ESCOLA TÈCNICA SUPERIOR ENGINYERIA
UNIVERSITAT ROVIRA I VIRGILI
Departament d'Enginyeria Electrònica Elèctrica i Automàtica

Implementació d'un protocol CANopen sobre microcontrolador ARM

TITULACIÓ: Enginyeria Tècnica Industrial en Electrònica Industrial

AUTOR: Patrick Vögeli Olivera
DIRECTOR: Dr. Ernest Gil Dolcet

DATA: juny de 2010

1 Index

1.1 Índex general

1 Index

1.2 Índex de figures

1.3 Índex de taules

2 Introducció

2.1 Antecedents

La oportunitat de desenvolupar un protocol CANopen comença quan el Dr. Ernest Gil ofereix un Projecte Final de Carrera en el qual es proposa estudiar i implementar un protocol CANopen Esclau.

El Dr. Gil ja disposava documentació bàsica del protocol per a dur a terme el projecte. Després de vàries reunions, es veu que el projecte es viable i que es disposa del material necessari per començar a treballar.

La idea inicial era adaptar el codi disponible per al PIC a la plataforma ARM, una plataforma molt més potent i capaç que el PIC, si bé es possible que menys econòmica. Aquesta idea però, es descarta després d'estudiar el codi font disponible per a PIC. La solució per a PIC sembla complicada, incompleta i poc modular, a més de diferències bastant grans en el funcionament del hardware. Així, es començar a desenvolupar, des de zero, la implementació de CANopen per als microcontroladors LPC2xxx, treballant sobre un controlador LPC2292. Aquesta feina inclou programar un driver de CAN i, sobre aquest, el protocol CANopen. A més, també s'ha hagut de crear un sistema de timers.

2.2 Objectius

Inicialment, l'objectiu del projecte era portar el codi font disponible per a PIC a la plataforma LPC Arm, encara que aquesta idea es va descartar ràpidament. Es va decidir que la millor solució seria desenvolupar el software necessari per a tenir un protocol CANopen funcional i conforme a l'estàndard partint de zero.

El més important es que, un cop acabat, el projecte pogués interactuar correctament amb solucions CANopen comercials i, sobretot, tingués certa fiabilitat. Caldria, doncs, implementar tots els serveis necessaris per tal de que el node resultant fos capaç d'interactuar correctament amb els elements comercials dels quals es disposa.

Evidentment, per a poder dur a terme el projecte, era necessari un estudi exhaustiu del protocol CANopen: modes de funcionament, particularitats, etc.

3 Estat de l'art

3.1 CAN

CAN és un protocol de comunicacions de baix nivell, que utilitza una tipologia en bus per a la transmissió de missatges. És un protocol utilitzat majoritàriament en sistemes de control distribuït i automatitzacions amb molts nodes. Inicialment, es va desenvolupar pensant en aplicacions a la indústria de l'automoció.

És un protocol basat en missatges, on la informació és descompon en diversos missatges i s'envien en trames a les quals s'assigna un identificador, mitjançant el qual els altres nodes de la xarxa decideixen si accepten (consumeixen) el missatge o no. Aquest missatges es distribueixen segons el model productor / consumidor, sistema en el qual hi ha un node que envia el missatge (productor) i cap, un o més consumidors.

Algunes característiques del protocol CAN:

- Prioritat en els missatges (a través de l'identificador: quant més baix, més prioritari)
- Flexibilitat en el disseny i configuració de la xarxa
- Enviament de missatges “multicast”
- Detecció i senyalització d'errors
- Sistema robust de detecció d'errors
- Retransmissió automàtica de trames errònies

És un protocol de comunicacions de transferència en serie, que suporta sistemes de control distribuït en temps real i amb un nivell alt de seguretat i multiplexatge.

El sistema, seguint el model de referència OSI per a sistemes de comunicacions, està format per 3 capes:

3.1.1 Capa física

La capa física especifica els aspectes del medi físic per a dur a terme la transmissió de dades entre els nodes d'una xarxa CAN, a nivell de senyal, representació, sincronització i temps en els quals els bits es transmeten cap al bus. La especificació no defineix una capa física, però els estàndards ISO 11898 estableixen les característiques que s'han de complir per dur a terme la transferència en alta i baixa velocitat.

3.1.2 Capa d'enllaç de dades

En la capa d'enllaç es defineixen les tasques a dur a terme per accedir al medi. A més, degut a que CAN preveu el suport per a la transmissió en temps real, l'intercanvi de missatges requereix d'una transmissió ràpida i amb el mínim retràs possible. Al ser un sistema on cada node pot accedir al medi per transmetre i, per tant, té dret d'accés i control de la xarxa, la tècnica d'accés al medi és extremadament important. Així doncs, la capa d'enllaç defineix els mètodes d'accés i els tipus de trames per a l'enviament de missatges. El mètode que utilitza CAN per accedir al medi és el d'Accés Múltiple mitjançant Detecció de Portadora, amb Detecció de Col·lisions i Arbitratge per Prioritat de Missatge (CSMA/CD+AMP, en les seves sigles en anglès). Aquest mètode especifica que, quan un node vol enviar un missatge, ha d'esperar a que el bus estigui buit (detecció de portadora) i, quan això passa, els nodes envien un bit d'inici. Després, cada node llegeix bit a bit el bus durant la

transmissió de la trama i comparen el valor rebut: mentre el valor llegit és l'esperat, es segueix amb la transmissió; si el valor és diferent, s'inicia el mètode d'arbitratge, mitjançant el qual es decideix quin dels nodes tindrà accés i pot tornar a enviar la trama.

3.1.3 Capa d'aplicació

Hi ha diversos estàndards que defineixen la capa d'aplicació. El que es fa en aquesta capa es encapsular d'una determinada manera els missatges en una trama de CAN. Entre aquests, els més coneguts son CANopen, DeviceNet i SDS.

El protocol CAN estableix 2 formats de trama diferents: la primera, CAN 2.0A, estableix un identificador d'11 bits; a CAN 2.0B, definida posteriorment, l'identificador creix fins als 29 bits.

En una xarxa CAN, cada node té el seu propi 'clock', el qual no es transmès durant la transmissió de dades. La sincronització es fa dividint cada bit de la trama en 4 parts: Sincronització, Propagació, Fase 1 i Fase 2:

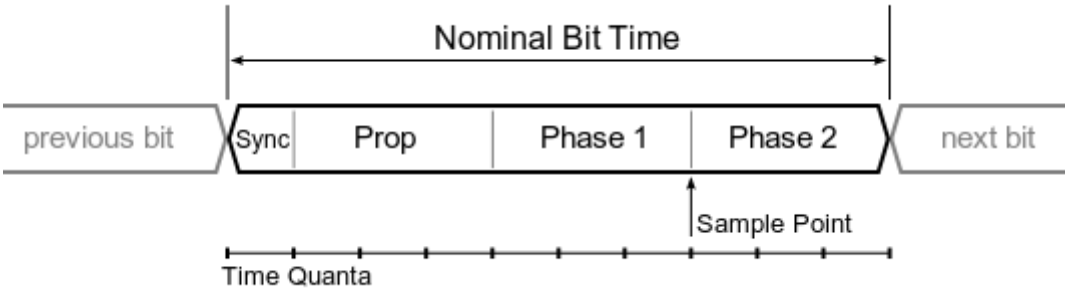

Fig. 1: Format d'un bit en una trama CAN

El punt de mostreig es situa entre la Fase 1 i la Fase 2, el que facilita la sincronització continua, que permet que el receptor sigui capaç de llegir el missatge rebut correctament.

Format d'una trama CAN completa:

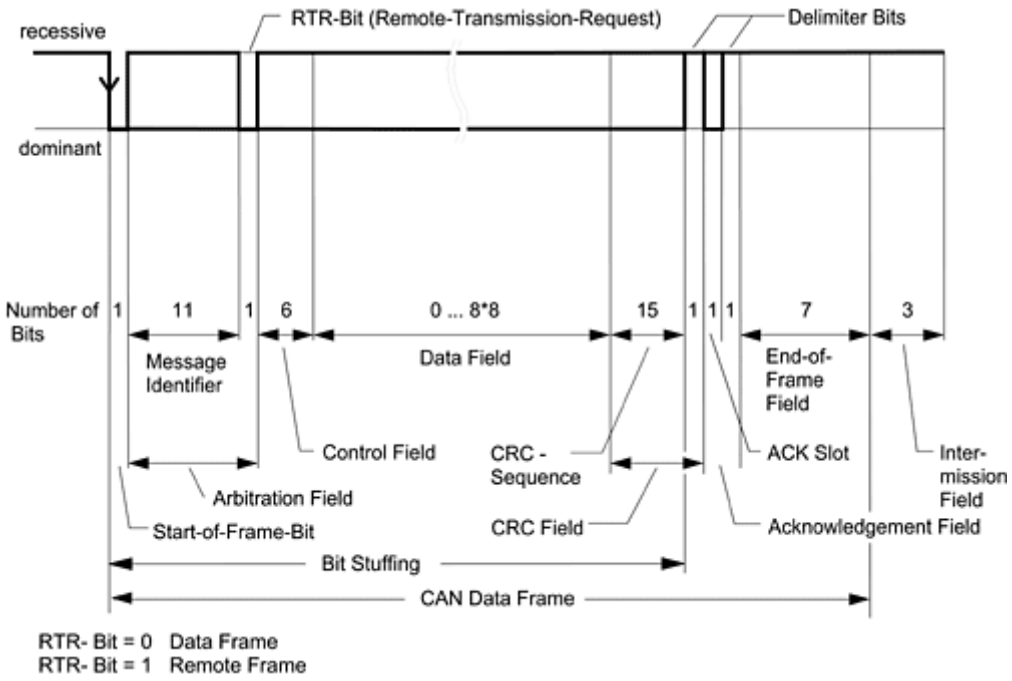

Fig. 2: Format d'una trama CAN

El protocol CAN esta especificat completament en diferents estàndards ISO:

- ISO 11898-1: CAN Data Link Layer and Physical Signalling
- ISO 11898-2: CAN High-Speed Medium Access Unit
- ISO 11898-3: CAN Low-Speed, Fault-Tolerant, Medium-Dependent Interface
- ISO 11898-4: CAN Time-Triggered Communication
- ISO 11898-5: CAN High-Speed Medium Access Unit with Low-Power Mode
- ISO 11992-1: CAN fault-tolerant for truck/trailer communication
- ISO 11783-2: 250 kbit/s, Agricultural Standard

3.2 CANopen

CANopen és un dels protocols d'aplicació sobre CAN més conegut i utilitzat, juntament amb DeviceNet.

Un dels problemes del protocol CAN és que, degut al seu disseny i especificacions, no resulta apte per a xarxes on es necessita una comunicació en temps real, ja que no hi ha establerta una forma de comunicació estandarditzada, sinó que només defineix la part física. CANopen implementa sobre CAN un sistema de comunicació molt robust, amb diversos tipus de comunicacions possibles i amb els mecanismes necessaris per obtenir informació fiable i en temps real.

3.2.1 Propietats de CANopen

- Transmissió de dades de procés crítiques seguint el principi de consumidor – productor
- Descripció dels dispositius estandarditzada (dades, paràmetres, programes, funcions) sota el diccionari d'objectes. L'accés a qualsevol objecte del dispositiu es realitza utilitzant un protocol estàndard i utilitzant comunicació mestre – esclau (peer to peer).
- Serveis estandarditzats per a la monitorització de nodes (node guarding / heartbeat), indicació d'errors ('emergency') i coordinació i control de la xarxa ('network management').
- Serveis estandarditzats per operacions síncrones (missatges Sync), amb objecte d'indicació de temps central (time stamp)
- Funcions d'ajuda per a la configuració de la velocitat de funcionament del bus i el número de node a través del mateix bus de comunicacions.
- Assignació estandarditzada d'identificadors de missatges per a xarxes simples ('predefined connection set').

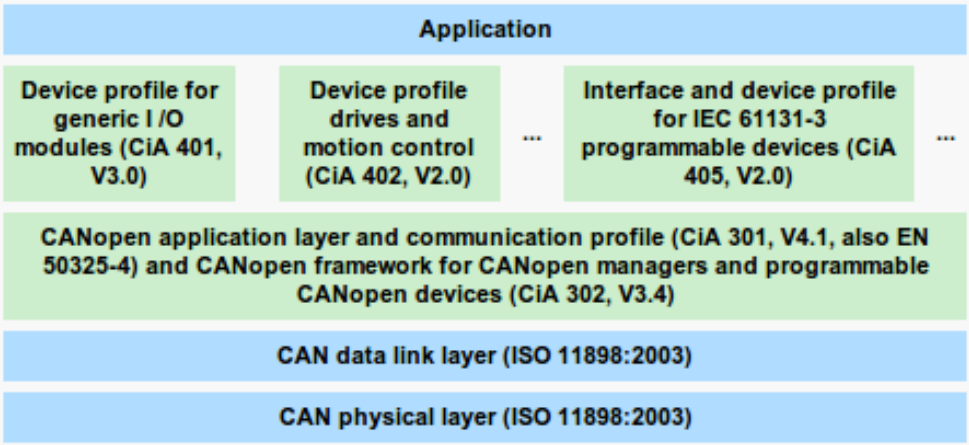

Fig. 3: Capes d'un node CANopen

Les especificacions bàsiques de CANopen estan recollides en el document CiA DS-301 "CANopen Application Layer and Communication Profile".

3.2.2 Estructura física d'una xarxa CANopen

CANopen utilitza una tipologia en forma de bus, en el qual, per evitar la reflexió de les senyals, ambdós extrems del bus estan tancats mitjançant una resistència de terminació.

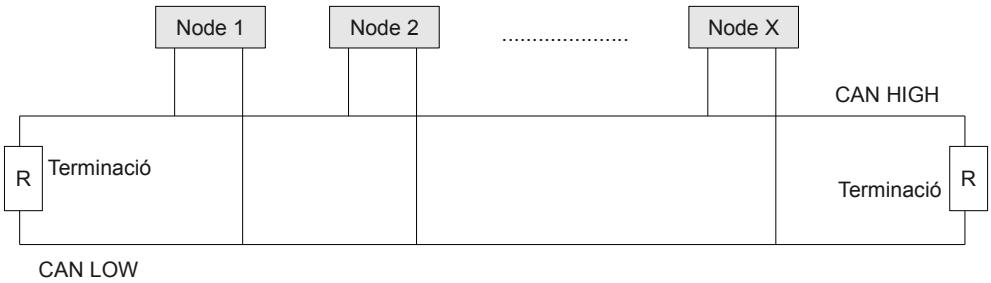

Fig. 4: Organització d'una xarxa CANopen

Les velocitats de funcionament del bus estàndards i recomanades per l'organització Can in Automation (CiA, a partir d'ara) son les següents: 10 kbps, 20 kbps, 50 kbps, 125 kbps, 250 kbps, 500 kbps, 800 kbps i 1000 kbps.

A més, hi ha 2 requisits a l'hora de configurar una xarxa CANopen:

- Tots els nodes han de funcionar a la mateixa velocitat de transmissió
- Cada node ha de tenir un identificador únic, entre 1 i 127

El compliment d'aquestes condicions és responsabilitat del creador de la xarxa, ja que el protocol no estableix cap tipus de mètode que automatitzi el procés. El més normal es que aquests valors es configurin a través d'interruptors situats en el dispositiu, encara que també es pot fer a través del Servei de Configuració de Capa (Layer Setting Service, LSS), definit a la especificació CiA 305, encara que no es obligatori i normalment no s'utilitza.

3.2.3 El diccionari d'objectes

Una part fonamental d'una xarxa CANopen és el Diccionari d'Objectes dels dispositius que la formen. El diccionari d'objectes és una taula que conté la descripció de cada dispositiu, de les seves variables y del que es capaç de fer. Per accedir al diccionari d'objectes des de l'exterior s'utilitza un sistema de direccionament lògic, on cada objecte té un index i, a més, aquest pot estar compost de varies variables, a les quals s'accedeix a través del subindex.

El document CiA 301 conté la descripció de les entrades que descriuen les propietats de comunicació d'un dispositiu, però, a part, hi ha documents que especifiquen "perfils de dispositius", que descriuen de manera estandarditzada diversos paràmetres d'un node segons la seva funcionalitat (motors, entrades / sortides, etc). Amb això es persegueix la estandardització de dispositius amb funcionalitats similars i de fabricants diversos, fet que facilita el canvi d'un dispositiu per un altre, sense haver de reconfigurar la xarxa completament.

El Diccionari d'Objectes presenta la següent estructura, segons la funcionalitat que es descriu en les entrades corresponents:

Índex	**Objecte**
0	No s'utilitza
1h – 1Fh	Tipus de dades estàtiques
20h – 3Fh	Tipus de dades complexes
40h – 5Fh	Tipus de dades complexes específiques del fabricant
60h – 7Fh	Tipus de dades estàtiques específiques del perfil de dispositiu
80h – 9Fh	Tipus de dades complexes específiques del perfil de dispositiu
A0h – FFFh	Reservats per a usos futurs
1000h – 1FFFh	Àrea del perfil de comunicacions
2000h – 5FFFh	Àrea del perfil específic del fabricant
6000h – 9FFFh	Àrea del perfil estandarditzat de dispositiu
A000h – FFFFh	Reservat per a usuari

Taula 1: Contingut del Diccionari d'Objectes

El Diccionari d'Objectes presentar la següent capçalera:

Índex	Objecte (nom simbolic)	Nom	Tipus	Atributs	Obligatori / Opcional

Taula 2: Capçalera del Diccionari d'Objectes

El subíndex no en forma part, ja que es fa servir per accedir als camps de dades d'un objecte de tipus complex, com un array o un record.

La columna d'objecte conté un nom curt que descriu quin tipus de dades hi ha a l'entrada corresponent. El nom, per la seva part, és una descripció segons la funcionalitat. A "Tipus" hi trobem el tipus de dades (UNSIGNED 8, REAL 32, etc). A la columna "Atributs" hi ha descrit el tipus d'accés que es permet: només lectura, només escriptura o lectura i escriptura. Per últim, també s'indica si la entrada es obligatòria o opcional.

Un Objecte del Diccionari pot tenir usos molt diferents: pot ser des d'una simple variable fins a codi de programa executable, passant per lectures de sensors o escriptures directes sobre un actuador.

En el Diccionari d'Objectes es pot veure la gran flexibilitat que té CANopen: molt poques entrades son obligatòries i es dona molta llibertat, cosa que permet adaptar fàcilment i quasi a la perfecció el desenvolupament d'un dispositiu per a una tasca concreta. A més, quantes menys entrades, més fàcil es entendre i/o controlar un node, a la vegada que n'augmenta la fiabilitat.

3.2.3.1 Fitxer EDS

El fitxer EDS (Electronic DataSheet, en les seves sigles en anglés) conté tota informació relativa al funcionament del node. En aquest fitxer s'hi descriuen algunes capacitats del node (velocitats de comunicació que suporta, etc) i, a més, inclou el Diccionari d'Objectes complert. El format del fitxer està descrit en el documents CiA DSP306.

Aquest fitxer, però, no substitueix de cap manera el manual del dispositiu, si bé ens pot donar una lleugera idea del seu funcionament.

Diversos fabricants ofereixen software encaminat a crear, editar i comprovar el fitxer EDS. És important que el fitxer EDS sigui correcte i estigui actualitzat.

3.2.4 Transferència de dades

La transferència de dades en una xarxa CANopen es pot fer, bàsicament, a través de 2 formes: els Objectes de Dades de Servei (Service Data Objectes, SDO a partir d'ara) i els Objectes de Dades de Procés (Process Data Objects, PDO).

Típicament, els missatges SDO es fan servir per a configurar el node i, un cop en mode Operacional, pel bus només hi circulen dades de procés via PDO i altres missatges de control.

3.2.4.1 SDO (Service Data Object)

Esta basat en una comunicació Client – Servidor (1 a 1, peer to peer), i permet accedir directament a una entrada del Diccionari d'Objectes a través del seu Index i Subindex. Normalment, l'accés a través d'SDO es fa servir per dur a terme la configuració inicial del dispositiu i escriure o llegir una quantitat considerable de dades i requereix protocol addicional.

La comunicació via SDO utilitza un mínim de 2 missatges, un de petició (el node que fa de client, típicament serà el Master de la xarxa) i un altre de confirmació (node servidor, un dels Esclaus). En cas d'algun error en la petició i/o resposta, tan el Client com el Servidor poden enviar un missatge d'error (per exemple, perquè l'objecte que es vol accedir no existeix); aquests missatges s'anomenen SDO Abort Transfer, contenen informació de l'error i els pot enviar qualsevol node en qualsevol moment.

La comunicació SDO utilitza un protocol propi, per tal d'identificar si és una petició, una resposta, una part d'una resposta, si es lectura / escriptura, etc. Podem distingir 5 tipus de trames SDO: Initiate SDO Download, Download SDO Segment, Initiate SDO Upload, Upload SDO Segment i Abort SDO transfer.

En una trama SDO, hi podem trobar els camps següents:

CAMP	DESCRIPCIÓ
CCS	Client Command Specifier. El Client indica al Servidor el tipus de trama que és.
SCS	Server Command Specifier. El Servidor indica al Client el tipus de trama a la qual respon.
X / Reserved	'0'

CAMP	DESCRIPCIÓ
n	Nombre de bytes al camp de dades que contenen dades invàlides
e	Transferència de dades segmentada (e=0) o no (e=1, la trama ja conté les dades)
s	Longitud de dades es coneguda (s=1) o no (s=0)
t	Bit de toogle. Canvia de valor en cada trama.
c	Control de segments: '0', s'esperen més segments; '1', no hi ha més segments.
m	Camp de 3 bytes que conté la informació sobre la entrada del Diccionari d'Objectes a la qual es vol accedir. El 2 bytes alts contenen l'index, i el byte baix el SubIndex
Data / Segment data	Dades que conté la trama

Taula 3: Paràmetres possibles en un missatge SDO

3.2.4.1.1 Download SDO

Quan el client vol escriure dades sobre una entrada del Diccionari d'Objectes del Servidor a través d'SDO, utilitzarà el protocol Download SDO. Per a escriure les dades, en funció de la mida, hi han dues maneres diferents, normal (segmentada) o expedida (una única trama).

Si el Client vol escriure 4 bytes o menys de dades, en tindrà prou amb enviar un missatge d'iniciació d'escriptura amb les dades que vol escriure. Aquesta forma de funcionament és la més comuna, sent bastant inusual que en una xarxa de CANopen es facin transferències per SDO d'objectes de més de 4 bytes. El següent gràfic il·lustra una escriptura expedida, amb els 2 missatges involucrats, una petició amb les dades a escriure i una resposta a mode de confirmació:

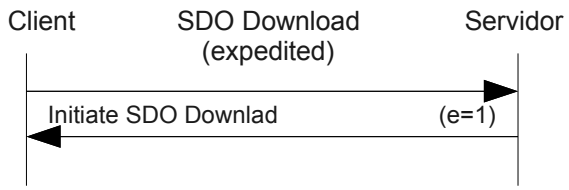

Fig. 5: Transferència SDO download de fins a 4 bytes de dades

En el cas que hi hagués més de 4 bytes de dades, el Client haurà de fer servir la transmissió segmentada: primer enviarà la trama d'iniciació, en la qual indicarà la mida total de dades a enviar i, a continuació, començarà a enviar les trames amb les dades segmentades:

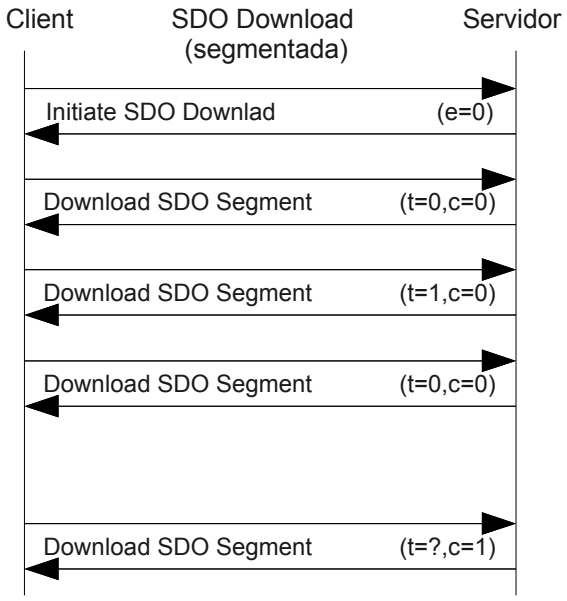

Fig. 6: Transferència SDO download de més de 4 bytes

3.2.4.1.2 Initiate SDO Download

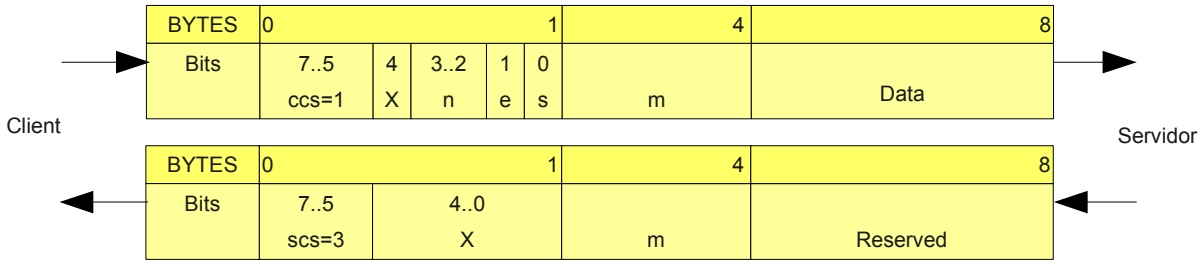

Fig. 7: Initiate SDO download

El client envia aquesta trama per transferir les dades o iniciar una transferència segmentada. Indica al Servidor sobre quina entrada vol escriure i aporta informació sobre la manera en que ho farà (camps n, e i s).

Si el Servidor indica que es farà una transferència segmentada (e = 0), el camp Data indicarà el nombre total de bytes que s'enviaran a continuació, utilitzant el mètode de transferència segmentada. Cas contrari, e = 1, aquest camp ja conté les dades a escriure a la entrada corresponent i, amb els camps s i n, es pot saber si s'indica la longitud de les dades i, si es així, quina es (longitud de les dades: 4 – n si s = 1)

3.2.4.1.3 Download SDO Segment:

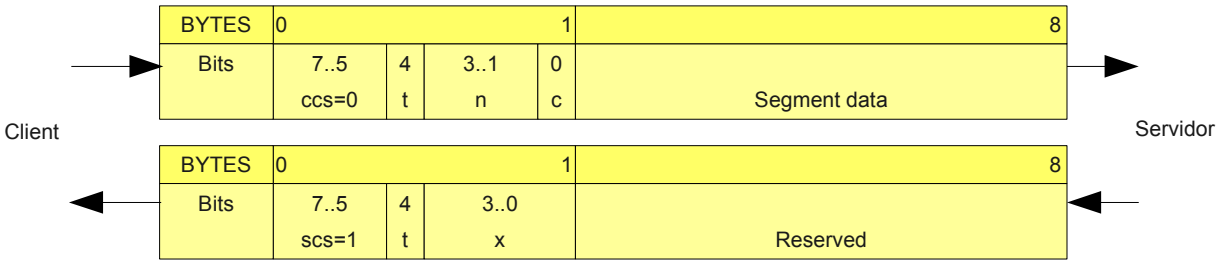

Fig. 8: Download SDO segment

Si el Client ha de fer una escriptura de més de 4 bytes, després de la trama d'iniciació, començarà a enviar trames que contenen fins a 7 bytes de dades. Cada cop que el Client envia una trama d'aquestes, ha d'alternar el bit de toogle (camp t) i ha d'indicar si és l'última trama (camp c). La resposta es una confirmació del Servidor i s'envia amb el bit de 'toogle' amb el mateix valor que en la trama que confirma.

La transferència de dades es fa en format 'little endian', s'envia primer el byte menys significatiu.

3.2.4.1.4 Upload SDO

Quan el servidor vol fer una lectura de dades del Client, fa servir el protocol Upload SDO. Els 2 protocols son básicament iguals entre si, ja que les trames son les mateixes que quan es fa un accés d'escriptura, peró invertides: la trama que abans era una resposta, en aquest és una petició.

Com en el cas de la escriptura, si hi han 4 bytes o menys de dades la comunicació només tindrà 2 missatges, una de petició de lectura i l'altra de confirmació i amb les dades solicitades:

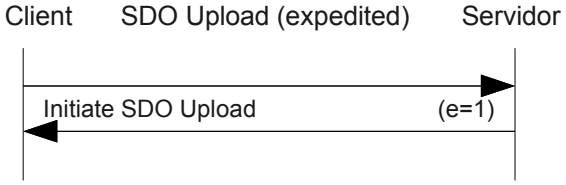

Fig. 9: Transferència SDO Upload de fins a 4 bytes de dades

Si la entrada que es vol llegir és de més de 4 Bytes, es necessitarà la trama d'inicialització i les trames de transmissió de dades, que al igual que en el cas d'escriptura, tenen els camps t i c per controlar les trames i indicar la última trama. En aquest cas, però, és el Client qui controla la indicació d'última trama (c=1) i de transferència segmentada (e=0):

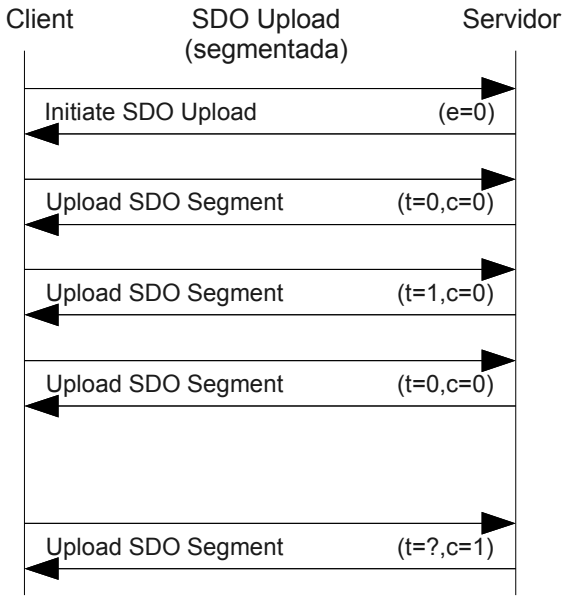

Fig. 10: *Transferència SDO upload de més de 4 bytes*

3.2.4.1.5 Initiate SDO Upload

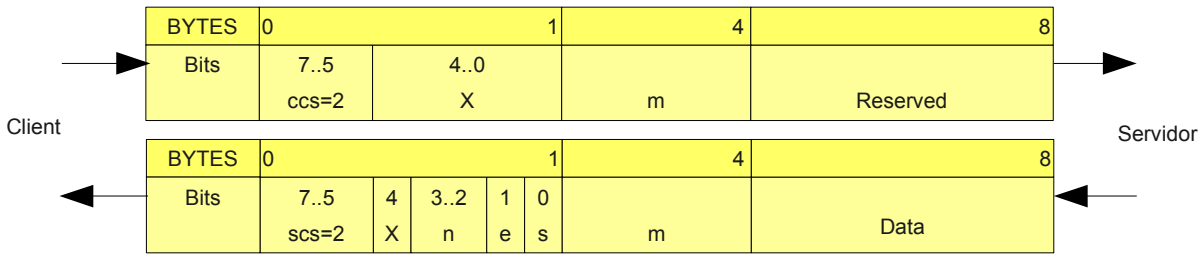

Fig. 11: Initiate SDO Download

En aquest cas, el Client només indica al Servidor quina entrada vol llegir (mitjançant el camp m). El Servidor pot contestar de 2 maneres: amb les dades si son 4 bytes o menys, o comunicant al Client que s'haurà de fer una transferència segmentada per a descarregar el nombre de bytes indicat al camp Data.

3.2.4.1.6 Upload SDO Segment

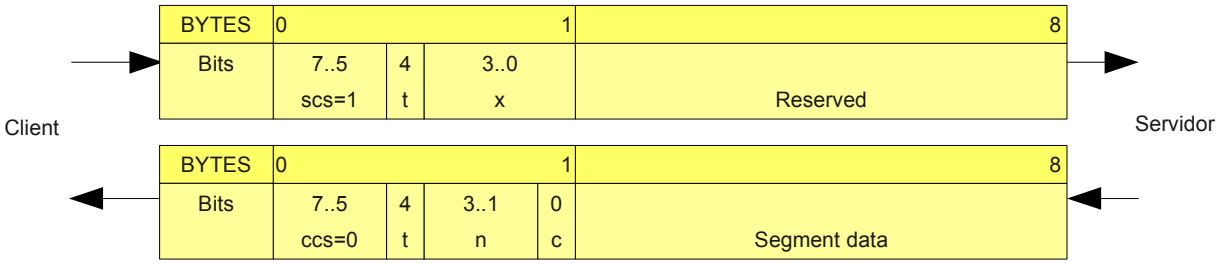

Fig. 12: Upload SDO segment

Si la transferència de dades ha de ser segmentada, el Client va enviant trames de petició de dades, alternant el bit de 'toogle', i el Client contesta amb una trama que inclou, igual que el cas de l'escriptura, fins a 7 bytes de dades i el mateix valor de 't' que s'ha rebut. A més, en aquest cas és el Servidor que hi ha d'indicar al Client si la trama en qüestió es l'última o si n'hi han més (c és '1' o '0', respectivament).

3.2.4.2 PDO (Process Data Object)

Esta basat en el mètode Productor – Consumidor, mitjançant el qual un node (el Productor) transmet una informació determinada i un o més o ningun node (Consumidor) processen les dades rebudes. En un PDO, la longitud de les dades esta limitada a 8 bytes, i pot contenir un o més objectes del Diccionari d'Objectes. No es requereix protocol addicional, encara que el consumidor ha de saber interpretar les dades que conté el PDO, ja que en la trama no hi ha res que identifiqui els diferents objectes, ni on comencen ni on acaben.

Com a Bus de camp que és, l'objectiu principal de CANopen es intercanviar dades de procés, i s'ha d'aconseguir fer d'una manera ràpida i senzilla. Típicament, després que una xarxa s'hagi inicialitzat correctament (a través dels SDO) i hagi entrat en mode d'operació normal, les dades de procés s'intercanvien utilitzant els PDO. Al ser dades de procés, aquestes han d'estar constantment actualitzades i, per tant, el número de PDO que circulen en el bus pot arribar a ser bastant alta. A més, cada dispositiu disposa, per defecte, de 4 PDO de sortida (TPDO) i 4 d'entrada (RPDO), tots configurables independentment dels altres i amb entre 1 i 64 entrades del Diccionari d'Objectes, que van transmetent de diferents maneres.

Els PDO només funcionen en el mode Operacional (si un node es troba en un estat diferent, no ha d'acceptar PDOs, ni els pot enviar), i no porta involucrat protocol adicional, sinó que el missatge son les dades directament (entre 1 i 8 bytes de dades).

Degut a l'absència de protocol, en les trames PDO no es poden identificar les dades que porta. Per tal d'identificar i entendre les dades que porta, cada dispositiu ha d'haver estat configurat prèviament per poder entendre el missatge PDO en concret. Tots els dispositius tenen, al seu Diccionari d'Objectes, un conjunt d'entrades específiques per a la configuració dels PDO, tant d'entrada com de sortida. Per a cada PDO hi han 2 tipus de configuració: els paràmetres de comunicació i els paràmetres del mapejat d'entrades del Diccionari. Cada una d'aquestes entrades té la següent forma:

Configuració del paràmetres de comunicació:

Índex	Subíndex	Descripció
0x1400 – 0x15FF (RPDO) 0x1800 – 0x19FF (TPDO)	0	Número de subíndex
	1	COB-ID
	2	Tipus de transmissió
	3	Inhibit time (no RPDO)
	4	Entrada de compatibilitat
	5	Timer de succés

Taula 4: Paràmetres de comunicacions d'un PDO

Configuració dels paràmetres del mapejat:

Índex	Subíndex	Descripció
0x1600 - 0x17FF (RPDO) 0x1A00 – 0x1BFF (TPDO)	0	Número de subíndex
	1	Mapejat del 1er objecte
	2	Mapejat del 2on objecte
	3 – 64	Mapejat del objecte n

Taula 5: Paràmetres del mapping d'un PDO

Les entrades que s'escriuen del mapejat de configuració tenen el següent format:

MSB LSB

Índex (16 bits)	Subíndex (8 bits)	Longitud [bits] (8 bits)

Cada una d'aquestes entrades descriuen un dels objectes que hi ha al PDO: la entrada al Diccionari que ocupa (índex i subíndex) i la seva longitud en bits.

Sabent quants objectes hi ha en un PDO, la seva longitud i la seva posició, és fàcil identificar i col·locar cada un dels objectes. La ubicació en un PDO d'un objecte es la següent:

Byte 0 Byte 7

Obj 1	Obj 2	Obj 3	Obj 4

Exemple de Configuració del mapejat d'un PDO i la seva distribució en el PDO:

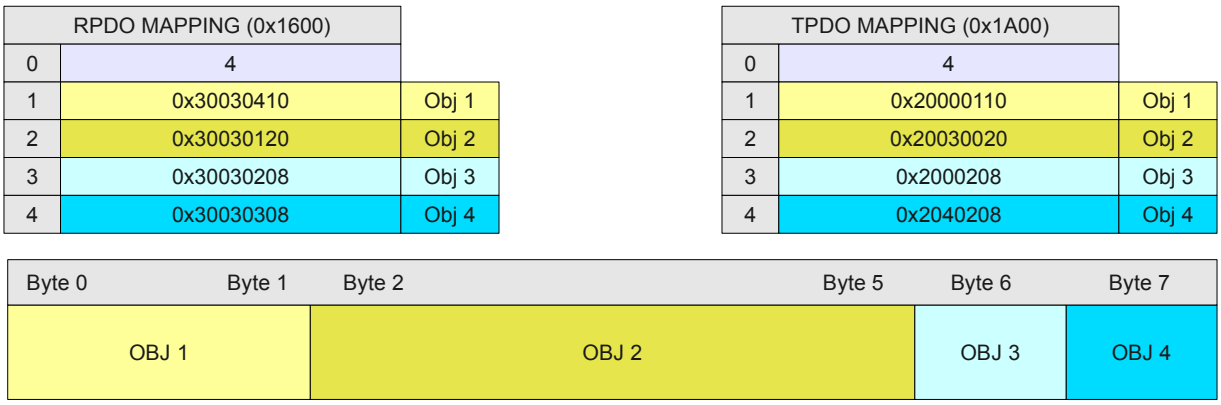

Fig. 13: Configuració i distribució d'objectes en un PDO

La configuració del mapejat d'entrada en el Consumidor i el mapejat de transmissió en el Productor no té perquè ser igual: l'únic que ha de coincidir és la longitud de cada objecte mapejat. Això es degut a que, per exemple, l'objecte 1 en el Productor pot ser la lectura d'un sensor de temperatura i, en el receptor, pot ser Sensor de temperatura 4 del dispositiu X.

3.2.5 Protocols de control d'errors

CANopen disposa de serveis de control d'estat que permeten al Master detectar errors en els dispositius de la xarxa. Ambdós funcionen amb l'identificador 0x700 + la id del node. Es disposa de 2 de protocols diferents, el Node Guarding i el Heartbeat. Els estats de node possibles son els següents:

Valor	Estat
0*	Boot up
4	Stopped
5	Operational
127	Pre-operational

Taula 6: Valors possibles d'estat de node

*: en el cas de l'estat 0, aquest només es possible en el heartbeat. S'envia quan el node ha acabat d'inicialitzar-se i pasa a l'estat pre-operacional. S'anomena missatge de Boot Up.

3.2.5.1 Node Guarding

En aquest protocol, el Master envia periòdicament un missatge de 'guarding' a mode de petició a l'esclau, amb el bit rtr actiu. Quan l'esclau corresponent rep el missatge, ha d'enviar la resposta. El format dels missatges node guarding és el següent:

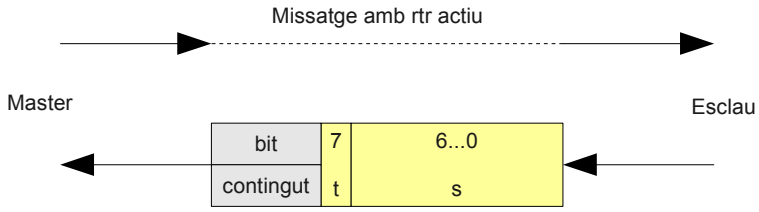

Fig. 14: Format missatge Node Guarding

El missatge que envia el Master és un missatge d'1 byte de longitud, les dades del qual son indiferents, amb el bit rtr actiu. L'identificador del missatge és 0x700 + id del node al qual va destinat.

La resposta que envia l'Esclau és un missatge d'1 byte en el qual, els bits 0..6 (s) son l'estat en que es troba el dispositiu i el bit 7 (t) és un bit de 'toogle', que va alternant el seu valor en cada transmissió.

3.2.5.2 Heartbeat

El Heartbeat és l'altre mètode de control d'estat del node. En aquest cas, el node esclau envia missatges Heartbeat periòdicament cap a la xarxa, en un interval de temps configurat. Els missatges son molt semblants als de Node Guarding, amb la diferència que en aquest cas no hi ha el bit de toogle. El format, doncs, és el següent:

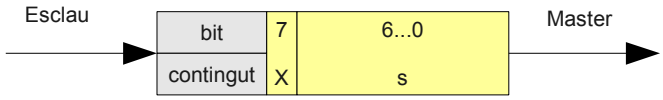

Fig. 15: Format d'un missatge Heartbeat

Igual que en el Heartbeat, els bits 0...6 son l'estat del node, mentres que la X és sempre 0.

3.2.6 SYNC: Missatges de sincronització

El missatges SYNC son missatges que envia, de forma periòdica, un únic productor de SYNC a la xarxa (típicament serà el Master). Aquests missatges es fan servir per sincronitzar diferents nodes, de manera que els nodes, a la recepció de l'objecte SYNC, reaccionin i enviïn un PDO o posin en marxa un temporitzador.

Aquest missatge té una doble funció: per una part permet controlar el flux de PDOs a la xarxa, ja que el node coneix cada quants SYNC s'ha d'enviar un PDO en cada dispositiu i, a més, procura als dispositius una manera d'enviar dades de manera cíclica sense haver-se de preocupar de 'clocks' ni sincronitzacions.

3.2.7 NMT: Missatges de control de la xarxa

El missatges NMT (Network Management Control), son missatges que envia, única i exclusivament, el Master de la xarxa. Mitjançant aquests missatges el Master controla el sistema de comunicacions dels Esclaus, ja que pot ordenar a un node posar-se en un estat determinat. Son missatges no confirmats i les ordres possibles son les següents:

Ordre	**CS**	**Descripció**
Start	1	L'Esclau entra en mode Operacional
Stop	2	L'Esclau passa a l'estat d'aturada
Enter Pre-Operational	128	L'Esclau passa a estar en mode Pre-Operacional
Reset Node	129	El node un reset complet i es restauren les valors per defecte del Diccionari d'Objectes
Reset Communication	130	El node fa un reset de comunicacions i carrega els valors per defecte en les entrades del Diccionari d'Objectes del perfil de comunicacions

Taula 7: Valors possibles d'estat de Node en un missatge NMT

Cada missatge NMT consta de 2 bytes i té el següent format:

BYTE	0	1
Contigut	CS	ID DEL NODE

Fig. 16: Format d'un missatge NMT

Els missatges NMT són els més prioritaris en una xarxa CANopen, i per això tenen l'identificador de missatge amb més prioritat: el 0. Tots els nodes de la xarxa reben tots els missatges NMT, encara que finalment només processaran la ordre si l'Identificador del missatge correspon amb el del node.

3.2.8 EMCY: Missatges d'Emergència

Degut a que CANopen no es un sistema jeràrquic Mestre – Esclau i que la monitorització del node només indica l'estat de comunicacions i no l'estat intern de l'aplicació i/o capa superior, cada node requereix un identificador molt prioritari per a indicar als demés situacions d'error. Aquest mecanisme s'anomena “Emergency Messaging” i, l'objecte associat, “Emergency Message” (EMCY a partir d'ara). Un missatge EMCY conté 8 bytes de dades, amb el següent format:

BYTE	0	1	2	3	4	5	6	7
Contingut	Codi de l'error		Registre de l'error	Codi d'error específic del fabricant				

Fig. 17: Format d'un missatge EMCY

Els codis d'errors estan especificats en el document CiA DS-301. A més, quan un dispositiu envia un missatge d'EMCY, guarda la informació de l'error en la seva historia d'errors.

El Registre de l'error correspon a l'objecte 1001h, Error Register, i és una entrada d'1 Byte de longitud amb informació de l'error. Els errors possibles son el següents:

Bit	Obligatori	Descripció
0	Si	Error genèric
1	No	Intensitat (corrent)
2	No	Voltatge
3	No	Temperatura
4	No	Error de comunicació
5	No	Específic del perfil de dispositiu
6	No	Reservat (sempre 0)
7	No	Específic del fabricant

Taula 8: Errors possibles en el registre d'errors

L'últim camp, codi específic del fabricant, és propi de cada fabricant i la interpretació és lliure.

3.3 NXP LPC2292

Per a fer el projecte, s'ha optat per la família de microcontroladors LPC 2290, de la casa Nexperia i basats en tecnologia ARM - RISC. El projecte s'ha creat fent servir el LPC2292, funcionant sobre una placa d'evaluació de la casa Phytec. El microcontrolador elegit havia de complir els següents requisits:

- Admetre programació en C
- Ser ràpid
- Disponibilitat d'entrades i sortides
- Disposar, al menys, de 2 ports CAN
- Disponibilitat de plaques d'evaluació / prototipatge ràpid

Amb aquestes característiques hi havia la disponibilitat d'una placa d'evaluació de Phytec, el model PCM-023, amb el microcontrolador LPC 2292, 2 ports CAN, 2 Uarts, moltes entrades / sortides i un cristall funcionant a 10MHz. Amb aquesta configuració, el micro pot funcionar fins a 60MHz, cosa que dona una velocitat prou bona per a poder gestionar adequadament la càrrega de treball que suposa el software de CANopen i totes les funcionalitats associades (timers, etc).

La gama LPC2000 de NXP és molt amplia, i hi han disponibles molts models, sent possible elegir des de micros molt senzills fins als més avançat, amb convertidors A/D, D/A, PWM i inclús controlador de pantalla.

4 Plantejament i solució

4.1 Idea bàsica

Desenvolupar una implementació bàsica de CANopen, que sigui conforme a l'estàndard i que, sense arribar a nivells industrials, tingui una fiabilitat destacable. Després de l'estudi de la documentació, i a nivell de comunicacions, el node resultant (la plataforma hardware + el software escrit) ha de reunir les següents característiques:

- Ser capaç de funcionar a les velocitat més comuns de CANopen
- Implementar el protocol SDO
- Implementar el protocol PDO d'intercanvi de dades, així com els serveis associats
- Implementar els sistemes de control d'errors Node Guarding i Heartbeat
- Implementar el sistema NMT de control remot de node
- Implementar el sistema de comunicació d'errors Emergency

Com que el protocol CANopen és un protocol de capa superior (d'aplicació) funcionant sobre un bus CAN i, amb l'objectiu de crear un software modular, s'hauria de desenvolupar també un Driver de CAN. Aquest driver hauria de tenir les següents característiques:

- Selecció de port CAN a utilitzar
- Velocitat de funcionament configurable
- Número de bytes a enviar configurable
- Utilització de diferents buffers d'enviament
- Identificadors configurables
- Implementació dels filtres de CAN, així com les funcions de control d'aquests
- Implementació del bit rtr
- Interfície amb el driver senzilla i ben definida

4.2 Especificacions

Després de tenir les idees clares sobre com es volia dividir el software i les prestacions que havia de tenir, es va dur a terme un estudi més detallat encara de la documentació disponible. Així, es va decidir que el node hauria d'incorporar les següents funcionalitats:

- Velocitats de CAN més comunes: 125kbps, 250kbps, 500kbps i 1000kbps
- Protocol SDO: ha de suportar transmissió normal (amb fins a 4 bytes de dades), en el qual un cicle de comunicació es duu a terme en 2 missatges i la transmissió segmentada, en la qual les dades es divideixen en segments de fins a 7 bytes. En aquesta última, s'haurà de crear un buffer per a guardar les dades abans d'escriure-les al Diccionari d'Objectes i, a més, la mida d'aquest buffer hauria de ser configurable.
- Protocol PDO: el node ha de disposar de 4 PDOs, cada un dels quals adment fins a 8 bytes de dades. A més, aquests haurien de ser completament configurables en temps d'execució, a través de missatges de configuració SDO. S'han de poder configurar els objectes que portarà cada PDO, dels quals s'ha de comprovar l'existència i si admeten accés a través de PDO i, a més, s'haurà de comprovar també que la longitud total del PDO no és superior a 8 bytes. També han de ser configurables els modes d'enviament i recepció i, a més, s'hauria de poder rebre peticions de PDO remotes (mitjançant el bit rtr).

- Serveis associats als PDO: bàsicament, l'objecte SYNC. El node ha de ser capaç de rebre i reconèixer correctament l'objecte SYNC i enviar, si toca, els PDO corresponents o processar els PDO que estan a l'espera de rebre un SYNC.
- Node Guarding i Heartbeat. El node hauria de suportar els 2 protocols de monitorització de nodes disponibles. A més, la documentació indica que el mètode recomanat i preferit és el Heartbeat. Si a l'arrancar el node un dels 2 està configurat correctament, s'haurà d'arrancar immediatament. El que no es pot tenir es funcionant, en un mateix node, els 2 protocols i, per tant, s'haurien d'implementar les mesures de control necessàries per a que, en el cas que tots dos estiguin configurats, només en funcioni un.
- Missatges NMT: el node ha d'implementar les funcions necessàries per a interpretar els missatges de control del node i, en cada cas, dur a terme les accions associades necessàries. A més, segons l'estat, el node ha d'ignorar certs tipus de missatges, per tant seria convenient desactivar les interrupcions d'aquests missatges (mitjançant els filtres de CAN) per a reduïr el nombre d'interrupcions innecessàries en el microcontrolador.
- Emergency: s'hauria d'implementar les funcions bàsiques corresponents als missatges d'emergència, que inclouen: l'enviament del missatge, un històric d'errors (de mida configurable) i un objecte de descripció de l'error.

A més, i degut a que el projecte començava de zero, s'hauria de desenvolupar un driver complet de CAN. Aquest driver hauria de ser senzill d'utilitzar però, a la vegada, implementar totes les funcionalitats necessàries derivades de CANopen. Després d'un breu estudi, tant del microcontrolador com de CANopen, es va decidir que el driver hauria d'implementar les funcionalitats següents:

- Intentar fer-lo escalable: si bé la tarja d'evaluació de Phytec només incorporava 2 ports de CAN, el processador LPC2294 (model immediatament superior al 2292 de la placa) n'incorpora 4. Així, i en vistes a futures ampliacions, el driver hauria de suportar de manera transparent fins a 4 ports.
- El LPC2292 incorpora, per a cada port CAN, 3 buffers d'enviament. El driver hauria de ser capaç d'enviar un missatges a través de qualsevol d'aquests buffers.
- RTR: el driver ha de ser capaç tant d'enviar missatges amb el bit RTR actiu com de rebre'ls i notificar a l'aplicació (en aquest cas l'stack de CANopen) que el missatge rebut tenia actiu el bit rtr. En el cas del nostre node de CANopen, al ser només esclau, la funcionalitat d'enviar missatges amb rtr no era necessària, però es va decidir implementar-lo igualment, per una qüestió de deixar el driver complet.
- El CAN del LPC2292 incorpora filtres, mitjançant els quals es pot limitar quins identificadors s'accepten i quins no. Amb els filtres configurats, els missatges amb identificadors acceptats generen una interrupció i es tracten de manera normal. Els identificadors no inclosos en els filtres s'ignoren i no generen cap mena d'interrupció. Aquesta prestació resulta bastant interessant ja que evita al processador treball innecessari (analitzar missatges i descartar-los, si no eren per nosaltres) i, per tant, s'haurien de crear el conjunt de funcions necessàries per tractar els diferents filtres.
- Funció d'enviament: el driver hauria d'incorporar una funció d'enviament de dades a través de CAN. Aquesta funció hauria de tenir els següents paràmetres: port can pel qual s'ha d'enviar, buffer a utilitzar, bit rtr actiu o no, identificador del missatge, longitud del missatges (fins a 1 byte) i punter a les dades.

- Funció de recepció: s'ha d'activar la interrupció corresponent. Cada cop que es rebi una interrupció s'hauria de cridar una funció a la qual es passarà com a paràmetre l'identificador, el punter a les dades i informació de la trama (bit rtr, longitud, etc). La funció d'atenció associada serà configurable.

4.3 Idea general del Hardware

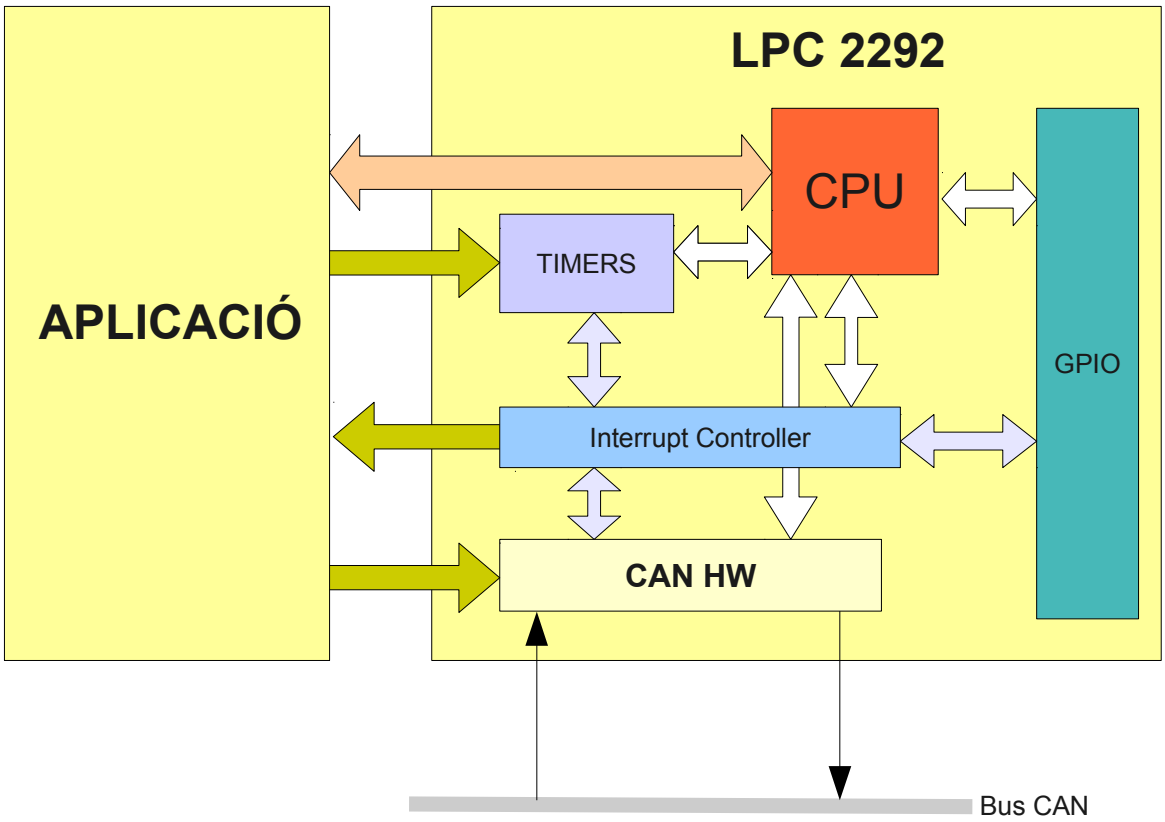

Fig. 18: Idea bàsica del Hardware

4.4 Diagrama de blocs del software

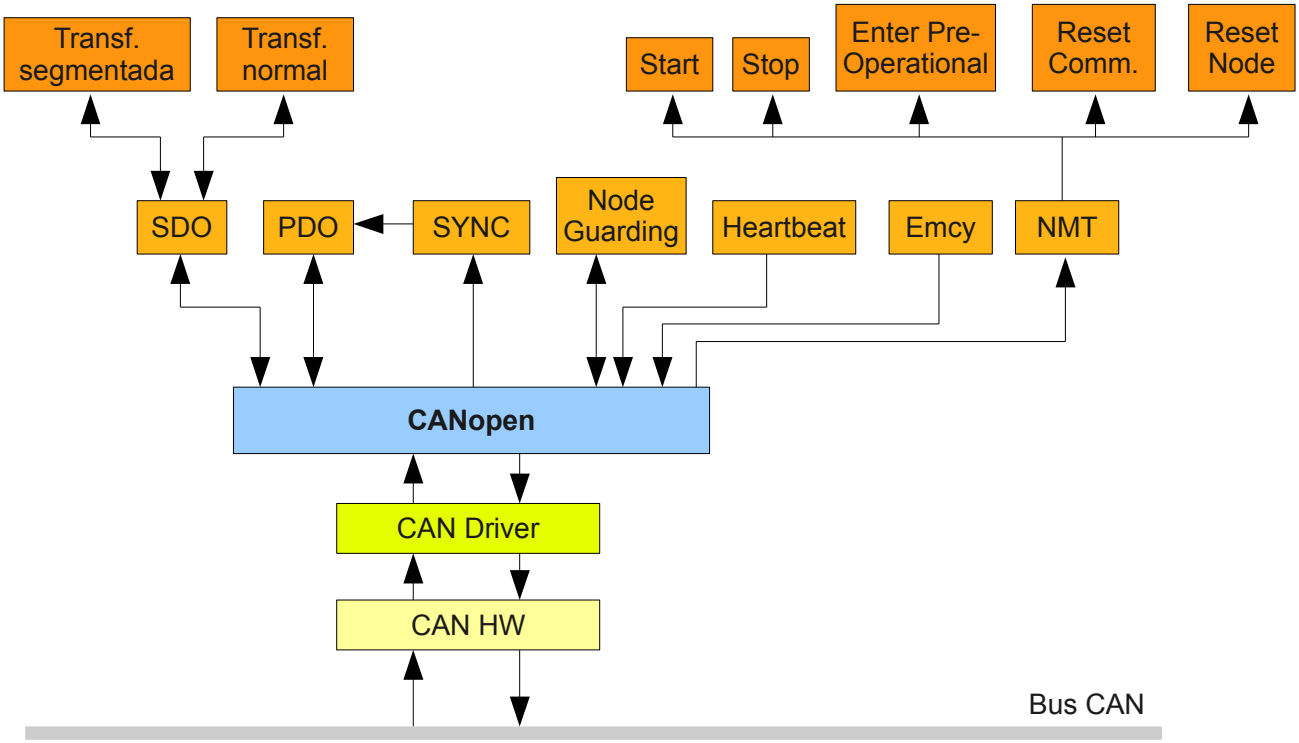

Fig. 19: Esquema general del Software CANopen

4.5 Diagrames de flux

4.5.1 SDO

El següent diagrama mostra la màquina d'estats corresponent a la gestió d'un cicle de comunicació a través d'SDO, mostrant el camí seguit tant en el cas de que sigui una transferència de 4 bytes o menys, on només arribarà un missatge Initiate amb les dades, o per el contrari la transferència és de més de 4 bytes i s'enviarà un missatge Initiate i, a continuació, els missatges amb les dades segmentades.

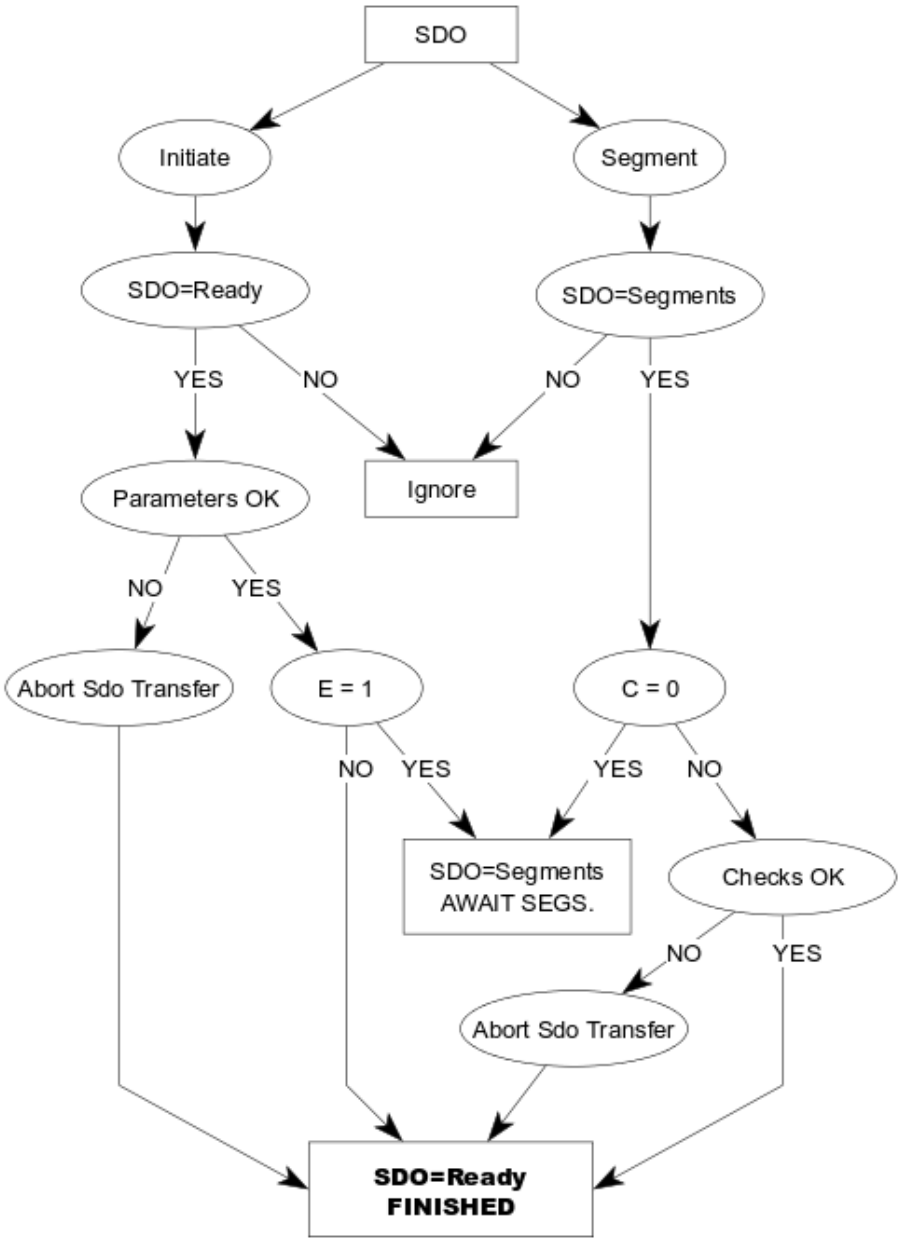

Fig. 20: Transferència SDO

4.5.2 Configuració de PDOs a través d'SDO

4.5.2.1 Configuració de Paràmetres de Comunicació

El següent diagrama mostra, a grans trets, com gestiona el software la configuració dels paràmetres de comunicació d'un PDO a través d'una transferència SDO. Cal dir que aquest diagrama no exclou el vist a l'apartat 3.5.1, sinó que totes les comprovacions que es mostren, entrarien en la casella Paràmetres OK del citat diagrama.

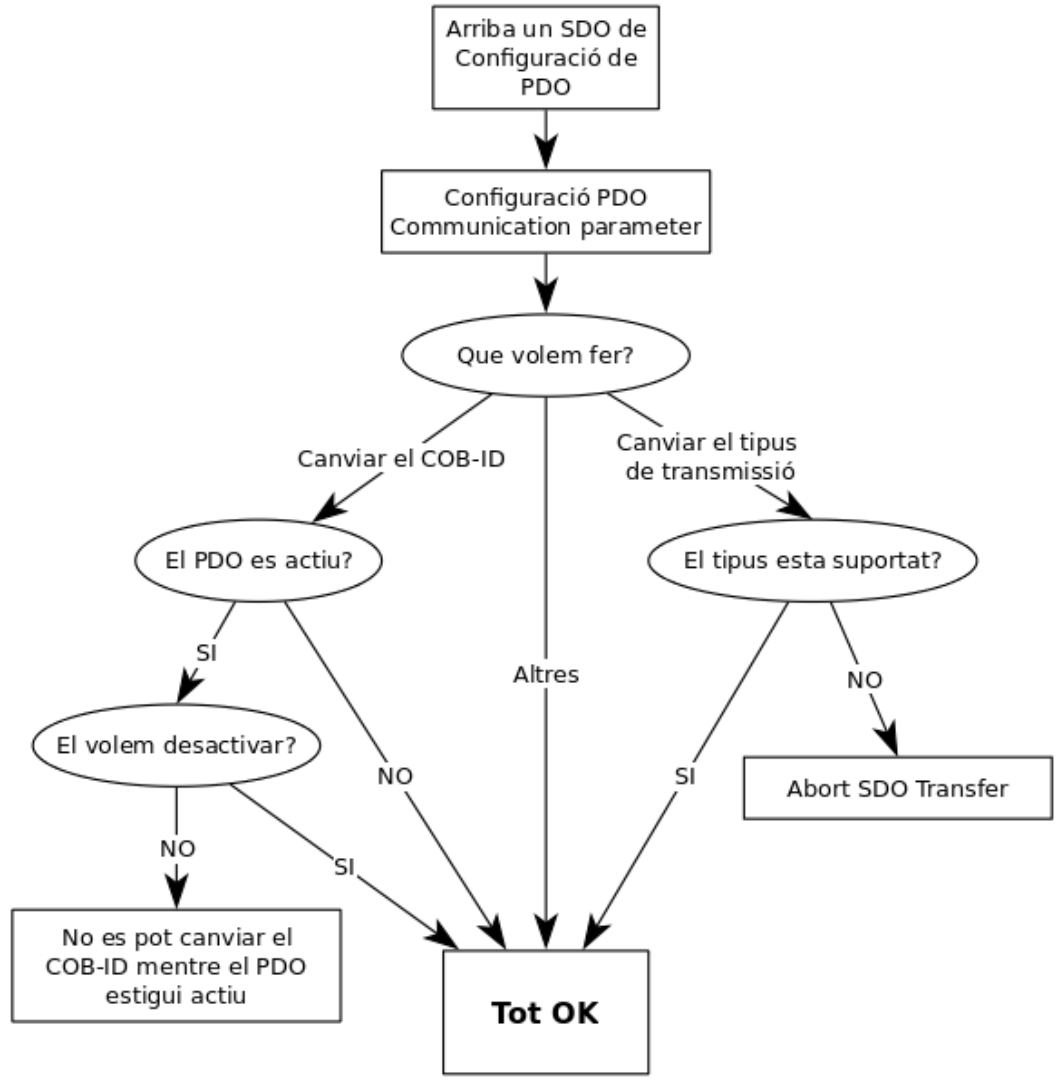

Fig. 21: Configuració paràmetres de comunicacions de PDO

4.5.2.2 Configuració de Paràmetres de Mapejat

En aquest apartat, el diagrama mostra la gestió de la configuració dels objectes que portarà un PDO. Al fer la configuració dels paràmetres del "mapping" s'han de tenir en compte moltes variables, ja que en la entrada de configuració dels objectes s'hi posa quin objecte hi va i la seva longitud. Així, a l'hora de configurar el PDO s'ha de comprovar cada objecte configurat i, a més, s'ha de comprovar que, al finalitzar, la longitud del PDO sigui la correcta (8 bytes de dades com a molt).

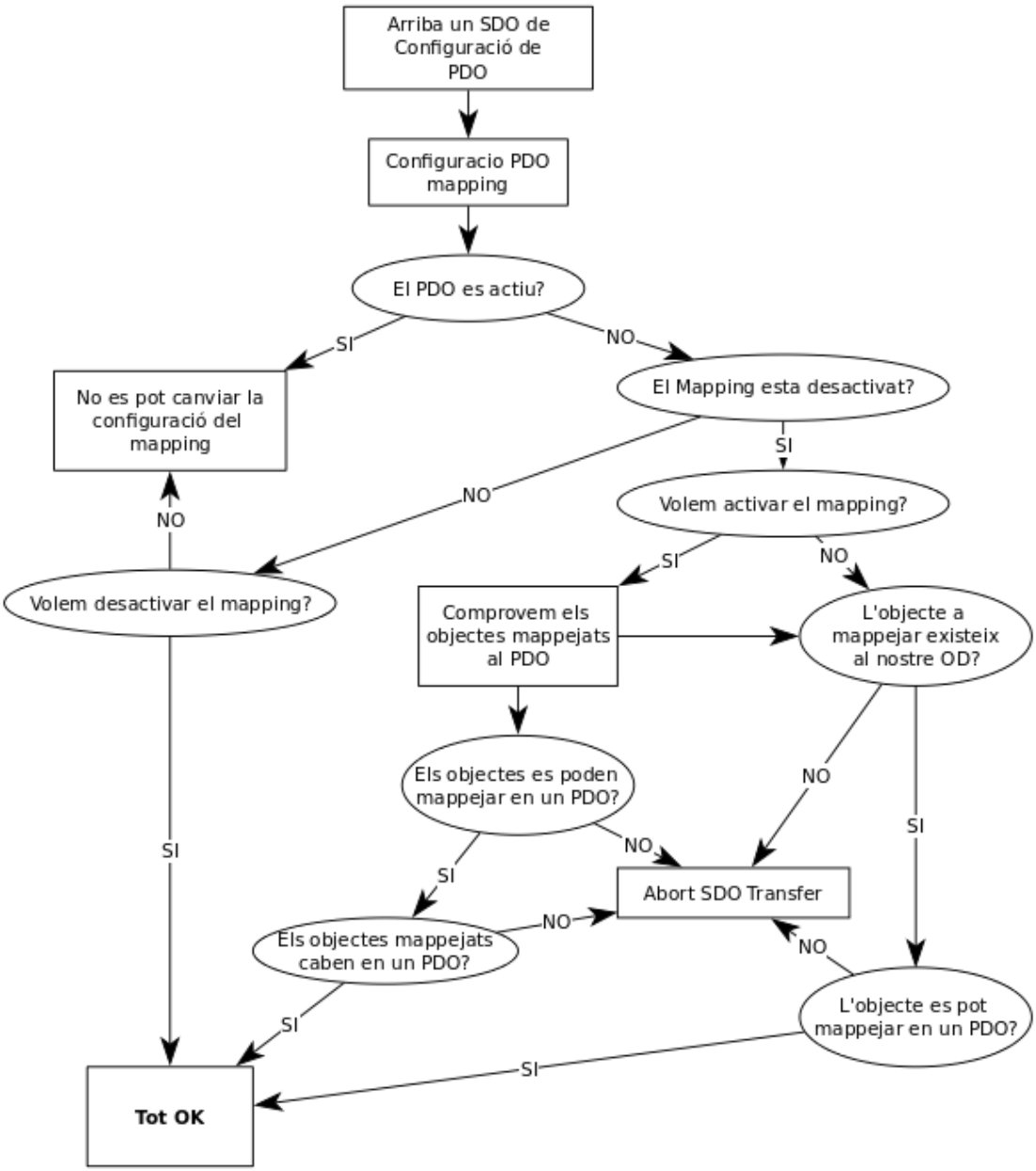

Fig. 22: Configuració paràmetres PDO Mapping

4.5.3 Recepció de l'objecte SYNC

A continuació es mostra la gestió que fa el protocol amb la recepció d'un objecte Sync. Bàsicament, es tracta de processar els PDOs pendents i actualitzar les variables dels PDOs d'enviament síncron i cíclic.

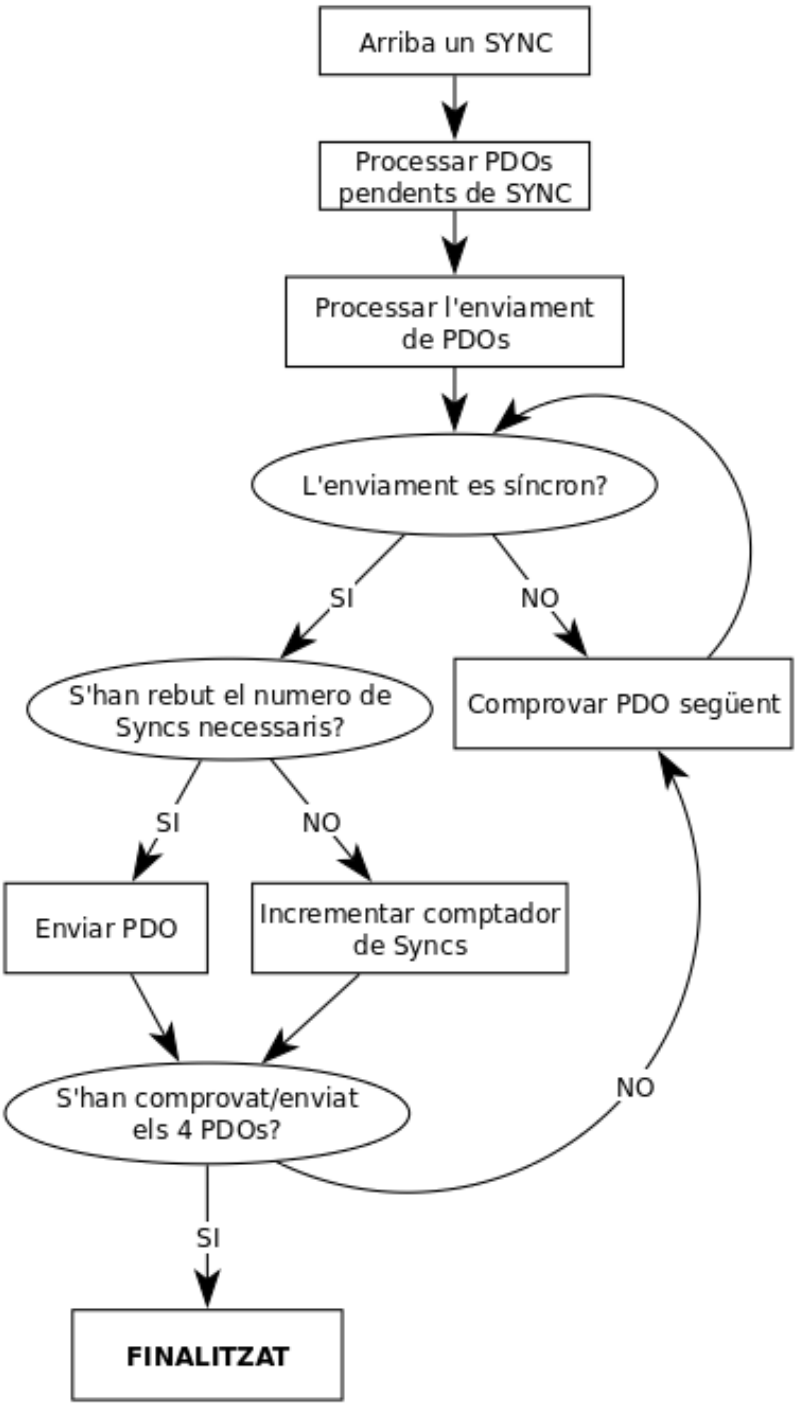

Fig. 23: Recepció del SYNC

4.5.4 Arrancada del node

La següent figura mostra els passos bàsics que segueix el node durant la seva arrancada i inicialització:

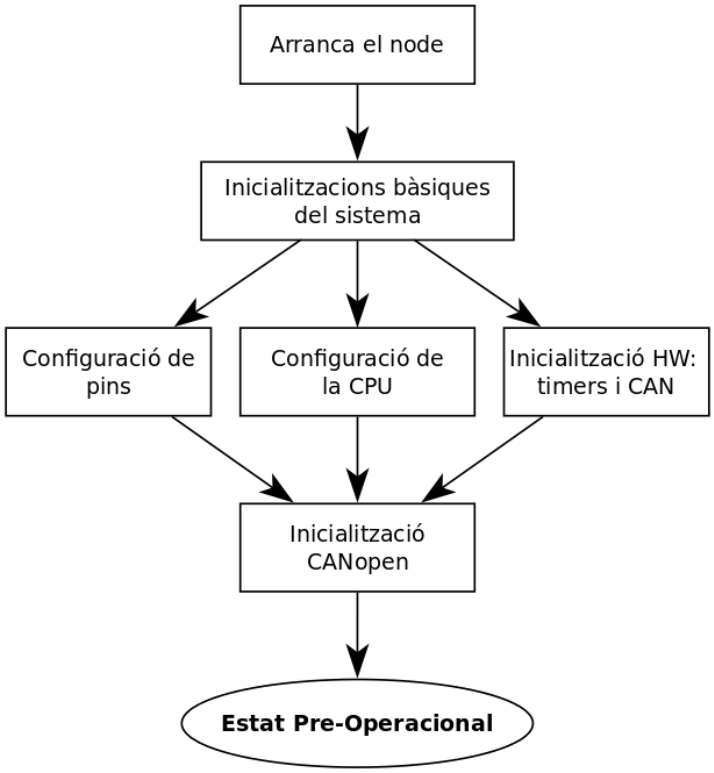

Fig. 24: Arrancada del node

5 Proves

Un cop es va donar per finalitzat la part bàsica de la implementació de CANopen, s'ha a pasat a la fase de proves. Es tractar de comprovar, amb l'ajuda d'un PLC capaç de funcionar com a Master de CANopen, que les funcionalitats implementades funcionen correctament.

5.1 Material emprat

Per a fer les proves necessàries, s'ha necessitat material per tal de crear un entorn bàsic on fer les proves necessàries. El material utilitzat ha estat el següent:

- PLC PowerPanel 45 amb pantalla tàctil
- Mòdul de comunicacions IF33 de CAN per al PowerPanel 45
- Entorn de programació i gestió del PLC
- Cable CAN

5.2 Preparació de l'entorn de proves

Per a poder fer les proves necessàries, el primer pas va ser programar un entorn mínim des del qual poder enviar missatges de CAN a través del PLC. Aquest programa havia de poder

- Enviar SDO
- Enviar PDO
- Enviar l'objecte SYNC
- Enviar missatges NMT de control
- Enviar missatges de petició de Heartbeat

Per tant es va procedir a estudiar els exemples disponibles, així com l'ajuda del programa per a poder programar l'entorn de proves.

Durant l'estudi del programa del PLC vam veure com, afegint un fitxer EDS, el mateix entorn de programació del PLC s'encarregaria d'enviar el SYNC i, si es requeria, les peticions de Node Guarding. Així doncs, el programa de proves quedava reduït a 3 funcions:

- Enviar / rebre SDO
- Enviar PDO
- Enviar NMT

D'altra banda es va crear un fitxer EDS per al node de proves, utilitzant el software CANeds de Vector, disponible gratuïtament a la xarxa. Aquest programa permet crear, des d'una interfície gràfica molt amigable, un fitxer EDS amb les entrades seleccionades; a més, el mateix programa permet comprovar el fitxer EDS resultant. Així doncs, va facilitar molt la creació del fitxer EDS, i a més amb la seguretat que el fitxer resultant esta conforme a la norma.

5.3 Proves efectuades

5.3.1 Missatges SDO

Les primeres proves que es van dur a terme van ser referents als missatges SDO. Aquest missatges es poden rebre just a l'arrancar el node, i van servir molt bé per acabar de definir l'ordre de les dades en el missatge. Degut a la complexitat, un cop es van poder rebre i enviar SDOs correctament, arreglar els errors en les altres implementacions va ser força senzill. Les proves es van centrar en:

5.3.1.1 Lectura d'un objecte

- **Objecte existent de 4 bytes o menys:** en aquest cas, el node respon amb un missatge que conté les dades a llegir, a més de la informació per identificar-ne la longitud.
- **Objecte existent de més de 4 bytes:** quan l'objecte té més de 4 bytes el node respon amb un missatge que indica que la transmissió haurà de ser segmentada i n'indica la longitud total de les dades. A partir d'aqui, respon als missatges amb les dades segmentades.
- **Objecte inexistent:** si s'intenta llegir un objecte inexistent, el node respon amb un missatge d'Abort SDO transfer amb el codi d'error corresponent a “Object does not exist in OD” o “Subindex does not exist”, segons el cas.
- **Objecte només d'escriptura:** en el cas que la petició de lectura faci referència a un objecte amb permisos de només escriptura, el node envia un missatge Abort SDO Transfer amb el codi d'error que correspon a “Write only object”.

5.3.1.2 Escriptura d'un objecte

- **Objecte existent de 4 bytes o menys:** en accessos de 4 bytes o menys, el missatge de petició conté directament les dades, que el node guarda i envia un missatge de confirmació.
- **Objecte existent de més de 4 bytes:** si s'ha d'escriure un objecte de més de 4 bytes, la transferència serà segmentada. En aquest cas, el node guarda tots els fragments en un buffer i, quan li arriba l'últim segment, comprova que la longitud de dades es l'esperada (la notificada al primer missatge) i, si es correcte, escriure les dades al Diccionari d'Objectes i envia la confirmació d'escriptura.
- **Objecte indexistent:** igual que en la lectura, el node envia un missatge Abort SDO Transfer amb els codis corresponents a “Object does not exist in OD” o “Subindex does not exist”
- **Objecte només de lectura:** si l'accés d'escriptura no esta permès, el node envia un missatge Abort SDO Transfer amb el codi d'error “Read only object”.
- **Objecte existent i d'escriptura però especificant una longitud de dades incorrecta:** en el cas que el missatge d'escriptura indiqui una longitud de dades diferent a la que marca el Diccionari d'Objectes, el node envia un missatge d'Abort SDO Transfer amb el codi d'error “Length of parameter missmatch”.

5.3.2 Missatges NMT

Els missatges NMT són bàsics en el funcionament de la xarxa CANopen, ja que defineixen l'estat dels nodes. Des del punt de vista dels missatges CAN utilitzats, aquest son molt més senzills que els SDO, ja que son missatges que només contenen 2 bytes de dades: un per al node al qual s'adrecen i l'altre per indicar l'estat. Els tipus d'NMT i les seves característiques son els següents:

- **Start:** en aquest estat, el node ha d'acceptar tots els tipus de missatge i, a més, ha d'enviar els PDO de transferència cíclica que hi hagi configurats.
- **Enter Pre-operational:** quan el node es troba en l'estat pre-operacional, processa tots el missatge excepte els PDO, que no processa ni envia.
- **Stop:** en l'estat de parada, el node només accepta el missatges NMT i els missatges de control d'estat (respon al node guarding o envia el heartbeat, segons la configuració).
- **Restart communication:** quan es rep la ordre de reiniciar les comunicacions, el node restaura tots els objectes del perfil de comunicacions (objectes amb index 0x1000 – 0x1FFF) al seu valor per defecte, el que tenien a l'arrancar el node. Posteriorment, el node torna a enviar el missatge de Boot Up.
- **Restart node:** al rebre un NMT restart node, el node ha de restaurar als valors per defecte del perfil de comunicacions, el perfil de fabricant i el perfil de dispositiu.

En tots els casos, s'ha comprovat que el node no acceptés ni enviés missatges que no havia de processar. A més, en cada cas s'ha comprovat que els valors del Diccionari d'Objectes es restauraven al valor per defecte correctament i, al reiniciar-se, el node tornés a iniciar la seqüència d'inici.

També es va comprovar que, amb els filtres de CAN, el missatges amb identificadors no acceptats no generaven cap mena de notificació cap a CANopen, sinó que simplement eren ignorats. Això, a part de simplificar el tractament del missatges, allibera al microcontrolador de feina innecessaria.

5.3.3 Missatges PDO

Un cop es va tenir tant el protocol SDO com el sistema NMT provat i funcionant correctament, es va passar a fer les comprovacions de tot el sistma PDO. El subsistema dels PDOs és bastant complicat, degut a que s'ha de gestionar diversos tipus de recepcions i enviaments i, a més, quan es rep un PDO s'han de comprovar els objectes mapejats un per un.

5.3.3.1 Recepció de PDO

Quan el node rep un PDO, s'ha comprovat que segueix la seqüència correcta i que els resultats son els esperats. La seqüència és la següent:

- Al rebre un Pdo, el node ha de determinar de quin es tracta i acceptar-lo només en casa de que estigui actiu.
- Segons el tipus de recepció, el PDO s'ha de guardar (es processarà quan arribi un SYNC) o processar directament.
- Quan es processi el missatge, s'ha de consultar el mapping i determinar si el PDO té la longitud esperada. En cas contrari s'envia un missatge EMCY amb el codi d'error de Longitud de PDO incorrecta.
- Amb la configuració del mapping, el node determina cada objecte del missatge i comprova si l'accés es correcte: permet escriptura, permet accés a través de PDO, la longitud de l'objecte és la correcta, etc. Segons es van determinant els objectes, si els paràmetres son correctes es van guardant a la entrada del Diccionari d'Objectes corresponent.

5.3.3.2 Transmissió de PDO

La transmissió d'un PDO pot venir donada per 3 motius:

- **Transmissió cíclica:** amb aquest tipus de transmissió, el PDO té definit el nombre de SYNCs a rebre abans d'enviar-se. Així doncs, el node ha de comptar el nombre de SYNCs que li arriben i, quan siguin iguals a la configuració d'un PDO, aquest s'envia.
- **Petició de transmissió remota:** es possible que al node li arribi un missatge amb el bit RTR actiu i l'identificador d'un dels PDOs de transmissió. En aquest cas, i si la configuració del PDO ho admet, s'està requerint l'enviament del PDO especificat. Segons la configuració del PDO, s'enviarà al moment (si es asíncron) o amb l'arribada del següent SYNC (si el PDO és síncron).
- **S'ha rebut un PDO amb un tipus de recepció que provoca l'enviament d'un PDO:** en la configuració dels tipus de recepció, els números 254 i 255 son específics del fabricant. En la nostra implementació, si es rep un PDO amb el tipus de recepció configurat a 254, es processa i s'envia el PDO de transmissió corresponent.

6 Pressupost

6.1 Hardware

Codi	Descripció	Unitats	Preu unitari	Preu parcial
PH_PCM023	Tarjeta d'evaluació amb CAN	1	210,00 €	210,00 €
Seg_JLINK	Mòdul USB per programar la tarjeta evaluadora	1	95,00 €	95,00 €
DB9_PlugM	Connector DB-9 Mascle	2	0,30 €	0,30 €
3_wire_mtr	Metres de cable amb 3 filaments	1,25	1,20 €	1,38 €
PP_45	PLC PowerPanel 45 amb funcions de CANopen Master	1	1.000,00 €	1.000,00 €
IF_33	Mòdul de comunicacions CAN IF33 per a PP45	1	45,00 €	45,00 €
Cat5_Cr_2m	Cable ethernet creuat de 2m per a programar el PLC	1	3,50 €	3,50 €
		Preu base total:		*1.310,18 €*
		Base imponible:		*209,63 €*
		Preu total:		**1.519,81 €**

6.2 Altres

Codi	Descripció	Unitats	Preu unitari	Preu parcial
HoresPREP	Hores emprades en la preparació del projecte	35	10,00 €	350,00 €
HoresPRG	Hores emprades en el projecte	560	10,00 €	5.600,00 €
HoresPRV	Hores emprades en les proves del projecte	75	10,00 €	750,00 €
HoresDOC	Hores emprades en redactar la documentació	50	10,00 €	500,00 €
HoresREV	Hores emprades en la revisió del codi	10	10,00 €	100,00 €
HoresMTG	Hores emprades en el muntatge de l'entorn	5	10,00 €	50,00 €
		Preu base total:		*7.350,00 €*
		Base imponible:		*1.176,00 €*
		Preu total:		**8.526,00 €**

6.3 Total Absolut

Codi	Descripció	Unitats	Preu unitari	Preu parcial
HW	Partida Hardware del projecte	1	1.310,18 €	1.310,18 €
Altres	Partida Altres del projecte	1	5.600,00 €	7.350,00 €
		Preu base total:		*8.660,18 €*
		Base imponible:		*1.385,63 €*
		Preu total:		***10.048,81 €***

7 Conclusions

Després de més de 7 mesos de feina s'ha obtingut el que es volia: una implementació bàsica del protocol CANopen. En aquests 7 mesos, evidentment, s'han aprés moltes coses, no només de CAN, CANopen i de programació, sinó també de com encarar un projecte. Entre d'altres:

- El més bàsic: llegir la documentació, analitzar-la i entendre-la és fonamental. No es pot començar un projecte sense entendre abans que ha de fer el conjunt.
- Programant: és indispensable portar un ordre i pensar globalment. Val més gastar 1 hora en organitzar les idees i tenir una idea bàsica del que es busca que començar a programar i, més endavant, haver de fer 2 passos enrere per tornar a organitzar i reestructurar tot el codi.
- CANopen és un protocol completament obert, del qual obtenir la informació necessària ha estat molt senzill. A més, a Internet hi ha molta informació disponible.

Encara que ha costat 7 mesos de feina, evidentment el software resultant es pot millorar. Entre d'altres, es podrien implementar les següents millores:

- Si bé s'ha intentat que el resultat fos molt modular, encara es podria aprofundir més en aquest sentit.
- Estudiar possibles aplicacions finals i pensar la millor interfície de crides a aplicació possible.
- Possiblement, el codi resultant es pugui reduir i millorar-ne la velocitat d'execució.

En qualsevol projecte, sempre hi han problemes i aquest no ha estat una excepció. Els problemes més significatius han estat:

- Degut a errors en el codi, la placa d'avaluació es va quedar bloquejada i no era possible tornar-la a programar. La solució, al final, va ser senzilla: una combinació d'interruptors a l'arrancada va permetre tornar a programar la placa. La solució, molt senzilla; trobar-la, no tant.
- En la programació: probablement degut a la falta d'experiència i d'ordre a l'hora de programar, en les primeres fases van ser freqüents els “buffer overflows”. Tot semblava funcionar bé fins que de sobte es penjava... una millor organització segurament hagués reduït aquests errors.
- A l'hora de fer algunes proves finals, es va descobrir una fallada en la gestió de filtres de CAN. Si s'hagués provat el driver de CAN amb més deteniment, aquests errors s'haguessin solucionat en la primera part del projecte, i no en la fase final.

Per sort, el resultat final ha estat molt positiu: el protocol funciona bé i no s'han detectat penjades del sistema, funciona a les velocitats estàndards i es capaç de rebre quantitats de dades bastants grans. A més, el protocol interactua correctament amb un element comercial com és el PLC PowerPanel45, fet que indica que, finalment, el resultat és una implementació complerta i, sobretot, conforme a l'estàndard establert.

8 Bibliografia

8.1 CAN

- http://es.wikipedia.org/wiki/Bus_CAN
- http://www.softing.com/home/en/industrial-automation/products/can-bus/more-can-bus/index.php?navanchor=3010320

8.2 CANopen

- http://en.wikipedia.org/wiki/CANopen
- http://www.can-cia.org/
- http://www.softing.com/home/en/industrial-automation/products/can-bus/more-can-open/index.php?navanchor=3010572
- http://www.canopensolutions.com/english/about_canopen/about_canopen.shtml
- Document CiA DS-301: especificació perfil de comunicacions
- Document CiA DSP-306: especificació fitxer EDS

8.3 Phytec PCM 023

- http://www.phytec.com/products/rdk/ARM-XScale/phyCORE-ARM7-LPC229x.html
- LPC2292 User Manual
- http://www.embeddedrelated.com/groups/lpc2000/show/39100.php
- http://www.embeddedrelated.com/groups/lpc2000/show/5785.php

8.4 PowerPanel 45

- Manual d'ajuda

Anex 1: Codi

Aquesta part del projecte és informació confidencial i s'ha eliminat. Contempla tot el codi font que s'ha desenvolupat per al projecte.

Anex 2: Fitxer EDS resultant

Aquesta part del projecte és informació confidencial i s'ha eliminat. Contempla el fitxer EDS que descriu les capacitats del protocol per al node CANopen desenvolupat.