Dissertação realizada sob a orientação do

Prof. Doutor Bruno Jorge Ferreira Ribeiro
Departamento de Engenharia Electromecânica
Universidade da Beira Interior

E co-orientação de

Prof. Doutor Pedro Miguel Figueiredo Dinis Oliveira Gaspar
Departamento de Engenharia Electromecânica
Universidade da Beira Interior

Prof. Doutor António Eduardo Vitória do Espírito Santo
Departamento de Engenharia Electromecânica
Universidade da Beira Interior

Dedico este trabalho aos meus pais, Daniel e Lídia, por todo o esforço que fizeram para que eu pudesse frequentar a universidade e à Aura, pelo tempo que não lhe pude dedicar ao longo deste ano.

CONCEPÇÃO DE UM MODELO DE SISTEMA PARA A GESTÃO DA ILUMINAÇÃO INTERIOR EM ESPAÇOS RESIDENCIAIS

Resumo

Desde muito cedo que o homem tem vindo a aplicar os avanços tecnológicos à sua habitação. Os motivos têm sido diversos ao longo do tempo embora aumentar a segurança e transformar o lar num lugar mais confortável e acolhedor seja o motivo principal.

Ultimamente os conceitos de Domótica e "Casa Inteligente" têm vindo a apresentar soluções visando o aumento da segurança e do conforto, a melhoria das comunicações interpessoais, e significava poupança de recursos. Uma especialidade funcional deste conceito propõe-se à gestão do sistema de iluminação interior, possibilitando o controlo automático da luminosidade em função de critérios previamente estabelecidos, visando o conforto e a racionalização de energia.

Pretende-se apresentar, nesta dissertação, um modelo de sistema de gestão da iluminação baseado num sistema de automação fortemente distribuído e integrador, com a capacidade de estabelecer ligações lógicas entre dispositivos (interruptores, lâmpadas, controladores, etc.), e acima de tudo, de suportar, intrinsecamente, a definição de "cenários" de funcionamento na linha da abordagem em instrumentação virtual.

Os modelos dos dispositivos apresentados são baseados na norma de comunicações para sistemas de automação distribuídos CANopen. Cada um dos dispositivos apresenta um perfil funcional que identifica a sua função específica, (sensor de luminosidade, actuador de estores, lâmpadas de intensidade regulável, entre outros) e, nesse perfil, estão definidos todos os parâmetros relativos à sua configuração/programação, assim como as suas entradas e/ou saídas virtuais (*data points*).

Agradecimentos

Em primeiro lugar, gostaria de expressar a minha profunda gratidão ao meu orientador, Bruno Jorge Ferreira Ribeiro, por todo o apoio que me prestou ao longo deste trabalho, pelo magnânimo suporte que foi ao longo da consolidação de conhecimentos e da concepção física deste trabalho, pelo constante acompanhamento e atenção em todos os aspectos relativos à evolução deste trabalho e pela extrema paciência e compreensão que teve comigo nos momentos de maiores dificuldades.

Gostaria também de expressar a minha gratidão aos meus co-orientadores, Pedro Dinis Gaspar e António Espírito Santo, pelo seu valoroso contributo ao longo da evolução do trabalho e pelas suas importantes sugestões que contribuíram para o enriquecimento deste trabalho.

Agradeço também aos meus colegas, em especial a Helder Neves, Henrique Teixeira e Filipe Casimiro, pelo seu constante encorajamento e acompanhamento durante os longos dias de trabalho e pelas suas valorosas opiniões.

Covilhã, 18 de Agosto de 2009

Bruno da Silva Abreu

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

GLOSSÁRIO

AC	*Acceptance Code*
AM	*Acceptance Mask*
BSP	*Bit Stream Processor*
BTL	*Bit Timing Logic*
CAL	*CAN Application Layer*
CAN	*Controller Area Network*
CiA	*CAN in Automation*
CIL	*CPU Interface Logic*
CMS	*CAN-based Message Specification*
COB	*Communication Object*
CRC	*Cyclic Redundancy Check*
DLC	*Data Length Code*
DU	*Data Unit*
EDS	*Electronic Data Sheet*
EML	*Error Management Logic*
EOF	*End of Frame*
EPA	*Enhanced Performance Architecture*
FFD	*Full Function Device*
IDE	*Identifier Extension*
ISO	*International Standards Organization*
MAC	*Medium Access Control*
MBM	*Message Buffer Memory*
NBR	*Nominal Bit Rate*
NBT	*Nominal Bit Time*
OD	*Object Dictionary*
OSI	*Open Systems Interconnection*
PCI	*Protocol Control Information*
PDO	*Process Data Object*
PDU	*Protocol Data Unit*
RFD	*Reduced Function Device*
RPDO	*Receive Process Data Object*

RTR	*Remote Transmition Request*
SCI	Subunidade Controladora Integradora
SCWF	Subunidade Controladora *Wireless – Full Function*
SCWR	Subunidade Controladora *Wireless – Reduced Function*
SDO	*Service Data Object*
SDU	*Service Data Unit*
SOF	*Start of Frame*
SPI	*Serial Peripheral Interface*
SYNC	*Synchronization Object*
TCL	*Transceiver Control Logic*
TPDO	*Transmit Process Data Object*
TQ	*Time Quantum*
UC	Unidade Controladora
UIS	Unidade Integradora Supervisora

1. INTRODUÇÃO

1.1. ENQUADRAMENTO E MOTIVAÇÃO

A Domótica tem auferido de uma divulgação crescente devido aos grandes benefícios que oferece, nomeadamente em aplicações que visam o aumento da autonomia a pessoas com necessidades especiais. Com o auxílio da domótica, as tarefas diárias como controlar a rega do jardim, ligar a iluminação exterior quando escurece, estabelecer níveis de conforto através do controlo da temperatura e iluminação adequados, descer ou subir os estores a determinadas horas do dia ou desligar lâmpadas em divisões desocupadas podem ser extremamente facilitadas. Deste modo, a Domótica poderá contribuir para maiores níveis de bem-estar e, ao mesmo tempo, optimizar os gastos energéticos, conduzindo a poupanças nos consumos de electricidade, gás e água [1].

A Domótica possibilita, também, o aumento da segurança de pessoas e bens, detectando e sinalizando situações de emergência, como incêndios, fugas de gás, inundações ou situações de intrusão. Para as pessoas idosas ou com mobilidade reduzida, os sistemas domóticos podem ser de grande utilidade, pois conferem-lhes uma maior autonomia e qualidade de vida. O uso de um telecomando ou de um dispositivo que reconheça mensagens de voz pode facilitar imenso determinadas tarefas tais como abrir a porta da rua, ligar a iluminação ou desligar o forno, evitando que a pessoa tenha de se deslocar [2].

Uma das áreas de acção mais importantes da Domótica é a gestão da iluminação, permitindo libertar as pessoas de acções repetitivas e rotineiras, como: (i) acender e apagar os pontos de luz quando se entra ou sai de uma divisão; (ii) abrir ou fechar os estores de manhã ou ao anoitecer; (iii) executar acções que visem uma melhor gestão energética, visando, sempre, o aumento do conforto dos seus utilizadores e uma maior racionalização de recursos.

O modelo de sistema de gestão da iluminação aqui abordado insere-se numa plataforma de suporte já existente, que utiliza a norma de comunicações CANopen [3]. Do ponto de vista prático, este sistema prima pela sua simplicidade e capacidade de definição de cenários de funcionamento, o que oferece uma grande flexibilidade de adaptação às necessidades dos utilizadores. De facto, após uma configuração inicial, onde são definidos todos os parâmetros necessários ao funcionamento de cada dispositivo, o utilizador não tem que voltar a preocupar-se com as configurações.

A plataforma onde este sistema se integra tem um elevado nível de integração, isto é, permite a coexistência de vários subsistemas que interagem entre si. Isto permite a troca de informações entre dispositivos de subsistemas diferentes. Com isto, este sistema de gestão da iluminação terá possibilidade de interacção com outros subsistemas e, desse modo, optimizar o funcionamento de um futuro sistema global.

1.2. OBJECTIVOS

As principais finalidades de um sistema domótico visam o conforto, tanto térmico como luminoso, sem menosprezar a gestão dos recursos energéticos disponíveis. Para tal, um sistema domótico deve estar encarregue do controlo de variados equipamentos domésticos, tais como os sistemas de climatização (aquecimento e refrigeração) ou os sistemas de iluminação (natural e artificial), entre outros.

O principal objectivo deste trabalho é desenvolver um modelo de sistema de gestão da iluminação de espaços residenciais, baseado num sistema de automação fortemente distribuído. Este é um modelo de sistema com elevada flexibilidade operacional e

modularidade, com a capacidade de estabelecer ligações lógicas entre os vários dispositivos, e acima de tudo, de suportar a definição de "cenários" de funcionamento. Com base nas necessidades específicas para este género de sistemas, será definido um conjunto de modelos de dispositivos que proporcione a correcta operação do sistema.

Os modelos de dispositivos cooperarão entre si conferindo autonomia ao sistema e, dessa forma, permitindo que este funcione com reduzida intervenção por parte do utilizador. Para o funcionamento de cada dispositivo e do sistema global ser o mais adequado, em cada um dos modelos de dispositivo, serão definidos: (i) parâmetros de configuração para que o utilizador possa adaptar o funcionamento do sistema às suas necessidades; e (ii) *data points,* entradas e saídas, que representam os pontos de ligação entre dispositivos, possibilitando a interacção entre estes.

Serão apresentados alguns cenários de funcionamento, onde serão enumeradas as diversas relações entre dispositivos na forma de ligações virtuais entre *data points* e as suas diferentes configurações.

Com o objectivo de colocar em prática o modelo idealizado, serão elaborados os diversos dicionários de objectos, tendo em conta os parâmetros de configuração e os parâmetros correspondentes às entradas e saídas virtuais de cada um dos dispositivos. Serão também realizados os *electronic data sheets* necessários à conformação da rede, tendo em conta as informações provenientes dos dicionários de objectos e considerando sempre as especificações.

Dentro dos objectivos deste trabalho insere-se ainda, a concepção de *hardware* de ensaio com capacidade para integrar a norma de comunicações CANopen. Neste objectivo está incluído o teste ao funcionamento das placas de hardware.

Tratando-se de um sistema implementado sobre uma plataforma de suporte já existente [3], este trabalho irá introduzir as necessárias metodologias de desenvolvimento para aplicações que constituem o universo da Domótica e conferir maior realismo à referida plataforma. De facto, os próximos trabalhos de evolução deste sistema, para um sistema domótico global, poderão apoiar-se na metodologia seguida aqui para desenvolver os novos modelos de sistemas destinados aos outros campos abordados pela Domótica.

1.3. ESTRUTURA DA DISSERTAÇÃO

A presente dissertação está estruturada em quatro partes que passo a descrever de seguida.

A primeira parte desta dissertação, correspondente ao capítulo 2, apresenta uma revisão do conceito de domótica e uma breve descrição do sistema global, no qual este trabalho se insere. Na descrição do sistema está incluído o seu princípio de funcionamento geral e ainda, alguns aspectos de maior relevância relativos aos protocolos de comunicação.

A segunda parte da dissertação é materializada pelos capítulos 3, 4 e 5. Nestes capítulos, é abordada a concepção do modelo de sistema de gestão da iluminação. São apresentados os modelos para cada um dos dispositivos constituintes do sistema e ainda, alguns cenários de funcionamento para as situações mais habituais, onde se tenta mostrar o modo de operação e algumas potencialidades do sistema. São também elaborados os dicionários de objectos e as *electronic data sheets* de cada um dos dispositivos do sistema.

Na terceira parte da dissertação, que corresponde ao capítulo 6, será demonstrado o funcionamento do sistema. Numa primeira parte são apresentadas as unidades intervenientes nas acções de teste, posteriormente é apresentada a bancada de testes, incluindo as ferramentas de análise e, por fim, são apresentados os resultados dos ensaios realizados.

A última parte desta dissertação corresponde às conclusões sobre o trabalho desenvolvido. Nesta parte são também apresentadas propostas de trabalhos para o futuro.

No sentido de não transformar a leitura da dissertação fastidiosa, muitos dos detalhes do protocolo de comunicação assim como alguns conceitos foram colocados em anexo. Estão presentes também em anexo as tabelas dos dicionários de objectos e um exemplo de uma das *electronic data sheets*.

2. ENQUADRAMENTO DO MODELO DO SISTEMA

2.1. O CONCEITO DE DOMÓTICA

Os custos de construção e de manutenção dos edifícios da sociedade actual são, em geral, muito elevados. Esses custos constituem gastos significativos para as empresas que exploram os edifícios. Numa tentativa de contenção de custos de exploração e manutenção, encetou-se por aplicar às novas construções, sobretudo aos grandes edifícios de serviços, sistemas de automação. Assim, a partir da década de 80, surge o conceito de "Edifício Inteligente" principalmente associado à optimização dos serviços prestados pelas empresas, no sentido de melhorar a produtividade das mesmas. Surgem, então, os Sistemas de Gestão Técnica de Edifícios onde se considera o edifício como sendo um conjunto de várias funções/serviços [4].

Com os Sistemas de Gestão Técnica de edifícios surgem dois novos conceitos: (i) o conceito de "serviço" associado às funções desempenhadas pelos equipamentos, tais como o serviço de iluminação, o serviço de controlo de acessos ou o serviço de detecção de incêndios; (ii) o conceito de "integração" entre serviços que surge com o intento de encontrar novas potencialidades resultantes das suas interacções e da necessidade de racionalização de recursos. A título de exemplo, o serviço de controlo de acessos comunica com o serviço de apoio à portaria. Este último recebe informação do primeiro e transfere para o serviço de vigilância, etc. [4].

A Domótica emergiu a partir do conceito de "edifício inteligente", embora tenha objectivos distintos. O conceito de edifício inteligente é aplicado a edifícios destinados à prestação de serviços e direccionado para o apoio das organizações de modo a optimizar os serviços por elas prestados e, desse modo, levar ao aumento da sua produtividade. O conceito de domótica está orientado, em primeira instância, para o utilizador, no sentido de suprir as suas necessidades de conforto e bem-estar.

A palavra domótica resulta da junção de *domus* (do latim domicílio) com robótica (que por sua vez teve origem na palavra checa *robota* que significa automação) [5] [6]. Um sistema domótico permite a gestão da maioria dos recursos habitacionais, libertando os moradores da execução das tarefas diárias e rotineiras [3]. Para além de aumentar significativamente o conforto pessoal do utilizador, um sistema domótico contribui também para a racionalização de recursos, poupança de tempo e energia, e consequentemente, de dinheiro.

2.2. O SISTEMA DOMÓTICO

O sistema domótico, no qual este trabalho se insere, é um sistema de automação fortemente distribuído, com capacidade de estabelecer ligações virtuais entre diversos dispositivos, e consequentemente, com capacidade de troca de informação em tempo real entre os diversos dispositivos do sistema. O ponto forte deste sistema baseia-se na possibilidade de definição de cenários de funcionamento, o que, do ponto de vista de utilização diária, é uma grande vantagem prática [3].

Na figura 2.1 é possível observar uma representação da configuração global do sistema. Esta configuração abrange as diferentes áreas de aplicação da Domótica bem como a possibilidade de acesso remoto através de *internet* [3]. A automação é uma das áreas de aplicação da domótica e é nesta que se insere o serviço de iluminação, que será alvo de estudo ao longo deste trabalho. De facto, o sistema aqui proposto é uma pequena peça constituinte do sistema global representado na figura 2.1.

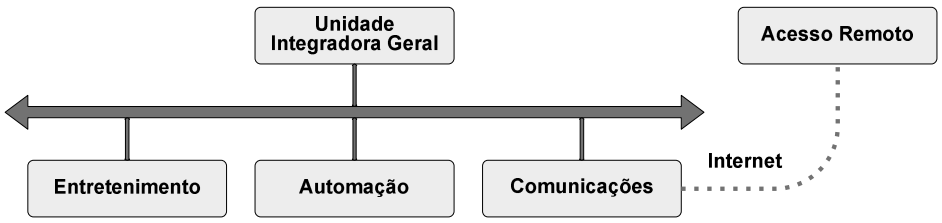

Fig. 2.1. Arquitectura geral de um sistema domótico [3].

Este sistema apresenta as propriedades características dos sistemas distribuídos, apresentando elevada flexibilidade, robustez e modularidade. No sentido de facilitar a sua utilização, este segue uma orientação de Instrumentação Virtual através da possibilidade de definição de cenários de funcionamento, o que lhe confere elevada facilidade de instalação e baixa manutenção evolutiva. No seu desenvolvimento, foi considerado que grande parte da actuação dos equipamentos domésticos se encontra localizada nos quadros eléctricos da habitação. Com isto, tornou-se útil utilizar dispositivos, localizados nos quadros eléctricos, interligados através de um barramento de dados baseado em cabo [3].

Por outro lado, alguns dos pontos de actuação e medida, encontram-se afastados dos quadros eléctricos. De modo a ultrapassar esta dificuldade prática e para garantir a flexibilidade de instalação são utilizados dispositivos remotos que comunicam por radiofrequência (ver figura 2.2) [3].

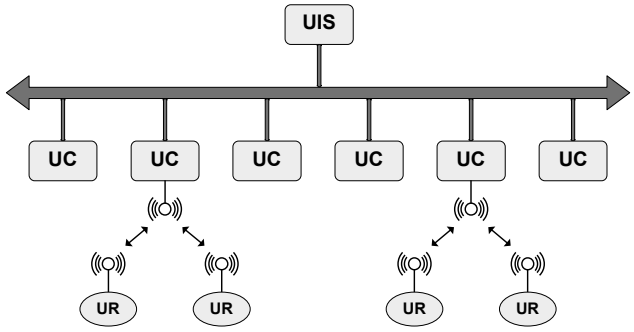

Fig. 2.2. Arquitectura do sistema de automação e controlo [3].

2.2.1. Princípio Geral de Funcionamento

O funcionamento dos dispositivos, que pertencem ao sistema domótico, baseia-se em perfis funcionais. Cada dispositivo possui um perfil funcional de acordo com a função desempenhada no sistema. É nos perfis funcionais que estão definidos os parâmetros relativos ao seu funcionamento, nomeadamente: entradas e saídas das ligações virtuais e parâmetros de configuração, como representado na figura 2.3. Assim, em cada perfil funcional, existem parâmetros de configuração que condicionam o funcionamento do dispositivo e, também, parâmetros que representam as entradas e saídas resultantes das funções desempenhadas por este. Essas entradas e saídas são os pontos de ligação das ligações virtuais entre os vários dispositivos presentes no sistema global. Todos os objectos pertencentes a um perfil funcional são mapeados na rede, isto é, possuem um endereço único em toda a rede. Desta forma é possível, em qualquer instante, verificar o estado de funcionamento de um elemento do sistema, bem como, alterá-lo para obter um novo desempenho.

No dispositivo 1 da figura 2.3 é possível ver, a título de exemplo, dois parâmetros (A e B) que correspondem às suas saídas, dois parâmetros (C e D) que correspondem a duas entradas e três parâmetros (E, F e G) que são parâmetros de configuração do dispositivo 1. No dispositivo 2, os parâmetros A e B são as suas entradas e os parâmetros C e D são as suas saídas. Os parâmetros H, I e J deste dispositivo representam os seus parâmetros de configuração. Consoante a função a desempenhar, será necessário introduzir mais ou menos parâmetros de configuração assim como entradas e saídas para ligações virtuais, *data points*.

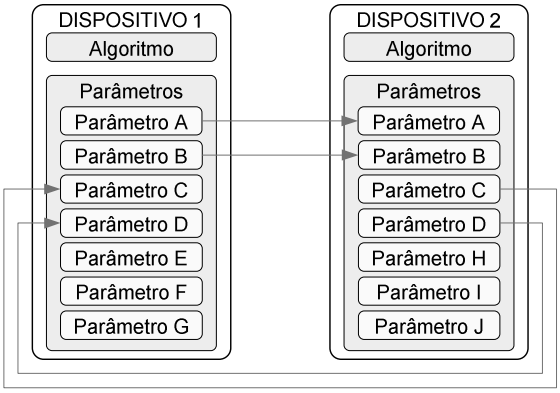

Fig. 2.3. Esquematização de perfis funcionais de dois dispositivos.

Os objectos definidos nos perfis funcionais estão contidos numa estrutura designada por Dicionário de Objectos (OD ou *Object Dictionary*), sendo esta definida pela norma CANopen. Todos os elementos do sistema, excepto as unidades *wireless*, possuem esta estrutura. O OD é a interface entre a informação contida nos dispositivos e a rede, atribuindo um endereço lógico a cada endereço físico da memória interna do dispositivo. O acesso aos objectos mapeados é possível através de serviços específicos do CANopen. Para tal, é necessário conhecer o endereço do dispositivo que contém o dicionário onde o objecto está definido, o endereço de 16 bits que identifica o objecto no interior do dicionário e por fim o sub-endereço de 8 bits que identifica o elemento dentro do objecto, caso este seja uma estrutura complexa [3]. Para aceder aos objectos podem ser usados SDOs (*Service Data Objects*) ou PDOs (*Process Data Objects*) [7]. No primeiro caso, um dispositivo "cliente" pode aceder ao OD do dispositivo "servidor" evocando o endereço e o sub-endereço que identifica o objecto. No segundo caso é feita a configuração no momento da instalação de modo a que o transmissor e o receptor conheçam o significado da informação transportada pelo PDO. Assim, não é necessário indicar os endereços e sub-endereços dos objectos a serem acedidos. Ambos os serviços são descritos com mais detalhe no anexo A.5.

Alguns dos parâmetros de configuração dos perfis funcionais podem ser alterados de modo a ajustar o funcionamento do sistema às necessidades do utilizador. Neste contexto surgem os cenários de funcionamento. Podem ser definidos vários cenários onde são "recriados" os vários modos de vida do utilizador. Em cada cenário são guardados os valores dos parâmetros contidos nos perfis de todas as unidades do sistema, como o representado na figura 2.4.

Em cada cenário, o utilizador selecciona os valores dos parâmetros de configuração adequados conforme as suas necessidades. Esses cenários são guardados em memória não volátil e, ao longo do dia, são descarregados para as unidades do sistema, introduzindo, desta forma, os valores previamente definidos pelo utilizador para os diversos dispositivos. Assim, o sistema adapta-se às várias situações que vão surgindo.

A título de exemplo, e olhando para a figura 2.4, se o dispositivo 1 presente no cenário **A** corresponder a um sensor de temperatura e os parâmetros 1 e 2 corresponderem à temperatura máxima e mínima respectivamente, o utilizador pode definir um valor

máximo de 19ºC e um valor mínimo de 17ºC. Sempre que o cenário **A** for descarregado para o sistema, o dispositivo 1 vai actuar sobre o sistema de acordo com esses valores e desta forma, o sistema pode controlar a temperatura para valores dentro desse intervalo.

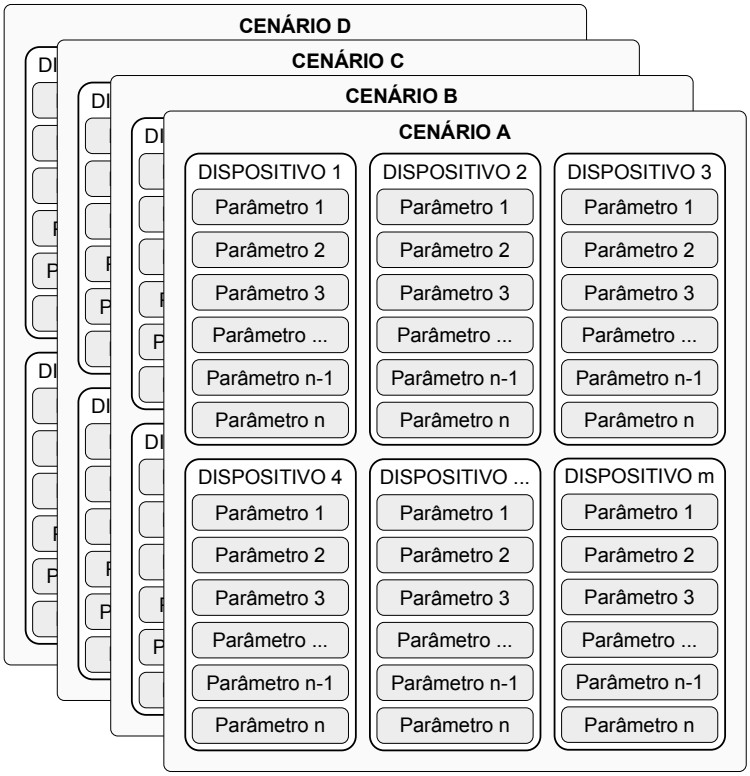

Fig. 2.4. Esquematização de quatro cenários de funcionamento de um sistema com m dispositivos.

No entanto, o utilizador pode definir outros valores num possível cenário **B**, por exemplo um máximo de 20ºC e um mínimo de 18ºC. Assim, sempre que for descarregado para o sistema o cenário **B**, o dispositivo 1 vai ter em conta os valores de 20ºC e de 18ºC e o sistema ajusta os valores da temperatura para valores dentro desse intervalo.

Uma prévia definição de todos os cenários desejados, assim como do seu modo de activação, quer por evento ou por temporizador, simplifica extremamente a operação do sistema domótico na utilização quotidiana, pois toda a complexidade é "escondida", restando apenas pequenas acções a ser realizadas pelo utilizador nos momentos de excepção à rotina. Este modo de operação possibilita uma maior convivialidade com o sistema, tornando-o mais acessível à maioria população [2].

Por exemplo, durante os períodos de tempo em que a casa está desabitada não é necessário que o sistema de iluminação artificial seja utilizado, mas quando está alguém na habitação já é justificável a utilização de luz artificial em combinação com a luz natural para iluminar a divisão. Desta forma, o utilizador pode definir dois cenários: um cenário "Casa Desabitada" onde o sistema é configurado para não utilizar a iluminação artificial e um cenário "Casa Habitada" onde o sistema é configurado para utilizar a iluminação artificial em conjunto com a luz natural. Quando todos os habitantes estiverem fora, o cenário "Casa Desabitada" é descarregado para o sistema. Da mesma forma, o cenário "Casa Habitada" é descarregado para o sistema sempre que alguém estiver em casa. Com isto, o utilizador não tem que realizar qualquer acção de configuração e assim ganha tempo livre e conforto.

2.2.2. As Unidades

O sistema global é formado por unidades de dois tipos:

1. Uma unidade integradora e supervisora (UIS) com função de gestão dos fluxos de informação entre o sistema de controlo e automação e os sistemas de mais alto nível [3]. Também pode ter a função de interface humana local num sistema mais desenvolvido.
2. Diversas unidades controladoras (UC) de hierarquia inferior à UIS. As funções das unidades controladoras consistem em medir e actuar sobre os equipamentos domésticos [3].

Cada UC pode corresponder a uma rede *cluster tree* formada pelos vários dispositivos interligados numa rede sem fios. Desta forma, a UC não existe fisicamente mas sim como entidade funcional, cuja função consiste em garantir a operacionalidade funcional do conjunto de dispositivos. Estas unidades possuem *data points* que garantem o funcionamento global do sistema [3].

2.2.2.1. A Unidade Integradora e Supervisora

Este sistema de automação pode funcionar isolado, isto é, sem estar integrado num sistema domótico multi-serviço. Para fazer face a esse desafio, o sistema está dotado de uma *interface* humana residente na unidade UIS. Esta unidade apresenta capacidade de programação de cenários que entrarão em funcionamento após o *download* dos parâmetros para as várias unidades de controlo, assumindo também funções de monitorização, arquivo e apresentação de informação [3].

A UIS tem capacidade para servir de ponte entre o sistema de automação e os níveis superiores de integração, quando existam. É nesta unidade que está presente a *interface* humana e é também nesta unidade que são programados os cenários que entram em funcionamento depois de serem descarregados todos os parâmetros para as várias UCs. A UIS assume também as funções de controlo, de monitorização, arquivo e apresentação de informação [3].

A *interface* com as UCs é através de rede CANopen, onde a UIS desempenha as funções de *master*, assegurando a formação e gestão da rede [3].

2.2.2.2. A Unidade Controladora

A unidade controladora é a unidade modularidade base do sistema [3]. Esta pode assumir diferentes configurações físicas, como mostrado na figura 2.5. As UCs podem interagir directamente com os equipamentos domésticos utilizando os recursos próprios (ver figura 2.5 à direita), ou assumir uma topologia distribuída (ver figura 2.5 esquerda e centro). Sempre que é necessário acrescentar mais funções à UC, existe a possibilidade de integrar mais do que uma unidade SCI (ver figura 2.5 ao centro).

Uma UC pode integrar várias subunidades remotas interligadas através de uma rede de comunicações sem fios. As subunidades que a poderão integrar são:

1. Subunidade Controladora Integradora (SCI);
2. Subunidade Controladora *Wireless Full Function* (SCWF);

3. Subunidade Controladora *Wireless Reduced Function* (SCWR).

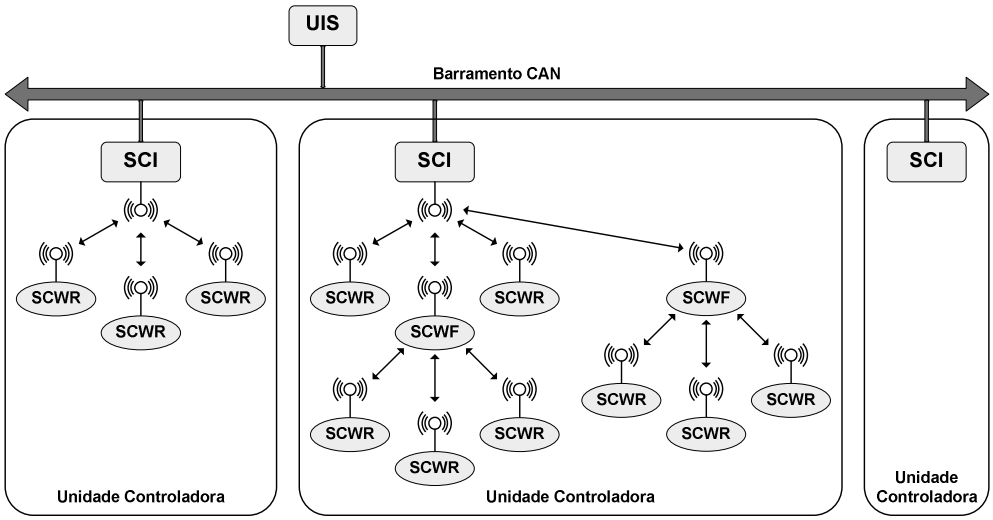

Fig. 2.5. Estrutura das unidades controladoras [3].

A presença de uma subunidade SCI é obrigatória, sendo opcional a inclusão de subunidades *wireless*, de acordo com as necessidades [3].

Em situações onde é necessário ter maior capacidade de *input/output* ou quando a sua localização o obriga, as UCs, cuja unidade base se localiza nos quadros eléctricos, podem assumir a forma de uma rede sem fios em *cluster tree*. Assim, a área habitacional coberta por cada UC é muito superior, não ficando limitada apenas aos quadros eléctricos [3]. As UCs têm a função de gestão de todo o *input* e *output*, incluindo a medida de grandezas físicas necessárias ao funcionamento do sistema, numa determinada zona [3].

2.2.2.2.1. A Subunidade Controladora Integradora

Devido ao facto de a energia eléctrica estar centralizada nos quadros, o sistema possui unidades com capacidade I/O alojadas nos mesmos, a partir das quais se procede ao controlo dos vários equipamentos domésticos. Estas unidades são designadas por unidades de quadro e são a base das UCs. Estas unidades podem funcionar em conjunto

com as unidades remotas, exercendo, desta forma, o papel de ponte entre uma rede cablada e uma rede sem fios [3].

Do ponto de vista da operacionalidade do sistema, a SCI desempenha o papel de escravo CANopen, cabendo o papel de mestre à unidade UIS. Assim, estas unidades são compatíveis com o perfil de comunicações CANopen. Portanto, a arquitectura interna destas unidades encontra-se organizada na forma de perfis funcionais, nos quais se encontram especificadas as suas funções [3]. De facto, o modelo de sistema, discutido ao longo deste trabalho, aborda o sistema de gestão da iluminação recorrendo apenas a unidades cabladas deste tipo.

A capacidade de comunicações sem fios é facultativa nas SCIs. Como participantes nas redes sem fios, estas unidades são FFD possuindo, por conseguinte, a capacidade de adoptar o papel de coordenador de rede. O acesso aos objectos da aplicação é efectuado por meio do barramento CAN, pela interface local (I/O) ou através da rede sem fios, como mostra a figura 2.6 [3].

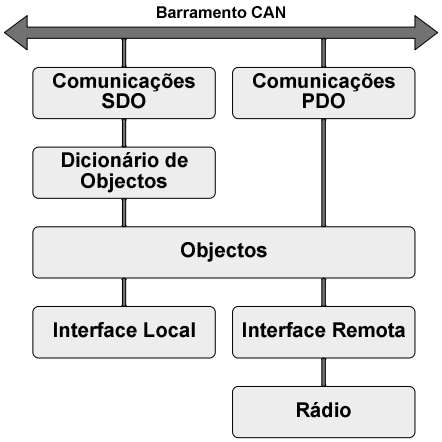

Fig. 2.6. Modelo de transferência de dados da subunidade de quadro [3].

2.2.2.2.2. As Unidades Remotas

As unidades remotas são prolongamentos das unidades de quadro que permitem a descentralização da acção destas e a expansão do sistema [3].

Neste sistema, estão presentes dois tipos de unidades remotas que se distinguem segundo as suas capacidades dentro da rede: (i) unidade remota FFD e (ii) unidade remota RFD.

A função principal da unidade remota FFD (SCWF) é aumentar o alcance da rede *wireless* coordenada por uma unidade SCI. Sempre que for necessário, esta unidade acumula as funções de *input* e *output* destinadas às unidades RFD. Neste caso, esta unidade assume a função de *router* de rede [3].

Esta unidade tem que estar permanentemente à escuta de modo a poder realizar o reencaminhamento das mensagens. Para que tal seja possível, é necessário que esteja permanentemente alimentada de energia eléctrica da rede.

A unidade remota RFD é a unidade com mais liberdade de localização. Esta unidade é autónoma sem fios e pode ter funções de medida, monitorização ou actuação, mas não pode ter a função de coordenador de rede [3].

2.2.3. Interacção entre Unidades Fixas e Móveis

Como pôde ser visto anteriormente, o sistema global, onde este modelo se integra, considera a utilização de unidades interligadas por dois meios físicos de transmissão de dados. As unidades localizadas nos quadros eléctricos são interligadas através de um barramento físico, enquanto que as unidades que necessitam estar perto dos equipamentos a monitorizar ou controlar são interligadas recorrendo a radiofrequência.

Hipoteticamente, se for considerada a integração deste sistema, apenas com capacidade de comunicação cablada, numa habitação, a sua instalação pode ser dispendiosa. No entanto, se a sua integração for considerada logo desde o planeamento de construção da habitação, a instalação torna-se significativamente mais acessível.

A partir do momento em que seja adicionada a *interface* de comunicações sem fios a este sistema, a sua instalação torna-se muito mais flexível, aumentando

significativamente a flexibilidade de distribuição espacial dos diversos dispositivos pela habitação, mesmo em habitações construídas previamente.

2.2.4. Os Protocolos de Comunicação

O protocolo de comunicações *Controller Area Network* (CAN) foi desenvolvido nos anos 80 por Robert Bosch GmbH [8], [9], [10], na Alemanha, numa tentativa de resolver os problemas de comunicação entre diferentes sistemas de controlo nos veículos. De facto, um dos primeiros usos do CAN foi a interligação de componentes de controlo como o sistema anti-bloqueio (ABS) e o sistema de controlo da aceleração (ASC). A complexidade das funções de controlo implementadas nestes sistemas requer mecanismos de transmissão de informação, que normalmente eram implementadas recorrendo a linhas de controlo dedicadas. Contudo, assim como a complexidade dos dispositivos vai aumentando, o número de linhas de controlo e ligações requeridas também vai aumentando. Isto conduziria à situação onde o número de linhas de controlo e de ligações, dos sistemas de controlo dos veículos da actualidade, ultrapassassem o seu limite físico máximo [8]. A ideia base por trás do CAN é ligar todos os dispositivos através de um único conjunto de cabos, minimizando o número de ligações, onde a informação possa ser inteiramente trocada, não apenas entre dois dispositivos mas entre numerosos dispositivos, em simultâneo.

Não levou muito tempo para que esta ideia migrasse dos veículos para os restantes mercados. Hoje em dia o CAN é encontrado em diversas áreas como na maquinaria agrícola, na instrumentação médica, em elevadores e nos sistemas de automação em ambiente industrial. Este uso generalizado ocorre, em certa parte, porque o protocolo apresenta características de funcionamento interessantes e porque o mercado automóvel é um mercado de massas, o que levou a que um grande número de produtores de semicondutores passassem a produzir controladores CAN tornando-os relativamente baratos. Só o facto de companhias como a Philips®, a Motorola®, a National Semiconductors®, a Siemens® e a Intel® estarem envolvidas no desenvolvimento de tecnologia CAN, é um forte indicador do sucesso desta tecnologia [8].

A especificação original desenvolvida por Robert Bosch GmbH é agora chamada *CAN Specification 2.0 – Part A* [8], [11]. A razão para isto foi a realização de um upgrade que contém as duas partes, a Parte A e uma nova versão do CAN chamada Parte B, sendo esta última compatível com a anterior [12]. A diferença principal entre as duas versões reside no facto de a versão anterior definir uma mensagem CAN com um identificador de 11 bits enquanto que a versão B permite usar um identificador de 29 bits.

Em 1993, o CAN passou a ser um standard ISO. O *ISO Draft International Standard* com referência ISO11898 está baseado no *CAN Specification 2.0* e adopta o formato clássico das mensagens CAN de identificadores de 11 bits. O *ISO11898 Standard* relaciona o CAN com o *ISO/OSI Reference Model*, especificando a *CAN Physical Layer* e a *CAN Data Link Layer* [8].

A *CAN Specification 2.0* pode ser caracterizada nas duas camadas mais baixas do *ISO/OSI Reference Model*: a *Physical Layer* e a *Data Link Layer*. A *CAN Data Link Layer* oferece serviços de transferência e requisição de informação, não maior que 8 bytes. Esses serviços são suficientes para o universo original das aplicações nas quais o CAN era usado. Contudo, nas aplicações distribuídas mais complexas, é requerida uma funcionalidade adicional, tal como a inicialização coordenada dos nós ou a transmissão de blocos de informação maiores que 8 bytes.

Em termos do *ISO/OSI Reference Model*, isso significa que são necessários serviços de camadas mais altas. Essa foi a razão pela qual a CiA (CAN in Automation) [13] especificou o *CAN Reference Model* e, dentro deste, a *CAN Application Layer* (CAL) [8]. O horizonte de aplicação do CAN foi muito alargado quando a CAL foi lançada e, nos dias de hoje, a CAL é usada numa ampla gama de campos de aplicação, desde aplicações médicas a controlo do trânsito.

O *CAN Reference Model* e a forma como ele se relaciona com o *ISO/OSI Reference Model* é mostrado na figura 2.7.

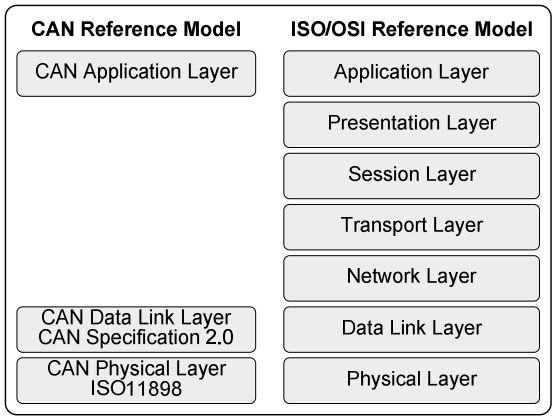

Fig. 2.7. O *CAN Reference Model* [8].

Como pode ser visto na figura 2.7, o *CAN Reference Model* é um modelo com três camadas [14], sendo estas a *Physical Layer*, a *Data Link Layer* e a *Application Layer*, o que é comum nos protocolos *fieldbus*.

A *Physical Layer* está encarregue de transformar ou codificar a informação passada para ela num sinal que possa ser enviado através do meio de transmissão. A ligação ao meio pode ser feita a uma taxa de transferência que pode atingir os 1000kbit/s num barramento com longitude não superior a 40 metros [15].

Independentemente do meio de transmissão utilizado, podem aparecer erros durante o processo de transmissão da informação. A função da *Data Link Layer* é lidar com esses erros e minimizar o número de erros enviados para as camadas superiores. Esta camada também lida com o problema de dois ou mais transmissores enviarem informação ao mesmo tempo. Estes reconhecem a colisão de informação e resolvem o problema de modo a que a informação não seja perdida.

No caso em que de dois ou mais dispositivos tentam transmitir ao mesmo tempo, é feita uma arbitragem baseada no identificador que as mensagens transportam. A prioridade é determinada analisando os bits dominantes do identificador da mensagem. Os dispositivos com menos prioridade interrompem a sua transmissão e passam a receptores. Com isto, as mensagens prioritárias são transmitidas e as menos prioritárias não são perdidas [15].

A *Application Layer* é a camada que o utilizador programa e o processo acede para comunicar com a rede. Normalmente é o único ponto de acesso disponível ao utilizador. Em termos gerais, é a interface entre o ambiente de comunicação de informação e a aplicação que usa esse ambiente para cooperar com outras aplicações.

O CANopen foi definido com base na CAN Application Layer. De facto, em redes CANopen, apenas é usado um subconjunto dos serviços especificados na CAL.

2.2.4.1. A Estrutura da Especificação CANopen

Um conceito muito importante quando se fala sobre CANopen é o conceito de perfil. Normalmente, um perfil define um subconjunto dos serviços prestados pelos protocolos existentes que podem ser usados para comunicação, e restringe a forma em como esses serviços podem ser aplicados.

A especificação CANopen é composta por um conjunto de perfis baseados no *CAN Reference Model*. Esses perfis são divididos na categoria *CANopen Device Profile* (perfil de dispositivo) e na categoria *CANopen Communication Profile* (perfil de comunicação).

O *CANopen Device Profile* define a forma como a funcionalidade de um tipo de dispositivo particular é tornada acessível através do barramento, e que ferramentas *CANopen Communication Profile* são necessárias para aceder a essa funcionalidade. Teoricamente, deveria haver um perfil de dispositivo para cada tipo de dispositivo que possa ser usado num ambiente industrial, ligado a um barramento CAN. Actualmente há alguns perfis estandardizados, tais como o *I/O Modules* (DS401), *Drives and Motion Control* (DS402), *Human Machine Interfaces* (DS403), entre outros. Também continuam a ser desenvolvidos outros perfis, pela CiA, para várias categorias de dispositivos que ainda não estão abrangidos pelas especificações já existentes [13].

O *CANopen Communication Profile*, DS301, é aplicável a todos os dispositivos. Define como um subconjunto de serviços da CAL pode ser usado para comunicar com um

dispositivo CANopen e como é esperado que esse dispositivo responda. É necessário que todos os dispositivos integrem o DS301.

A especificação CANopen relaciona estes conceitos com o *ISO/OSI Reference Model* definindo o *CANopen Reference Model* mostrado na figura 2.8.

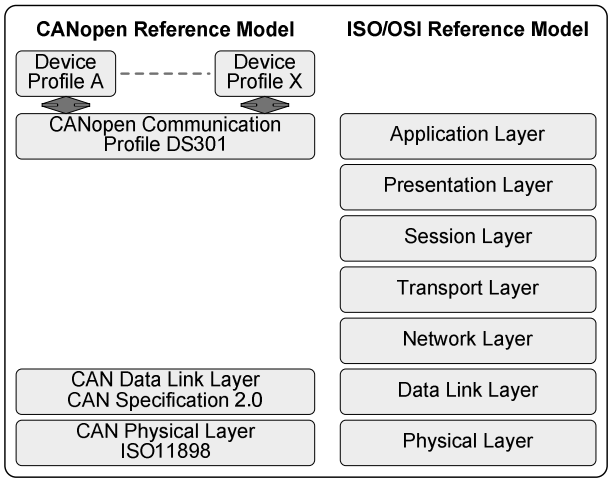

Fig. 2.8. O *CANopen Reference Model* [8].

O *CANopen Reference Model* é uma versão estendida do *CAN Reference Model*. De facto, o perfil CANopen permite o desenvolvimento de aplicações que usem os serviços da CAL para comunicar. Além disso, os dispositivos CANopen são obrigados a usar a CAL numa forma compatível, de modo a que o integrador do sistema tenha que escolher apenas os dispositivos certos e configurar algumas tarefas básicas, na construção da rede distribuída. Esta é uma vantagem importante do CANopen quando comparado com o CAL, onde o integrador do sistema tinha que gastar muito tempo a configurar a aplicação em termos dos conceitos da CAL.

Na figura 2.8 pode ser visto que o CANopen ultrapassa mesmo os limites definidos pelo *ISO/OSI Reference Model* entrando no domínio da aplicação do utilizador. De facto, o *CANopen Device Profile* pode ser visto como sendo uma camada adicional no topo do modelo de 7 camadas ISO/OSI. Alguns autores chamam esta camada de *User Layer* ou *Layer 8* [8].

O perfil CANopen define requisitos obrigatórios e requisitos opcionais. Os requisitos obrigatórios são essenciais para a compatibilidade e interoperacionalidade do CANopen e devem ser implementados em todos os dispositivos, desde o mais simples até o mais complexo. Os requisitos opcionais não têm de ser implementados nos dispositivos. Não são críticos para o funcionamento do sistema mas, se forem implementados, têm de ser segundo as especificações. Adicionalmente, um dispositivo tem de conter dados relativos ao produtor, que não são abrangidos pelos requisitos mencionados anteriormente.

2.2.4.2. A Funcionalidade da CANopen Application Layer

O perfil de comunicações CANopen divide a funcionalidade da *Application Layer* em numerosos *Service Objects*, em que cada um oferece uma funcionalidade específica. Quando uma aplicação invoca um serviço para comunicar com um dispositivo remoto, o *Service Object* que oferece o serviço referido vai interagir com um *Service Object* semelhante no nó remoto de modo a levar a cabo a operação de troca de informação.

A qualquer momento, um dispositivo CANopen pode estar envolvido em relações múltiplas de comunicação com vários nós da rede. Essas relações de comunicação podem pertencer a um de três modelos de comunicação básicos: (i) *Master/Slave*, (ii) *Client/Server* e (iii) *Producer/Consumer*.

No anexo A.5 está presente uma descrição destes três modelos de comunicação.

2.2.4.3. O Dicionário de Objectos

O modelo de um dispositivo CANopen contém dois tipos básicos de objectos:

- *Communication Objects* – cada um representa uma funcionalidade de comunicação específica do dispositivo. Os *Communication Objects* são especificados no perfil de comunicações CANopen.

- *Application Objects* – representam funcionalidades específicas do dispositivo tais como o estado de um sinal de input digital. Os *Application Objects* são definidos no perfil do dispositivo.

Os *Communication Objects* podem ser divididos em duas categorias segundo a sua funcionalidade:

- *Communication Objects for Network Management* – permite estabelecer comunicações *Master/Slave* e pode ser usado para controlo global de dispositivos e verificação do estado.
- *Communication Objects for Application Data Transfer* – estes podem ainda ser divididos em duas categorias:

    - Alguns objectos fornecem acesso indexado a todos os objectos do dispositivo através do OD, o que permite uma comunicação *Client/Server* com outro dispositivo. Usando estes *Communication Objects* é possível aceder a todas as características do dispositivo, normalmente para propósitos de configuração. Contudo, uma vez que este tipo de operação requer trocas de informação com o OD, o seu funcionamento não é adequado a um acesso em tempo real, pois sobrecarregaria o protocolo. Um exemplo deste tipo de *Communication Object* é o *Service Data Object* (SDO).
    - Outros objectos fornecem acesso directo aos *Application Objects* tornando possível implementar mecanismos de troca de informação síncrona ou assíncrona em tempo real. Este tipo de COBs segue a relação de comunicação *Producer/Consumer* envolvendo um congestionamento mínimo do protocolo. Os *Process Data Objects* (PDOs) são um exemplo deste tipo de mecanismo.

Todos os objectos de um dispositivo podem ser acedidos através do dicionário de objectos. O OD é a parte mais importante do modelo do dispositivo já que mapeia a estrutura interna do dispositivo: se a estrutura do OD de um determinado dispositivo é conhecida, então tudo o que é necessário para construir uma aplicação distribuída usando esse dispositivo é conhecido [8].

Cada objecto dentro do OD é endereçado com um índice de 16 bits e um sub-índice de 8 bits. Cada dispositivo na rede possui o seu dicionário de objectos que contem todos os parâmetros que descrevem o dispositivo. A estrutura geral de um dicionário de objectos é apresentada na tabela 2.1.

Tabela 2.1. Organização do dicionário de objectos [8].

Index	Object
0000	Não usado
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types
00A0-0FFF	Reservado
1000-1FFF	Communication Profile Area
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardised Device Profile Area
A000-FFFF	Reservado

2.2.4.4. Os Process Data Objects

Os PDOs (ou objectos de dados do processo) são utilizados para transmitir dados do processo em tempo real. Esta transferência de dados é uma ligação do tipo produtor/consumidor onde o produtor pode transmitir os dados para um ou mais consumidores. O conteúdo de cada PDO é definido previamente no dicionário de objectos e pode ser configurado recorrendo a mensagens do tipo SDO [15].

A transmissão dos PDOs pode ser assíncrona ou síncrona. Os PDOs síncronos são transmitidos com a recepção de um objecto de sincronização (SYNC). Dentro dos PDOs síncronos podem ainda existir os PDOs acíclicos e os cíclicos. Os PDOs assíncronos são transmitidos na ocorrência de um evento específico da aplicação ou na recepção de uma requisição remota.

2.2.4.5. Os Service Data Objects

Os SDOs (ou objectos de dados de serviço) permitem ligações do tipo cliente/servidor para o acesso às entradas do dicionário de objectos e permite a transferência de dados de tamanho superior a 8 bytes [15].

No anexo A.5 está presente uma descrição mais detalhada da especificação CANopen incluindo uma descrição mais detalhada da camada *Application Layer*, assim como do dicionário de objectos, dos SDOs e dos PDOs.

3. O MODELO DO SISTEMA

3.1. OS DISPOSITIVOS E AS SUAS FUNÇÕES

Sendo do âmbito deste trabalho a concepção de um modelo de sistema para a gestão do sistema de iluminação em espaços residenciais, que possibilite o aumento da liberdade dos habitantes sem, com isso, diminuir a flexibilidade de utilização e sem criar o sentimento de limitações físicas, para tal, foi necessário definir um determinado conjunto de modelos de dispositivos necessário à correcta operação do sistema.

Este modelo de sistema consiste num conjunto de dispositivos, com funções bem definidas (medida da luminosidade, accionamento de estores e toldos e de iluminação artificial), que comunicam entre si através de um barramento de comunicações, com base na norma de comunicações CANopen.

Em cada divisão da habitação existe um conjunto de dispositivos (actuadores e sensores) interligados e que estão incumbidos da gestão da iluminação dessa divisão. Cada divisão tem um comportamento independente embora essas unidades sejam parte do sistema global distribuído pela habitação.

Em resultado do levantamento das necessidades apresentadas por um sistema de gestão da iluminação foram definidos os modelos dos seguintes dispositivos:

- **Sensor de luminosidade exterior**: permite adquirir o nível de luminosidade exterior;
- **Sensor de luminosidade interior**: permite adquirir o nível de luminosidade interior de cada divisão da habitação;
- **Lâmpada de intensidade regulável**: é utilizada quando a luz natural não é suficiente para criar um ambiente confortável no interior da habitação;
- **Actuador de estores**: permite regular a intensidade luminosa no interior da habitação com base na luz natural;
- **Actuador de toldos**: permite proteger o interior da habitação dos raios solares directos e, consequentemente, reduzir a temperatura média da habitação por efeito da radiação solar. Assim, é possível aproveitar a luz natural nos dias quentes sem que para isso a habitação aqueça demasiado, reduzindo os consumos energéticos dos dispositivos de refrigeração;
- **Teclado de controlo manual de estores**: apenas é utilizado nas situações em que o utilizador deseja alterar a posição do estore contrariamente à acção do sistema;
- **Teclado de controlo manual de toldos**: apenas é utilizado nas situações em que o utilizador deseja alterar a posição do toldo contrariamente à acção do sistema;
- **Teclado de controlo manual de lâmpadas**: apenas é utilizado nas situações em que o utilizador deseja aumentar ou diminuir a intensidade luminosa definida pelo sistema.

Os sensores de luminosidade estão permanentemente a monitorizar o valor desta. Sempre que esta ultrapassa os valores definidos, comunica com os dispositivos actuadores de modo a que estes actuem restabelecendo os valores desejados. Com base nesses dados, os actuadores (de estores, de toldos e de lâmpadas) decidem as acções a tomar de modo a repor a luminosidade adequada. Neste sistema, cada um dos actuadores tem a capacidade de decidir sobre a acção a tomar em cada situação. Para isso, deve ser realizada uma parametrização adequada de modo a representar fidedignamente o meio envolvente.

Como ao longo do dia os desejos e as necessidades dos utilizadores vão alterando, é necessário ajustar os parâmetros de configuração do sistema de modo a que este se

adapte da melhor forma. Para tal, são definidos vários cenários de funcionamento que incorporam os parâmetros de configuração adequados às vontades e necessidades dos utilizadores em cada parte do dia.

Com isto, depois de realizada a configuração inicial, os utilizadores apenas terão de usufruir da comodidade da sua habitação relativamente à luminosidade. O sistema, automaticamente, ajustará a luminosidade óptima e o utilizador deixará de se ter que preocupar em acender o interruptor das lâmpadas, fechar os estores de noite ou desligar as lâmpadas ao sair de casa ou ao se deitar, ou mesmo fechar os estores nos dias mais quentes do Verão. Todas essas tarefas e mesmo outras são realizadas automaticamente pelo sistema sem qualquer tipo de intervenção do utilizador, concedendo assim mais tempo livre ao utilizador. Há sempre casos de excepção à rotina, em que o utilizador quer criar alguma condição que é contrária ao definido no sistema. Para isso, vão existir os teclados de controlo manual dos dispositivos como dos estores, lâmpadas ou toldos.

De seguida é apresentada uma descrição mais detalhada do conjunto de dispositivos que permite criar um sistema prático para o utilizador, com autonomia de funcionamento que, ao mesmo tempo, retire preocupações e dê um pouco de mais tempo livre ao utilizador, podendo ainda cooperar com os sistemas de gestão de outras funções.

3.1.1. Sensores de Luminosidade

Para que o sistema funcione correctamente é necessário medir a luminosidade interior e exterior à divisão com sensores. Será com base nos valores medidos e na vontade exercida pelo utilizador que o sistema irá decidir sobre os actuadores disponíveis para o efeito.

O sensor de luminosidade interior vai permitir medir a luminosidade interior e assim manter um nível confortável de luz no interior da habitação. O sensor de luminosidade exterior vai permitir distinguir principalmente o dia da noite possibilitando o fecho completo dos estores depois do crepúsculo e determinar quando é que os raios solares estão a incidir directamente no interior da divisão.

3.1.2. Accionadores

Para que o nível de luminosidade no interior da habitação se mantenha no nível desejado, devem estar presentes no sistema os dispositivos de actuação. Esses dispositivos, nesta primeira abordagem, são os accionadores de estores, accionadores de toldos e lâmpadas de intensidade regulável. Sempre que a luminosidade exterior o permita, a luz interior será apenas de origem natural pela abertura e fecho dos estores. Quando a luminosidade exterior não for suficiente, são accionadas as lâmpadas de intensidade regulável de modo a criar um ambiente confortável no interior da habitação.

Como apenas se está a tratar do sistema de iluminação, a presença de pessoas não é considerada, pois esta funcionalidade engloba-se no sistema de segurança. Para uma optimização do funcionamento da iluminação artificial, a presença de pessoas numa determinada divisão é um dado muito importante. Num trabalho futuro, onde seja desenvolvido o sistema de segurança, é importante introduzir esta informação nos controladores de luminosidade. Com isto, tornar-se-á possível accionar a iluminação artificial apenas na presença de pessoas.

Os accionadores de toldos têm a função de proteger o interior da habitação da incidência de raios solares directos. Para esta finalidade poderiam ser utilizados apenas os accionadores de estores, mas implicariam algumas limitações. Nos dias mais quentes poderiam ser fechados completamente os estores mas, neste caso, apenas poderia ser utilizada a iluminação artificial durante esse mesmo período. Com os toldos será criada uma sombra na envolvente das janelas, de modo a evitar a incidência de raios solares, trazendo a vantagem de poder continuar a utilizar a iluminação natural e, também por motivos mais estéticos e sensoriais, a visão para a paisagem exterior e/ou o acesso ao jardim não são restringidos.

Os accionadores de toldos poderiam operar em função não só da luminosidade exterior mas também em função da temperatura ambiente exterior. Assim, num trabalho futuro, estes accionadores poderiam receber também informações provenientes de um possível sistema de controlo da temperatura. Assim, o dispositivo poderia distinguir os dias frios dos dias quentes. Neste caso, os toldos apenas seriam abertos em dias de Verão muito

quentes. Nos dias mais frios os toldos permaneceriam fechados de modo a utilizar a energia radiante do sol para aquecer o interior da habitação.

3.1.3. Teclados de Controlo Manual

Os teclados de acção manual devem ser inseridos no sistema, para que as pessoas que habitam a casa não sintam as suas vontades limitadas. Estes dispositivos permitem accionar manualmente as lâmpadas, os estores e os toldos. O funcionamento automático dos dispositivos na divisão em questão só acontece se estes teclados estiverem numa posição automática. Se isso não acontecer, prevalece a vontade momentânea do utilizador. Em situações de rotina, os utilizadores apenas devem saber que o sistema está em funcionamento, não tendo que se preocupar com configurações, pressionar teclas ou ligar e desligar luzes. Depois da configuração inicial, ao utilizador apenas resta desfrutar do seu lar. A partir desse momento a habitação encarregar-se-á de gerir toda a iluminação, segundo vontades previamente estabelecidas.

3.2. OS MODELOS DOS DISPOSITIVOS

De seguida serão apresentados os modelos de todos os dispositivos incluindo os seus parâmetros de configuração. O modelo de um dispositivo representa a caracterização desse dispositivo através de: (i) um conjunto de parâmetros em que cada parâmetro representa uma opção de funcionamento distinto; (ii) funções que representam as acções associadas aos dispositivos; e (iii) *data points* que representam as entradas e saídas das funções dos dispositivos, permitindo a sua actuação em rede [1].

3.2.1. Modelo do Sensor de Luminosidade Interior

O sensor de luminosidade interior permite medir o nível de luz interior de modo a que o sistema opere em função desse valor, contribuindo dessa forma para manter um nível

confortável de luz no interior da habitação. Na figura 3.1 é possível ver o modelo do sensor de luminosidade interior com todos os parâmetros relativos à luminosidade.

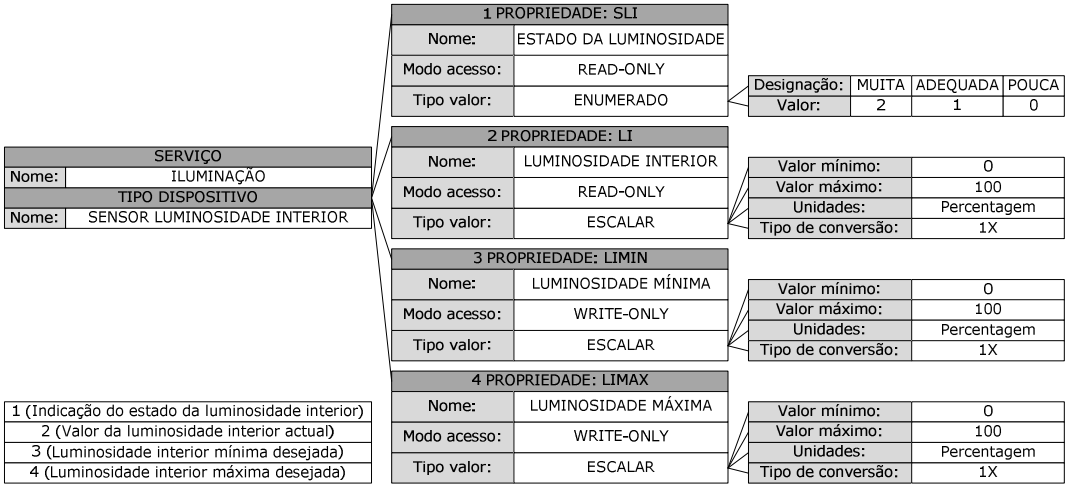

Fig. 3.1. Modelo do dispositivo sensor de luminosidade interior.

Para que o sensor de luminosidade interior desempenhe as suas funções de forma adequada, um dos parâmetros deste dispositivo deve ser o valor da luminosidade interior (LI). Essa luminosidade tem um valor mínimo e um valor máximo configurável pelo utilizador através dos parâmetros de configuração LIMIN e LIMAX. Esses valores correspondem ao intervalo luminoso considerado confortável pelo utilizador e, através deles, o dispositivo determina se está pouca ou muita luz ou se está a luz adequada no interior da divisão (corresponde ao parâmetro SLI). Este parâmetro é um *data point*, corresponde à saída deste dispositivo. O dispositivo comunica com o sistema sempre que o nível de luz na divisão em questão atingir valores fora desse intervalo. Deste modo é possível actuar nos dispositivos adequados, restabelecendo os valores de luminosidade confortáveis.

3.2.2. Modelo do Sensor de Luminosidade Exterior

Os sensores de luminosidade exterior permitem medir a intensidade luminosa exterior de modo a que o sistema opere em função desse valor. Na figura 3.2 é possível ver o modelo do sensor de luminosidade exterior com todos os seus parâmetros.

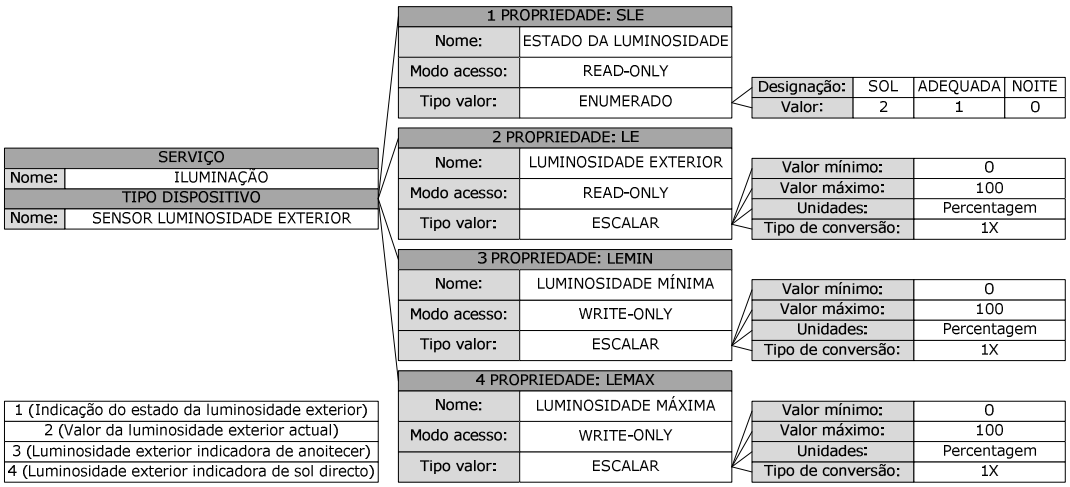

Fig. 3.2. Modelo do dispositivo sensor de luminosidade exterior.

O sensor de luminosidade exterior permite distinguir principalmente o dia da noite possibilitando o fecho completo dos estores depois do crepúsculo. Tal como o sensor de luminosidade interior, o sensor exterior também deve possuir uma propriedade correspondente ao nível de luz (LE). No entanto, os valores mínimo e máximo (LEMIN e LEMAX) têm significados ligeiramente diferentes. O valor mínimo de luminosidade identifica hipoteticamente a luminosidade à qual o utilizador quer identificar o crepúsculo e deseja que os estores sejam completamente fechados. O valor máximo de luminosidade corresponde à situação em que está a dar o sol directamente no dispositivo e, dependendo da localização dele, directamente para o interior da habitação.

O dispositivo compara a luminosidade exterior LE com os valores mínimo LEMIN e máximo LEMAX, definidos pelo utilizador, determinando um de três estados definidos no parâmetro de saída SLE. Se a luminosidade exterior é menor que o valor mínimo LEMIN o parâmetro SLE toma o valor 0 indicando que é NOITE, se a luminosidade é superior ao valor máximo LEMAX o parâmetro SLE toma o valor 2 indicando que o sol está a incidir directamente e se a luminosidade está entre os dois valores o parâmetro SLE toma o valor 1 que indica uma luminosidade normal no exterior.

3.2.3. Modelo do Accionador de Estores

Na figura 3.3 é possível ver uma esquematização do modelo do dispositivo accionador de estores com todos os parâmetros utilizados.

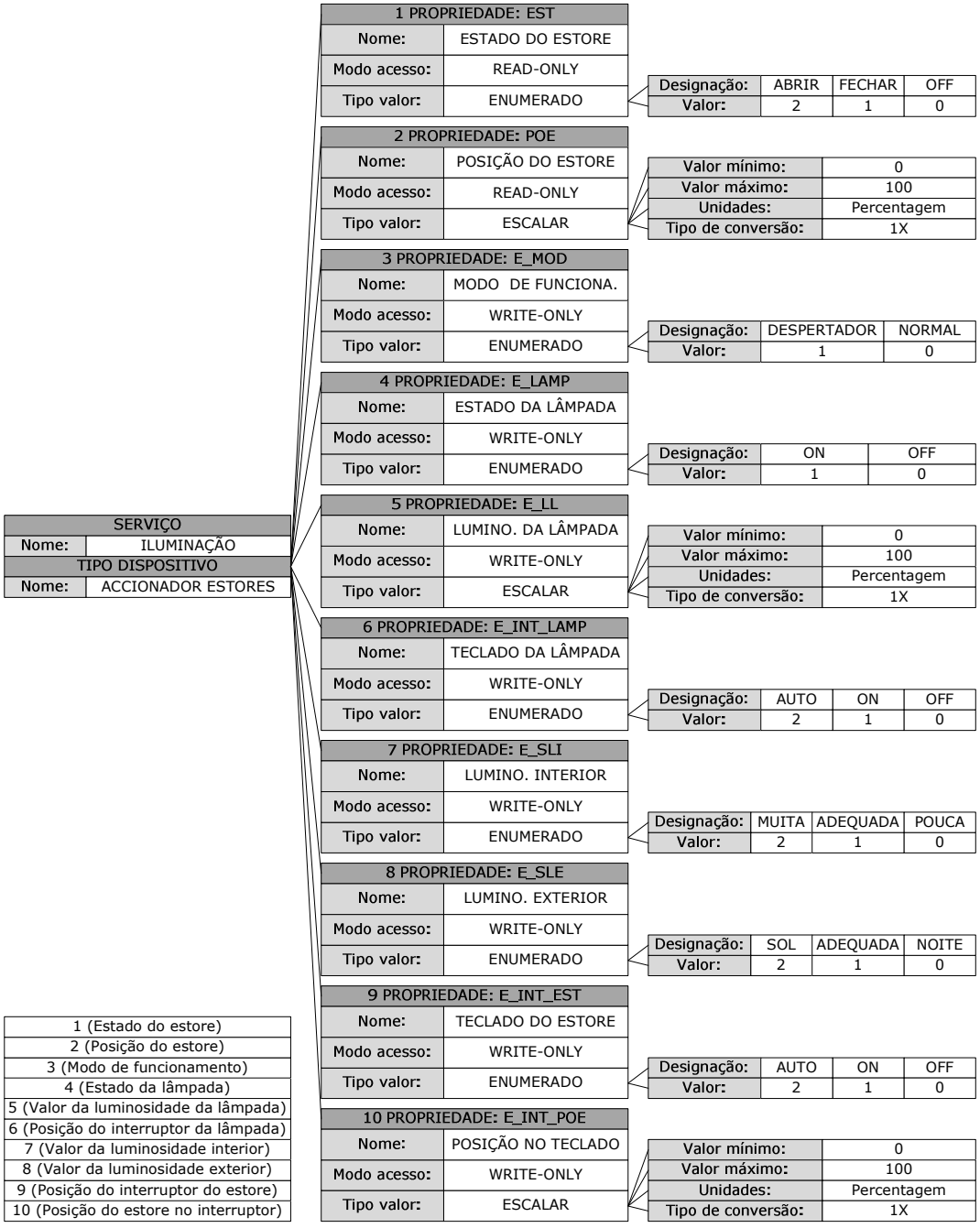

Fig. 3.3. Modelo do dispositivo accionador de estores.

Os accionadores de estores têm a finalidade de regular o nível de luz no interior da habitação com base na luz natural. Sempre que a luminosidade exterior não é suficiente, são utilizadas as lâmpadas de intensidade regulável para iluminar o interior da habitação. Isto permite fazer uma optimização dos recursos, reduzindo o consumo de energia eléctrica para fins de iluminação.

Em função da configuração realizada pelo utilizador e dos valores enviados pelos sensores de luminosidade, este actuador vai abrir ou fechar os estores da divisão em questão, de modo a manter uma luminosidade confortável.

A partir do momento em que a luz natural não é suficiente para criar o ambiente desejado pelo utilizador, este dispositivo passa a actuar em conjunto com os sistemas de iluminação artificial. Para um funcionamento adequado, no modelo do accionador de estores devem estar presentes vários parâmetros para flexibilizar o seu modo de funcionamento. Neste deve existir um parâmetro que defina o estado de funcionamento do estore e um parâmetro que dá a conhecer a sua posição. O parâmetro correspondente ao estado de funcionamento do estore designa-se de EST e tem três estados possíveis que são "Abrir", "Fechar" e "Off". O parâmetro correspondente à posição do estore designa-se POE e assume valores em percentagem, o seu valor mínimo corresponde a totalmente fechado e o seu valor máximo corresponde a totalmente aberto.

Para além destes dois parâmetros também é útil existir um parâmetro de configuração que permita definir vários modos de funcionamento no que diz respeito à velocidade de abertura e de fecho. Esse parâmetro é principalmente útil para poder abrir os estores dos quartos, no cenário LEVANTAR, de forma mais gradual, de modo a não aumentar a luminosidade interior bruscamente. A sua designação é E_MOD e tem dois estados possíveis. No estado DESPERTADOR o estore abre ou fecha em pequenos troços e no estado NORMAL o estore abre ou fecha de forma contínua até atingir a luminosidade adequada.

Para um funcionamento adequado, o dispositivo accionador de estores necessita destes parâmetros, que caracterizam ou seu próprio estado, mas também precisa de informação proveniente de outros dispositivos. São necessárias entradas (*data points*) que recebam informação das lâmpadas de iluminação artificial (E_LAMP e E_LL) e do seu teclado

de controlo manual (E_INT_LAMP). O parâmetro E_LAMP indica se as lâmpadas estão ligadas ou desligadas, o parâmetro E_LL indica a sua luminosidade em percentagem e o parâmetro E_INT_LAMP é proveniente do teclado de controlo manual das lâmpadas e serve para poder distinguir se o estado actual das lâmpadas é devido às condições do sistema ou devido à acção do utilizador. Estes três *data points* permitem que o accionador de estores funcione em conjunto com as lâmpadas. Assim é possível começar a fechar os estores apenas depois de as lâmpadas estarem desligadas, nos casos em que é necessário diminuir a luminosidade.

Também devem ser incluídas duas entradas correspondentes à luminosidade interior (E_SLI) e à luminosidade exterior (E_SLE). O parâmetro E_SLI indica se a luminosidade interior é excessiva, insuficiente ou a adequada. Desta forma é possível posicionar os estores de modo a regular o valor da luminosidade interior. A entrada E_SLE permite distinguir se é noite, se está a incidir sol no interior da habitação ou se as condições são adequadas. Com este parâmetro é possível fazer com que os estores fechem completamente assim que anoitecer.

Para um funcionamento manual, também é necessário introduzir as entradas correspondentes ao estado de configuração do teclado de controlo manual dos estores. Essas entradas representam o estado definido no teclado (E_INT_EST) e a posição do estore (E_INT_POE) em modo manual. O parâmetro correspondente ao estado de operação possui três estados possíveis que são: o automático (AUTO), o ligado (ON) e o desligado (OFF). O parâmetro que define a posição do estore assume valores em percentagem. Se o parâmetro E_INT_EST apresenta o estado AUTO, o estore vai funcionar normalmente mas se este não se encontra nesse estado, vai permanecer na posição definida em E_INT_POE.

3.2.4. Modelo do Accionador de Toldos

Na figura 3.4 é possível ver uma esquematização do modelo do dispositivo accionador de toldos com todos os parâmetros que o caracterizam.

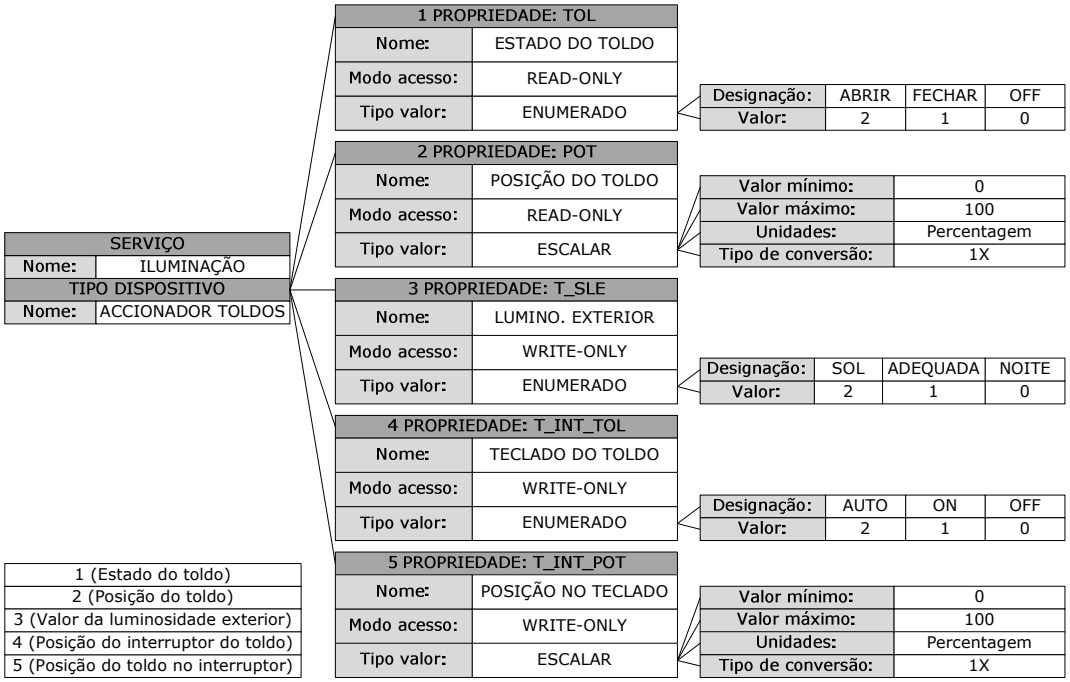

Fig. 3.4. Modelo do dispositivo accionador de toldos.

O accionador de toldos tem a finalidade de protecção dos raios solares directos e, a consequente redução das transferências de energia térmica através das janelas da habitação. Para tal, este accionador abre e fecha os toldos colocados sobre as janelas da habitação em função da incidência de raios solares.

Tal como no caso do estore, deve existir um parâmetro que defina o estado de funcionamento do toldo e um parâmetro que defina a sua posição. O parâmetro correspondente ao estado de funcionamento do toldo designa-se TOL e assume três estados possíveis que são “Abrir”, “Fechar” e “Off”. O parâmetro correspondente à posição do toldo designa-se POT e assume valores em percentagem. O seu valor mínimo corresponde a totalmente fechado e o seu valor máximo corresponde a totalmente aberto.

Também deve ser introduzida uma entrada correspondente à luminosidade exterior, o parâmetro T_SLE. Sempre que a T_SLE indica que o sol está a incidir no interior da habitação, o toldo vai abrir completamente de modo a criar sombra no entorno da janela e assim a divisão não aquece excessivamente. Caso contrário, o toldo permanece fechado.

Para um funcionamento manual, também é necessário incorporar uma entrada que receba a configuração do teclado de controlo manual dos toldos. As entradas necessárias são o estado definido no teclado (T_INT_TOL) e a posição do toldo (T_INT_POT) em modo manual. Tal como no caso anterior, o parâmetro correspondente ao estado assume três estados possíveis que são o automático (AUTO), o ligado (ON) e o desligado (OFF). A entrada que recebe a posição do toldo assume valores em percentagem. Se a entrada T_INT_TOL assume o valor AUTO, o toldo vai funcionar normalmente mas se esta não estiver nesse estado, irá permanecer na posição definida em T_INT_POT.

3.2.5. Modelo da Lâmpada de Intensidade Regulável

Na figura 3.5 é possível ver uma esquematização do modelo do dispositivo da lâmpada de intensidade regulável com todos os parâmetros e *data points* que o caracterizam.

As lâmpadas de intensidade regulável são utilizadas sempre que a luz natural não é suficiente para criar um ambiente confortável no interior da habitação. Segundo isto, as lâmpadas vão começar a acender quando os estores estiverem já completamente abertos e a luz natural no interior continuar a ser insuficiente. Assim, o dispositivo que controla a lâmpada de intensidade regulável deve conhecer a posição do estore através da entrada L_POE.

A posição do estore não é suficiente para conseguir um funcionamento adequado da lâmpada de intensidade regulável. É necessário também conhecer o estado do estore definido no teclado de controlo manual dos estores (L_INT_EST) de modo a poder identificar se a posição do estore está segundo as condições do sistema, ou se é por vontade momentânea do utilizador. Esta função é útil numa situação em que o utilizador deseja manter o estore numa determinada posição fixa. Assim, ao longo do crepúsculo, as lâmpadas são activadas mesmo que o estore não esteja completamente aberto (ou fechado durante a noite).

Deve existir um parâmetro que defina o estado de funcionamento das lâmpadas (LAMP) e um parâmetro que dá a conhecer a sua intensidade luminosa actual (LL). O parâmetro correspondente ao estado de funcionamento assume um de dois estados possíveis que

são “ON” e “OFF”. O parâmetro correspondente à intensidade luminosa assume valores em percentagem, o seu valor mínimo corresponde a totalmente apagada e o seu valor máximo corresponde à intensidade luminosa máxima.

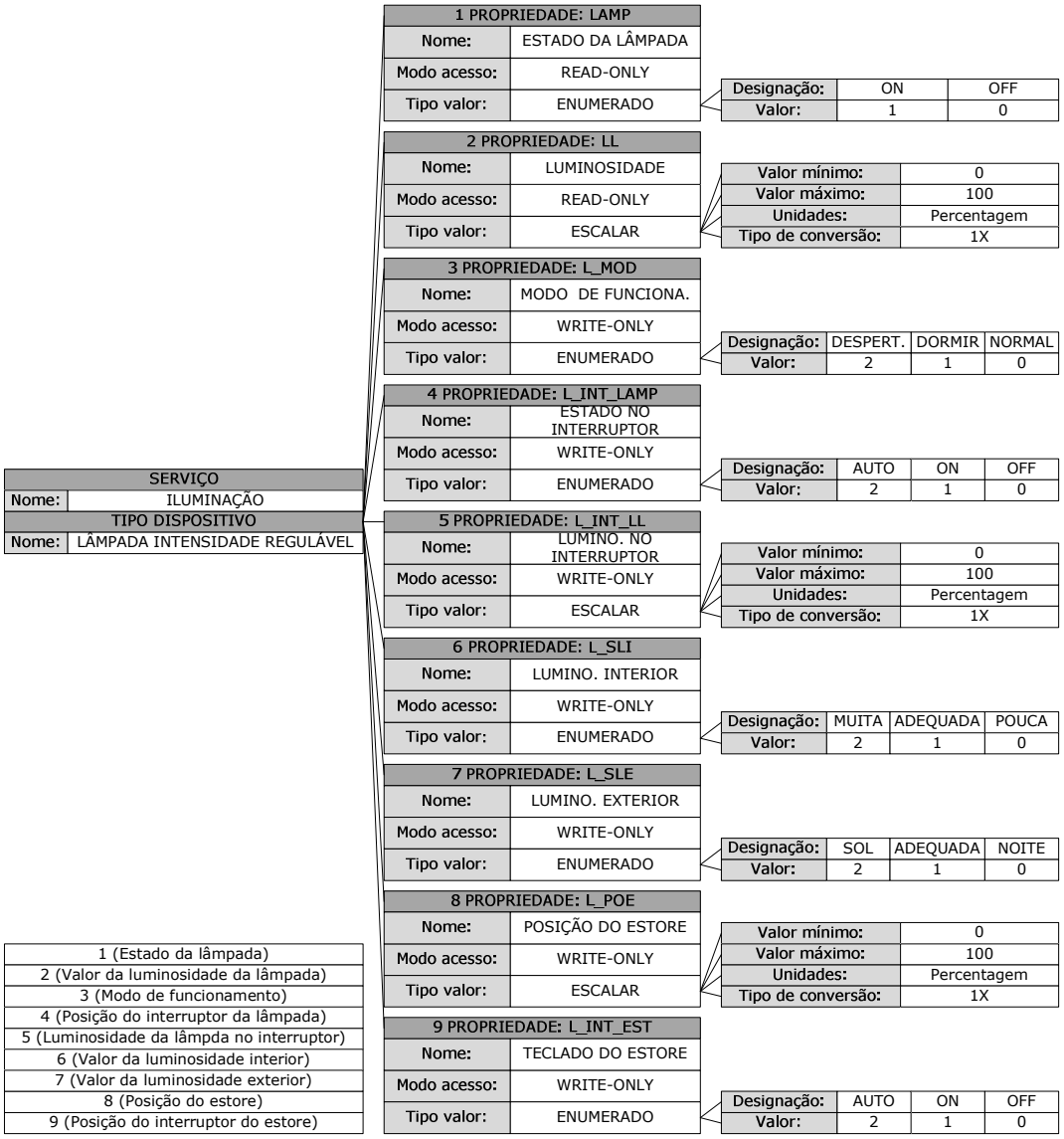

Fig. 3.5. Modelo do dispositivo lâmpada de intensidade regulável.

Para além destes dois parâmetros também é proveitoso existir um parâmetro que permita definir vários modos de funcionamento no que respeita à variação de luminosidade das lâmpadas. Esse parâmetro é principalmente útil para permitir acender as lâmpadas dos quartos, no cenário LEVANTAR, de forma mais gradual, de modo a não aumentar a luminosidade bruscamente. A sua designação é L_MOD e assume um

de três estados possíveis. No estado DESPERTADOR, a lâmpada aumenta ou diminui a sua intensidade luminosa de forma gradual em pequenos intervalos. No estado NORMAL, esta aumenta ou diminui a sua luminosidade de forma contínua até atingir a luminosidade adequada. No estado DORMIR, esta desliga as lâmpadas dos quartos de forma gradual, ao fim de um determinado tempo em que seja detectada a presença de alguém na cama e ninguém nas outras zonas do quarto. Neste último caso, o dispositivo deveria receber as informações do sistema de segurança de modo a optimizar o funcionamento do sistema. Como apenas se está a tratar o sistema de gestão da iluminação, esse assunto não é considerado.

Devem ser incluídas duas entradas correspondentes à luminosidade interior e à luminosidade exterior (L_SLI e L_SLE). A entrada L_SLI indica se a luz no interior da divisão é excessiva, reduzida ou a adequada. Se a luminosidade interior é insuficiente, a lâmpada vai aumentar o seu brilho de modo a restabelecer uma luminosidade interior adequada, se esta é excessiva, a lâmpada vai diminuir o seu brilho. A entrada L_SLE é utilizada para indicar se é noite ou dia. Se é noite, as lâmpadas vão funcionar de forma independente dos estores, pois estes vão permanecer fechados, se é dia, ambos vão trabalhar em conjunto para regular a intensidade luminosa no interior da divisão.

Para um funcionamento manual, também é necessário introduzir informação relativa à configuração do teclado de controlo manual da iluminação artificial. Essas entradas são o estado de funcionamento seleccionado no teclado (L_INT_LAMP) e a intensidade luminosa da lâmpada (L_INT_LL) no modo manual. A entrada correspondente ao estado assume um de três estados possíveis que são: o automático (AUTO), o ligado (ON) e o desligado (OFF). A entrada referente à intensidade luminosa da lâmpada assume valores em percentagem do brilho máximo. Se a entrada L_INT_LAMP assumir o estado AUTO, a lâmpada vai funcionar normalmente mas se esta não se encontrar nesse estado, vai funcionar com a intensidade luminosa fixa e definida em L_INT_LL.

3.2.6. Modelos dos Teclados de Controlo Manual

São propostos dispositivos de controlo manual para os actuadores de estores, toldos e lâmpadas, com o intuito de facultar maior liberdade de opção às pessoas que habitam a

casa. O funcionamento automático dos dispositivos só acontece se estes teclados estiverem no estado automático. Se isso não acontecer, prevalece a vontade momentânea do utilizador. Estes dispositivos são dispositivos com as teclas e funções indicadas na tabela 3.1.

Tabela 3.1. Teclas de cada teclado de controlo manual.

Lâmpada de intensidade regulável	**Accionador de estores**	**Accionador de toldos**
ON	ON	ON
OFF	OFF	OFF
AUTO	AUTO	AUTO
LUM. MAIS	POS. MAIS	POS. MAIS
LUM. MENOS	POS. MENOS	POS. MENOS

3.2.6.1. Modelo do Teclado de Controlo Manual de Lâmpadas

Na figura 3.6 está esquematizado o modelo do teclado de controlo das lâmpadas. O teclado de controlo manual das lâmpadas de intensidade regulável é constituído por cinco teclas. Existem três teclas com as funções ON, OFF e AUTO que permitem ligar/desligar a lâmpada ou coloca-la no modo automático, permitindo accionar as lâmpadas automaticamente em função dos valores registados pelos sensores de intensidade luminosa e da configuração realizada. Também possui duas teclas que são o LUM. MAIS e o LUM. MENOS que permitem aumentar ou diminuir a luminosidade quando as lâmpadas de intensidade regulável se encontram no modo manual. Para operar este teclado, normalmente pressionam-se as teclas LUM. MAIS ou LUM. MENOS até definir o nível de luz desejado e de seguida acciona-se o botão ON para ligar as lâmpadas, para as desligar pressiona-se a tecla OFF. Nestes dois casos, as lâmpadas da divisão permanecem no estado definido pelo utilizador até que seja premida a tecla AUTO ou seja carregado um novo cenário de funcionamento.

Quanto ao modelo do dispositivo, este deve conter uma saída, *data point*, que assume o estado de funcionamento desejado da lâmpada (INT_LAMP) e uma saída com a intensidade luminosa desejada (INT_LL). A saída correspondente ao estado de

funcionamento seleccionado no teclado assume um de três estados possíveis que são "ON", "OFF" e "AUTO". A saída correspondente à intensidade luminosa pretendida no modo manual assume valores em percentagem da intensidade máxima disponível, o seu valor mínimo é 0 e corresponde à lâmpada apagada e o seu valor máximo é 100 e corresponde à intensidade luminosa máxima da lâmpada ou sistema de iluminação.

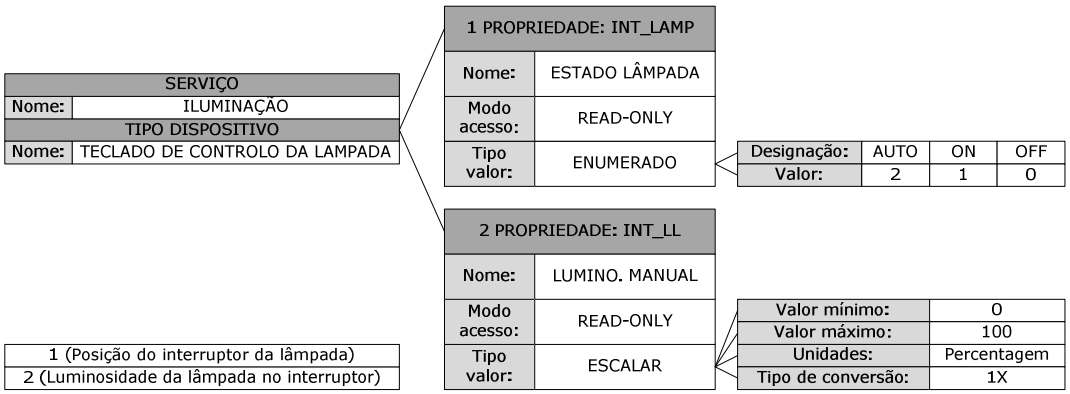

Fig. 3.6. Modelo do teclado de controlo manual das lâmpadas.

3.2.6.2. Modelo do Teclado de Controlo Manual de Estores

Na figura 3.7 é está esquematizado o modelo do teclado de controlo de estores. O teclado de controlo manual dos estores também possui cinco teclas. Possui três teclas associadas à saída INT_EST, com os estados ON, OFF e AUTO e que possibilitam ligar/desligar o accionamento ou coloca-lo no modo automático. Também existem mais duas teclas, associadas à saída INT_POE, que são a POS. MAIS e a POS. MENOS. Estas permitem definir uma posição mais aberta ou mais fechada dos estores quando o sistema se encontra no modo manual. Esta saída assume valores em percentagem e o seu valor mínimo é 0, que corresponde a totalmente fechado e o seu valor máximo é 100, que corresponde a totalmente aberto.

Para operar este teclado, normalmente pressionam-se as teclas POS. MAIS ou POS. MENOS até definir a posição desejada na saída INT_POE e de seguida pressiona-se a tecla ON para activar o accionamento dos estores. Os estores começam a deslocar-se até à posição desejada e, assim que atingem a posição pretendida, o sistema assume o

estado de funcionamento OFF. Para desligar o accionamento em qualquer momento, deve ser pressionada a tecla OFF. Para que o teclado reconheça o posicionamento correcto do estore, este deve possuir informação sobre o estado de funcionamento do accionador, bem como conhecer a sua posição. São definidas as entradas referentes ao estado de funcionamento (FEED_EST) e à sua posição actual (FEED_POE). Só assim é possível fazer com que a saída INT_EST do teclado passe do estado ON para o estado OFF quando o estore atingir a posição definida no teclado. Os estores permanecem no estado definido pelo utilizador até que seja premida a tecla AUTO.

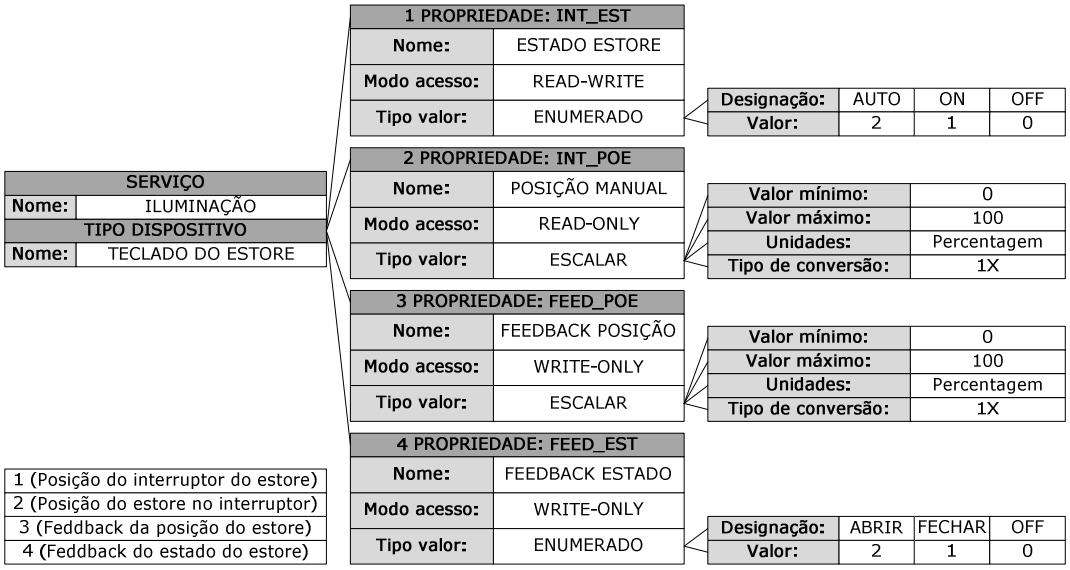

Fig. 3.7. Modelo do teclado de controlo manual dos estores.

3.2.6.3. Modelo do Teclado de Controlo Manual de Toldos

O modelo do teclado de controlo manual dos toldos está representado na figura 3.8. Este também é constituído por cinco teclas, tal como os outros modelos de teclados. Três teclas estão associadas à saída INT_TOL, que assume os valores ON, OFF e AUTO. Esta permite ligar/desligar o accionamento ou coloca-lo no modo automático para o controlo automático dos toldos, em função dos valores recolhidos pelos sensores e da configuração realizada. Também existem duas teclas associadas à saída INT_POT. Estas são o POS. MAIS e o POS. MENOS e permitem definir uma posição mais aberta ou mais fechada dos toldos, quando o teclado está em modo manual. Esta saída assume

valores em percentagem. O seu valor mínimo é 0, que corresponde à posição totalmente fechado e o seu valor máximo é 100, que corresponde à posição totalmente aberto.

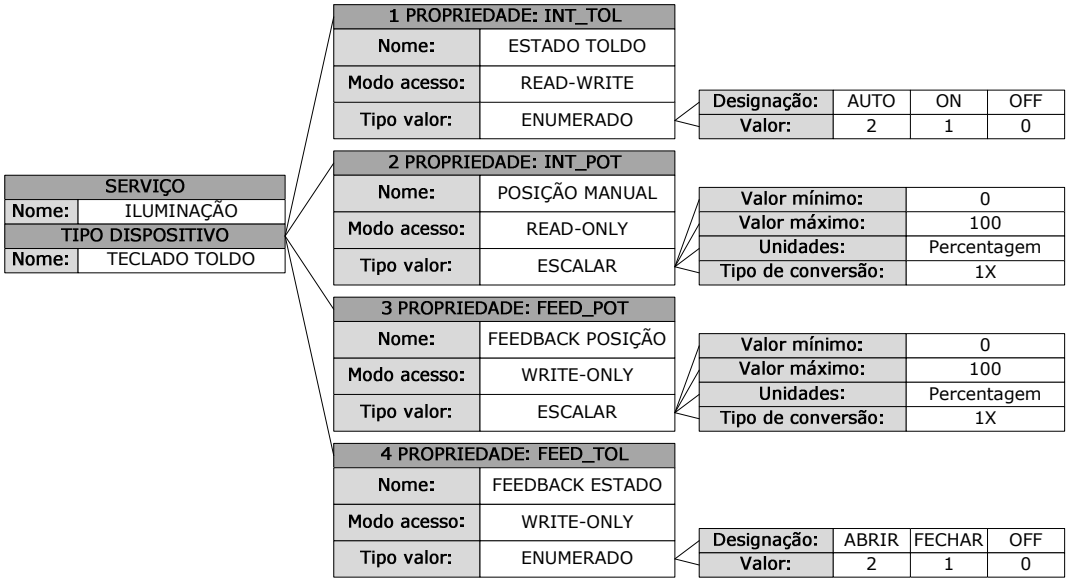

Fig. 3.8. Modelo do teclado de controlo manual dos toldos.

Para operar este teclado, normalmente pressionam-se as teclas POS. MAIS ou POS. MENOS até definir a posição desejada na saída INT_POT e de seguida pressiona-se a tecla ON para activar o accionamento dos toldos. Os toldos vão começar a deslocar-se para a posição desejada e, assim que chegam à posição pretendida, o modo de funcionamento transita automaticamente para o estado OFF. Para desligar o accionamento a qualquer momento, carrega-se na tecla OFF. Para que o teclado reconheça o posicionamento correcto do toldo, este deve possuir informação sobre o estado de funcionamento do accionador, bem como conhecer a sua posição. São definidas as entradas referentes ao estado de funcionamento (FEED_TOL) e à sua posição actual (FEED_POT). Só assim é possível fazer com que o teclado passe do estado ON para o estado OFF quando o toldo atinge a posição definida no teclado. Os toldos vão permanecer no estado definido pelo utilizador até que seja premida a tecla AUTO.

4. OS CENÁRIOS DE FUNCIONAMENTO

4.1. OS CENÁRIOS

Todos os indivíduos têm as suas preferências pessoais e, como tal, cada um deles tem as suas exigências e desejos relativamente ao que considera conforto na sua habitação. No entanto, considerando um grupo maioritário de indivíduos activos, existe um conjunto de rotinas diárias que segue um padrão que é comum a todos esses indivíduos.

Geralmente, a rotina diária de uma família comum consiste em levantar de manhã cedo, preparar-se e tomar o pequeno-almoço e, posteriormente, os pais vão para os seus empregos e os filhos mais novos vão para a escola, ficando a casa vazia. Mais tarde, voltam para casa, realizam as suas actividades pessoais, como *hobbies* ou afazeres domésticos, e jantam. A última parte da rotina diária chega com a hora de dormir e tudo recomeça com a manhã seguinte.

Estas rotinas permitem definir, à partida, quatro cenários gerais de funcionamento correspondentes ao período de LEVANTAR, ao período em que a estão TODOS FORA, ao período em que regressam a casa, passando a haver ALGUEM EM CASA, e, mais tarde chega a hora de DORMIR. Cada um destes cenários corresponde a um conjunto global de parâmetros de configuração de todos os dispositivos do sistema e representa uma forma distinta de funcionamento do sistema.

Cada um destes cenários é armazenado e gerado numa unidade concebida para tal e é descarregado para o sistema por acção de um evento específico, por acção de um relógio ou simplesmente por vontade própria do utilizador

É claro que nem sempre uma família segue uma rotina e, mesmo numa família que siga este padrão, pode não o seguir todos os dias do ano, por exemplo em período de férias ou noutras situações em que nem toda a gente sai de casa logo pela manhã ou sai de noite e entra já depois da hora de se deitar. Muitas vezes também acontecem situações em que estes cenários não se adaptam completamente. Para ultrapassar esses e outros pontos e para não transmitir aos habitantes um sentimento de limitação, existem os teclados no qual o utilizador pode accionar manualmente as luzes, os estores ou os toldos da divisão pretendida. Neste caso, o dispositivo em questão entra em modo de funcionamento manual até que o utilizador volte a colocá-lo em modo automático. As modificações de operação manuais que o utilizador realiza numa divisão não vão afectar o funcionamento do sistema nas outras divisões, isto é, não alteram o cenário instalado.

No texto seguinte são apresentados os cenários propostos com maior pormenor. Em cada cenário são mostrados os dispositivos e o seu modo de funcionamento. Contudo, apenas é apresentado um dispositivo de cada género. Por exemplo, no cenário LEVANTAR é apresentado um accionador de estores mas em toda a casa há inúmeros accionadores de estores. Alguns destes dispositivos vão funcionar de forma semelhante mas outros podem assumir diferentes comportamentos.

Neste caso, imaginemos o accionador de estores de um quarto e o accionador de estores da cozinha. No cenário LEVANTAR, o accionador de estores do quarto vai abrir às parcelas em intervalos de tempo previamente definidos de modo a aumentar a luminosidade gradualmente no quarto e assim as pessoas acordarem sem sobressalto. O accionador de estores da cozinha não necessita de abrir da mesma forma, podendo abrir de forma contínua até atingir a luminosidade pretendida na cozinha. Para tal, apenas existe uma diferença de configuração num parâmetro de funcionamento no cenário.

4.1.1. O Cenário Levantar

Iniciando a exposição pelo cenário "Levantar", presente na figura 4.1, este é hipoteticamente descarregado para todos os dispositivos a uma determinada hora, definida previamente pelos utilizadores. Por acção deste cenário, e tentando explicar sucintamente o funcionamento geral do sistema, à hora programada pelo utilizador os estores dos quartos começam a abrir lentamente de modo a entrar a claridade do dia. Se, por algum motivo, ainda não houver luz suficiente no exterior, são accionadas as lâmpadas com uma luminosidade confortável para os habitantes acordarem.

Através do parâmetro E_MOD, o accionador de estores vai entrar no modo despertador que consiste em abrir o estore às porções, em intervalos de tempo definidos. O estore vai abrir deste modo até a luminosidade interior LI atingir o valor mínimo LIMIN definido no sensor de luminosidade interior.

No caso de a luz natural não ser suficiente para a luminosidade interior atingir o valor mínimo definido, o estore vai atingir a sua posição de abertura máxima e, a partir desse momento, as luzes vão encetar o aumento de luminosidade, através do parâmetro LL, até que a luminosidade interna mínima seja atingida.

Se ainda não tiver amanhecido, a luminosidade exterior LE é inferior à mínima definida no parâmetro LEMIN e assim, o estore permanece fechado. Neste caso apenas são accionadas as lâmpadas aumentando a luminosidade lentamente em intervalos de tempo definidos. Para as lâmpadas aumentarem de luminosidade deste modo, devem estar em modo despertador através do parâmetro L_MOD.

Para que este sistema funcione na sua forma óptima, deve interagir com outros sistemas presentes na habitação. No caso das lâmpadas, se o sistema de segurança indicar que não existem habitantes numa determinada divisão, estas podem simplesmente permanecer desligadas.

Depois do nível de luz no interior atingir o valor pretendido, o sistema não fica estático. É empírico que, na maioria dos dias ao longo da manhã, a luminosidade no exterior tende a aumentar. Com isto, a luminosidade no interior da habitação vai continuar a

aumentar. Assim, é possível que o seu valor ultrapasse o valor máximo LIMAX definido pelo utilizador. Neste caso, se as lâmpadas estiverem ligadas, estas vão diminuir o seu brilho até o valor de LI ficar abaixo de LIMAX. Se, mesmo depois de as lâmpadas estarem desligadas, a luminosidade interior continuar acima do valor máximo, os estores vão começar a fechar até que LI fique dentro do intervalo definido pelos valores máximo e mínimo (LIMAX e LIMIN).

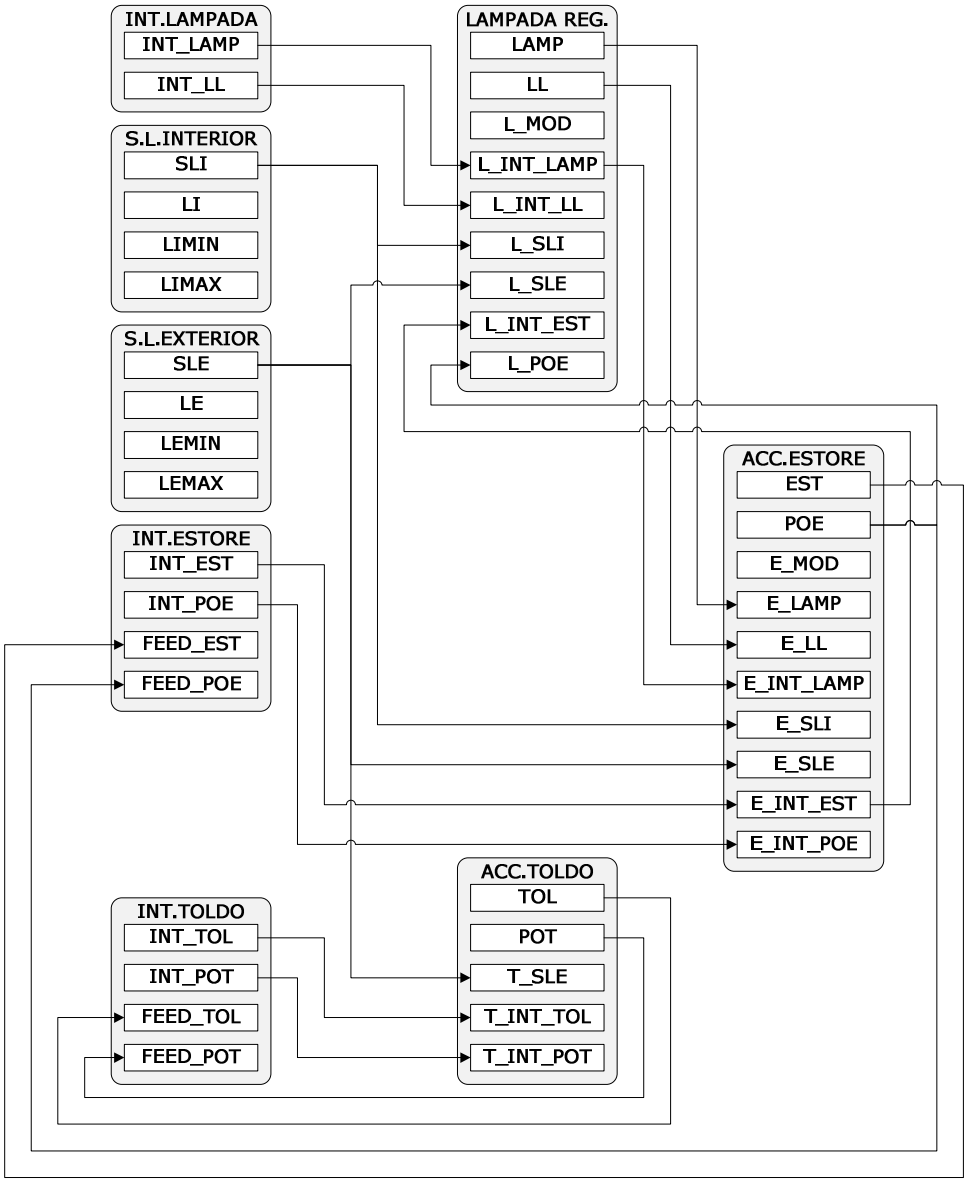

Fig. 4.1. Representação dos cenários Levantar e Alguém em Casa.

Para estes dois dispositivos funcionarem em conjunto, é necessário que a lâmpada conheça a posição do estore L_POE e o estore conheça a luminosidade E_LL e o estado da lâmpada E_LAMP. Mas isto não é suficiente para que ambos funcionem adequadamente. Para que isso aconteça, e devido ao facto de ambos terem a possibilidade de funcionarem manualmente, é necessário que ambos conheçam se o outro dispositivo está em modo manual ou automático através dos parâmetros L_INT_EST e E_INT_LAMP. Assim, se o estore permanece numa posição determinada pelo utilizador, a lâmpada vai continuar a funcionar automaticamente ou se a lâmpada está com uma luminosidade fixa devido à vontade do utilizador, o estore pode continuar a funcionar automaticamente.

O estore funciona de forma manual por acção do teclado de controlo manual que lhe transmite os parâmetros de estado e posição desejadas pelo utilizador, E_INT_EST e E_INT_POE. O utilizador define a posição desejada, pressiona a tecla ON e o estore movimenta-se até à posição definida. Assim que chega a essa posição, o teclado passa para o estado OFF.

No caso da lâmpada, o seu teclado de controlo manual transmite-lhe os parâmetros L_INT_LAMP e L_INT_LL que correspondem ao estado e luminosidade desejados pelo utilizador. O utilizador pode definir a luminosidade desejada e pressionar a tecla ON. A partir desse momento, a lâmpada vai começar a aumentar ou diminuir a luminosidade de forma progressiva até o valor pretendido. Neste caso o teclado permanece no estado ON até que o utilizador pressione a tecla OFF para desligar as lâmpadas ou então coloque em estado automático através da tecla AUTO.

Quanto ao accionador de toldos, este dispositivo não tem uma função de despertar concreta. Este dispositivo funciona apenas em função do parâmetro T_SLE. O toldo permanece fechado até este parâmetro de entrada indicar que a luz solar está a incidir directamente no interior da habitação. Quando tal acontece, o toldo abre completamente de modo a criar sombra no entorno das janelas onde o sol estava a incidir e assim evitar o aquecimento excessivo da divisão. Tal como o accionador de estores, este dispositivo também pode funcionar de forma manual através do teclado de controlo manual. O utilizador, define a posição que deseja no teclado, pressiona a tecla ON e o toldo

posiciona-se na posição desejada. Quando o toldo atinge a posição desejada, o teclado passa para o estado OFF.

Num possível trabalho futuro, em que seja desenvolvido um sistema de controlo da temperatura, o accionador de toldos poderia funcionar em função das temperaturas no interior e no exterior da habitação. Sempre que estiver frio no interior da habitação, é útil aproveitar a radiação solar para a aquecer. Neste caso, mesmo que o sol esteja a incidir directamente na janela, seria útil que o toldo permanecesse fechado, permitindo que a radiação solar ajudasse a aquecer a habitação. O toldo permaneceria fechado até que a temperatura no interior da habitação estivesse acima do valor mínimo definido. Além disso, é importante também que este accionador conheça a velocidade do vento pois, nos dias mais ventosos, os toldos correriam o risco de se danificarem quando abertos.

4.1.2. O Cenário Alguém em Casa

Assim que toda a gente está levantada, é descarregado para o sistema o cenário "Alguém em Casa", representado na figura 4.1. A representação esquemática deste cenário é idêntica à representação do cenário "Levantar", contudo, existem algumas diferenças relativamente aos valores de alguns parâmetros. Os accionadores de estores e as lâmpadas de intensidade regulável vão passar ao modo de funcionamento normal por efeito da alteração dos parâmetros E_MOD e L_MOD, respectivamente. Neste caso, as lâmpadas vão passar a alterar a sua luminosidade de forma contínua e os estores vão passar a abrir e fechar também de forma contínua.

Resumidamente, este cenário vai fazer com que o sistema actue os estores e as lâmpadas da habitação de modo a haver uma luminosidade confortável dentro da habitação. Sempre que a luminosidade interior LI está fora do intervalo de valores definidos por LIMIN e LIMAX (que podem ser diferentes dos valores definidos para o cenário LEVANTAR), os estores vão abrir ou fechar conforme LI esteja muito baixa ou muito alta, respectivamente. Sempre que o estore atingir a posição de abertura máxima e LI continuar abaixo do valor mínimo, vão entrar em funcionamento as lâmpadas de intensidade regulável, mas apenas quando houver alguém na divisão (como já foi

mencionado, esta funcionalidade é conseguida em cooperação com o sistema de segurança). Estas vão aumentar a sua luminosidade LL até o valor de LI ultrapassar LIMIN.

O caso oposto também é possível, ou seja, ao longo do dia, sempre que a luminosidade interior ultrapassar o valor máximo definido, as lâmpadas vão reduzir a sua luminosidade. Se LL atingir o valor 0 e LI continuar maior que o valor máximo LIMAX, os estores vão fechar até que LI fique dentro do intervalo definido. Este funcionamento é semelhante ao funcionamento do sistema no cenário LEVANTAR. Apenas poderão diferir os valores de luminosidade desejados pelo utilizador.

Quando o nível de luz exterior diminuir até um valor mínimo definido pelo utilizador (LEMIN), todos os estores são fechados completamente e apenas são activadas as lâmpadas nas divisões onde estão pessoas presentes para iluminar a divisão.

Tal como foi explicado no cenário "Levantar", para estes dois dispositivos funcionarem em conjunto, é necessário que a lâmpada conheça a posição do estore L_POE e o estore conheça a luminosidade E_LL e o estado da lâmpada E_LAMP. É também necessário que ambos conheçam se o outro dispositivo está em modo manual ou automático.

O modo de funcionamento manual explicado no cenário Levantar também se aplica aqui, aos estores, às lâmpadas e aos toldos. O funcionamento automático dos toldos é, também, exactamente igual ao explicado no cenário "Levantar". Estes apenas abrem quando está a incidir a luz solar directamente no interior da divisão.

4.1.3. O Cenário Todos Fora

Quando toda a gente sai de casa, para ir trabalhar ou para ir para a escola, é lançado para o sistema o cenário "Todos Fora", presente na figura 4.2. Com este cenário, todos os pontos de iluminação interiores são desligados.

Como não está ninguém presente, as variações de luminosidade interiores não são tão importantes. Por isso, apenas são accionados os estores mas em intervalos muito menos

frequentes, levando a variações de luz interior maiores. No entanto, a vontade do utilizador prevalece. Se o utilizador deseja que os estores funcionem conforme o mencionado, apenas necessita configurar os valores de luminosidade interior máxima e mínima mais intervalados. Se deseja que os estores continuem a funcionar conforme funcionariam na presença de pessoas, deve configurar valores de luminosidade máxima e mínima mais próximos ou até iguais aos valores configurados no cenário "Alguém em Casa".

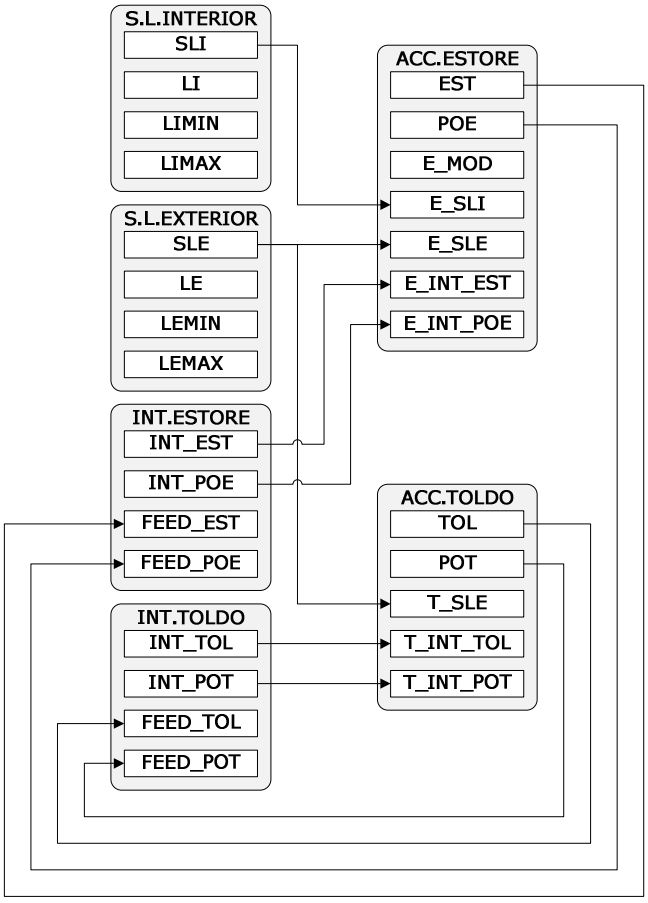

Fig. 4.2. Representação do cenário Todos Fora.

O funcionamento dos estores apresenta apenas uma diferença. Como não há habitantes presentes em casa, os estores vão associar-se à função dos toldos. Sempre que o sensor de luminosidade exterior indica que está a incidir a luz solar directamente no interior da habitação, os estores vão fechar completamente assim como os toldos vão abrir. Assim,

a divisão é protegida dos raios solares e não aquece excessivamente na ausência de pessoas, levando a uma redução do consumo energético.

Como já foi referido, quando for desenvolvido um sistema de controlo da temperatura, sempre que a radiação solar estiver a incidir no interior da habitação, os estores fecham completamente e os toldos abrem, apenas quando a temperatura interior está acima do valor mínimo confortável. Sempre que a temperatura interior estiver abaixo do valor mínimo desejado significa que a divisão necessita ser aquecida. Nesse caso, os estores permanecem abertos e os toldos permanecem fechados permitindo que a radiação solar aqueça a divisão.

4.1.4. O Cenário Dormir

Quando se aproxima a hora de dormir, previamente definida pelo utilizador, é descarregado para o sistema o cenário "Dormir", representado na figura 4.3. Este cenário vai fazer com que o nível de luz das lâmpadas das divisões se reduza para um nível mais baixo, criando um ambiente mais propício ao sono. Para isso, devem ser definidos níveis de luminosidade mais baixos. Na cozinha ou no escritório, por exemplo, podem ser definidos níveis de luminosidade mais elevados que no resto das divisões, pois esses são locais de trabalho e é necessário ter mais luz. No entanto, nos quartos, salas e corredores é possível definir níveis de luminosidade mais baixos de modo a estimular o sono [16].

Quando for detectada a presença de pessoas nas camas (recorrendo a interacções com o sistema de segurança), ao fim de um determinado tempo, as luzes desligam-se progressivamente. Para isso, as lâmpadas dos quartos devem estar configuradas no modo DORMIR através do parâmetro L_MOD.

Neste cenário, tanto os estores como os toldos permanecem tacitamente na posição totalmente fechados. Os toldos permanecem fechados pois durante a noite não têm utilidade. Contudo, os dispositivos accionadores de estores e toldos e as lâmpadas continuam a ter a possibilidade de serem controlados manualmente. Assim, por

exemplo, se o utilizador deseja ficar a ler na cama, basta que coloque as lâmpadas ligadas com a luminosidade pretendida no teclado de controlo manual correspondente.

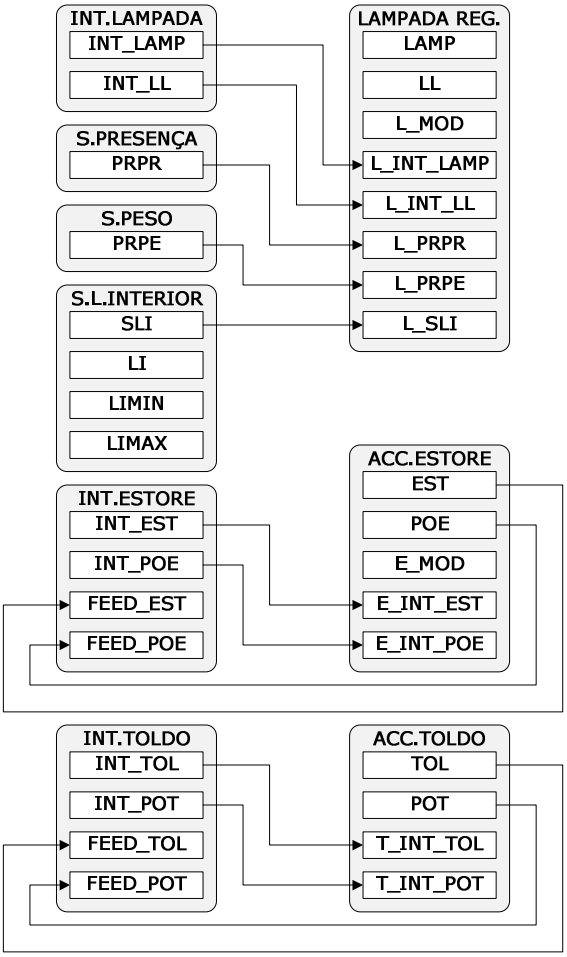

Fig. 4.3. Representação do cenário Dormir.

5. DICIONÁRIOS DE OBJECTOS E OS EDS

5.1. OS DICIONÁRIOS DE OBJECTOS

As entradas de um OD (Dicionário de Objectos), isto é os objectos, são referenciadas através de um simples endereço que consiste num índice (*Index*) de 16 bits e um sub-índice (*Sub-index*) de 8 bits.

O *Sub-index* é usado de diferentes formas de acordo com o tipo de objecto. Para um objecto simples o *Sub-index* é sempre 00H. Para uma ARRAY ou RECORD o *Sub-index* referencia os sub-elementos, sendo o valor 00H usado para guardar o número de elementos que compõe o objecto. Para objectos STRING o *Sub-index* 00H indica o comprimento da STRING. Cada ARRAY ou RECORD apenas pode possuir 254 elementos (o Sub-index FFh é reservado) [7].

Os objectos passíveis de interface com a rede (parâmetros e *data points*) são guardados no OD de cada dispositivo do sistema. Além disso, estão reservadas determinadas zonas com endereços para grupos de objectos com funções específicas, como apresentado na tabela 5.1.

O endereço abaixo de 1000H é utilizado para indicar definições de tipos, que são suportados pelo OD, estes objectos são apenas para referência e que estão descritas na

documentação do dispositivo. A tabela 5.2 mostra os tipos de dados pré-definidos no CANopen e o correspondente endereço (*Index*).

Tabela 5.1. Organização do Dicionário de Objectos [7].

Índice	**Objecto**
0000	Não usado
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types
00A0-0FFF	Reservado
1000-1FFF	Communication Profile Area
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardised Device Profile Area
A000-FFFF	Reservado

Tabela 5.2. Representação dos tipos de dados no Dicionário de Objectos [7].

Índice	**Tipo de dados**	**Índice**	**Tipo de dados**
0001	BOOLEAN	0011	REAL64
0002	INTEGER8	0012	INTEGER40
0003	INTEGER16	0013	INTEGER48
0004	INTEGER32	0014	INTEGER56
0005	UNSIGNED8	0015	INTEGER64
0006	UNSIGNED16	0016	UNSIGNED24
0007	UNSIGNED32	0017	Reserved
0008	FLOAT	0018	UNSIGNED40
0009	VISIBLE STRING	0019	UNSIGNED48
000A	OCTET STRING	001A	UNSIGNED56
000B	DATE	001B	UNSIGNED64
000C	TIME OF DAY	001C-001F	Reserved
000D	TIME REFERENCE	0020	PDO COMMUNICATION PARAMETER
000E	BIT STRING	0021	PDO MAPPING PARAMETER
000F	DOMAIN	0022	SDO COMMUNICATION PARAMETER
0010	INTEGER24	0023	IDENTITY

O intervalo de endereços de 1000H a 1FFFH é reservado para objectos que representam os parâmetros do perfil de comunicações CANopen. Todos os dispositivos devem possuir os objectos obrigatórios para este perfil. O intervalo entre 2000H a 5FFFH é reservado para objectos que representam as características específicas do fabricante do dispositivo. Os objectos não especificados nos perfis de dispositivos *standard*, normalmente são implementados neste intervalo. O intervalo de endereços entre 6000H a 9FFFH é reservado para objectos que caracterizam o funcionamento dos dispositivos normalizados pelo CiA. Não querendo sobrepor os perfis de dispositivos normalizados pelo CiA, os perfis dos dispositivos para o sistema de gestão da iluminação serão mapeados para a zona reservada ao fabricante.

A tabela 5.3 mostra uma representação ordinária de um OD para um dispositivo CANopen.

Tabela 5.3. Exemplo de um dicionário de objectos [7].

Índice	**Objecto**	**Nome**	**Tipo**	**At**	**M/O**
1000	VAR	Device type	UNSIGNED32	ro	M
1001	VAR	Error register	UNSIGNED8	ro	M
1002	VAR	Manufacturer status register	UNSIGNED32	ro	O
1003	ARRAY	Pre-defined error field	UNSIGNED8	ro	O
1008	VAR	Manufacturer device name	VISIBLE STRING	ro	O
1009	VAR	Manufacturer hardware version	VISIBLE STRING	ro	O
100A	VAR	Manufacturer software version	VISIBLE STRING	ro	O
1200	RECORD	Server SDO Comm. Parameter	SDO COM. PAR.	ro	M
1400	RECORD	1st RxPDO Comm. Parameter	PDO COM. PAR.	rw	O
1600	ARRAY	1st RxPDO Mapping Parameter	PDO MAP. PAR.	rw	O
6000	ARRAY	Read 8 input lines	UNSIGNED8	ro	O
6200	ARRAY	Write 8 output lines	UNSIGNED8	rw	O

As seis colunas da tabela 5.3 indicam as seguintes características dos objectos:

1. O Índice do objecto é representado na base hexadecimal e é utilizado como um endereço lógico, endereço de rede, possuindo uma extensão de 16 bits.

2. O Tipo de Objecto é usado com o propósito de referência e para normalização através de diferentes plataformas computacionais. Esses códigos estão listados na tabela 5.4.
3. O Nome do objecto contém uma simples descrição textual da função do objecto.
4. O Tipo de entrada, que pode ser simples ou complexa, pode também ser pré--definida ou específica do fabricante.
5. O Atributo de Acesso ao objecto, que define os direitos de acesso, pode assumir o atributo: **ro** (*read only*), **wo** (*write only*), **rw** (*read/write*) ou **const**. (apenas leitura).
6. O atributo M/O de um objecto indica se a implementação desse objecto é obrigatória ou opcional e pode assumir o atributo **M** (objecto obrigatório) ou o atributo **O** (objecto opcional para esse perfil).

Tabela 5.4. Códigos dos objectos [7].

Nome do obj.	**Descrição**	**Código**
NULL	Entrada do dicionário vazia (sem campos de dados)	0
DOMAIN	Grande quantidade de variáveis (ex. código de programa executável)	2
DEFTYPE	Definição do tipo *Simple* (ex. UNSIGNED8, FLOAT…)	5
DEFSTRUCT	Definição do tipo *Record*	6
VAR	Variável simples (ex. UNSIGNED8, INTEGER16…)	7
ARRAY	Objecto múltiplo que consiste num conjunto de VARs do mesmo tipo	8
RECORD	Objecto múltiplo que consiste num conjunto de VARs de vários tipos	9

Os dicionários de objectos dos dispositivos foram implementados segundo as normas e regras de orientação apresentadas pelo CiA [7], [9], [15], [13], [17], [18] e [19]. Na elaboração de cada um dos dicionários são considerados todos os parâmetros referidos nos modelos de dispositivos mencionados anteriormente assim como as relações de comunicação identificadas nos cenários de funcionamento.

De modo a não tornar este capítulo excessivamente longo, todas as tabelas dos ODs encontram-se no anexo B.

5.1.1. O Dicionário de Objectos do Sensor de Luminosidade Interior

No modelo do sensor de luminosidade interior foram previstos três parâmetros de configuração e um *data point* de saída como resultado da operação do dispositivo. Estes parâmetros encontram-se alojados na área do perfil reservado ao fabricante, tal como foi referido anteriormente. A entrada do OD referente ao estado da luminosidade no interior (SLI) deve ainda incluir a possibilidade de mapeamento em PDO.

Como este dispositivo possui um parâmetro de saída, é necessário definir um TPDO para permitir a comunicação e, por conseguinte, o refrescamento da informação nos outros dispositivos. A configuração do PDO é realizada na área do perfil de comunicações *standard* onde é indicado o número de PDOs necessários e a sua forma de transmissão. O tipo de transmissão deste PDO, assim como de todos os outros, é assíncrona. Na figura 5.1 está representado o mecanismo de transmissão do TPDO no sensor de luminosidade interior. Neste caso, o estado da luminosidade interior é representado na entrada 2001h e sempre que esta sofrer alteração de estado, a transmissão do PDO é disparada.

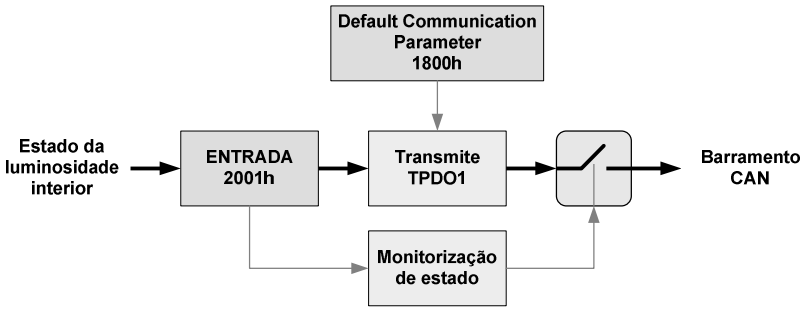

Fig. 5.1. Diagrama de blocos representativo do mecanismo de transmissão do TPDO1 do sensor de luminosidade interior.

5.1.2. O Dicionário de Objectos do Sensor de Luminosidade Exterior

Tal como o sensor de luminosidade interior, o sensor de luminosidade exterior também possui três parâmetros de configuração e um *data point* de saída. De igual modo, estes parâmetros encontram-se alojados na área do perfil reservado ao fabricante. O

parâmetro relativo ao estado da luminosidade no exterior (SLE) possui a possibilidade de mapeamento em PDO.

Tal como no dispositivo anterior, este também recorre a um TPDO para refrescar a informação nos outros elementos da rede. Desta forma, na área do perfil de comunicações *standard* está indicado que é necessário um TPDO assíncrono transmitido por acção de um evento específico do fabricante. Na figura 5.2 está representado o mecanismo de transmissão do TPDO deste dispositivo. Neste caso, o estado da luminosidade exterior é representado na entrada 2003h e sempre que esta sofre alteração de estado, a transmissão do PDO é disparada.

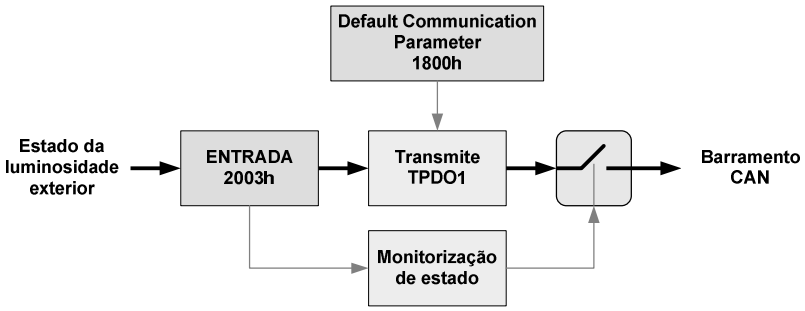

Fig. 5.2. Diagrama de blocos representativo do mecanismo de transmissão do TPDO1 do sensor de luminosidade exterior.

5.1.3. O Dicionário de Objectos do Teclado de Controlo Manual de Lâmpadas

O modelo do teclado de controlo manual de lâmpadas contempla dois parâmetros. Um dos parâmetros corresponde ao estado definido no teclado (automático, ligado ou desligado) e está alojado na entrada 201Ch do dicionário, (ver a figura 5.3). O outro parâmetro corresponde à intensidade luminosa em modo ligado manual e está localizado na entrada 201Dh. Ambos os parâmetros referidos são *data points* de saída.

Ambas as entradas estão mapeadas num PDO de transmissão. Então, na área do perfil de comunicações *standard* é indicado que é necessário um TPDO assíncrono transmitido por acção de um evento específico do fabricante. A transmissão do PDO é disparada sempre que o estado definido no teclado alterar.

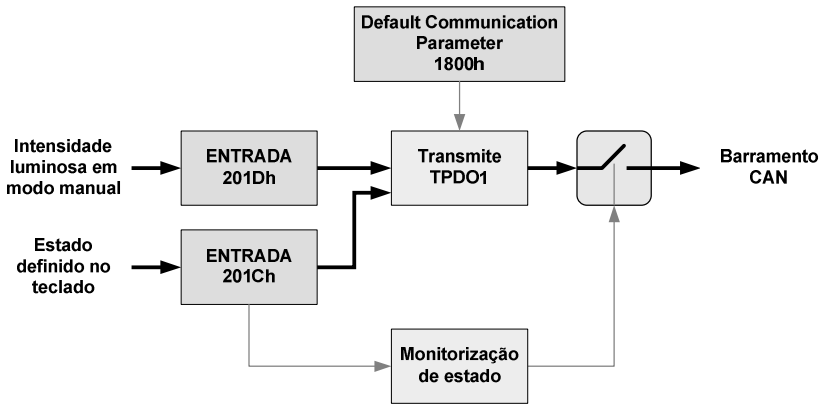

Fig. 5.3. Diagrama de blocos representativo do mecanismo de transmissão do TPDO1 do teclado de controlo manual de lâmpadas.

5.1.4. O Dicionário de Objectos do Teclado de Controlo Manual de Estores

No modelo do teclado de controlo manual de estores foram apresentados quatro parâmetros. Dois dos parâmetros são saídas e correspondem ao estado e à posição do estore definida no teclado. Os outros dois são entradas de *feedback* provenientes do estore e indicam o estado e a posição actual desse. Sendo *data points*, estes parâmetros possuem a possibilidade de mapeamento em PDO, sendo necessário um PDO de transmissão e um de recepção. Assim, na área do perfil de comunicações *standard* é indicada a necessidade de um TPDO e de um RPDO, ambos apresentando tipo de transmissão assíncrona.

Na figura 5.4 está representado o mecanismo de transmissão do TPDO deste teclado. Neste caso, o estado definido no teclado (automático, ligado ou desligado) é representado na entrada 2021h e sempre que esta sofre alteração de estado, dispara a transmissão do PDO.

5.1.5. O Dicionário de Objectos do Teclado de Controlo Manual de Toldos

No modelo do teclado de controlo manual de toldos foram definidos quatro parâmetros, semelhantes aos parâmetros do dispositivo mencionado anteriormente. Dois dos

parâmetros são as saídas correspondentes ao estado de funcionamento e à posição do toldo seleccionada e os outros dois são entradas de *feedback* provenientes do toldo. Tal como no dispositivo anterior, estas quatro entradas do dicionário suportam mapeamento em PDO. Com isto, este dispositivo necessita de um PDO de transmissão e de um PDO de recepção, ambos com tipo de transmissão assíncrona.

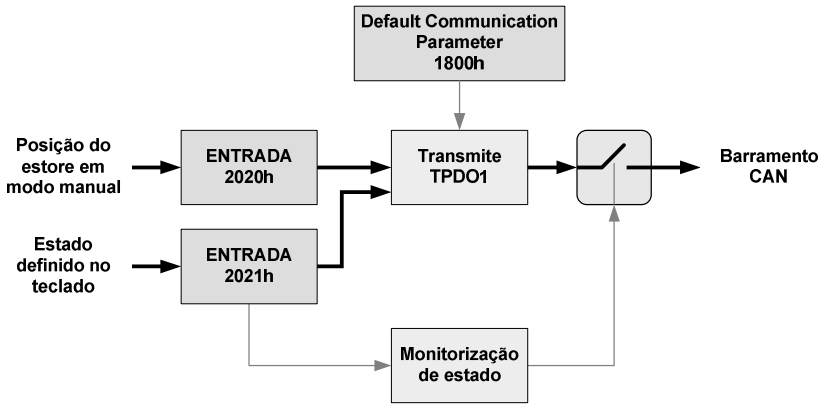

Fig. 5.4. Diagrama de blocos representativo do mecanismo de transmissão do TPDO1 do teclado de controlo manual de estores.

Na figura 5.5 está representado o mecanismo de transmissão do TPDO para este dispositivo. Neste caso, o estado definido no teclado é representado na entrada 2025h e sempre que apresenta alteração de estado, a transmissão do PDO é disparada.

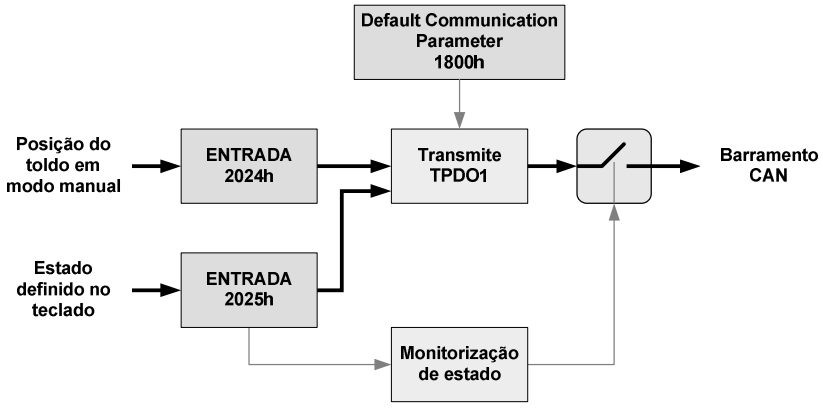

Fig. 5.5. Diagrama de blocos representativo do mecanismo de transmissão do TPDO1 do teclado de controlo manual de toldos.

5.1.6. O Dicionário de Objectos da Lâmpada de Intensidade Regulável

No modelo de dispositivo de actuação da lâmpada de intensidade regulável estão definidos todos os seus parâmetros. Três dos parâmetros são as suas saídas, que correspondem ao estado da lâmpada (ligado ou desligado), à sua intensidade luminosa e ao seu modo de funcionamento (manual ou automático). Estas três saídas são mapeadas num único PDO de transmissão assíncrono. A lâmpada de intensidade regulável recebe também quatro PDOs provenientes de outros dispositivos e que transportam informação necessária para o seu funcionamento.

Na figura 5.6 está representado o mecanismo de transmissão do TPDO deste dispositivo. Neste caso, o estado da lâmpada é representado na entrada 2005h, a sua intensidade luminosa corresponde à entrada 2006h e o modo de funcionamento definido pelo seu teclado corresponde à entrada 2007h. Sempre que o estado da lâmpada ou o estado no seu teclado alterar, a transmissão do PDO é disparada.

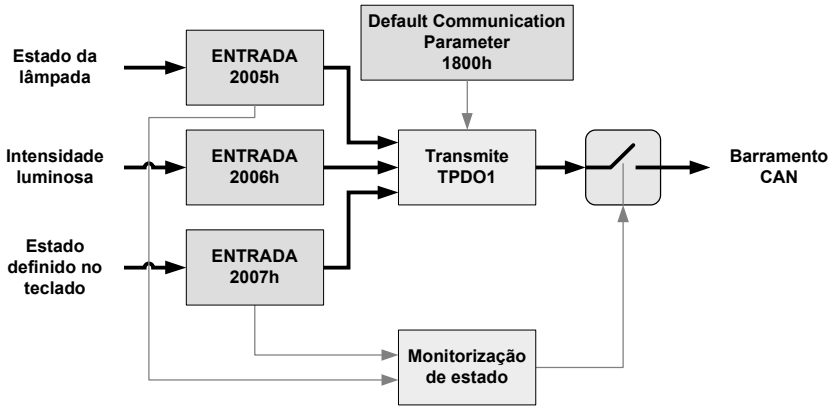

Fig. 5.6. Diagrama de blocos representativo do mecanismo de transmissão do TPDO1 da lâmpada de intensidade regulável.

5.1.7. O Dicionário de Objectos do Accionador de Estores

O modelo de dispositivo accionador de estores apresenta três saídas, que correspondem ao seu estado, à sua posição e ao seu modo de funcionamento (manual ou automático) definido pelo seu teclado. Estes três parâmetros são mapeados num único PDO de

transmissão. O accionador de estores recebe quatro PDOs de alguns dos dispositivos mencionados anteriormente. Como tal, este dispositivo necessita de quatro PDOs de recepção e um PDO de transmissão.

Na figura 5.7 está representado o mecanismo de transmissão do TPDO deste dispositivo. Neste caso, o estado do estore é representado na entrada 200Eh, a sua posição está representada na entrada 200Fh e estado definido no seu teclado corresponde à entrada 2016h. Sempre que o estado do estore ou o estado no seu teclado alterar, a transmissão do PDO é disparada.

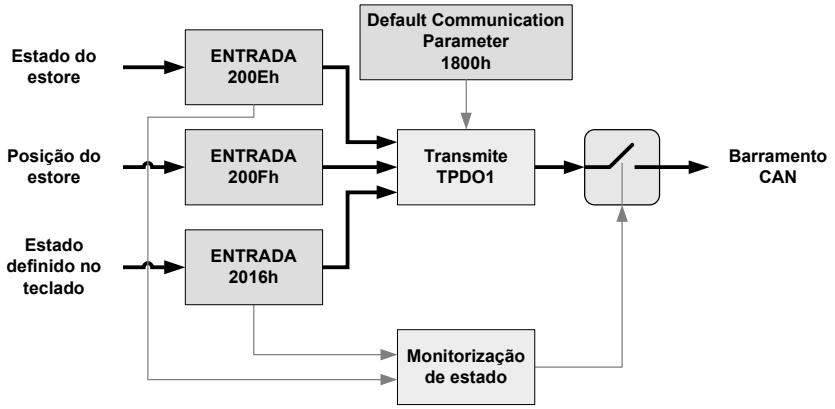

Fig. 5.7. Diagrama de blocos representativo do mecanismo de transmissão do TPDO1 do accionador de estores.

5.1.8. O Dicionário de Objectos do Accionador de Toldos

O dicionário de objectos do accionador de toldos possui cinco *data points*. Dois destes são as suas saídas, correspondentes ao seu estado e à sua posição e que são mapeadas num PDO de transmissão. Os outros três são entradas provenientes do sensor de luminosidade exterior e do teclado de controlo manual de estores. Assim, este dispositivo necessita de dois PDOs de recepção e um PDO de transmissão.

Na figura 5.8 está representado o mecanismo de transmissão do TPDO deste dispositivo. Neste caso, o estado do toldo é representado na entrada 2017h e a sua posição na entrada 2018h. O PDO é disparado sempre que o estado do toldo alterar.

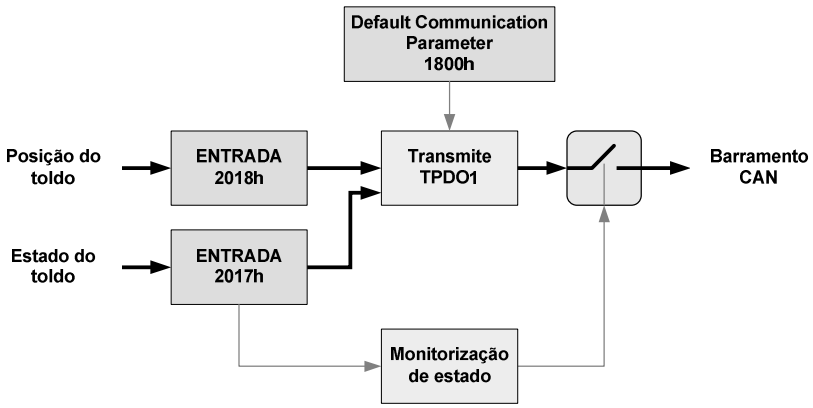

Fig. 5.8. Diagrama de blocos representativo do mecanismo de transmissão do TPDO1 do accionador de toldos.

5.2. OS ELECTRONIC DATA SHEETS (EDS)

De modo a proporcionar ao instalador de sistemas CANopen, a utilização de ferramentas de integração e configuração de redes, todos os dispositivos CANopen fazem-se acompanhar de uma descrição do dispositivo, seguindo uma formatação normalizada, para proporcionar a sua interpretação pelas referidas ferramentas [20]. As ferramentas proporcionam: (i) a configuração dos dispositivos CANopen; (ii) a conformação e gestão da rede; e (iii) a gestão da informação do projecto em diferentes plataformas.

O EDS (*Electronic Data Sheet*) é fornecido pelo vendedor do dispositivo. Se este não for fornecido poder-se-á utilizar um EDS normalizado (o dispositivo deve estar implementado em acordo com as especificações do CiA) [21].

O CiA possui um vasto conjunto de dispositivos normalizados na forma de perfil. Contudo, ainda há um grande caminho a percorrer para que todos os dispositivos passíveis de pertencerem a sistemas CANopen sejam normalizados pelo CiA [13]. No caso do presente trabalho, os EDSs foram realizados especificamente para cada um dos dispositivos aqui discutidos, pois estes ainda não possuem um perfil normalizado pelo CiA.

Devido ao grande volume dos EDSs, não é prático apresentá-los todos neste trabalho escrito. Contudo, no anexo C está presente o EDS do dispositivo accionador de estores, a título de exemplo. Os ficheiros de cada um dos EDSs estão presentes no cd-rom que acompanha este trabalho escrito.

6. ENSAIOS

6.1. AS UNIDADES DE ENSAIO

Sendo objectivo principal deste trabalho, a concepção de um modelo de sistema para a gestão automática da iluminação interior em espaços residenciais, fomentando, sobretudo, o conforto e a gestão energética sem, com isso, diminuir a flexibilidade de utilização, foi necessário definir um conjunto de dispositivos básicos, para os quais foram especificadas as suas funções e enquadras na forma de perfil, em acordo com a norma CANopen. Contudo, foram desenvolvidas apenas duas unidades de teste para que se torne possível avaliar os princípios base do modelo do sistema.

6.1.1. O Hardware

O projecto de um dispositivo CAN requer a escolha criteriosa do hardware de comunicações. A oferta é alargada, muitos são os fabricantes que disponibilizam controladores CAN, apresentando-se nas formas: (i) *stand alone* ou integrado num microcontrolador; (ii) *full* ou *basic* CAN. Contudo, outras restrições conduziram à selecção do controlador, nomeadamente, os níveis de tensão utilizados pelo hardware para definir os níveis lógicos. No presente trabalho, o hardware apresenta duas interfaces de comunicação, uma das quais, utiliza tecnologia de 3,3V, que em conjunto com as necessidades de gestão de energia das unidades *wireless* conduziu à utilização de microcontroladores da família MSP430 da *Texas Instruments*, também com

tecnologia de baixo consumo de 3,3V. Em resultado destas opções e restrições seleccionou-se um controlador *stand alone* de funcionalidade *Full* CAN e com tecnologia de 3,3V.

Os controladores *Full* CAN reservam secções de memória, para a recepção ou transmissão de mensagens, com identificadores programáveis. Quando é recebida uma mensagem e o identificador coincide com o programado, a informação é guardada nessa posição de memória. Se o identificador não coincide com nenhum dos identificadores programados, a mensagem é rejeitada pelo hardware. Os controladores *Basic* CAN recebem todas as mensagens independentemente dos seus identificadores e colocam-nas num buffer de recepção de mensagens. Sempre que há a recepção de uma mensagem, é invocada uma rotina do software, independentemente de a mensagem ser destinada à aplicação ou não.

A ligação do controlador CAN ao barramento deve de ser efectuada utilizando um *transceiver* CAN, cuja missão é garantir os níveis lógicos no barramento, em regra utiliza-se a norma ISO11898. É também aconselhada a utilização de isolamento galvânico entre o controlador CAN e o barramento, para evitar alguma falha eléctrica que afecte o barramento e destrua a totalidade dos dispositivos. Pela mesma razão, é comum utilizar uma fonte de alimentação independente para alimentar a electrónica de comunicações, sendo a energia distribuída pelo próprio barramento em cabos apropriados para o efeito.

Uma ligação com isolamento óptico entre o controlador CAN e o *transceiver* é exibida na figura 6.1. O uso de isolamento óptico é importante para proteger o dispositivo de sobrecargas eléctricas transportadas pelo barramento. Os sinais TxD e RxD são sinais usados pelo controlador para enviar e receber informação. O *transceiver* faz a conversão desses sinais num formato específico para ser transmitido pelo barramento CAN e converte o sinal do barramento CAN num sinal legível pelo controlador.

Relativamente à ligação física do barramento às unidades CANopen, é recomendada a utilização de um conector de 9 pinos D_Sub *standard*, como o representado na figura 6.2. Outras formas de ligação estão, igualmente, normalizadas pelo CiA [18]. Contudo,

e devido à robustez da ligação, foi seleccionado para o *hardware* das unidades de teste o conector D9.

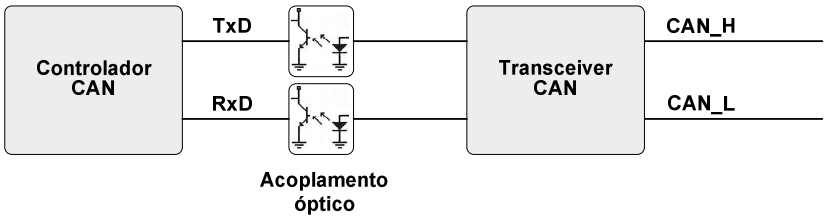

Fig. 6.1. Ligação entre o controlador e o *transceiver* com isolamento óptico [8].

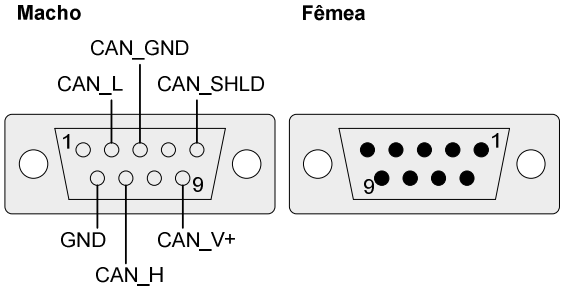

Fig. 6.2. Conector *standard* de 9 pinos D_Sub [18].

A descrição de cada um dos pinos encontra-se na tabela 6.1.

Tabela 6.1. Descrição dos pinos do conector CAN [18].

Pino	Sinal	Descrição
1	Reserved	Pino reservado
2	CAN_L	Linha de barramento LOW
3	CAN_GND	Referência do CAN
4	Reserved	Pino reservado
5	CAN_SHLD	Blindagem opcional CAN
6	GND	Referência opcional CAN
7	CAN_H	Linha de barramento HIGH
8	Reserved	Linha de erro
9	CAN_V+	Fonte de tensão externa opcional

A figura 6.3 mostra uma das placas desenvolvidas para a realização dos ensaios. O microcontrolador utilizado é o MSP430 da série 2 fabricado pela *Texas Instruments* (ver figura 6.4). Para além do seu baixo consumo energético, este microcontrolador pode operar a 16MHz e disponibiliza até 8kB de memória RAM, características essas que são decisivas, face à complexidade e magnitude do software de comunicações, que será necessário integrar nestes dispositivos.

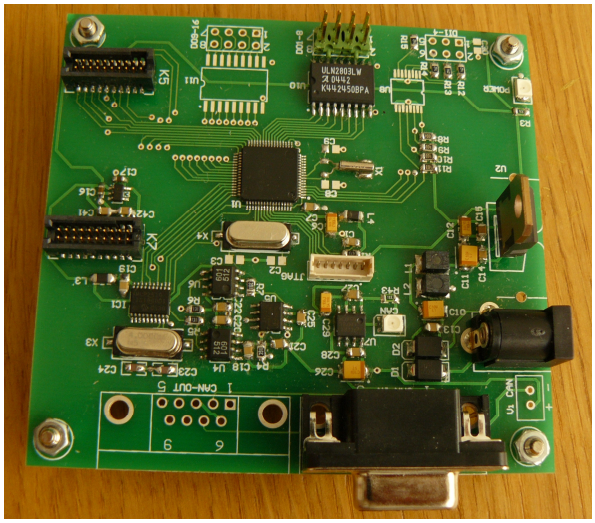

Fig. 6.3. Fotografia de uma placa de ensaio.

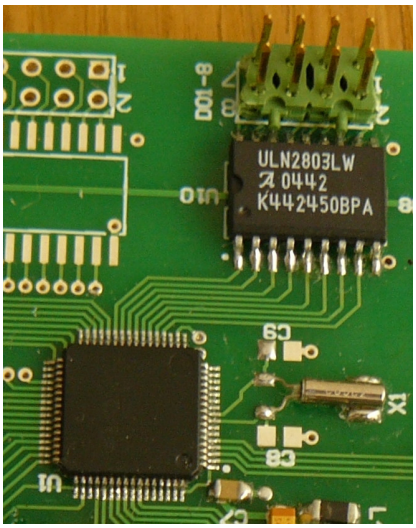

Fig. 6.4. Detalhe do microcontrolador MSP430 e do circuito ULN2803LW.

A interface entre o microcontrolador e o controlador CAN é realizada mediante um barramento SPI (*Serial Peripheral Interface*) e por um conjunto restrito de linhas de IO de controlo. Ficaram disponíveis 16 pinos de IO para que possam ser utilizados como entradas e/ou saídas. Ligado a 8 desses pinos encontra-se o driver ULN2803 que integra 8 transístores, permitindo alimentar cargas até 260W de potência [22].

O controlador CAN utilizado é fabricado pela *Microchip Technology* MCP2510 (ver figura 6.5). Como já foi referido, este controlador apresenta a funcionalidade *Full* CAN, significando que permite a pré-definição de identificadores em secções de memória, para a recepção e envio de mensagens, de modo a que, quando for recebida uma mensagem CAN com um identificador idêntico ao definido previamente, a informação seja guardada nessa posição de memória. Este circuito permite também a transmissão e recepção de mensagens nos formatos *standard* e estendido (suporta as especificações CAN 2.0 Parte A e Parte B). Possui três *buffers* de transmissão e dois *buffers* de recepção. [23].

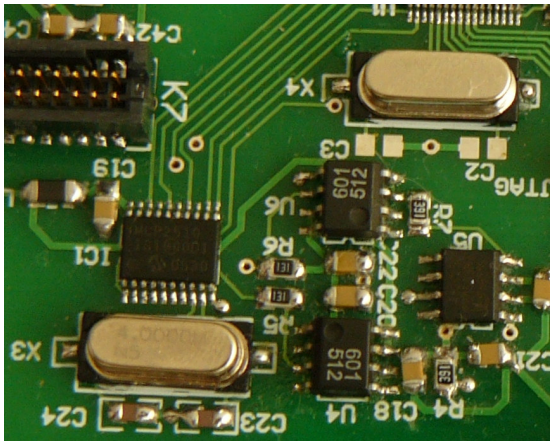

Fig. 6.5. Detalhe do controlador CAN MCP2510, dos acoplamentos ópticos e do *transceiver*.

A alimentação da placa é realizada através de um transformador cuja tensão é regulada para 3.3V. Tanto o microcontrolador MSP430 como o controlador CAN MCP2510 operam a essa tensão.

Posteriormente ao controlador CAN estão presentes dois acoplamentos ópticos de alta velocidade. Estes circuitos têm a função de proteger o controlador CAN de possíveis

sobrecargas que ocorram no barramento. Se todos os dispositivos na rede utilizam isolamento óptico, apenas serão necessários dois cabos para a comunicação: CAN_L e CAN_H. Neste caso, o comum é descartável.

A ligação do controlador CAN ao barramento foi realizada recorrendo a um *transceiver* CAN, como foi referido anteriormente. Neste caso é utilizado o circuito fabricado pela *Philips* com a referência PCA82C250 [24]. Este *transceiver* faz a conversão dos sinais enviados pelo controlador CAN para o barramento ou converte o sinal do barramento num sinal legível pelo controlador. A ligação física ao barramento é realizada através de um conector de 9 pinos D_Sub *standard* (ver figura 6.6).

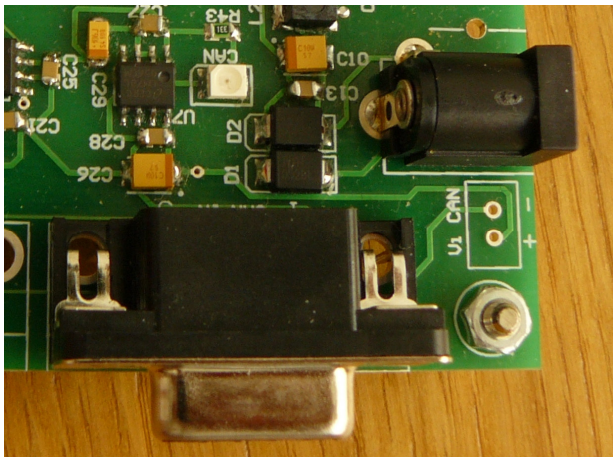

Fig. 6.6. Detalhe do conector de 9 pinos D_Sub.

6.1.2. O Software de Interface com o Controlador CAN

Depois do *power-up*, o controlador CAN necessita ser configurado através da ligação que mantém com microcontrolador. A primeira tarefa de inicialização a ser realizada será a configuração das comunicações entre o MSP430 e o MCP2510, através do barramento SPI. Uma vez o SPI configurado, procede-se à realização do *reset* do controlador de modo a que este entre no modo de configuração [23]. Com o controlador em modo de configuração, é agora possível definir o *bit timing*, as máscaras e filtros de recepção e os buffers de transmissão.

Para configurar as máscaras, os filtros de recepção e os buffers de transmissão é necessário consultar o *User Guide* do controlador e seleccionar os valores a serem introduzidos nos respectivos registos de acordo com as opções de configuração seleccionadas para o barramento. A configuração do controlador é realizada através do envio de comandos específicos através do SPI para os registos de configuração correspondentes. O diagrama de fluxo presente na figura 6.7 esquematiza o processo de escrita nos registos do controlador CAN.

Fig. 6.7. Diagrama de fluxo correspondente à escrita através de SPI.

Como se pode observar na figura 6.7, para escrever num registo, envia-se um comando de escrita, posteriormente envia-se o endereço do registo no qual se quer escrever e, por fim, envia-se a informação a colocar no registo. Através deste processo, é possível definir todos os parâmetros de configuração das máscaras, dos filtros de recepção e dos *buffers* de transmissão do controlador CAN.

Para configurar o *bit timing* é necessário realizar uns cálculos prévios. É através da configuração do *bit timing* que se vai definir o *baudrate* a que o sistema irá comunicar. Este é um passo importante para o funcionamento do sistema pois todos os dispositivos ligados ao barramento devem estar sintonizados no mesmo *baudrate* [23]. A norma CAN, com o objectivo de estabelecer formas de ressincronização expeditas introduz três parâmetros, fincando o *bit timing* dividido por várias secções com missões distintas:

- O *Nominal Bit Rate* (NBR) é basicamente o número de bits transmitidos por segundo que corresponde ao *baudrate* de comunicação desejado;

- O *Nominal Bit Time* (NBT) é definido como o inverso do *Nominal Bit Rate* (ver equação 6.1);
- O *Time Quantum* (TQ) é uma unidade de tempo fixa derivada do período do oscilador. O TQ é definido como mostra a equação 6.2.

$$NBT = \frac{1}{NBR} \tag{6.1}$$

$$TQ = 2\times(BRP+1)\times T_{OSC} \tag{6.2}$$

Na equação 6.2, a variável BRP é um parâmetro de configuração localizado no registo CNF1 do controlador CAN MCP2510. É através da escolha de um valor para este parâmetro que se define um TQ adequado, em função do período do oscilador [23].

O processo de configuração do *baudrate* consiste em definir a duração do TQ e, consequentemente, a duração do NBT com base no TQ. A estrutura do parâmetro NBT (*Nominal Bit Time*) é definida nas especificações CAN como um conjunto de segmentos, conforme mostra a figura 6.8. Cada um desses segmentos é composto por um número inteiro de unidades TQ.

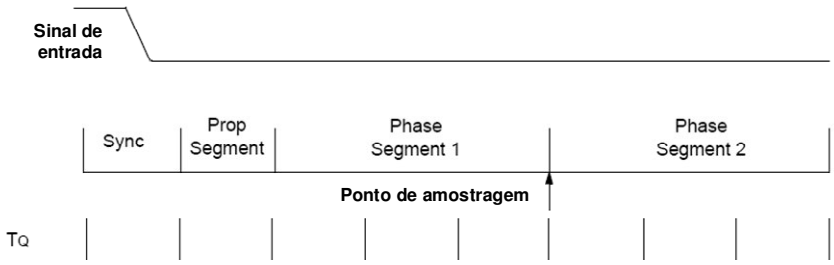

Fig. 6.8. Estrutura do parâmetro *Nominal Bit Time* [23].

O SYNC_SEG é a parte do *bit timing* utilizado para sincronizar os diferentes relógios dos dispositivos ligados ao barramento. O PROP_SEG considera atrasos de propagação no barramento. O PHASE_SEG1 e o PHASE_SEG2 podem ser automaticamente ajustados pelo controlador CAN para compensar erros de sincronização. Entre estes dois está localizado o ponto de amostragem.

A extensão total do NBT pode ser definida com o valor desejado. Contudo, as especificações CAN estabelecem algumas limitações nos comprimentos dos segmentos:

- O SYNC_SEG tem de ser sempre 1 TQ de comprimento;
- O PROP_SEG é programável de 1 a 8 TQ de comprimento;
- O PHASE_SEG1 é programável de 1 a 8 TQ de comprimento;
- O PHASE_SEG2 é programável de 1 a 8 TQ de comprimento, mas é aconselhado que seja definido como igual ao PHASE_SEG1 e nunca menor que 2TQ [23].

Conhecendo as regras estabelecidas, pode-se proceder ao cálculo que irá determinar o *baudrate* do barramento. Tendo em conta as condições mencionadas anteriormente, escolheram-se as seguintes extensões para os segmentos:

- SYNC_SEG=1TQ;
- PROP_SEG=1TQ;
- PHASE_SEG1=3TQ;
- PHASE_SEG2=3TQ.

De modo a obter um *baudrate* normalizado, escolheu-se um valor de BRP=4. Assim, recorrendo à equação 6.2, obtém-se um TQ=2.5µs.

Olhando para a figura 6.8, pode dizer-se que:

$$NBT = \left(SYNC\_SEG + PROP\_SEG + PHASE\_SEG1 + PHASE\_SEG2\right) \quad (6.3)$$

Então, NBT toma o valor NBT=8TQ.

Por fim, recorrendo à equação 6.1, obtém-se um NBR=50kbit/s.

Uma vez obtidos os parâmetros que caracterizam o barramento, é possível configurar os registos de forma a obter o comportamento correcto do controlador, incluindo a configuração das máscaras e dos filtros de recepção. Uma vez a configuração efectuada,

é necessário configurar um último registo para que o controlador CAN entre em modo de funcionamento ordinário. Assim termina o processo de arranque e configuração deste.

Com o controlador a operar como o desejado, fica possível comunicar com o barramento. O procedimento para recepção de mensagens está esquematizado na figura 6.9.

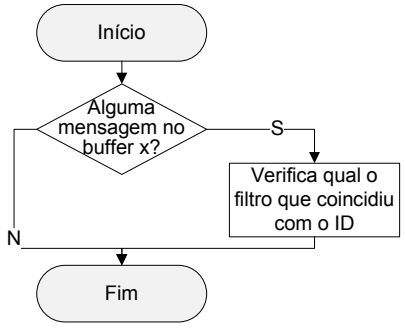

Fig. 6.9. Diagrama de fluxo na recepção de mensagens.

O controlador apresenta um comando especial para verificar a recepção de mensagens, o comando *Read Status*. Com o auxílio desse comando, o controlador CAN vai verificar a recepção de mensagens nos seus *buffers* de recepção e, em caso afirmativo, verifica qual o filtro que coincidiu com o identificador da mensagem.

Para a transmissão de mensagens, o software procede da forma exemplificada no diagrama da figura 6.10.

Primeiro é verificado se existe alguma mensagem pendente no *buffer* de transmissão. Se não existirem mensagens pendentes, esta é colocada num dos *buffers* de transmissão. Posteriormente é definida a prioridade desta mensagem e é executado o comando de requisição de envio.

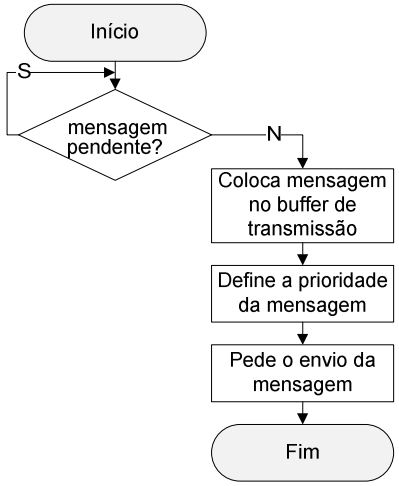

Fig. 6.10. Diagrama de fluxo para o envio de mensagens.

6.2. A BANCADA DE TESTES

Para testar as placas de ensaio realizadas foi necessário recorrer a placas de *interface* CAN e *software* de monitorização assim como a realização de uma aplicação de teste que permita verificar o funcionamento das comunicações entre as duas placas de ensaio.

Com este material, foi realizada a montagem representada na figura 6.11. Ambas as placas de ensaio, assim como a placa de *interface* CAN, foram ligadas ao barramento.

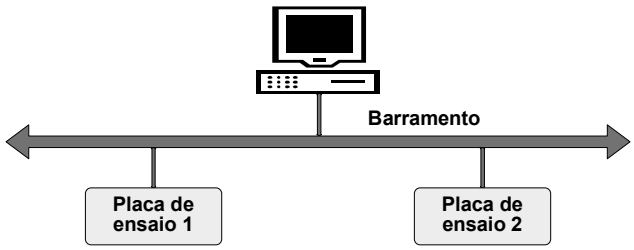

Fig. 6.11. Esquematização da bancada de ensaio.

6.2.1. A Placa e o Programa de Interface com o Barramento

Para testar a comunicação entre as duas placas de ensaio é necessário ter acesso ao barramento CAN, para que seja possível ler as mensagens que são enviadas para este. Para tal, recorreu-se a uma placa de *interface* CAN iPC-I 165/PCI (ver figura 6.12) instalada num computador pessoal. Esta placa é ligada ao barramento CAN, tornando possível, em conjunto com o *software* miniMon, a captura de todas as mensagens que circulam no barramento.

Como já foi referido, para capturar as mensagens no barramento CAN, foi utilizada a ferramenta miniMon desenvolvida pela IXXAT [25]. O miniMon é um simples programa de monitorização do barramento CAN que permite a monitorização *online* do tráfego no barramento CAN bem como a transmissão de mensagens simples.

Fig. 6.12. Fotografia da placa de *interface* CAN iPC-I 165/PCI da IXXAT [26].

A *interface* deste programa apresenta-se como o mostrado na figura 6.13. Na janela do lado esquerdo são apresentadas todas as mensagens que circulam no barramento. Essa janela divide-se em três colunas. A coluna mais à esquerda indica o tempo de recepção da mensagem desde o momento em que a ligação foi estabelecida, a coluna central indica o identificador da mensagem e a coluna da direita indica os dados transportados pela mensagem. Esta ferramenta permite ainda o envio de mensagens para o barramento.

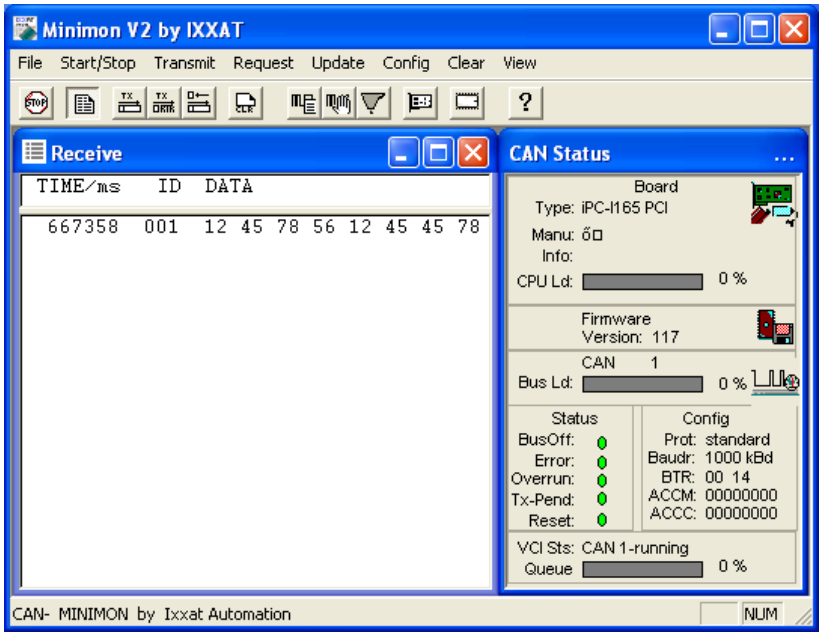

Fig. 6.13. *Interface* da ferramenta de monitorização miniMon.

6.2.2. A Aplicação de Teste

Para testar as comunicações entre as duas placas de ensaio e, assim, atestar o funcionamento do *hardware*, foi desenvolvida uma aplicação que permite simular o comportamento dos dispositivos e a transferência de informação entre eles. Executando esta aplicação nas duas placas de ensaio e recorrendo à ferramenta de monitorização miniMon é possível verificar o comportamento dos dispositivos em função dos dados enviados.

Numa primeira instância, esta aplicação foi executada nas duas placas para que estas se comportem como uma lâmpada de intensidade regulável e como um accionador de estores. A aplicação foi desenvolvida de modo a que fosse possível seleccionar rapidamente o dispositivo pretendido, alterando apenas uma variável e compilando de novo. Com as duas placas a executar a aplicação, é possível verificar a transmissão de mensagens para o barramento e as relações entre os dois dispositivos mencionados. Nas figuras 6.14 e 6.16 é possível ver os diagramas de fluxo da aplicação correspondente à lâmpada de intensidade regulável e ao accionador de estores, respectivamente.

Fig. 6.14. Diagrama de fluxo da aplicação de teste da lâmpada de intensidade regulável.

No diagrama da figura 6.14 é possível ver que a aplicação de teste toma em consideração os parâmetros provenientes do teclado de controlo manual da lâmpada e dos sensores de luminosidade e, ainda, os dados relativos ao estado do estore e ao seu próprio estado, pois as lâmpadas e os estores funcionam em cooperação.

É necessário indicar também o formato dos objectos enviados pela lâmpada de intensidade regulável e pelo accionador de estores de modo a facilitar a leitura dos resultados. As mensagens enviadas por estes dispositivos possuem o formato indicado na figura 6.15. Os seis parâmetros presentes nesta figura estão descritos nos modelos destes dispositivos, no capítulo 3.

	Identificador	Campo de dados 1	Campo de dados 2	Campo de dados 3
Lâmpada de intensidade regulável	06	LAMP	LL	INT_LAMP
Accionador de estores	07	EST	POE	INT_EST

Fig. 6.15. Formato das mensagens da lâmpada de intensidade regulável e do accionador de estores.

No diagrama da figura 6.16 é possível observar que a aplicação de teste do accionador de estores toma em consideração os parâmetros provenientes do teclado de controlo manual de estores, dos sensores de luminosidade e os dados relativos ao seu próprio estado e ao estado das lâmpadas, pois as lâmpadas e os estores cooperam para controlar a luminosidade no interior da habitação.

Como pode ser visto nestes dois diagramas de fluxo, a informação enviada pelos teclados de controlo manual de cada dispositivo tem prioridade relativamente às informações provenientes dos outros dispositivos. Só depois de verificada a informação proveniente do teclado é que a aplicação segue uma de três opções, que condicionam o funcionamento do dispositivo. Consoante a opção seguida, a aplicação faz com que o dispositivo adopte o funcionamento automático, o manual ligado ou o manual desligado. Isto aplica-se tanto à lâmpada de intensidade regulável (ver figura 6.14) como ao accionador de estores (ver figura 6.16).

Interpretando os diagramas das figuras 6.14 e 6.16 é possível identificar o funcionamento já explicado nos capítulos 3 e 4, correspondentes aos modelos dos dispositivos e aos cenários de funcionamento. Como tal, tornar-se-ia repetitivo explicar aqui o funcionamento de toda a aplicação de teste, pois esta é uma interpretação do já explicado nesses capítulos.

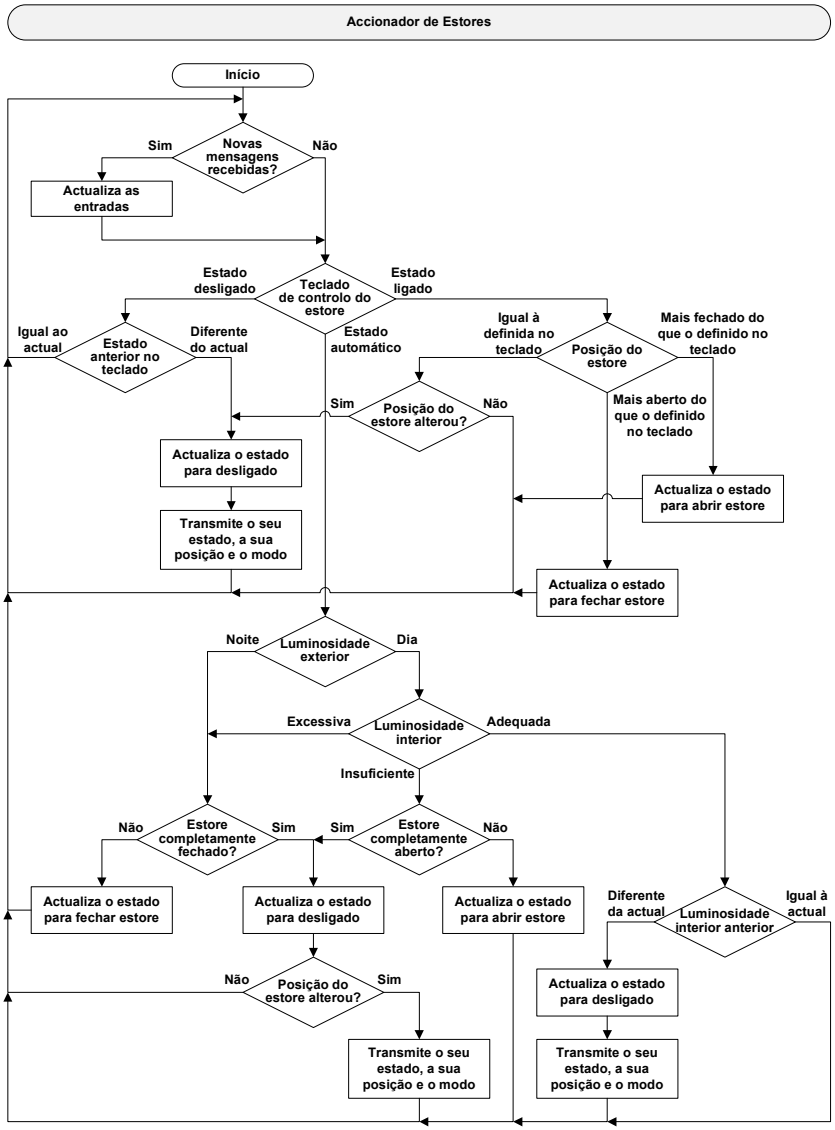

Fig. 6.16. Diagrama de fluxo da aplicação de teste do accionador de estores.

No segundo ensaio, foi apenas testada a aplicação de teste do accionador de toldos. Neste caso, a aplicação foi executada apenas numa placa, pois este dispositivo não tem qualquer relação directa de cooperação com outros accionadores. Na figura 6.17 é possível ver o diagrama de fluxo da aplicação correspondente ao accionador de toldos. É possível identificar, neste diagrama, que a aplicação tem em conta os parâmetros provenientes do teclado de controlo manual do toldo, do sensor de luminosidade exterior e os dados relativos ao seu próprio estado.

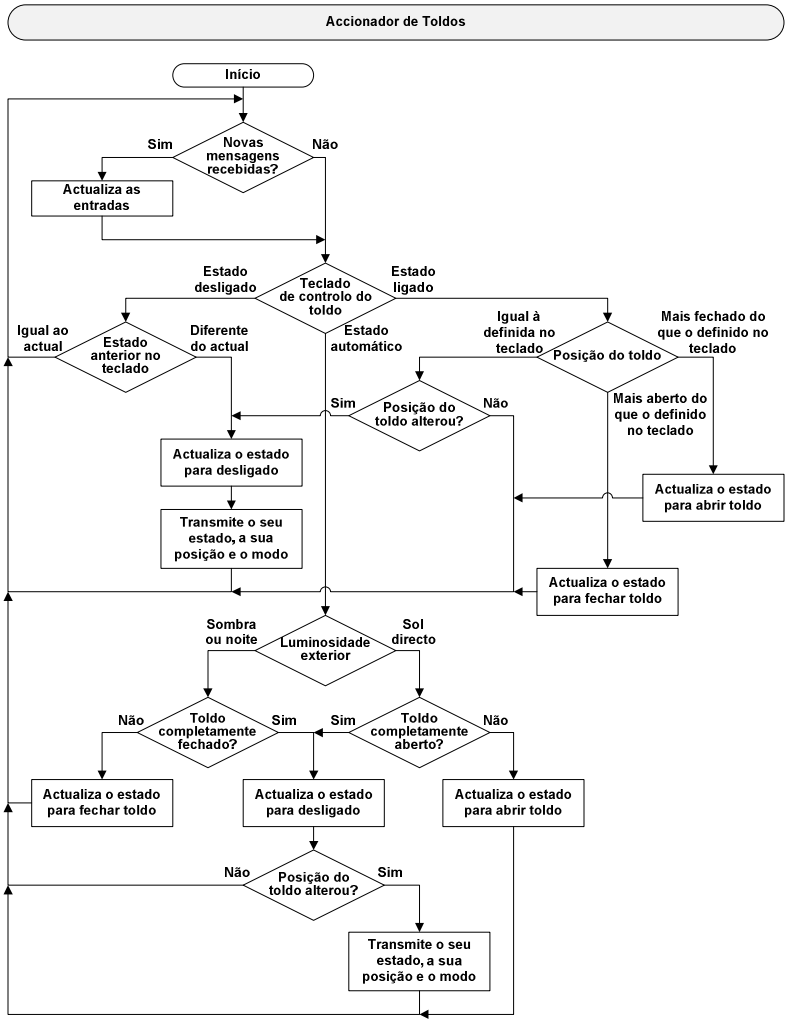

Fig. 6.17. Diagrama de fluxo da aplicação de teste do accionador de toldos.

Tal como no caso anterior, o funcionamento representado neste diagrama de fluxo já foi explicado nos capítulos 3 e 4. Como tal, tornar-se-ia repetitivo explicar novamente aqui o funcionamento da aplicação, pois esta é uma interpretação do já explicado nesses capítulos.

Relativamente aos sensores de luminosidade e aos teclados de controlo manual, devido às suas funções dentro do sistema, a sua acção sobre o barramento foi feita com o apoio da ferramenta miniMon. Através deste programa, foi possível enviar para o barramento mensagens com os identificadores dos sensores de luminosidade e dos teclados de controlo manual com as informações que seriam enviadas por estes, simulando assim a presença destes dispositivos na rede.

O formato das mensagens enviadas pelo accionador de toldos possui a estrutura indicada na figura 6.18. Os três parâmetros presentes nesta figura estão descritos no modelo deste dispositivo.

	Identificador	Campo de dados 1	Campo de dados 2	Campo de dados 3
Accionador de toldos	08	TOL	POT	INT_TOL

Fig. 6.18. Formato das mensagens do accionador de toldos.

O formato das mensagens dos restantes dispositivos é mostrado na figura 6.19. As mensagens com os identificadores representados nesta figura foram enviadas a partir do programa miniMon, de modo a simular a presença destes dispositivos na rede e verificar o comportamento dos accionadores em função destas mensagens.

	Identificador	Campo de dados 1	Campo de dados 2
Sensor de luminosidade interior	01	SLI	00
Sensor de luminosidade exterior	02	SLE	00
Teclado de controlo manual da lâmpada	03	INT_LAMP	INT_LL
Teclado de controlo manual do estore	04	INT_EST	INT_POE
Teclado de controlo manual do toldo	05	INT_TOL	INT_POT

Fig. 6.19. Formato das mensagens dos sensores e dos teclados de controlo manual.

Todos os parâmetros presentes nesta figura estão descritos nos modelos dos dispositivos correspondentes.

6.3. OS RESULTADOS OBTIDOS

Com a montagem representada na figura 6.11 efectuada e as placas a executarem a aplicação de teste da lâmpada de intensidade regulável e do accionador de estores, procedeu-se ao envio de mensagens através do programa miniMon. Depois de enviadas as mensagens é verificado o comportamento das placas de ensaio.

Os resultados obtidos estão presentes na figura 6.20. Antes de mais, é possível constatar que ambas as placas de ensaio estão a funcionar e a comunicar com o barramento. Uma das placas está a executar a aplicação relativa à lâmpada de intensidade regulável e a outra placa está a executar a aplicação relativa ao accionador de estores. Nas duas primeiras linhas presentes na figura 6.20 (Campo A) é possível ver as duas mensagens transmitidas por estes dispositivos quando foram activados. A lâmpada de intensidade regulável indicou que está com luminosidade nula e está em modo automático de funcionamento. O accionador de estores está também em modo de funcionamento automático e permanece numa determinada posição.

As duas mensagens seguintes (Campo B) são mensagens enviadas a partir do programa miniMon a indicar que a luminosidade no interior e no exterior é adequada. Como é possível observar, não houve qualquer resposta por parte dos dispositivos.

De seguida é enviada uma mensagem a indicar que a luminosidade interior é insuficiente (Campo C) e, imediatamente depois é recebida uma mensagem do accionador de estores a indicar que abriu completamente. Como foi dito, neste caso, as lâmpadas só são utilizadas quando a iluminação natural é insuficiente. A mensagem seguinte é da lâmpada e mostra que só depois de o estore transmitir que está aberto é que as lâmpadas iniciam a sua actividade. Então, nesta situação o estore está completamente aberto e as lâmpadas estão na sua luminosidade máxima. Eles atingiram este estado porque não receberam qualquer indicação de que a luminosidade interior voltou a ser adequada, durante o intervalo de tempo no qual o accionador de estores esteve a abrir e a lâmpada a aumentar a sua intensidade luminosa.

De seguida é enviada uma mensagem a indicar que, por fim, a luminosidade interior é adequada (Campo D). Neste caso obteve-se uma resposta por parte de ambos os dispositivos a indicar que mantêm os seus estados.

Quando é enviada uma nova mensagem a indicar que a luminosidade interior é excessiva (Campo E), verifica-se que o primeiro dispositivo a enviar a mensagem foi a lâmpada. A mensagem recebida indica que a lâmpada tem uma luminosidade nula, o que está de acordo com o planeado. Quando existe luminosidade em excesso no interior, as lâmpadas devem desligar antes de os estores fecharem, de modo a aproveitar ao

máximo a luminosidade natural e utilizar apenas a luz artificial quando a natural não for suficiente. Depois da lâmpada transmitir que se desligou, o accionador de estores recebeu essa informação e iniciou o fecho. Quando cessou a sua actividade, transmitiu uma mensagem a indicar a sua posição.

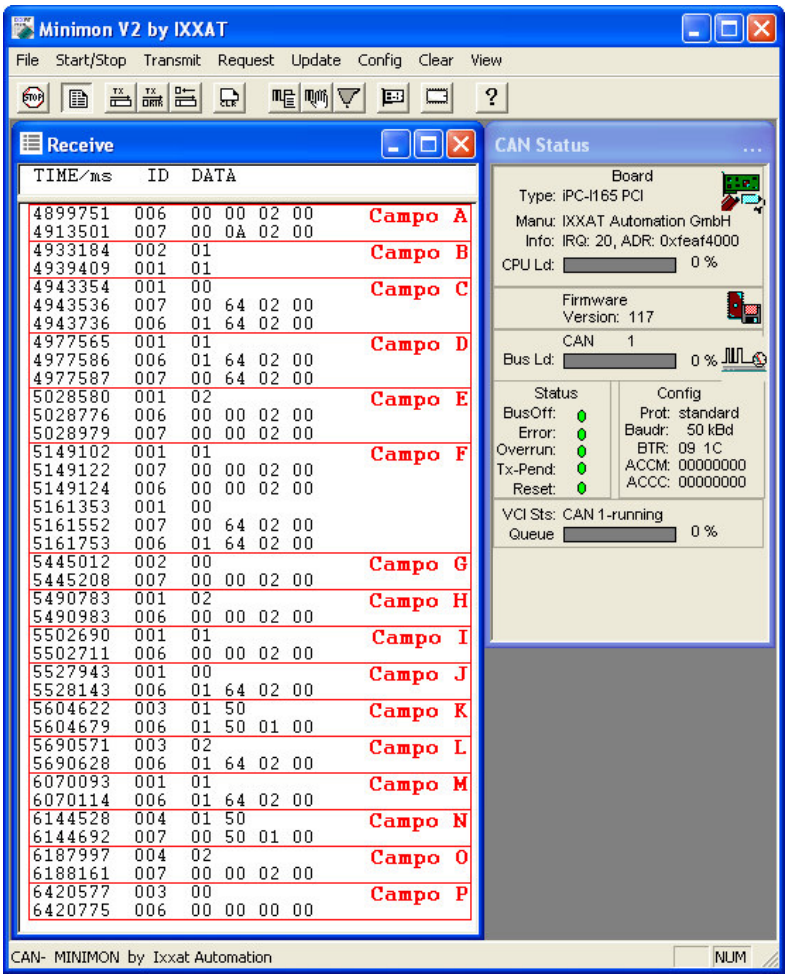

Fig. 6.20. Resultados do ensaio realizado à lâmpada de intensidade regulável e ao accionador de estores.

Na mensagem seguinte (Campo F), transmitida a partir do miniMon, é indicado novamente que a luminosidade interior é adequada e na seguinte é indicado que a luminosidade interior voltou a ser insuficiente, servido, apenas, para simular o anoitecer.

A mensagem seguinte indica que no exterior é noite (Campo G). Pode-se observar que o estore fechou imediatamente e transmitiu uma mensagem a indicar que está

completamente fechado. Enquanto não for indicado que há luz no exterior, o estore vai permanecer fechado.

No campo H é transmitida a indicação de que a luminosidade interior está acima do desejado. Neste caso é possível observar que apenas responderam as lâmpadas a indicar que diminuíram a sua luminosidade. Atingiram o seu mínimo porque entretanto não houve a indicação de que a luminosidade interior voltou a ser adequada.

No campo I é transmitida a informação de que a luminosidade interior é adequada e a lâmpada manteve o seu estado. No campo J indicou-se que a luminosidade interior é insuficiente e obteve-se uma mensagem da lâmpada a indicar que a sua luminosidade aumentou.

No campo K é transmitida uma mensagem, com o identificador do teclado de controlo manual da lâmpada, com a indicação de que esta deve permanecer ligada com uma luminosidade 50. Imediatamente depois é recebida uma mensagem da lâmpada a indicar que está ligada e adoptou a luminosidade indicada.

No campo L é indicado novamente que a lâmpada deve entrar em modo automático. É recebida uma mensagem a indicar que a lâmpada está em modo automático e apresenta a luminosidade máxima. Isto aconteceu porque a última informação enviada pelo sensor de luminosidade interior indicava que esta era insuficiente.

No campo M é indicada que a luminosidade interior voltou a ser adequada e a lâmpada transmitiu a indicação que manteve o seu estado.

No campo N é transmitida uma mensagem, com o identificador do teclado de controlo do estore, com a indicação de que este deve colocar-se na posição 50. Imediatamente depois é recebida uma mensagem do estore a indicar que atingiu a posição imposta pelo teclado. Como já tinha sido mencionado, os teclados de controlo manual prevalecem perante as outras informações. Isso é mostrado neste campo onde o sistema tinha definido que o estore devia permanecer fechado e o teclado de controlo manual fez com que o estore abrisse.

No campo O, o accionador de estores é colocado novamente em modo automático. Como no exterior continua noite, o estore fechou imediatamente e transmitiu a sua posição.

No campo P é transmitida uma mensagem com o identificador do teclado de controlo manual da lâmpada, com a indicação de que esta deve permanecer desligada. Imediatamente depois é recebida uma mensagem da lâmpada a indicar que está desligada.

Como o accionador de toldos não tem uma interacção directa com os outros accionadores, a aplicação de teste do toldo foi executada apenas numa das placas. Com a aplicação em execução, procedeu-se ao envio de mensagens através do programa miniMon.

Os resultados obtidos neste segundo teste estão presentes na figura 6.21.

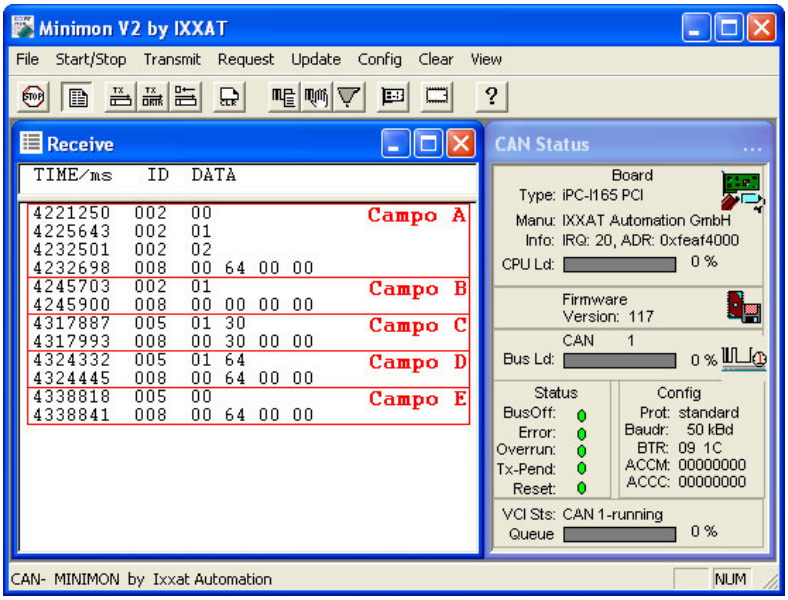

Fig. 6.21. Resultados do ensaio realizado ao accionador de toldos.

Na configuração inicial do accionador de toldos é definido que este deve permanecer fechado. Por isso, neste teste, o accionador de toldos inicia na sua posição fechada.

No campo A da figura 6.21 são enviadas três mensagens com o identificador do sensor de luminosidade exterior. Na primeira mensagem é indicado que é noite e na segunda mensagem que a luminosidade exterior é adequada. Em ambos os casos, o accionador de toldos não realiza qualquer acção. Na terceira mensagem, enviada a partir do miniMon, é indicado que a luz está a incidir directamente no interior da habitação. Neste caso, o toldo inicia a sua abertura e, quando este se encontra completamente aberto, transmite a sua posição e o seu estado de funcionamento.

No campo B é enviada uma mensagem com a indicação de que luminosidade exterior voltou a ser adequada. É possível observar que o toldo fecha, pois deixou de ser necessário, e transmite a informação de que está completamente fechado.

No campo C é transmitida uma mensagem, com o identificador do teclado de controlo do toldo, com a indicação de que o toldo deve colocar-se na posição 30. Imediatamente depois é recebida uma mensagem deste a indicar que atingiu a posição imposta pelo teclado.

No campo D é ordenado ao toldo, por acção do seu teclado de controlo manual, que se coloque na posição de abertura máxima. Imediatamente depois é recebida uma mensagem do toldo a indicar que atingiu a sua posição de abertura máxima, imposta pelo teclado.

No campo E, é enviada uma mensagem a partir do miniMon com a indicação que o toldo deve desligar. Como este está desligado, apenas transmite uma mensagem com o seu estado e a sua posição. Se o toldo estivesse em andamento, este iria parar na posição em que estava e transmitiria uma mensagem com a sua posição actual.

Com estes testes ficou provado que as placas de ensaio estão a funcionar e estão a receber e a transmitir dados, pois não houve qualquer falha de comunicação entre estas durante os ensaios. Ficou também provado, que os modelos de dispositivos desenvolvidos são adequados e permitem que o sistema funcione adequadamente.

7. CONCLUSÕES

7.1. CONCLUSÕES

Neste trabalho foi concebido um modelo de sistema para a gestão da iluminação em espaços residenciais, que aufere de elevada flexibilidade operacional e modularidade. Com base no levantamento das necessidades específicas para este género de sistemas foi definido um conjunto de dispositivos que proporcionam a correcta operação do sistema, atingindo os intentos desejados.

Os modelos de dispositivos concebidos têm funções bem definidas: (i) medida da luminosidade interior e exterior; (ii) accionamento de estores e toldos e (iii) actuação de iluminação artificial. Estes cooperam entre si fazendo com que o sistema funcione de forma completamente autónoma, com reduzida ou completa ausência de intervenção por parte do utilizador. Para o funcionamento de cada dispositivo e do sistema global ser o mais adequado, em cada um dos modelos de dispositivo, foram definidos: (i) parâmetros de configuração para que o utilizador possa adaptar o funcionamento do sistema às suas necessidades, sem que seja necessário recorrer a nova programação do *hardware*; e (ii) *data points,* entradas e saídas, que representam os pontos de ligação entre dispositivos, possibilitando a interacção entre estes na forma de aplicação distribuída.

De forma a demonstrar maior realismo e reforçar a forma de acção prevista para o sistema, foram, também, apresentados quatro cenários de funcionamento, que

representam quatro aplicações distintas, numa perspectiva de instrumentação virtual. Estes representam as rotinas diárias da maioria das famílias activas, onde foram enumeradas as diversas relações entre dispositivos na forma de ligações virtuais entre *data points* e as suas diferentes configurações.

Com o objectivo de colocar em prática o modelo idealizado, foram elaborados os diversos dicionários de objectos, assim como, os diferentes EDS (*electronic data sheets*) necessários à conformação da rede através de ferramenta de configuração. Foram, também, realizadas duas placas de ensaio com capacidade para integrar a norma de comunicações CANopen. Depois do *hardware* desenvolvido, foram realizados testes que provaram o funcionamento das placas, nos quais foi possível verificar também, em termos práticos, a aptidão dos modelos de dispositivos propostos.

O sistema de iluminação aqui proposto visa a sua aplicação num sistema domótico com elevado nível de integração, isto é, num sistema onde os diferentes subsistemas (ex: segurança, climatização, etc.) interagem entre si, possibilitando a troca de informação entre dispositivos pertencentes aos diferentes sistemas sem recurso a *gateways* ou outras formas de ligação. Neste contexto, as principais potencialidades do sistema de iluminação só serão evidentes se este se encontrar ligado ao sistema de segurança e de climatização. De facto, este sistema necessita ter em consideração a presença de pessoas nas divisões da habitação assim como a temperatura no interior e no exterior da habitação para operar de forma optimizada.

Tratando-se de um sistema autónomo implementado sobre uma plataforma de suporte já definida [3], este trabalho veio introduzir as necessárias metodologias de desenvolvimento para aplicações que constituem o universo da Domótica e conferir maior realismo à referida plataforma.

7.2. TRABALHOS FUTUROS

Este trabalho pode constituir a base de novos trabalhos. A partir deste ponto, pode ser desenvolvido todo o *hardware* e *software* ligado às aplicações locais, nomeadamente as *interfaces* de accionamento e monitorização.

Tratando-se de uma nova tecnologia, será necessário integrar o *stack* do protocolo CANopen desenvolvido em trabalhos anteriores [27] a este novo hardware. Tratando de uma plataforma híbrida, a integração do *stack* de comunicações sem fios desenvolvido em [3] também se reveste de elevada importância.

Como já foi mencionado, seria útil desenvolver um sistema de controlo da climatização que interagisse com o sistema proposto. Com a troca de informações entre estes dois sistemas, seria possível actuar os toldos em função das temperaturas. Assim, seria possível reduzir o aquecimento da habitação no Verão e utilizar a radiação solar para a aquecer no Inverno. Desta forma, o consumo energético destinado à climatização seria reduzido.

O desenvolvimento de um sistema de segurança também é importante para a evolução deste sistema. Havendo troca de informações de presenças entre estes dois sistemas, seria possível actuar a iluminação artificial apenas nas divisões onde se verifica a presença de pessoas e, desse modo, reduzir o consumo energético destinado à iluminação.

BIBLIOGRAFIA

[1] R. Nunes; *"Modelo de Especificação e Programação de um Sistema Domótico";* Instituto Superior Técnico / INESC-ID, Lisboa, 2004.

[2] J. García, A. Nieto, A. Barbolla, J. Corredera; *"El Hogar Digital como Solución a las Necesidades de las Personas Mayores";* Ceditec (Centro de Difusión de Tecnologías), Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid.

[3] B. Ribeiro; *"Um Sistema Distribuído para a Automação de Espaços Residenciais e de Serviços";* Tese de Doutoramento para a Obtenção do Grau de Doutor em Engenharia Electrotécnica, Universidade da Beira Interior, Covilhã, 2008.

[4] R. Nunes, C. Sêrro; *"Edifícios Inteligentes: Conceitos e Serviços";* Instituto Superior Técnico / INESC, Lisboa, 1999.

[5] Linkuti Domótica. http://linkuti.com (08/07/09)

[6] R. Nunes; *"Domótica: Presente e Futuro";* Instituto Superior Técnico / INESC, Lisboa.

[7] *"CANopen Application Layer and Communication Profile: CiA Draft Standard 301";* CiA - CAN in Automation, 2002.

[8] M. Farsi, M. Barbosa; *"CANopen Implementation: applications to industrial networks";* RSP-Research Studies Press, Baldock, 2000.

[9] W. Lawrenz; *"CAN System Engineering: From Theory to Pratical Applications";* Springer, New York, 1997.

[10] A. Guimarães; *"O Barramento Controller Area Network: Conceituação";* Departamento de Engenharia de Computação e Sistemas Digitais/USP, São Paulo.

[11] *"CAN Specification 2.0: Part A";* CiA - CAN in Automation.

[12] *"CAN Specification 2.0: Part B";* CiA - CAN in Automation.

[13] CAN in Automation. http://www.can-cia.org/ (03/10/2008)

[14] G. Fernandes, M. Correia, S. Almeida; *"Sistema de Instrumentação Distribuído Multi-sensorial";* Projecto Final de Curso, Universidade do Porto/FEUP, Porto, 2001.

[15] B. Ribeiro; *"Projecto de um Sensor Inteligente para Medida de Temperatura sobre uma Rede CANopen";* Provas de Aptidão Pedagógica e Capacidade Científica: Componente Pedagógica – Aula Prática, Universidade da Beira Interior, Covilhã, 2001.

[16] J. Alves, J. Mota; *"Casas Inteligentes";* Centro Atlântico, Famalicão, 2003.

[17] *"CANopen Device Profile for Generic I/O Modules: CiA Draft Standard 401";* CiA - CAN in Automation, 2008.

[18] *"CANopen Additional specification: CiA Draft Recommendation 303";* CiA - CAN in Automation, 2008.

[19] *"CANopen Device Profile Measuring Devices and Closed-Loop Controllers: CiA Draft Recommendation 404";* CiA - CAN in Automation, 2002.

[20] *"CANopen Configuration Studio for the Configuration of CANopen devices and Systems: Software Version 1.4";* IXXAT Automation GmbH, Weingarten.

[21] *"Electronic data sheet specification for CANopen: CiA Draft Standard 306";* CiA - CAN in Automation, 2005.

[22] *"High-Voltage, High-Current Darlington Arrays";* Allegro MicroSystems, Inc. http://www.allegromicro.com (12/01/2009)

[23] *"Stand-Alone CAN Controller with SPI Interface";* Microchip Technology Inc. http://www.microchip.com (12/01/2009)

[24] *"PCA82C250 / 251 CAN Transceiver";* Philips Semiconductors. http://www.nxp.com (12/01/2009)

[25] *"miniMon: CAN Monitoring Tool for Windows";* IXXAT Automation GmbH, Weingarten.

[26] IXXAT Automation GmbH. http://www.ixxat.com (03/10/2008)

[27] B. Ribeiro; *"Uma Plataforma Universal Para a Automação e a Instrumentação Industrial";* Provas de Aptidão Pedagógica e Capacidade Científica: Trabalho de Síntese, Universidade da Beira Interior, Covilhã, 2001.

[28] N. Çenesiz, M. Esin; *"Controller Area Network (CAN) for Computer Integrated Manufacturing Systems";* Journal of Intelligent Manufacturing, 2004.

[29] R. Díaz; *"CANopen";* Universidad de Salamanca, Salamanca, 2007.

ANEXOS

ANEXO A – AS TECNOLOGIAS

A.1. CONCEITO DE FIELDBUS

O *fieldbus* (ou rede de campo) surgiu na década de 80 e 90 para substituir as ligações físicas tradicionais ponto a ponto entre os controladores e os dispositivos de campo (aquisição de dados e actuação). Esta tecnologia permite a transmissão bidireccional de dados em formato digital entre os diversos dispositivos presentes na rede.

As redes *fieldbus* têm por base a estrutura por camadas do modelo de referência ISO/OSI mas, devido à simplicidade dos dispositivos que integram as redes, não é necessário utilizar a totalidade das camadas do modelo de referência. A estrutura da maioria dos *fieldbus* só apresenta três das camadas do modelo de referência ISO/OSI (ver figura A.1.).

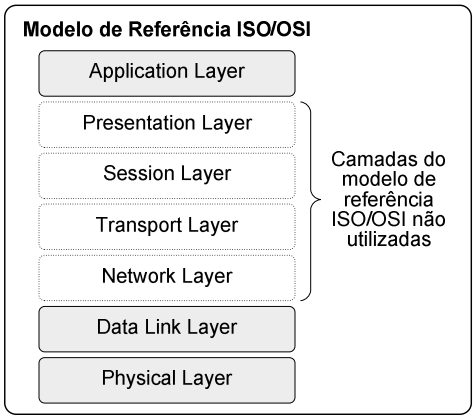

Fig. A.1. Estrutura de um *fieldbus* [3].

Esta estrutura é designada por EPA (*Enhanced Performance Architecture*) [3] e tem por base as camadas *Physical Layer*, *Data Link Layer* e *Application Layer* do modelo de referência ISO/OSI.

A tecnologia *fieldbus* trouxe muitos benefícios para além do simples meio de comunicação entre os dispositivos. Esses benefícios estendem-se desde a instalação à manutenção do sistema [27].
A interligação entre os diversos dispositivos de aquisição de dados e de actuação do sistema sofreu uma grande redução de custos. Tradicionalmente, a ligação dos

elementos de aquisição de dados e de actuação aos controladores era feita através de cabos dedicados de 4-20mA. Isto exigia uma grande quantidade de cabos para os dispositivos comunicarem com o controlador para além do grande comprimento que estes deviam ter. Isto tornava toda a cablagem extremamente complexa e exigia um grande esforço de verificação de toda a cablagem em situações de avaria [27].

A tecnologia *fieldbus* veio terminar com este problema pois todos os dispositivos passaram a partilhar um barramento de comunicação único que, na maioria dos casos é composto por apenas um par de cabos eléctricos.

A tecnologia *fieldbus* permitiu a utilização de sensores e actuadores inteligentes. Estes dispositivos têm capacidade de tratamento do sinal antes de ser transmitido. A transmissão de informação passou a ser em formato digital, o que tornou a informação mais credível. Nos meios de transmissão tradicionais, a informação era transmitida de forma analógica e era susceptível a erros que podiam tornar os dados corruptos. Actualmente, os protocolos de comunicação *fieldbus* têm mecanismos de detecção e correcção de erros que aumenta a confiança nos dados transmitidos [27].

Nos sistemas tradicionais, em caso de avaria, era necessário fazer longas inspecções a toda a cablagem e, muitas vezes, era necessário substituir ligações analógicas, o que eram tarefas que demoravam muito tempo. Os dispositivos dentro da rede *fieldbus* têm possibilidade de realizar auto-diagnósticos e emitir mensagens de erro sempre que surge alguma falha o que permite determinar rapidamente a origem da avaria, tornando o sistema muito mais robusto.

A.2. SISTEMAS CENTRALIZADOS VS DESCENTRALIZADOS

A figura A.2. ilustra um sistema conhecido como um sistema de controlo centralizado. Neste tipo de sistema todos os dispositivos de controlo estão ligados a um controlador central. O controlador central é o responsável por coordenar o funcionamento da célula ou máquina.

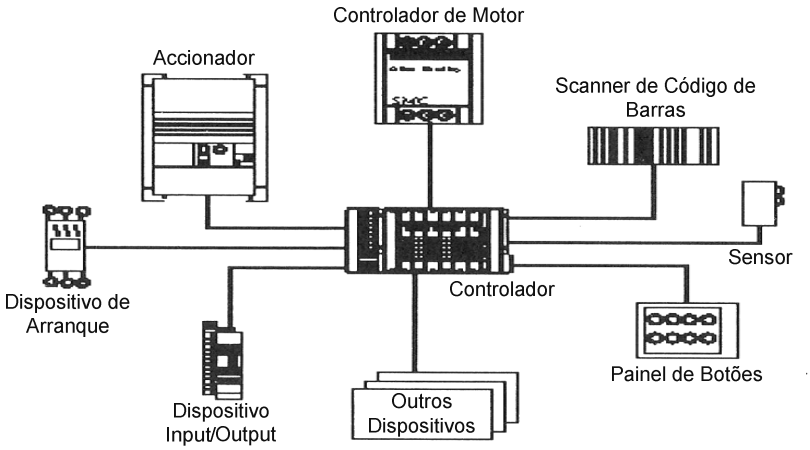

Fig. A.2. Sistema de controlo centralizado [8].

Diferentes componentes de controlo são montados em diferentes áreas geográficas, por exemplo, na envolvente de uma célula de produção. Cada tipo de dispositivo tem uma *interface* eléctrica independente com o controlador. Com isto, terá de existir uma grande quantidade de cabos eléctricos, em torno da célula, que correspondem à ligação entre cada dispositivo e o controlador central. Isto é desvantajoso porque conforme as células de produção vão ficando cada vez mais complexas, também a complexidade da cablagem vai aumentando. Com uma cablagem mais complexa, as falhas são muito mais difíceis de encontrar e, por outro lado, a maior quantidade de cabos vai aumentar o custo do conjunto do sistema. Uma grande quantidade de cabos pode também ser difícil de ligar a uma unidade de controlo simples por causa do tamanho físico dos conectores.

Este sistema tem outra desvantagem quando é necessário adicionar um novo dispositivo. Nesta situação, é necessário passar novos cabos até ao controlador central.

Para ultrapassar as desvantagens mencionadas, isto é para reduzir o custo da implementação e para aumentar a facilidade de integração, a utilização de *fieldbus* é uma solução que está, cada vez mais, a ser adoptada. Um sistema típico baseado em *fieldbus* está representado na figura A.3. Nesta figura pode ser visto que, em vez de cada dispositivo ligar a um controlador, todos os dispositivos estão localmente ligados ao mesmo *fieldbus.* Um *fieldbus* é uma solução inovadora que assegura uma comunicação eficiente entre os dispositivos constituintes da célula de produção. Este é um meio de comunicação partilhado.

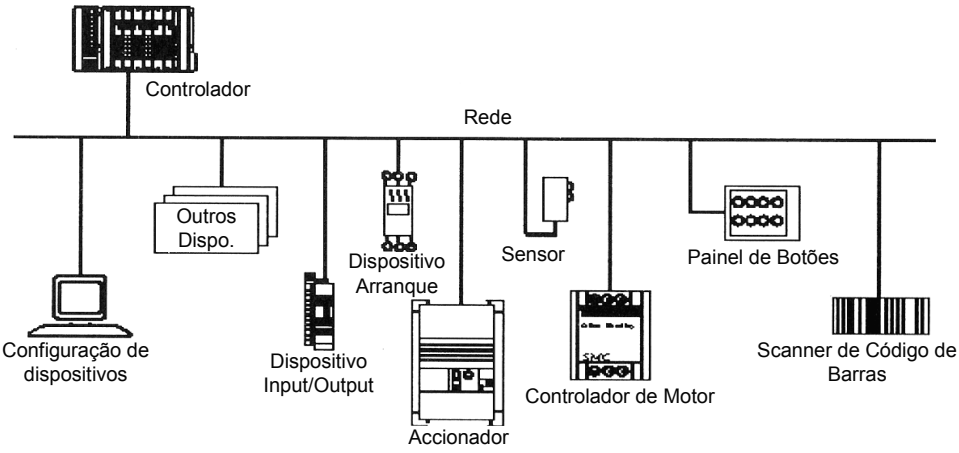

Fig. A.3. Sistema de controlo descentralizado [8].

A tecnologia da informação e o desenvolvimento de microcontroladores de baixo custo com capacidade de comunicação integrada permitiram dotar os componentes de controlo de funcionamento simples, como dispositivos ON/OFF, de inteligência e capacidade de comunicar com outros dispositivos. Isto abriu o caminho para um novo conceito: o Sistema de Controlo Distribuído também chamado de Sistema de Controlo Descentralizado.

Com o Sistema de Controlo Descentralizado são necessários menos cabos assim como menos conectores. Isto permite reduzir o custo e ainda permite adicionar novos dispositivos de forma mais fácil. Este tipo de sistema também está associado a uma maior flexibilidade e apenas necessita de uma unidade de *interface*.

A.3. O MODELO DE REFERÊNCIA ISO/OSI

O Modelo de Referência ISO/OSI é um conjunto de regras estruturantes apresentada pela *International Standards Organization* (ISO) [8] para apoiar o desenvolvimento e a implementação de protocolos de comunicação abertos. Este modelo é usado como uma referência pelas especificações dos protocolos mais actuais.

O modelo é baseado na suposição de que há dois dispositivos ou aplicações cooperantes que desejam comunicar. Cada aplicação está a servir um processo particular. A hierarquia de cada uma das camadas está representada na figura A.4.

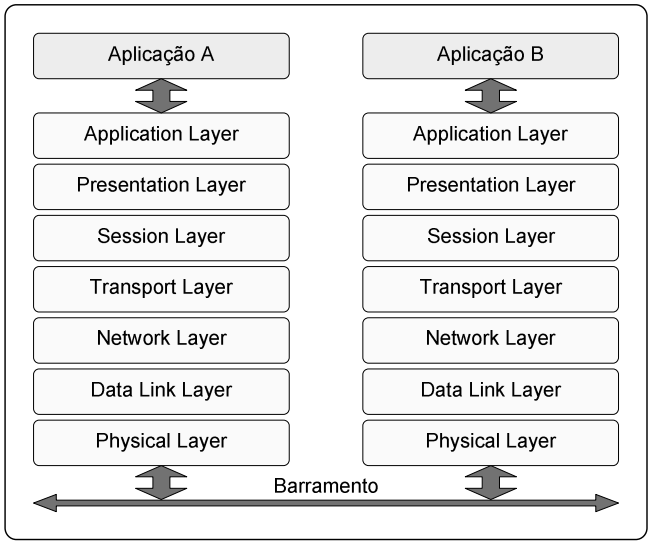

Fig. A.4. Modelo de Referência de 7 camadas ISO/OSI [8].

A essência do modelo consiste na transmissão de informação entre duas aplicações, A e B, por métodos que são desconhecidos à própria aplicação. Como as duas aplicações operam em máquinas diferentes, elas são conhecidas como aplicações remotas (*remote application process*). Essas aplicações ligam ao sistema de comunicações utilizando os serviços da camada superior do modelo de referência, a *Application Layer*. Por sua vez, a *Application Layer* utiliza os serviços da camada inferior a ela, a *Presentation Layer*, sem necessidade de conhecer nenhuma das camadas inferiores. Há 7 camadas no modelo, cada uma das camadas é usada pela camada imediatamente anterior a ela. Por sua vez, cada uma das camadas usa os serviços da camada imediatamente depois.

A informação é transmitida entre duas aplicações, sendo passada através das várias camadas imediatamente inferiores antes de ser transmitida para o meio físico de comunicação. A informação é transferida entre camadas na forma de *data units* (DU). Cada camada pode adicionar informação de controlo a cada DU, e essa informação de controlo é utilizada para organizar a comunicação entre as entidades dos dois sistemas remotos.

Cada camada passa a sua informação para a camada inferior na forma de *protocol data unit* (PDU). A camada seguinte adiciona *protocol control information* (PCI) a essa DU, agora conhecida como *service data unit* (SDU), para produzir o seu próprio PDU. Este novo PDU é depois passado para baixo para a próxima camada, e depois para baixo até a *Physical Layer*. Com isto, a *Physical Layer* possui uma *data frame* completa e transmissível que pode ser transmitida através do meio de transmissão. Este processo pode ser verificado na figura A.5.

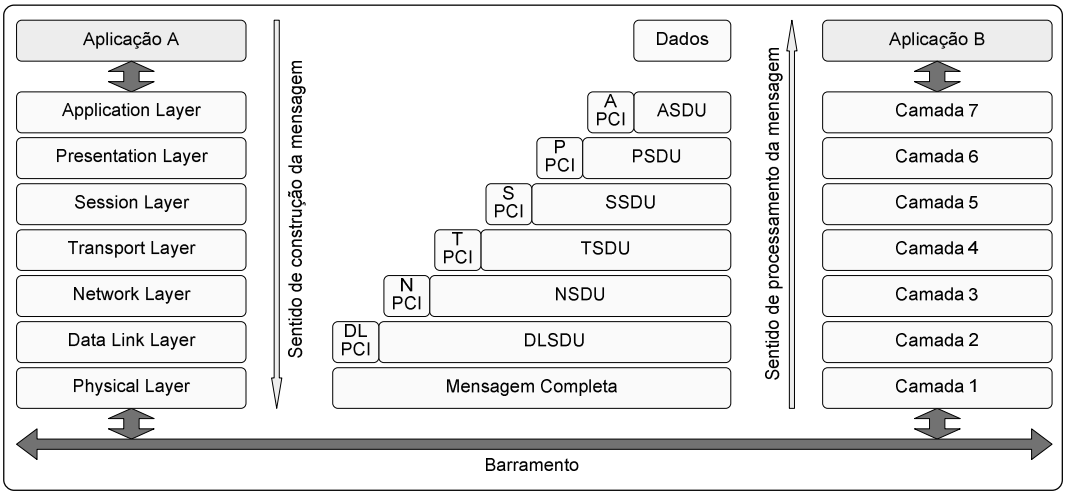

Fig. A.5. Esquematização da construção da mensagem no Modelo de Referência ISO/OSI [8].

A mensagem completa transmitida é depois recebida pelo segundo sistema. Conforme a mensagem recebida vai passando através da estrutura de camadas, no sentido ascendente, do sistema receptor, cada PCI vai sendo retirado na sua respectiva camada. Quando a informação passa pela *Application Layer*, apenas resta a informação da aplicação. A informação contida nos vários PCI, adicionada nas camadas do sistema A, é removida nas camadas do sistema B.

Com este conceito, os serviços de cada camada funcional do Modelo de Referência podem ser isolados dos outros, evitando os problemas de interacção entre funções que podem acontecer em arquitecturas menos bem definidas [8].

Cada camada comunica com a camada inferior invocando serviços que essa camada oferece, ou funções particulares com esses serviços, conhecidas como *service elements*.

O Modelo de Referência ISO/OSI foi desenvolvido de modo a abranger as comunicações mais complexas. Alguns sistemas de comunicação podem não usar a totalidade das sete camadas do modelo. Isto pode ser devido ao facto de algumas redes de comunicações não terem parâmetros muito estritos e limitativos, por isso a funcionalidade de algumas camadas não é necessária.

A norma de comunicações CANopen apenas utiliza a funcionalidade das camadas *Application*, *Data Link* e *Physical Layers*.

A.4. CAN: CONTROLLER AREA NETWORK

O *Controller Area Network* (CAN) foi desenvolvido nos anos 80 por Robert Bosch GmbH [8], [9], [10]. Um dos seus primeiros usos foi a interconexão de componentes de controlo de veículos, tais como o sistema anti-bloqueio (ABS) e o sistema de controlo da aceleração (ASC). A complexidade das funções de controlo implementadas nestes sistemas requer mecanismos de transmissão de informação, que normalmente eram implementadas recorrendo a linhas de controlo dedicadas. Contudo, assim como a complexidade dos dispositivos vai aumentando, o número de linhas de controlo e conexões requeridas também vai aumentando. Isto leva à situação onde o número de linhas de controlo e de conexões, de muitos dos sistemas de controlo dos veículos da actualidade, já ultrapassaram o seu limite físico máximo. Além disso, agora há sistemas, nos veículos, que requerem uma troca de informação entre um elevado número de dispositivos.

Não levou muito tempo para que esta ideia migrasse dos veículos para os restantes mercados. Hoje em dia o CAN é encontrado em diversas áreas como na maquinaria agrícola, na instrumentação médica, nos controlos dos elevadores e nos componentes de controlo da indústria automatizada. Este uso generalizado acontece porque um grande número de produtores de semicondutores agora também produzem dispositivos CAN, tornando-os relativamente baratos. Só o facto de companhias como a Philips®, a Motorola®, a National Semiconductors®, a Siemens® e a Intel® estarem envolvidas no

desenvolvimento de tecnologia CAN, é um forte indicador de que esta tecnologia será bem sucedida no futuro.

A especificação original desenvolvida por Robert Bosch GmbH é agora chamada *CAN Specification 2.0 – Part A* [8], [11]. A razão para isto deve-se à realização de um upgrade que contém as duas partes, a *Part A* e uma nova versão do CAN chamada *Part B*, sendo esta última compatível com a anterior [12]. A diferença principal entre as duas versões reside no facto de a versão anterior definir uma mensagem CAN com um campo do identificador de 11 bits enquanto que a última versão permite usar um identificador de 29 bits.

Em 1993, o CAN passou a ser um standard ISO. O *ISO Draft International Standard* com referência ISO11898 está baseado no *CAN Specification 2.0* e adopta o formato clássico das mensagens CAN de identificadores de 11 bits. O *ISO11898 Standard* relaciona o CAN com o *ISO/OSI Reference Model*, especificando a *CAN Physical Layer* e a *CAN Data Link Layer* [8].

A *CAN Specification 2.0* não define completamente a *Physical Layer*, não obstante são especificados alguns dos requisitos, que devem ser conhecidos pela *Physical Layer*. Estes requisitos são a codificação e descodificação dos bits, o *bit timing* e a sua sincronização. De facto, muito da *Physical Layer* foi deixada para que o utilizador defina a sua aplicação específica. Na actualidade a definição *ISO11898 CAN Physical Layer* é usada ao longo de todo o campo de aplicação do CAN e é adoptada também no CANopen.

A ideia básica por trás do CAN é ligar todos os dispositivos de controlo através de um único conjunto de cabos, onde a informação pode ser inteiramente partilhada, não apenas entre dois dispositivos mas entre numerosos dispositivos.

O CAN é um protocolo de comunicação série [15], [22], [28], especialmente desenvolvido para criar redes entre numerosos dispositivos inteligentes, incluindo sensores e actuadores. É um sistema com capacidades *multi-master*, isto é, os dispositivos CAN podem tentar transmitir informação a qualquer momento. Consequentemente, no sistema CAN, pode ocorrer a transmissão simultânea de

informação a partir de vários nós. Cada mensagem transporta um identificador que estabelece um sistema de prioridades e confere o critério de arbitrariedade à rede: o nó com a maior prioridade terá acesso ao barramento e os nós restantes terão de esperar.

O identificador é um número transportado num campo da mensagem CAN e determina a prioridade da mensagem. As mensagens são enviadas a todos os nós CAN. Depois, cada nó decide, com base no identificador recebido, se a mensagem lhe é dirigida e, em caso afirmativo, deve processar a mensagem.

A funcionalidade especificada no *CAN Specification 2.0* pode ser interpretada, em termos do *ISO/OSI Reference Model*, como sendo um protocolo de duas camadas, como representado na figura A.6.

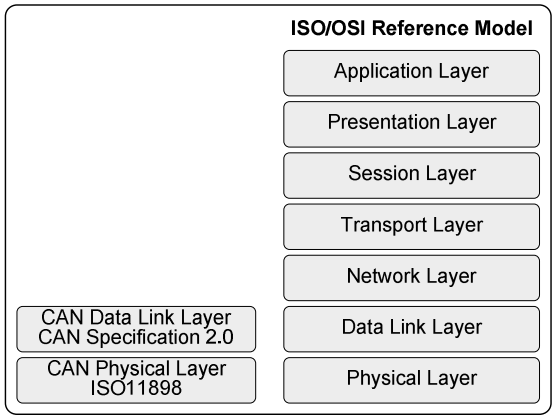

Fig. A.6. O protocolo CAN e o *ISO/OSI Reference Model* [8].

A.4.1. A CAN Physical Layer

A *Physical Layer* está encarregue de transformar ou codificar a informação passada para ela num sinal que possa ser enviado através do meio de transmissão. A *Physical Layer* tem também as funções:

- Traduzir a informação numa forma de sinal transmissível e traduzir as mensagens que chegam para uma forma de informação utilizável.

- Gerar sinais de controlo para um funcionamento apropriado da *Data Link Layer*. A *Physical Layer* detecta e assinala os eventos mas é a *Data Link Layer* que toma decisões e realiza as acções no sinal.
- Define ligações físicas para o meio.

O meio físico tipicamente usado para implementar redes CAN é um barramento de dois cabos dispostos conforme mostra a figura A.7.

Os dois cabos são chamados CAN_H e CAN_L, de acordo com a polaridade da tensão diferencial entre eles. O barramento CAN também tem de ser terminado em ambos os extremos por resistências do valor recomendado de 124Ω.

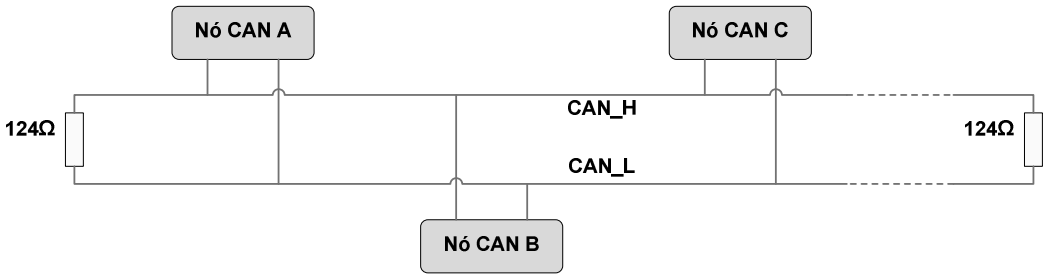

Fig. A.7. Diagrama de blocos de uma rede CAN [8].

O comprimento máximo do barramento CAN depende do *baudrate* ao qual será usado [28]. O CANopen especifica um conjunto de *baudrates* que poderão ser implementados por dispositivos CANopen. Cada um dos *baudrates* tem um comprimento de barramento recomendado, como representado na tabela A.1.

A representação do bit utilizada para transmitir mensagens no barramento CAN é a codificação NRZ (*non-return-to-zero*) que garante uma eficiência máxima na codificação dos bits. Contudo, a necessidade de transições frequentes no barramento (de modo a manter a sincronização) obrigam a implementação de um mecanismo chamado *bit-stuff*. Neste mecanismo o transmissor insere um bit depois de cinco bits consecutivos da mesma polaridade. O bit inserido tem polaridade inversa de modo a forçar a ocorrência de transições no barramento CAN. O receptor remove o bit antes de processar a mensagem. Este mecanismo é também usado para controlo de erros.

Tabela A.1. *Baudrates* e comprimentos do barramento recomendados no CANopen [8].

Baudrate	**Comprimento do barramento**
1000 kbit/s	25 m
800 kbit/s	50 m
500 kbit/s	100 m
250 kbit/s	250 m
125 kbit/s	500 m
50 kbit/s	1000 m
20 kbit/s	2500 m
10 kbit/s	5000 m

A.4.1.1. Configuração do Bit Timing

De modo a explicar o processo típico de configuração de um controlador CAN para comunicar a um determinado *baudrate*, é necessário introduzir conceitos definidos nas especificações CAN relativamente ao *bit timing*. As especificações definem três parâmetros:

- O *Nominal Bit Rate* (NBR) é basicamente o número de bits por segundo que corresponde ao *baudrate* de comunicação desejado.
- O *Nominal Bit Time* (NBT) é definido como 1/NBR. A configuração do *bit timing* de uma aplicação CAN é definida através do parâmetro NBT.
- O *Time Quantum* (TQ) é uma unidade de tempo fixa derivada do período do oscilador.

O processo de programação de um *baudrate* particular consiste em estabelecer a duração do TQ e, consequentemente, a duração do NBT com base no TQ. A estrutura do parâmetro NBT é definido nas especificações CAN como um conjunto de segmentos de tempo, conforme mostra a figura A.8.

O SYNC_SEG é a parte do tempo usado pelo bit para sincronizar os diferentes relógios dos nós ligados ao barramento e onde deve terminar a transição do bit anterior para o bit

corrente. O PROP_SEG considera atrasos de propagação no barramento. O PHASE_SEG1 e o PHASE_SEG2 podem ser automaticamente ajustados pelo controlador CAN para compensar erros de sincronização. No CANopen é recomendado que, por defeito, o *bit timing* num nó seja preparado para suportar o atraso máximo de propagação no barramento.

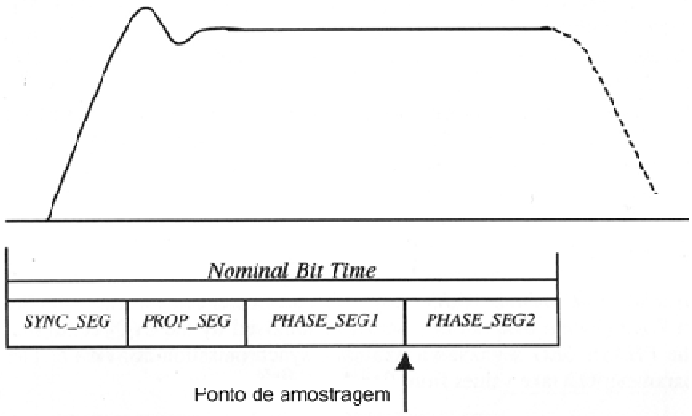

Fig. A.8. Estrutura do parâmetro *Nominal Bit Time* [8].

O comprimento total do NBT pode ser definido no valor desejado. Além disso, o ponto de amostragem pode ser colocado em diferentes localizações dentro do NBT. As especificações CAN estabelecem algumas limitações nos comprimentos dos segmentos:

- O SYNC_SEG tem de ser sempre 1 TQ de comprimento.
- O PROP_SEG é programável de 1 a 8 TQ de comprimento.
- O PHASE_SEG1 é programável de 1 a 8 TQ de comprimento.
- O PHASE_SEG2 é programável de 1 a 8 TQ de comprimento, mas é aconselhado que seja definido como igual ao PHASE_SEG1 e nunca menor que 2TQ [23].

A forma como estes parâmetros são configurados vai depender do controlador CAN utilizado.

A.4.1.2. Ligações Físicas

A ligação de um controlador CAN a um barramento CAN tem de ser feita usando um *transceiver* CAN que esteja de acordo com ISO11898. A ligação entre o controlador CAN e o *transceiver* pode ser directa ou pode ser feita recorrendo a isolamento óptico. O uso de isolamento óptico é importante para proteger o dispositivo de sobrecargas eléctricas originadas no barramento. Adicionalmente, se todos os dispositivos na rede CAN utilizam isolamento óptico, o comum é descartável e apenas serão necessários dois cabos para a comunicação: CAN_L e CAN_H.

Uma ligação directa entre o controlador CAN e o *transceiver* é semelhante ao circuito da figura A.9. Os sinais TxD e RxD são sinais usados pelo controlador para enviar e receber informação. O *transceiver* faz a conversão desses sinais num formato diferencial usado no barramento CAN ou converte o sinal do barramento CAN num sinal legível pelo controlador.

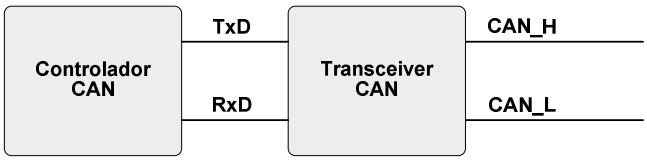

Fig. A.9. Ligação directa entre o controlador CAN e o *transceiver* CAN [8].

Alguns controladores CAN interpretam os sinais série TxD e RxD de forma diferencial, comparando-os com a referência. Quando se verifica este caso, haverá quatro cabos que ligam o controlador ao *transceiver*: Tx0, Tx1, Rx0 e Rx1. O Rx0 e o Tx0 estão ligados à referência: o Tx0 no lado do controlador e o Rx0 no lado do *transceiver*.

Uma ligação com isolamento óptico entre o controlador e o *transceiver* é mostrado na figura A.10. Os acoplamentos ópticos usados nesta situação têm de ser de alta velocidade, especialmente desenvolvidos para este tipo de aplicações.

Relativamente à ligação física, o CANopen recomenda um conector de 9 pinos D_Sub standard (DIN41652) [18], representado na figura A.11. Contudo, são permitidos outros tipos de conectores.

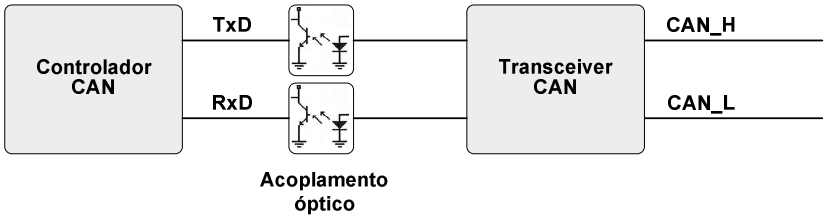

Fig. A.10. Ligação entre o controlador e o *transceiver* com isolamento óptico [8].

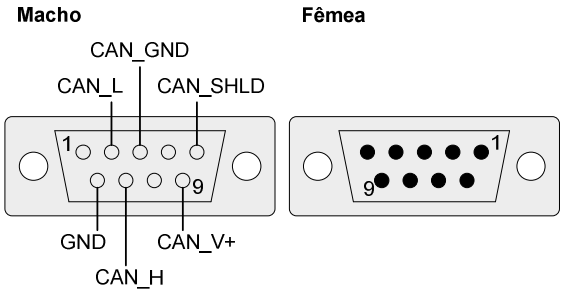

Fig. A.11. Conector standard de 9 pinos D_Sub [18].

A descrição de cada um dos pinos encontra-se na tabela A.2.

Tabela A.2. Descrição dos pinos do conector CAN [18].

Pino	Sinal	Descrição
1	Reserved	Pino reservado
2	CAN_L	Linha de barramento LOW
3	CAN_GND	Referência do CAN
4	Reserved	Pino reservado
5	(CAN_SHLD)	Blindagem opcional CAN
6	(GND)	Referência opcional CAN
7	CAN_H	Linha de barramento HIGH
8	Reserved	Linha de erro
9	(CAN_V+)	Fonte de tensão externa opcional

A.4.1.3. Funcionamento Básico do Barramento CAN

Um barramento CAN e os nós a ele conectados, em teoria, assemelham-se ao circuito representado na figura A.12.

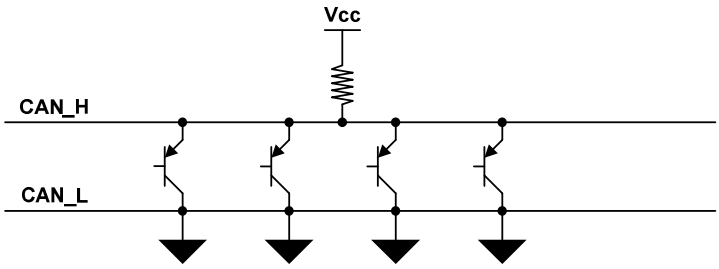

Fig. A.12. Configuração *wired_AND* [8].

Cada um dos transístores presentes na figura A.12 representa a ligação de um nó CAN ao barramento. Cada nó pode transmitir um 1 lógico colocando o transístor em estado OFF (existe uma tensão Vcc entre as duas linhas) ou um 0 lógico colocando o transístor em estado ON (ambas as linhas ficam à mesma tensão de 0V). Este circuito representa o que é chamado de configuração *wired-AND* porque todos os nós têm de estar a transmitir um 1 lógico para haver um 1 lógico no barramento. Para que haja um 0 lógico no barramento, é suficiente que um único nó transmita um 0 lógico. Esta é a razão pela qual o 0 é o valor dominante e o 1 é o valor recessivo. Este mecanismo é a base de funcionamento da *CAN Data Link Layer*, que é discutida de seguida.

A.4.2. A CAN Data Link Layer

Independentemente do meio de transmissão utilizado, podem aparecer erros durante o processo de transmissão da informação. A intenção da *Data Link Layer* é lidar com esses erros de modo a minimizar o número de erros enviados para as camadas superiores.

Adicionalmente, a *Data Link Layer* trabalha em três outras áreas da comunicação:

- Controlo do tráfego: isto é usado para cobrir os problemas inerentes ao facto de haver um transmissor rápido que envia para um receptor lento.
- Gestão da ligação: isto lida com as regras que um transmissor e um receptor têm que seguir de forma a trocar a informação. Por exemplo, um transmissor e um receptor podem necessitar identificar-se, um relativamente ao outro, e estar preparados para comunicar antes de trocar qualquer informação.
- Controlo de acesso ao meio: isto controla o acesso ao meio de transmissão. O seu propósito é lidar com o problema de dois ou mais transmissores enviarem informação ao mesmo tempo. Estes reconhecem a colisão de informação e resolvem o problema de modo a que a informação não seja perdida.

A *Data Link Layer* pode ser dividida em duas sub-camadas: o *Medium Access Control* e o *Logical Link Control*. O *Medium Access Control* é responsável por determinar que nó da rede tem direito a aceder ao barramento para transmitir a mensagem. O *Logical Link Control* é responsável pela detecção e correcção de erros quando a informação é trocada através do barramento.

A.4.2.1. A Estrutura da Mensagem

A *Data Link Layer* agrega a informação a ser transmitida em grupos (*Data Frames*) como o mostrado na figura A.13.

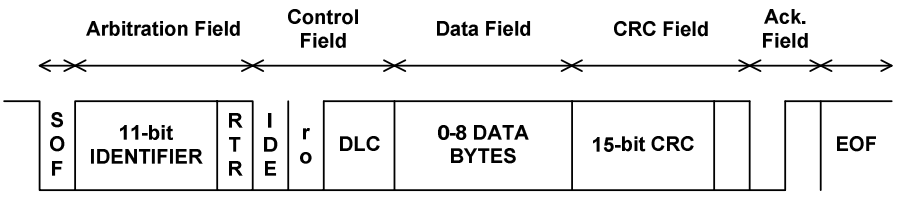

Fig. A.13. Formato da mensagem segundo *CAN 2.0-Part B* [8].

Este formato é definido nas especificações CAN 2.0-Part B para mensagens com identificadores de 11 bits [12], [28].

Como se vê na figura A.13, a mensagem é composta por um determinado número de campos:

- *Start Of Frame* (SOF) *Field*: este campo é composto por um bit de sincronização dominante.
- *Arbitration Field*: este campo tem duas componentes. A primeira é um identificador de 11 bits que estabelece a prioridade da mensagem e a sua identidade. A segunda é o bit *Remote Transmition Request* (RTR) que, se for definido como dominante, indica que a mensagem contém informação e, se for definido como recessivo, indica uma *Remote Frame*. As Remote Frames são exactamente o mesmo que as *Data Frames* excepto que elas não transportam informação e são usadas para requerer a transmissão de informação com o mesmo identificador por outro nó do barramento.
- *Control Field*: este campo tem três componentes. A primeira é o *Bit Identifier Extension* (IDE) que, se for definido como recessivo, indica a transmissão de mensagem com identificador de 29 bits, se for definido como dominante, indica a transmissão de mensagem com identificador de 11 bits. A segunda componente é um bit reservado. A terceira componente é de 4 bits com o nome de *Data Length Code* (DLC). O campo DLC pode conter valores entre 0 e 8, indicando o número de bytes de informação na mensagem.
- *Data Field*: este é o campo onde a informação é enviada. O tamanho da informação pode ser entre 0 e 8 bytes.
- *Cyclic Redundancy Check (CRC) Field*: este campo é usado para verificar os erros e contém um código de 15 bits seguido de um bit delimitador recessivo.
- *Ack. Field*: este campo é usado para assegurar que uma mensagem foi bem recebida por outro nó. A recepção da mensagem é reconhecida inserindo um bit dominante na primeira posição deste campo. Este bit é sempre transmitido como recessivo pelo transmissor mas, na resposta, é esperado ser detectado como dominante pelo transmissor. Se o campo *Ack. Field* não for preenchido pelo nó receptor com um bit dominante, o transmissor sabe que a mensagem não foi recebida correctamente e cancela a transmissão. O segundo bit deste campo é um bit de delimitação recessivo.
- *End Of Frame* (EOF) *Field*: este campo é composto por sete bits recessivos.

Para além de *Data Frames* e *Remote Frames*, o protocolo CAN também usa *Error Frames* e *Overload Frames*.

As *Error Frames* são compostas por seis bits consecutivos da mesma polaridade e são usadas para abortar a transmissão. As *Overload Frames* são usadas pelo receptor para atrasar a transmissão quando não está preparado para receber as mensagens.

Todas as mensagens CAN devem ser seguidas de pelo menos três bits recessivos. Isto é chamado de período *Intermission*. Depois deste período pode ser iniciada uma nova transmissão.

A.4.2.2. A Filtragem das Mensagens

Quando uma mensagem é transmitida no barramento, não é endereçada a nenhum dispositivo. Em vez disso, cada mensagem é designada por um identificador único de 11 bits, como mostra a figura A.13, no campo *Arbitration Field*. Se um nó deseja transmitir informação, ele simplesmente passa a informação e o identificador para o seu controlador CAN e activa a requisição de transmissão. É depois o controlador CAN que formata o conteúdo de mensagem e transmite a informação na forma de uma mensagem CAN. Quando o nó tiver acesso ao barramento e enviar a sua mensagem, todos os outros nós se tornam receptores. Tendo recebido a mensagem correctamente, os nós realizam um teste para verificar se a mensagem é relevante para eles. Este processo é chamado de *message filtering*. A comunicação não tem que ser apenas de um nó para outro nó. Com uma única transmissão, vários nós podem aceitar a mesma mensagem.

A.4.2.3. A Arbitragem do Barramento

O identificador de uma mensagem CAN define a prioridade da mensagem e é a base do mecanismo de controlo de acesso ao meio (*Medium Access Control*). O funcionamento do MAC pode ser descrito do seguinte modo:

- Quando um dispositivo detecta que o barramento está livre para transmissão, começa a transmitir a mensagem.
- Durante a transmissão, o dispositivo mantém-se a monitorizar o barramento, verificando se o barramento está a responder à sua transmissão.
- Se, durante a transmissão do campo *Arbitration Field*, o nó tenta transmitir um bit recessivo mas detecta um bit dominante no barramento, pára de transmitir e espera que o barramento esteja livre de novo. Quando isso acontecer, o processo pode recomeçar.

Um bit dominante enviado por um nó para o barramento prevalece sempre sobre qualquer número de bits recessivos enviados para o barramento por outros nós. Sempre que um nó tenta transmitir, vai manter o controlo do barramento se a mensagem a ser transmitida contém o identificador com o valor mais baixo. Este mecanismo é a essência do MAC no CAN e está representado na figura A.14.

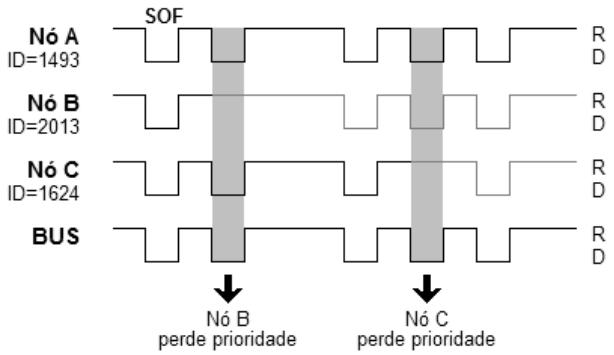

Fig. A.14. Exemplo de funcionamento do mecanismo MAC.

A.4.2.4. A Detecção de Erros

Adicionalmente ao MAC, a *Data Link Layer* é também responsável pelo *Logical Link Control*, que fornece um mecanismo de transmissão de confiança, através da implementação de uma série de mecanismos de detecção de erros. Os mecanismos são:

- *Bit Error* – um nó, depois de transmitir o *Arbitration Field* está ainda a monitorizar o barramento, verificando se este está a responder correctamente à transmissão. Isto permite a detecção de erros no barramento e erros de transmissão locais causados pelo transmissor. Há excepções a esta regra, tais como a transmissão de um bit recessivo e a detecção de um bit dominante nos campos *Arbitration* e *Ack. Fields*.
- *Bit Stuffing Error* – é usado para assegurar o número mínimo de transmissões, para propósitos de sincronização. É também usado para controlo de erros de transmissão quando um receptor detecta seis bits consecutivos iguais. Nesta situação sabe que houve um erro na transmissão de bits.
- *Cyclic Redundancy Check Error* – é executado usando o campo *CRC Field* na mensagem CAN. O transmissor calcula um código CRC baseado no conteúdo da mensagem que está a enviar e insere esse código no campo *CRC Field*. Quando a mensagem chega ao receptor, o código CRC é recalculado e comparado com o código recebido. Se os códigos forem diferentes, aconteceu algum erro CRC.
- *Form Error* – há um mecanismo que verifica a estrutura da mensagem transmitida analisando os campos de bits contidos nela em contraste com o formato e tamanho padrão.
- *Acknowledgement Error* – este mecanismo já foi mencionado anteriormente. Todos os receptores colocam um bit dominante no campo *Ack*. como forma de reconhecimento da mensagem. Se isso não acontece, aconteceu um erro de transmissão e nenhum nó recebeu a mensagem de forma correcta.

Um nó CAN, que detecte um erro de transmissão, tem capacidade de abortar a transmissão enviando um aviso de erro (*Error Flag*) para os outros nós do barramento. O tipo de *Error Flag* (activa ou passiva) depende da condição de detecção do erro. De facto, relativamente ao erro, um dispositivo pode ter um de três estados:

- *Error Active* – o dispositivo tem a certeza que aconteceu um erro e que esse erro não foi seu. Nesta situação, os erros são assinalados usando *Active Error Flags*. Uma *Active Error Flag* é composta por seis bits dominantes fazendo com que qualquer transmissão no barramento seja interrompida. Isto vai accionar o mecanismo de detecção de erros *Bit Stuffing Rule*, prevenindo outros nós de

receber mensagens erróneas e tentarem activar uma retransmissão por parte do transmissor.

- *Error Passive* – é quando um dispositivo detecta um erro e suspeita que a fonte de erro pode ser local. Por esta razão, o erro é assinalado usando *Passive Error Flags*. Uma vez que as *Passive Error Flags* são compostas por seis bits recessivos, geralmente a transmissão só é abortada se o dispositivo é o transmissor da mensagem ou se é o único receptor.
- *Bus-off* – neste caso, o dispositivo acredita que é a causa do erro, isto é, ele não é capaz de comunicar correctamente. Portanto, o nó erróneo cessa a sua participação nas comunicações de modo a não perturbar os outros nós. Esta situação normalmente previne nós erróneos de congestionar o barramento e, como resultado disso, permite um funcionamento normal dos outros nós.

Para determinar em que situação de erro o nó está, o CAN especifica que cada unidade tem de ter dois contadores: um para erros de transmissão e outro para erros de recepção. As contagens desses contadores incrementam quando são detectados erros e decrementam quando a mensagem é transferida com sucesso. Conforme os erros vão acontecendo, o contador vai incrementando e, eventualmente, atinge um limiar que faz com que o nó entre em estado de *Error Passive*. Se os erros continuam, o contador vai continuando a incrementar até que o nó se desconecta, entrando em estado de *Bus-off*.

A.4.3. Implementações de Hardware CAN

Quando se implementa qualquer tipo de dispositivo que comunica usando CAN, o projectista não tem de estar preocupado com a comunicação CAN em si, pois toda a funcionalidade requerida está implementada no *hardware* [8]. Essas implementações de *hardware* CAN são chamadas *CAN Controllers*. Tudo o que é requerido pelo projectista é:

- Conhecimento básico de como o protocolo CAN funciona.

- Implementação da interface de *hardware* entre o controlador CAN e o barramento CAN (possivelmente também entre o controlador CAN e o microcontrolador).
- Capacidade de programar o controlador CAN de acordo com as necessidades da aplicação.

A.4.3.1. Implementações Basic CAN vs Full CAN

Relativamente à filtragem das mensagens, podem ser identificadas duas categorias de implementações CAN. Essas categorias são conhecidas como implementações *Basic CAN* e *Full CAN*. A diferença entre as duas é cada vez menos importante, considerando que a maior parte dos novos controladores *Full CAN* também disponibilizam a funcionalidade *Basic CAN*.

Nos controladores *Full CAN*, são reservadas secções individuais ou objectos do buffer de memória da mensagem, para a recepção ou transmissão de mensagens, com identificadores pré-estabelecidos programáveis. Quando é recebida uma mensagem, se o identificador coincide com o programado no objecto, a informação é guardada nesse objecto ou posição de memória. Se o identificador não coincide com nenhum dos identificadores programados, a mensagem é rejeitada pelo *hardware* (ver figura A.15). Este tipo de montagem pode poupar o projectista de problemas no desenvolvimento do *software* CAN, comparativamente com o requerido nos controladores *Basic CAN*.

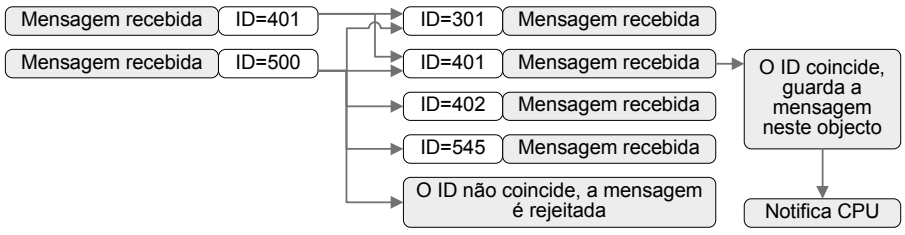

Fig. A.15. Princípio de funcionamento do *Full CAN* [8].

Nos controladores *Basic CAN*, o controlador recebe todas as mensagens independentemente dos seus identificadores e coloca-as num buffer de recepção de

mensagens. Sempre que há a recepção de uma mensagem, é invocada uma rotina do *software*, independentemente de a mensagem ser destinada à aplicação ou não. Isto pode complicar bastante o *software*. Contudo, pode ser reduzido em algumas situações nas quais um controlador *Basic CAN* tem um filtro de aceitação que permite que o controlador aceite todos ou um conjunto de identificadores CAN dentro de um intervalo. Por exemplo no microcontrolador 8x592 da Philips®, o filtro consiste num registo *Acceptance Mask* (AM) e num registo *Acceptance Code* (AC), como mostrado na figura A.16.

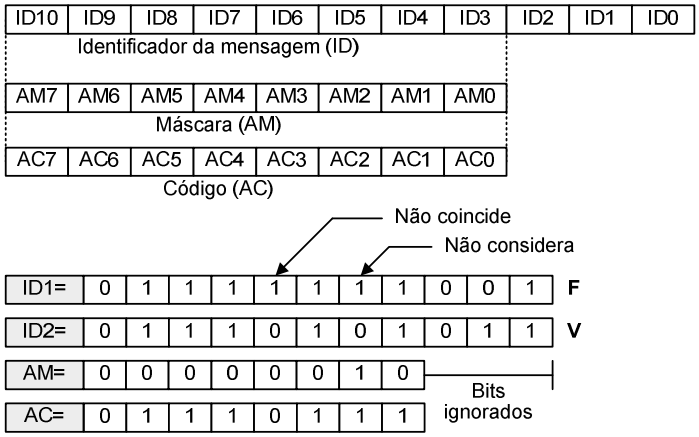

Fig. A.16. Registos de filtragem do *Basic CAN* [8].

Os registos AC e AM normalmente têm o tamanho de 8 bits e a filtragem é baseada nos 8 bits mais altos do identificador CAN. O registo AM define quais os bits correspondentes do AC têm de coincidir para que a mensagem seja aceite. Isto é feito colocando os bits que interessam analisar a 0.

Na figura A.16 podemos ver que o ID1 não passa no teste enquanto o ID2 passa. Pode-se verificar que os bits do identificador para os quais o AM tem valor 1 não são considerados. O mesmo aplica-se aos últimos três bits do identificador, onde não há filtragem.

Se, no registo AM estiverem todos os bits definidos como *must-match* (ou seja a 0) e o AC do exemplo tiver o código 01110111, o intervalo de identificadores que passaria no teste do controlador CAN seria de 952 a 959 (ver a figura A.17).

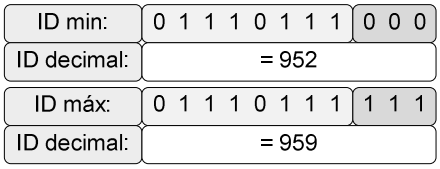

Fig. A.17. Exemplo de código num registo AC.

A.4.3.2. Controladores Stand-alone vs Embedded

Os controladores *Stand-alone CAN* são implementações *hardware* num único circuito integrado físico. Por causa disso, é necessário implementar toda a funcionalidade requerida para partilhar um barramento de informação e para funcionar como um periférico típico em circuitos baseados em microprocessadores.

Os controladores *Embedded CAN* são implementações de *hardware* fornecidas como um periférico adicional dentro de um microcontrolador. Este tipo de controlador está implementado no mesmo chip que o microprocessador e, com isso, todas as ligações físicas necessárias entre os dois dispositivos já estão lá.

Os dois tipos de implementações podem ser vistas na figura A.18. É de notar que as ligações entre os controladores e o *transceiver* são diferentes em cada caso.

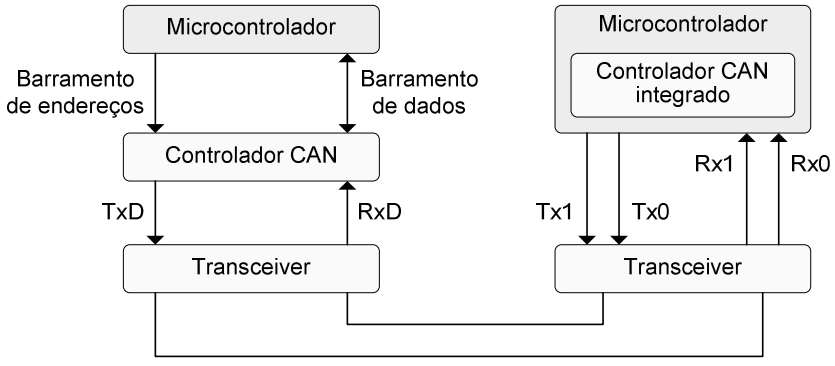

Fig. A.18. Representação dos dois tipos de implementações de *hardware* CAN [8].

A.5. CANOPEN

A especificação CAN pode ser caracterizada nas duas camadas mais baixas do *ISO/OSI Reference Model*: a *Physical Layer* e a *Data Link Layer*. O *CAN Data Link Layer* oferece serviços de transferência e requisição de informação, não maior que 8 bytes. Esses serviços são suficientes para o universo original das aplicações nas quais o CAN era usado. Contudo, nas aplicações distribuídas mais complexas, é requerida uma funcionalidade adicional, tal como a inicialização coordenada dos nós ou a transmissão de blocos de informação maiores que 8 bytes.

Em termos do *ISO/OSI Reference Model*, isso significa que são necessários serviços de camadas mais altas. Essa foi a razão pela qual a CiA especificou o *CAN Reference Model* e, dentro deste, a *CAN Application Layer* (CAL). O horizonte de aplicação do CAN foi muito alargado quando a CAL foi lançada e, nos dias de hoje, a CAL é usada numa ampla gama de campos de aplicação desde aplicações médicas a controlo do trânsito.

O *CAN Reference Model* e a forma como ele se relaciona com o *ISO/OSI Reference Model* é mostrado na figura A.19.

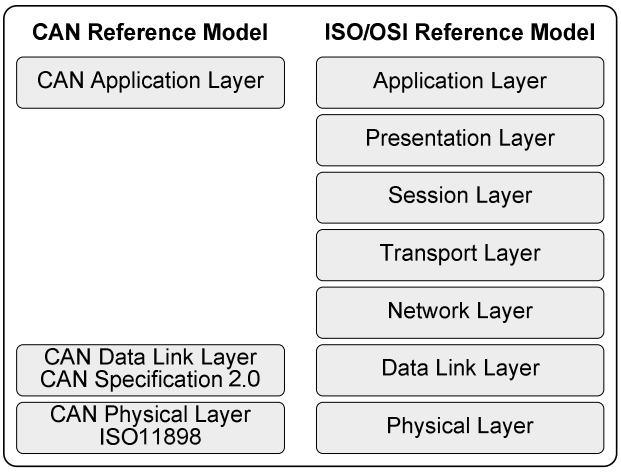

Fig. A.19. O *CAN Reference Model* [8].

Como pode ser visto na figura A.19, o *CAN Reference Model* é um modelo com três camadas [14], sendo estas a *Physical Layer*, a *Data Link Layer* e a *Application Layer*, o que é comum nos protocolos *fieldbus*.

A *Physical Layer* e a *Data Link Layer* deste modelo já foram discutidas anteriormente. A *Application Layer* é, em termos gerais, a *interface* entre o ambiente de comunicação de informação e a aplicação que usa esse ambiente para cooperar com outras aplicações.

No *CAN Reference Model*, as comunicações entre aplicações podem ser conseguidas utilizando os serviços disponibilizados pela *CAN Application Layer*. Esses serviços permitem a troca de informação através do barramento CAN, usando os serviços da *Data Link Layer* e da *Physical Layer*, de acordo com o protocolo especificado na CAL.

A *CAN Application Layer* fornece um vasto número de serviços, que são divididos por quatro grupos, de acordo com a sua natureza [29]. Esses grupos são:

- *CAN-based Message Specification* (CMS) – oferece uma comunicação aberta, orientada para o ambiente dos objectos no qual se desenvolvem as aplicações do utilizador. É uma linguagem usada para especificar que COBs (*Communication Objects*, mensagens CAN com identificadores específicos) um módulo usa e como eles são formatados, e pode descrever todas as características da *CAN Data Link Layer*. O CMS usa variáveis, eventos e objectos para especificar como pode ser acedida a funcionalidade de um módulo através da *interface* CAN e também inclui regras de codificação de informação que definem como codificar e descodificar a informação da aplicação para a sintaxe de transferência e vice-versa.
- *Network Management* (NMT) – oferece uma comunicação aberta, orientada para o ambiente dos objectos no qual um módulo (o *NMT Master*) tem capacidade de lidar com a inicialização e possíveis falhas dos outros nós (os *NMT Slaves*). O NMT fornece as ferramentas básicas para a construção e manutenção de aplicações distribuídas.
- *Distributor* (DBT) – manuseia um problema essencial na construção de uma aplicação CAN, isto é a localização dos identificadores. Um identificador determina a prioridade de uma mensagem e, com isso, o valor usado por um

determinado módulo CAN não pode ser fixo pelo seu produtor. Em vez disso, o integrador do sistema deve ter um controlo total sobre todos os identificadores. O DBT oferece um mecanismo para a distribuição dinâmica dos identificadores a partir de um módulo (o *DBT Master*) para os outros módulos (os *DBT Slaves*).

- *Layer Management* (LMT) – oferece a possibilidade de deixar um nó *LMT Master* controlar as propriedades de certos parâmetros das camadas de outros módulos (*LMT Slaves*).

Normalmente a funcionalidade NMT, DBT e LMT do *master* está centrada num único dispositivo que controla o funcionamento da rede. Isto, contudo, não é obrigatório e as funções podem ser executadas por vários dispositivos.

Como é mostrado de seguida, o CANopen foi definido com base na *CAN Application Layer*. De facto, em redes CANopen, apenas é usado um subconjunto dos serviços especificados na CAL.

A.5.1. A Estrutura da Especificação CANopen

Um conceito muito importante quando se fala sobre o CANopen é o conceito de perfil. Na terminologia do protocolo, um perfil é uma especificação de protocolo que é baseada em outras especificações de protocolos já existentes. Normalmente, um perfil define um subconjunto dos serviços prestados pelos protocolos existentes que podem ser usados para comunicação, e restringe a forma em como esses serviços podem ser aplicados.

A especificação CANopen é composta por um conjunto de perfis baseados no *CAN Reference Model*. Esses perfis são divididos em duas categorias:

- *CANopen Device Profiles* – um *Device Profile* define a forma como a funcionalidade de um tipo de dispositivo particular é tornada acessível através do barramento, e que ferramentas *CANopen Communication Profile* são necessárias para aceder a essa funcionalidade. Teoricamente, deveria haver um *Device Profile* para cada tipo de dispositivo que possa ser usado num ambiente industrial, ligado a um barramento CAN. Actualmente há alguns *Device Profiles*

standard, tais como o *I/O Modules* (DS401), *Drives and Motion Control* (DS402), *Human Machine Interfaces* (DS403), entre outros [13]. Também continuam a ser desenvolvidos outros *Device Profiles*, pela CiA, para várias categorias de dispositivos que ainda não estão abrangidos pelas especificações existentes [13].

- *CANopen Communication Profile*, DS301 – o *CANopen Communication Profile* é aplicável a todos os dispositivos. Define como um subconjunto de serviços da CAL podem ser usados para comunicar com um dispositivo CANopen e como é esperado que esse dispositivo responda. É necessário que todos os dispositivos integrem o DS301.

A especificação CANopen relaciona estes conceitos com o *ISO/OSI Reference Model* definindo o *CANopen Reference Model* mostrado na figura A.20.

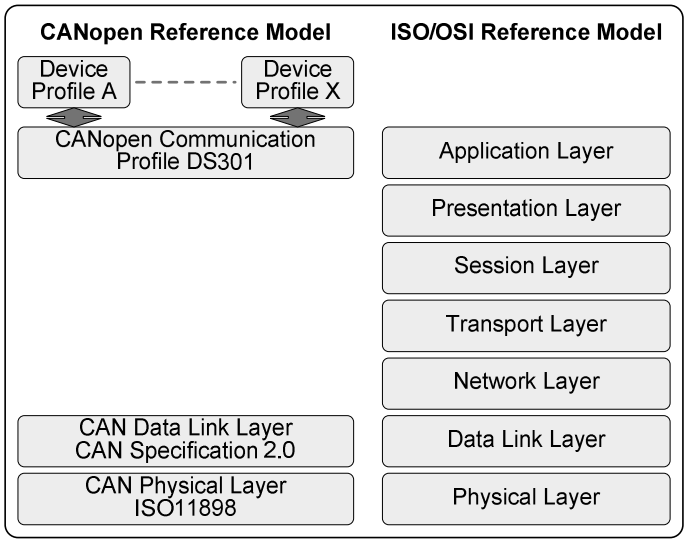

Fig. A.20. O *CANopen Reference Model* [8].

O *CANopen Reference Model* é uma versão estendida do *CAN Reference Model*. De facto, o perfil CANopen permite o desenvolvimento de aplicações que usem os serviços da CAL para comunicar. Além disso, os dispositivos CANopen são obrigados a usar a CAL numa forma compatível, de modo a que o integrador do sistema tenha que escolher apenas os dispositivos certos e configurar algumas tarefas básicas, na construção da rede distribuída. Esta é uma vantagem importante do CANopen quando

comparado com o CAL, onde o integrador do sistema tinha que gastar muito tempo a configurar a aplicação em termos dos conceitos da CAL.

Resumindo, a CAL fornece uma grande quantidade de ferramentas de comunicação mas deixa a configuração a cargo do integrador para que este decida como as usar para implementar cada um dos sistemas distribuídos. O CANopen complementa a CAL definindo uma forma *standard* de usar os serviços CAL para implementar as aplicações distribuídas, libertando o integrador dessas tarefas.

Na figura A.20 pode ser visto que o CANopen ultrapassa mesmo os limites definidos pelo *ISO/OSI Reference Model* entrando no domínio da aplicação do utilizador. De facto, o *CANopen Device Profile* pode ser visto como sendo uma camada adicional no topo do modelo de 7 camadas ISO/OSI. Alguns autores chamam esta camada de *User Layer* ou *Layer 8* [8].

O perfil CANopen define requisitos obrigatórios e requisitos opcionais. Os requisitos obrigatórios são essenciais para a compatibilidade e interoperacionalidade do CANopen e devem ser implementados em todos os dispositivos, desde o mais simples até o mais complexo. Os requisitos opcionais não têm de ser implementados nos dispositivos. Não são críticos para o funcionamento do sistema mas, se forem implementados, têm de ser segundo as especificações. Adicionalmente, um dispositivo tem de conter dados relativos ao produtor, que não são abrangidos pelos requisitos mencionados anteriormente.

A.5.2. A Funcionalidade da CANopen Application Layer

O perfil de comunicações CANopen divide a funcionalidade da *Application Layer* em *Service Objects*, cada um oferecendo uma funcionalidade. Quando uma aplicação invoca um serviço para comunicar com um dispositivo remoto, o Service *Object* que oferece o serviço referido vai interagir com um *Service Object* semelhante no nó remoto para levar a cabo a operação de troca de informação.

De acordo com o *ISO/OSI Reference Model*, uma aplicação interage com a *Application Layer* para invocar serviços, usando quatro tipos de serviços primitivos:

- *Request* – emitido pela aplicação para a *Application Layer* para requisitar um serviço;
- *Indication* – emitido pela *Application Layer* para a aplicação para informar que aconteceu um evento na rede ou que foi requisitado um serviço pela aplicação de outro nó;
- *Response* – emitido pela aplicação para a *Application Layer* como resposta a uma *Indication*;
- *Confirmation* – emitido pela *Application Layer* para a aplicação para reportar o resultado de uma requisição realizada anteriormente.

Segundo os serviços primitivos que são usados para invocar os serviços da *Application Layer*, estes podem ser classificados como:

- *Local service* – apenas é usado um *Request*, no qual a aplicação desencadeia um evento local;
- *Provider-initiated service* – apenas é usada uma *Indication*, assinalando a ocorrência de um evento local à aplicação;
- *Unconfirmed service* – uma aplicação produz um *Request*, o qual requer que seja dada uma *Indication* a uma aplicação de outro nó;
- *Confirmed service* – quando uma aplicação emite um *Request*, a aplicação remota recebe uma *Indication* e emite uma *Response*. Essa resposta é assinalada à aplicação local na forma de uma *Confirmation*.

No CAN, o protocolo no qual o CANopen está baseado, a *Data Link Layer* oferece um serviço de *broadcast* no qual as mensagens não são enviadas para uma aplicação particular num nó remoto. Cada aplicação tem de decidir se recebe ou não a informação transportada por um *Communication Object* (COB). Essa decisão é baseada no identificador da mensagem. As aplicações que recebem as mensagens não são conhecidas pelo emissor. O CANopen é semelhante ao CAN nestes aspectos.

A qualquer momento, um dispositivo CANopen pode estar envolvido em relações múltiplas de comunicação com vários nós da rede. Essas relações de comunicação podem pertencer a um de três modelos de comunicação básicos:

- Master/Slave.
- Client/Server.
- Producer/Consumer.

Os diferentes tipos de relações de comunicação serão descritos com algum detalhe de seguida.

A.5.2.1. Relações de Comunicação Master/Slave

Nas relações de comunicação *Master/Slave*, um dispositivo *Master* é responsável por coordenar um aspecto particular da operação na rede, comunicando com todos os dispositivos *Slave* na rede.

Os serviços fornecidos pela *CANopen Application Layer*, baseados em comunicações *Master/Slave* estão relacionados com mecanismos de gestão da rede. Esses serviços podem ser confirmados (*Confirmed Service*) ou não confirmados (*Unconfirmed Service*).

A.5.2.2. Relações de Comunicação Client/Server

As relações *Client/Server* envolvem unicamente dois dispositivos. É uma interacção onde um dispositivo (*Client*) lê ou escreve informação noutro dispositivo (*Server*).

Os serviços fornecidos pela *CANopen Application Layer*, baseados em comunicações *Client/Server* são tipicamente para a configuração dos dispositivos. Esses serviços são sempre confirmados através dos quais o *Client* emite uma requisição para descarregar ou enviar informação e espera uma resposta do *Server* indicando o resultado da operação.

A.5.2.3. Relações de Comunicação Producer/Consumer

Numa relação *Producer/Consumer* um dispositivo é designado de *Producer* quando possui ou produz um determinado troço de informação que pode transmitir para o barramento. Pode haver um variado número de dispositivos na rede que recebam essa informação quando é transmitida pelo *Producer*. Esses dispositivos são chamados de *Consumers*.

Os serviços fornecidos pela *CANopen Application Layer*, baseados em comunicações *Producer/Consumer* são tipicamente usados para troca de informação em tempo real. Esses serviços podem ser confirmados ou não confirmados.

A.5.3. Os Objectos CANopen e o Dicionário de Objectos

Cada dispositivo é representado como um conjunto de objectos que podem ser acedidos através da rede. Cada aspecto do funcionamento de um dispositivo é mapeado em um ou mais desses objectos. Portanto, é possível alterar a configuração ou o estado de um dispositivo simplesmente usando a rede para alterar um objecto particular. Todos os objectos que representam um dispositivo estão representados no dicionário de objectos do dispositivo correspondente

A.5.3.1. Modelos de Dispositivos CANopen

O modelo de um dispositivo CANopen é representado na figura A.21. Nesta figura podemos ver que existem dois tipos básicos de objectos:

- Objectos de comunicação – são casos da classe *Service Object* definida na *CANopen Application Layer*. Cada um representa uma funcionalidade de comunicação específica do dispositivo. Os *Communication Objects* são especificados no perfil de comunicações CANopen.

- Objectos da aplicação – representam funcionalidades específicas do dispositivo tais como o estado de um sinal de *input* digital. Os *Application Objects* são definidos no perfil do dispositivo.

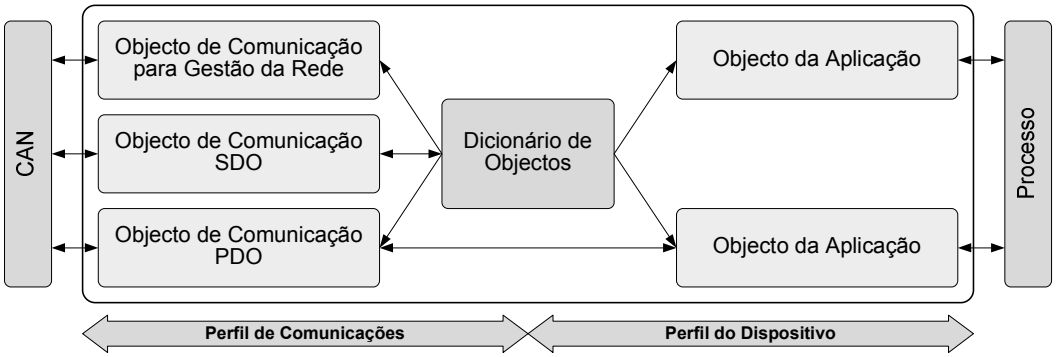

Fig. A.21. Modelo de um dispositivo CANopen.

Todos os objectos de um dispositivo podem ser acedidos através de um *Object Dictionary*, onde eles estão representados. Este princípio é mostrado na figura A.21 que mostra o OD a interagir com todos os objectos do dispositivo. O OD é a parte mais importante do modelo do dispositivo já que mapeia para a estrutura interna do dispositivo: se a estrutura do OD de um determinado dispositivo é conhecida, então tudo o que é necessário para construir uma aplicação distribuída usando esse dispositivo é conhecido. Noutras palavras, conhecendo o OD de um dispositivo é possível prever o comportamento do dispositivo na rede [8].

Os *Communication Objects* podem ser divididos em duas categorias segundo a sua funcionalidade:

- *Communication Objects for Network Management* – permite estabelecer comunicações *Master/Slave* e pode ser usado para controlo global de dispositivos e verificação do estado.
- *Communication Objects for Application Data Transfer* – estes podem ainda ser divididos em duas categorias:

    - Alguns objectos fornecem acesso indexado a todos os objectos do dispositivo através do OD, o que permite uma comunicação *Client/Server*

com outro dispositivo. Usando estes *Communication Objects* é possível aceder a todas as características do dispositivo, normalmente para propósitos de configuração. Contudo, uma vez que este tipo de operação requer trocas de informação com o OD, o seu funcionamento não é adequado a um acesso em tempo real, pois sobrecarregaria o protocolo. Um exemplo deste tipo de *Communication Object* é o *Service Data Object*.

- Outros objectos fornecem acesso directo aos *Application Objects* tornando possível implementar mecanismos de troca de informação síncrona ou assíncrona em tempo real. Este tipo de COBs segue a relação de comunicação *Producer/Consumer* envolvendo um congestionamento mínimo do protocolo. Os *Process Data Objects* são um exemplo deste tipo de mecanismo.

Como consequência da classificação anterior dos COBs, os *Application Objects* também podem ser divididos em dois grupos: (i) os que podem ser acedidos em tempo real usando COBs dedicados; e (ii) os que podem apenas ser acedidos através do OD.

A.5.3.2. Tipos de Informação CANopen e Regras de Codificação

Dependendo da propriedade que um objecto representa, este pode ser composto por um único atributo ou pode ser mais complexo possuindo muitos atributos. Os objectos com apenas um atributo são tratados como variáveis simples e são de tipos simples, tais como FLOAT, BOOLEAN, STRING ou INTEGER. Os objectos complexos são tratados como ARRAYs ou RECORDs. Uma ARRAY é usada para representar uma colecção de atributos do mesmo tipo de informação e uma RECORD representa uma colecção de atributos de vários tipos de informação.

Um exemplo de um objecto simples que possa ser representado por uma variável simples é o objecto *Device Type*, que é guardado numa palavra de 32 bits. Uma ARRAY deve ser usada para representar as saídas digitais de um módulo I/O. Se um dispositivo deste tipo tem 2 conjuntos de 8 *outputs* digitais, deve ser assinalado como um objecto simples representado por uma ARRAY com dois elementos do tipo sem sinal de 8 bits (UNSIGNED8). Os RECORDs podem ser usados, por exemplo, para

representar diferentes parâmetros associados a uma mensagem que um dispositivo transmite: o identificador (palavra de 16 bits sem sinal, UNSIGNED16), o tipo de informação trocada (palavra de 8 bits sem sinal, UNSIGNED8), o tempo mínimo entre transmissões (palavra de 16 bits sem sinal, UNSIGNED16), entre outros.

A.5.3.3. A Estrutura do Dicionário de Objectos

As entradas de um OD, isto é os objectos, são referenciadas por um simples endereço que consiste num índice (*Index*) de 16 bits sem sinal e um sub-índice (*Sub-index*) de 8 bits sem sinal.

O *Sub-index* é usado de diferentes formas de acordo com o tipo de objecto. Para um objecto simples que consiste num atributo simples o *Sub-index* é sempre 00H. Para uma ARRAY ou RECORD o *Sub-index* referencia os elementos individuais sendo o valor 00H usado para guardar o número de elementos. Para objectos STRING o *Sub-index* 00H indica o comprimento da STRING. Cada ARRAY ou RECORD apenas pode ter até 254 elementos (o Sub-index FFH é reservado) [7].

Os objectos são guardados no OD de acordo com a sua função. Além disso, determinados intervalos de *Index* são reservados para grupos de objectos com funções específicas, como mostrado na tabela A.3.

O *Index* abaixo de 1000H é usado para indicar definições do tipo de dados, que são representadas no OD apenas para referência e que são descritas na documentação do dispositivo. A tabela A.4 mostra os tipos de dados pré-definidos no CANopen e o correspondente *Index*.

O intervalo de *Index* de 1000H a 1FFFH é reservado para objectos que representam os parâmetros do *CANopen Communication Profile*. Todos os dispositivos devem ter estes parâmetros implementados da mesma forma e a configuração destes parâmetros deve também ser feita da mesma forma em todos os dispositivos.

Tabela A.3. Organização do Dicionário de Objectos [7].

Índice	Objecto
0000	Não usado
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types
00A0-0FFF	Reservado
1000-1FFF	Communication Profile Area
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardised Device Profile Area
A000-FFFF	Reservado

Tabela A.4. Representação dos tipos de dados no Dicionário de Objectos [7].

Índice	Tipo de dados	Índice	Tipo de dados
0001	BOOLEAN	0011	REAL64
0002	INTEGER8	0012	INTEGER40
0003	INTEGER16	0013	INTEGER48
0004	INTEGER32	0014	INTEGER56
0005	UNSIGNED8	0015	INTEGER64
0006	UNSIGNED16	0016	UNSIGNED24
0007	UNSIGNED32	0017	Reserved
0008	FLOAT	0018	UNSIGNED40
0009	VISIBLE STRING	0019	UNSIGNED48
000A	OCTET STRING	001A	UNSIGNED56
000B	DATE	001B	UNSIGNED64
000C	TIME OF DAY	001C-001F	Reserved
000D	TIME REFERENCE	0020	PDO COMMUNICATION PARAMETER
000E	BIT STRING	0021	PDO MAPPING PARAMETER
000F	DOMAIN	0022	SDO COMMUNICATION PARAMETER
0010	INTEGER24	0023	IDENTITY

O intervalo de *Index* de 2000H a 5FFFH é reservado para objectos que representam as características específicas do produtor do dispositivo.

O intervalo de *Index* de 6000H a 9FFFH é reservado para objectos que representam características de um dispositivo, que são standard no perfil do dispositivo correspondente.

A.5.3.4. Representação do Dicionário de Objectos

A tabela A.5 mostra uma representação normal de um OD de um dispositivo.

Tabela A.5. Exemplo de um dicionário de objectos [7].

Índice	**Objecto**	**Nome**	**Tipo**	**At**	**M/O**
0001	DEFTYPE	Boolean			
0005	DEFTYPE	Unsigned8			
0006	DEFTYPE	Unsigned16			
0007	DEFTYPE	Unsigned32			
0009	DEFTYPE	Visible string			
0020	DEFSTRUCT	PDO Communication Parameter			
0021	DEFSTRUCT	PDO Mapping			
0022	DEFSTRUCT	SDO Communication Parameter			
1000	VAR	Device type	UNSIGNED32	ro	M
1001	VAR	Error register	UNSIGNED8	ro	M
1002	VAR	Manufacturer status register	UNSIGNED32	ro	O
1003	ARRAY	Pre-defined error field	UNSIGNED8	ro	O
1008	VAR	Manufacturer device name	VISIBLE STRING	ro	O
1009	VAR	Manufacturer hardware version	VISIBLE STRING	ro	O
100A	VAR	Manufacturer software version	VISIBLE STRING	ro	O
1200	RECORD	Server SDO Comm. Parameter	SDO COM. PAR.	ro	M
1400	RECORD	1st RxPDO Comm. Parameter	PDO COM. PAR.	rw	O
1600	ARRAY	1st RxPDO Mapping Parameter	PDO MAP. PAR.	rw	O
6000	ARRAY	Read 8 input lines	UNSIGNED8	ro	O
6200	ARRAY	Write 8 output lines	UNSIGNED8	rw	O

As seis colunas desta tabela indicam as seguintes características dos objectos:

1. O Índice do objecto é representado na base hexadecimal e é utilizado como um endereço lógico, endereço de rede, possuindo uma extensão de 16 bits.
2. O Tipo de Objecto é usado com o propósito de referência e para normalização através de diferentes plataformas computacionais. Esses códigos estão listados na tabela A.6.
3. O Nome do objecto contém uma simples descrição textual da função do objecto.
4. O Tipo de entrada, que pode ser simples ou complexa, pode também ser pré--definida ou específica do fabricante.
5. O Atributo de Acesso ao objecto, que define os direitos de acesso, pode assumir o atributo: **ro** (*read only*), **wo** (*write only*), **rw** (*read/write*) ou **const**. (apenas leitura).
6. O atributo M/O de um objecto indica se a implementação desse objecto é obrigatória ou opcional e pode assumir o atributo **M** (objecto obrigatório) ou o atributo **O** (objecto opcional para esse perfil).

Tabela A.6. Códigos dos objectos [7].

Nome do obj.	Descrição	Código
NULL	Entrada do dicionário vazia	0
DOMAIN	Grande quantidade de variáveis (ex. código de programa executável)	2
DEFTYPE	Definição do tipo Simple (ex. UNSIGNED8, FLOAT…)	5
DEFSTRUCT	Definição do tipo Record	6
VAR	Variável simples	7
ARRAY	Objecto múltiplo que consiste num conjunto de VARs do mesmo tipo	8
RECORD	Objecto múltiplo que consiste num conjunto de VARs de vários tipos	9

Quando é necessária uma descrição mais detalhada do OD, cada entrada de objecto é descrita separadamente e, por cada objecto complexo, o significado de cada *Sub-index* é também mostrado.

A.5.3.5. Implementação do Dicionário de Objectos

Um dispositivo CANopen apenas tem de ter capacidade de mapear o seu estado interno em termos das entradas do OD correctas. Na prática, isso significa que quando um nó remoto lê uma entrada do OD, o dispositivo tem de ser capaz de reconhecer que parte da informação interna representa essa entrada, codifica essa informação de acordo com as especificações CANopen e emite uma resposta apropriada. Igualmente, quando é escrita uma entrada do OD por um nó remoto, o dispositivo simplesmente tem de ter capacidade de reconhecer que parte da informação local vai ser influenciada por esta operação e altera o seu estado de acordo com esta informação.

Dependendo do dispositivo, normalmente o número de entradas do OD que é necessário implementar actualmente é baixo. Não é necessário implementar os objectos abaixo de 1000H e apenas são listados no OD por razões de perfeição.

Os objectos não especificados no perfil CANopen normalmente são implementados no intervalo de *Index* de 2000H e 5FFFH. Não foram feitas grandes restrições mas é necessário que essas entradas sejam minuciosamente descritas na documentação do dispositivo.

A.5.3.6. Objectos de Comunicação CANopen para transferência de Application Data

Os dados da aplicação são transferidos usando dois tipos de *Communication Objects*: os *Service Data Objects* (SDO) e os Process *Data Objects* (PDO). Ambos podem ser usados para transferir dados entre os dispositivos mas têm diferentes tipos de funcionalidade, como mostrado na tabela A.7.

Da tabela A.7 vê-se que os SDO e os PDO são usados para propósitos diferentes dentro da rede CANopen.

Adicionalmente aos SDO e aos PDO, o CANopen define *Communication Objects* úteis para a implementação de técnicas específicas de troca de dados. Esses objectos são:

- *Synchronisation Object* – pode ser usado para sincronizar o funcionamento de dispositivos CANopen.
- *Time Stamp Object* – pode ser usado para estabelecer uma conformação global do tempo através da rede CANopen.
- *Emergency Object* – pode ser usado para assinalar situações de emergência na rede, o que é essencial para a implementação de mecanismos de recuperação de erros.

Tabela A.7. Comparação entre comunicações SDO e PDO.

Service Data Objects	**Process Data Objects**
Grande volume de transferência de dados	Pequeno volume de transferência de dados
Baixa prioridade das transferências	Alta prioridade das transferências
Utiliza índices de 16 bits e sub-índices de 8 bits para o endereçamento dos objectos	Os objectos são mapeados em bytes individuais da mensagem CAN
Transferências assíncronas	Transferências assíncronas ou síncronas
As transferências de mais do que 8 bytes são possíveis recorrendo a múltiplos telegramas	Apenas transferências de dados até 8 bytes usando um único telegrama.

A.5.4. Service Data Objects e sua Configuração

Os *Service Data Objects* são usados para estabelecer relações do tipo *Client/Server* entre dois dispositivos onde o *Client* tem acesso para ler ou escrever no OD do *Server*.

Um SDO implementado num dispositivo CANopen é chamado de *Server SDO* ou *Client SDO* dependendo da parte corrida pelo dispositivo na relação SDO estabelecida. Um dispositivo tem de ter implementado pelo menos um *Server SDO* através do qual fornece acesso ao seu OD. Adicionalmente, é possível implementar mais *Server SDOs* e qualquer número de *Client SDOs*, em ambos os casos até o máximo de 128 SDOs.

Os SDOs transportam informação do *Index* e do *Sub-index*, para permitir o endereçamento do objecto no OD.

Dois identificadores de mensagem CAN devem ser localizados num SDO. Um deles é usado para mensagens enviadas pelo *Client* para o *Server* e o outro para mensagens enviadas na direcção contrária.

Os dois serviços usados para transferências SDO são chamados de *SDO Download* e *SDO Upload* e podem ser usados para escrever e ler do OD respectivamente. Contudo, dado que a informação transferida para/de uma entrada de um OD é de um comprimento arbitrário, isto é pode variar de quantidade de bytes dependendo do objecto acedido, as transferências SDO podem seguir um de três procedimentos:

- *Segmented Transfer* – procedimento geral que pode ser usado em todas as transferências.
- *Expedited Transfer* – procedimento optimizado para pequenos troços de informação.
- *Block Transfer* – procedimento optimizado para grandes troços de informação.

O procedimento particular a ser usado em cada transferência vai depender da implementação. Contudo, a especificação CANopen declara que as *Expedited Transfers* são de implementação obrigatória e devem ser usadas para dados não maiores que 4 bytes de comprimento. As *Segmented Transfers* devem ser implementadas se o dispositivo suporta objectos que requerem transferências de dados maiores que 4 bytes e as *Block Transfers* são opcionais.

É sempre o *Client* quem inicia a transferência SDO mas tanto o *Client* como o *Server* podem abortar a transferência. Nos protocolos de transferência SDO, os 8 bytes no campo de dados CAN são sempre transmitidos, mesmo que não contenham informação importante.

Todos os SDOs que um dispositivo implementa estão representados no seu OD por uma entrada do tipo *SDO Communication Parameter*. Acedendo a essa entrada no OD de um dispositivo, os SDOs podem ser configurados. Os *Indexes* actuais da entrada *SDO Communication Parameter* estão definidos no *CANopen Communication Profile*:

- Para SDOs onde o dispositivo é um servidor, o intervalo de *Index* válido é de 1200H a 127FH. A implementação de um *Server SDO* é obrigatória, juntamente com a sua representação no OD no *Index* 1200H. Os parâmetros desse SDO não podem ser configuráveis pois esse é a ligação por defeito para comunicação com um dispositivo CANopen.
- Para SDOs onde o dispositivo é um *Client*, o intervalo de *Index* é de 1280H a 13FFH.

A estrutura de cada entrada do *SDO Communication Parameter* é mostrada na tabela A.8. Acedendo os *Sub-indexes* 1H e 2H podem ser modificados os identificadores da mensagem usados em comunicações SDO. Adicionalmente, o bit mais significativo desses campos permite habilitar ou desabilitar a transmissão/recepção dessas mensagens. Um SDO é só habilitado se ambas as mensagens estão habilitadas. A estrutura actual destas sub-entradas é mostrada na tabela A.9. O *Sub-index* 3H do *SDO Communication Parameter* guarda o *Node ID* do dispositivo par na relação *Client/Server*.

Tabela A.8. Estrutura de uma entrada do tipo *SDO Communication Parameter*.

Sub-índice	Campo do registo	Tipo de dados
0H	Número de sub-entradas suportadas	UNSIGNED8
1H	Identificador CAN para a mensagem Client-to-Server	UNSIGNED32
2H	Identificador CAN para a mensagem Server-to-Client	UNSIGNED32
3H	ID do nó do dispositivo par na comunicação	UNSIGNED8

A.5.5. Process Data Objects e sua Configuração

Os *Process Data Objects* fornecem acesso directo aos *Application Objects* dentro de um dispositivo. São usados para realizar transferências em tempo real de pequenos blocos de informação de elevada prioridade. A informação transferida tem de ser menor ou igual a 8 bytes de comprimento. Todos os bytes do campo de dados da mensagem são usados para transportar informação. O transmissor e o receptor conhecem o significado da informação transportada pelo PDO. O significado de cada byte do telegrama é

pré-definido através de informações de mapeamento do PDO, que é guardado no OD dos dispositivos que produzem e consomem o PDO. Este mapeamento, que em alguns dispositivos pode ser dinamicamente modificado utilizando SDOs, também determina o tipo de dados da informação e a forma como esta é codificada no telegrama CAN de acordo com as regras de codificação CANopen.

Tabela A.9. Descrição da entrada do identificador da mensagem SDO.

Número do bit	**Valor**	**Significado**
31	0	SDO válido
	1	SDO não válido
30	0	Reservado (sempre 0)
29	0	Identificador da mensagem de 11 bits
	1	Identificador da mensagem de 29 bits
28 a 11	0	Se o bit 29 é 0: 0
	X	Se o bit 29 é 1: valor dos bits de 28 a 11 do identificador de 29 bits
10 a 0	X	Valor dos bits de 10 a 0 do identificador de 11 bits

O funcionamento dos PDOs é baseado em relações *Producer/Consumer* onde ambos dos modelos *Push* e *Pull* podem ser usados. Um PDO para o qual um dispositivo é o *Producer* é chamado de *Transmit PDO* desse dispositivo. Um PDO para o qual um dispositivo é o *Consumer* é chamado de *Receive PDO* desse dispositivo.

Cada PDO requer a atribuição de um identificador de mensagem para o seu funcionamento. O CANopen define identificadores de mensagem CAN por defeito que podem ser usados por dispositivos para implementar oito PDOs: em quatro desses PDOs, o dispositivo será o *Producer* e nos outros quatro o dispositivo será o *Consumer*.

São especificados dois serviços para comunicações PDO: *Write PDO* e *Read PDO*.

O serviço *Write PDO* segue o modelo *Push*. Há um *Producer* e qualquer número de *Consumers* do PDO. Usando o serviço *Write PDO* o *Producer* pode enviar, por iniciativa própria, os dados da aplicação associados com o PDO para os *Consumers*.

O protocolo para o serviço *Write PDO* é mostrado na figura A.22. O serviço é não confirmado e a mensagem CAN usada na transferência carrega apenas dados do processo. O número de bytes transportados pelo PDO é determinado através da aplicação das regras de codificação de dados aos *Application Objects* mapeados no PDO.

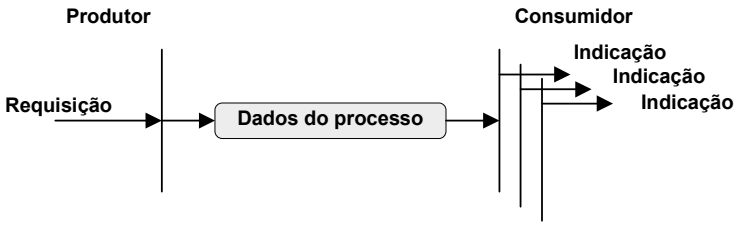

Fig. A.22. Esquematização do *Write PDO* [8].

O serviço *Read PDO* segue o modelo *Pull*. Aqui há o *Producer* e pelo menos um *Consumer* do PDO. Usando o serviço *Read PDO* um dos *Consumers* pode requisitar do *Producer* a informação da aplicação associada com o PDO.

O protocolo para o serviço *Read PDO* é mostrado na figura A.23.

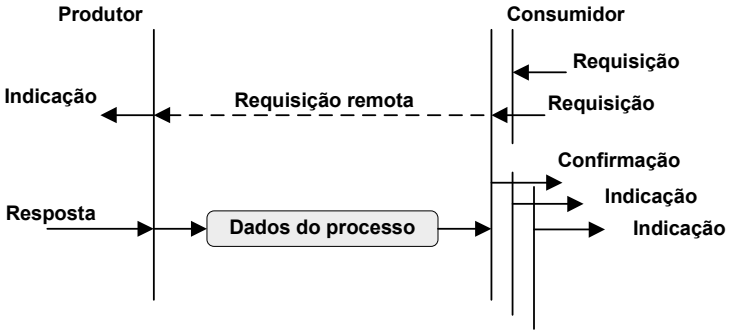

Fig. A.23. Esquematização do *Read PDO* [8].

Como pode ser visto na figura A.23 o serviço é confirmado. O protocolo para o *Read PDO* é uma versão estendida do protocolo do *Write PDO*. A diferença reside no facto de que o desencadeamento da transmissão da mensagem é realizado por um dos *Consumers*. Para desencadear a transmissão do PDO, o *Consumer* coloca no barramento uma requisição remota com o identificador atribuído ao PDO.

A transmissão de um PDO pode ser desencadeada de três modos:

- *Event driven* – a transmissão PDO é desencadeada pela ocorrência de um evento no dispositivo. Este evento pode estar relacionado com um *Communication Object* ou com um *Application Object* dentro do dispositivo. Neste modo é usado o serviço *Write PDO*.
- *Timer driven* – um PDO também pode ser configurado de modo a que a sua transmissão seja desencadeada quando tenha passado um determinado tempo sem a ocorrência do evento associado com esse PDO. Neste modo é usado o serviço *Write PDO*.
- *Remotely requested* – a transmissão do PDO é solicitada por outro dispositivo usando uma requisição remota, de acordo com o serviço *Read PDO*. Os PDOs cuja transmissão seja desencadeada usando requisições remotas são ideais para a implementação de mecanismos de exploração.

Normalmente, um PDO é configurado para operar no modo *Event driven*, mas é possível configurar um PDO para funcionar apenas no modo *Remotely requested*. Contudo, os três modos podem ser usados em conjunto.

O tipo de transmissão de um PDO é determinado pela natureza do evento que desencadeia a sua transmissão. Os PDOs são divididos nas seguintes categorias, de acordo com o seu tipo de transmissão:

- *Synchronous transmition* – a transmissão de PDOs síncronos está dependente da recepção de uma mensagem de sincronização periódica, o *Synchronisation Object*, transmitida por outro dispositivo. Isto torna possível implementar mecanismos de aquisição de dados coordenados através da rede CANopen. No entanto, a informação transportada pelos PDOs síncronos é apenas usada pelos seus *Consumers* na recepção do próximo *Synchronisation Object*. Isto torna possível a implementação de mecanismos de actuação coordenados através da rede.

- *Asynchronous transmition* – a transmissão do PDO é desencadeada por um evento não relacionado com o *Synchronisation Object*. O evento que desencadeia a transmissão pode ter origem num *Application Object* ou num *Communication Object* dentro do dispositivo. O PDO com origem num *Application Object* funciona no modo *Event driven* e o evento associado a este pode ser por exemplo a troca de estado num dos sinais de entrada do dispositivo.

Os PDOs síncronos também podem ser divididos em cíclicos e acíclicos. Os cíclicos são transmitidos depois de um certo número de *Synchronisation Objects* recebidos. Os acíclicos são transmitidos depois da recepção de um *Synchronisation Object*, mas apenas quando um determinado evento interno ao dispositivo tenha ocorrido.

Normalmente, em aplicações de automação, a natureza da informação do processo transportada no barramento pertence a uma de duas categorias:

- *Command messages* – tipicamente emitidas por um dispositivo de controlo para a actuação remota de dispositivos. Através destas mensagens, o controlador induz o seu controlo sobre os dispositivos de actuação de modo a que eles exerçam uma determinada acção. Usando comunicações PDO síncronas, a execução desses comandos pode ser realizada simultaneamente por múltiplos dispositivos.
- *Feedback messages* – normalmente emitidas por dispositivos remotos de aquisição de dados para um controlador, transportando dados do processo que adquiriram. Usando comunicações PDO síncronas, os dados do processo podem ser recolhidos em instantes bem definidos, o que pode ser muito útil em aplicações onde a constante de tempo é importante.

O diagrama da figura A.24 representa a estrutura típica numa comunicação usando PDOs síncronos, numa rede CANopen.

Em intervalos de tempo pré-definidos é transmitido o *Synchronisation Object* (SYNC) por um dos dispositivos da rede, chamado de *Synchronisation Object Producer*. O intervalo de tempo que separa a transmissão de dois SYNCs é chamado de

Communication Cycle Period. O *Synchronisation Object Producer* não tem que ser o mesmo dispositivo que actua como *Master* na rede mas pode ser outro dispositivo.

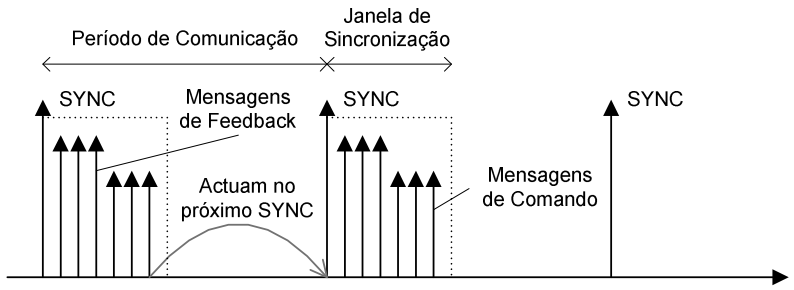

Fig. A.24. Estrutura típica de comunicações usando PDOs síncronos [8].

Quando é recebido um SYNC, todos os dispositivos que produzem PDOs síncronos colocam as suas mensagens no barramento. Algumas das mensagens serão de comando e as restantes serão de *feedback*. A figura A.24 assume que as mensagens de *feedback* têm identificadores com maior prioridade. Todos os PDOs síncronos são transmitidos dentro do intervalo de tempo que começa com a recepção do SYNC. As acções que são desencadeadas através das mensagens de comando apenas são levadas a cabo depois da recepção do próximo SYNC.

Como foi previamente mencionado, a transmissão de um PDO assíncrono pode estar associado com a ocorrência de um evento da aplicação detectado no dispositivo. Neste caso, pode ser transmitido a qualquer momento durante o funcionamento da aplicação. Permitindo a transmissão da informação o mais cedo possível, assim que o evento em questão ocorre.

A configuração de um PDO consiste em dois passos diferentes:

- Mapeamento dos dados da mensagem PDO, isto é definição de que dados serão transferidos em cada campo da mensagem CAN. Isto permite que o *Producer* e os *Consumers* conheçam que parte da informação está a ser transmitida sem introduzir informação de controlo adicional na mensagem. Assim maximiza a eficiência da transmissão.

- Definição dos parâmetros de comunicação do PDO, isto é o identificador da mensagem que usa, o tipo de transmissão, etc.

Deve-se também ter em mente, quando se realiza a configuração do PDO, aspectos globais da rede, tais como entre que dispositivos deve haver transferência do PDO, prioridades relativas dadas aos diferentes PDOs e a carga do barramento.

Qualquer PDO que um dispositivo implementa tem de ser representado no seu OD por uma entrada do tipo *PDO Communication Parameter*. Os *Indexes* actuais das entradas são definidos no *CANopen Communication Profile*:

- O intervalo válido para o *Index* dos PDOs de recepção é de 1400H a 15FFH.
- O intervalo válido para o *Index* dos PDOs de transmissão é de 1800H a 19FFH.

Acedendo a essas entradas, podem ser configurados os parâmetros de comunicação dos PDOs. A estrutura de cada *PDO Communication Parameter* é mostrada na tabela A.10.

Tabela A.10. Estrutura de uma entrada do tipo *PDO Communication Parameter* [8].

Sub-index	**Campo do registo**	**Tipo de dados**
00H	Número de sub-entradas suportadas	UNSIGNED8
01H	Identificador da mensagem CAN que o PDO usa	UNSIGNED32
02H	Tipo de transmissão	UNSIGNED8
03H	Inhibit Time (Tempo de espera)	UNSIGNED16
04H	Reservado	UNSIGNED8
05H	Event Timer (Temporizador do evento)	UNSIGNED16

Como no caso das entradas SDO, o identificador da mensagem CAN, no *Sub-index* 1H, permite também a configuração de outros aspectos do funcionamento do PDO, como mostrado na tabela A.11. Para habilitar ou desabilitar um PDO é suficiente usar um SDO para aceder a esta sub-entrada e definir o bit mais significativo para 0 ou 1 conforme o desejado.

O parâmetro do tipo de transmissão localizado no *Sub-index* 2H da tabela A.11 define as características de transmissão do PDO. A utilização desta sub-entrada é mostrada na tabela A.12.

Tabela A.11. Descrição da entrada do identificador da mensagem PDO [8].

Número do bit	Valor	Significado
31	0	PDO válido
	1	PDO não válido
30	0	É permitido Remote Request neste PDO
	1	Não é permitido Remote Request neste PDO
29	0	Identificador da mensagem de 11 bits
	1	Identificador da mensagem de 29 bits
28 a 11	0	Se o bit 29 é 0: 0
	X	Se o bit 29 é 1: valor dos bits de 28 a 11 do identificador de 29 bits
10 a 0	X	Valor dos bits de 10 a 0 do identificador de 11 bits

Tabela A.12. Tipos de transmissão do PDO.

Tipo de transmissão	Característica da transmissão				
	Cíclica	**Acíclica**	**Síncrona**	**Assíncrona**	**Remote Request**
0		✓	✓		
1 a 240	✓		✓		
241 a 251	Reservado				
252			✓		✓
253				✓	✓
254				✓	
255				✓	

Relativamente à tabela A.12, um PDO síncrono acíclico é um evento desencadeado depois da recepção do primeiro telegrama SYNC, depois da ocorrência de um evento interno da aplicação com o qual o PDO está associado.

Escrevendo um valor 0 no *Sub-index* 02H do *PDO Communication Parameter* configurará o PDO para ser deste tipo. O evento da aplicação associado com o PDO deve ser definido na documentação do dispositivo ou no perfil do dispositivo.

Escrevendo o valor 1 a 240 no mesmo *Sub-index* configurará o PDO para ser síncrono e cíclico. Isto significa que o PDO será transmitido na recepção de todos os n telegramas SYNC, onde n é o número introduzido no *Sub-index*.

Um valor do tipo de transmissão de 252 significa que a transmissão do PDO acontecerá quando for recebida uma requisição remota mas os dados enviados corresponderão aos do instante de tempo no qual foi recebido o último telegrama SYNC.

Um valor de 253 indica que o PDO será transmitido na recepção de uma requisição remota e os dados serão actualizados na recepção do *remote request*.

Um valor de 254 indica que o PDO é puramente assíncrono e o evento da aplicação que desencadeia a transmissão é específico do produtor.

Um valor de 255 indica que o PDO é puramente assíncrono e o evento da aplicação que desencadeia a transmissão é definido no *Device Profile*.

Os *Sub-indexes* 03H e 05H do *PDO Communication Parameter* contêm dois parâmetros adicionais que podem ser úteis para transmitir PDOs dos tipos 254 e 255: o *Inhibit Time* e o *Event Timer* (estes parâmetros não têm significado nos PDOs de recepção).

O *Inhibit Time* é usado para limitar a taxa máxima de transmissão de um PDO e pode ser útil para prevenir que PDOs de alta prioridade encham o barramento, impossibilitando a transmissão de mensagens de baixa prioridade. O PDO não pode ser transmitido antes do tempo definido neste parâmetro ter passado.

O *Event Timer* é usado para assegurar a taxa mínima de transmissões de PDOs, de forma independente da ocorrência de eventos da aplicação associados com o PDO. Se o tempo definido neste parâmetro termina e não é detectada a ocorrência do evento associado com o PDO, este é transmitido.

Os valores para os parâmetros *Inhibit Time* e *Event Timer* são definidos em múltiplos de 100 microssegundos e 1 milissegundo respectivamente. Em ambos os casos um valor nulo indica que o mecanismo está desabilitado.

Finalmente, é importante notar que a implementação do *Inhibit Time* e do *Event Timer* é opcional.

O mapeamento dos dados do processo para cada PDO deve ser representado no OD por uma entrada do tipo *PDO Mapping Parameter*. O *Index* dessa entrada é encontrado adicionando 200H ao *Index* do *PDO Communication Parameter* correspondente. De facto, os intervalos de *Indexes* para as entradas de mapeamento são definidos no *CANopen Communication Profile* como:

- O intervalo de *Indexes* válido para o mapeamento de PDOs de recepção é de 1600H a 17FFH.
- O intervalo de *Indexes* válido para o mapeamento de PDOs de transmissão é de 1A00H a 1BFFH.

O mapeamento do PDO descreve o seu conteúdo listando a sequência e o comprimento das entradas do OD mapeadas que são transportadas pelo PDO. A estrutura de cada entrada *PDO Mapping Parameter* é mostrada na tabela A.13.

Tabela A.13. Estrutura de uma entrada do tipo *PDO Mapping Parameter* [8].

Sub-index	**Campo do registo**	**Tipo de dados**
00H	Número de objectos mapeados no PDO	UNSIGNED8
01H	1º objecto mapeado	UNSIGNED32
02H	2º objecto mapeado	UNSIGNED32
::::	::::::::::::::::::::::::	::::::::::::::
40H	64º objecto mapeado	UNSIGNED32

Para descrever o conteúdo de um PDO é suficiente indicar quantas entradas do OD serão transportadas pelo PDO, quais as entradas mapeadas (*Index* e *Sub-index*), em que ordem e qual é o seu tamanho individual. A descrição é guardada nos valores das

entradas do *Sub-index* representadas na tabela A.13. A estrutura das entradas dos *Sub-indexes* de 01H a 40H é mostrada na figura A.25.

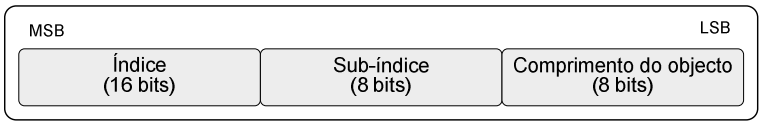

Fig. A.25. Estrutura de uma entrada de mapeamento de um objecto [8].

O comprimento do objecto pode ser um valor dentro do intervalo de 1 a 64, indicando o número de bits ocupados por um objecto particular.

É importante notar que o *Device Profile* define mapeamentos standard por defeito que todos os dispositivos de um tipo particular devem usar. Se estes mapeamentos por defeito são adequados para uma aplicação particular, a quantidade de configuração necessária é reduzida a definir os parâmetros de comunicação do PDO.

Concluindo, o mapeamento de um PDO é simplesmente a forma de informar um dispositivo do que está a receber ou transmitir num PDO. Olhando para a informação do mapeamento, o dispositivo é capaz de conseguir a informação correcta do OD e colocá--la no PDO ou conseguir a informação do PDO e usá-la para actualizar as entradas correctas do OD.

A.5.6. Os Electronic Data Sheets

Os ficheiros da EDS têm codificação ASCII e o comprimento total das linhas não deve exceder os 255 caracteres [21].

A EDS tem inúmeras secções que devem ser listadas no formato seguinte:

```
[nome da secção]
entrada=valor
```

A primeira linha indica o nome da secção e os parênteses rectos são obrigatórios. O parêntese da esquerda deve estar na coluna mais à esquerda [21].

A segunda linha indica o valor de cada entrada. Do lado esquerdo é introduzido o nome da entrada. O nome consiste numa combinação de caracteres e deve ser imediatamente seguido por um sinal de igual. Do lado direito é introduzido o valor.

A interpretação do parâmetro "valor" depende da entrada em questão. Há algumas regras gerais:

- Os espaços em branco são cortados;
- Os números inteiros são escritos como decimais, hexadecimais ou octais. Os números hexadecimais são precedidos de 0x e os octais são precedidos de 0.

Há valores que dependem do COB-ID, como no caso dos PDOs. A sintaxe da fórmula é como a mostrada de seguida:

entrada=$NODEID{"+"número}

A.5.6.1. Informação do Ficheiro

A EDS contém informação sobre ela mesma. Isto é útil para o controlo da versão. Toda a informação é guardada na secção [FileInfo] que contém as entradas mostradas na tabela A.14.

De seguida é apresentada a secção [FileInfo] da EDS de um dos modelos de dispositivos (accionador de estores):

```
[FileInfo]
FileName=AE.EDS
FileVersion=0
FileRevision=0
Description=EDS-File file for testing a light sensor
```

CreationTime=15:36AM

CreationDate=15-07-09

CreatedBy=BSA

ModificationTime=15:37AM

ModificationDate=15-07-09

ModifiedBy=BSA

Tabela A.14. Entradas da secção FileInfo [21].

Entrada	**Descrição**
FileName	indica o nome do ficheiro
FileVersion	indica a versão do ficheiro
File Revision	indica a revisão do ficheiro
EDSVersion	indica a versão da especificação no formato "x.y". Se falta esta entrada, equivale a "3.0".
Description	fornece uma descrição do ficheiro
CreationTime	indica a hora de criação do ficheiro no formato "hh:mm(AM|PM)"
CreationDate	indica a data de criação do ficheiro no formato "mm-dd-yyyy"
CreatedBy	indica o nome do autor (no máximo 245 caracteres)
ModificationTime	indica a hora da última modificação do ficheiro no formato "hh:mm(AM|PM)"
ModificationDate	indica a data da última modificação do ficheiro no formato "mm-dd-yyyy"
ModifiedBy	indica o nome do autor da modificação

A.5.6.2. Informação Geral do Dispositivo

A EDS contém também informação específica do dispositivo. Essa informação deve ser dada na secção [DeviceInfo] que contém as entradas mostradas na tabela A.15.

Tabela A.15. Entradas da secção DeviceInfo [21].

Entrada	**Descrição**
VendorName	indica o nome do vendedor (244 caracteres no máximo)
VendorNumber	indica o ID único do vendedor
ProductName	indica o nome do produto (243 caracteres no máximo)

ProductNumber	indica o número do produto
RevisionNumber	indica o número da revisão do produto
OrderCode	indica o código do produto
BaudRate_10	indica os baudrates suportados (0= não suportado e 1= suportado)
BaudRate_20	indica os baudrates suportados
BaudRate_50	indica os baudrates suportados
BaudRate_125	indica os baudrates suportados
BaudRate_250	indica os baudrates suportados
BaudRate_500	indica os baudrates suportados
BaudRate_800	indica os baudrates suportados
BaudRate_1000	indica os baudrates suportados
SimpleBootUpMaster	indica a funcionalidade de arranque como master (0= não suportado e 1= suportado)
SimpleBootUpSlave	indica a funcionalidade de arranque como slave (0= não suportado e 1= suportado)
Granularity	indica a granularidade permitida para o mapeamento do dispositivo (a maior parte dos dispositivos suporta uma granularidade de 8)
DynamicChannelsSupported	indica a facilidade de geração dinâmica. Se o valor é diferente de 0, existe uma secção adicional designada de DynamicChannels
GroupMessaging	indica a facilidade de PDOs multiplexados (0= não suportado e 1= suportado)
NrOfRXPDO	indica o número de PDOs de recepção suportados
NrOfTXPDO	indica o número de PDOs de transmissão suportados
LSS_Supported	indica se a funcionalidade LSS é suportada (0= não suportado e 1= suportado)

A.5.6.3. Dicionário de Objectos

Nesta parte lógica da EDS deve ser fornecida informação relativa (i) aos objectos do OD que são suportados, (ii) aos valores limites dos parâmetros, (iii) aos valores por defeito e (iv) aos tipos de dados.

A descrição dos objectos deve ser estruturalmente dividida em três partes separadas correspondentes a:

- Objectos obrigatórios [MandatoryObjects]: contém apenas os objectos obrigatórios. Estes são pelo menos os objectos 1000h e 1001h.
- Objectos opcionais [OptionalObjects]: contém todos os outros objectos da área 1000h-1FFFh e 6000h-FFFFh.
- Objectos específicos do produtor [ManufacturerObjects]: contém todos os objectos específicos do produtor localizados na área 2000h-5FFFh.

Cada uma destas secções deve conter uma lista dos objectos suportados que contém as entradas para os mesmos.

SupportedObjects = número de entradas da secção.

As entradas são decimais e começam no número 1. Desta forma, a última entrada indica o número de entradas disponíveis. Um exemplo disto pode ser o seguinte:

```
[OptionalObjects]
SupportedObjects=4
1=0x1003
2=0x1004
3=0x1005
4=0x1008
```

Cada um dos objectos listados deve ser descrito numa secção própria. Os nomes dessas secções devem ser construídos seguindo uma determinada estrutura. O nome da secção deve ser do tipo [<Index>] usando valores hexadecimais para o Índice e para o Sub--índice sem o "0x" precedente [21]. Dentro da secção devem existir as entradas indicadas na tabela A.16.

Tabela A.16. Entradas da secção de cada um dos objectos do OD [21].

Entrada	**Descrição**
SubNumber	Indica o número de sub-índices disponíveis no índice correspondente, não contando com o sub-índice FFh. Se não houver sub-índices esta entrada não existe.
ParameterName	Indica o nome do parâmetro.
ObjectType	Indica o código do objecto.
DataType	Indica o índice do tipo de dados do objecto.
LowLimit	Indica o limite mínimo do valor do objecto.
HighLimit	Indica o limite máximo do valor do objecto.
AccessType	Indica o tipo de acesso ao objecto e é representado pelas seguintes strings: "ro" – read only; "wo" – write only; "rw" – read/write; "rwr" – read/write on process input; "rww" – read/write on process output; "const" – constant value.
DefaultValue	Indica o valor do objecto por defeito.
PDOMapping	Indica se é possível o mapeamento deste objecto através de PDO (0= não mapeável; 1= mapeável).
ObjFlags	É uma entrada opcional.

ANEXO B – OS DICIONÁRIOS DE OBJECTOS

B.1. DICIONÁRIO DO ACCIONADOR DE ESTORES

Tabela B.1. Área do dicionário de objectos para o perfil de comunicações standard do accionador de estores.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE COMUNICAÇÕES STANDARD DSP301							
1000	.	Device type	U32	ro	0x00030000	N	DSP 0x000 Digital outputs and inputs=0x0003
1001	.	Error register	U8	ro	0	S	Erros indicados em DSP301
1002	.	Manufacturer status register	U32	ro	0	N	
1004	.	# PDOs supported	ARRAY				
	0	# PDOs supported	U32	ro	0x00040001	N	4RPDO e 1TPDO
	1	# PDOs sync	U32	ro	0x00000000	N	0RPDO e 0TPDO
	2	# PDOs async	U32	ro	0x00040001	N	4RPDO e 1TPDO
1005	.	COB-ID SYNC message	U32	rw	0x00000080	N	SYNC não usada
1006	.	Communication cycle period	U32	rw	0	N	Não utilizado
1007	.	Synchronous window length	U32	rw	0	N	Não utilizado
1008	.	Manufacturer device name	VisStr	c	AE	N	AE - Accionador de estores
1009	.	Man. hardware version	VisStr	c	0	N	
100A	.	Man. software version	VisStr	c	0	N	
100F	.	# SDOs supported	U32	ro	0x00000001	N	0 SDO client e 1 SDO server
1010	.	Store parameters	ARRAY				Guarda os parâmetros em memória não volátil
	0	Largest supported sub index	U8	ro	4	N	
	1	Save all parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	2	Save comm. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	3	Save app. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	4	Save man. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
1011	.	Restore default parameters	ARRAY				
	0	Largest supported sub index	U32	ro	4	N	
	1	Restore all parameters	U32	rw	0x00000001	N	Restaura parâmetros
	2	Restore comm. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	3	Restore app. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	4	Restore man. parameters	U32	rw	0x00000001	N	Restaura parâmetros
1400	.	1 RPDO parameter	RECORD				RPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1401	.	2 RPDO parameter	RECORD				RPDO2 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1402	.	3 RPDO parameter	RECORD				RPDO3 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1403	.	4 RPDO parameter	RECORD				RPDO4 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1800	.	1 TPDO parameter	RECORD				TPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	180h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono

Tabela B.2. Área do dicionário de objectos para o perfil específico do fabricante do accionador de estores.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE DISPOSITIVO STANDARD							
ÁREA DO PERFIL ESPECÍFICO DO FABRICANTE							
200D	.	Modo de funcionamento (E_MOD)	U8	rw	0x00	N	Normal=0x00 Despertar=0x01
200E	.	Estado do estore (EST)	U8	rw	0x00	S	Off=0x00 Fechar=0x01 Abrir=0x02
200F	.	Posição do estore (POE)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
2010	.	Estado definido no teclado da iluminação (E_INT_LAMP)	U8	rw	0x02	S	Off=0x00 On=0x01 Automático=0x02
2011	.	Estado da lâmpada (E_LAMP)	U8	rw	0x00	S	Off=0x00 On=0x01
2012	.	Luminosidade da lâmpada (E_LL)	U8	rw	0x00	S	Min=0%=0x00 Max=100%=0x64
2013	.	Luminosidade interior (E_SLI)	U8	rw	0x01	S	Pouca=0x00 Adequada=0x01 Muita=0x02
2014	.	Luminosidade exterior (E_SLE)	U8	rw	0x01	S	Noite=0x00 Adequada=0x01 Sol=0x02
2015	.	Posição do estore no interruptor (E_INT_POE)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
2016	.	Estado definido no teclado do estore (E_INT_EST)	U8	rw	0x02	S	Off=0x00 On=0x01 Automático=0x02

B.2. DICIONÁRIO DO ACCIONADOR DE TOLDOS

Tabela B.3. Dicionário de objectos do accionador de toldos.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE COMUNICAÇÕES STANDARD DSP301							
1000	.	Device type	U32	ro	0x00030000	N	DSP 0x000 Digital outputs and inputs=0x0003
1001	.	Error register	U8	ro	0	S	Erros indicados em DSP301
1002	.	Manufacturer status register	U32	ro	0	N	
1004	.	# PDOs supported	ARRAY				
	0	# PDOs supported	U32	ro	0x00020001	N	2RPDO e 1TPDO
	1	# PDOs sync	U32	ro	0x00000000	N	0RPDO e 0TPDO
	2	# PDOs async	U32	ro	0x00020001	N	2RPDO e 1TPDO
1005	.	COB-ID SYNC message	U32	rw	0x00000080	N	SYNC não usada
1006	.	Communication cycle period	U32	rw	0	N	Não utilizado
1007	.	Synchronous window length	U32	rw	0	N	Não utilizado
1008	.	Manufacturer device name	VisStr	c	AT	N	AT - accionador de toldos
1009	.	Man. hardware version	VisStr	c	0	N	
100A	.	Man. software version	VisStr	c	0	N	
100F	.	# SDOs supported	U32	ro	0x00000001	N	0 SDO client e 1 SDO server
1010	.	Store parameters	ARRAY				Guarda os parâmetros em memória não volátil
	0	Largest supported sub index	U8	ro	4	N	
	1	Save all parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	2	Save comm. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	3	Save app. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	4	Save man. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
1011	.	Restore default parameters	ARRAY				
	0	Largest supported sub index	U32	ro	4	N	
	1	Restore all parameters	U32	rw	0x00000001	N	Restaura parâmetros
	2	Restore comm. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	3	Restore app. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	4	Restore man. parameters	U32	rw	0x00000001	N	Restaura parâmetros
1400	.	1 RPDO parameter	RECORD				RPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1401	.	2 RPDO parameter	RECORD				RPDO2 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1800	.	1 TPDO parameter	RECORD				TPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	180h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
ÁREA DO PERFIL DE DISPOSITIVO STANDARD							
ÁREA DO PERFIL ESPECÍFICO DO FABRICANTE							
2017	.	Estado do toldo (TOL)	U8	rw	0x00	S	Off=0x00 Fechar=0x01 Abrir=0x02
2018	.	Posição do toldo (POT)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
2019	.	Luminosidade exterior (T_SLE)	U8	rw	0x01	S	Noite=0x00 Adequada=0x01 Sol=0x02
201A	.	Posição do toldo (T_INT_POT)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
201B	.	Estado definido no teclado do toldo (T_INT_TOL)	U8	rw	0x02	S	Off=0x00 On=0x01 Automático=0x02

B.3. DICIONÁRIO DA LÂMPADA DE INTENSIDADE REGULÁVEL

Tabela B.4. Área do dicionário de objectos para o perfil de comunicações standard da lâmpada de intensidade regulável.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE COMUNICAÇÕES STANDARD DSP301							
1000	.	Device type	U32	ro	0x00030000	N	DSP 0x000 Digital outputs and inputs=0x0003
1001	.	Error register	U8	ro	0	S	Erros indicados em DSP301
1002	.	Manufacturer status register	U32	ro	0	N	
1004	.	# PDOs supported	ARRAY				
	0	# PDOs supported	U32	ro	0x00040001	N	4RPDO e 1TPDO
	1	# PDOs sync	U32	ro	0x00000000	N	0RPDO e 0TPDO
	2	# PDOs async	U32	ro	0x00040001	N	4RPDO e 1TPDO
1005	.	COB-ID SYNC message	U32	rw	0x00000080	N	SYNC não usada
1006	.	Communication cycle period	U32	rw	0	N	Não utilizado
1007	.	Synchronous window length	U32	rw	0	N	Não utilizado
1008	.	Manufacturer device name	VisStr	c	LIR	N	LIR - Lâmpada de intensidade regulável
1009	.	Man. hardware version	VisStr	c	0	N	
100A	.	Man. software version	VisStr	c	0	N	
100F	.	# SDOs supported	U32	ro	0x00000001	N	0 SDO client e 1 SDO server
1010	.	Store parameters	ARRAY				Guarda os parâmetros em memória não volátil
	0	Largest supported sub index	U8	ro	4	N	
	1	Save all parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	2	Save comm. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	3	Save app. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	4	Save man. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
1011	.	Restore default parameters	ARRAY				
	0	Largest supported sub index	U32	ro	4	N	
	1	Restore all parameters	U32	rw	0x00000001	N	Restaura parâmetros
	2	Restore comm. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	3	Restore app. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	4	Restore man. parameters	U32	rw	0x00000001	N	Restaura parâmetros
1400	.	1 RPDO parameter	RECORD				RPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1401	.	2 RPDO parameter	RECORD				RPDO2 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1402	.	3 RPDO parameter	RECORD				RPDO3 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1403	.	4 RPDO parameter	RECORD				RPDO4 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1800	.	1 TPDO parameter	RECORD				TPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	180h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono

Tabela B.5. Área do dicionário de objectos para o perfil específico do fabricante da lâmpada de intensidade regulável.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE DISPOSITIVO STANDARD							
ÁREA DO PERFIL ESPECÍFICO DO FABRICANTE							
2004	.	Modo de funcionamento (L_MOD)	U8	rw	0x00	N	Normal=0x00 Dormir=0x01 Despertar=0x02
2005	.	Estado da lâmpada (LAMP)	U8	rw	0x00	S	Off=0x00 On=0x01
2006	.	Luminosidade da lâmpada (LL)	U8	rw	0x00	S	Min=0%=0x00 Max=100%=0x64
2007	.	Estado definido no teclado da iluminação (L_INT_LAMP)	U8	rw	0x02	S	Off=0x00 On=0x01 Automático=0x02
2008	.	Luminosidade definida no teclado (L_INT_LL)	U8	rw	0x00	S	Min=0%=0x00 Max=100%=0x64
2009	.	Luminosidade interior (L_SLI)	U8	rw	0x01	S	Pouca=0x00 Adequada=0x01 Muita=0x02
200A	.	Luminosidade exterior (L_SLE)	U8	rw	0x01	S	Noite=0x00 Adequada=0x01 Sol=0x02
200B	.	Posição do estore (L_POE)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
200C	.	Estado definido no teclado do estore (L_INT_EST)	U8	rw	0x02	S	Off=0x00 On=0x01 Automático=0x02

B.4. DICIONÁRIO DO SENSOR DE LUMINOSIDADE EXTERIOR

Tabela B.6. Dicionário de objectos do sensor de luminosidade exterior.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE COMUNICAÇÕES STANDARD DSP301							
1000	.	Device type	U32	ro	0x00020000	N	DSP 0x000 Digital outputs=0x0002
1001	.	Error register	U8	ro	0	S	Erros indicados em DSP301
1002	.	Manufacturer status register	U32	ro	0	N	
1004	.	# PDOs supported	ARRAY				
	0	# PDOs supported	U32	ro	0x00000001	N	0RPDO e 1TPDO
	1	# PDOs sync	U32	ro	0x00000000	N	0RPDO e 0TPDO
	2	# PDOs async	U32	ro	0x00000001	N	0RPDO e 1TPDO
1005	.	COB-ID SYNC message	U32	rw	0x00000080	N	SYNC não usada
1006	.	Communication cycle period	U32	rw	0	N	Não utilizado
1007	.	Synchronous window length	U32	rw	0	N	Não utilizado
1008	.	Manufacturer device name	VisStr	c	SLE	N	SLE - Sensor de luminosidade exterior
1009	.	Man. hardware version	VisStr	c	0	N	
100A	.	Man. software version	VisStr	c	0	N	
100F	.	# SDOs supported	U32	ro	0x00000001	N	0 SDO client e 1 SDO server
1010	.	Store parameters	ARRAY				Guarda os parâmetros em memória não volátil
	0	Largest supported sub index	U8	ro	4	N	
	1	Save all parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	2	Save comm. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	3	Save app. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	4	Save man. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
1011	.	Restore default parameters	ARRAY				
	0	Largest supported sub index	U32	ro	4	N	
	1	Restore all parameters	U32	rw	0x00000001	N	Restaura parâmetros
	2	Restore comm. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	3	Restore app. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	4	Restore man. parameters	U32	rw	0x00000001	N	Restaura parâmetros
1800	.	1 TPDO parameter	RECORD				TPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	180h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
ÁREA DO PERFIL DE DISPOSITIVO STANDARD							
ÁREA DO PERFIL ESPECÍFICO DO FABRICANTE							
2002	.	Parâmetros relativos à luminosidade exterior					
	0	Number of objects	U8	ro	3	N	Número de objectos
	1	Luminosidade mínima LEMIN	U8	rw	0x0A	N	Min=0%=0x00 Max=100%=0x64
	2	Luminosidade máxima LEMAX	U8	rw	0x5A	N	Min=0%=0x00 Max=100%=0x64
	3	Luminosidade actual LE	U8	rw	N	N	Min=0%=0x00 Max=100%=0x64
2003	.	Estado da luminosidade exterior SLE	U8	rw	N	S	Noite=0x00 Adequada=0x01 Sol=0x02

B.5. DICIONÁRIO DO SENSOR DE LUMINOSIDADE INTERIOR

Tabela B.7. Dicionário de objectos do sensor de luminosidade interior.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE COMUNICAÇÕES STANDARD DSP301							
1000	.	Device type	U32	ro	0x00020000	N	DSP 0x000 Digital outputs=0x0002
1001	.	Error register	U8	ro	0	S	Erros indicados em DSP301
1002	.	Manufacturer status register	U32	ro	0	N	
1004	.	# PDOs supported	ARRAY				
	0	# PDOs supported	U32	ro	0x00000001	N	0RPDO e 1TPDO
	1	# PDOs sync	U32	ro	0x00000000	N	0RPDO e 0TPDO
	2	# PDOs async	U32	ro	0x00000001	N	0RPDO e 1TPDO
1005	.	COB-ID SYNC message	U32	rw	0x00000080	N	SYNC não usada
1006	.	Communication cycle period	U32	rw	0	N	Não utilizado
1007	.	Synchronous window length	U32	rw	0	N	Não utilizado
1008	.	Manufacturer device name	VisStr	c	SLI	N	SLI - Sensor de luminosidade interior
1009	.	Man. hardware version	VisStr	c	0	N	
100A	.	Man. software version	VisStr	c	0	N	
100F	.	# SDOs supported	U32	ro	0x00000001	N	0 SDO client e 1 SDO server
1010	.	Store parameters	ARRAY				Guarda os parâmetros em memória não volátil
	0	Largest supported sub index	U8	ro	4	N	
	1	Save all parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	2	Save comm. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	3	Save app. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	4	Save man. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
1011	.	Restore default parameters	ARRAY				
	0	Largest supported sub index	U32	ro	4	N	
	1	Restore all parameters	U32	rw	0x00000001	N	Restaura parâmetros
	2	Restore comm. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	3	Restore app. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	4	Restore man. parameters	U32	rw	0x00000001	N	Restaura parâmetros
1800	.	1 TPDO parameter	RECORD				TPDO1 com. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	180h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
ÁREA DO PERFIL DE DISPOSITIVO STANDARD							
ÁREA DO PERFIL ESPECÍFICO DO FABRICANTE							
2000	.	Parâmetros relativos à luminosidade interior					
	0	Number of objects	U8	ro	3	N	Número de objectos
	1	Luminosidade mínima (LIMIN)	U8	rw	0x28	N	Min=0%=0x00 Max=100%=0x64
	2	Luminosidade máxima (LIMAX)	U8	rw	0x3C	N	Min=0%=0x00 Max=100%=0x64
	3	Luminosidade actual (LI)	U8	rw	N	N	Min=0%=0x00 Max=100%=0x64
2001	.	Estado da luminosidade interior (SLI)	U8	rw	N	S	Pouca=0x00 Adequada=0x01 Muita=0x02

B.6. DICIONÁRIO DO TECLADO DE CONTROLO MANUAL DE ESTORES

Tabela B.8. Dicionário de objectos do teclado de controlo manual de estores.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE COMUNICAÇÕES STANDARD DSP301							
1000	.	Device type	U32	ro	0x00030000	N	DSP 0x000 Digital outputs and inputs=0x0003
1001	.	Error register	U8	ro	0	S	Erros indicados em DSP301
1002	.	Manufacturer status register	U32	ro	0	N	
1004	.	# PDOs supported	ARRAY				
	0	# PDOs supported	U32	ro	0x00010001	N	1RPDO e 1TPDO
	1	# PDOs sync	U32	ro	0x00000000	N	0RPDO e 0TPDO
	2	# PDOs async	U32	ro	0x00010001	N	1RPDO e 1TPDO
1005	.	COB-ID SYNC message	U32	rw	0x00000080	N	SYNC não usada
1006	.	Communication cycle period	U32	rw	0	N	Não utilizado
1007	.	Synchronous window length	U32	rw	0	N	Não utilizado
1008	.	Manufacturer device name	VisStr	c	TE	N	TE - teclado de controlo manual do estore
1009	.	Man. hardware version	VisStr	c	0	N	
100A	.	Man. software version	VisStr	c	0	N	
100F	.	# SDOs supported	U32	ro	0x00000001	N	0 SDO client e 1 SDO server
1010	.	Store parameters	ARRAY				Guarda os parâmetros em memória não volátil
	0	Largest supported sub index	U8	ro	4	N	
	1	Save all parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	2	Save comm. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	3	Save app. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	4	Save man. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
1011	.	Restore default parameters	ARRAY				
	0	Largest supported sub index	U32	ro	4	N	
	1	Restore all parameters	U32	rw	0x00000001	N	Restaura parâmetros
	2	Restore comm. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	3	Restore app. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	4	Restore man. parameters	U32	rw	0x00000001	N	Restaura parâmetros
1400	.	1 RPDO parameter	RECORD				RPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1800	.	1 TPDO parameter	RECORD				TPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	180h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
ÁREA DO PERFIL DE DISPOSITIVO STANDARD							
ÁREA DO PERFIL ESPECÍFICO DO FABRICANTE							
201E	.	Feedback do estado do estore (FEED_EST)	U8	rw	0x00	S	Off=0x00 Fechar=0x01 Abrir=0x02
201F	.	Feedback da posição do estore (FEED_POE)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
2020	.	Posição do estore no teclado (INT_POE)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
2021	.	Estado definido no teclado do estore (INT_EST)	U8	rw	0x02	S	Off=0x00; On=0x01 Automático=0x02

B.7. DICIONÁRIO DO TECLADO DE CONTROLO MANUAL DE LÂMPADAS

Tabela B.9. Dicionário de objectos do teclado de controlo manual de lâmpadas.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE COMUNICAÇÕES STANDARD DSP301							
1000	.	Device type	U32	ro	0x00020000	N	DSP 0x000 Digital outputs=0x0002
1001	.	Error register	U8	ro	0	S	Erros indicados em DSP301
1002	.	Manufacturer status register	U32	ro	0	N	
1004	.	# PDOs supported	ARRAY				
	0	# PDOs supported	U32	ro	0x00000001	N	0RPDO e 1TPDO
	1	# PDOs sync	U32	ro	0x00000000	N	0RPDO e 0TPDO
	2	# PDOs async	U32	ro	0x00000001	N	0RPDO e 1TPDO
1005	.	COB-ID SYNC message	U32	rw	0x00000080	N	SYNC não usada
1006	.	Communication cycle period	U32	rw	0	N	Não utilizado
1007	.	Synchronous window length	U32	rw	0	N	Não utilizado
1008	.	Manufacturer device name	VisStr	c	TL	N	TL - teclado de controlo manual da lâmpada
1009	.	Man. hardware version	VisStr	c	0	N	
100A	.	Man. software version	VisStr	c	0	N	
100F	.	# SDOs supported	U32	ro	0x00000001	N	0 SDO client e 1 SDO server
1010	.	Store parameters	ARRAY				Guarda os parâmetros em memória não volátil
	0	Largest supported sub index	U8	ro	4	N	
	1	Save all parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	2	Save comm. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	3	Save app. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	4	Save man. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
1011	.	Restore default parameters	ARRAY				
	0	Largest supported sub index	U32	ro	4	N	
	1	Restore all parameters	U32	rw	0x00000001	N	Restaura parâmetros
	2	Restore comm. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	3	Restore app. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	4	Restore man. parameters	U32	rw	0x00000001	N	Restaura parâmetros
1800	.	1 TPDO parameter	RECORD				TPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	180h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
ÁREA DO PERFIL DE DISPOSITIVO STANDARD							
ÁREA DO PERFIL ESPECÍFICO DO FABRICANTE							
201C	.	Estado definido no teclado da iluminação (INT_LAMP)	U8	rw	0x02	S	Off=0x00 On=0x01 Automático=0x02
201D	.	Luminosidade definida no teclado (INT_LL)	U8	rw	0x00	S	Min=0%=0x00 Max=100%=0x64

B.8. DICIONÁRIO DO TECLADO DE CONTROLO MANUAL DE TOLDOS

Tabela B.10. Dicionário de objectos do teclado de controlo manual de toldos.

Índice (hex)	Sub-índice	Nome	Dados/ objecto	Atr.	Default	PDO Map	Comentários
ÁREA DO PERFIL DE COMUNICAÇÕES STANDARD DSP301							
1000	.	Device type	U32	ro	0x00030000	N	DSP 0x000 Digital outputs and inputs=0x0003
1001	.	Error register	U8	ro	0	S	Erros indicados em DSP301
1002	.	Manufacturer status register	U32	ro	0	N	
1004	.	# PDOs supported	ARRAY				
	0	# PDOs supported	U32	ro	0x00010001	N	1RPDO e 1TPDO
	1	# PDOs sync	U32	ro	0x00000000	N	0RPDO e 0TPDO
	2	# PDOs async	U32	ro	0x00010001	N	1RPDO e 1TPDO
1005	.	COB-ID SYNC message	U32	rw	0x00000080	N	SYNC não usada
1006	.	Communication cycle period	U32	rw	0	N	Não utilizado
1007	.	Synchronous window length	U32	rw	0	N	Não utilizado
1008	.	Manufacturer device name	VisStr	c	TT	N	TT - teclado de controlo manual do toldo
1009	.	Man. hardware version	VisStr	c	0	N	
100A	.	Man. software version	VisStr	c	0	N	
100F	.	# SDOs supported	U32	ro	0x00000001	N	0 SDO client e 1 SDO server
1010	.	Store parameters	ARRAY				Guarda os parâmetros em memória não volátil
	0	Largest supported sub index	U8	ro	4	N	
	1	Save all parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	2	Save comm. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	3	Save app. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
	4	Save man. parameters	U32	rw	0x00000002	N	Guarda parâmetros de forma autónoma
1011	.	Restore default parameters	ARRAY				
	0	Largest supported sub index	U32	ro	4	N	
	1	Restore all parameters	U32	rw	0x00000001	N	Restaura parâmetros
	2	Restore comm. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	3	Restore app. parameters	U32	rw	0x00000001	N	Restaura parâmetros
	4	Restore man. parameters	U32	rw	0x00000001	N	Restaura parâmetros
1400	.	1 RPDO parameter	RECORD				RPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	200h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
1800	.	1 TPDO parameter	RECORD				TPDO1 comm. parameter
	0	Largest sub-index supported	U8	ro	2	N	
	1	COB-ID used by PDO	U32	rw	180h+NODE ID	N	
	2	Transmission type	U8	rw	FE	N	Assíncrono
ÁREA DO PERFIL DE DISPOSITIVO STANDARD							
ÁREA DO PERFIL ESPECÍFICO DO FABRICANTE							
2022	.	Feedback do estado do toldo (FEED_TOL)	U8	rw	0x00	S	Off=0x00 Fechar=0x01 Abrir=0x02
2023	.	Feedback da posição do toldo (FEED_POT)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
2024	.	Posição do toldo no teclado (INT_POT)	U8	rw	0x00	S	Fechado=0%=0x00 Aberto=100%=0x64
2025	.	Estado definido no teclado do toldo (INT_TOL)	U8	rw	0x02	S	Off=0x00 On=0x01 Automático=0x02

ANEXO C – AS ELECTRONIC DATA SHEETS

C.1. A EDS DO ACCIONADOR DE ESTORES

```
;--------------------------------------------------------------------
;--- Bruno da Silva Abreu
;--- Departamento de Eng. Electromecânica
;--- Universidade da Beira Interior
;--- Covilhã
;--------------------------------------------------------------------

[FileInfo]
FileName=AE.EDS
FileVersion=0
FileRevision=0
Description=EDS-File Accionador de Estores
CreationTime=11:37AM
CreationDate=15-07-09
CreatedBy=BSA
ModificationTime=11:37AM
ModificationDate=15-07-09
ModifiedBy=BSA

[DeviceInfo]
VendorName=UBI
VendorNumber=4
ProductName=Accionador de estores
ProductNumber=777-999
ProductVersion=1
ProductRevision=01
OrderCode=777-999
BaudRate_10=0
BaudRate_20=0
BaudRate_50=1
BaudRate_100=0
BaudRate_125=0
BaudRate_250=0
BaudRate_500=0
BaudRate_800=0
BaudRate_1000=0
SimpleBootUpMaster=0
SimpleBootUpSlave=1
ExtendedBootUpMaster=0
ExtendedBootUpSlave=0
Granularity=8

[StandardDataTypes]
0x0001=0
0x0002=0
0x0003=0
0x0004=0
0x0005=1
0x0006=1
0x0007=1
0x0008=0
0x0009=1
0x000A=1
0x000B=0
0x000C=0
0x000D=0
0x000E=0
0x000F=0
0x0020=1
0x0021=1
0x0022=1

[DummyUsage]
Dummy0001=0
Dummy0002=0
Dummy0003=0
Dummy0004=0
Dummy0005=0
Dummy0006=0
Dummy0007=0

[MandatoryObjects]
SupportedObjects=2
1=0x1000
2=0x1001
```

```
[1000]
SubNumber=0
ParameterName=device type
ObjectType=0x07
DataType=0x0007
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=0x00030000
PDOMapping=0

[1001]
SubNumber=0
ParameterName=error register
ObjectType=0x07
DataType=0x0005
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=2
PDOMapping=1

[OptionalObjects]
SupportedObjects=16
1=0x1002
2=0x1004
3=0x1005
4=0x1006
5=0x1007
6=0x1008
7=0x1009
8=0x100A
9=0x100F
10=0x1010
11=0x1011
12=0x1400
13=0x1401
14=0x1402
15=0x1403
16=0x1800
```

```
[1000]
SubNumber=0
ParameterName=device type
ObjectType=7
DataType=0x0007
DefaultValue=0x30000
AccessType=ro
PDOMapping=0

[1001]
SubNumber=0
ParameterName=error register
ObjectType=0x07
DataType=0x0005
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=
PDOMapping=1

[1002]
SubNumber=0
ParameterName=manufacturer status register
ObjectType=0x07
DataType=0x0007
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=
PDOMapping=0

[1004]
SubNumber=2
ParameterName=number of PDOs supported

[1004sub0]
ParameterName=number of PDOs supported
ObjectType=0x0008
DataType=0x07
```

AccessType=ro
DefaultValue=0x40001
PDOMapping=0

[1004sub1]
ParameterName=number of synchronous PDOs
ObjectType=0x0008
DataType=0x07
AccessType=ro
DefaultValue=0x00000
PDOMapping=0

[1004sub2]
ParameterName=number of asynchronous PDOs
ObjectType=0x0008
DataType=0x07
AccessType=ro
DefaultValue=0x40001
PDOMapping=0

[1005]
SubNumber=0
ParameterName=COB-ID SYNC message
ObjectType=0x07
DataType=0x0007
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00000080
PDOMapping=0

[1006]
SubNumber=0
ParameterName=communication cycle period
ObjectType=0x07
DataType=0x0007
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1007]
SubNumber=0
ParameterName=synchronous window length
ObjectType=0x07
DataType=0x0007
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1008]
SubNumber=0
ParameterName=manufacturer device name
ObjectType=0x07
DataType=0x0009
AccessType=const
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1009]
SubNumber=0
ParameterName=manufacturer hardware version
ObjectType=0x07
DataType=0x0009
AccessType=const
LowLimit=
HighLimit=
DefaultValue=0
PDOMapping=0

[100A]
SubNumber=0
ParameterName=manufacturer software version
ObjectType=0x07
DataType=0x0009

AccessType=const
LowLimit=
HighLimit=
DefaultValue=0
PDOMapping=0

[100F]
SubNumber=0
ParameterName=number of SDO's supported
ObjectType=0x07
DataType=0x07
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=0x00000001
PDOMapping=0

[1010]
SubNumber=4
ParameterName=store parameters

[1010sub0]
ParameterName=largest supported Subindex
ObjectType=0x0008
DataType=0x05
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=0x04
PDOMapping=0

[1010sub1]
ParameterName=save all parameters
ObjectType=0x0008
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1010sub2]
ParameterName=save communication parameters
ObjectType=0x0008
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1010sub3]
ParameterName=save application parameters
ObjectType=0x0008
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1010sub4]
ParameterName=save manufacturer defined parameters
ObjectType=0x0008
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1011]
SubNumber=4
ParameterName=restore default parameters

[1011sub0]
ParameterName=largest supported Subindex
ObjectType=0x0008
DataType=0x05
AccessType=ro

LowLimit=
HighLimit=
DefaultValue=0x04
PDOMapping=0

[1011sub1]
ParameterName=restore all default parameters
ObjectType=0x0008
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1011sub2]
ParameterName=restore communication default parameters
ObjectType=0x0008
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1011sub3]
ParameterName=restore application default parameters
ObjectType=0x0008
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1011sub4]
ParameterName=restore manufacturer defined default parameters
ObjectType=0x0008
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[1400]
SubNumber=3
ParameterName=RPDO 1 Communication Parameter

[1400sub0]
ParameterName=number of supported entries
ObjectType=0x0009
DataType=0x05
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=2
PDOMapping=0

[1400sub1]
ParameterName=COB-ID used by PDO
ObjectType=0x0009
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=
PDOMapping=0

[1400sub2]
ParameterName=transmission type
ObjectType=0x0009
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0xfe
PDOMapping=0

[1401]
SubNumber=3
ParameterName=RPDO 2 Communication Parameter

[1401sub0]
ParameterName=number of supported entries
ObjectType=0x0009
DataType=0x05
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=2
PDOMapping=0

[1401sub1]
ParameterName=COB-ID used by PDO
ObjectType=0x0009
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=
PDOMapping=0

[1401sub2]
ParameterName=transmission type
ObjectType=0x0009
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0xfe
PDOMapping=0

[1402]
SubNumber=3
ParameterName=RPDO 3 Communication Parameter

[1402sub0]
ParameterName=number of supported entries
ObjectType=0x0009
DataType=0x05
AccessType=ro
LowLimit=
HighLimit=
DefaultValue=2
PDOMapping=0

[1402sub1]
ParameterName=COB-ID used by PDO
ObjectType=0x0009
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=
PDOMapping=0

[1402sub2]
ParameterName=transmission type
ObjectType=0x0009
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue= 0xfe
PDOMapping=0

[1403]
SubNumber=3
ParameterName=RPDO 4 Communication Parameter

[1403sub0]
ParameterName=number of supported entries
ObjectType=0x0009
DataType=0x05
AccessType=ro

LowLimit=
HighLimit=
DefaultValue=2
PDOMapping=0

[1403sub1]
ParameterName=COB-ID used by PDO
ObjectType=0x09
DataType=0x0007
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=
PDOMapping=0

[1403sub2]
ParameterName=transmission type
ObjectType=0x09
DataType=0x0005
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0xfe
PDOMapping=0

[1800]
SubNumber=2
ParameterName=TPDO 1 Communication Parameter

[1800sub0]
ParameterName=number of supported entries
ObjectType=0x0009
DataType=0x05
AccessType=ro
DefaultValue=2
PDOMapping=0

[1800sub1]
ParameterName=COB-ID used by PDO
ObjectType=0x0009
DataType=0x07
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=
PDOMapping=0

[1800sub2]
ParameterName=transmission type
ObjectType=0x0009
DataType=0x05
AccessType=rw
LowLimit=0
HighLimit=0xff
DefaultValue=0xfe
PDOMapping=0

[ManufacturerObjects]
SupportedObjects=10
1=0x200D
2=0x200E
3=0x200F
4=0x2010
5=0x2011
6=0x2012
7=0x2013
8=0x2014
9=0x2015
10=0x2016

[200D]
ParameterName=Modo de funcionamento E_MOD
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=0

[200E]
ParameterName=Estado do estore EST
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x00
PDOMapping=1

[200F]
ParameterName=Posição do estore POE
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=0x00
HighLimit=0x64
DefaultValue=0x00
PDOMapping=1

[2010]
ParameterName=Estado definido no teclado da iluminação E_INT_LAMP
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x02
PDOMapping=1

[2011]
ParameterName=Estado da lâmpada E_LAMP
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x01
PDOMapping=1

[2012]
ParameterName=Luminosidade da lâmpada E_LL
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=0x00
HighLimit=0x64
DefaultValue=0x00
PDOMapping=1

[2013]
ParameterName=Luminosidade interior E_SLI
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x01
PDOMapping=1

[2014]
ParameterName=Luminosidade exterior E_SLE
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x01
PDOMapping=1

[2015]
ParameterName=Posição do estore no interruptor E_INT_POE
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=0x00
HighLimit=0x64
DefaultValue=0x00

PDOMapping=1

[2016]
ParameterName=Estado definido no teclado do estore
E_INT_EST
ObjectType=0x0007
DataType=0x05
AccessType=rw
LowLimit=
HighLimit=
DefaultValue=0x02
PDOMapping=1