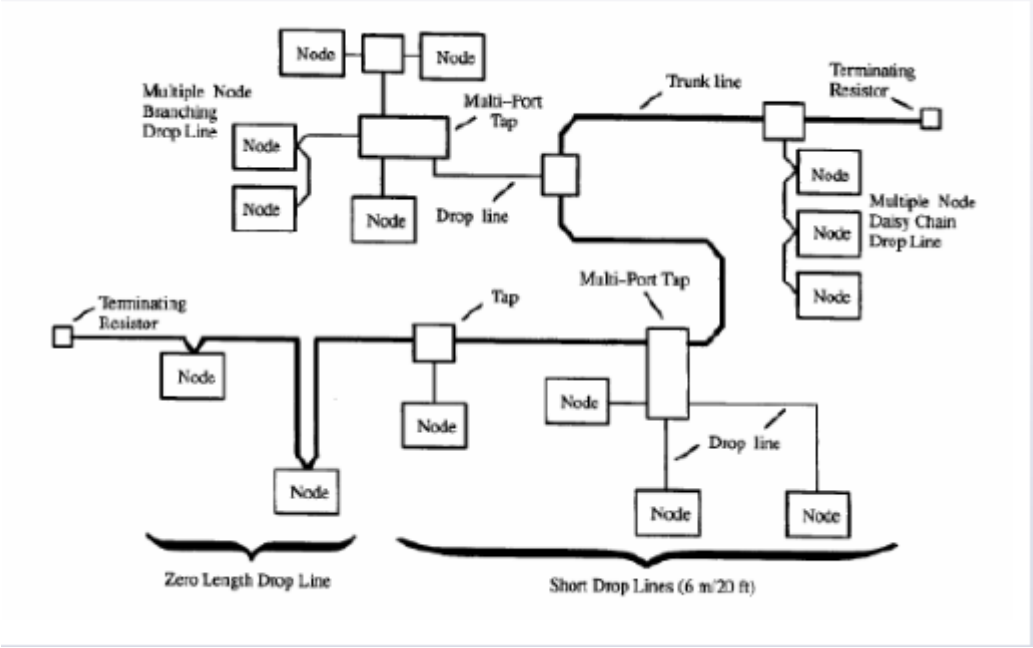

Figura 4.2.3: Topología de la red DeviceNet.

En la figura 4.2.3 se observa la topología de esta red como se observa es por derivaciones.

El método de acceso al medio que utiliza es CSMA/NBA que es mediante arbitraje de bus.

4.3. Ethernet\IP

Ethernet/IP es un protocolo de red en niveles para aplicaciones de automatización industrial. Está Basado en los protocolos estándar TCP/IP, el hardware y software Ethernet para la automatización de procesos de fabricación que incluye control, seguridad, sincronización, movimiento, configuración e información. Ethernet/IP clasifica los nodos de acuerdo a los tipos de dispositivos preestablecidos, con sus actuaciones específicas. El protocolo de red Ethernet/IP está basado en el Protocolo de Control e Información (Control and Information Protocol - CIP) utilizado en DeviceNet y ControlNet. Basados en esos protocolos, Ethernet/IP ofrece un sistema integrado completo desde la planta industrial hasta la red central de la empresa.

Ethernet/IP es un protocolo de red en niveles, apropiado al ambiente industrial. Es el producto acabado de cuatro organizaciones que aunaron esfuerzos en su desarrollo y divulgación para aplicaciones de automatización industrial: La Open DeviceNet Vendor Association (ODVA), la Industrial Open Ethernet Association (IOANA), la Control Net International (CI) y la Industrial Ethernet Association (IEA).

Ethernet/IP utiliza todos los protocolos del Ethernet tradicional, incluso el Protocolo de Control de Transmisión (TCP), el Protocolo Internet (IP) y las tecnologías de acceso mediático y señalización disponibles en todas las tarjetas de interfaz de red (NICs) Ethernet.

Ethernet/IP está diseñado a partir de un estándar ampliamente implementado y utilizado en DeviceNet y ControlNet, denominado Protocolo de Control e Información (CIP). Este estándar organiza los mecanismos en red como una colección de objetos (o elementos) y define los accesos, atribuciones y extensiones.

Son muchas las ventajas del nivel del Protocolo de Control e Información (CIP) sobre Ethernet/IP. La oferta de un acceso consistente a aplicaciones físicas significa que se puede utilizar una sola herramienta para configurar dispositivos CIP en distintas redes desde un único punto de acceso sin la necesidad de software propietario. Al clasificar todos los mecanismos como objetos o elementos, se reduce la necesidad de adiestramiento y los costos de puesta en marcha requeridos cuando se incorporan nuevos mecanismos al perímetro de la red. Ethernet/IP disminuye el tiempo de respuesta e incrementa la capacidad de transferencia de datos respecto al DeviceNet o al ControlNet.

A través de un mismo medio de interconexión, Ethernet/IP conecta distintos mecanismos industriales con el control de planta y con la gestión central, mediante una interfaz consistente con las aplicaciones.

En la figura 4.2.1 se puede observar la arquitectura CIP para el protocolo Ethernet/IP.

4.4. ProfiNet

ProfiNet es el estándar abierto de Ethernet Industrial de la asociación PROFIBUS Internacional (PI) según IEC 61784-2; y uno de los estándares de comunicación más utilizados en redes de automatización.

ProfiNet está basado en Ethernet Industrial, TCP/IP y algunos estándares de comunicación pertenecientes al mundo TI. Entre sus características destaca que es Ethernet en tiempo real, donde los dispositivos que se comunican por el bus de campo acuerdan cooperar en el procesamiento de solicitudes que se realizan dentro del bus.

Partiendo de una conectividad básica, como es el cable Ethernet, y unas tramas de comunicaciones establecidas que correspondería a los niveles 1 y 2 del modelo OSI, PROFINET va incorporando nuevas funcionalidades denominadas "perfiles" de utilidad como ProfiSafe o ProfiEnergy, mediante una interpretación específica para cada caso de los datos transmitidos, modificando el nivel 7, que es el nivel de aplicación. En el caso de Profisafe, se transmiten datos de seguridad (safety), y en el caso de ProfiEnergy, datos y comandos para el ahorro y control energético.

Con PROFINET es posible conectar dispositivos, sistemas y celdas, que son conjuntos de dispositivos aislados entre sí, mejorando tanto la velocidad como la seguridad de sus comunicaciones, reduciendo costes y optimizando la producción. Por sus características, PROFINET permite la compatibilidad con comunicaciones Ethernet más propias de entornos TI, aprovechando todas las características de éstas, salvo la diferencia de velocidad que posee una comunicación Ethernet situada en una red corporativa frente al rendimiento en tiempo real que necesita una red industrial.

Adicionalmente el uso del estándar PROFINET en el nivel E/S pueden proporcionar las siguientes ventajas:

- Mejora la escalabilidad en las infraestructuras.

- Acceso a los dispositivos de campo a través de la red. PROFINET al ser un protocolo que utiliza Ethernet en su comunicación facilita acceder a dispositivos de campo desde otras redes de una forma más fácil.
- Ejecución de tareas de mantenimiento y prestación de servicio desde cualquier lugar. Es posible acceder a dispositivos de campo mediante conexiones seguras como por ejemplo VPN para realizar mantenimientos remotos.

PROFINET utiliza 3 servicios de comunicación que se enumeran a continuación:

- **Standard TCP/IP:** Este servicio se utiliza para funciones no deterministas, como parametrización, transmisiones de vídeo/audio y transferencia de datos a sistemas TI de nivel superior.
- **Real Time**: Las capas TCP/IP no son utilizadas para dar un rendimiento determinista a las aplicaciones de automatización, funcionando con unos tiempos de retardo en el rango 1-10ms. Este hecho representa una solución basada en software adecuada para aplicaciones típicas de E/S, incluyendo control de movimiento y requisitos de alto rendimiento.

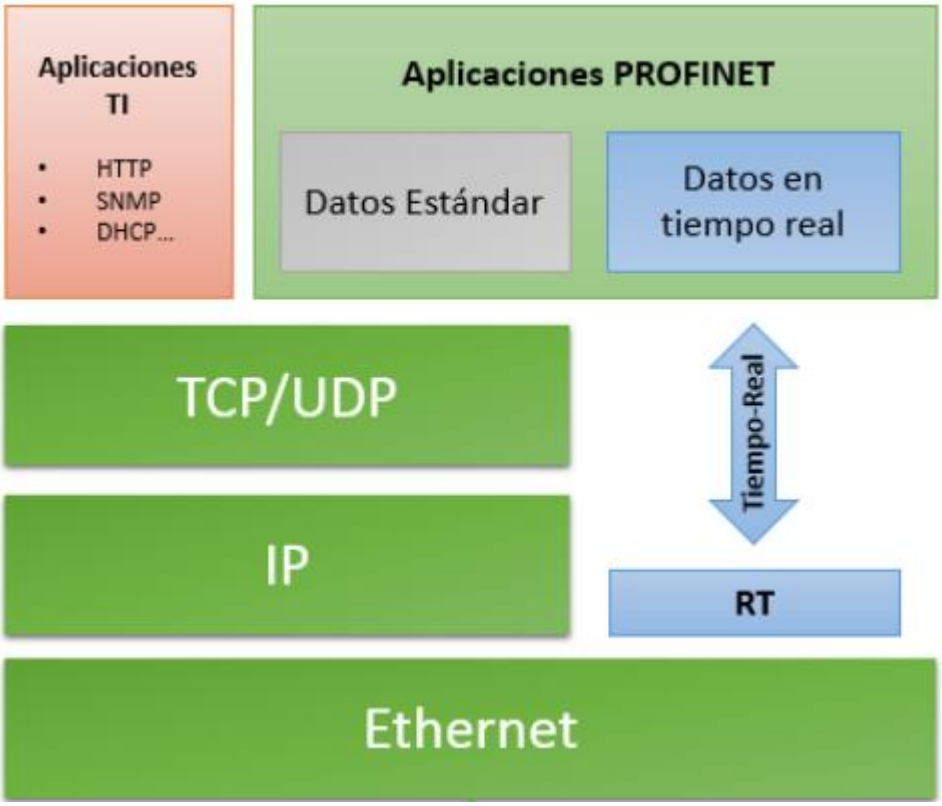

Figura 4.4.1: Servicios de comunicación en la red ProfiNet.

- **Isochronous Real Time:** La priorización de señal y la conmutación programada proporcionan una sincronización de alta precisión para aplicaciones como el control de movimiento. Las velocidades de ciclo en rangos de sub-milisegundos son posibles, con jitter (variabilidad temporal durante el envío de señales digitales) en el rango de sub-microsegundos.

4.5. CanOpen

CAN (Controller Area Network, por sus siglas en inglés) fue desarrollado por Bosch en 1985 para redes en vehículos. CAN, un sistema de bus serial de alta integridad destinado para comunicar dispositivos inteligentes, emergió como la red estándar para vehículos. La industria automotriz adoptó rápidamente CAN y, en 1993, se convirtió en el estándar internacional conocido como ISO 11898. Desde 1994, se han estandarizado varios protocolos de alto nivel a partir de CAN, como CANopen y DeviceNet, y su uso se ha extendido a otras industrias.

El bus de campo CAN sólo define las capas físicas y de enlace por lo que es necesario definir cómo se asignan y utilizan los identificadores y datos de los mensajes CAN. Para ello se definió el protocolo CANopen, que está basado en CAN, e implementa la capa de aplicación. Actualmente está ampliamente extendido, y ha sido adoptado como un estándar internacional. La construcción de sistemas basados en CAN que garanticen la interoperatividad entre dispositivos de diferentes fabricantes requiere una capa de aplicación y unos perfiles que estandaricen la comunicación en el sistema, la funcionalidad de los dispositivos y la administración del sistema:

- Capa de aplicación (application layer). Proporciona un conjunto de servicios y protocolos para los dispositivos de la red.
- Perfil de comunicación (communication profile). Define cómo configurar los dispositivos y los datos, y la forma de intercambiarlos entre ellos.
- Perfiles de dispositivos (device profiles). Añade funcionalidad específica a los dispositivos.

Los dispositivos CAN envían datos a través de una red CAN en paquetes llamados marcos.

Figura 4.5.1: Trama para el bus CAN.

Un marco de CAN consiste en las siguientes secciones como se puede observar en la figura 4.5.1 y que se explican a continuación:

- **Marco CAN – una transmisión completa de CAN**: arreglo de identificación, bytes de datos, bit de acknowledge, etc. Los marcos también son referidos como mensajes.
- **Bit SOF (start-of-frame**): indica el inicio de un mensaje con un bit dominante (lógica 0)
- **Arreglo de Identificación:** identifica el mensaje e indica la prioridad del mismo. Los marcos se presentan en dos formas:
    - Estándar, que utiliza un arreglo de identificación de 11 bits.
    - Extendido, que utiliza un arreglo de identificación de 29 bits.
- **Bit IDE (identifier extension):** permite la diferenciación entre marcos estándar y extendidos.
- **Bit RTR (remote transmission request):** sirve para diferenciar un marco remoto de un marco de datos. Un bit RTR dominante (lógica 0) indica un marco de datos. Un bit RTR recesivo (lógica 1) indica un marco remoto.
- **DLC (data length code):** indica el número de bytes que contiene el campo de datos.
- **Campo de Datos:** contiene de 0 a 8 bytes de datos.
- **CRC (cyclic redundancy check):** contiene un código de revisión cíclica redundante de 15 bits y un bit recesivo para delimitar. El campo CRC se utiliza para detectar errores.
- **Ranura ACK (ACKnowledgement):** cualquier controlador CAN que recibe mensajes correctamente envía un bit de ACK al final del mensaje. El nodo transmisor revisa la presencia del bit ACK en el bus e intenta nuevamente la transmisión en caso de no detectarlo.
- **Canal CAN:** un pedazo individual de datos contenidos dentro del campo de datos del marco CAN. También puede referirse a los canales CAN como señales. Debido a que el campo de datos puede contener hasta 8 bytes de datos, un solo marco CAN puede contener de 0 a 64 canales individuales (para 64 canales, todos tendrían que ser binarios)

4.6. Comunicación Serial

La comunicación serial es un protocolo muy común (no hay que confundirlo con el Bus Serial de Comunicación, o USB) para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. La mayoría de las computadoras incluyen dos puertos seriales RS-232. La comunicación serial es también un protocolo común utilizado por varios dispositivos para instrumentación; existen varios dispositivos compatibles con GPIB que incluyen un puerto RS-232. Además, la comunicación serial puede ser utilizada para adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo.

El concepto de comunicación serial es sencillo. El puerto serial envía y recibe bytes de información un bit a la vez. Aun y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. Por ejemplo, la especificación IEEE 488 para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre cualesquier dos dispositivos; por el otro lado, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: (1) Tierra (o referencia), (2) Transmitir, (3) Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben datos por otra. Existen otras líneas disponibles para realizar handshaking, o intercambio de pulsos de sincronización, pero no son requeridas. Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

- **Velocidad de transmisión (baud rate):** Indica el número de bits por segundo que se transfieren, y se mide en baudios (bauds). Por ejemplo, 300 baudios representan 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz. Las velocidades de transmisión más comunes para las líneas telefónicas son de 14400, 28800, y 33600. Es posible tener velocidades más altas, pero se reduciría la distancia máxima posible

entre los dispositivos. Las altas velocidades se utilizan cuando los dispositivos se encuentran uno junto al otro, como es el caso de dispositivos GPIB.

- **Bits de datos:** Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usar para referirse a todos los casos.
- **Bits de parada:** Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo, la transmisión será más lenta.
- **Paridad:** Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados.

4.6.1. RS-232

RS-232 (Estándar ANSI/EIA-232) es el conector serial hallado en las PCs IBM y compatibles. Es utilizado para una gran variedad de propósitos, como conectar un ratón, impresora o modem, así como instrumentación industrial. Gracias a las mejoras que se han ido desarrollando en las líneas de transmisión y en los cables, existen aplicaciones en las que se aumenta el desempeño de RS-232 en lo que respecta a la distancia y velocidad del estándar. RS-232 está limitado a comunicaciones de punto a punto entre los dispositivos y el puerto serial de la computadora. El hardware de RS-232 se puede utilizar para comunicaciones seriales en distancias de hasta 50 pies.

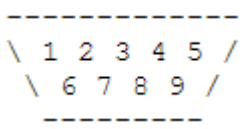

Figura 4.6.1.1: Pines del conector DB9.

En la figura 4.6.1.1 podemos observar los pines del conector para este bus, los pines indican los siguiente:

- Datos: TXD (pin 3), RXD (pin 2)
- Handshake: RTS (pin 7), CTS (pin 8), DSR (pin 6), DCD (pin 1), DTR (pin 4)
- Tierra: GND (pin 5)
- Otros: RI (pin 9)

4.6.2. RS-485

RS-485 (Estándar EIA-485) es una mejora sobre RS-422 ya que incrementa el número de dispositivos que se pueden conectar (de 10 a 32) y define las características necesarias para asegurar los valores adecuados de voltaje cuando se tiene la carga máxima. Gracias a esta capacidad, es posible crear redes de dispositivos conectados a un solo puerto RS-485. Esta capacidad, y la gran inmunidad al ruido, hacen que este tipo de transmisión serial sea la elección de muchas aplicaciones industriales que necesitan dispositivos distribuidos en red conectados a una PC u otro controlador para la colección de datos, HMI, u otras operaciones. RS-485 es un conjunto que cubre RS-422, por lo que todos los dispositivos que se comunican usando RS-422 pueden ser controlados por RS-485. El hardware de RS-485 se puede utilizar en comunicaciones seriales de distancias de hasta 4000 pies de cable.

En la figura 4.6.1.1 podemos observar los pines del conector para este bus, los pines indican los siguiente:

- Datos: TXD+ (pin 8), TXD- (pin 9), RXD+ (pin 4), RXD- (pin 5)
- Handshake: RTS+ (pin 3), RTS- (pin 7), CTS+ (pin 2), CTS- (pin 6)

- Tierra: GND (pin 1).

4.7. EtherCat

EtherCAT es un protocolo industrial abierto. Desarrollado específicamente para aplicaciones en las que el tiempo real es un elemento crucial, usa el ancho de banda disponible en una red Ethernet estándar con una gran eficiencia.

Las características principales del protocolo son: tiempo real (determinismo y sincronismo mediante relojes distribuidos), redundancia ante cortes en el bus de comunicación y una topología de conexionado flexible. El protocolo EtherCat está impulsado por The EtherCat Technology Group (ETG) del cual forman parte más de 2500 empresas del sector de la automatización, entre las que Beckhoff es la desarrolladora principal.

El protocolo transmite la información mediante el estándar de tramas de Ethernet IEEE 802.3, siendo su ethertype el 0x88a4, lo que evita modificar el marco Ethernet estándar.

Una característica fundamental del protocolo es que se comparte entre todos los esclavos un único reloj que funcionara como referencia temporal del sistema. Cada esclavo relacionará su reloj interno con el reloj de referencia, midiendo los retrasos entre los paquetes del par de ida y del par de regreso, de manera que en todo momento el tiempo en todos los elementos de la red queda completamente determinado y sincronizado.

EtherCat usa una topología o en estrella, o en árbol, o en anillo. Utiliza la arquitectura de envío de datos Maestro-Esclavo, Siendo el maestro el ordenador con sistema operativo Linux y el esclavo el controlador de LinMot.

El controlador de LinMot que se ha comprado de todos los buses de comunicación que dispone LinMot tiene este bus de comunicación.

5. Comunicación mediante EtherCat entre un ordenador con sistema operativo Linux y el controlador LinMot

En este apartado se va a explicar el control del motor lineal mediante el protocolo de comunicación EtherCat entre un maestro que es un ordenador con sistema operativo Linux y un esclavo que es el controlador de LinMot.

En este apartado se explicará también un pequeño programa que se ha realizado, para ejecutar este programa cada vez que se inicia el ordenador hayq eu introducir lo siguiente en el terminal de Linux:

LD_LIBRARY_PATH=/usr/local/lib

Export LD_LIBRARY_PATH

Para realizar las líneas de código puestas anteriormente tienes que ser superusuario en Linux por lo cual tienes que estar como root para poder ejecutarlas.

5.1. Configuración del Maestro

En este apartado se va a explicar cómo configurar el maestro y los archivos necesarios para que funcione.

En el maestro se necesita tener dos archivos específicos para poder establecer la comunicación, el primero el archivo del firmware, archivo que proporciona la propia compañía que desarrolla la placa de Linux Hilscher y que tiene extensión ".nxf".

El otro archivo que se necesita es el archivo de configuración que necesita el maestro, para la creación de este fichero se tiene que instalar el software SYCON.net de la propia compañía Hilscher, en un ordenador con sistema operativo Windows, y también se inserta la tarjeta de comunicación instalando el driver que se obtiene de la página del fabricante en Windows, hay que instalar tanto hardware como software, y con ella configurar la comunicación entre el maestro y el esclavo.

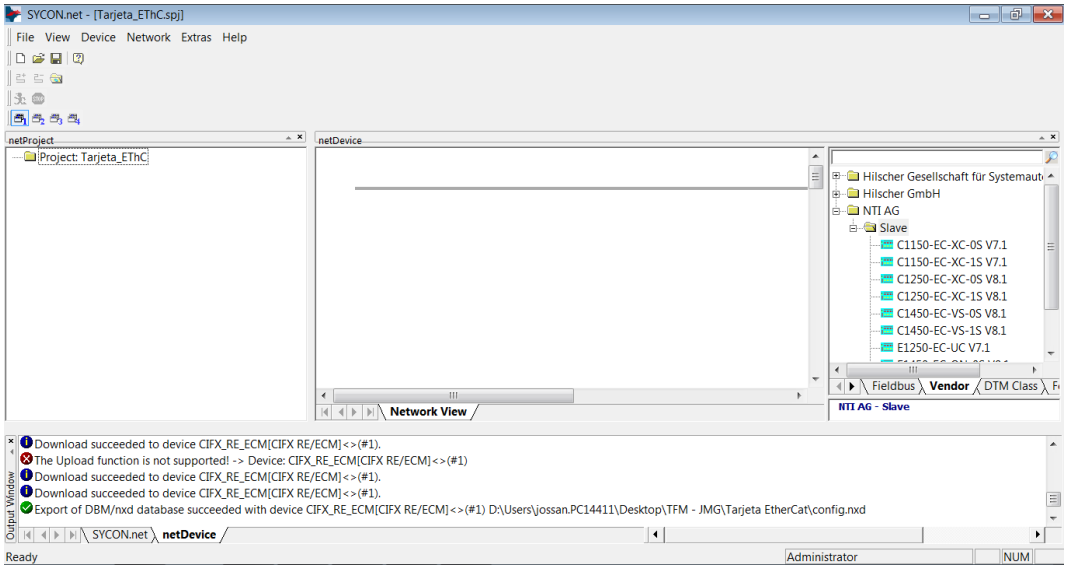

Figura 5.1.1: Configuración maestro EtherCat.

En el software lo primero que se tiene que hacer es insertar los ficheros .xml del esclavo, proporcionados por la compañía LinMot, para ello se observa cómo hacerlo en la siguiente imagen:

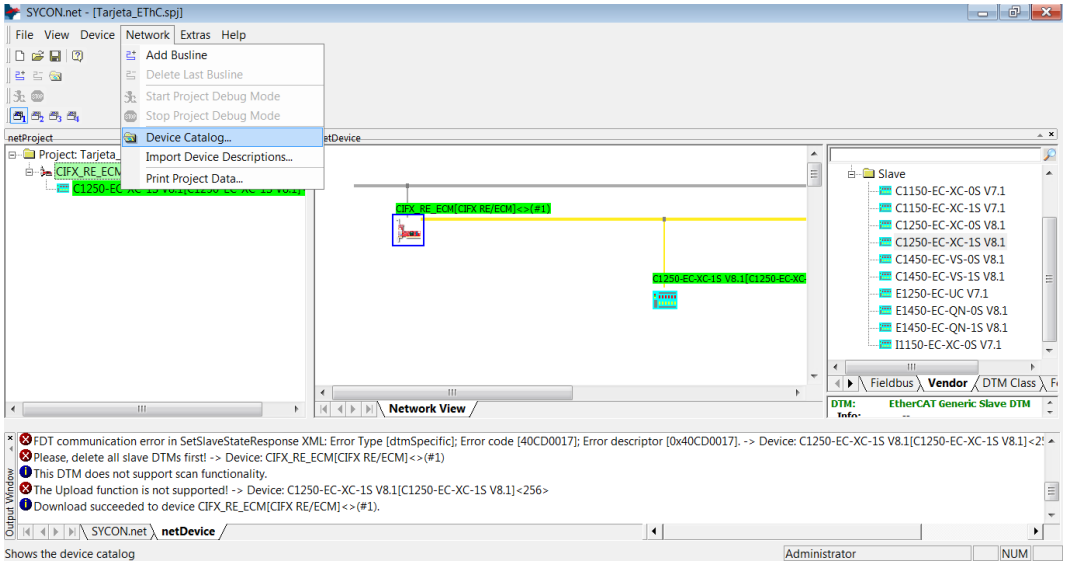

Figura 5.1.2: Configuración maestro EtherCat.

Ahora pinchando en import device descripcions introducimos los ficheros .xml del esclavo para que nos aparezca en el catálogo del programa

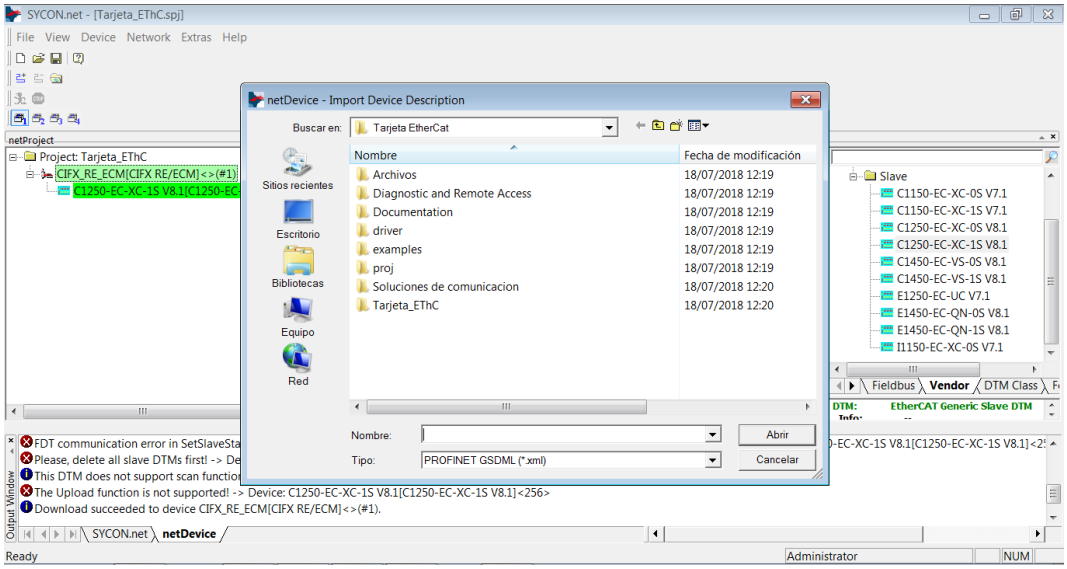

Figura 5.1.3: Configuración maestro EtherCat.

Una vez que se han importado nos aparecerá en el recuadro de la derecha en la pestaña Vendor como se ve en la siguiente imagen:

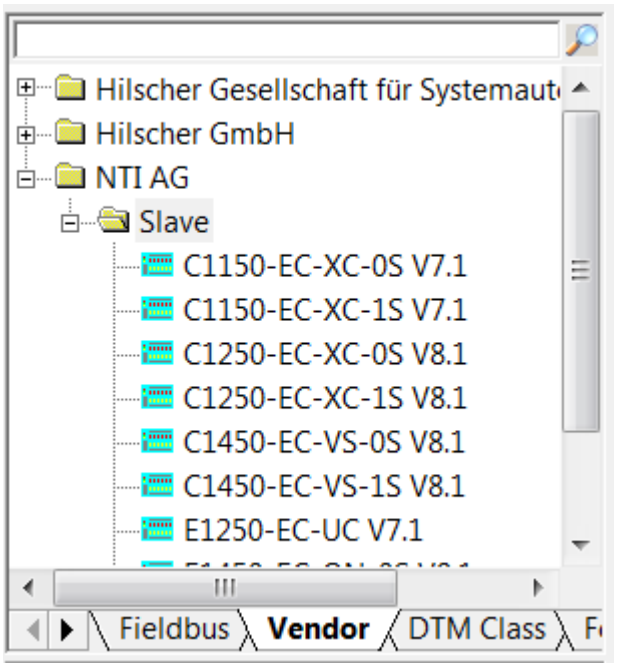

Figura 5.1.4: Configuración maestro EtherCat.

En este caso introducimos el controlador del que disponemos que es C1250-EC-XC-1S y le insertamos en el bus del recuadro siguiente:

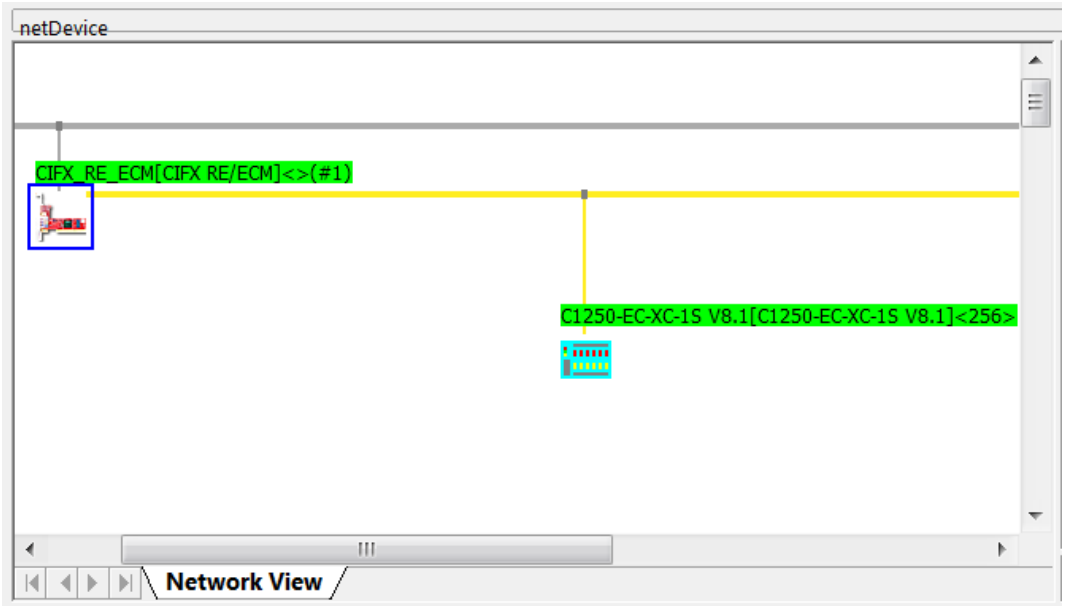

Figura 5.1.5: Configuración maestro EtherCat.

Una vez que esta insertado pulsamos con el botón derecho sobre el esclavo y damos a conectar, una vez conectado se pone de color verde.

Por ultimo pinchamos encima del maestro, CIFX_RE_ECM, con el botón derecho y pulsamos Download, con ello ya queda configurado el bus de comunicación y se están comunicando maestro y esclavo como se observa en la siguiente figura:

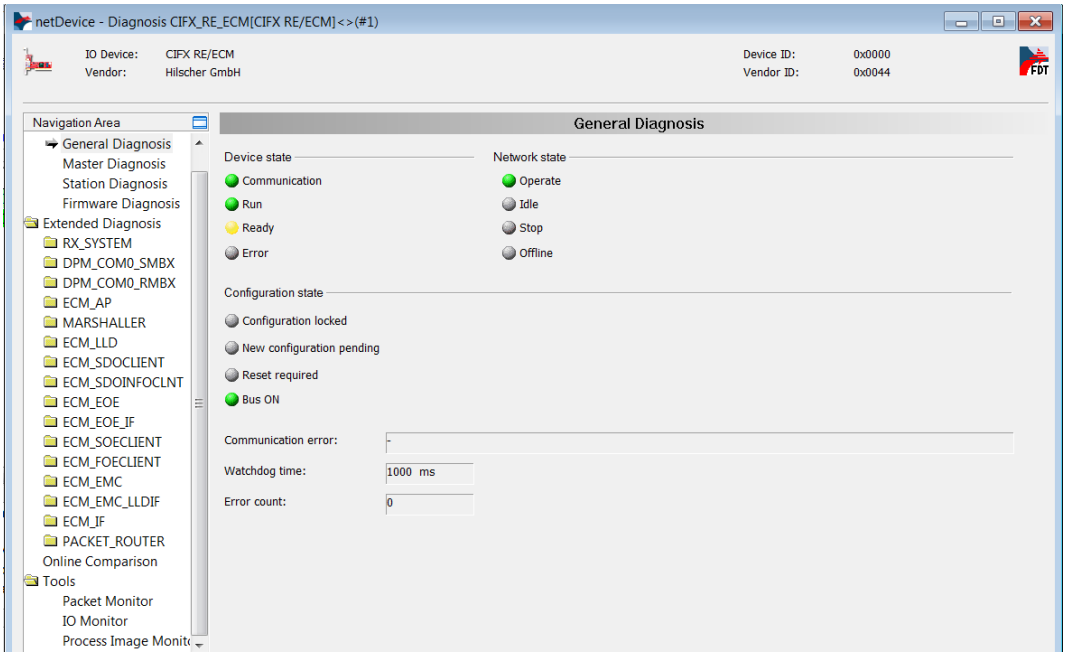

Figura 5.1.6: Configuración maestro EtherCat.

Para obtener la imagen anterior pulsamos encima del maestro y pulsamos donde pone diagnosis.

Con esto ya obtenemos el archivo de configuración que se necesita con extensión .nxd.

Para exportar este fichero se tiene que desconectar la comunicación con el maestro y encima del maestro botón derecho export y con ello obtenemos el archivo .nxd necesario para linux.

La comunicación entre maestro y esclavo se realiza con un cable ethernet pero tiene que ser cruzado si no no se puede comunicar, el cable cruzado se muestra a continuación como es:

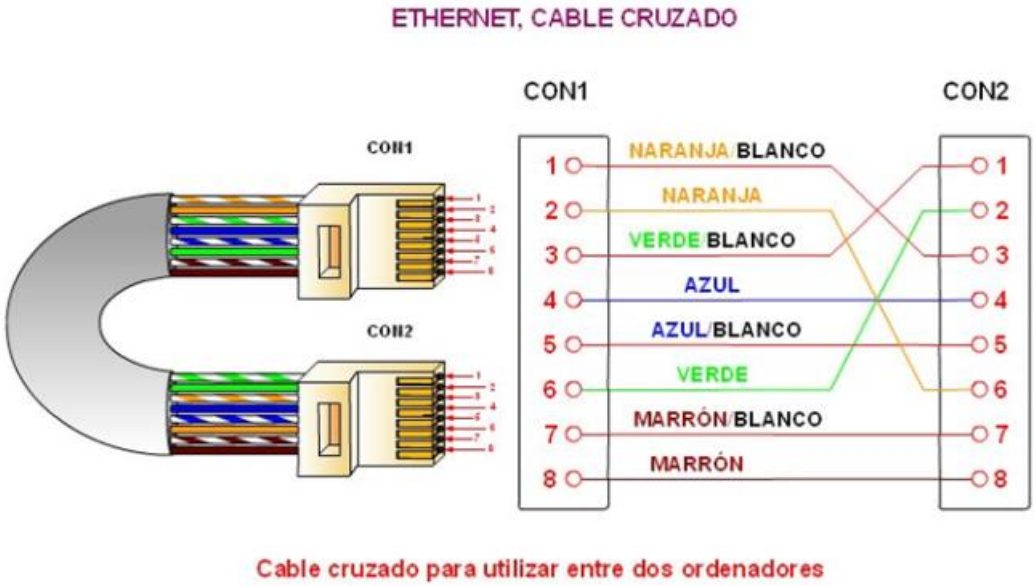

Figura 5.1.7: Configuración cable cruzado Ethernet.

El esclavo tiene la dirección 256 ya que el maestro pone esa dirección al esclavo como se observa a continuación:

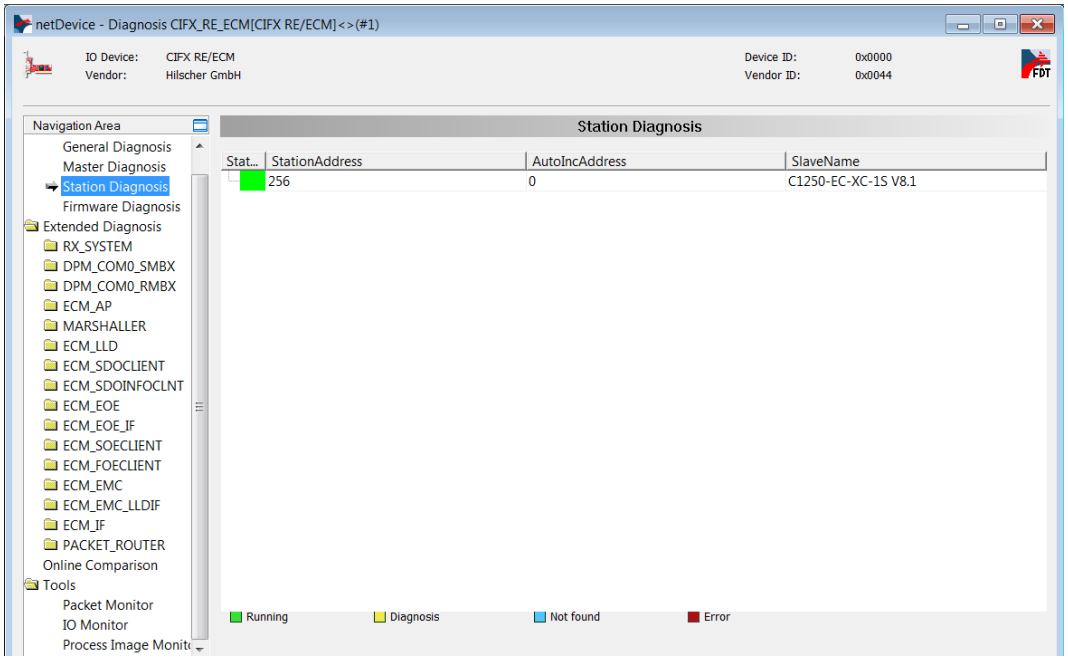

Figura 5.1.8: Configuración maestro EtherCat.

Una vez realizado esto se desmonta el hardware del ordenador de Windows y se vuelve a montar en el ordenador de Linux.

5.2. Tramas de envío y recepción.

En este apartado se va a explicar las tramas que se envían y se reciben entre el controlador de LinMot, el esclavo, y el maestro, el ordenador con sistema operativo Linux.

Las tramas que se han configurado en el maestro según lo que tiene que recibir el esclavo se pueden observar en las imágenes, 5.2.1 y 5.2.2.

Index	Size [Byte]	Byte Offset	Name	Data Type
0x1B00	***18***	**-**	***Variables***	***RECORD***
0x1B62:00	2	0	StateVar	Uint16
0x1D51:00	2	2	StatusWord	Uint16
0x1D8E:00	2	4	WarnWord	Uint16
0x1B8A:00	4	6	DemandPosition	Int32
0x1B8D:00	4	10	ActualPosition	Int32
0x1B93:00	4	14	DemandCurrent	Int32

Figura 5.2.1: Trama de recepción para el maestro EtherCat.

Index	Size [Byte]	Byte Offset	Name	Data Type
0x1700	***24***	**-**	***Variables***	***RECORD***
0x1D52:00	2	0	ControlWord	Uint16
0x1DB0:00	2	2	MotionCommandHeader	Uint16
0x1E40:00	4	4	MotionCommand Par 1	Word32
0x1E41:00	4	8	MotionCommand Par 2	Word32
0x1E42:00	4	12	MotionCommand Par 3	Word32
0x1E43:00	4	16	MotionCommand Par 4	Word32
0x1E44:00	4	20	MotionCommand Par 5	Word32

Figura 5.2.2: Trama de envío para el maestro EtherCat.

En las imágenes 5.2.1 y 5.2.2 se observa Uint16 que significa entero sin signo de 16 bits, también aparece Word32, que es una palabra de 32 bits, y también podemos ver Int32 que es un entero de 32 bits.

En la imagen 5.2.1 se observa la trama de recepción del maestro EtherCat o lo que es lo mismo la trama que envía el esclavo. La información que recibimos del esclavo es la palabra de estado, la palabra de alarma, explicadas ambas en los apartados anteriores, la posición actual, la posición demandada y la corriente demandada.

En la figura 5.2.2 se observa la trama que se envía desde el maestro al esclavo, se envía la palabra de control, la cabecera y los parámetros.

El maestro está configurado para que envíe 32 byte, aunque el esclavo solo tiene configura para recibir 24 byte, y para enviar 18byte, el resto hasta 32 byte se pondrá a cero.

A continuación, se muestra un ejemplo de cómo configurar la trama de envío al esclavo para que vaya a home:

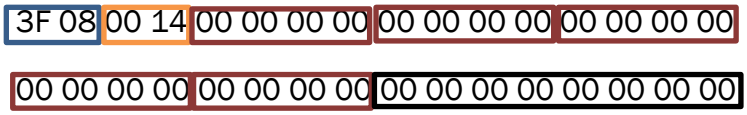

Figura 5.2.3: Campos de la Trama de envío.

En la figura 5.2.3 tenemos los campos para una trama de envío para el maestro Ethercat, en el recuadro azul tenemos 2 byte que es la palabra de control, en el recuadro naranja son 2 byte y es la cabecera, después tenemos los campos de color rojo que son de 4 byte cada uno en la que son los parámetros que hay que enviarle para haga lo que le hemos mandado, y de color negro son siempre a cero puesto que solo nos hacen falta 24 byte, pero se configura con 32 byte, por lo cual son siempre a cero. Para enviar la trama del maestro EtherCat son los mismos campos lo único que hay que cambiar dependiendo de que se desee que haga el motor los valores de dichos campos.

En la figura 5.2.3 tenemos 3F 08, que es la palabra de control para que vaya a home, teniendo en cuenta de que se escribe primero el byte de menor peso y después el de mayor. Después tenemos 00 14, que es para indicarle que vamos a cambiar la palabra de control, en este caso que vaya a home y como no requiere de ningún parámetro más el resto hasta 24 se pone a 0, se pueden enviar hasta 5 parámetros para cambiar en la misma trama.

5.3. Funciones realizadas para la comunicación mediante EtherCat

En este apartado lo que se va a poner y explicar es la intefaz que se ha desarrollado para usar en Linux, en ello lo que se realiza son las funciones en las que se desarrolla las diferentes opciones que se pueden realizar.

También se ha realizado un pequeño programa con un menú, dicho menú se observa en la imagen 5.3.1, que según se introduzca el número que pone realiza funciones como pedir al esclavo que envié los datos por EtherCat y mostrarlos por pantalla, o establecer comunicación por EtherCat, o enviar para que vaya a Home, o enviar para que vaya a la posición inicial, o que vaya a la posición máxima configurada en los apartados anteriores, este programa se puede observar en el anexo 1.

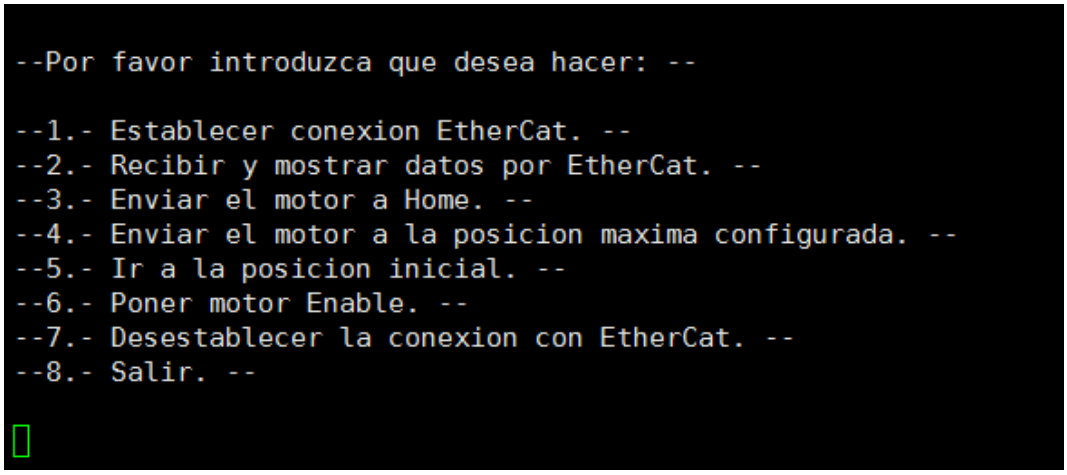

Figura 5.3.1: Menú del programa realizado en lenguaje C para Linux.

Antes de comenzar a explicar las funciones realizadas se va a hablar sobre ingeniería software, los procesos realizados antes de llegar a estas funciones.

- Lo primero que se realiza es la obtención de los requisitos y su análisis, se valoró lo que se necesitaba antes de ponerse a programar lo que se requeria era hacer un interfaz para manejar un motor, para en el futuro cuando se instalasen varios motores ya tener la base de la programación de ellos. En este punto se cogío los documentos del fabricante de LinMot para ver lo que se podía realizar con el motor, para observar las especificaciones, en una primera idea se pretendía realizar la interfaz con un sistema operativo de tiempo real, pero se observo que la dinámica de un solo motor no es muy rápida por lo que no requería un sistema operativo de tiempo real, si se conectan varios motores ya si que sería necesario.
- Luego una vez realizado el análisis de los requisitos y con la documentación del fabricante observando las especificaciones tanto del motor como del controlador LinMot como de la tarjeta del maestro de EtherCat, el siguiente paso era pensar en la programación la arquitectura del programa y como se iba a desarrollar el código. En este punto lo que se hizo fue estudiar las funciones que proporcionó el fabricante de la tarjeta del maestro EtherCat, puesto que esas funciones eran las que se iban a necesitar. La programación se fue realizando paso a paso y comprobando que funcionaba, de manera incremental, es decir hasta que no funcionase una función no se avanzaba.

- Cuando ya se obtuvo la aplicación había que observar que funcionaba por lo cual se realizaron las pruebas que se comentan en el apartado 5.4 para ver que realmente funcionaba de manera correcta lo que se diseño al principio.
- Por último, se realizó la documentación necesaria para el manejo de la aplicación con las funciones realizadas y su uso.
- Y el ultimo paso sería el mantenimiento de esta aplicación para que su funcionamiento sea correcto con el paso del tiempo.
- En este caso se utilizo un modelo en cascada para el desarrollo del software, como se puede observar en la figura 5.3.2, primero fueron los requisitos, luego el diseño, después la implementación, con sus respectivas pruebas, después la verificación y el ultimo paso sería el mantenimiento de la aplicación que en este caso pues no se realiza.

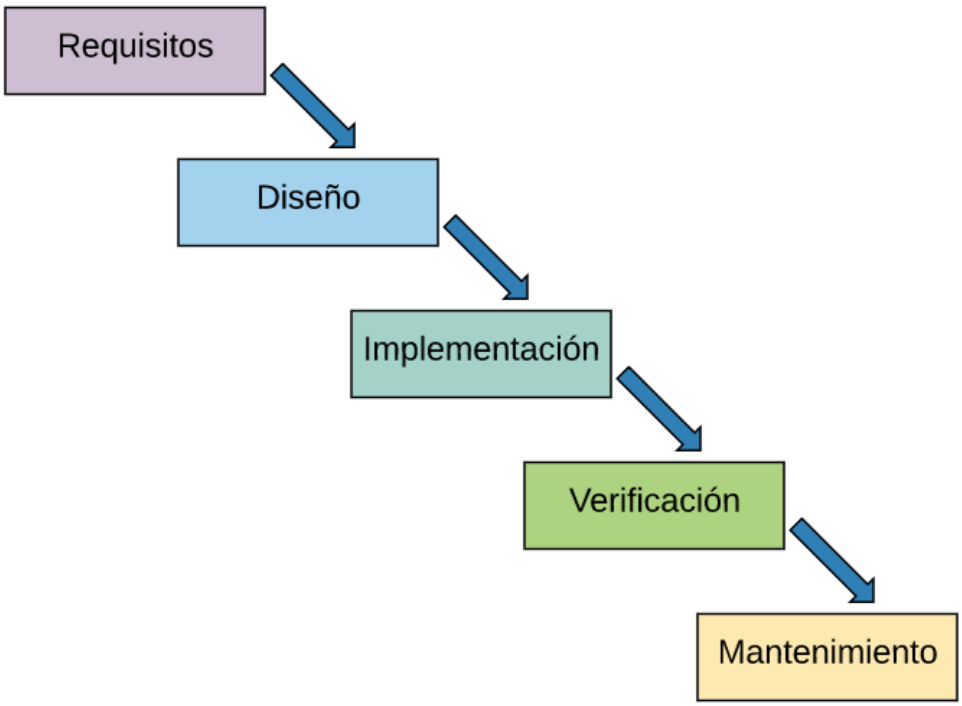

Figura 5.3.2: Modelo en Cascada.

5.3.1. Establecer comunicación por EtherCat

```
/*****************************************************************
*****************/
/*! Funcion para establecer conexion por Ethercat abrir el
driver y el canal
*
*/
/*****************************************************************
*****************/
void EstablecerConexion( void)
{
  lRet  = xDriverOpen(&hDriver);

  printf("---------- Estableciendo comunicacion mediante
EtherCat ----------\r\n");

  if(CIFX_NO_ERROR == lRet)
  {
    /* Driver/Toolkit successfully opened */
    //CIFXHANDLE hChannel = NULL;
    lRet = xChannelOpen(NULL, CIFX_DEV, 0, &hChannel);

    if(CIFX_NO_ERROR != lRet)
    {
        printf("Error de apertura de canal");
    }
    if( CIFX_NO_ERROR != (lRet = xChannelInfo(hChannel,
sizeof (CHANNEL_INFORMATION), &tChannelInfo)))
    {
        printf("Error querying system information
block\r\n");
    }
  }
}
```

En la función que se muestra arriba, EstablecerConexion, es la función que abre el driver y el canal para poder comunicarse por EtherCat, si se intenta enviar o recibir por EtherCat sin establecer la conexión se producirá un fallo.

5.3.2. Función para enviar por EtherCat

```
/*****************************************************************
*****************/
```

```
/*! Funcion para enviar los parametros al controlador
mediante ethercat, segun sea el valor del parametro enviado,
se le mandara a home o a la posicion maxima configurada.
*
*/
/*************************************************************
*****************/
void EnviarDatos( int n)
{
  // Read and write I/O data (32Bytes).
  unsigned char abSendData[32] = {0};
  unsigned char abRecvData[32] = {0};
  unsigned long ulCycles       = 0;
  unsigned long ulState;
  int num=0;

  unsigned char *St=abSendData;  //zz
  //St=abSendData;    //zz
  if (n==1){
     *St=0x3F;        //zz
     *(St+1)=0x08;          //zz
     *(St+2)=0x00;          //zz
     *(St+3)=0x14;          //zz
  }

  if (n == 2){
     *St=0x7F;
     *(St+1)=0x00;
     *(St+2)=0x00;
     *(St+3)=0x14;
  }

  if (n == 3){
           *St=0x3F;
     *(St+1)=0x20;
     *(St+2)=0x00;
        *(St+3)=0x14;
  }

  if (n == 4){
     *St=0x3F;
     *(St+1)=0x00;
     *(St+2)=0x00;
     *(St+3)=0x14;
  }

  if(CIFX_NO_ERROR != (lRet = xChannelIOWrite(hChannel, 0,
0, sizeof(abSendData), abSendData, 10)))
  {
     printf("Error writing to IO Data area!\r\n");
     //break;
  }
```

```
  #ifdef DEBUG
     printf("IOWrite Data:");
     DumpData(abSendData, sizeof(abSendData));
  #endif


}
```

En la función que se observa arriba, EnviarDatos, dependiendo del parámetro n que se le envíe, en este caso está configurado para recibir un valor entre 1 y 4, si se le envía 1 irá a home el motor, si se le envía 2 irá a la posición máxima configurada anteriormente, si se le envía 3 irá a la posición inicial, y si se le envía 4 el motor tendrá tensión y no se moverá.

Para mandarle al esclavo que es lo que tiene que hacer el motor se cambian los primeros 4 byte, la palabra de control, y los 4 byte siguientes que es la cabecera para decirle al esclavo que se le va a cambiar la palabra de control.

Para obtener las palabras de control que había que enviar al esclavo se han sacado mediante el software de LinMot Talk, explicado en los apartados anteriores, en el que se le iba activando la opción deseada y apuntando la palabra de control que se le enviaba.

Por ello las palabras de control que se han sacado son las siguientes:

- Home: 083F
- Ir a posición inicial: 0007
- Ir a posición máxima:2003
- Enable: 0003

5.3.3. Función para recibir por EtherCat

```
void RecibirDatos( void)
{
  // Read I/O data (32Bytes).
  unsigned char abRecvData[32] = {0};
  unsigned long ulCycles       = 0;
  unsigned long ulState;
  int num=0;

  if(CIFX_NO_ERROR != (lRet = xChannelIORead(hChannel, 0, 0,
sizeof(abRecvData), abRecvData, 10)))
  {
     printf("Error reading IO Data area!\r\n");
```

```
      //break;
   } else
   {
         //printf("ulCycles = %lu\r\n", ulCycles);
fflush(stdout);
         //printf("lRet= %d\r\n", lRet); fflush(stdout);
      // ZZ
         #ifdef DEBUG
            printf("IORead Data:");
            DumpData(abRecvData, sizeof(abRecvData));
         #endif

    }
}
```

En el código de arriba se muestra la función para recibir los datos que envía el controlador por EtherCat, se utiliza la función xchannelIOread, después lo que se recibe se muestra por pantalla.

5.3.4. Función para desconectarse de la comunicación por EtherCat

```
void Desconexion(void)
{
    printf("Cerrando el canal y la conexion\n");
    xChannelClose(hChannel);

    xDriverClose(hDriver);
}
```

En la función de arriba, Desconexion, lo que se realiza es cerrar el canal y cerrar el driver para cerrar la comunicación por etherCat.

5.4. Pruebas realizadas

Para comprobar que las funciones explicadas anteriormente funcionan correctamente una vez que se actuaban se procedían a una serie de pruebas.

- Función para establecer comunicación: Si establece comunicación entre el maestro y el esclavo, en la tarjeta del maestro EtherCat instalada en el ordenador de Linux se encienden 3 leds de color verde, si están los 3 leds encendidos de color verde según el fabriacante es que la comunicación es exitosa.
- Funciones para enviarle un comando: Cuando se le enviaba que fuera a una posición para comprobar que efectivamente iba a esa posición, se realizaban las comprobaciones según el fabricante, se conectaba al motor también por interfaz RS-232 al ordenador con sistema operativo Windows en el que se conectaba con el propio software del fabricante, LinMot Talk, y en el se puede observar si el motor llegaba a la posición que se le mandaba, viendo la distancia en mm, por ejemplo si se le enviaba a Home se tenía que observar que estaba en 0 mm, si se le enviaba a la máxima posición se tenía que observar que estaba a 100 mm, y cuando se conectaba al motor con el maestro, tiene una pestaña de enable que decía si estaba conetado. Y otra prueba que se realizaba es se marcaba donde estaba Home en la madera donde está montado el motor y cuando se le enviaba se observaba que llegaba a ese punto.

Para ver si funcionaba se conectaba al motor ambas comunicaciones, tanto EtherCat como RS-232, todos los comandos que se enviaban desde EtherCat se observaba en el propio software del fabricante como cambiaba lo que estábamos enviando, puesto que según el fabricante era la mejor forma de comprobar que realmente el controlador estaba recibiendo bien los datos.

Basandonos en las pruebas que el fabricante dijo se puede decir que el motor se movia según a lo programado.

6. Conclusiones y líneas futuras

Después de realiza este TFM se pueden sacar las siguientes conclusiones, el bus EtherCat es un bus bastante usada en la industria ya que proporciona seguridad y es un bus para sistemas de tiempo real, un bus para máquinas críticas.

Lo más complicado es configurar el maestro para que se comunique correctamente con el esclavo, una vez que se consigue configurarlo correctamente y con la información que proporciona el fabricante de LinMot como se compone la trama de envío y de recepción, ya simplemente es ir cambiando los parámetros para que el motor realice lo que se desee.

También otra cosa complicada es instalar correctamente los drivers del controlador en el ordenador Linux, ya que tiene que ser en la versión de Ubuntu correcta y la kernel compatible con ella ya que si no se producen errores al instalarle.

También mencionar el motor y controlador LinMot que son usados en la industria por su robustez y eficacia.

Como líneas futuras mencionar que la dínamica con un motor no es muy rápida por lo que no es necesario un sistema operativo de tiempo real, pero en aplicaciones futuras en el departamento de Sistemas y Automática se pretende desarrollar una plataforma, con 6 motores, en la que la dínamica al ser mas rápida se necesitara un sistema operativo de tiempo real.

Otra línea futura que se puede desarrollar es una suspensión activa para vehículos o para puentes con los motores líneales, en la que con el motor lo que se tratará es de compensar las vibraciones en los puentes o el estado de las carreteras en los vehículos.

Bibliografía

- 0185-1063-D_6V7_IG_Drives_C1250.pdf. “C1200 Servo Drive Installation”.
- 0185-1093-E_6V7_MA_MotionCtrlSW-SG5-SG7.pdf. “Motion Control SW”.
- 0185-1092-E_3V19_MA_MotionCtrlSW.pdf. “Documentation of the Motion Control SW”.
- 0185-1055-E_1V12_MA_EC-Motors-with-LinMot-Drives.pdf. “EC Motors with LinMot Drives”.
- 0185-1037-E_3V15_MA_EasySteps-Application.pdf. “Documentation of the EasySteps Application of the following Drive Series 1200”
- cifX API PR 03 EN.pdf. “CIFX API”.
- cifX Device Driver - Linux DRV 08 EN.pdf. “CIFX Device Driver”.

Anexo 1: Código utilizados

Los códigos utilizados se entragan en formato electrónico.