MICROCONTROLADORES Y ELECTRÓNICA DE POTENCIA

COMUNICACIÓN EN SISTEMAS MULTI-EJE

Jordán, Emmanuel

Iriarte, Eduardo

Tabla de contenido

1. Introducción

Un sistema de control multi-eje distribuido no se comunica por cables específicos para cada tipo de señal, sino con un único ***bus de campo***. Por este bus debe circular toda la información para sincronizar, diagnosticar y comandar cada eje.

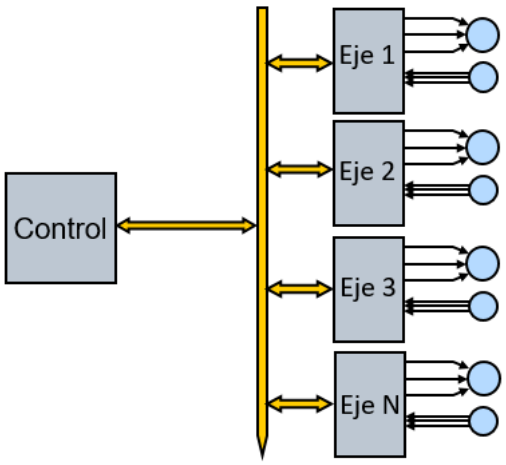

Figura 1. *Bus de campo en sistema de control multi-eje.*

Podemos clasificar los parámetros en función de la frecuencia o habitualidad con que se transmiten:

- Los que se comunican una vez durante toda la operación. Son parámetros como configuración de tipo de sensores (finales de carrera), parámetros del mecanismo (factores de reducción, rangos de desplazamientos), y otros como límites de velocidad, aceleración y parámetros de sintonía de los servos. Estos suelen quedar incluso en memoria permanente del controlador de eje.
- Los de mediana frecuencia son aquellos que cambian luego de varios tramos o entre modos, como aceleraciones, torques, etc.
- Los críticos son aquellos que deben ser entregados instantáneamente para cumplir de forma satisfactoria las trayectorias, como las posiciones, los estados de cada eje, etc.

Esta clasificación no es rígida y depende de la aplicación.

Lo que se deduce de esta caracterización es que, para los parámetros críticos, la comunicación es muy exigida en ***sincronización***, ***determinismo*** (poca dispersión en el tiempo), ***velocidad*** (para lograr ciclos de lazo cortos), ***capacidad para responder a eventos*** (cambios importantes en el estado de cada eje) y, por cuestiones de fiabilidad, la ***capacidad de recuperación de errores***.

A continuación se describen algunos protocolos estándar de comunicación:

1.1. MODBUS (RS-485)

Este protocolo trabaja sobre **RS-485**. Por su sencillez está incorporado en casi todos los servocontroladores, pero no es apropiado para control multi-eje en tiempo real. Sí se puede utilizar como bus secundario, para monitoreo o supervisión de variables de baja frecuencia.

En este protocolo el modo de comunicación es **Maestro-Esclavo**, es decir, un ***nodo maestro*** (por ejemplo, el encargado de coordinar los movimientos) ordena la comunicación con los ***nodos esclavos*** (por ejemplo, cada uno de los ejes esclavos).

El principal inconveniente de este esquema es que todas las comunicaciones son iniciadas por el maestro, por lo que para conocer el estado de los nodos esclavos es necesario realizar *polling*. Si algún servo sufriera un estado excepcional (ej. falla), sólo sería posible detectarlo si se le consulta. Ver animación en presentación de esta unidad.

1.2. Ethernet (CSMA/CD)

CSMA/CD: Carrier sense Multiple Access/Collision Detection

En un esquema de acceso múltiple al medio, cualquier estación puede iniciar una transmisión, con lo que se podría ahorrar el esquema de consultas (*polling*) del Maestro-Esclavo.

En caso de colisión de mensajes, las estaciones involucradas desechan los respectivos mensajes y esperan un tiempo distinto antes de volver a intentar transmitir. El tiempo distinto se consigue mediante aleatoriedad. En redes congestionadas se podría dar la situación de que una estación deba intentar varias veces antes de conseguir emitir su mensaje, lo que se traduce en alta dispersión del tiempo necesario para comunicar un parámetro. Ver animación en presentación de esta unidad.

Esta dispersión disminuye con la Ethernet conmutada, que segmenta los dominios de colisión. También usando el mismo hardware pero con otros esquemas de acceso al medio, como **maestro-esclavo** o **cliente-servidor** (Modbus TCP), o **anillo/línea** (SERCOS III, Ethercat, etc.)

1.3. CAN (CSMA/CA)

CSMA/CA: Carrier sense Multiple Access/Collision Avoidance

El *CANBUS* cumple con los requerimientos de un sistema de control multi-eje. Fue desarrollado por Bosch para el intercambio de información entre unidades de control electrónicas del automóvil. Es flexible, eficiente y robusto frente a interferencia electromagnética, e incorpora detección de errores. Este bus se compone de un **controlador** y un **transceptor** por cada nodo, **dos elementos finales** (resistencias de 120Ω) y **dos cables** trenzados.

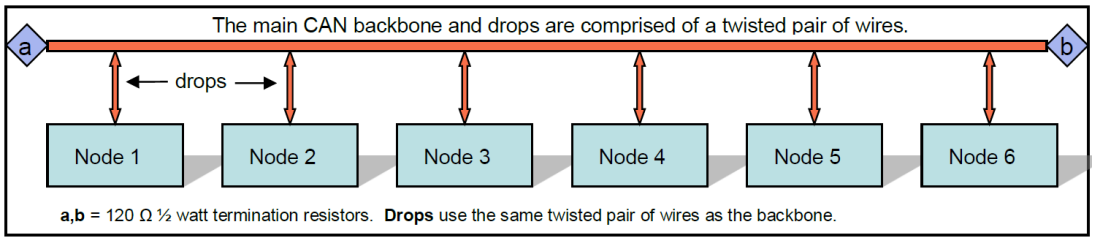

Figura 2. *Ejemplo de* CANBUS *con 6 nodos [1].*

En este esquema, en caso de contienda (que dos o más estaciones comiencen a transmitir al mismo tiempo), sólo una de las estaciones involucradas retira su mensaje (la de menor prioridad) y luego vuelve a reintentar, y el mensaje de la de otra no se corrompe. Esto hace más eficiente el uso del bus. Ver animación en presentación de esta unidad.

Toda la información se transmite a través de los cables bidireccionales ("*CANH*" y "*CANL*"), independientemente de la cantidad de nodos y de la cantidad de información transmitida. Además en ambos cables se transmite la misma información.

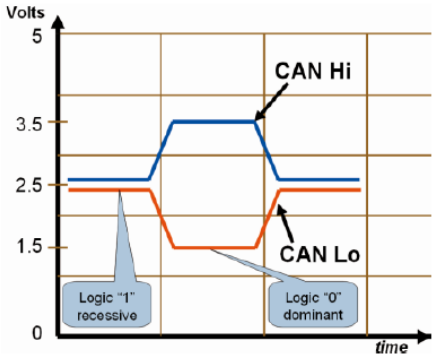

Figura 3. CANH y CANL, '1' lógico recesivo, '0' lógico dominante *[1].*

El sistema funciona con conexión a tierra en diferentes niveles de *CC*, ya sea cableado o conectado a chasis (o sin conexión a tierra). El par diferencial trenzado brinda una excelente inmunidad al ruido ya que lo que importa es la diferencia de tensión entre el par y no sus valores a tierra. De este modo, ante una *EMI* (interferencia electromagnética) que afecta al bus, las dos líneas se afectan por igual y no varía la diferencia de tensión original entre ambas (interferencia de modo común entrante). Además se cancela la *EMI* potencial saliente. Si el cable a tierra se corta (o no existe) siempre que *CANH* y *CANL* estén intactas, el sistema funcionará con capacidades de alto rendimiento. El par trenzado además brinda protección contra fallas del bus (como líneas abiertas) siempre que se disponga de tierra. Es muy sencillo agregar nuevos nodos a la red, "colgándose" de los cables trenzados.

El *CANBUS* es una red multiplexada (capaz de transmitir informaciones simultáneas por el mismo medio), orientada hacia el mensaje y no al destinatario. No es una topología de red Maestro-Esclavo, sino que todos los nodos pueden ser emisores o receptores en todo momento (es un bus "multi-master"). Sin embargo, en la aplicación cierto nodo puede trabajar como "Maestro" y los demás nodos como "Esclavos", por ejemplo un nodo coordinador que envía consignas de movimiento a distintos nodos subordinados.

La información es transmitida en forma de mensajes estructurados (**TRAMAS**), en los que una parte de los mismos es un identificador que indica la clase de dato que contiene.

Físicamente, *CAN* tiene una configuración de transistores similar a colector abierto (*open-drain*), pero de tipo diferencial. Esta configuración da más robustez para cubrir mayores distancias y mayor velocidad (máxima 1Mbps especificada por *CAN 2.0A/B*) comparado con colector abierto. Si bien es más lenta que otras configuraciones del tipo salida complementaria (*push-pull*) es más confiable, sin problemas de colisión de datos.

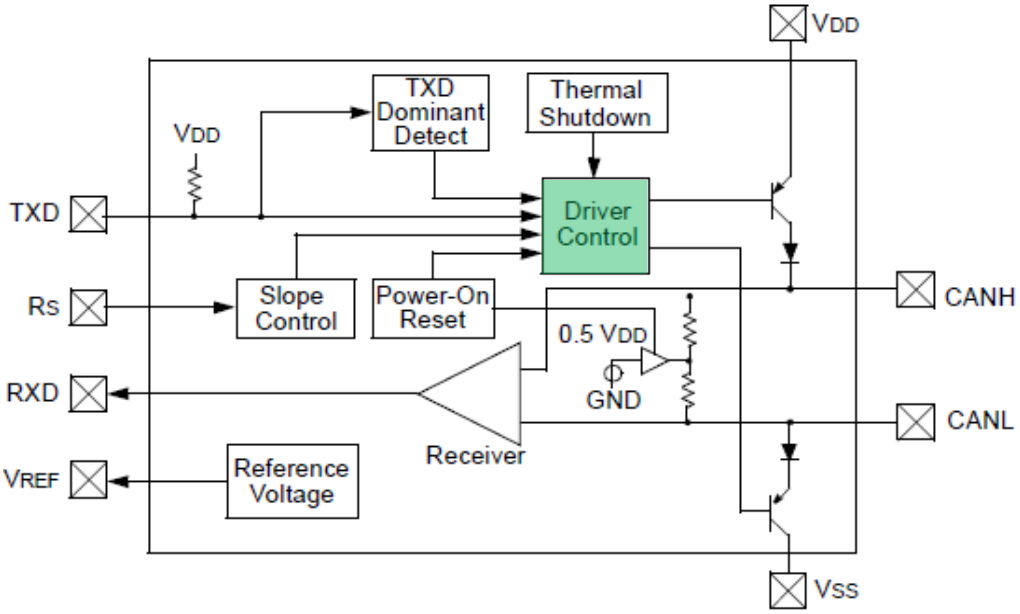

Figura 4. MCP2551 como transceptor CAN.

En caso de transmisiones simultáneas, la trama con más ceros en los bits más significativos continuará su transmisión y la otra trama será descartada hasta que finalice la primera, y luego volverá a reintentar. Es decir, existe una **prioridad** para identificadores menores (no está definida una jerarquía por dispositivo sino por tipo de mensaje) y las colisiones no son destructivas. Este conexionado por su comportamiento se conoce como "*wired AND*", con '0' como bit dominante y '1' como bit recesivo.

2. CANopen

CANopen es un ***protocolo estándar de comunicación***, abierto y en expansión, especificado para aplicaciones como control de movimiento, automatización de procesos industriales, equipamiento médico, energía, transporte, etc. Ha sido ratificado como estándar europeo *EN50325-4* e internacional *ISO15745-2*. Es un protocolo de ***capa 7*** (***aplicación***) en el modelo de referencia ***OSI*** (*Open System Interconection*) creado por la norma ***ISO*** (*International Organization for Standardization*) en 1984. Este modelo se creó para enfrentar el problema de incompatibilidad de redes informáticas, creando un conjunto de reglas aplicables de forma general a todas ellas. *CANopen* está directamente apoyado en las ***capas 1*** y ***2*** (***física*** y de ***enlace***) especificadas por el protocolo de comunicación serie ***CAN*** o ***CANBUS*** (*Controller Area Newtwork*, *ISO 11898*).

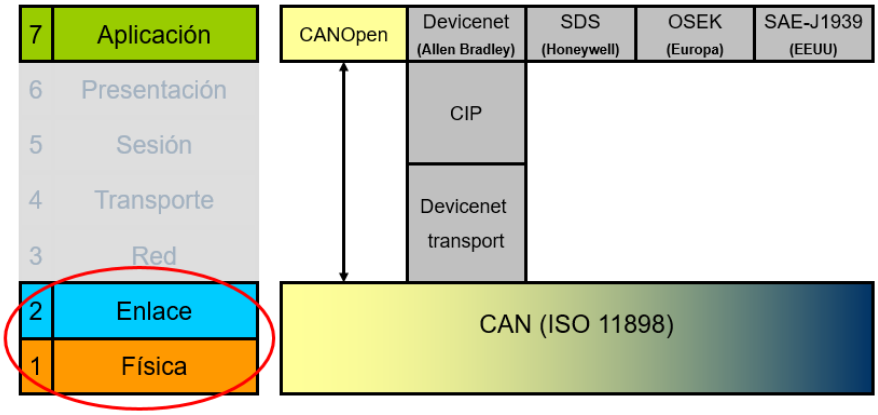

Figura 5. *Protocolo* CANopen*, según el modelo de referencia* OSI *de la* ISO *[2].*

CANopen especifica una máquina de estados para la ***Comunicación*** (Draft Standard 301) y otra para la ***Aplicación*** o ***Proceso*** (Draft Standard 402).

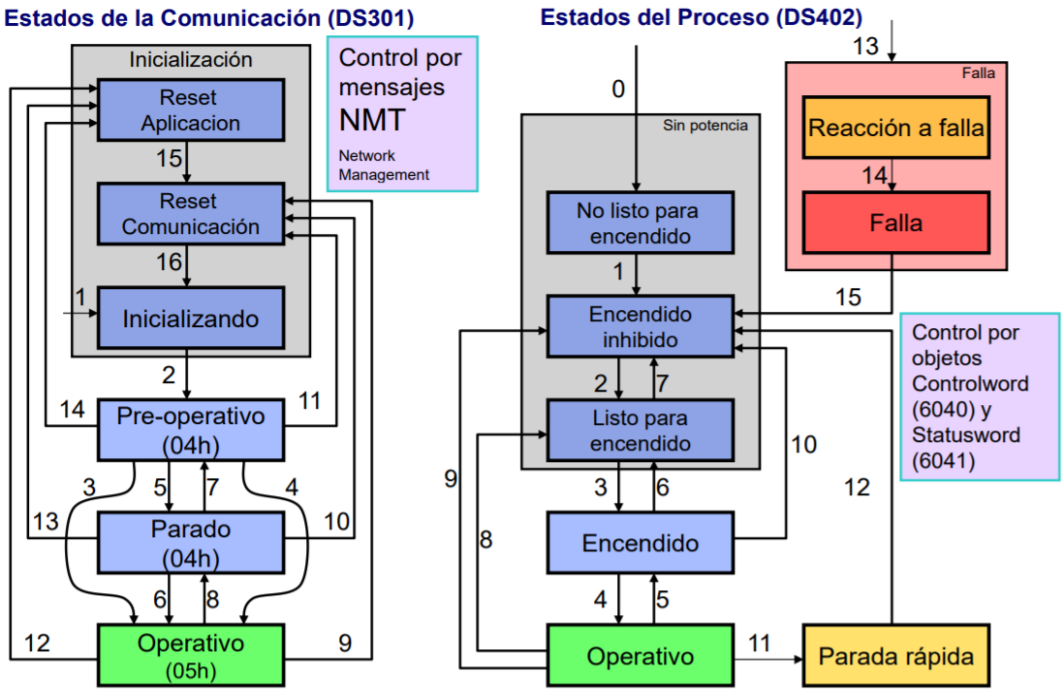

Figura 6. *Protocolo* CANopen*, máquinas de estados de la Comunicación y del Proceso [2].*

Actualmente *CANopen* está siendo incorporado como primer o segundo protocolo por los principales fabricantes de servocontroladores industriales (cumple un papel semejante al de *MODBUS* en la automatización industrial). Los equipos y dispositivos basados en el mismo son compatibles a nivel comunicación e interoperables a nivel aplicación, lo que garantiza la integración de sistemas independientes respecto de proveedores específicos.

Con *CANopen* se obtiene un acceso a prácticamente todos los puntos de un sistema motor-controlador, desde las variables cinemáticas (posición, velocidad, aceleración), dinámicas (fuerzas, torques) y electrónicas (límites de corriente y tensión, frecuencias de PWM), térmicas, geométricas (factores de reducción, configuraciones), parámetros de los lazos de control (ganancias y tiempos de lazos de posición, velocidad, corriente), máquinas de estados de comunicación y operación, modos de operación (referenciación, perfil de posición, posición interpolada), etc. Este acceso integral, sumado a la confiabilidad y bajo jitter del *CANBUS* sobre el cual se asienta, hace posible integrar con *CANopen* aplicaciones completas de movimientos interpolados, sincronizados con grillas de tiempo entre puntos de 1 a 10ms, tales como manipuladores robóticos.

2.1. Esquema general de CANopen

CANopen especifica la estructura lógica completa de un Dispositivo (ejemplo: servocontrolador), el cual consta de tres etapas:

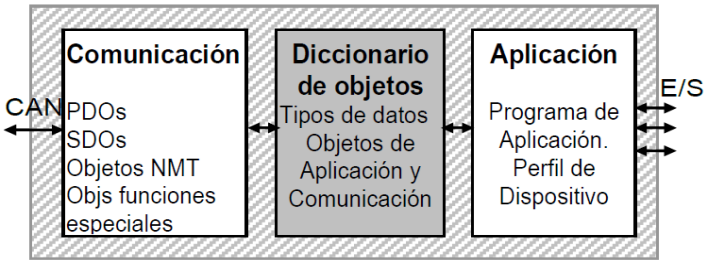

Figura 7. *Modelo de Dispositivo* CANopen *[2].*

- ***COMUNICACIÓN***: incluye la *interfaz* y el *software de protocolo* encargado de componer e interpretar los *tipos de tramas* (*objetos de Comunicación*) para vincularse con otros Dispositivos *CANopen*.
- ***APLICACIÓN*** *o* ***PROCESO***: comprende la *interfaz* para comunicarse con el proceso, el *programa de aplicación* y la implementación del *Perfil de Dispositivo*, el cual especifica cómo los *objetos de Aplicación* se vinculan a parámetros del proceso real.
- ***DICCIONARIO DE OBJETOS***: describe todos los *tipos de datos* y *objetos de Comunicación* y *Aplicación*, y los articula a través de tablas.

2.1.1. Comunicación

Los *objetos de Comunicación*, especificados en el *DS301* (CiA, 2002), definen tipos de comunicación: Maestro-Esclavo con y sin confirmación, Cliente-Servidor y Productor-Consumidor (push-model o pull-model). Estos objetos se agrupan en los ***SDO*** (objetos de datos de servicio), ***PDO*** (objetos de datos de proceso), ***NMT*** (Network Management) y mensajes especiales. Se los diferencia en los primeros cuatro bits o ***Función***, como se muestra en la Tabla 1.

Objeto	Func	ID (hex)	Contenido Normal
NMT	0000	000	Control estados comunicación
SYNC	0001	080	Sincronismo nodo productor
EMCY	0001	081-0FF	Emergencias nodos 1 a 127
TSTMP	0010	100	Timestamp nodo productor
TPDO1	0011	181-1FF	Status (6041)
RPDO1	0100	201-27F	Control (6040)
TPDO2	0101	281-2FF	Status (6041)+Modo (6061)
RPDO2	0110	301-37F	Control (6040)+Modo (6060)
TPDO3	0111	381-3FF	Status (6041)+Pos Act (6064)
RPDO3	1000	401-47F	Control (6040)+SetPos (607A)
TPDO4	1001	481-4FF	Status (6041)+Vel Act (606C)
RPDO4	1010	501-57F	Control (6040)+SetVel (60FF)
tSDO	1011	581-5FF	Respuesta a rSDO
rSDO	1100	601-67F	Cualquier Objeto DS301/402

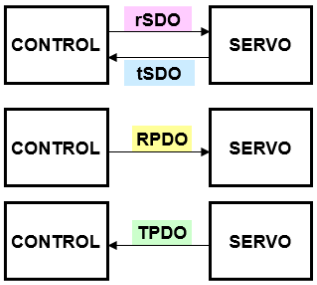

Tabla 1. Objetos básicos de Comunicación (Transmisión y Recepción vistas desde Dispositivo) [2].

Nota: los rangos de *ID* para los ***PDO***, ***SDO*** y ***EMCY*** son de 127 elementos, uno para cada Dispositivo *CANopen* direccionable.

Los ***SDO*** permiten acceder a los *objetos de Comunicación* y *Aplicación* mediante direccionamiento directo, es decir, en la propia trama se explicita la dirección del objeto.

Al enviar una ***SDO***, el Dispositivo que recibe siempre responde, es decir, confirma cualquier operación de lectura o escritura.

Los ***PDO*** utilizan direccionamiento implícito para acceder a objetos predefinidos mediante tramas de menor longitud, normalmente aquellos utilizados con mayor frecuencia.

El mapeo de los objetos referenciados por los ***PDO***, su activación y su comportamiento (intervalo de transmisión, máscara de eventos, etc.) se modifican mediante los ***SDO***.

Al enviar una ***RPDO***, la respuesta del Dispositivo (***TPDO***) es opcional, se puede silenciar. Si no está silenciada, responderá siempre que éste cambie su estado.

Los ***NMT*** permiten inicializar el Dispositivo y controlar los estados de la comunicación esquematizados en la Figura 6. Son mensajes enviados por el nodo maestro para encender, parar, resetear, etc., los nodos esclavos del sistema. Su identificador le brinda la máxima prioridad de emisión.

Nota: para modificar un ***PDO*** el estado de la comunicación debe ser *Pre-operativo*, y para comunicarse con el Dispositivo, debe ser *Operativo*.

2.1.2. Diccionario de Objetos

Es un mapa con todos los *objetos de Comunicación* y *Aplicación* del Dispositivo. Los objetos pueden ser simples, de tipos *int8*, *int16* o *int32*, o *arrays* de estos tipos. Las direcciones de mapeo están normalizadas, como se muestra en la Figura 8, ubicándose en el rango *1000h a 1FFFh* todos los objetos del ***Perfil de Comunicación***, y en el rango *6000h a 9FFFh* todos los objetos del ***Perfil de Dispositivo Normalizado***.

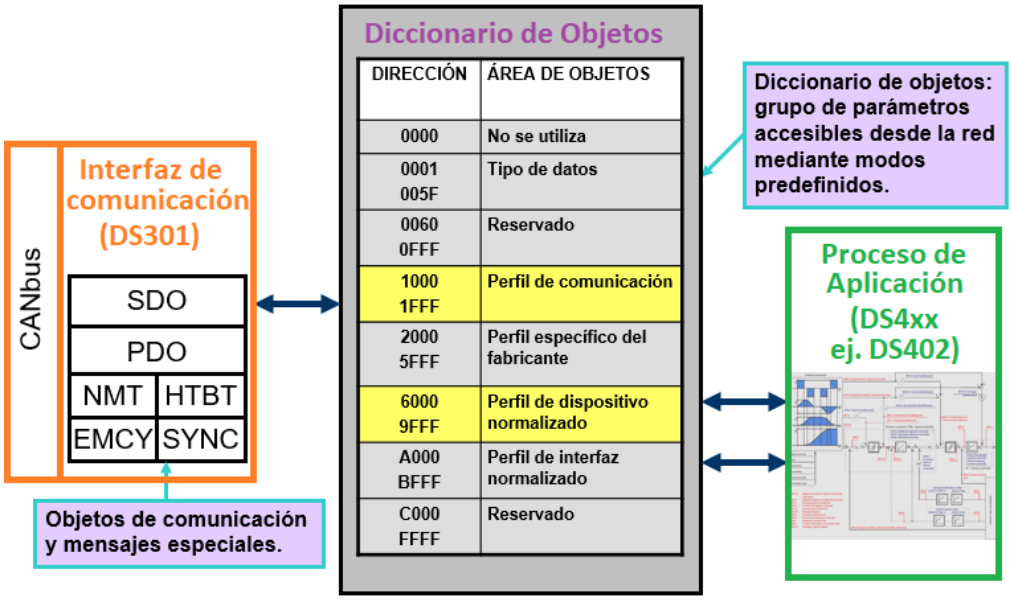

Figura 8. *Diccionario de objetos [2].*

El rango de los objetos del ***Perfil de Dispositivo Normalizado*** se subdivide en 8 rangos para un total de 8 ejes por cada Dispositivo servocontrolador. Esto para cada uno de los 127 Dispositivos *CANopen* direccionables.

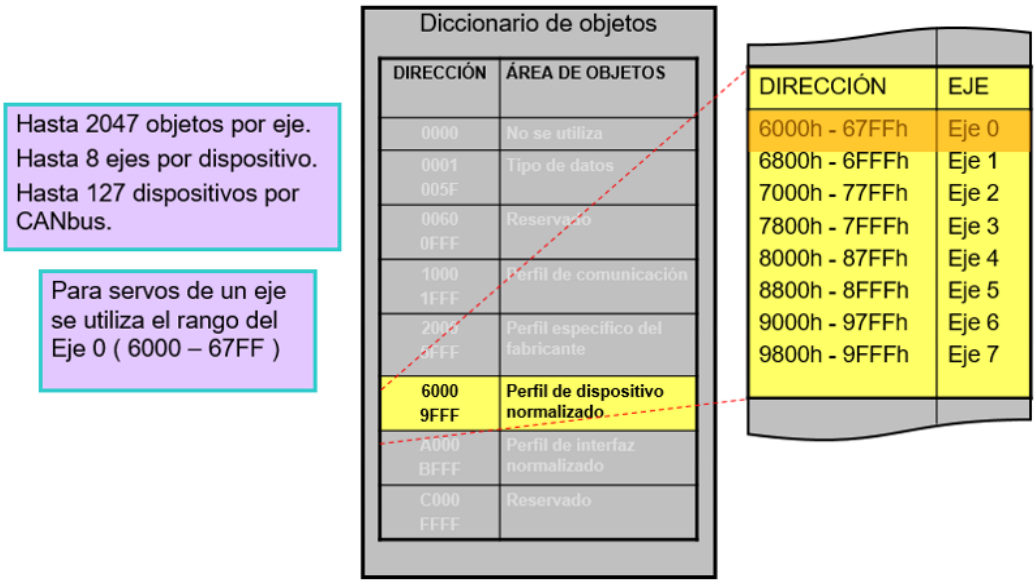

Figura 9. *Perfil de Dispositivo Normalizado [2].*

Nota: en la presente aplicación, se dispone de tres Dispositivos servocontroladores, con un solo eje cada uno (se utiliza el rango de Eje 0 para cada Dispositivo).

A continuación se muestran algunos ejemplos de objetos con sus direcciones asociadas (DS402):

Index	Object	Name	Type	Attr.	M/O
$607A_h$	VAR	target_position	Integer32	rw	M
$607B_h$	ARRAY	position_range_limit	Integer32	rw	O
$607D_h$	ARRAY	software_position_limit	Integer32	rw	O
$607F_h$	VAR	max_profile_velocity	Unsigned32	rw	O
6080_h	VAR	max_motor_speed	Unsigned16	rw	O
6081_h	VAR	profile_velocity	Unsigned32	rw	M
6082_h	VAR	end_velocity	Unsigned32	rw	O
6083_h	VAR	profile_acceleration	Unsigned32	rw	M
6084_h	VAR	profile_deceleration	Unsigned32	rw	M
6085_h	VAR	quick_stop_deceleration	Unsigned32	rw	O
6086_h	VAR	motion_profile_type	Integer16	rw	M
$60C5_h$	VAR	max_acceleration	Unsigned32	rw	O
$60C6_h$	VAR	max_deceleration	Unsigned32	rw	O

Figura 10. *Algunos objetos DS402 [3].*

Nota: *M*: obligatorio - *O*: opcional - *rw*: lectura/escritura - *ro*: sólo lectura.

2.1.3. Aplicación o Proceso

CANopen especifica en sus *DS4xx* numerosos ***Perfiles de Dispositivo***. El Perfil de Dispositivo para ***servocontroladores*** se encuentra en el *DS402* (CiA, 2006), cuya estructura se muestra en la Figura 11. El Dispositivo comprende un nodo *CAN* (*ISO 11898*), una capa basada en el *DS301*, y un Perfil de Dispositivo basado en el *DS402*.

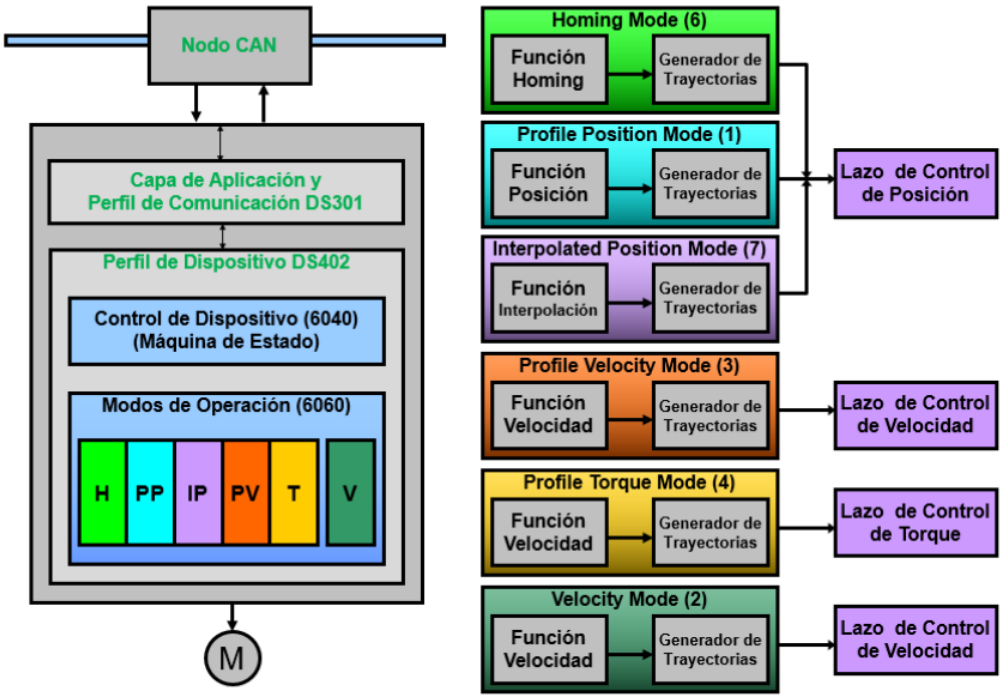

Figura 11. *Perfil de un servocontrolador [2].*

El Dispositivo funciona mediante una máquina de estados (Figura 6), cuyas transiciones son activadas remotamente con la ***Palabra de Control*** (*6040h*), enviada en un ***SDO*** o en un ***RPDO***. El estado del Dispositivo puede consultarse en la ***Palabra de Estado*** (*6041h*), enviada desde el servocontrolador en un ***TPDO*** o solicitando mediante un ***SDO***. Obsérvese en la Tabla 1 que los contenidos normales de los ***PDO 1*** a ***4*** incluyen las palabras de Estado y Control, lo que permite agilizar el manejo de los estados.

Por otra parte, el *DS402* ha previsto 6 ***modos de operación*** (ver Figura 11), seleccionados mediante el objeto *6060h* o consultados mediante el objeto *6061h*:

- ***Homing*** (***H***): Búsqueda de 1 o 2 referencias. Requerido para inicializar el eje.
- ***Profile Position*** (***PP***): Para control de posición punto a punto, con o sin detención, con control de velocidad y aceleraciones.
- ***Interpolated Position*** (***IP***): Para generar perfiles de movimiento complejos o movimientos coordinados, en el caso de sistemas multi-eje con control de trayectoria.
- ***Profile Velocity*** (***PV***): Para generar perfiles de velocidad.
- ***Torque*** (***T***): Control de torque para aplicaciones en las que se requiere aplicar fuerza o perfiles de fuerza determinados.
- ***Velocity Mode*** (***V***): Para regulación de velocidad (más sencillo que el *PV*).

2.2. Trama CAN 2.0A

La trama *CAN* versión *2.0* contiene campos perfectamente definidos, de los cuales algunos deben ser generados por la capa 7 (*CANopen*), detallados a continuación:

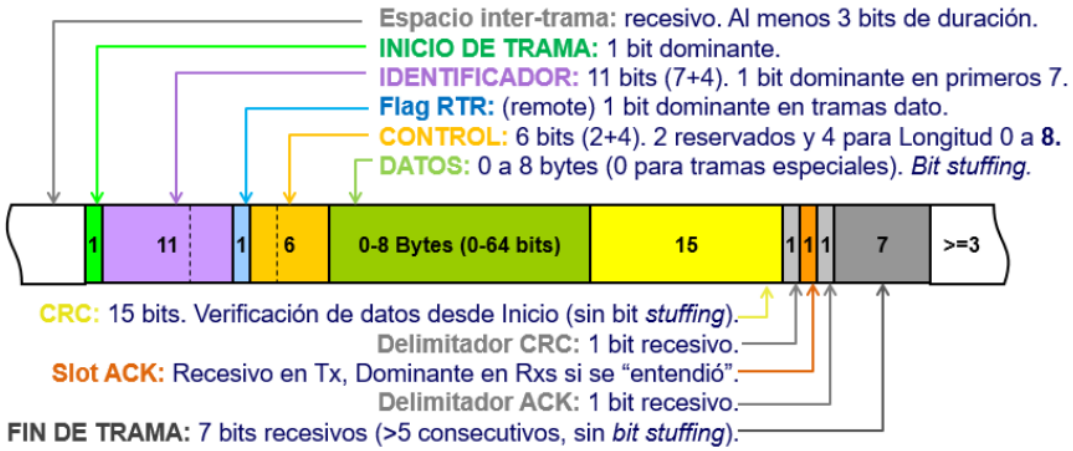

Figura 12. *Trama* CAN 2.0A *[2].*

Los bits del **IDENTIFICADOR** (11 en *CAN 2.0A* o 29 en *CAN 2.0B*) corresponden al *ID* de la Tabla 1. Para el caso de 11 bits, los 4 bits más significativos corresponden a ***Función*** (*NMT*, *PDO*, *SDO*, etc.) y los 7 bits restantes corresponden a la ***identificación de nodo*** (1 a 127).

- Ejemplo: *TPDO1* para servocontroladores 1, 2 y 3 (ver Tabla 1).

$$ID_1 = 181_h = 001\ 1000\ 0001_b = Func_{TPDO1} + 1$$
$$ID_2 = 182_h = 001\ 1000\ 0010_b = Func_{TPDO1} + 2$$
$$ID_3 = 183_h = 001\ 1000\ 0011_b = Func_{TPDO1} + 3$$

Objeto	Func	ID (hex)	Contenido Normal
TPDO1	0011	181-1FF	Status (6041)

Tabla 2. *Objeto básico de Comunicación TPDO1 [2].*

- Ejemplo: *RPDO4* para servocontroladores 125, 126 y 127 (ver Tabla 1).

$$ID_{125} = 57D_h = 101\ 0111\ 1101_b = Func_{RPDO4} + 125$$
$$ID_{126} = 57E_h = 101\ 0111\ 1110_b = Func_{RPDO4} + 126$$
$$ID_{127} = 57F_h = 101\ 0111\ 1111_b = Func_{RPDO4} + 127$$

Objeto	Func	ID (hex)	Contenido Normal
RPDO4	1010	501-57F	Control (6040)+SetVel (60FF)

Tabla 3. *Objeto básico de Comunicación RPDO4 [2].*

El **flag RTR** es un bit recesivo en las solicitudes y dominante en las emisiones de datos. Así, IDENTIFICADOR junto con flag RTR conforman la zona de arbitraje, en la que se evalúa, en caso de simultaneidad de mensajes, cuál permanece en el bus y cuál se retira, priorizando además la emisión por sobre la solicitud.

Los 6 bits de **CONTROL** corresponden a 4 bits que indican longitud o tamaño de datos (0 a 8 bytes), y 2 bits extra para indexado de elementos en caso de *arrays* (indexado *00b* para elementos comunes).

El campo de **DATOS** es donde se ubica la ***información*** propiamente dicha, es decir, direcciones y/o datos de los objetos solicitados o enviados en los *PDO* o *SDO*. Este campo será de longitud variable (0 a 8 bytes) según la información que se transmita.

- El direccionamiento de objetos se realiza con 16 bits, más 8 bits para indexado de elementos en el caso de *arrays* (indexado *00h* para elementos comunes).
- Para la parte correspondiente a la dirección, en la trama se utiliza orden "*LSB first*". Por ejemplo, el objeto "***consigna de Posición***" está en la dirección *607Ah*, índice *00h*, es decir *607A00h*. Por lo tanto, para dar consigna de Posición, se debe enviar la secuencia |*7A*|*60*|*00*|.

El resto de los campos (CRC, delimitadores) y el "*bit stuffing*" del campo de datos y el control de errores, los debe generar el hardware del microcontrolador conforme a la *ISO 11898*. Por ejemplo, el **bit Ack** es recesivo en el transmisor y dominante en el receptor. "Escuchando" este campo, el transmisor puede saber si la trama fue recibida correctamente sin necesidad de una trama de confirmación.

Nota: "*bit stuffing*" significa la inserción de un bit invertido cada 5 bits iguales consecutivos en el dato, para garantizar el sincronismo.

2.2.1. Trama NMT

Una trama NMT se arma como sigue:

COB-ID	rtr	len	NMT Function	Target Node
0x000	0	2	1 byte	1 byte

Figura 13. *Formato Network Management [2].*

El byte de *nodo objetivo* puede ser el número de esclavo de un nodo en particular, o bien *00h* para comunicarse con todos los nodos.

La *función NMT* puede ser:

Code	NMT Function
0x01	Enter Operational
0x02	Enter Stop
0x80	Enter Pre-operational
0x81	Reset node
0x82	Reset communication

Figura 14. *Función Network Management [2].*

2.2.2. Ejemplos

- ***Ejemplo 1***: para enviar consigna ($1013d = 3F5h$) de aceleración (dirección $608300h$ en Figura 10), la cual es de tipo *uint32*, al nodo 1 mediante una ***rSDO*** ($ID = 601h$ en Tabla 1), se arma la siguiente trama:

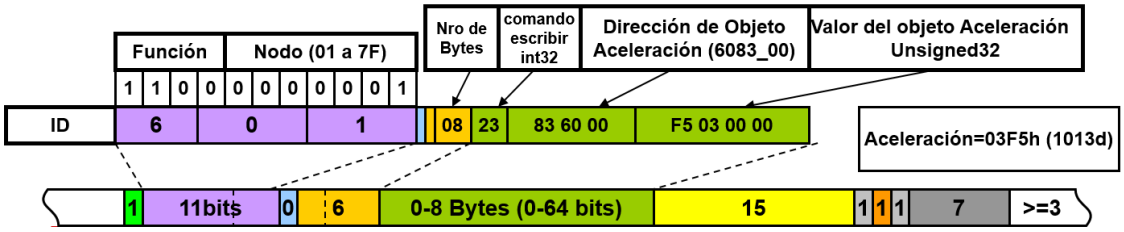

Figura 15. *Ejemplo de trama* CAN 2.0A *con* rSDO *(direccionamiento explícito) [2].*

- ***Ejemplo 2***: para cambiar el modo (6) de operación del Dispositivo del nodo 1 mediante una ***RPDO2*** ($301h$ en Tabla 1, cargado con *control* + *modo*), se arma la siguiente trama:

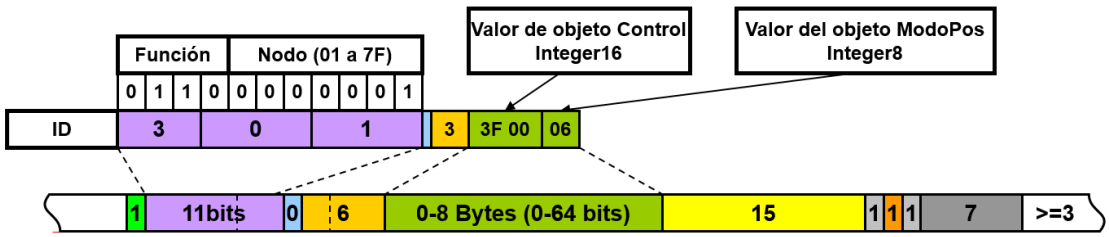

Figura 16. *Ejemplo de trama* CAN 2.0A *con* RPDO2 *(direccionamiento implícito) [2].*

- ***Ejemplo 3***: para enviar consigna ($65301d = FF15h$) de posición al nodo 1 mediante una ***RPDO3*** ($401h$ en Tabla 1, cargado con *control + consigna de posición*), se arma la siguiente trama:

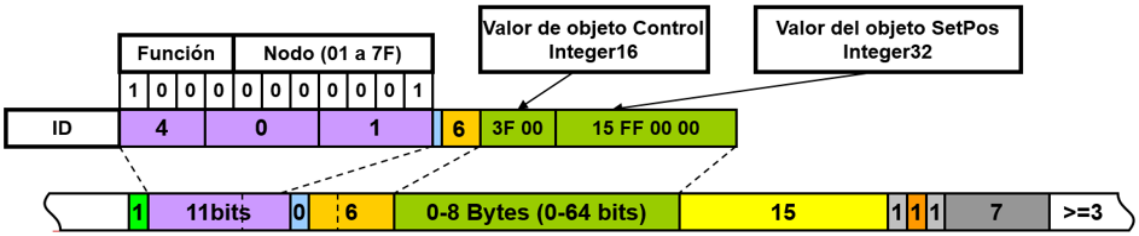

Figura 17. *Ejemplo de trama* CAN 2.0A *con* RPDO3 *(direccionamiento implícito) [2].*

2.3. Nodo Maestro CANopen

En aplicaciones de control de movimiento de uno o más ejes coordinados con arquitectura jerárquica, como un manipulador robótico, es conveniente disponer de un nodo Maestro como PLC, PC o microcontrolador, para dicha coordinación.

El nodo Maestro debe poder interpretar los estados de los Dispositivos para provocar las transiciones requeridas. Así podrá inicializar todos los ejes (activando las transiciones permitidas de las máquinas de estado vistas en la Figura 6) y conducirlos hasta el modo de operación deseado (por ejemplo modo *IP*). Por otra parte, sin considerar el cálculo cinemático, debe ser capaz de enviar consignas y solicitudes a los Dispositivos controlados e interpretar sus respuestas, sean lecturas de parámetros o estados. Además, hay que considerar que un Dispositivo con *TPDOn* activados, transmite cada vez que cambia su estado. Esto ocurre frecuentemente cada vez que recibe consigna, alcanza velocidad final, alcanza posición, etc. Si un nodo Maestro controla simultáneamente varios Dispositivos, debe estar preparado para recibir e interpretar la gran cantidad de mensajes que estos generan.

2.4. Nodos Esclavos CANopen

Las descripciones del *DS402* corresponden a Dispositivos a ser controlados desde un nivel superior. Los fabricantes de servocontroladores deben implementar estas especificaciones, y en tal sentido se encuentran disponibles notas de aplicación de fabricantes de chips *CAN* para implementar los *stacks CANopen* en microcontroladores (Fosler, 2004). Estos deben ser capaces de interpretar las solicitudes del nodo Maestro e informar el estado y parámetros del sistema que controlan.

3. Servocontroladores industriales

Los ***servocontroladores industriales***, como otros dispositivos industriales, comienzan a disponer de ***CANopen*** como protocolo estándar de comunicación. Los equipos y dispositivos basados en el mismo son compatibles a nivel comunicación e interoperables a nivel aplicación, lo que garantiza la integración de sistemas independientes respecto de proveedores específicos. Este protocolo además proporciona acceso a prácticamente todos los puntos de un sistema motor-controlador. Este acceso integral, sumado a la confiabilidad y bajo jitter del ***CANBUS*** (***ISO 11898***) sobre el cual se asienta, hace posible integrar con *CANopen* aplicaciones completas de movimientos interpolados, sincronizados con grillas de tiempo entre puntos de 1 a 10ms, tales como manipuladores robóticos.

En el Instituto de Automática y Electrónica Industrial (*IAEI*) se desarrolló un manipulador paralelo en configuración trípode o Delta Lineal (***Keops***) cuyos 3 primeros grados de libertad son manejados con la explotación plena de las posibilidades de *CANopen* y de un ***DSPIC*** para realizar trayectorias analíticas en el espacio de la tarea (rectas, arcos de círculo, parábolas, otras, con rotaciones y traslaciones) [4] [5].

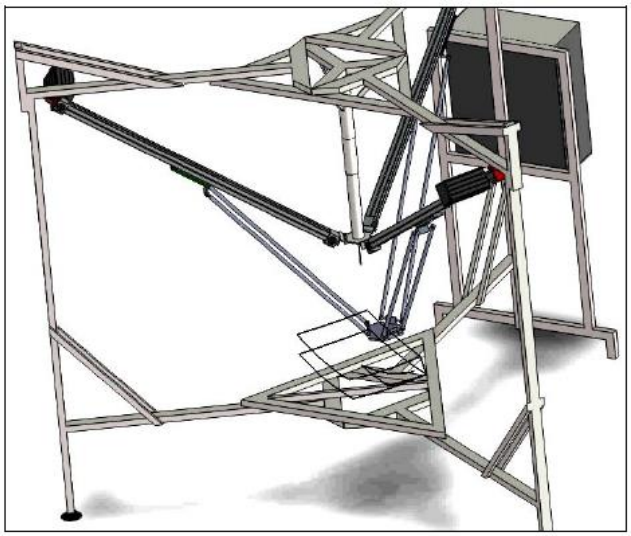

Figura 18. *Manipulador paralelo en configuración Delta Lineal [5].*

Con este mecanismo se ganó el 3° premio en la categoría de Robótica del Concurso Nacional de Innovaciones (2011) y el 1° premio en la categoría Mendoza Emprende del Concurso Mendoza Innova (2013).

Como alternativa al uso del *DSPIC*, queremos explorar las posibilidades de un *microcontrolador* con núcleo ***ARM*** y conectividad *CANopen* para implantar en él estas rutinas de control de movimiento. Se dispone para ello de placas de desarrollo basadas en *ARM Cortex-M4 (Discovery – STM32F407VG)*.

El mecanismo es impulsado por tres servomotores *EMMS-AS* de tipo brushless de *Festo* [6], controlados por servocontroladores *CMMS-AS* de *Festo* [7], con conectividad *CANopen*. El movimiento de rotación de cada eje de motor se convierte a movimiento lineal de cada articulación del robot a través de transmisiones por correa dentada.

Figura 19. *Servomotor EMMS-AS y servocontrolador CMMS-AS de Festo [7] [6].*

Además se cuenta con un eje desacoplado de la estructura del robot, compuesto por un servomotor *BCH* de tipo brushless de *Schneider Electric*, controlado por un servocontrolador *Lexium 23* (*LXM23A*) de *Schneider Electric* [8], con conectividad *CANopen*. Al igual que los ejes anteriores, el movimiento de rotación del eje del motor se convierte a movimiento lineal a través de transmisión por correa dentada.

Figura 20. *Servomotor Lexium 23 y servocontrolador LXM23A de Schneider Electric [8].*

Sobre este último conjunto se realizarán todas las pruebas de funcionamiento que se describirán en el presente informe, experimentando los distintos modos de trabajo del servocontrolador y los distintos modos operativos *CANopen*. Posteriormente se comandarán los tres conjuntos servocontrolador-servomotor de *Festo*, los cuales están montados sobre el robot trípode.

4. Servocontroladores Lexium 23

Se dispone de un servocontrolador *Lexium 23* (*LXM23A*) de *Schneider Electric*, con opción de conectividad *CANopen*, el cual mueve a un servomotor *Lexium BCH* de tipo Brushless [8]. El movimiento de rotación del eje del motor se convierte a movimiento lineal a través de una transmisión por correa dentada.

En la siguiente figura se muestra el frente del servocontrolador, con su interfaz hombre máquina (*HMI*) y los distintos conectores disponibles.

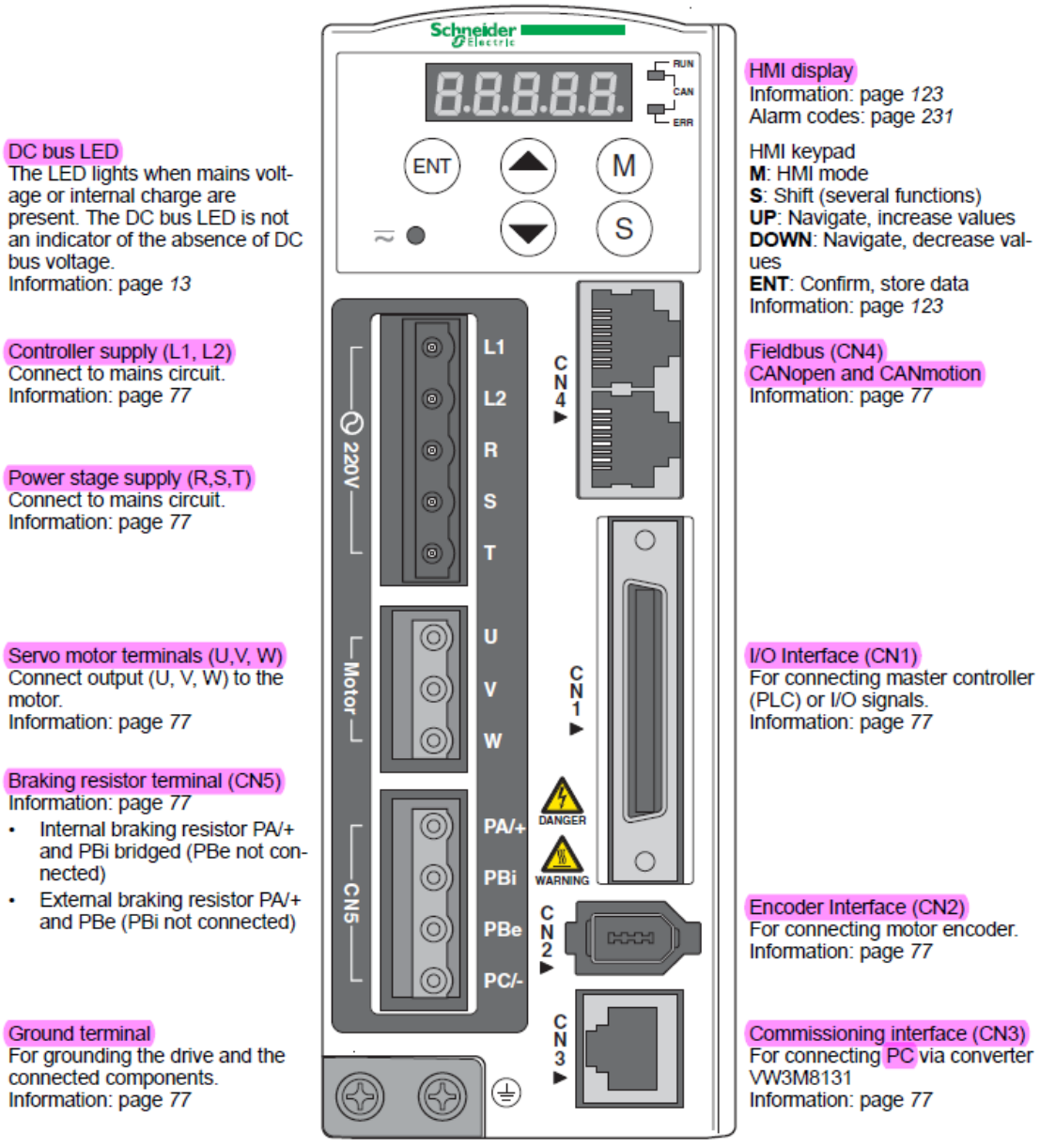

Figura 21. *Frente de servocontrolador LXM23A* ***[8].***

A continuación se muestra la estructura del sistema de accionamiento del mismo:

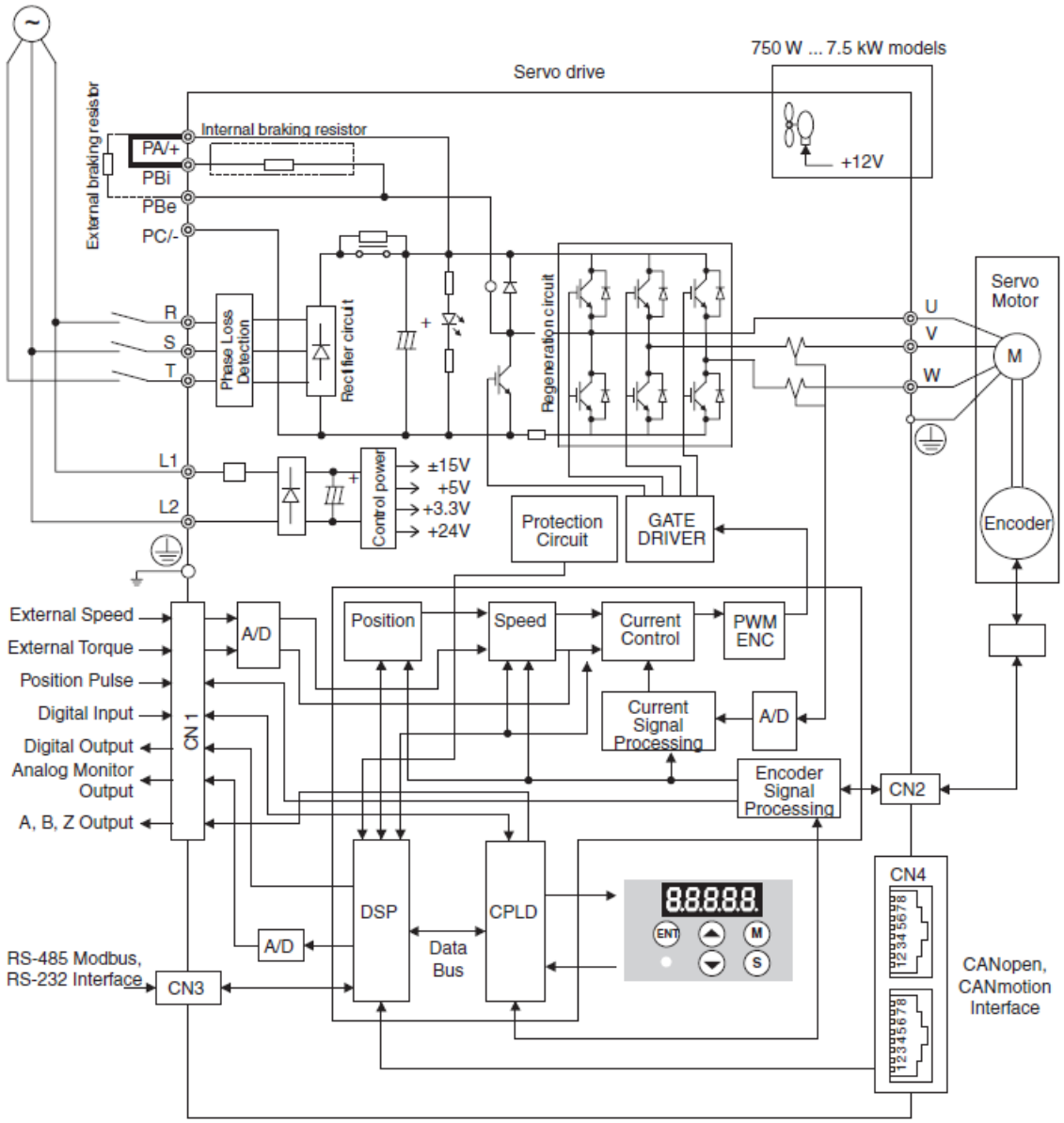

Figura 22. *Estructura del sistema de accionamiento del servocontrolador LXM23A* ***[8]****.*

Alimentación

Para servocontroladores de 0.2 a 1.5 kW la alimentación puede ser monofásica o trifásica, mientras que para servocontroladores de 2 a 7.5 kW debe ser trifásica. En el presente proyecto se cuenta con servocontroladores de 0.4 kW y se utiliza alimentación monofásica, conectando el neutro y la fase de la red (220V - 50Hz) con las entradas ***L1*** y ***L2*** (para el puente rectificador) y en paralelo con las entradas ***R*** y ***S*** (para la etapa de potencia). Se conecta además la tierra de la red con las entradas ***Ground*** tanto del servocontrolador como del servomotor.

Motor y Encoder

La conexión del motor se realiza mediante las salidas ***U***, ***V***, ***W*** y ***Ground*** del servocontrolador, con las correspondientes entradas del motor, según lo indica el manual de usuario. Además, las salidas del motor envían las señales de su encoder al servocontrolador a través del conector ***CN2***.

Grupos de parámetros

La modificación de los parámetros del servocontrolador permite configurar el funcionamiento del mismo. Estos se dividen en los siguientes grupos:

- Grupo 0: Parámetros del monitor (P0-xx)
- Grupo 1: Parámetros básicos (P1-xx)
- Grupo 2: Parámetros de extensión (P2-xx)
- Grupo 3: Parámetros de comunicación (P3-xx)
- Grupo 4: Parámetros de diagnóstico (P4-xx)
- Grupo 5: Parámetros de control de movimiento (P5-xx)
- Grupo 6: Parámetros de definición del path Pr (P6-xx)

En el manual del dispositivo se encuentra el detalle de la función de los parámetros de cada grupo y sus opciones de configuración (sección 10).

Interfaz HMI

La interfaz hombre-máquina es necesaria para la configuración de parámetros, elección del modo de trabajo, etc. En la misma:

- El pulsador ***M*** permite entrar y salir de los diferentes grupos de parámetros y además para cambiar el **modo HMI** (*Monitor* o *Editor de parámetros*).
- El pulsador ***S*** permite navegar hacia distintos grupos de parámetros, y dentro de la configuración de un parámetro específico permite mover el cursor a la izquierda para agilizar el cambio de sus valores.
- Las flechas ***Up*** y ***Down*** permiten moverse dentro de un mismo grupo de parámetros y también modificar sus valores.
- El pulsador ***Ent*** selecciona el parámetro a configurar, y además guarda y confirma los valores elegidos.

Modos de trabajo

El servocontrolador tiene la capacidad de operar en distintos modos según cómo se lo configure:

- ***JOG Mode***: modo simple de movimiento desde la interfaz *HMI* el cual permite elegir la velocidad (seteando un parámetro) y el sentido de giro (con los pulsadores ***Up*** y ***Down***). Útil para verificación de conexiones y como primera prueba de movimiento del motor.

- ***Single Mode***
  - External Position Control (Pt)
  - Internal Position Control (Pr)
  - Speed Control (S)
  - Internal Speed Control (Sz)
  - Torque Control (T)
  - Internal Torque Control (Tz)
- ***Dual Mode***
  - Pt-S,
  - Pt-T
  - Pr-S
  - Pr-T
  - S-T
- ***CANopen Mode***: control externo desde nodo Maestro por interfaz *CANopen*.

Se muestra el diagrama de conexión para el modo *CANopen* en la siguiente imagen.

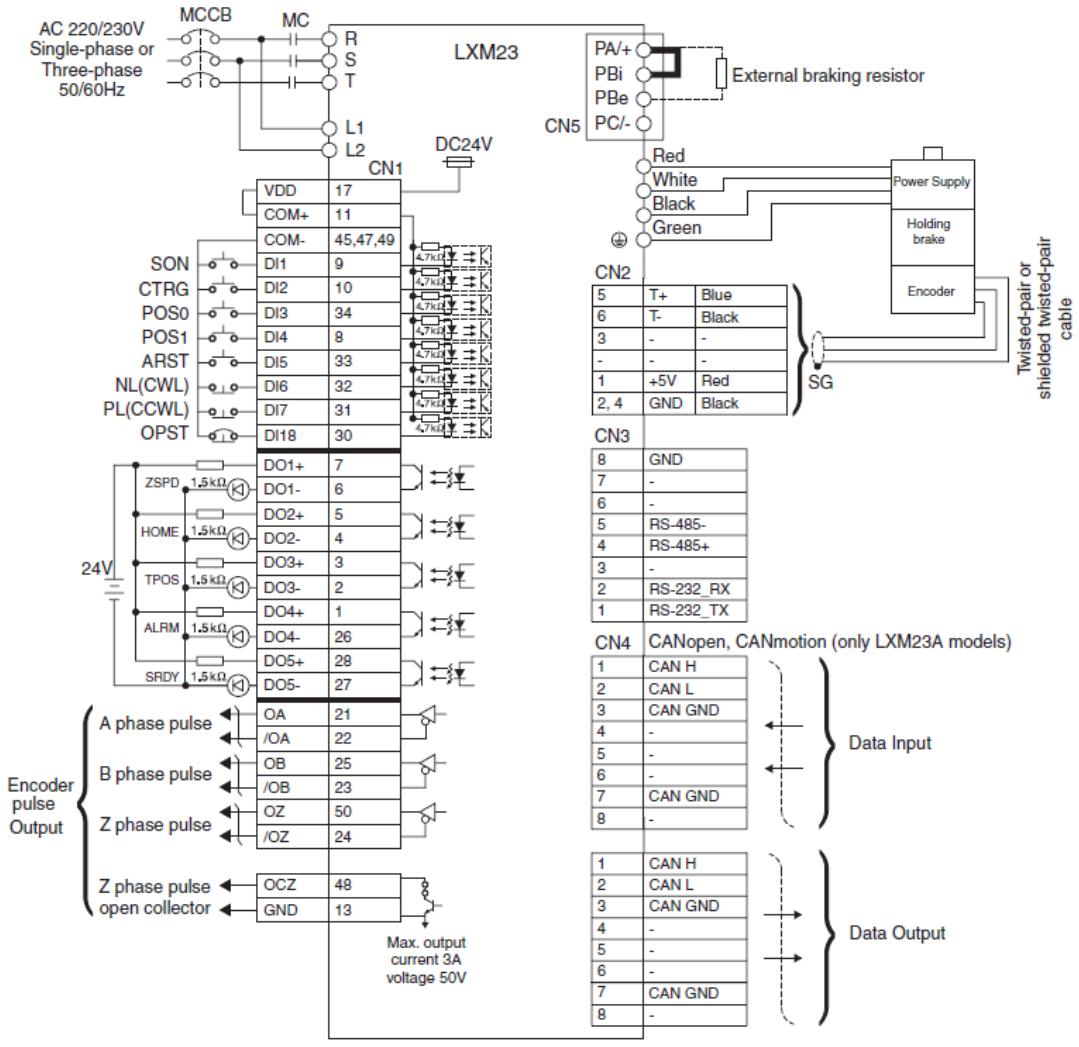

Figura 23. *Conexionado para modo* CANopen *del servocontrolador LXM23A* ***[8]****.*

Entradas y salidas (I/O) digitales

Las I/O digitales del servocontrolador, accesibles desde el conector ***CN1***, permiten interactuar desde un circuito externo enviando y/o recibiendo señales. Estas tienen una asignación estándar definida (según el modo de trabajo actual) pero se pueden modificar según la necesidad del usuario.

Señales generales

El conector ***CN1*** brinda además otro grupo de pines para señales generales, tales como control analógico de torque y velocidad, referencia del encoder del motor, entradas de pulso/dirección y voltajes de referencia. Estas señales, a diferencia de las I/O digitales, no se pueden modificar respecto de su asignación estándar.

Software Lexium 23

Schneider Electric brinda programas de interacción con sus dispositivos. Para el presente proyecto se utilizó el software "*Lexium 23 CT V1.03.14*". Este programa permite la comunicación con el servocontrolador, permite ver y modificar/forzar las alarmas, las entradas y salidas, el estado de los parámetros de software y físicos (corriente, velocidad, temperatura, etc.). La guía para su instalación, configuración y uso puede verse en el informe "*Guía Paso a Paso - Plataformas compactas para robótica y teleoperación*" de Federico Fabiancic [9].

5. Desarrollo de la aplicación

En el presente trabajo se desarrolla un nodo Maestro (microcontrolador *STM32F407*) para el control de tres ejes simultáneos Esclavos (3 conjuntos servocontrolador – servomotor de *Festo* [8]), los cuales son las tres articulaciones actuadas del manipulador paralelo en configuración Delta Lineal (*Keops*) [4] [5]. Para esto se planteó el sistema de la Figura 24, con un microcontrolador STM32F407VG [10] comunicado mediante dos interfaces series asíncronas *UART* (*USART3* y *UART5*) con una computadora de control, mediante conversores *RS232-TTL*. La *USART3* se utiliza para comunicarse con la aplicación *CANopenMaster.vi* realizada en Labview o, de manera alternativa, para enviar comandos desde una terminal virtual hacia el microcontrolador y para observar las tramas de recepción (respuestas de servocontroladores). Por otro lado, la *UART5* se utiliza para observar las tramas de transmisión (enviadas hacia los servocontroladores). Se cuenta además con un adaptador a *CANBUS* mediante el chip *MCP2551*.

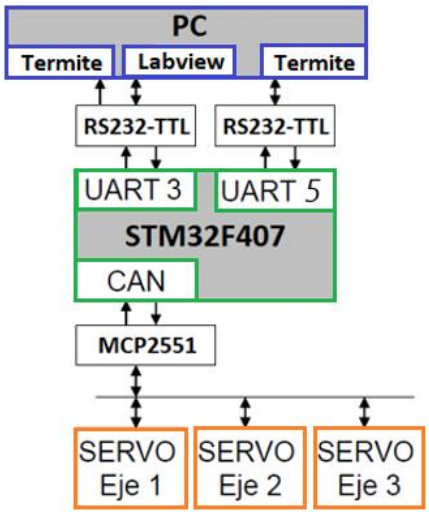

Figura 24. *Sistema* CANopen *implementado para control de 3 ejes.*

El integrado *MCP2551* funciona como transceptor *CAN*, llevando a cabo la conversión entre las señales *CANTx* y *CANRx* del controlador *CAN*, al par diferencial bidireccional del bus (*CANH* y *CANL*).

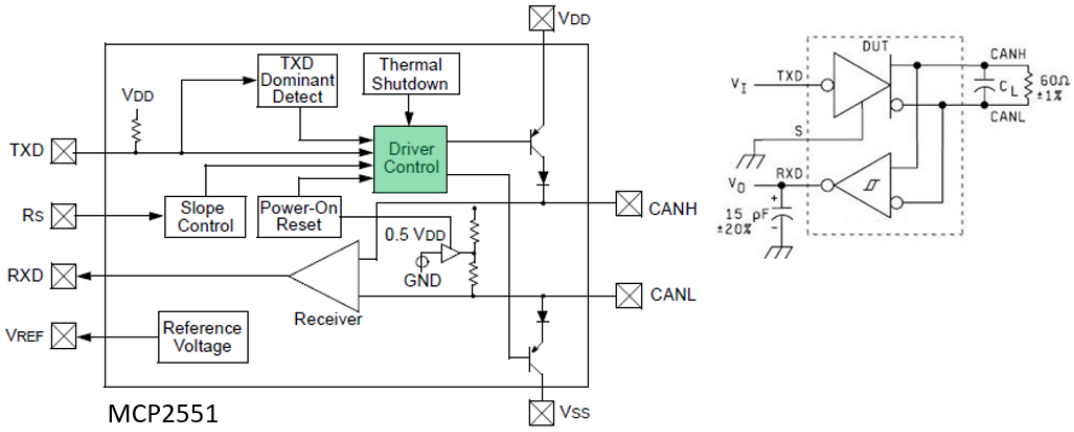

Figura 25. *MCP2551 como transceptor* CAN.

En esta aplicación se hace que los tres ejes completen cada etapa antes de pasar a la siguiente, para evitar configuraciones que puedan dañar al mecanismo. Por ejemplo, se lleva a los tres ejes al modo *Homing* con simultaneidad y una vez referenciados todos ellos, se los lleva al modo de operación. Por ejemplo, una vez alcanzado el modo *IP* en los tres ejes, el nodo Maestro procede a enviar órdenes de movimiento en el espacio de la tarea, con trayectorias analíticas (rectas en el espacio, arcos de circunferencia, etc.).

5.1. Nodo maestro

Se elige como nodo Maestro al microcontrolador *STM32F407VG* [10] (*ARM Cortex-M4*) en su placa *Discovery* [11]. El mismo deberá cumplir todas las especificaciones descriptas en la sección 2.3 como nodo Maestro *CANopen*. Se utilizan los siguientes recursos del microcontrolador:

GPIO: Para señales generales de E/S.

- *PD12, 13, 14 y 15* (leds integrados en la placa Discovery)
- *PA0* (pulsador de usuario)

USART3: Para recibir comandos de control desde la aplicación *MasterCANopen.vi* desarrollada en *Labview* o, alternativamente, para recibir comandos por parte del usuario y observar las tramas de recepción (enviadas por los servocontroladores). En ambos modos, con la información recibida (del usuario o de la aplicación) se arman las tramas *CANopen* para enviar a los servocontroladores. Configuración: 115200 bps, 8 bits de datos, 1 bit de STOP, sin paridad. Se habilita interrupción por recepción de datos serie.

- *USART3Tx*: PC11
- *USART3Rx*: PC10

Para lograr la comunicación con la computadora desde la cual se reciben los comandos, se utiliza un conversor *RS232-TTL*.

UART5: Para desplegar información en una terminal virtual en la computadora del usuario. Útil para verificar tramas *CANOpen* y el funcionamiento del programa en general. Configuración: 230400 bps, 8 bits de datos, 1 bit de STOP, sin paridad. No se habilita interrupción por recepción de datos serie, ya que solo se transmite.

- *UART5Tx*: PC12
- *UART5Rx*: PD2

Para lograr la comunicación con la computadora donde se despliega la información, se utiliza un segundo conversor *RS232-TTL*.

CAN1: *CAN Master* para comunicarse con los servocontroladores por *CANopen*. Velocidad inicial 250 kbps, luego llevada a 1 Mbps.

- *CANTx*: PB9
- *CANRx*: PB8

Para lograr la comunicación con los servocontroladores de la aplicación, se utiliza el integrado *MCP2551*, el cual transforma bidireccionalmente entre señales *CANTx* y *CANRx* del microcontrolador, y las señales físicas *CANH* y *CANL* según protocolo *CAN*.

TIMER2: Reservado para enviar la trama de sincronismo *CANopen*. Determina la grilla de tiempo, es decir, el intervalo entre puntos de consignas de posición enviadas a los servocontroladores. Inicialmente está en 10 ms, luego se lleva a 1 ms.

RTOS: Para dividir la aplicación.

- *mainTask*: coordina la inicialización de los ejes, gestiona los comandos.
- *taskEje1-2-3*: lleva máquina de estados de ejes 1-2-3.
- *taskInterpolador*: genera los puntos intermedios de la trayectoria X, Y y Z, y el cálculo de la cinemática inversa para obtener las consignas para cada eje.

En cuanto al funcionamiento del algoritmo programado en el microcontrolador, el sistema puede funcionar en dos modos según el tipo de mensaje que recibe por la *UART3*.

- ***MODO PUENTE***

Creado inicialmente para desarrollar el diccionario *CANopen* y depurar mensajes mediante la aplicación *MasterCANopen.vi* desarrollada en *Labview*. Estos mensajes comienzan con "**=**" y terminan con "***X***".

Los mensajes *CANopen* que se reciben de los servocontroladores mientras se está en este modo (si el último mensaje de *UART3* fue iniciado con "**=**") se los formatea y se los transmite por la *UART3* para evaluar desde la aplicación *MasterCANopen.vi*.

- ***MODO CONTROLADOR***

Es el modo en el que va a trabajar finalmente, cuando los mensajes *CANopen* ya están depurados. Estos mensajes comienzan con "***:***" y terminan con "***\r***" (ASCII 0D).

Los mensajes *CANopen* que se reciben de los servocontroladores mientras se está en este modo (si el último mensaje de *UART3* fue iniciado con "***:***") se los procesa para supervisar y controlar los servomotores.

5.2. Nodos esclavos

Los tres nodos Esclavos del mecanismo son tres conjuntos servocontrolador - servomotor de *Festo* [6]. Sin embargo, las primeras pruebas de funcionamiento se realizaron en un eje desacoplado de la estructura del robot, con el conjunto servocontrolador - servomotor de *Schneider Electric*, descripto en la sección 4. Se experimentan los distintos modos de trabajo del servocontrolador, incluido el modo *CANopen*, desde el cual se accede a los distintos modos operativos descriptos en la sección 2.1.3. Los resultados obtenidos se muestran en la siguiente sección.

6. Ejecución de los distintos modos de trabajo

En esta sección se mostrará el procedimiento seguido para experimentar los distintos modos de trabajo del servocontrolador, así como también los distintos modos operativos *CANopen*. En primer lugar alimentamos el sistema según lo descripto en la sección 4, es decir, 220V – 50Hz con conexión monofásica en las entradas ***L1*** y ***L2*** y en paralelo con las entradas ***R*** y ***S***, además de la conexión a tierra (***Ground***).

Encendemos el servocontrolador y observamos los mensajes de error que se despliegan en la pantalla de la HMI (alarmas *AL013*, *AL014* y *AL015*). Éstas se deben a la configuración por defecto de las I/O digitales del conector CN1, las cuales inicialmente no están conectadas. Una opción para evitar las alarmas es deshabilitar las funciones de todas las I/O momentáneamente, escribiendo un 0 en los parámetros P2-15 a P2-17 y P2-36 a P2-41 (ver pág. 138 del manual de usuario para más detalles) [8].

$$\boldsymbol{P2-15/17=0000}$$

$$\boldsymbol{P2-36/41=0000}$$

Otra forma es cambiar la configuración por defecto de los pines que provocan las alarmas. En específico, las entradas DI6, DI7 y DI8 están configuradas como normales cerradas, y se pueden cambiar a normales abiertas modificando los parámetros P2-15, P2-16 y P2-17 cambiando 0 por 1 el tercer dígito de derecha a izquierda en cada uno de ellos (ver pág. 334-336 del manual de usuario para más detalles) [8].

$$\boldsymbol{P2-15=0122}$$

$$\boldsymbol{P2-16=0123}$$

$$\boldsymbol{P2-17=0121}$$

A continuación se verificaron las conexiones de cada conector y se siguieron las recomendaciones y precauciones descriptas en el manual de usuario [8].

6.1. JOG mode

Este modo es ideal como primera prueba de movimiento del motor. Se debe habilitar el motor desde la *HMI*, escribiendo el valor 1 al parámetro 30 del grupo 2, es decir:

$$\boldsymbol{P2-30=0001}$$

Nota: este parámetro vuelve a 0 en cada reseteo del servocontrolador.

A continuación escribimos el valor de la velocidad en rpm deseada en el parámetro 5 del grupo 4. Comenzamos con una velocidad de 100 rpm para el eje del motor.

$$\boldsymbol{P4-05=0100}$$

Al guardar y confirmar este valor, la pantalla muestra ***-JOG-*** y se puede mover al motor en ambas direcciones con las flechas ***Up*** y ***Down***, a la velocidad seteada.

6.2. Homing Mode

Este modo es requerido para referenciar el motor en cuanto a posicionamiento, para posteriores movimientos. A continuación se muestra la escritura de parámetros asociados a este modo de trabajo. Con la configuración elegida en cuanto a dirección de desplazamiento antes y después de encontrar referencia, escribimos:

$$\boldsymbol{P5-04=0126}$$

En este modo, el motor se mueve a una determinada velocidad (***P5-05***) en una dirección, se detiene cuando se levanta el flag ***ORGP*** y se mueve en la dirección contraria a una velocidad menor (***P5-06***) hasta que se baja el flag ***ORGP***. Este flag está asociado a la entrada ***DI5***. Para la configuración de este pin escribimos:

$$\boldsymbol{P2-14=0124}$$

6.3. Pr mode (Internal Position Control)

Este modo está compuesto por comandos de posición individuales o múltiples, disparados por la señal de entrada digital *CTRG*. Además, las entradas digitales *POS0*, *POS1* y *POS2* se utilizan para especificar la posición de disparo requerida.

La unidad de comando de posición de este modo se representa por *PUU* (pulso unitario de usuario). Esto también indica la relación entre la unidad de comando de posición de un maestro (externo) y la unidad de comando de posición interna del servocontrolador, es decir, la relación electrónica de engranaje del servocontrolador.

1) Unidad de comando de posición del servocontrolador (pulso): unidad de encoder, 1280000 *pulsos/rev*.
2) Unidad de usuario (*PPU*): unidad de maestro (externo). Si el número de pulsos por revolución es *P* pulsos (*PPU/rev*) entonces la relación electrónica de engranaje se debe setear a:

$$\frac{GEAR_{NUM}(P1-44)}{GEAR_{DEN}(P1-45)}=\frac{1280000}{P}$$

6.4. CANopen mode

Este es el modo en el que se trabajará definitivamente en la aplicación. Dentro del mismo se puede llevar al Dispositivo a distintos ***modos de operación*** a través del objeto de control.

Object 6060_h: Modes of operation

INDEX	6060_h
Name	Modes of operation
Object Code	VAR
Data Type	INTEGER8
Access	RW
PDO Mapping	Yes
Value Range	INTEGER8
Default Value	0
Comment	-1: Jog mode 0: Reserved 1: Profile position mode 3: Profile velocity mode 4: Profile torque mode 6: Homing mode 7: Interpolated Position mode

Figura 26. *Modos de operación dentro de modo de trabajo* CANopen *[12].*

Nota: Es importante diferenciar los modos de trabajo del servocontrolador de los modos de operación a los que se puede acceder y configurar desde el modo de trabajo *CANopen*. Los modos de operación de la Figura 26 son independientes de los modos de trabajo del servocontrolador. Por ejemplo, los parámetros asociados al modo de trabajo *Homing* vistos en la sección 6.2 no están relacionados al modo de operación *Homing* y no afectan a su comportamiento.

En primer lugar seteamos el parámetro P1-01 a 0Bh para pasar a modo *CANopen*.

$$\boldsymbol{P1-01=000b}$$

Además, seteamos P3-05 con la dirección de Dispositivo esclavo a cada servocontrolador (1 a 127, valor 0 por defecto) y P3-01 con la velocidad de baudrate del protocolo *CAN* (103h por defecto para 250kbps). Esta velocidad debe coincidir con la de todos los nodos que comparten el mismo bus *CAN*.

$$\boldsymbol{P3-05=NID}$$

$$\boldsymbol{P3-01=0100}$$

Nota: Se debe resetear al servocontrolador para hacer efectivos los cambios.

De las entradas digitales del servocontrolador a través de su conector CN1, en el modo *CANopen* éstas quedan por defecto como sigue:

- ***DI1*** a ***DI4*** → desactivadas.
- ***DI5*** → ***ORGP*** *(final de carrera)*, normal cerrado
- ***DI6*** → ***NL(CCWL)*** *(límite inferior) n*ormal cerrado
- ***DI7*** → ***PL(CCWL)*** *(límite superior)* normal cerrado
- ***DI8*** → ***OPTS*** *(Operational Stop)* normal cerrado

Debido a que se realizó una placa didáctica con pulsadores normales abiertos para acceder a las entradas digitales, se configuraron las entradas DI5 a DI8 como normales abiertas conservando sus funciones. Además, la entrada DI1 se configuró como ARST y normal abierta para limpiar las alarmas del sistema. Esto último fue necesario principalmente debido a que el servocontrolador arroja una alarma (AL401) cada vez que se resetea al mismo estando operativo. En tal situación, sin una entrada configurada como ARST, se debe apagar el sistema, esperar al menos un minuto (manual de usuario Lexium [11]) y volver a encender. Esta entrada se utilizó con tanta frecuencia que se decidió activarla automáticamente desde el microcontrolador, a través de un relé, cada vez que se resetea el mismo. Los parámetros escritos para lograr esto, junto con las funciones asociadas, se muestran a continuación:

- ***DI1*** → $\boldsymbol{P2-10=0102}$ → ***ARST*** *(resetea alarmas)*
- ***DI2*** → $\boldsymbol{P2-11=0100}$ → desactivada
- ***DI3*** → $\boldsymbol{P2-12=0100}$ → desactivada
- ***DI4*** → $\boldsymbol{P2-13=0100}$ → desactivada
- ***DI5*** → $\boldsymbol{P2-14=0124}$ → ***ORGP*** *(final de carrera)*, normal abierto
- ***DI6*** → $\boldsymbol{P2-15=0122}$ → ***NL(CCWL)*** *(límite inferior)* normal abierto
- ***DI7*** → $\boldsymbol{P2-16=0123}$ → ***PL(CCWL)*** *(límite superior)* normal abierto
- ***DI8*** → $\boldsymbol{P2-17=0121}$ → ***OPTS*** *(Operational Stop)* normal abierto

6.4.1. Inicialización del Dispositivo

Para inicializar el Dispositivo se deben activar las transiciones adecuadas en las máquinas de estado vistas en la Figura 6. Se muestra un ejemplo de envío de tramas con microcontrolador y respuesta del servocontrolador con identificador 2 para lograr llevar al Dispositivo al modo Operativo, con habilitación de TPDO1 y RPDO1. El formato adoptado es el siguiente:

$$:\boldsymbol{ID, longitud\ de\ datos, datos\ (hex)}$$

- Reset Aplication → NMT con función "Reset Node" (0x81) y NID

$$\boldsymbol{\rightarrow 0,2,81-02}$$

$$\boldsymbol{\leftarrow 702,1,00}\ \text{(si estaba reseteado)}$$

$$\boldsymbol{\leftarrow 282,3,000000}\ \text{(si no estaba reseteado)}$$

- Reset Communication → NMT con función "Reset Communication" (0x82) y NID

$$\rightarrow \mathbf{0, 2, 82 - 02}$$

$$\leftarrow \mathbf{702, 1, 00}$$

- Pre-operative → NMT con función "Enter Pre-operational" (0x80) y NID

$$\rightarrow \mathbf{0, 2, 80 - 02}$$

Nota: para modificar un PDO, el estado de la comunicación debe ser Pre-operativo.

- Habilita RPDO1 → Mensaje SDO con dirección de RPDO1 (0x1400_01), identificador de RPDO1 (0x200+NID_00) y habilitación (0x40)

$$\rightarrow \mathbf{602, 8, 23 - 00 - 14 - 01 - 02 - 02 - 00 - 40}$$

$$\leftarrow \mathbf{582, 8, 60 - 00 - 14 - 01 - 00 - 00 - 00 - 00}$$

- Habilita TPDO1 → Mensaje SDO con dirección de TPDO1 (0x1800_01), identificador de TPDO1 (0x180+NID_00) y habilitación (0x40)

$$\rightarrow \mathbf{602, 8, 23 - 00 - 18 - 01 - 82 - 01 - 00 - 40}$$

$$\leftarrow \mathbf{582, 8, 60 - 00 - 18 - 01 - 00 - 00 - 00 - 00}$$

- Operative (Start Communication) → NMT con función "Enter Operational" (0x01) y NID.

$$\rightarrow \mathbf{0, 2, 01 - 02}$$

$\leftarrow \mathbf{182, 2, 40 - 02}$ (si pasó por Pre-operativo)

$\leftarrow \mathbf{702, 1, 40}$ (si no pasó por Pre-operativo)

Nota: para comunicarse con el Dispositivo, el estado de la comunicación debe ser Operativo.

- Ready (not ready to switch on) → SDO con Objeto de control (0x6040_00) y función Ready (0x0600)

$$\rightarrow \mathbf{602, 6, 2B - 40 - 60 - 00 - 06 - 00}$$

$$\leftarrow \mathbf{282, 3, 60 - 40 - 60}$$

$$\leftarrow \mathbf{582, 8, 21 - 02 - 00 - 00 - 00 - 00 - 00 - 00}$$

$$\leftarrow \mathbf{182, 2, 21 - 02}$$

- Switch on → SDO con Objeto de control (0x6040_00) y función Switch on (0x0700)

$$\rightarrow \mathbf{602, 6, 2B - 40 - 60 - 00 - 07 - 00}$$

$$\leftarrow \mathbf{282, 3, 60 - 40 - 60}$$

$$\leftarrow \mathbf{582, 8, 23 - 02 - 00 - 00 - 00 - 00 - 00 - 00}$$

$$\leftarrow \mathbf{182, 2, 23 - 02}$$

- Operational Enable → SDO con Objeto de control (0x6040_00) y función Operational enable (0x0F00)

$$\rightarrow 602, 6, 2B - 40 - 60 - 00 - 0F - 00$$

$$\leftarrow 282, 3, 60 - 40 - 60$$

$$\leftarrow 582, 8, 23 - 06 - 00 - 00 - 00 - 00 - 00 - 00$$

$$\leftarrow 182, 2, 23 - 06$$

$$\leftarrow 182, 3, 37 - 06 - 00$$

$$\leftarrow 182, 2, 37 - 06$$

Nota: en este estado el Dispositivo se encuentra bloqueado y listo para recibir instrucciones de movimiento.

6.4.2. Modo de operación Homing

- Modo Homing → PDO2 con palabra de control (0x000F) y función Homing (0x06)

$$\rightarrow 302, 3, 0F - 00 - 06$$

$$\leftarrow 282, 3, 37 - 02 - 00$$

$$\leftarrow 182, 2, 37 - 02$$

$$\leftarrow 282, 3, 37 - 02 - 06$$

- Ejecuta → PDO1 con palabra de control (0x001F)

$$\rightarrow 202, 2, 1F - 00$$

$$\leftarrow 282, 3, 37 - 92 - 06$$

$$\leftarrow 182, 2, 37 - 92$$

6.4.3. Modo de operación Profile Position

- Modo Profile Position → PDO2 con palabra de control (0x000F) y función PP (0x01)

$$\rightarrow 302, 3, 0F - 00 - 01$$

$$\leftarrow 182, 3, 37 - 82 - 00$$

$$\leftarrow 182, 2, 37 - 82$$

$$\leftarrow 282, 3, 37 - 82 - 01$$

$$\leftarrow 282, 3, 37 - 86 - 00$$

$$\leftarrow 282, 2, 37 - 86$$

- Ejecuta → PDO1 con palabra de control (0x001F)

$$\rightarrow 202, 2, 1F - 00$$

Cuando comienza el movimiento:

$$\leftarrow 282, 3, 37 - 92 - 01$$

$$\leftarrow 182, 2, 37 - 92$$

Cuando llega a posición final:

$$\leftarrow 282, 3, 37 - 96 - 00$$

$$\leftarrow 182, 2, 37 - 96$$

- Nueva consigna → PDO3 con palabra de control (0x000F) y consigna (ejemplo 0x00504030

$$\rightarrow 402, 6, 0F - 00 - 30 - 40 - 50 - 00$$

$$\leftarrow 182, 2, 37 - 86$$

$$\leftarrow 182, 2, 37 - 86$$

- Ejecuta → PDO1 con palabra de control (0x001F)

$$\rightarrow 202, 2, 1F - 00$$

Cuando comienza el movimiento:

$$\leftarrow 282, 3, 37 - 92 - 01$$

$$\leftarrow 282, 3, 37 - 92 - 00$$

Cuando llega a posición final:

$$\leftarrow 182, 2, 37 - 96$$

$$\leftarrow 282, 3, 37 - 96 - 00$$

- Nueva consigna → PDO3 con palabra de control (0x000F) y consigna (ejemplo 0x00102030

$$\rightarrow 402, 6, 0F - 00 - 30 - 20 - 10 - 00$$

$$\leftarrow 182, 2, 37 - 86$$

$$\leftarrow 182, 2, 37 - 86$$

- Ejecuta → PDO1 con palabra de control (0x001F)

$$\rightarrow 202, 2, 1F - 00$$

Cuando comienza el movimiento:

$$\leftarrow \mathbf{282,3,37-92-01}$$

$$\leftarrow \mathbf{282,3,37-92-00}$$

Cuando llega a posición final:

$$\leftarrow \mathbf{182,2,37-96}$$

$$\leftarrow \mathbf{282,3,37-96-00}$$

6.4.4. Modo de operación Interpolated Position

Cada servocontrolador trabajando en modo *IP* asume una grilla de tiempo especificada (objetos *60C2_01h* y *60C2_02h*) e intenta cumplir cada nueva consigna de posición en ese tiempo. La velocidad queda implícita como el cociente entre la diferencia de posición a cubrir y dicho intervalo de tiempo, por lo que no existen más restricciones de velocidad ni aceleración. Los perfiles de aceleración/desaceleración y velocidad máxima deben ser elaborados íntegramente por el *DSP* coordinador (o microcontrolador Maestro). En el modo *IP*, las consignas de posición a cada eje corresponden al objeto *60C1h*, y se transmiten en la *PDO4* (la cual fue modificada respecto del valor por defecto visto en la Tabla 1). Queda para trabajos futuros experimentar este modo de operación.

Bibliografía

[1] R. Boys, "CAN Primer: Creating Your Own Network".

[2] E. Iriarte, "Microcontroladores y Electrónica de Potencia," [Online]. Available: http://fing.uncu.edu.ar/catedras/microcontroladores-y-electronica-de-potencia.

[3] C. i. Automation, "CiA Draft Standard Proposal 402 - CANopen - Device Profile Drives and Motion Control," 2002.

[4] E. Iriarte and M. Distéfano, "Control de ejes mediante protocolo CANopen. XXII° Congreso Argentino de Control Automático, AADECA 2010," 2010.

[5] E. Iriarte and P. Noguera, "Controlador robótico por protocolo CANopen," 2012.

[6] Festo, "Servo Motors EMMS-AS," 2009.

[7] Festo, "Motorcontroller CMMP - Mounting and installation Type CMMP-AS-11A," 2008.

[8] S. Electric, "LXM23A and BCH Servo drive system - Product manual," 2014.

[9] F. Fabiancic, "Plataformas Compactas para robótica y teleoperación," 2017.

[10] STMicroelectronics, "RM0090 Reference Manual - STM32F4xx advanced Arm-based 32-bit MCUs".

[11] STMicroelectronics, "UM1472 User Manual - Discovery kit for STM32F407/417 lines," 2014.

[12] S. Electric, "LXM23A CANopen - Fieldbus protocol for servo drive - Fieldbus manual," 2011.

[13] "Petrolhead Garage," [Online]. Available: https://petrolheadgarage.com/Posts/caracteristicas-de-un-sistema-can-bus/.

[14] C. i. Automation, "CiA Draft Standard Proposal 301 - CANopen - Application layer and communication profile," 2002.

[15] W. Gay, "Begining STM32 - Developing with FreeRTOS, libopencm3 and GCC," 2018.

[16] E. Jordán and E. Iriarte, "Programación de microcontroladores PIC, AVR y ARM," 2019.