

Miguel Angel Simões Santiago

**Tecnologias de Apoio para a Autonomia
em
Ambiente Doméstico**

Dissertação submetida para a obtenção do grau de Mestre
em Engenharia Electrotécnica e de Computadores

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores

Novembro de 1999

Tese realizada sob supervisão do
Prof. Dr. José Manuel Martins Ferreira
Professor Auxiliar do
Departamento de Engenharia Electrotécnica e de Computadores da
Faculdade de Engenharia da Universidade do Porto

Aos meus Pais.

Resumo

Palavras chave: *Paralisia Cerebral, Sistemas de Controlo Ambiental, Comunicação Aumentativa e Alternativa, Acessibilidade, Controller Area Network (CAN), Edifícios Inteligentes.*

Esta dissertação descreve um sistema desenvolvido em parceria com a Associação Portuguesa de Paralisia Cerebral (APPC) e com o Centro de Reabilitação de Paralisia Cerebral do Porto (CRPCP), com o objectivo de proporcionar às pessoas com paralisia cerebral uma maior autonomia em ambiente doméstico. Uma vez que a autonomia não se restringe às funções de controlo ambiental, as soluções desenvolvidas permitem igualmente aos utilizadores o acesso a diversas outras actividades que podem ser realizadas em casa, tais como a comunicação assistida (oral e telefónica) e o trabalho com aplicações em ambiente Windows (e.g. processamento de texto, acesso à Internet, etc.). O conjunto de soluções desenvolvidas compreende duas componentes principais, dizendo uma respeito à rede domótica subjacente ao controlo ambiental e a outra à aplicação Windows de interface com o utilizador. Esta aplicação permite igualmente o acesso aos recursos necessários para a comunicação assistida, local ou remota, neste caso através de um telefone do tipo mãos-livres.

A rede domótica recorreu à especificação CAN (*Controller Area Network*), pelas suas boas características em termos de custo, disponibilidade e fiabilidade. Foi desenvolvida uma carta de baixo custo baseada num microcontrolador da família 80C51, que em conjunto com um periférico CAN dedicado, permitiram implementar uma solução económica e de fácil instalação. O desenvolvimento de um conjunto de rotinas em C, que codificam as principais funções comuns a cada nó, proporcionou uma solução que simplifica grandemente o desenvolvimento do código a instalar em cada carta.

O desenvolvimento da aplicação de interface com o utilizador constitui a segunda componente principal do trabalho descrito e proporciona uma solução uniforme para o acesso às funções de controlo ambiental, de comunicação assistida e de lançamento e trabalho com aplicações Windows, recorrendo como dispositivo de selecção a um *switch* (interruptor de contacto binário, adaptado às facilidades motoras de cada utilizador), de qualquer dos vários tipos que são normalmente usados pelas pessoas com paralisia cerebral. A comunicação assistida permite a reprodução de mensagens digitalizadas, que podem servir para comunicar com um interlocutor local ou através do telefone. Neste segundo caso, a aplicação desenvolvida recorre a um modem normalizado para estabelecer a chamada através de um telefone mãos-livres, dispensando o utilizador de ter que efectuar directamente a marcação dos dígitos. No que respeita ao acesso a aplicações em ambiente Windows, é possível a emulação do rato e do teclado, mantendo-se o mesmo dispositivo de selecção. Esta emulação garante o acesso à generalidade das aplicações através do mesmo interface que é usado com as restantes componentes do sistema, permitindo implementar soluções de (tele-) trabalho, capazes de contribuir de forma significativa para a autonomia em ambiente doméstico.

Todas as soluções desenvolvidas foram instaladas na Casa da Maceda, pertencente ao Núcleo Regional do Norte da APPC, onde estão disponíveis para a experimentação prática por parte dos terapeutas e de utentes com paralisia cerebral.

Abstract

Keywords: *Cerebral Palsy, Environmental Control Systems, Augmentative and Alternative Communication, Accessibility, Controller Area Network (CAN), Intelligent Buildings.*

This thesis presents a set of assistive technology solutions developed under partnership with the Portuguese Cerebral Palsy Association (APPC) and the Cerebral Palsy Rehabilitation Centre of Porto (CRPCP), with the objective of improving the autonomy of persons with cerebral palsy in their homes. Since autonomy is not restricted to environmental control, the solutions that were developed include features to support a range of other activities normally carried out at home, such as assistive communication (locally or via telephone) and access to Windows applications (e.g. text processing, Internet access, etc.). The work done comprises two main components: the home network that supports the environmental control features and a Windows user interface application. This application enables access to the assistive communication features as well, either for local or remote communication, in the latter case through a hands-free telephone.

The CAN (Controller Area Network) specification was selected to support the home network due to its wide availability, low cost and high reliability. A small printed circuit board, based in an 80C51 microcontroller and a dedicated CAN peripheral, was developed to provide a low-cost and easy to install solution, able to implement the required home automation features. The development of a library of C routines, coding the functions that have to be present in every node, provided an efficient solution to simplify software development for each node.

The user interface application enables a uniform access solution to environmental control and assistive communication, as well as to launch and work with Windows-based software applications, using as selection device a switch (a binary switch adapted to the special needs of each user) of the type commonly used by persons with cerebral palsy. The assistive communication functions enable access to digitised messages, which can be used for local or remote communication. A standard modem is used in the latter case to establish the call through a hands-free telephone, so that the user does not have to dial the number himself. Access to Windows software applications is done through keyboard and mouse emulation, using the same selection device previously referred. The same interface package that is used for environmental control and assistive communication is therefore used also to access Windows-based applications, enabling the implementation of (tele-) working environments which can effectively contribute to improve the autonomy at home.

All the solutions developed were installed in the House of Maceda, belonging to the northern branch of APPC, where they can be tested by therapists and by users with cerebral palsy.

Résumé

Mots-clés: *Paralysie Cérébrale, Système de contrôle de l'environnement, Transmission Augmentative et Alternative, Accessibilité, Controller Area Network (CAN), Bâtiments Intelligents.*

Cette thèse présente un ensemble de solutions assistive de technologie développées sous partenariat avec l'Association Portugaise de Paralysie Cérébrale (APPC) et le Centre de Réadaptation de Paralysie Cérébrale de Porto, avec l'objectif d'améliorer l'autonomie des personnes avec la paralysie cérébrale dans leurs maisons. Puisque l'autonomie n'est pas limitée au contrôle de l'environnement, les solutions développées incluent des dispositifs pour supporter un intervalle d'autres activités normalement effectuées à la maison, comme la transmission assistive (localement ou par l'intermédiaire de téléphone) et l'accès aux applications de Windows (par exemple traitement de textes, accès Internet, etc.). Le travail effectué comporte deux composants principaux: le réseau à la maison qui supporte les dispositifs de contrôle de l'environnement et une application d'interface Windows avec l'utilisateur. Cette application permet l'accès aux dispositifs de transmission assistive, pour la transmission locale ou à distance, dans le dernier cas par un téléphone mains libres.

Le CAN (Controller Area Network) a été choisi pour le réseau à la maison dû à sa large disponibilité, bas coût et fiabilité. Une petite carte électronique, basée dans un microcontrôleur 80C51 et un périphérique dédié CAN, a été développée pour fournir une solution de bas coût et facile d'installer, capable de mettre en application les dispositifs d'automatisation exigés à la maison. Le développement d'une bibliothèque des sous-programmes en C, codant les fonctions qui doivent être présentes dans chaque noeud, à devenu une solution efficace pour simplifier le développement de software pour chaque noeud. L'application d'interface avec l'utilisateur permet une solution uniforme pour l'accès aux fonctions de contrôle de l'environnement et de la transmission assistive, comme au lancement et travail avec des applications de software Windows-basée, utilisant comme dispositif de sélection un commutateur (un commutateur binaire adapté aux besoins spéciaux de chaque utilisateur) du type généralement utilisé par des personnes avec la paralysie cérébrale. La transmission assistive permettent l'accès aux messages numérisés, qui peuvent être utilisés localement ou pour la transmission à distance. Un modem standard est utilisé dans le dernier cas pour établir l'appel par un téléphone mains libres, ainsi l'utilisateur n'a pas besoin de s'occuper lui-même pour obtenir la connexion. L'accès aux applications de software de Windows est fait par l'émulation de clavier informatique et de curseur-souris, à l'aide du même dispositif de sélection précédemment référé. Le même module d'interface qui est utilisé pour le contrôle de l'environnement et la transmission assistive est donc employée pour accéder à des applications Windows-basée, permettant la mise en place des environnements de (tele-) travail qui peuvent pertinemment contribuer pour améliorer l'autonomie à la maison.

Toutes les solutions développées ont été installées dans la Maison de Maceda, appartenant au branch nord de l'APPC, où elles peuvent être testées par des therapists et par des utilisateurs avec paralysie cérébrale.

Agradecimentos

Este trabalho não teria sido possível sem a contribuição de um conjunto de pessoas que me apoiaram em diversas fases da sua realização e às quais quero manifestar publicamente o meu agradecimento.

Os utentes da APPC e do CRPCP, bem como todos os terapeutas com quem tive oportunidade de contactar, desempenharam um papel chave na identificação dos requisitos e no acompanhamento prestado durante a fase de desenvolvimento e testes.

O grupo em que estive inserido durante estes últimos três anos desempenhou igualmente um papel relevante na criação de um ambiente de trabalho propício à realização de tarefas de investigação e desenvolvimento, quer pelo espírito de entre-ajuda quer pelo excelente ambiente e camaradagem existente entre todos. Neste sentido, quero deixar também uma palavra de agradecimento aos seus membros, nomeadamente aos Mestres Álvaro Teixeira e Manuel Gericota, bem como à Eng.^a Li Xin. Ainda neste âmbito, mas com um destaque especial pela ajuda directamente prestada, quero referir a Dr.^a Inês Cambeiro e o Doutor Gustavo Alves, cuja ajuda para a conclusão deste documento foi inestimável.

Ao Prof. Martins Ferreira quero em particular manifestar o meu reconhecimento, pela paciência com que orientou este trabalho e pela disponibilidade sempre manifestada, quer em termos de orientação científica quer em diversos aspectos da sua execução.

Por fim, mas principalmente, quero agradecer à Maria José e à minha família pela compreensão manifestada em inúmeras ocasiões, face às horas de atenção que lhes foram roubadas para permitir a conclusão deste trabalho.

Índice

1	Introdução	1
1.1	Âmbito do trabalho.....	1
1.2	Motivação para o trabalho apresentado.....	1
1.3	Organização da tese.....	2
2	Tecnologias de apoio e autonomia	5
2.1	Utilizadores com necessidades especiais	5
2.2	A importância (relativa) da tecnologia.....	6
2.3	Metodologia de desenvolvimento	7
2.4	Estado actual das soluções disponíveis comercialmente.....	8
2.4.1	Sistemas Domóticos.....	9
2.4.2	Sistema de controlo ambiental.....	13
2.4.3	Apoio à comunicação	16
2.4.4	Acesso a aplicações em ambiente PC ou MAC.....	20
3	Requisitos do utilizador e especificação do sistema	23
3.1	Identificação dos requisitos dos utilizadores.....	23
3.2	Paralisia Cerebral	24
3.3	Especificação para o controlo ambiental.....	27
3.4	Especificação para a comunicação oral.....	29
3.5	Especificação para a comunicação telefónica	30
3.6	Especificação para o acesso a aplicações Windows.....	30
3.7	Especificação funcional do interface com o utilizador	32
4	Arquitectura física	35
4.1	Apresentação global do sistema	35
4.2	Interface para ligação de um <i>switch</i> ao PC	36
4.3	Recursos para a comunicação oral	38
4.4	Recursos para a comunicação telefónica.....	39
4.5	Recursos para o controlo ambiental	40
4.5.1	CAN	41
4.5.2	Nó CAN.....	49
4.5.3	<i>Software</i> da aplicação	70
5	Interface com o utilizador	85
5.1	Elementos da Interface.....	85
5.2	Descrição da implementação.....	88

5.2.1 Classe TScanApplication.....	88
5.2.2 Classe TScanWindow.....	90
5.2.3 Classe “TPointerWindow”	91
5.2.4 Classe "Botão"	92
5.2.5 Classe “Mini-Botão”	94
5.3 Descrição da aplicação "Fernão Magalhães"	94
5.3.1 Adaptação a cada utilizador.....	95
5.3.2 Controlo Ambiental	104
5.3.3 Comunicação Assistida.....	105
5.3.4 Estabelecimento de ligações telefónicas.....	105
5.3.5 Definição de cenários	105
5.3.6 Lançamento de aplicações Windows	106
5.4 Exemplo de aplicação.....	108
6 Experimentação prática: A instalação na APPC	111
6.1 Características do edifício na Travessa da Maceda.....	111
6.2 Instalação do sistema de controlo ambiental.....	113
6.2.1 Hall de entrada.....	113
6.2.2 Recepção.....	114
6.2.3 Corredor junto às escadas	116
6.2.4 Sala de trabalhos	116
6.3 Configuração do interface com o utilizador.....	117
7 Conclusão	119
7.1 Os resultados obtidos	119
7.2 Futuro da instalação	120
7.3 Perspectivas de futuro	120
8 Referências	123
9 Anexo	
Código Fonte da Gestão de Comunicações CAN	
Código Fonte do CAN sobre RS-232	
Código Fonte de um Nó CAN do tipo Iluminação	
Código Fonte de um Nó CAN do tipo Dimmer	
Código Fonte de um Nó CAN do tipo IntDoor	
Código Fonte de um Nó CAN do tipo Tomada	
Código Fonte de um Nó CAN do tipo Persiana	
Código Fonte de um Nó CAN do tipo Janela	

Lista de Figuras

Figura 2.1: Excesso tecnológico que não resolve os problemas.....	7
Figura 2.2: Tecnologia inadequada às necessidades do utilizador.....	7
Figura 2.3: Kit X10 para controlo por computador.....	9
Figura 2.4: CEBus sobre a rede telefónica.....	10
Figura 2.5: NeuronChip.....	10
Figura 2.6: Módulos EIB.....	12
Figura 2.7: Módulo BatiBus.....	12
Figura 2.8: Possum Companion.....	13
Figura 2.9: Possum FreeWay.....	14
Figura 2.10: Director e Scanning Director.....	15
Figura 2.11: Interface gráfica do CyberHouse.....	15
Figura 2.12: Pictogramas.....	16
Figura 2.13: <i>Switches</i> simples e duplos.....	16
Figure 2.14: Pictogramas , Minspeak e Bliss.....	17
Figura 2.15: SpeakEasy.....	18
Figura 2.16: Um utilizador que usa o Liberator.....	19
Figura 2.17: Uso do apontador por Infravermelhos no Liberator.....	19
Figura 2.18: HandiVoice.....	20
Figura 2.19: MouseTrap.....	20
Figura 2.20: WiViK2.....	21
Figura 2.21: Emulação de teclado com o TTAM.....	21
Figura 3.1: Cérebro humano.....	25
Figura 4.1: Esquema global do sistema.....	35
Figura 4.2: Adaptador de <i>switch</i> simples.....	36
Figura 4.3: <i>Pinout</i> da porta paralela.....	37
Figura 4.4: Esquema de um adaptador de <i>switch</i>	37
Figura 4.5: Placa de som.....	38
Figura 4.6: <i>Software</i> de tratamento de som.....	38
Figura 4.7: Modem interno US Robotics.....	39
Figura 4.8: Esquema de ligação do Modem.....	40
Figura 4.9: Sistema Domótico CAN.....	40
Figura 4.10: Estrutura de um nó CAN.....	41
Figura 4.11: Camadas OSI implementadas pelo CAN.....	42
Figura 4.12: Trama de Dados.....	43
Figura 4.13: Trama Remota.....	45
Figura 4.14: Trama de Erro.....	45
Figura 4.15: Trama de Sobrecarga.....	46
Figura 4.16: Processo de arbitragem entre três estações.....	48
Figura 4.17: Nó CAN.....	49
Figura 4.18: Diagrama de blocos de um nó CAN.....	50
Figura 4.19: Esquema eléctrico de um nó CAN.....	51

Figura 4.20: Esquema interno do Controlador CAN.....	52
Figura 4.21: Mapa de endereçamento interno do PCA82C200	54
Figura 4.22: Receptor CAN configurável.....	58
Tabela 4.7: Conteúdo do registo 0 de regulação do tempo do barramento	61
Tabela 4.8: Conteúdo do registo 1 de regulação do tempo do barramento	62
Figura 4.23: Registo de controlo da saída e andar de saída	63
Tabela 4.10: Efeito dos <i>bit</i> OCMODE1 e OCMODE0.....	64
Tabela 4.11: Conteúdo do registo de teste.....	64
Tabela 4.12: <i>Buffer</i> de Transmissão.....	64
Figura 4.24: Período do <i>bit</i>	66
Figura 4.25: Diagrama de blocos do PCA82C250.....	69
Figura 4.26: Ligação do transceptor CAN.....	69
Figura 4.27: Código presente em cada nó CAN.....	70
Figura 4.28: Fluxograma da função Pwrite	71
Figura 4.29: Fluxograma da função Pread	71
Figura 4.30: Fluxograma de can_int , para o atendimento de interrupções	72
Figura 4.31: Fluxograma da função clear_can_buffer	73
Figura 4.32: Fluxograma da função get_can_message	73
Figura 4.33: Fluxograma da função send_can_message	74
Figura 4.34: Fluxograma da função can_receive	75
Figura 4.35: Fluxograma da função can_transmit	75
Figura 4.36: Fluxograma da função can_error	76
Figura 4.37: Fluxograma da função can_busy	76
Figura 4.38: Fluxograma da função new_can_message	77
Figura 4.39: Fluxograma da função can_setup	77
Figura 4.40: Fluxograma da camada de aplicação	78
Figura 4.41: Árvore de descodificação de mensagens.....	80
Figura 5.1: Elementos da interface.....	86
Figura 5.2: Organização hierárquica dos quadros da interface com o utilizador.....	86
Figura 5.3: Varrimento linear	87
Figura 5.4: Varrimento linha-coluna.....	88
Figura 5.5: Representação da estrutura de dados.....	89
Figura 5.6: Estado inicial da interface, antes de criados os menus.....	95
Figura 5.7: Botões iniciais de um sub-menu.....	96
Figura 5.8: O menu " <i>File</i> "	96
Figura 5.9: Mensagem de alerta quando se quer abandonar uma interface alterada.....	97
Figura 5.10: <i>File Requester</i> correspondente à opção <i>Open</i>	97
Figura 5.11: <i>File Requester</i> correspondente à opção <i>Save As</i>	98
Figura 5.12: Opções do menu " <i>Insert</i> "	98
Figura 5.13: Criação do objecto Menu.....	99
Figura 5.14: Criação do objecto Comando	100
Figura 5.15: Criação do objecto Mensagem.....	100
Figura 5.16: Criação do objecto Telefone.....	101
Figura 5.17: Criação do objecto Aplicação.....	102
Figura 5.18: Menu " <i>Menus</i> "	102

Figura 5.19: Caixa de diálogo para especificar o tempo de varrimento	103
Figura 5.20: Menu para selecção de cores	103
Figura 5.21: Menu de edição de Botões	104
Figura 5.23: Emulação de rato	107
Figura 5.24: Emulação de teclado	107
Figura 5.25: Interface de demonstração	108
Figura 5.26: Quadro de apoio à comunicação oral	109
Figura 5.27: Quadro de controlo ambiental	109
Figura 5.28: Quadro de apoio ao estabelecimento de chamadas telefónicas	110
Figura 5.29: Quadro de lançamento de aplicações Windows	110
Figura 6.1: A Casa da Maceda	112
Figura 6.2: Casa da Maceda - Planta do R/C	112
Figura 6.3: Casa da Maceda - Planta do 1º Andar	113
Figura 6.4: Módulo de controlo da Porta exterior	114
Figura 6.5: Módulo de controlo lâmpada	114
Figura 6.6: Fonte de alimentação	115
Figura 6.8: Tomada de energia controlada	116
Figura 6.9: Porta interior e respectivo actuador	116

Lista de Tabelas

Tabela 4.1: Segmento de controle	55
Tabela 4.2: Conteúdo do Registo de Controle.....	56
Tabela 4.3: Conteúdo do Registo de Comando.....	57
Tabela 4.4: Combinação dos <i>bit</i> RX0A e RX1A	58
Tabela 4.5: Conteúdo do Registo de Estado.....	59
Tabela 4.6: Conteúdo do Registo de Interrupção	60
Tabela 4.7: Conteúdo do registo 0 de regulação do tempo do barramento	61
Tabela 4.8: Conteúdo do registo 1 de regulação do tempo do barramento	62
Tabela 4.9: Conteúdo do registo de controlo da saída.....	63
Tabela 4.10: Efeitos dos <i>bit</i> OCMODE1 e OCMODE0	64
Tabela 4.11: Conteúdo do registpo de teste.....	64
Tabela 4.12: <i>Buffer de Transmissão</i>	64
Tabela 4.13: Composição dos <i>Buffers de Recepção</i>	65

Capítulo 1

Introdução

Neste capítulo inicial será apresentado o contexto tecnológico e social no qual os resultados do trabalho se inserem, a que se segue a motivação que levou à realização deste trabalho. O capítulo conclui-se com a apresentação da organização da tese e com uma breve descrição dos assuntos abordados.

1.1 Âmbito do trabalho

O trabalho realizado insere-se nas tecnologias de apoio a pessoas com necessidades especiais, nomeadamente no campo das ajudas técnicas [Miln93, User96]. O objectivo principal que presidiu ao seu desenvolvimento consistiu em criar um ambiente domótico controlável da forma tradicional, ou alternativamente através de um programa de computador com uma interface adaptada aos requisitos de utilizadores com necessidades especiais.

Os principais utilizadores alvo são os membros da Associação Portuguesa de Paralisia Cerebral (APPC), que frequentam as instalações da Casa da Maceda, no Porto. Devido à sua condição médica, estes utilizadores enfrentam limitações de várias ordens, uma vez que a falta de coordenação motora dificulta a utilização de portas, janelas, interruptores, candeeiros, televisores, rádios, etc. As dificuldades apresentadas na expressão oral, por outro lado, tornam a comunicação com as outras pessoas uma tarefa nem sempre fácil.

1.2 Motivação para o trabalho apresentado

O trabalho apresentado constitui a continuação lógica do projecto de fim de curso do autor, intitulado “Tecnologias de apoio para uma maior independência de pessoas com necessidades especiais” (realizado na Faculdade de Engenharia da Universidade do Porto em 1995), durante o qual se deram os seus primeiros contactos com as tecnologias de apoio e com a realidade das pessoas com necessidades especiais.

A nível pessoal, o autor teve sempre também um gosto especial pelo desenvolvimento de *hardware* baseado em microcontroladores e do respectivo *software* de controlo, o que tornou este projecto bastante aliciante. A sua vida profissional esteve sempre associada ao desenvolvimento de aplicações em ambientes gráficos, o que facilitou a realização de uma interface adaptável às necessidades específicas dos utilizadores referidos.

Outro factor fundamental foi a tradição que o grupo de investigação coordenado pelo Prof. Martins Ferreira (orientador desta dissertação) tinha no desenvolvimento de projectos na área das tecnologias de apoio. Estes projectos permitiram ao grupo estabelecer contactos importantes com a APPC e com o Centro de Reabilitação de Paralisia Cerebral do Porto (CRPCP), fundamentais para determinar as necessidades reais dos utilizadores alvo e para criar um ambiente de experimentação adequado.

A experiência do grupo na área dos edifícios inteligentes, decorrente por sua vez do projecto TIDE HS-ADEPT, recomendou a utilização do sistema CAN, como uma alternativa previsivelmente eficiente e económica para a implementação do controlo ambiental em edifícios. Considerou-se igualmente desejável que o sistema a implementar incluísse possibilidades de gestão e monitorização, quer localmente quer de forma remota, via Internet.

O factor final que pesou na escolha do tema do trabalho foi o financiamento por parte do Projecto HORIZON/ESTIA, que permitia assegurar todos os recursos necessários para o seu desenvolvimento.

1.3 Organização da tese

Após este capítulo introdutório, apresenta-se no capítulo 2 o tema das tecnologias de apoio e a sua importância para a autonomia de pessoas com necessidades especiais. Neste capítulo inclui-se igualmente uma breve referencia às soluções disponíveis no mercado e à forma como estas soluções são integradas no dia a dia dos seus utilizadores.

No capítulo 3 apresenta-se a metodologia usada para identificar as necessidades específicas dos utilizadores com paralisia cerebral, a que se segue a descrição de funcionalidades e cuidados a ter para desenvolver ajudas técnicas nas áreas do controlo ambiental, comunicação assistida, controlo de interfaces Windows e, principalmente, na construção de uma interface para estes utilizadores.

O capítulo 4 descreve a arquitectura física necessária para suportar o sistema de apoio ao utilizador. Apresenta-se inicialmente o *hardware* necessário para desenvolver tarefas de apoio com um PC, passando-se daí para uma descrição do sistema domótico usado para suportar as funções de controlo ambiental num edifício.

O capítulo 5 trata da interface com o utilizador, a que foi dada a designação de “Fernão de Magalhães”, primeiro sob um ponto de vista técnico, no qual se descreve a arquitectura do

software, e depois sob um ponto de vista mais prático, no qual se faz uma descrição da aplicação e da sua utilização.

No capítulo 6 faz-se uma descrição sucinta da utilização do sistema domótico na casa da Maceda, para permitir aos utilizadores alvo o contacto com as soluções desenvolvidas.

Por fim, o capítulo 7 apresenta algumas reflexões sobre o trabalho realizado e indica caminhos de evolução.

Capítulo 2

Tecnologias de apoio e autonomia

Este capítulo aborda as questões relacionadas com o uso da tecnologia para aumentar a autonomia de pessoas com necessidades especiais. Não se pretende efectuar uma caracterização exaustiva do estado da tecnologia, que pela sua extensão e âmbito não se enquadram nesta dissertação, mas sim introduzir os conceitos fundamentais inerentes a este domínio. A aplicação da tecnologia para aumentar a autonomia de pessoas com necessidades especiais apresenta características específicas, que serão melhor compreendidas se começarmos por considerar o universo de utilizadores que serão os seus beneficiários directos. É importante realçar o facto de a tecnologia surgir neste domínio (mais do que em qualquer outro) como um *meio* e não como um *fim*, o que condiciona claramente a metodologia de desenvolvimento de aplicações, sendo estes dois assuntos também abordados neste capítulo. Finalmente, apresenta-se de forma sucinta o estado actual do mercado das tecnologias de apoio na área em que se situa este trabalho, de modo a realçar as lacunas que o sistema desenvolvido pretende colmatar.

2.1 Utilizadores com necessidades especiais

A Organização Mundial de Saúde define *Deficiência* como “Qualquer perda ou anomalia psicológica, fisiológica, estrutural ou funcional”. Define ainda *Incapacidade* como “Qualquer restrição ou falta de habilidade de executar uma tarefa de uma forma considerada normal para um ser humano”. *Deficiente* é definido como “Uma desvantagem para um dado indivíduo, resultante de uma incapacidade ou deficiência, que limita ou impossibilita a concretização de um papel normal (dependendo da idade, sexo e factores culturais)” [WHO].

Os utilizadores com necessidades especiais possuem capacidades físicas ou intelectuais que se encontram diminuídas de forma permanente ou temporária, em relação ao padrão considerado “normal”. Esta definição sugere um grupo de pessoas bastante abrangente, incluindo as pessoas com deficiência, idosas e todos aqueles que por acidente se encontram temporariamente impossibilitados de levar uma vida normal.

A população da Europa está estimada em 800 milhões, dos quais 355 milhões pertencem à Comunidade Europeia. Segundo estatísticas do EUROSTAT, existem 77 milhões de pessoas idosas, sendo o número de pessoas com deficiência de cerca de 43 milhões. No entanto, é difícil de calcular o número de pessoas com necessidades especiais, uma vez que 70% dos indivíduos portadores de deficiência têm uma idade superior a 65 anos [Soed93, User93]. Estes números reflectem o envelhecimento da população europeia, sendo possível indicar vários motivos para este fenómeno, tais como os progressos da medicina, a diminuição da taxa de natalidade, a melhoria das tecnologias de reabilitação e a própria melhoria geral das condições de vida.

O elevado número de pessoas com necessidades especiais é o factor fundamental que levou a Comissão Europeia a promover programas como o TIDE e o HORIZON, nas áreas da deficiência, com o objectivo de desenvolver as tecnologias necessárias para melhorar a qualidade de vida desta fracção da população.

2.2 A importância (relativa) da tecnologia

O objectivo final das tecnologias de apoio consiste em integrar (ou reintegrar) indivíduos com necessidades especiais na sociedade, aumentando a qualidade de vida e o conforto daqueles que não conseguem adaptar-se às soluções “normais” do dia-a-dia.

O desenvolvimento de tecnologias de apoio não abrange todos os tipos de deficiência, concentrando-se em campos bem determinados, uma vez que as diminuições de capacidade são específicas. Assim, encontramos grupos com interesse nas seguintes áreas [Sche93]:

- Visual
- Auditivo
- Problemas de comunicação
- Membros inferiores (pés e pernas)
- Membros superiores (mãos e braços)
- Coordenação de movimentos
- Mental (cognitivo)

Não obstante os méritos próprios das tecnologias desenvolvidas em todas estas áreas, a sua importância é relativa, uma vez que não conseguem por si só resolver os problemas específicos de cada indivíduo. Na opinião do autor, o factor mais importante no processo de reabilitação é o estado psíquico do indivíduo, a sua força vontade e a tenacidade para ultrapassar os problemas que enfrenta. As ajudas técnicas tomam assim a forma de um “meio” para superar certas dificuldades e não de uma solução (“fim”) dos problemas, como se sugere através da ilustração apresentada na figura 2.1. As ajudas técnicas são desenvolvidas com o objectivo de serem fáceis de utilizar e adaptáveis aos seus utilizadores, mas a realidade é que todas elas implicam um período de adaptação e treino.

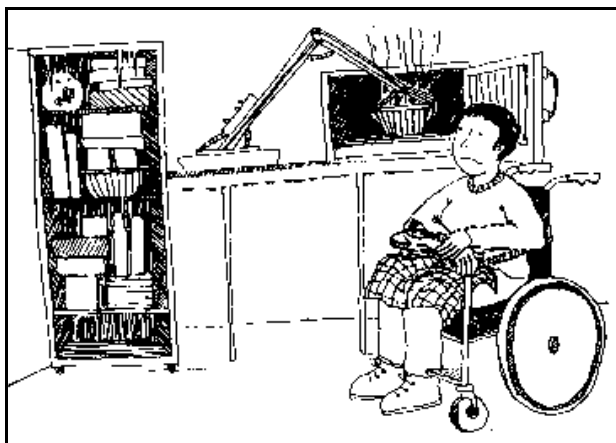


Figura 2.1: Excesso tecnológico que não resolve os problemas

2.3 Metodologia de desenvolvimento

A diferença fundamental entre o desenvolvimento de tecnologias de apoio, face ao desenvolvimento de outras tecnologias, consiste na necessidade de uma forte participação dos utilizadores finais durante todo o processo, por forma a encontrar soluções correctas para as necessidades efectivas dos indivíduos em causa. Todo o trabalho de desenvolvimento é centrado nos requisitos apresentados pelos utilizadores e não por condicionalismos tecnológicos ou de mercado [Teix95], para evitar o desencontro que frequentemente se representa como ilustrado na figura 2.2.

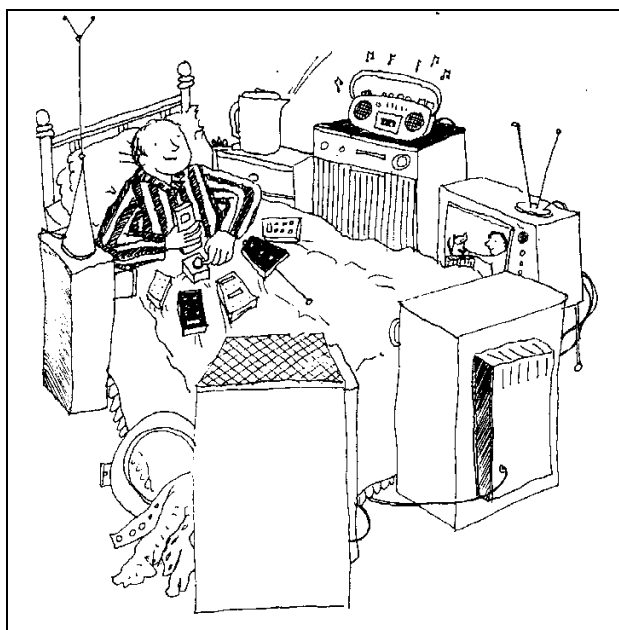


Figura 2.2: Tecnologia inadequada às necessidades do utilizador

A sequência de etapas que levam ao desenvolvimento de uma solução podem ser resumidas da seguinte forma:

- Identificação dos requisitos do utilizador
- Elaboração da especificação técnica
- Desenvolvimento das soluções tecnológicas necessárias
- Avaliação das soluções por parte dos utilizadores
- Reabilitação dos utilizadores, que implica adaptação e treino com estas novas soluções
- Avaliação final dos resultados obtidos.

Existem metodologias para a fase de identificação de requisitos e para a elaboração da especificação técnica, podendo citar-se o exemplo do UserFit, que pretende sistematizar todo o processo de concepção de um produto [User93]. Segundo o UserFit, os utilizadores finais não são apenas as pessoas com necessidades especiais, mas sim todas as pessoas que interagem com os utilizadores do produto, tais como a família, terapeutas, enfermeiros, etc. O objectivo final de qualquer metodologia é obter soluções de apoio que maximizem a eficácia, eficiência, segurança e o conforto.

Na procura destes objectivos encontramos várias iniciativas, entre as quais podemos referir as seguintes: EASY, W3C WAI, TIDE UserFit, Apple Acessability Initiative, etc. Todas elas fornecem orientações para o desenho de produtos com interfaces amigáveis com o utilizador, devendo o termo “interface” ser entendido no sentido mais abrangente da palavra, incluindo os mecanismos físicos de manipulação de um objecto [Apple].

É importante realçar que as metodologias referidas não se aplicam apenas aos produtos para pessoas com necessidades especiais, mas sim a todos os tipos de produtos, na procura do “*design for all*”. Este objectivo tem no entanto vindo a ser substituído por outro mais realista, nomeadamente o “*design for more*”. Ao aumentar a população alvo de um produto, o efeito imediato é o aumento das probabilidades de lucro associadas à sua comercialização. Como exemplo de um produto concebido para uma população alargada, podemos citar um telefone com teclas grandes, com números grandes desenhados em alto contraste e com inscrições em Braille. Este produto tanto poderia ser utilizado por indivíduos normais como por indivíduos com dificuldades de coordenação motora, visão limitada, ou mesmo nula.

2.4 Estado actual das soluções disponíveis comercialmente

Efectuada uma introdução sumária ao vasto domínio das tecnologias de apoio, passaremos de seguida a abordar os sistemas domóticos, que podem desempenhar um papel de grande relevância na autonomia em ambiente doméstico.

2.4.1 Sistemas Domóticos

Os sistemas domóticos constituem a base da tecnologia para as casas inteligentes, onde os sub-sistemas de um edifício se encontram interligados por uma infra-estrutura de comunicação que lhes permite trocarem informações, com o objectivo final de tornar o edifício mais eficiente, confortável e seguro [Ekbe95, Coop95, Nune96, Cost98].

Existem várias soluções disponíveis no mercado, algumas do tipo aberto e outras totalmente proprietárias. Optar por soluções normalizadas melhora as perspectivas de integração, ao mesmo tempo que liberta os compradores da dependência de um único fabricante, o que constitui uma vantagem importante no domínio em que o trabalho realizado se insere.

X10

Foi lançado em 1978 pela empresa Pico Electronics, da Escócia, tendo-se tornado muito popular nos EUA. É um sistema de baixo custo, capaz de controlar uma grande variedade de equipamentos domésticos. A infra-estrutura de comunicação é a rede de distribuição de energia eléctrica, tendo a capacidade de trocar uma mensagem por segundo. O número de componentes que podem ser interligados está limitado a 64 [X10].



Figura 2.3: Kit X10 para controlo por computador

A figura 2.3 ilustra alguns componentes X10. A simplicidade de instalação e configuração de sistemas baseados nesta tecnologia tornou o X10 muito apetecido para pequenas aplicações, onde os próprios utilizadores realizam a totalidade das operações envolvidas (a venda de Kits X10 “do-it-yourself” é prática usual nos EUA).

CEBus

O Consumer Electronics Bus (CEBus) surgiu em 1984, nos EUA, promovido pela Electronic Industries Association [CEBus]. A infra-estrutura de comunicação pode ser implementada através da rede de distribuição de energia eléctrica, cabo em par entrançado, cabo coaxial, infra-vermelhos, rádio frequência ou fibra óptica. A capacidade de troca de informação usando

este barramento é de 7500 bps, ilustrando-se na figura 2.4 uma representação geral de um sistema CEBus baseado na rede telefónica.

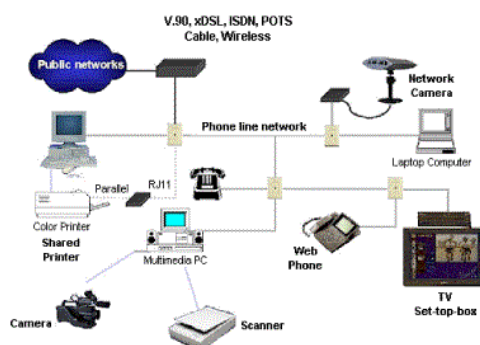


Figura 2.4: CEBus sobre a rede telefónica

O sucesso do CEBus está relacionado com a implementação de circuitos integrados que implementem os mecanismos necessários à comunicação, facilitando assim o desenvolvimento de aplicações baseadas neste sistema.

LonWorks

O LonWorks surgiu em 1990 [Eche], proposto pela Echelon Corporation, constituindo um sistema avançado de comunicação, baseado no circuito integrado NeuronChip, que se ilustra na figura 2.5. Este integrado é composto por três microprocessadores com funcionalidades bem definidas, dois dos quais são responsáveis pelas comunicações, implementando a totalidade das funcionalidades do protocolo, e sendo o terceiro responsável pela execução da aplicação de controlo. A integração da aplicação e da comunicação num único dispositivo físico permite reduzir as dimensões, o peso e o custo das soluções baseadas nesta tecnologia.

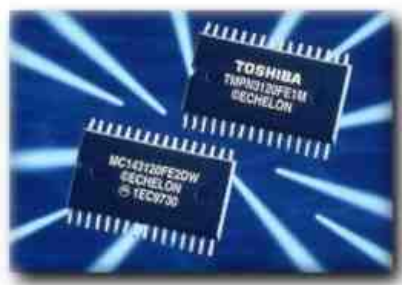


Figura 2.5: NeuronChip

A infra-estrutura de comunicação LonWorks pode ser implementada através da rede de distribuição de energia eléctrica, cabo em par entrançado, cabo coaxial, infravermelhos, rádio frequência ou fibra óptica. A implementação com vários meios físicos de transmissão é útil no projecto de sistemas domóticos, onde o ambiente de funcionamento pode apresentar ruídos

eléctricos, grandes distâncias, campos magnéticos, impossibilidade de passagem de cabos, localizações variáveis, etc.

O NeuronChip permite uma taxa máxima de transmissão de dados de 1,25Mbps, mas a velocidade real depende do meio físico utilizado. As elevadas taxas de transmissão permitem a gestão de sistemas muito grandes, em que o número de entradas e saídas é elevado, e por isso geram muita informação, que tem de ser processada por forma a otimizar o processo.

Smart House

Foi apresentado em 1987 pela National Association of Home Builders, dos EUA. É um sistema centralizado vocacionado para o controlo de habitações, sendo uma tecnologia bastante proprietária, ao ponto de depender do distribuidor do equipamento, não existindo uma especificação para a infra-estrutura de comunicação propriamente dita.

O SmartHouse é um sistema aplicado em sistemas construídos de raiz, devido à carga de cablagem que necessita.

EHS

A especificação European Home Systems (EHS), também conhecida por Esprit Home Systems [Teix95], foi desenvolvida em 1980, com o apoio da Comunidade Europeia, para criar uma norma domótica [Sant98]. O sistema é estruturalmente semelhante ao Home Bus System do Japão (HBS), embora não sejam compatíveis. A velocidade de transmissão máxima é de 9600 bps para o caso do par entrançado e de 2400 bps para a rede de distribuição eléctrica. Podem no entanto ser utilizados outros meios físicos, tal como rádio frequência, infravermelhos e fibra óptica.

Os sistemas baseados em EHS são bastante complexos, apresentando-se como sistemas distribuídos com um elevado grau de inteligência por nó. Outras funcionalidades, como o “*Plug-and-Play*” também contribuem para tornar esta especificação complexa e pouco popular.

EIB

O European Installation Bus, foi introduzido pela Siemens em 1982. É um sistema aberto, promovido pela European Installation Bus Association (EIBA), que pretende garantir a compatibilidade dos produtos com a norma e fomentar a sua utilização em novas instalações eléctricas. A EIBA conta actualmente com mais de 80 fabricantes associados [Eiba].

Na figura 2.6 ilustram-se alguns módulos EIB disponíveis comercialmente. A infra-estrutura de comunicação usada pode ser o cabo de par entrançado ou a rede de distribuição de energia eléctrica (PowerNet EIB).



Figura 2.6: Módulos EIB

A facilidade de instalação dos componentes EIB, associada à disponibilidade de um grande leque de produtos e ferramentas de apoio ao projecto, levou a uma grande utilização deste sistema. A utilização de PowerNet EIB é particularmente importante no processo de reconversão de instalações eléctricas já existentes.

BatiBus

O grupo Schneider lançou em 1989 este protocolo, especialmente desenvolvido para edifícios que usam o barramento FieldBus [Bati]. Permitindo uma reconversão de sistemas puramente centralizados em sistemas semi-distribuídos, o BatiBus é suportado por uma infra-estrutura de comunicações implementada por cabos de par entrançado. Os módulos disponíveis comercialmente têm o aspecto ilustrado na figura 2.7.



Figura 2.7: Módulo BatiBus

CAN

O protocolo CAN foi inicialmente desenvolvido pela Bosch para a indústria automóvel, cujos requisitos são basicamente os seguintes: elevadas taxas de transmissão; imunidade aos ruídos eléctricos; tolerância a falhas mecânicas; simplicidade de instalação; robustez dos conectores e redução do número de cabos necessários [Cia].

O protocolo apenas especifica as primeiras três camadas do protocolo OSI, pelo que houve necessidade de especificar protocolos para as restantes camadas, por forma a garantir a abertura e inter-funcionamento de dispositivos baseados neste protocolo. A associação “CAN in Automation” tem vindo a apoiar o protocolo CANOPEN como norma para produtos baseados em CAN, mas outros protocolos proprietários estão também disponíveis no mercado, como forma de garantir tempos de desenvolvimento muito baixos, sendo o CANLAYER um bom exemplo.

A infra-estrutura de dados é primariamente desenvolvida para par entrançado, com velocidades de transmissão até 1 Mbps, mas outros meios físicos podem também ser usados, tal como o cabo coaxial ou a fibra óptica. Devido às altas taxas de transmissão, a utilização de infravermelhos e de rádio frequência é conseguida através de dispositivos *Bridge store-and-forward*, que realizam a filtragem e adaptação de velocidade entre os dois meios físicos.

2.4.2 Sistema de controlo ambiental

A expressão “controlo ambiental” é entendida, na área das tecnologias de apoio, como o conjunto de soluções que permitem a um utilizador controlar os dispositivos e sistemas que o rodeiam. Para este efeito, os fabricantes de sistemas domóticos disponibilizam comandos remotos por infravermelhos e rádio frequência, havendo outros ainda que disponibilizam aplicações para controlo através do computador [Blvd].

COMPANION

O Companion está ilustrado na figura 2.8 e é uma unidade portátil de controlo ambiental comercializada pela firma POSSUM Controls Limited [Poss]. Tem como característica mais interessante o painel de cristais líquidos, sobre o qual é implementado um esquema de selecção de menus por varrimento sequencial. A representação das operações de controlo é feita na forma textual e o número de quadros de selecção está limitado a sete écrans.



Figura 2.8: Possum Companion

A comunicação com o sistema domótico do edifício é realizada através de infravermelhos ou rádio frequência. A capacidade de aprendizagem de novos comandos, através da emulação dos controlos remotos originais, constitui uma função importante deste dispositivo. É possível, por exemplo, controlar um sistema de Hi-Fi, um vídeo-gravador e um televisor, através da aprendizagem de novos comandos, efectuada a partir dos controlos remotos originais destes equipamentos.

O Companion incorpora um botão de selecção com dimensões bastante aceitáveis, que permite a selecção fácil dos comandos. Em alternativa, pode ser ligado um interruptor (*switch*) externo de selecção.

POSSUM FREEWAY

Também da POSSUM, este dispositivo apresenta-se como se ilustra na figura 2.9 e constitui uma versão muito simples de um sistema de controlo ambiental, em que o número de funções está limitado a 15. Apresenta como vantagem em relação ao Companion o facto de os comandos estarem representados por ícones e existir um som de retorno, sob a forma de voz, que assinala as selecções realizadas.



Figura 2.9: Possum FreeWay

O Possum Freeway não incorpora um botão de selecção, pelo que se torna necessário o uso de uma unidade externa para este fim. Como método alternativo de selecção, a unidade é capaz de reconhecer até oito comandos, a partir de palavras ou frases simples.

Director and Scanning Director

Estes dispositivos estão representados na figura 2.10 e são comercializados pela firma Prentke Romich, permitindo o controlo de Hi-Fi, vídeo-gravadores, televisores e qualquer outro dispositivo que possa ser comandado através de infravermelhos.

O Director é um módulo que pode ser adicionado ao Liberator, dispositivo de selecção por quadros de que falaremos mais adiante, na secção da comunicação aumentativa e alternativa.

O Scanning Director é um dispositivo independente, que pode ser operado através de um *switch* externo do tipo simples, ou com selecção dupla. A vantagem principal deste dispositivo é o seu tamanho, que lhe permite facilmente ser instalado numa cadeira de rodas ou numa cama. A única forma de representação é no entanto a textual, permitindo uma opção de cada vez. Esta característica torna muito difícil a sua utilização por indivíduos com problemas visuais ou cognitivos, bem como por todos aqueles que têm dificuldades em manter o olhar fixo.

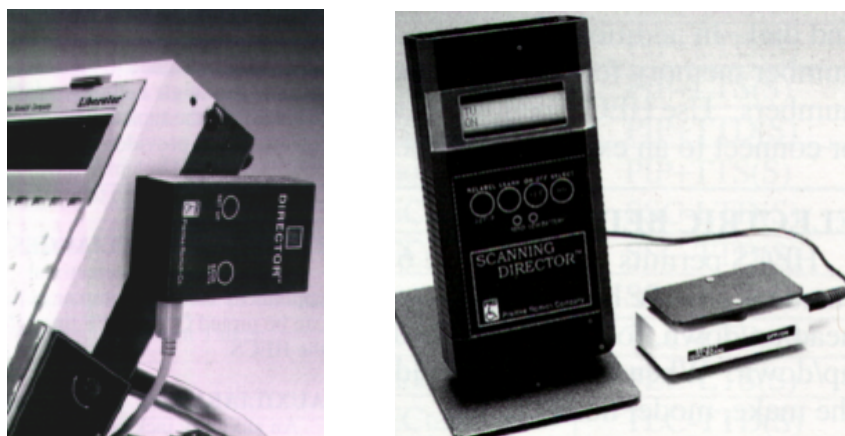


Figura 2.10: Director e Scanning Director

Cyberhouse Suite

O Cyberhouse Suite é um sistema de controlo ambiental baseado em computador, capaz de controlar uma instalação X10 [Smart]. Esta aplicação está representada na figura 2.11 e permite construir vistas da instalação ao gosto do utilizador, com animações que reflectem o estado dos dispositivos X10. É ainda possível realizar a gestão do edifício, uma vez que o sistema monitoriza permanentemente o estado da instalação, podendo tomar medidas correctivas, se necessário. No caso de uma emergência, o sistema pode enviar um FAX ou uma mensagem para um *Pager*.

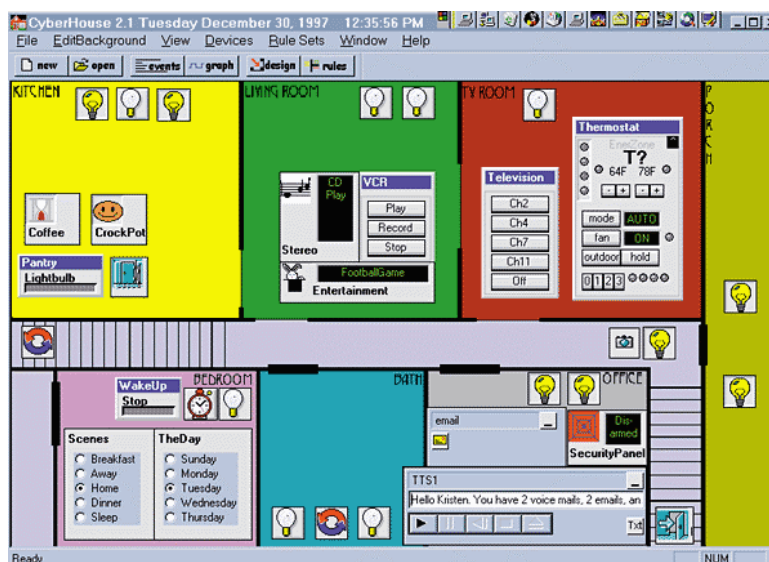


Figura 2.11: Interface gráfica do CyberHouse

2.4.3 Apoio à comunicação

Os sistemas de comunicação aumentativa e alternativa são usados para apoiar pessoas com necessidades especiais na área da comunicação, sendo o mais básico baseado em quadros onde as mensagens são representadas por imagens, como se ilustra na figura 2.12 [Tetz93, Tull94]. A selecção é feita através de vários métodos, que podem variar desde o dedo indicador até um capacete de selecção.

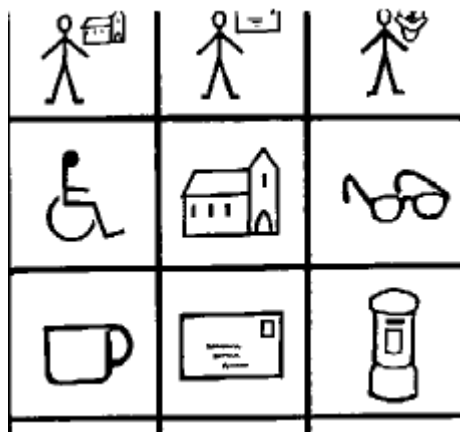


Figura 2.12: Pictogramas

Os sistemas do tipo electrónico permitem normalmente a emulação de quadros, com os quais é usado um sistema de selecção por varrimento. Os tipos de varrimento implementados podem ser do tipo linear, linha-coluna, coluna-linha e selecção por grupos, recorrendo-se a um *switch* adaptado às necessidades especiais do indivíduo em causa. A figura 2.13 ilustra alguns *switches* usados com frequência para este fim.

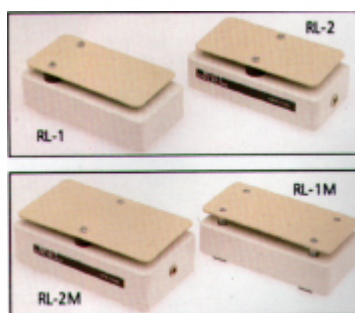


Figura 2.13: *Switches* simples e duplos

Existe ainda outro tipo de sistemas, que se apresentam mais compactos que os anteriores, por representarem as mensagens de forma numérica. Estes sistemas requerem no entanto capacidades cognitivas superiores às da representação por quadros de comunicação, uma vez que é necessário memorizar o número das mensagens. Os sistemas de varrimento para a representação

numérica são mais complexos, pois baseiam-se na construção de números a partir de um *switch*, quer realizando um varrimento linear, quer realizando a selecção dos dígitos das dezenas e das unidades.

O sistema mais complexo é no entanto o baseado em sintetizadores de voz, uma vez que requerem a construção de frases e a sua escrita na forma textual. Estes sistemas são, normalmente, baseados em teclados, ou na sua emulação através de um processo de varrimento. A sua grande vantagem consiste em não estarem limitados a um conjunto predefinido de mensagens.

Quadros de Comunicação clássicos

Estes quadros são normalmente construídos em madeira e contêm um número pré-estabelecido de imagens. A sua utilização primária dá-se na comunicação entre os utilizadores com necessidades especiais e os terapeutas ou família, uma vez que é necessário um conhecimento prévio do significado correcto das imagens.

Existem vários sistemas de imagens, entre os quais são de salientar os Pictogramas e o MinSpeak, ilustrados na figura 2.14, que definem conjuntos de imagens com significados bem definidos [Mins]. Existem vários tipos de complexidade associados às imagens, sendo possível encontrar imagens que representam uma palavra ou objecto e imagens que representam uma frase ou acção.



Figure 2.14: Pictogramas , Minspeak e Bliss

SpeakEasy

O SpeakEasy é um sistema de emulação de quadros de comunicação com uma saída de voz, que permite a comunicação em grupo através de doze pictogramas, acedidos através do teclado incorporado ou através de *switches* externos. Este produto está representado na figura 2.15 e permite gravar até dois minutos de mensagens, que podem ser repartidos da forma mais conveniente pelas doze mensagens possíveis.



Figura 2.15: SpeakEasy

Liberator

O Liberator é um dos produtos mais sofisticados para apoio a indivíduos com problemas na comunicação. Este produto é baseado na emulação de quadros de comunicação, com representação de mensagens através de pictogramas e acesso às mensagens através do teclado, ou de mecanismos de varrimento accionados por um *switch* comum.

O Liberator apresenta uma saída de voz sintetizada, em que o tipo de voz pode ser seleccionado por forma a adaptar-se ao sexo e idade do indivíduo. Outra funcionalidade interessante é a impressora incorporada no sistema, que permite imprimir textos inseridos pelo utilizador, ou então imprimir a tradução da sequência de pictogramas seleccionados. Ainda a nível da sequência de pictogramas seleccionados, existe um painel de cristais líquidos que confirma as selecções realizadas pelo utilizador. O Liberator incorpora ainda uma tecnologia chamada “Minserts”, que permite acelerar o processo de comunicação através da inserção de frases genéricas.

Este sistema é facilmente adaptado a uma cadeira de rodas, como se ilustra na figura 2.16, e permite também controlar instalações eléctricas baseadas no sistema X10, usando um módulo opcional chamado Director, que foi já referido na secção anterior.



Figura 2.16: Um utilizador que usa o Liberator

Como método de acesso alternativo, este sistema apresenta uma opção muito interessante, baseada num apontador por infravermelhos, que se ilustra na figura 2.17. Esta opção permite um acesso mais rápido e eficaz, quando comparada com os métodos de varrimento por *switch*. No entanto, a sua utilização requer capacidade de controlo estável dos movimentos da cabeça.



Figura 2.17: Uso do apontador por Infravermelhos no Liberator

HandiVoice

Este dispositivo está representado na figura 2.18 e constitui um sintetizador de voz, que representa as mensagens através de números. A indicação dos números pode ser realizada através do teclado, por um processo de varrimento com selecção, a partir um *switch* standard adaptado às necessidades do utilizador.



Figura 2.18: HandiVoice

2.4.4 Acesso a aplicações em ambiente PC ou MAC

Os produtos para acesso a aplicações de computador podem ser divididos em duas categorias principais: soluções baseadas em *hardware* e as que se baseiam em *software*.

As soluções baseadas em *hardware* permitem a emulação de um teclado convencional através de um mecanismo especial de selecção. Sob o ponto de vista do computador, existe apenas um teclado normal, o mesmo acontecendo no caso do dispositivo apontador, seja este um rato ou uma caneta óptica.

As soluções baseadas em *software* mostram uma grande dependência em relação ao sistema operativo, sendo frequente encontrarem-se produtos que funcionam muito bem no Windows 3.1 e não funcionam em versões posteriores do Windows. O mesmo acontece entre o Windows 95 e o Windows NT, devido às diferenças no acesso directo ao *hardware*, que não é possível de forma directa no Windows NT [Step97].

MOUSETRAP

“Simple is Better” é máxima personificada por este dispositivo, ilustrado na figura 2.19, que permite aos utilizadores com dificuldades de controlo dos membros superiores, nomeadamente tremuras e espasmos, a utilização de um rato de computador.



Figura 2.19: MouseTrap

Em analogia com a filosofia de concepção do MouseTrap, existem teclados com um acessório que permite a escrita num teclado “normal”. Este dispositivo pode ser descrito como uma placa

de composto plástico com orifícios nas posições das teclas, estabilizando assim a posição dos dedos dos utilizadores.

WiViK 2

Esta aplicação, ilustrada na figura 2.20, permite a emulação do teclado usando a técnica do “On-Screen Keyboard”. O funcionamento deste teclado virtual é realizado através de um dispositivo apontador, como por exemplo o rato, o *joystick*, o *trackball*, o *touchscreen* e as canetas ópticas. Para acelerar o processo de escrita estão disponíveis dois mecanismos avançados: a predição de palavras e a expansão de abreviaturas. Em opção existe um módulo para realizar varrimento e selecção através de um *switch* externo.

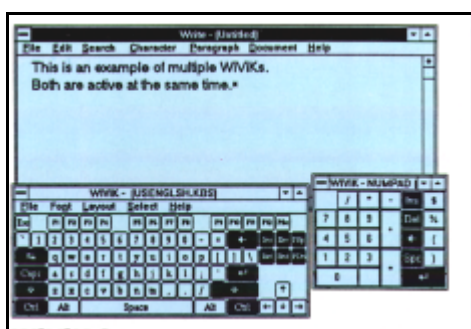


Figura 2.20: WiViK2

TTAM

O TTAM funciona como um acessório do Liberator, para emular por *hardware* o teclado e o rato do computador. Este dispositivo está ilustrado na figura 2.21 e é compatível com todas as máquinas Macintosh e “IBM AT compatíveis”.



Figura 2.21: Emulação de teclado com o TTAM

Concluída a apresentação de um conjunto de questões principais relacionadas com as tecnologias de apoio e com a sua área de aplicação, estão agora reunidas as condições para passarmos à descrição da metodologia seguida para identificar as necessidades específicas dos principais utilizadores do sistema desenvolvido: as pessoas com paralisia cerebral.

Capítulo 3

Requisitos do utilizador e especificação do sistema

Este capítulo apresenta os requisitos que foram identificados para os utilizadores principais do sistema desenvolvido, a que se segue uma pequena introdução à paralisia cerebral e ao seu impacto sobre a vida dos utilizadores. O capítulo conclui-se com uma apresentação da especificação do sistema nas suas vertentes principais: controlo ambiental, apoio à comunicação oral, apoio à comunicação telefónica, acesso a aplicações em ambiente Windows (PC) e características funcionais do interface com o utilizador.

3.1 Identificação dos requisitos dos utilizadores

A identificação dos requisitos do utilizador foi efectuada através de um processo interactivo no qual participaram todas as partes envolvidas, nomeadamente os utilizadores finais, os profissionais de reabilitação e os responsáveis pelo desenvolvimento do sistema.

Como o grupo de investigação onde este projecto foi desenvolvido já possuía experiência de desenvolvimento com os mesmos utilizadores, nomeadamente através do projecto HS-ADEPT, o ponto de partida foram os resultados obtidos num inquérito realizado previamente. Por outro lado, e como a orientação do trabalho foi no sentido de evoluir o sistema previamente desenvolvido no projecto HS-ADEPT, houve uma preocupação de verificar quais foram as funcionalidades então implementadas, que efectivamente respondiam aos requisitos dos utilizadores, e quais as que poderiam ser melhoradas. Por fim, a interacção dos utilizadores com aquele sistema suscitou novos requisitos, que inicialmente não tinham sido identificados.

Os principais requisitos podem, deste modo, ser agrupados pelas quatro vertentes específicas do sistema, convergindo num conjunto de necessidades globais que é representado pela especificação da interface com o utilizador. Para compreender melhor estes requisitos é necessário

fazer uma breve introdução à paralisia cerebral e ao seu impacto na vida diária das pessoas com este tipo de deficiência.

3.2 Paralisia Cerebral

A paralisia cerebral resulta de um dano (cerebral) que leva à inabilidade, dificuldade ou descontrolo de certos músculos e movimentos do corpo. O termo *Cerebral* refere-se à parte do corpo atingida, nomeadamente o cérebro (Sistema Nervoso Central – SNC), e a palavra *Paralisia* refere-se ao resultado do dano no SNC [Defnet].

É importante notar que a paralisia cerebral não é uma doença, mas sim uma condição médica especial que afecta as crianças, ocorrendo antes ou durante o parto, ou mesmo em consequência de certos problemas, que poderão ocorrer até os nove anos de idade, em resultado de falta de oxigenação no cérebro. Na maioria dos casos não é possível determinar com exactidão a causa das lesões, no entanto as seguintes situações podem potenciar o surgimento desta condição médica [Defnet, CMC].

- Durante a gravidez
 - Toxemia gravídica (acumulação de toxinas no sangue)
 - Anemia Grave
 - Infecções renais e urinárias graves
 - Hemorragias
 - Diabetes
- Problemas durante e logo após o parto:
 - Obstruções pélvicas
 - Anóxia ou Hipóxia Neonatal (problemas de oxigenação)
 - Distúrbios circulatórios cerebrais
 - Traumatismos no parto
 - Rubéola
 - Infecções pré-natais e peri-natais
 - Nascimento prematuro
 - Icterícia neonatal grave (incompatibilidade sanguínea)
- Problemas até os nove anos de idade:
 - Asfixia
 - Fracturas ou feridas penetrantes na cabeça
 - Infecções no Sistema Nervoso Central
 - Meningite

Não existem dois casos semelhantes de crianças com paralisia cerebral, já que algumas apresentam perturbações subtis, quase imperceptíveis, aparentando serem desajeitadas ao caminhar, ao falar e ao manipular objectos, enquanto as que sofreram lesões cerebrais mais graves, como

por exemplo a anóxia neonatal, podem apresentar incapacidade motora acentuada e impossibilidade de comunicação oral.

Dependendo da localização das lesões e das áreas do cérebro que foram afectadas (ver figura 3.1), as manifestações podem ser diferentes. Na paralisia cerebral há uma confusão de mensagens entre o cérebro e os músculos, existindo três tipos mais comuns e um quarto que resulta da combinação dos outros [CMC], nomeadamente:

- **Espástica (*Spastic*):** Caracteriza-se por uma paralisia na tonicidade dos músculos, resultante de lesões a nível do Córtex. Dá-se o nome de hemiparesia quando um lado do corpo é afectado, diplegia quando se trata dos membros inferiores e tetraplegia quando se trata dos membros inferiores e superiores.
- **Atetóide (*Athetoid*):** Caracteriza-se por uma variação da tonicidade muscular e pela existência de movimentos involuntários, afectando o sistema extrapiramidal, resultantes de lesões nos gânglios da base.
- **Atáxica (*Ataxic*):** Caracteriza-se por uma diminuição da tonicidade muscular, dificuldade de equilíbrio e uma descoordenação de movimentos, podendo ainda haver tremura nos movimentos das mãos e problemas na fala.

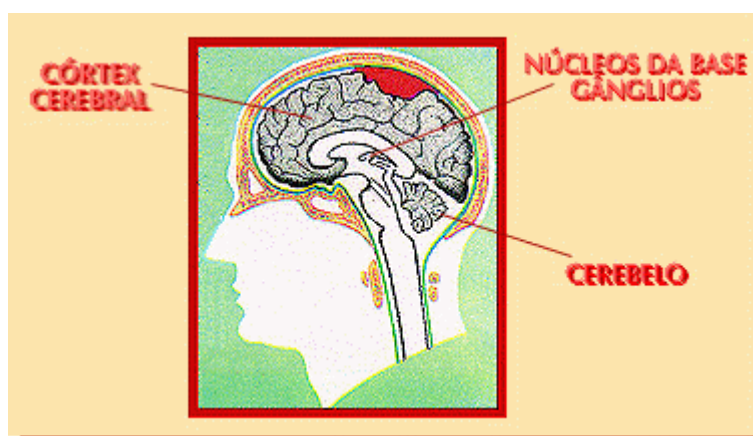


Figura 3.1: Cérebro humano

As crianças com paralisia cerebral apresentam normalmente outros problemas de saúde que não estão directamente relacionados com as lesões cerebrais, nomeadamente:

- **Epilepsia:** é comum ocorrerem convulsões ou crises epiléticas, de maior ou menor intensidade e dentro de variadas formas de manifestação neurológica, sendo mais comuns no período pré-escolar, estando associadas ao prognóstico e à evolução de outros problemas de saúde.
- **Deficiência Mental:** com uma ocorrência de aproximadamente 50% dos casos [Defnet], tem levado a distorções e preconceitos acerca do potencial intelectual das crianças com paralisia cerebral.

- **Problemas Visuais:** ocorrem casos de baixa-visão, estrabismos e erros de refração, que podem ser precocemente diagnosticados e tratados.
- **Dificuldades de Aprendizagem:** As crianças com paralisia cerebral podem apresentar alguns problemas de aprendizagem, pelo que é necessário utilizar métodos de ensino adaptados.
- **Dificuldades na fala e alimentação:** devido à falta de coordenação dos músculos, as crianças apresentam dificuldades na comunicação verbal e dificuldades na alimentação.
- **Outros problemas:** dificuldades auditivas, disartria, défice sensorial, escoliose, espasmos musculares, problemas odontológicos, salivação incontrolável. Além destes problemas de saúde, podemos encontrar problemas ao nível do comportamento emocional e social.

É importante referir que não existem medicamentos ou operações que possam curar a paralisia cerebral, embora existam diversas e inovadoras possibilidades de melhorar e minimizar os seus efeitos, nomeadamente através da terapia de reabilitação e pelo auxílio de ajudas técnicas. Este processo é, no entanto, moroso e geralmente significa a transferência da dependência de uma pessoa (assistente, terapeuta, enfermeira, etc.) para um dado recurso tecnológico. De entre os vários tipos de terapia podemos destacar as seguintes:

- **Fisioterapia:** Permite às crianças aprender a movimentar-se melhor e a manter o equilíbrio. De um ponto de vista prático, esta terapia pretende ajudar as crianças a caminhar melhor, a conseguirem dominar a cadeira de rodas, e a subirem e descenderem escadas de uma forma segura. Além disso, permite exercitar conjuntos de músculos de uma forma correcta.
- **Terapia da fala:** Tem como objectivo aumentar as capacidades de comunicação das crianças, quer através da linguagem verbal, quer através de linguagens de comunicação alternativa, tais como a linguagem gestual e a linguagem simbólica. O uso de ajudas técnicas é fundamental para a interacção com pessoas que desconheçam as linguagens alternativas.
- **Terapia ocupacional:** Tem como objectivo ensinar as crianças a usarem melhor os seus braços, mãos e corpo, bem como a realizarem tarefas como escrever, desenhar, cortar papel, bordar, lavar os dentes, vestir-se, comer, etc. É também tarefa dos terapeutas ocupacionais encontrarem as ajudas técnicas que melhor se adaptam à realização das tarefas do quotidiano.
- **Terapia recreativa:** Tem como objectivo criar actividades recreativas e lúdicas, tais como a dança, desporto, arte, floricultura, etc.

Os efeitos da paralisia cerebral fazem-se sentir a vários níveis, desde o pessoal ao social, nomeadamente pelas dificuldades sentidas na realização de tarefas do quotidiano, como andar, falar, comer, tratar da higiene pessoal, etc. A repercussão que estas limitações têm a nível social leva a dificuldades de integração no mundo do trabalho e nas relações interpessoais.

Concluída esta caracterização sumária da paralisia cerebral, estamos em condições de justificar a especificação funcional que orientou o desenvolvimento do trabalho descrito nesta dissertação.

3.3 Especificação para o controlo ambiental

O sistema de controlo ambiental constitui uma extensão de um sistema convencional de controlo, pelo que os comandos e funcionalidades tradicionais devem ser mantidos, acrescentando-se formas alternativas de accionamento, nomeadamente através de comandos remotos por infra vermelhos e rádio frequência. A utilização de dispositivos adaptados às necessidades especiais dos utilizadores, tal como o Possum Companion, e interfaces de controlo por computador, devem ser suportados pelo sistema.

A utilização de accionamentos automáticos deve ser evitada, a não ser que se trate de um requisito especial para um determinado indivíduo. Este requisito pretende dar ao utilizador uma sensação de domínio do sistema, em antagonismo à sensação de ser dominado pela tecnologia. Em termos ergonómicos, os comandos devem ter um tamanho generoso e não deve ser necessária muita força para realizar o seu accionamento. A disposição dos comandos pela casa deve levar em conta a coexistência de utilizadores com várias formas de locomoção, não podendo os utilizadores em cadeira de rodas ser esquecidos, o que condiciona a altura à qual se colocam os comandos. Uma solução possível para este problema é a colocação de dois comandos a alturas diferentes para accionar o mesmo dispositivo. Os comandos devem ser facilmente acessíveis, pelo que a sua localização deve ser cuidadosamente escolhida, preferencialmente em espaços livres de obstáculos, sempre com o objectivo de tornar o sistema mais funcional, levando em conta uma vez mais que a locomoção em cadeira de rodas é difícil e limitada.

A resposta ao accionamento de um comando deve ser imediata, não deixando quaisquer dúvidas ao utilizador sobre o sucesso da acção, sendo desejável um mecanismo de realimentação visual ou sonora que indique o início e o fim da acção. Deve, no entanto, haver alguma histerese no accionamento consecutivo de um comando do tipo “toggle”¹, uma vez que estes utilizadores apresentam falta de coordenação motora e tremuras, o que levaria em caso contrário a dificuldades em controlar o estado dos dispositivos.

Um sistema de controlo ambiental pode apresentar-se como uma entidade flexível, permitindo a reconfiguração funcional da instalação eléctrica, por forma a permitir alterações a nível da utilização das divisões de uma casa (por exemplo, um quarto tem necessidades de controlo diferentes das apresentadas por um corredor ou por uma sala). Esta flexibilidade é um factor importante a nível de expansão, pois permitirá no futuro a integração de novos dispositivos e sistemas, que complementarão os que já se encontram instalados.

¹ Os comandos do tipo *toggle* comutam o estado dos dispositivos controlados (acende/apaga, liga/desliga, etc).

Os sistemas de controlo ambiental devem ainda ser do tipo distribuído, devido às vantagens apresentadas por este tipo de solução, nomeadamente pela sua fácil instalação, flexibilidade e fiabilidade. A cablagem necessária para implementar sistemas distribuídos é substancialmente menor do que a utilizada em sistemas centralizados, uma vez que o nó inteligente de controlo pode ser colocado junto do dispositivo a controlar, realizando todas as comunicações com os outros nós do sistema através de um barramento de dados único. Outra vantagem, em termos de cablagem, é a distribuição da corrente eléctrica para os nós através do barramento utilizado para os dados. Os sistemas distribuídos podem utilizar a rede de distribuição eléctrica para implementar a comunicação entre os nós, sendo esta a solução preferencial para a reconversão de instalações eléctricas tradicionais em sistemas domóticos, embora esta alternativa não seja a melhor em termos de desempenho. No que respeita à fiabilidade, os sistemas distribuídos apresentam como ponto crítico o barramento de dados, sem o qual não podem comunicar com os outros nós. No entanto, uma falha num dispositivo controlador apenas desactiva a função por ele realizada.

A opção por sistemas de controlo ambiental centralizados não é recomendável porque estes sistemas apresentam topologias em estrela, necessitando por isso de um quadro eléctrico de onde partem condutores para todos os sensores e actuadores instalados no edifício, o que representa um elevado número de condutores. Em termos de fiabilidade, estes sistemas são os mais críticos, uma vez que tudo depende de um único dispositivo de controlo.

A rede de comunicação de dados que interliga todos os dispositivos de um sistema distribuído serve para difundir comandos de accionamento e informações sobre o estado da instalação. É por isso vital que esta infra-estrutura apresente uma elevada fiabilidade, já que a sua falha compromete o funcionamento da instalação. Para garantir esta fiabilidade, a rede deve ser tolerante às seguintes falhas:

- Curto circuitos envolvendo linhas de dados
- Circuito aberto em linhas de dados
- Ruídos electromagnéticos
- Falhas nos transmissores (*transceivers*) de acesso à rede de dados
- Falha nos controladores de rede

Um nó da rede no qual ocorrer uma falha irrecuperável deve passar ao estado inactivo, por forma a não perturbar o funcionamento dos outros nós. Por outro lado, a passagem de um nó ao estado inactivo deve ser detectada através uma de estação de monitorização da rede, que disparará um alarme indicativo da falha.

No caso de falha de energia eléctrica, o comportamento de todos os dispositivos controladores deve ser conhecido e determinado pelas necessidades de segurança dos utilizadores. Por exemplo, em caso de um incêndio, se houver uma falha de energia eléctrica, todas as portas devem ficar abertas, por forma a facilitar a evacuação do edifício. Este princípio vai contra as regras

de combate aos incêndios, que determinam que todas as portas devem ficar fechadas por forma a conter a propagação do fogo e a evitar a sua alimentação com oxigénio, mas no caso de indivíduos com dificuldades motoras ou em cadeiras de rodas, estas portas fechadas tornam-se fatais.

A implementação dos nós de controlo da rede deve ser baseada numa única carta de circuito impresso de dimensões reduzidas, para facilitar o processo de fabrico e para tornar o produto economicamente viável, através do fabrico em maiores quantidades.

3.4 Especificação para a comunicação oral

A comunicação oral assistida é um requisito importante para o relacionamento entre pessoas com necessidades especiais e os indivíduos à sua volta, principalmente quando estes não têm formação em alguma linguagem de comunicação alternativa, como a gestual. Os sistemas de apoio à comunicação podem ser baseados em soluções generalizadas, como as cartas de som disponíveis para PC, ou em *hardware* dedicado, como é o caso dos sistemas da OKI e do sistema Director [Gall94].

Há ainda a possibilidade de recorrer a mensagens sintetizadas ou a mensagens pré-gravadas. As primeiras são uma solução flexível e genérica, mas não se adequam tão bem à terapia da fala, devido à falta de qualidade sonora das mensagens sintetizadas. Deste modo, a saída de som para a comunicação oral deve ser baseada no sistema normalizado para o mesmo fim nos PC e deve ainda facilitar a gravação e edição de mensagens, o que pode ser feito através do acessório Windows “*Sound Recorder*”. Recorde-se que a aplicação do sistema é a reprodução de voz, pelo que as mensagens devem ser gravadas com o CODEC GSM 11.025 KHz MONO, por forma a minimizar o espaço em disco necessário ao armazenamento das mensagens. Em alternativa, o uso do CODEC ADPCM 11.025 KHz 4 bit MONO pode ser considerado, mas o espaço em disco utilizado será três vezes maior, para uma qualidade de som semelhante.

O sistema de comunicação oral necessita de uma interface gráfica que represente as mensagens de uma forma intuitiva, compatível com os sistemas apresentados nas sessões de terapia da fala, nomeadamente os quadros de comunicação com figuras dos sistemas internacionais, iconografia ou fotografias.

Estes quadros têm que ser facilmente construídos e personalizados para o utilizador em causa e para os contextos em que se pretende realizar a comunicação, podendo por exemplo existir quadros para a alimentação, o afecto, as necessidades, os jogos, etc. [Gard95, Tull94]. Os quadros devem estar organizados hierarquicamente, por forma a facilitar a navegação entre eles e a construção de frases. Devem ainda ser considerados como entidades em constante evolução, quer pelo incremento na perícia de navegação e controlo do utilizador, quer pela seu progresso, no que respeita à capacidade de construir mensagens com maior complexidade.

3.5 Especificação para a comunicação telefónica

As dificuldades de comunicação, aliadas à falta de coordenação motora, tornam muito difícil a utilização de um telefone convencional. Surge assim a necessidade de se encontrar uma solução para facilitar a marcação de números telefónicos, que em combinação com o sistema de apoio à comunicação oral, permita dialogar com um interlocutor remoto.

Uma solução possível para este problema consiste em desenvolver um telefone inteligente, capaz de receber comandos de uma porta RS-232, e dos circuitos que permitam o seu acoplamento a um sistema de comunicação oral assistida. Esta solução, no entanto, implicaria a produção deste novo dispositivo, associando assim novas dificuldades, provenientes de condicionamentos de mercado e de legalização perante as instituições competentes (neste caso, o ICP).

A solução a adoptar deve portanto passar pela utilização de um Modem normalizado para realizar a marcação, passando depois a chamada para um telefone com a função *mãos-livres*. Esta solução torna-se mais vantajosa pelo facto de os utilizadores poderem reaproveitar dispositivos que já tenham adquirido, bem como por recorrer apenas a soluções facilmente encontradas no mercado, não havendo uma dependência directa de um determinado fabricante e conseguindo-se preços mais vantajosos que os praticados para as ajudas técnicas.

O sistema de comunicação telefónica deve ser facilmente expansível às novas necessidades de um utilizador ou de grupos de utilizadores, apresentando-se sob a forma de uma interface com elementos organizados de maneira hierárquica, de modo a facilitar a sua utilização. A integração deste sistema com as funções de comunicação oral assistida é fundamental, uma vez que no momento em que se estabelece a chamada telefónica há necessidade de iniciar o diálogo com a pessoa que se encontra no outro extremo da linha. Assim, nos quadros de apoio ao estabelecimento de chamadas telefónicas, deve haver pelo menos um botão de apoio à comunicação oral, com a seguinte mensagem: “ Por favor tenha paciência, sou um utilizador com necessidades especiais e utilizo um dispositivo para apoio à comunicação”. Esta mensagem informa imediatamente o interlocutor do tipo de diálogo que terá lugar e evita assim que este desligue, perante a existência de atrasos na comunicação.

3.6 Especificação para o acesso a aplicações Windows

Permitir o acesso de utilizadores com necessidades especiais ao ambiente gráfico do sistema operativo Microsoft Windows não é tarefa fácil, especialmente quando se trata de indivíduos com dificuldades de coordenação motora, em relação aos quais os dispositivos como teclados e ratos convencionais se revelam completamente inadequados. Torna-se assim necessário criar um mecanismo que suporte as mesmas funcionalidades de controlo, mas agora através de dispositivos externos de emulação de teclado e rato, como por exemplo o Liberator, ou então por recurso a programas de computador que efectuem a síntese dos eventos gerados por aqueles

dispositivos de entrada tradicionais. Estes programas designam-se geralmente por “*On-screen-emulators*” e, considerando um sistema com uma interface gráfica unificada, revelam-se os mais práticos, já que não necessitam de nenhum dispositivo de *hardware* adicional, usando para a selecção o mesmo *switch* utilizado para as restantes funções de interface [Wai].

A primeira função a implementar consiste no lançamento de aplicações através de um quadro de selecção, onde as aplicações aparecem representadas por uma imagem, um título e uma mensagem oral que as identifica. Estes quadros devem estar integrados na interface geral com o utilizador, trabalhando da mesma forma e mantendo o mesmo “*look & feel*”.

Após o lançamento das aplicações, é necessário que a emulação de rato seja lançada automaticamente. Esta operação deve ser realizada através de uma régua com botões seleccionáveis por um processo de varrimento linear, em que a escolha dos botões é efectuada através do mesmo *switch* da interface de selecção por quadros. A versão mínima da régua deve ser constituída por seis botões, que implementam as seguintes funções:

- Voltar à interface de quadros de selecção
- Escolher a direcção de movimentação do rato
- Movimentar o rato na direcção seleccionada
- Seleccionar o botão esquerdo
- Efectuar a selecção dupla no botão esquerdo
- Comutar para o modo de emulação de teclado

Pode, no entanto, haver ainda necessidade da emulação de outras funções, nomeadamente para trabalhar com programas de desenho, como por exemplo:

- Premir / Largar o botão esquerdo, para desenhar linhas e efectuar funções de *Drag & Drop* (arrastar) de ícones.
- Premir / Largar o botão direito, para desenhar com a cor de fundo e aceder às funções de atalho disponibilizadas pelas aplicações do *MS-Office*.

O processo de varrimento da régua de emulação de teclado deve apresentar uma velocidade igual à seleccionada para a interface dos quadros de selecção, mas a velocidade do movimento do apontador sobre o écran deve ser adaptada ao utilizador, possivelmente apresentando um valor proporcional à do processo de varrimento dos botões. O varrimento linear deve incluir um tempo de repouso sempre que passa pela posição inicial, que deverá ser superior ao tempo normalmente disponível para a selecção, de modo a que o utilizador consiga focar de novo o botão seleccionado, após um salto de posição.

A emulação de teclado pode ser realizada através da simulação do funcionamento e disposição de um teclado convencional sobre o écran, embora esta solução requeira maiores capacidades cognitivas para trabalhar com os múltiplos conjuntos de caracteres seleccionáveis a partir de combinações das teclas SHIFT e ALT Graphics. A escolha para a implementação da síntese de

eventos de teclado consistiu em realizar três quadros de selecção com os seguintes conjuntos de símbolos:

- Maiúsculas, Espaço, *Carriage Return* e Ponto.
- Minúsculas, Espaço, *Carriage Return*, Vírgula e Ponto.
- Números e símbolos.

Deve estar presente um botão de selecção que permita a comutação entre o teclado e a régua de emulação de rato, de forma a facilitar o trabalho com as aplicações Windows, que na prática nunca necessitam de uma intervenção simultânea do teclado e do rato.

O processo de varrimento utilizado será igual ao seleccionado para a interface de selecção por quadros, mantendo também a mesma velocidade. Tal como foi já referido para a régua de emulação de teclado, sempre que o processo de varrimento passar pela posição inicial, deve realizar uma pausa. Também é importante que o varrimento volte à posição inicial sempre que é realizada uma selecção.

3.7 Especificação funcional do interface com o utilizador

A interface com o utilizador é o ponto chave de todo o trabalho, uma vez que a sua facilidade de utilização e capacidade de adaptação serão fundamentais para a qualidade e eficácia da ajuda técnica desenvolvida, favorecendo ou dificultando a total utilização dos recursos disponibilizados.

De modo a facilitar a aprendizagem do sistema, os ícones e outros elementos gráficos devem estar directamente relacionados com os dispositivos que os utilizadores já possuem para realizar a comunicação assistida, expandindo as suas funções no sentido de suportar o controlo ambiental, o apoio à marcação telefónica e o controlo de aplicações Windows.

A filosofia eleita recomenda que se organize a interface em quadros de selecção de objectos, representados através de uma imagem (Pictograma, MinSpeak, Fotografia, Desenho), um título e uma mensagem de voz pré-gravada, indicadora da função implementada. Estes quadros podem ser ainda organizados de uma forma hierárquica através de uma identidade do tipo menu, que contém um novo quadro de selecção. A existência de múltiplos quadros de selecção, organizados hierarquicamente, será possível se existirem entidades que permitam ao utilizador passar de uns para os outros de forma simples. Esta necessidade dá origem a três tipos de botões, com as seguintes funções: “Raiz”, que permite ao utilizador saltar para o quadro de selecção principal, “Pai”, que permite saltar para o quadro anterior, e “SubMenu”, que permite saltar para um novo quadro de selecção.

A capacidade de adaptação da interface contempla duas vertentes principais: a adaptação física do indivíduo, no que diz respeito ao suporte dos dispositivos de selecção adaptados, e a adaptação visual e funcional da interface. O suporte físico dos dispositivos de selecção deve ser reali-

zado através da porta paralela do computador, uma vez que a porta série será utilizada para as comunicações com a infra-estrutura de controlo ambiental

A leitura de sinais eléctricos na porta paralela deve ser feita forma a facilitar a passagem aos novos sistemas operativos Windows 9x e Windows NT, já que a respectiva API² não permite o acesso directo aos controladores da porta paralela, sendo necessário realizar esta operação através da simulação de uma situação de erro (como por exemplo o “*Paper Out*” e o “*Line-Out*”). De forma geral, os dispositivos de selecção apresentam-se electricamente como interruptores, pelo que há necessidade de os adaptar às especificações dos pinos de entrada da porta paralela. Esta adaptação pode ser feita por simples recurso a uma resistência de “*pull-up*” entre o pino de entrada e V_{CC} , sendo o dispositivo de selecção ligado entre a massa e o pino de entrada (é usada esta configuração porque os pinos de entrada trabalham em lógica negativa).

Uma vez concluídos os aspectos relacionados com os requisitos dos utilizadores e a consequente especificação do sistema, será apresentada no próximo capítulo a arquitectura física (*hardware*) que lhe serve de suporte.

² *Application Program Interface*

Capítulo 4

Arquitectura física

Este capítulo apresenta os recursos físicos necessários à implementação do sistema desenvolvido, em cada uma das vertentes que o integram (comunicação oral e telefónica, controlo ambiental). Esta apresentação é no entanto precedida pela consideração do sistema na sua globalidade, de modo a facilitar a percepção do conjunto de recursos presentes e da sua interacção.

4.1 Apresentação global do sistema

A apresentação global do sistema desenvolvido pode efectuar-se como se ilustra na figura 4.1, onde o PC representa o elemento central e executa a aplicação de interface com todos os recursos existentes.

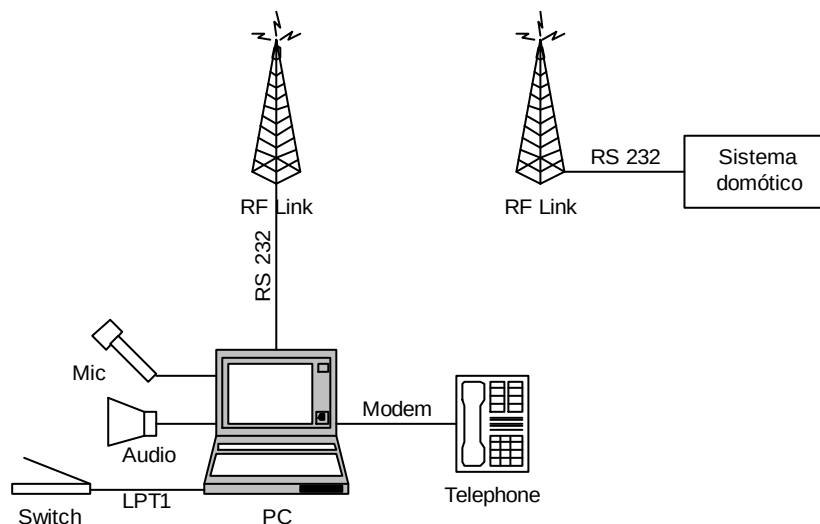


Figura 4.1: Esquema global do sistema

A mínima configuração de funcionamento requer no entanto apenas um computador (PC) com carta de som, assegurando nestas circunstâncias o serviço de comunicação oral assistida, sendo o dispositivo de selecção o botão direito do rato. Para um utilizador com necessidade de

um dispositivo de selecção especial é necessário adicionar uma interface na porta paralela, para acoplamento de um *switch* de selecção simples. Com este *switch* é possível igualmente realizar as funções de controlo de aplicações Windows através da emulação do teclado e do rato, sendo esta realizada através da sintetização dos eventos gerados pelos dispositivos em lados.

A função de chamada telefónica assistida necessita do acoplamento de um Modem normalizado para realizar a marcação dos números de telefone. Este Modem deve ter duas entradas do tipo RJ-11 e passar a chamada para o telefone após a marcação, devendo este ser escolhido em função das capacidades do utilizador (e.g. um telefone convencional ou um telefone com a função de mãos-livres).

A função de controlo ambiental depende de uma infra-estrutura domótica que permita o controlo dos seus dispositivos através do PC (via uma porta série RS-232), como sucede com os sistemas CAN, EIB, X10, etc. No caso de utilizadores em cadeiras de rodas, que pretendam controlar um edifício remotamente, é necessário incluir um par de transceptores RF, que substituem o cabo de ligação RS232.

4.2 Interface para ligação de um *switch* ao PC

A figura 4.2 ilustra um protótipo do adaptador de *switch* de selecção simples para um computador do tipo PC, compatível com uma porta paralela do tipo Centronics. O conector utilizado nesta interface é do tipo *jack* fêmea de 3.5 mm, uma vez que a generalidade dos dispositivos existentes no mercado utilizam uma ficha *jack* mono macho de 3.5 mm.



Figura 4.2: Adaptador de *switch* simples

A figura 4.3 descreve as funções associadas aos pinos da porta paralela, devendo a sua leitura obedecer às regras de acesso ao *hardware* dos sistemas operativos Windows 9x e Windows NT, uma vez que a respectiva API não permite o acesso directo aos controladores da porta paralela. A leitura do estado do *switch* pode ser facilitada através da simulação de uma situação de erro, como por exemplo, o “*Paper Out*” e o “*Line-Out*”, que originam mensagens que são enviadas à aplicação de controlo.

PIN	Name	I/O	PIN	Name	I/O
1	/STROBE	Output	13	SLCT	Input
2	DATA0	Output	14	/AUTO FEED	Output
3	DATA1	Output	15	/PERROR	Input
4	DATA2	Output	16	/INIT	Output
5	DATA3	Output	17	/SLCT IN	Output
6	DATA4	Output	18	GND	Input
7	DATA5	Output	19	GND	Input
8	DATA6	Output	20	GND	Input
9	DATA7	Output	21	GND	Input
10	/ACK	Input	22	GND	Input
11	BUSY	Input	23	GND	Input
12	PE	Input	24	GND	Input
			25	GND	Input

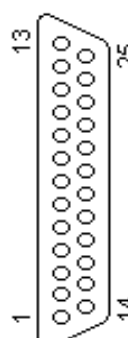


Figura 4.3: *Pinout* da porta paralela

De uma forma geral, os dispositivos de selecção apresentam um comportamento equivalente ao de um interruptor, pelo que há necessidade de os adaptar aos sinais esperados nos pinos de entrada da porta paralela. Para ler o estado do dispositivo de selecção é utilizada uma configuração de “*Pull-up*”, com uma resistência entre o pino de entrada “*Paper Out*” (12) e o pino de saída “*AutoSelect*” (14), programado para ter o valor V_{CC} , estando o dispositivo de selecção colocado entre a massa (21) e o pino “*Paper Out*” (12), tal como se ilustra na figura 4.4.

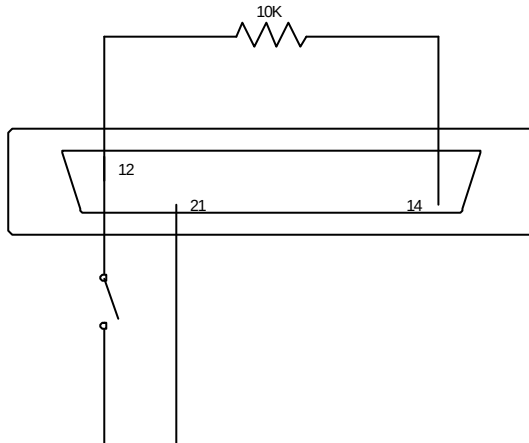


Figura 4.4: Esquema de um adaptador de *switch*

Este adaptador tem como principais vantagens a facilidade de construção, custo bastante reduzido, simplicidade e adaptabilidade a outros tipos de *switch*. O suporte de dispositivos de selecção dupla consegue-se replicando a montagem anterior por forma a incorporar um novo interruptor sobre o pino de entrada “*Busy*” (11).

4.3 Recursos para a comunicação oral

Os recursos físicos necessários para a comunicação oral podem ser conseguidos com um “kit multimédia” composto por uma placa de som (como a apresentada na figura 4.5), um par de colunas e um microfone.

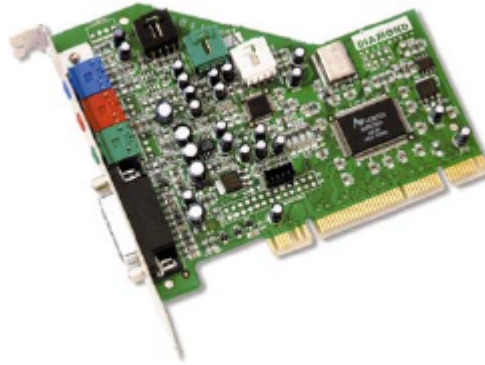


Figura 4.5: Placa de som

O computador, equipado com este recurso adicional, está apto a reproduzir e a gravar sons. A natureza da aplicação de apoio à comunicação oral apresenta no entanto alguns requisitos na qualidade das mensagens utilizadas, que tornam necessário um pacote de *software* de tratamento de áudio, como o apresentado na figura 4.6.

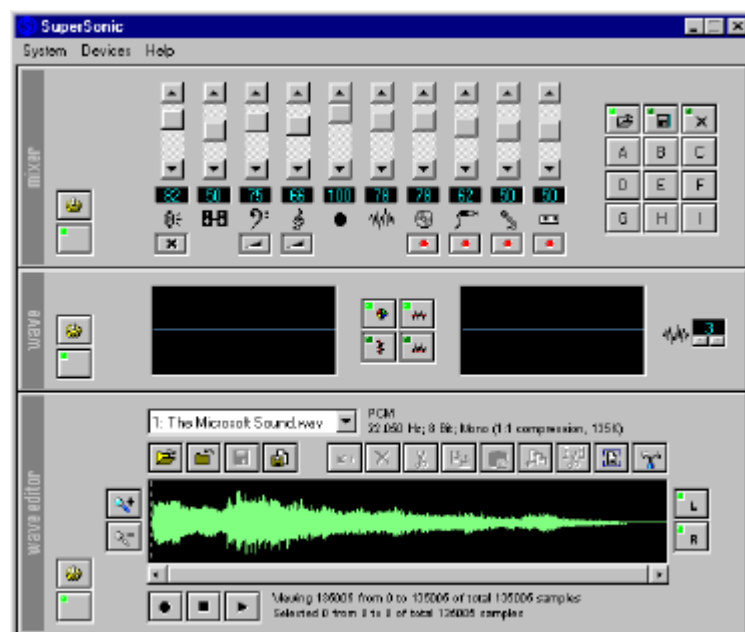


Figura 4.6: *Software* de tratamento de som

Para uma produção de gravações de qualidade é necessária uma infra-estrutura de gravação adequada, uma espécie de mini-estúdio de gravação, composto por um quarto ou sala pequ

na, que não apresentem ruído ambiente, um microfone dinâmico e um par de auscultadores de *feedback*. Com a ajuda do *software* de gravação, deve ser ajustado o ganho do microfone por forma a obter-se um nível de som médio entre os 60 e os 70%, com picos de 90%, de modo a garantir-se uma gravação forte e sem distorção. Com algum investimento, pode ser adquirido um microfone de estúdio com funções de compressão de voz, o que permite aumentar a potência vocal das gravações através de um processo de variação automática do ganho do microfone. A compressão deve no entanto ser limitada a 10dB, para evitar perdas de qualidade na gravação.

A gravação pode ser realizada em vários formatos digitais, como por exemplo o PCM@44KHz com 16 *bits* estéreo (usado nos CD's) e um débito de 172Kbps. No entanto, este formato não traz vantagens à aplicação concreta de apoio à comunicação, uma vez que não se encontra orientado para a representação digital de voz. Os formatos GSM e ADPCM apresentam muita boa qualidade de reprodução de voz, ao mesmo tempo que minimizam o espaço necessário (em disco e em memória) para o seu armazenamento e reprodução, pelo que constituem as alternativas recomendadas para este efeito. O formato preferencial é o GSM 6.10@11.025Hz, suportado nativamente pelo Windows 95 com um débito de apenas 2Kbps.

4.4 Recursos para a comunicação telefónica

A marcação assistida de números telefónicos requer um Modem normalizado como o apresentado na figura 4.7, com conectores para a linha telefónica. Durante a marcação a linha é cortada ao telefone e ligada aos circuitos internos do Modem para marcação por tons *on-i* pulsos, sendo a linha comutada de novo para o telefone após esta fase.



Figura 4.7: Modem interno US Robotics

A ligação do Modem deve ser efectuada da forma apresentada figura 4.8. A solução recomendada consiste no recurso a um telefone com a função de mão livres.

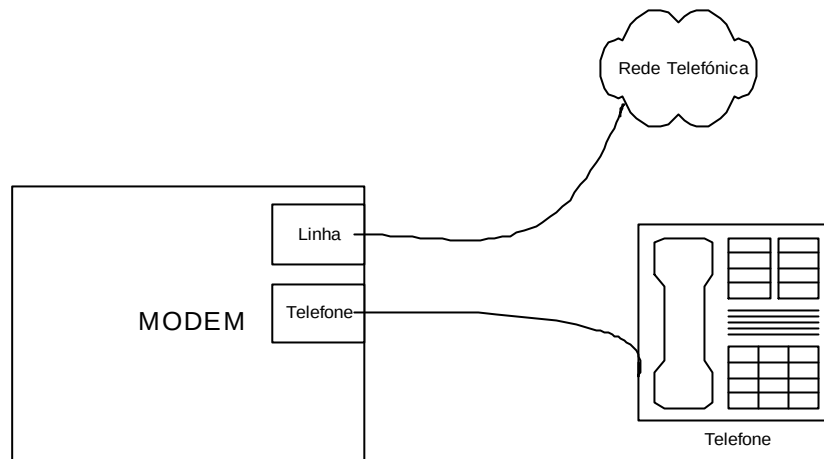


Figura 4.8: Esquema de ligação do Modem

4.5 Recursos para o controlo ambiental

O sistema domótico desenvolvido tem como principal aplicação o controlo e a monitorização de pequenos edifícios, tendo a capacidade de controlar o sistema de iluminação, as tomadas eléctricas, o acesso às divisões, o sistema de abertura e fecho de janelas e o sistema de segurança, com detecção de presenças e de incêndios. Outras aplicações construídas sobre este sistema domótico permitem o controlo por infra-vermelhos de electrodomésticos, como por exemplo televisores, rádios, vídeos, HI-FI, desde que utilizem o protocolo RC5.

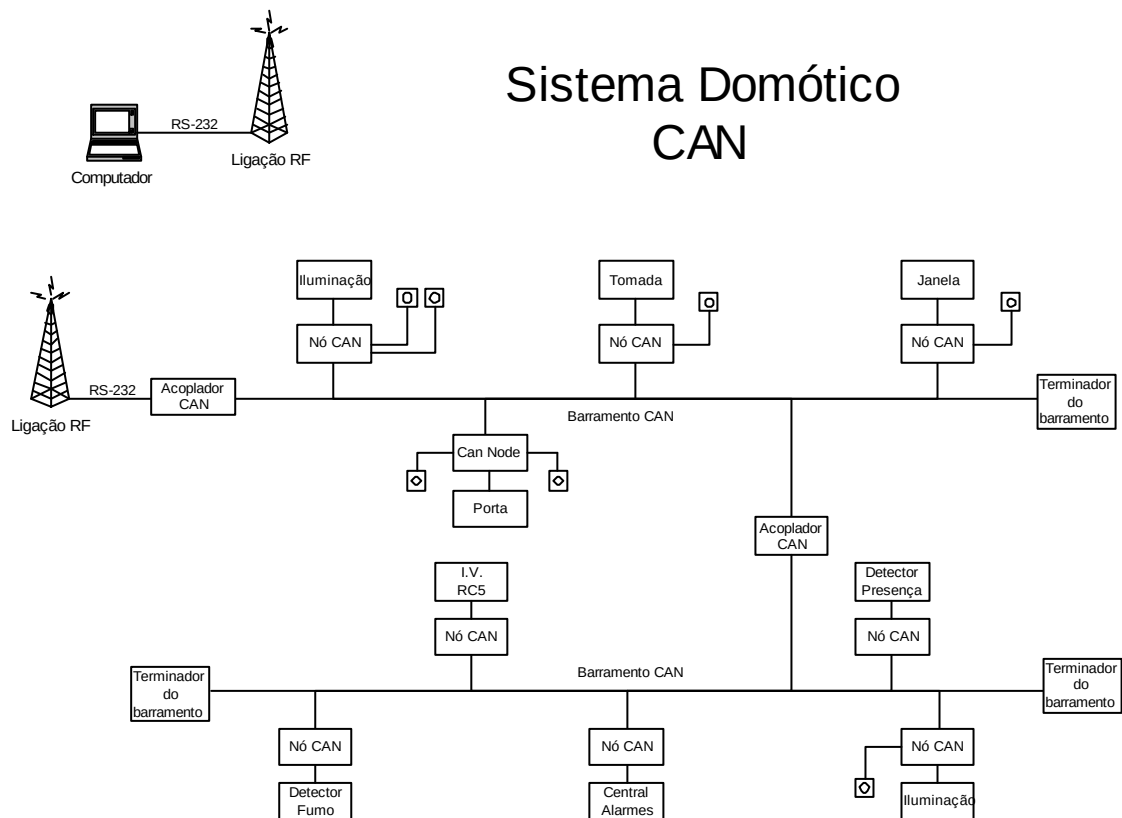


Figura 4.9: Sistema Domótico CAN

O sistema é composto por um conjunto de nós “inteligentes” que comunicam entre si através do protocolo de comunicação CAN (*Controller Area Network*), com um débito de 128 Kbps, usando mensagens *debroadcast* e garantindo que essas mensagens chegam a todos os nós. Isto permite que todos os nós estejam sempre informados do estado da instalação e possam assim reagir a eventos, por forma a optimizarem o processo de gestão do edifício.

Os nós “inteligentes” são implementados por uma carta de circuito impresso única, à qual se deu o nome de nó CAN. Esta carta apresenta uma arquitectura modular, possibilitando o aumento da funcionalidade pela inclusão de componentes electrónicos, sendo a versão mínima composta por circuitos de comunicação CAN e um microcontrolador, e a versão máxima enriquecida por uma porta de comunicação RS-232 e um barramento de *8 bit* de E/S (entradas / saídas) para controlo e monitorização de actuadores e sensores.

A estratégia de funcionamento de um nó CAN é dividida em três áreas funcionais: comunicações, controlo de aplicações e E/S, como se ilustra na figura 4.10.

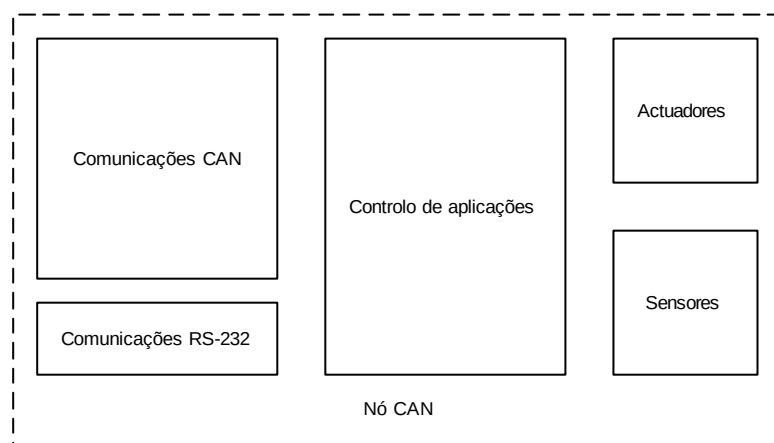


Figura 4.10: Estrutura de um nó CAN

Dentro desta estratégia de desenvolvimento, apresentaremos em seguida o protocolo CAN, a estrutura de *hardware* de um nó CAN e finalmente o *software* de comunicações e de aplicação.

4.5.1 CAN

O CAN foi desenvolvido originalmente na Alemanha pela Bosch, tendo como área principal de aplicações a indústria automóvel. No entanto, a sua aplicação já se estendeu hoje em dia a áreas como a medicina, a indústria, os sistemas de empacotamento, etc. Uma descrição mais detalhada destas aplicações pode ser encontrada em [Khoh95].

O protocolo CAN implementa apenas as duas camadas inferiores do modelo OSI da ISO (*International Standards Organization*) a camada física e a camada de ligação de dados. Existem outros protocolos de alto nível baseados na arquitectura CAN, tais como o DeviceNet, da Allen-Bradley, o Smart Distributed Sensors, da HoneyWell e o CAL (*CAN Application Layer*), da organização Europeia CiA (*CAN in Automation*) [Tind95].



Figura 4.11: Camadas OSI implementadas pelo CAN

O CAN usa o método CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) com atribuição de prioridades em função da sequência *de bit* do cabeçalho (*bit wise*), com arbitração de acesso não destrutiva para contenção do barramento. A transmissão de informação em tempo real emprega o tipo de endereçamento funcional utilizado em difusão (*broadcast*), pelo que uma mensagem enviada por uma estação é recebida por todas as restantes. Como as tramas do protocolo CAN não contêm campo de remetente nem campo de destinatário, o *overhead* resultante é muito pequeno. Utiliza-se o campo do identificador do tipo de mensagem (*Message Identifier*) para estabelecer prioridades e assim conseguir níveis de serviço com informação do tipo alarme (alta prioridade) e informação de gestão (baixa prioridade). A configuração da versão CAN 1.0 permite ter 2048 tipos de mensagens, um número suficientemente elevado mesmo para sistemas distribuídos de elevada complexidade. No entanto, a versão actual do protocolo CAN 2.0 permite 2²⁹ tipos de mensagens, o que permite utilizar este campo como um endereço de destino com 11 *bit*, sobrando ainda 5 *bit* para 32 tipos de mensagens.

O protocolo CAN usa o método de codificação *de bit* NRZ (*Non Return to Zero*) com *bit stuffing* e taxas de transmissão (*Data Rates*) que vão desde 15 Kbps a 1 Mbps. A codificação NRZ, pela sua natureza, não garante um número suficiente de transições para a sincronização dos relógios dos controladores de comunicação CAN. Para superar esta deficiência, é utilizado o processo *de bit stuffing*, que consiste na inserção de um *bit* complementar sempre que ocorrem sequências de 5 *bit* com o mesmo sinal, o qual é automaticamente removido pelas estações receptoras. O método NRZ ainda tem a vantagem de produzir menor emissão electromagnética a níveis de transferência mais elevados, face aos métodos de codificação RZ [CiA]. Isto deve-se ao facto dos harmónicos da portadora fundamental serem de frequência pelo menos duas vezes menor que os gerados pelos outros métodos de codificação, uma vez que o número de transições *por bit* nos métodos RZ é sempre superior a um.

As aplicações para o ramo automóvel implicam elevados níveis de segurança, pelo que o CAN foi desenvolvido para ser um meio muito eficiente para a transmissão de dados, com uma elevada capacidade de correcção de erros, tolerância a falhas físicas e imunidade ao ruído electromagnético [CiA].

Tal como acontece em qualquer sistema que use CSMA/CD, a taxa de colisões depende do comprimento do barramento. Sendo assim, o limite físico do barramento para taxas de transmissão de 1 Mbps é de 50 metros, enquanto que para 100 Kbps é de 500 metros.

Protocolo

O protocolo CAN suporta quatro tipos de tramas de comunicação:

- Trama de Dados (*Data Frame*)
- Trama Remota (*Remote Frame*)
- Trama de Erro (*Error Frame*)
- Trama de Sobrecarga (*Overload Frame*)

Cada um destes tipos de tramas será apresentado de forma sumária nos próximos parágrafos.

Trama de Dados

Uma Trama de Dados, apresentada na figura 4.12, transporta mensagens de um transmissor CAN para todos os receptores.

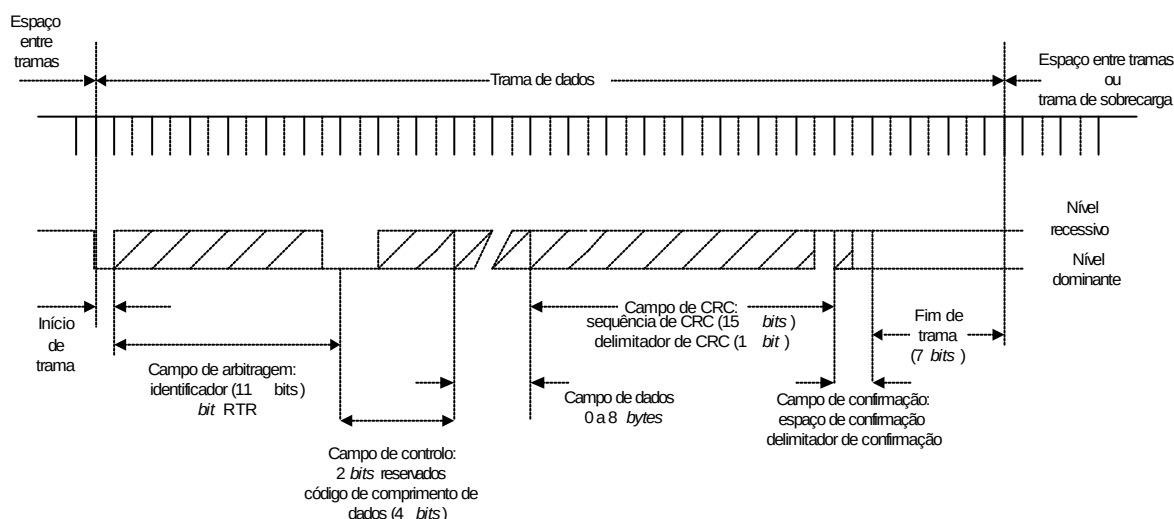


Figura 4.12: Trama de Dados

Esta trama é composta por 7 campos diferentes, nomeadamente:

Bit de Início de Trama (*Start of Frame*) - É composto por 1 bit dominante ("0"), que assinala o início de uma Trama de Dados ou de uma Trama Remota e força a sincronização de um controlador CAN no modo de recepção.

Campo de Arbitragem (*Arbitration Field*) - É composto por 11 bit para o Identificador da Mensagem (*Message Identifier*) e o bit de Pedido de Transmissão Remota (RTR, *Remote Transmission Request*). O Identificador de Mensagem serve para fornecer informação acerca da mensagem e para definir a prioridade de acesso ao barramento. O bit RTR indica se a trama transmitida é uma Trama de Dados (bit a 0) ou uma Trama Remota (bit a 1). Quando há transmissões simultâneas, o conflito é resolvido por arbitragem bit wise durante a transmissão do Campo de Arbitragem.

Campo de Controle (*Control Field*) - É composto por 6bit, 2 bit reservados e o Código de Comprimento de Dados (DLC, *Data Length Code*), que indica o número de bytes de dados (sem *bit stuffing*) a serem transmitidos/recebidos no Campo de Dados. Os valores admissíveis para o DLC vão de 0 a 8 bytes.

Campo de Dados (*Data Field*) - Os dados guardados no Campo de Dados do *Buffer de Transmissão* são transmitidos de acordo com o Código de Comprimento de Dados. Inversamente, os dados recebidos através de uma Trama de Dados são guardados no Campo de Dados do *Buffer de Recepção*. O Campo de Dados pode ter de 0 a 8 bytes.

Código de Redundância Cíclica (CRC, *Cyclic Redundancy Code*) - Tem 15 bit para a sequência CRC e 1 bit recessivo ("1") para o Delimitador de CRC. O Código de Redundância Cíclica é o quociente da divisão por $(2^{15} - 1)$ da soma das sequências de bit (sem *bit stuffing*) do Bit de Início de Trama, do Campo de Arbitragem, do Campo de Controle e do Campo de Dados. O Delimitador de CRC serve para delimitar o campo anterior e para detectar a ocorrência de eventuais colisões no barramento.

Campo de Confirmação (*Acknowledge Field*) - É composto por 2bit, o Intervalo de Confirmação (*Acknowledge Slot*) e o Delimitador de Confirmação (*Acknowledge Delimiter*), que são transmitidos com um nível recessivo pelo transmissor da Trama de Dados. Quando um controlador CAN recebe a sequência de CRC correspondente, sobrepõe um bit dominante ao bit recessivo do Intervalo de Confirmação. O Delimitador de Confirmação é um bit recessivo e serve para delimitar o campo anterior.

Fim de trama (*End of Frame*) - Cada Trama de Dados ou Trama Remota é delimitada por este campo, que é composto por 7bit recessivos, o que excede a largura de *bit stuffing* por 2 bit. Ao usar este método, o receptor detecta o fim de trama independentemente de existirem ou não erros, pois o receptor espera que todos os bits, até o fim da sequência de CRC, estejam codificados pelo método *bit stuffing*.

Trama Remota

A Trama Remota, apresentada na figura 4.13, serve para informar as outras estações na rede que já é possível a recepção de mais informação. Esta trama endereça a fonte de dados através do identificador e põe o bit RTR a 1, para indicar que se trata de uma Trama Remota.

Esta trama é idêntica à Trama de Dados, a menos das seguintes exceções:

- bit RTR é posto a 1
- Código de Comprimento de Dados é ignorado
- Não contém um Campo de Dados

O Código de Comprimento de Dados deve ser o mesmo da correspondente Trama de Dados, apesar de ser ignorado para a Trama Remota.

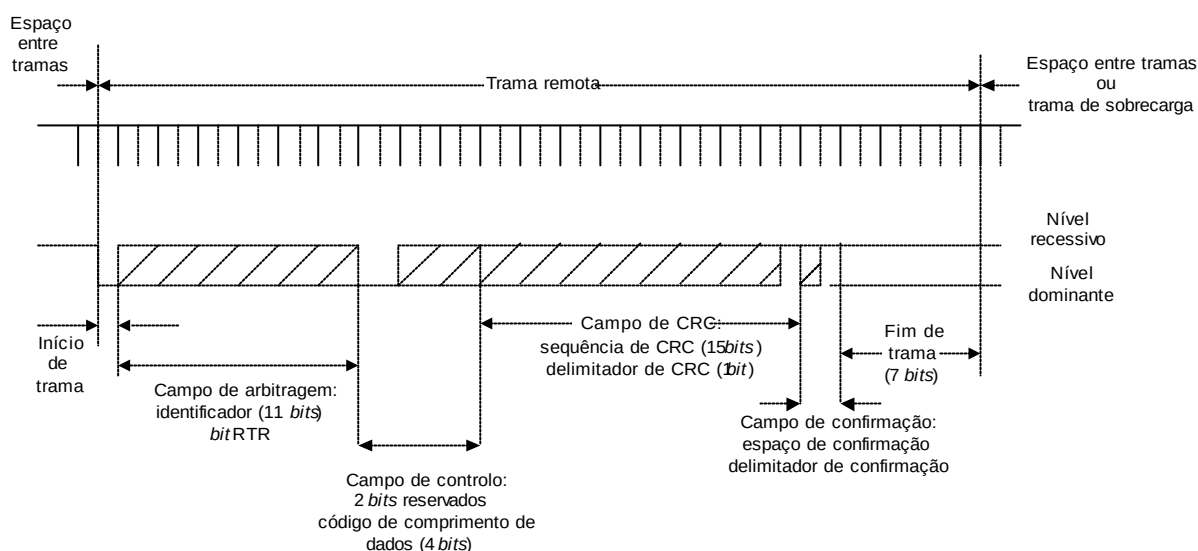


Figura 4.13: Trama Remota

Trama de erro

A Trama de Erro é transmitida pelos controladores ligados ao barramento quando detectam um erro de *bit* ou de forma da trama, sendo composta por dois campos.

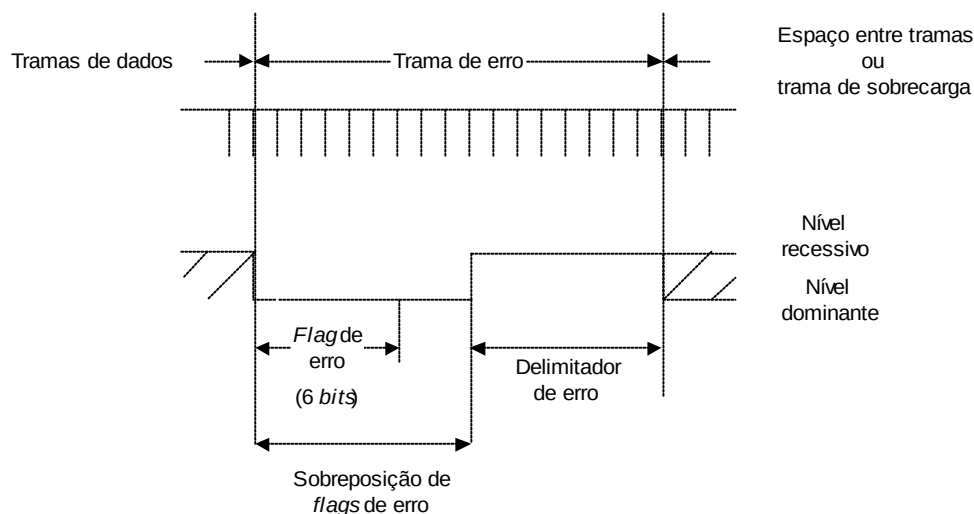


Figura 4.14: Trama de Erro

Sobreposição de Flags de Erro (Superimposing of Error Flag) - Existem dois tipos de *flag* de erro:

- *Flag* de erro activa, que consiste em 6 *bits* dominantes consecutivos e que é transmitida por um controlador CAN activo para erros.
- *Flag* de erro passiva, que consiste em 6 *bits* recessivos consecutivos, a não ser que sejam sobrepostos por *bits* dominantes de outros controladores. Esta *flag* é transmitida por um controlador CAN passivo para erros.

Quando um controlador CAN, activo para erros, detecta um erro, transmite um *flag* de erro activa. Como esta *flag* viola a regra de *bit stuffing*, ou viola a forma fixa dos campos a serem transmitidos no momento, os outros controladores detectam um erro e começam a transmitir as suas *flags* de erro. Deste modo, a sequência de *bit* que se encontra no barramento é composta pela sobreposição de várias *flags* de erro e pode ter no mínimo 6 e no máximo 12 *bit*.

Delimitador de Erro (*Error Delimiter*) - É composto por 8 *bit* recessivos. Depois de terem transmitido a *Flag* de Erro, os controladores CAN escutam o barramento até detectarem uma transição de dominante para recessivo. Neste momento já todos os controladores acabaram de enviar a *Flag* de Erro e já enviaram os primeiros 3 *bit* do Delimitador de Erro. Depois de acabarem de enviar o resto do delimitador e depois de um Campo de Intervalo, todos os controladores CAN podem reiniciar a transmissão.

Se o erro foi detectado durante a transmissão de uma Trama Remota ou de uma Trama de dados, a mensagem fica danificada e tem que ser retransmitida.

Se um controlador CAN detecta um desvio na Trama de Erro, transmite uma nova Trama de Erro. Várias Tramas de Erro consecutivas podem pôr o controlador no estado passivo para erros.

Trama de Sobrecarga

A Trama de Sobrecarga, apresentada na figura 4.15, é composta por dois campos: *Flag* de sobrecarga e o delimitador de sobrecarga.

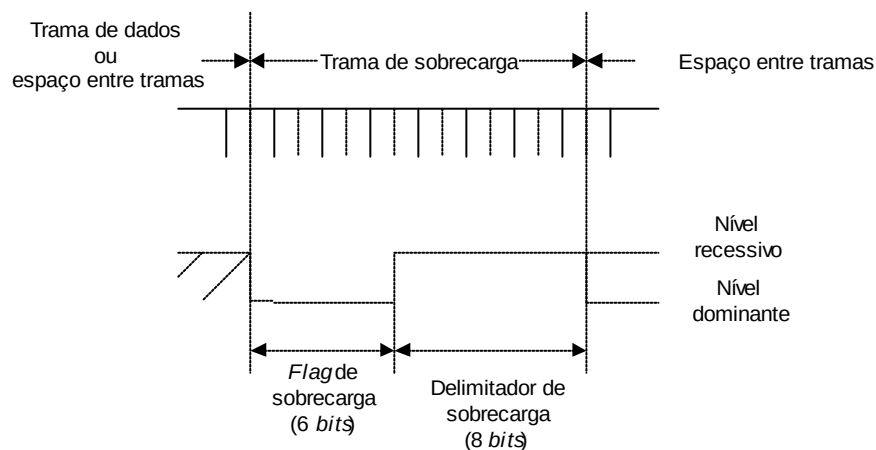


Figura 4.15: Trama de Sobrecarga

Existem duas condições que podem levar à transmissão de uma Trama de Sobrecarga:

- receptor não está pronto, pois necessita de mais tempo para processar os dados que acabou de receber.
- Detectou um *bit* dominante durante um Campo de Intervalo pertencente ao Espaço entre Tramas. Este campo só deveria ter *bit* recessivos, como será explicado na secção seguinte.

A transmissão de uma Trama de Sobrecarga só pode ser iniciada nas seguintes condições:

- Durante o primeiro período de *bit* do Campo de Intervalo.

- Um período *debt* após ter detectado *umbit* dominante durante o Campo de Intervalo. A Trama de Sobrecarga é idêntica à Trama de Erro. Estas tramas são no entanto são tratadas de forma diferente, pois uma Trama de Sobrecarga não obriga à retransmissão da trama anterior.

Espaço entre Tramas

As Tramas de Dados e as Tramas Remotas são separadas das tramas precedentes por um Espaço entre Tramas, que é composto por:

Campo de Intervalo (*Intermission Field*) - É composto por 3bit recessivos. Durante este período não são iniciadas transmissões pelo controlador CAN. Este Campo de Intervalo deve ter um tempo fixo, para que o controlador CAN execute os processos internos que antecedem a próxima transmissão ou recepção.

Barramento Inactivo (*Bus Idle*) - Tempo que pode ser de duração arbitrária (mínimo de 0 bit) e que permite resolver os conflitos de acesso ao barramento, por várias estações ao mesmo tempo. Ao determinar que o barramento está livre, um controlador que tenha informação a transmitir pode aceder ao barramento.

Modo de Funcionamento

As mensagens transmitidas por qualquer estação para o barramento CAN não contêm endereço de emissor nem endereço de receptor. Em vez disso, o conteúdo da mensagem é etiquetado por um identificador que é único dentro da rede e pelo bit RTR, que indica o tipo de mensagem. Todas as estações recebem todas as mensagens que se encontram no barramento e fazem um teste de aceitação ao Identificador, para saberem se querem receber essa mensagem. O Identificador e o bit RTR, além de identificarem a mensagem, servem para definir a sua prioridade. A mensagem com o valor binário mais baixo destes dois campos tem a maior prioridade. Sendo assim, uma Trama de Dados tem maior prioridade que uma Trama Remota pois esta última tem o bit RTR a 1.

Enquanto o barramento está ocupado, ou seja, enquanto existe alguma estação a transmitir, as outras estações esperam. Quando o barramento fica inactivo (*Bus Idle*), todas as estações com mensagens a transmitir entram num processo de arbitragem, iniciando a transmissão do identificador da sua mensagem mais prioritária, do bit mais significativo para o bit menos significativo.

Como todas as estações começam a transmitir simultaneamente, seria de esperar que houvesse uma colisão e que todas as mensagens tivessem que ser retransmitidas. No entanto, o barramento CAN tem propriedades físicas que o tornam equivalente a um Wired-Or:

- Se uma estação transmite um 0, o barramento terá um 0 independentemente do valor que as outras estações tiverem transmitido, pois este é o valor dominante.
- Só quando todas as estações transmitirem um 1 é que o barramento terá um 1.

Como cada estação monitoriza o barramento ao mesmo tempo que transmite uma mensagem, durante o processo de arbitragem, se uma estação transmitir um 1 e encontrar um 0, então sabe que ocorreu uma colisão e que existe uma mensagem de maior prioridade a ser transmitida. O protocolo obriga a que uma estação que detecte uma colisão pare imediatamente de transmitir, para não corromper a mensagem a ser transmitida (contenção não destrutiva).

Para entender o processo de arbitragem no acesso ao barramento, tomemos como exemplo a figura 4.16, que ilustra o caso de três estações, A, B e C, que tentam transmitir simultaneamente. A trama de maior prioridade que a estação A tem para transmitir tem o Identificador $59C_H$ (10110011100_B), a trama de maior prioridade da estação B tem o ID 537_H (10100110111_B) e a da estação C tem o ID 539_H (10100111001_B). A competição pelo barramento é ganha pela estação B, porque os seus *bits* dominantes sobrepõem-se ao *bit* 7 da estação A, que é recessivo, e ao *bit* 3 da estação C, que é igualmente recessivo. Do ponto de vista do barramento, a mensagem de maior prioridade foi a única a ser transmitida.

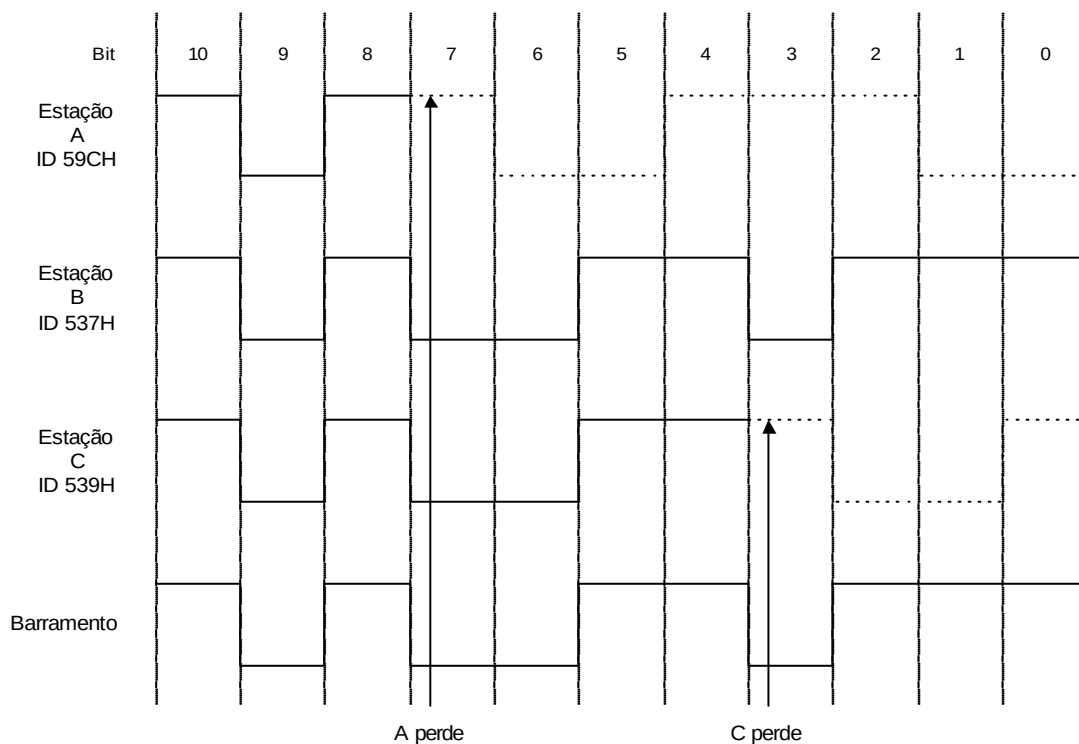


Figura 4.16: Processo de arbitragem entre três estações

Os mecanismos de integridade de dados implementados pelo protocolo CAN contemplam a detecção de cinco tipos de erros: erro de *bit*, erro de *bit stuffing*, erro de CRC, erro de forma e erro de confirmação.

Erro de *bit* - se o nível do *bit* monitorizado no barramento for diferente do enviado, é sinalizado um erro de *bit*. As exceções a esta regra são o Campo de Arbitragem, onde esta situação significa que o controlador não tem a mensagem de maior prioridade, e o espaço de confirmação, onde só os receptores podem detectar este erro.

Erro de *bit stuffing* - este erro ocorre quando os campos de início de trama, de arbitragem, de controlo, de dados e de CRC, não obedecem às regras *dbit stuffing*.

Erro de CRC - ocorre quando a verificação de CRC falha.

Erro de forma - resulta da violação das formas fixas dos seguintes campos: delimitador de CRC, delimitador de confirmação, fim de trama, delimitador de erro e delimitador de sobre carga.

Erro de confirmação - é detectado por um transmissor CAN sempre que este não detecta um *bit* dominante durante o espaço de confirmação.

Quando um controlador CAN detecta um erro transmite um *Flag* de Erro. Esta *flag* é transmitida após o delimitador de confirmação, se for um erro de CRC, ou ~~bit~~ seguinte ao erro, em todos os outros tipos. *AFlag* de Erro viola o *bit stuffing*, de modo que todos os controladores detectam o erro e iniciam igualmente a transmissão de um ~~Flag~~ *Flag* de Erro.

4.5.2 NÓ CAN

O principal elemento de *hardware* que compõe o sistema domótico é a carta de circuito impresso ilustrada na figura 4.17, à qual se deu a denominação de nó CAN, uma vez que está sempre presente na ligação de um dispositivo à rede CAN.



(a) Carta do nó CAN.



(b) Barramento CAN.



(c) Cartas CAN, de alimentação e para o comando de actuadores.

Figura 4.17: Nó CAN

Esta carta desempenha as funções de comunicação com o barramento CAN e com dispositivos externos via RS-232, para além de executar a aplicação de controlo, dispondo de oito sinais de E/S, tal como está esquematizado na figura 4.18.

O nó CAN foi concebido por forma a minimizar as dimensões e o custo, pelo que foi escolhido um microcontrolador da família 80C51, com 4Kb de memória EPROM interna, para armazenamento do código de comunicações e de aplicação. Este microcontrolador dispõe de 128 bytes de memória RAM, que são usados para armazenamento de variáveis *buffers* de transmissão, disponibilizando ainda 4 portas de E/S paralela reconfiguráveis.

A comunicação com o barramento CAN é realizada por dois circuitos integrados da Philips, que executam as funções de implementação do protocolo (através do controlador PCA82C200 ou SJA1000) e de acoplamento ao barramento (desempenhadas pelo transceptor PCA82C250).

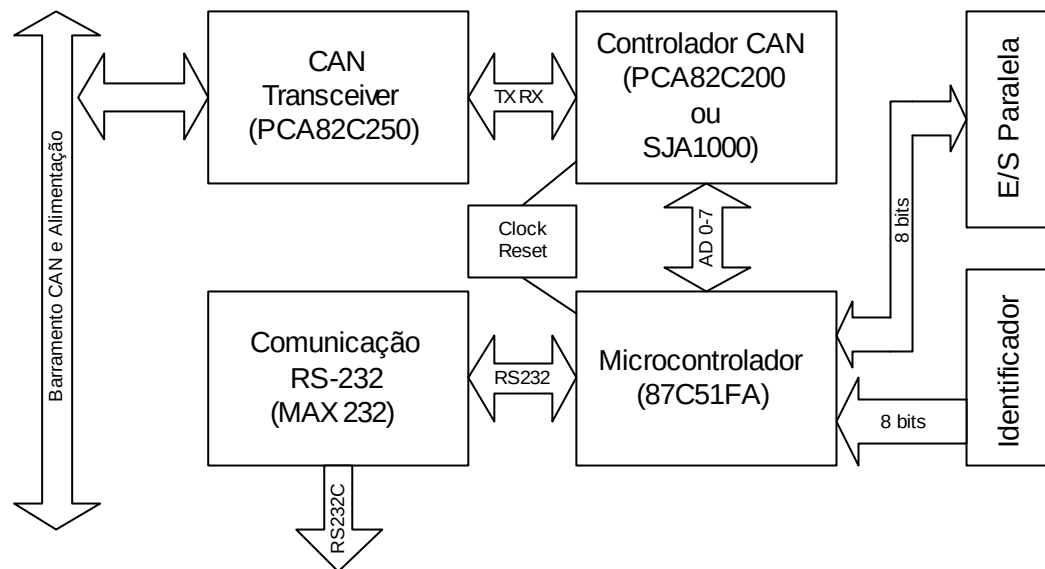


Figura 4.18: Diagrama de blocos de um nó CAN

Para efeitos de reconfiguração dos nós CAN, foi adicionado um interruptor **DIP** de 8 contactos para a definição de um Identificador de nó, que permite atribuir-lhe uma identificação definida pelo instalador. Existe um barramento com **8 bit** de E/S para controlo de actuadores e leitura de sensores acoplados ao nó CAN, conseguindo-se assim reutilizar o mesmo nó para o controlo de iluminação e para o controlo de electrodomésticos via infravermelhos.

A comunicação com um PC para controlo remoto e monitorização a partir de uma consola com uma interface gráfica é possível graças à porta RS232 disponibilizada pelo microcontrolador, configurada para comunicar a **9600baud**, **8 bit**, sem paridade e com um **stop bit** (a adaptação de níveis é realizada pelo circuito integrado MAX 232 ou por um seu equivalente, como o LT1181).

O esquema eléctrico de um nó CAN é apresentado na figura 4.19, no qual podemos constatar que o controlador CAN PCA 82C250 está ligado como uma memória, localizada no endereço **0000H**, com descodificação parcial de endereços. O interruptor **DIP** de selecção do identificador está ligado à porta P1 com uma série de resistências de **pull-up** externas de **10KΩ** cada. O conector de E/S do nó CAN é ligado à porta P2 do microcontrolador e inclui ainda os sinais de **V_{CC}** e **GND**, para alimentação de uma carta externa de **drivers**. O circuito MAX232 encontra-se ligado aos sinais da UART do microcontrolador, que no caso do 87C51 pertencem à porta P3. O sistema de relógio é partilhado entre o microcontrolador e controlador CAN, pelo que há necessidade de estudar a forma de propagação deste sinal pela carta de circuito impresso por forma a evitar ruídos desnecessários **cross-talk**. O sistema de **auto-reset** no arranque foi planeado por forma a realizar primeiro o **reset** no controlador CAN e depois no microcontrolador, uma vez que aquele necessita de mais ciclos de relógio para realizar a sua inicialização interna.

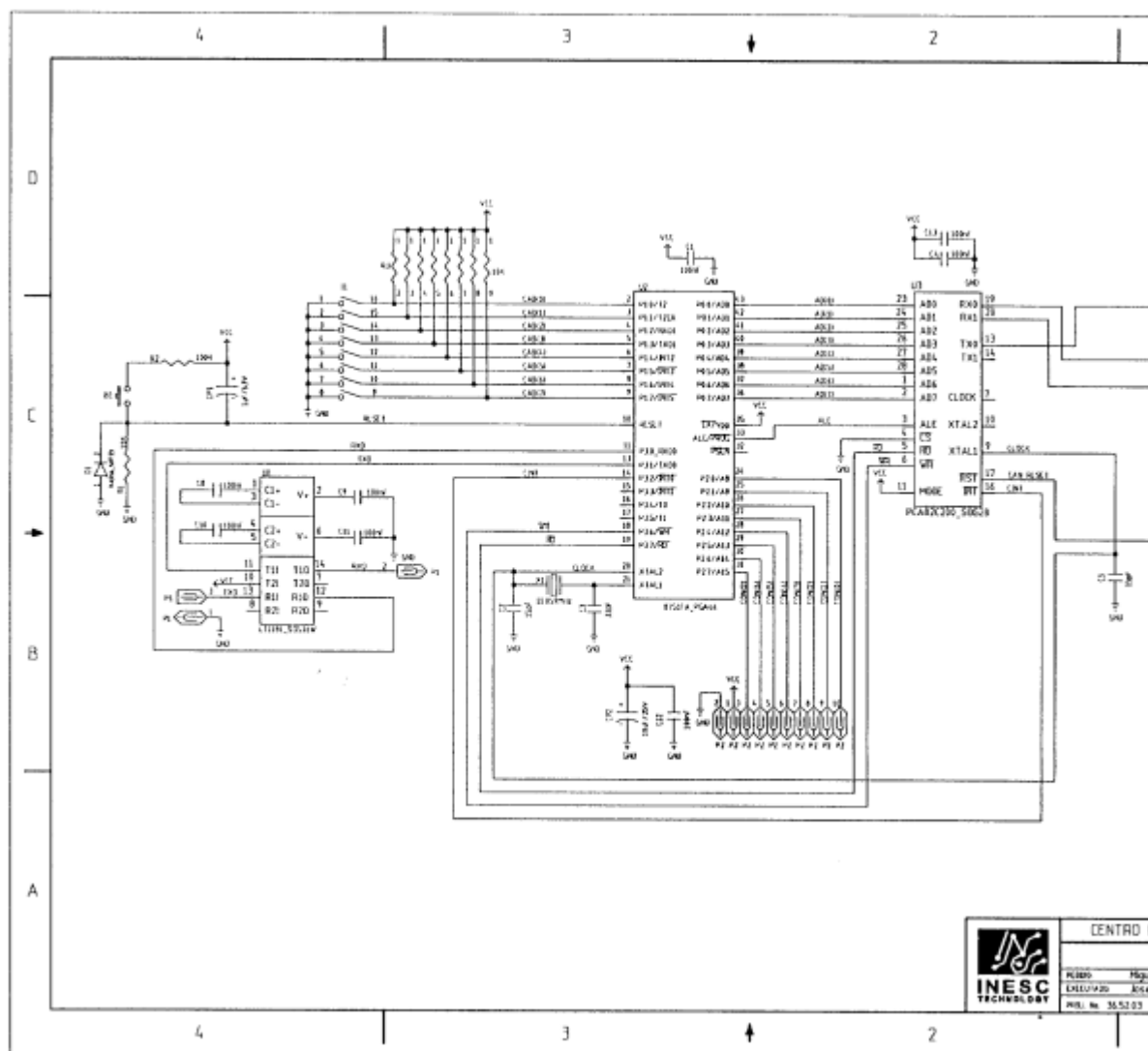


Figura 4.19: Esquema eléctrico de um nó CAN

No esquema apresentado não foi incluído o circuito implementado numa carta montada *pluggyback* para a conversão da alimentação, que se ilustra na figura 4.17.c (de 24 para 5 V_{DC}), necessária porque a tensão de distribuição do barramento respeita o valor para distribuição industrial (24 V_{DC}). Um esquema desta carta conversora encontra-se um anexo.

Após esta breve descrição das ligações que compõem um nó CAN, é necessário aprofundar a descrição no que respeita ao sistema CAN utilizado, por forma a compreender o funcionamento do *software* desenvolvido para as comunicações.

O controlador PCA80C200

O Philips PCA80C200 e o (actual) SJA1000 incluem todos os módulos *hardware* necessários para implementar a camada de transferência que representa o núcleo do protocolo CAN, versão 2.0, estando representado na figura 4.20 o seu esquema interno [Phil].

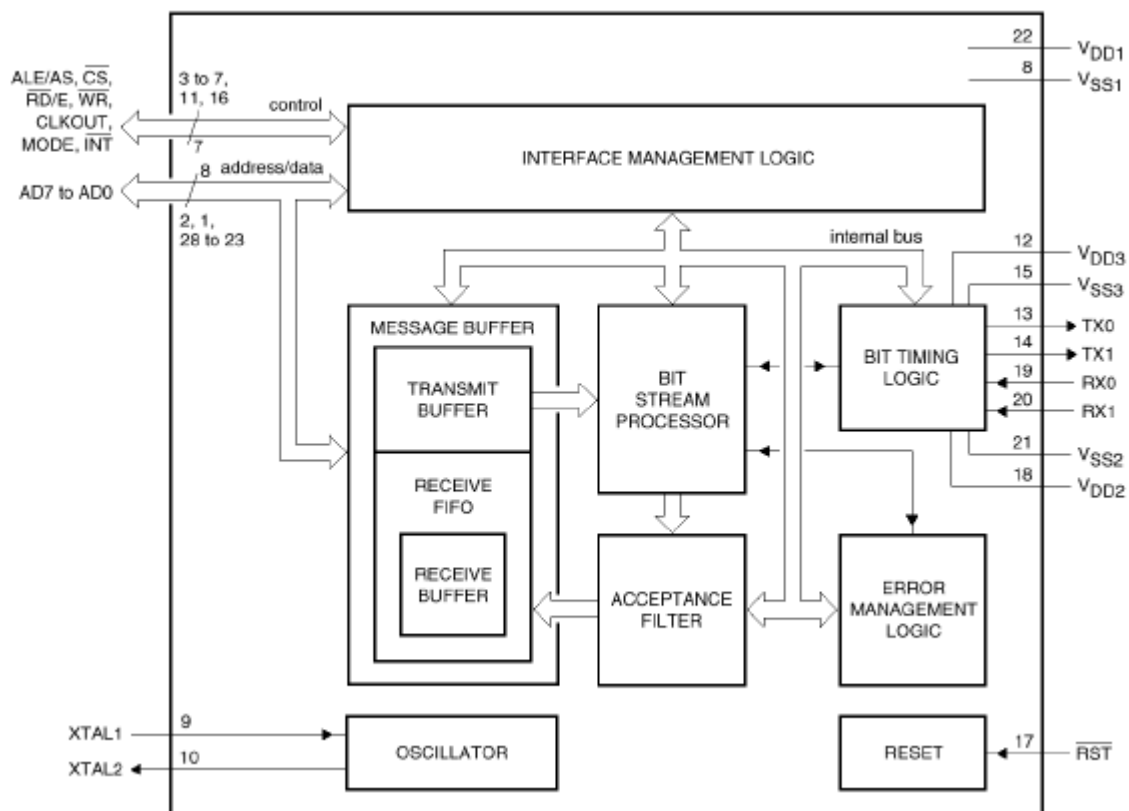


Figura 4.20: Esquema interno do Controlador CAN

Lógica de Gestão de Interface (IML, *Interface Management Logic*) - Este bloco interpreta os comandos vindos do CPU, controla a alocação dos *buffers* de mensagens (*Buffer de Transmissão*, *Buffer de Recepção* e *Buffer de Recepção*) e fornece ao CPU informações de estado e de interrupções.

Buffer de Transmissão (TBF, *Transmit Buffer*) - O *Buffer de Transmissão* faz a interface entre o CPU e o Processador da Sequência de *Bit* e pode armazenar uma mensagem completa, já que tem a capacidade máxima de 10 *bytes* (o que se traduz em 2 *bytes* para o Descritor e 8 *bytes* para o Campo de Dados). Este *buffer* é escrito pelo CPU e é lido pelo Processador da Sequência de *Bit*.

Buffer de Recepção (RBF0, *Receive Buffer 0* RBF1, *Receive Buffer 1*) - O *Buffer de Recepção* faz a interface entre o Processador da Sequência de *Bit* e o CPU, guardando a mensagem recebida do barramento. Este *buffer* não pode guardar outra mensagem enquanto a mensagem anterior não for lida pelo CPU, sendo por isso que existem dois *buffers* de mensagens (RBF0 e RBF1). Deste modo, enquanto um *buffer* está alocado ao CPU, o outro *buffer* pode ser preenchido pelo Processador da Sequência de *Bit* com outra mensagem. Estes dois *buffers* têm igualmente o tamanho de 10 *bytes*.

Processador da Sequência de Bit (BSP, *Bit Stream Processor*) - Este bloco contém um sequenciador, que controla a sequência de *bit* entre o barramento (dados em série) e os *buffers* de transmissão e recepção (dados em paralelo). O BSP contém o filtro de aceitação e controla a lógica do transceptor e a lógica de gestão de erros, de modo a que os processos de recepção, arbitragem, transmissão e sinalização de erros, sejam feitos de acordo com o protocolo. Este bloco envia sinais ao IML, indicando quando uma mensagem foi aceite, quando um *Buffer de Recepção* contém uma mensagem válida e também quando o *Buffer de Transmissão* já não é mais necessário, após uma transmissão com sucesso.

Lógica do Tempo do Bit (BTL, *Bit Timing Logic*) - Monitoriza o barramento e trata da regulação do tempo do *bit*. O BTL sincroniza numa transição recessivo-dominante no início de uma mensagem (*hard synchronisation*) e resincroniza em transições seguintes, durante a recepção da mensagem (*soft synchronisation*). O BTL também fornece segmentos de tempo programáveis, para compensar atrasos de propagação e desvios de fase (devido aos desvios dos osciladores), e para definir o número de amostras que se tiram (uma ou três), dentro do tempo *bit*.

Lógica do Transceptor (TCL, *Transceiver Logic*) - Este bloco controla o *driver* da saída de transmissão.

Lógica de Gestão de Erros (EML, *Error Management Logic*) - Este bloco é responsável pela gestão de erros. O EML recebe avisos de erro do BSP e informa o BSP, o IML e o TCL, das estatísticas de erro.

Organização dos registos internos

Do ponto de vista do microcontrolador, o controlador CAN é um periférico mapeado em memória. A área de endereçamento do controlador CAN está ilustrada na figura 4.21 e é composta pelo Segmento de Controlo (*Control Segment*) e pelos *buffers* de mensagens: *Buffer de Transmissão* (*Transmit Buffer*) e *Buffers de Recepção* (*Receive Buffer*).

CAN ADDRESS	SEGMENT	OPERATING MODE		RESET MODE	
		READ	WRITE	READ	WRITE
0	control	control	control	control	control
1		(FFH)	command	(FFH)	command
2		status	—	status	—
3		interrupt	—	interrupt	—
4		(FFH)	—	acceptance code	acceptance code
5		(FFH)	—	acceptance mask	acceptance mask
6		(FFH)	—	bus timing 0	bus timing 0
7		(FFH)	—	bus timing 1	bus timing 1
8		(FFH)	—	output control	output control
9		test	test; note 2	test	test; note 2
10	transmit buffer	identifier (10 to 3)	identifier (10 to 3)	(FFH)	—
11		identifier (2 to 0), RTR and DLC	identifier (2 to 0), RTR and DLC	(FFH)	—
12		data byte 1	data byte 1	(FFH)	—
13		data byte 2	data byte 2	(FFH)	—
14		data byte 3	data byte 3	(FFH)	—
15		data byte 4	data byte 4	(FFH)	—
16		data byte 5	data byte 5	(FFH)	—
17		data byte 6	data byte 6	(FFH)	—
18		data byte 7	data byte 7	(FFH)	—
19		data byte 8	data byte 8	(FFH)	—
20	receive buffer	identifier (10 to 3)	identifier (10 to 3)	identifier (10 to 3)	identifier (10 to 3)
21		identifier (2 to 0), RTR and DLC	identifier (2 to 0), RTR and DLC	identifier (2 to 0), RTR and DLC	identifier (2 to 0), RTR and DLC
22		data byte 1	data byte 1	data byte 1	data byte 1
23		data byte 2	data byte 2	data byte 2	data byte 2
24		data byte 3	data byte 3	data byte 3	data byte 3
25		data byte 4	data byte 4	data byte 4	data byte 4
26		data byte 5	data byte 5	data byte 5	data byte 5
27		data byte 6	data byte 6	data byte 6	data byte 6
28		data byte 7	data byte 7	data byte 7	data byte 7
29		data byte 8	data byte 8	data byte 8	data byte 8
30		(FFH)	—	(FFH)	—
31		clock divider	clock divider; note 3	clock divider	clock divider

Figura 4.21: Mapa de endereçamento interno do PCA82C200

O Segmento de Controlo é programado durante a inicialização, de modo a configurar os parâmetros de comunicações, e é usado pelo CPU para a comunicação através do barramento CAN. O controlador CAN possui dois *Buffers de Recepção* para aumentar a capacidade de recepção do controlador, pois um *Buffer de Recepção* só pode voltar a ser escrito pelo controlador CAN após ter sido libertado pelo CPU. Sendo assim, enquanto o CPU lê um *Buffer de Recepção* o controlador CAN pode escrever no outro *buffer*. O controlador CAN comuta de um *buffer* para o outro, automaticamente e sem qualquer intervenção por parte do microcontrolador. Os dois *Buffers de Recepção* têm o mesmo endereço, para que não seja necessário saber qual é o *buffer* utilizado em cada momento.

Segmento de Controlo

A troca de sinais de estado ou de comando, entre o CPU e o controlador CAN, é executada no Segmento de Controlo (*Control Segment*), que é composto por 9 registos, apresentados na tabela 4.1.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Endereço 0: Registo de Controlo (CR <i>Control Register</i>)							
TM	S	RA	OIE	EIE	TIE	RIE	RR
Endereço 1: Registo de Comando (CMR <i>Command Register</i>)							
RX0A	RX1A	WUM	SLP	COS	RRB	AT	TR
Endereço 2: Registo de Estado (SR <i>Status Register</i>)							
BS	ES	TS	RS	TCS	TBS	DO	RBS
Endereço 3: Registo de Interrupção (IR <i>Interrupt Register</i>)							
Reservado	Reservado	Reservado	WUI	OI	EI	TI	RI
Endereço 4: Registo do Código de Aceitação (ACR <i>Acceptance Code Register</i>)							
AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0
Endereço 5: Registo da Máscara de Aceitação (AMR <i>Acceptance Mask Register</i>)							
AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0
Endereço 6: Registo 0 de Regulação do Tempo do Barramento (BTR <i>Bus Timing Register 0</i>)							
SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0
Endereço 7: Registo 1 de Regulação do Tempo do Barramento (BTR <i>Bus Timing Register 1</i>)							
SAM	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0
Endereço 8: Registo de controlo da saída (OCR <i>Output Control Register</i>)							
OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCMODE1	OCMODE0
Endereço 9: Registo de teste (TR <i>Test Register</i>)							
Reservado	Reservado	MIP	CRB0	CTBC	AIB	NRC	FOD

Tabela 4.1: Segmento de controlo

Depois da inicialização, o conteúdo dos registos Código de Aceitação (*Acceptance Code*), Máscara de Aceitação (*Acceptance Mask*), Controlo de Saída (*Output Control*), e dos registos 0 e 1 de Regulação do Tempo do Barramento (*Bus Timing 0 & 1*), não podem ser mudados, pois só estão acessíveis quando o bit de Pedido de Reinicialização no Registo de Controlo (*Control Register*) está a 1.

Registo de Controlo

O conteúdo do Registo de Controlo é usado para modificar o comportamento do controlador CAN. Estes *bit* podem ser modificados pelo CPU, que usa este registo como uma memória de leitura/escrita. A tabela 4.2 descreve o *bit* do Registo de Controlo, apresentando os símbolos e a nomenclatura, tanto em inglês como em português, que irá ser utilizada na exposição subsequente.

Bit	Símbolo	Nome em Inglês	Designação adoptada
CR.0	RR	<i>Reset Request</i>	Pedido de Reinicialização
CR.1	RIE	<i>Receive Interrupt Enable</i>	Habilitação da Interrupção de Recepção
CR.2	TIE	<i>Transmit Interrupt Enable</i>	Habilitação da Interrupção de Transmissão
CR.3	EIE	<i>Error Interrupt Enable</i>	Habilitação da Interrupção por Erro
CR.4	OIE	<i>Overrun Interrupt Enable</i>	Habilitação da Interrupção por Atropelamento
CR.5	RA	<i>Reference Active</i>	Referência Activa
CR.6	S	<i>Synch</i>	Sincronização
CR.7	TM	<i>Test Mode</i>	Modo de Teste

Tabela 4.2: Conteúdo do Registo de Controlo

Pedido de Reinicialização (RR) - Quando este *bit* é posto a 1, o controlador CAN aborta a transmissão/recepção e entra no estado de reinicialização. Quando o controlador detecta uma transição de 1 para 0 neste *bit*, volta ao seu estado de operação normal.

Habilitação da Interrupção por Recepção (RIE) - Quando este *bit* é posto a 1 o controlador CAN envia um sinal de interrupção por recepção ao CPU, que indica que a recepção se fez sem erros. Quando este *bit* está a 0 não é enviado qualquer sinal.

Habilitação da Interrupção por Transmissão (TIE) - Quando este *bit* está a 1 o controlador CAN transmite um sinal de interrupção por transmissão para o CPU, a indicar que a transmissão foi bem sucedida ou que o *Buffer de Transmissão* está outra vez disponível (após um comando de Abortar Transmissão). Quando este *bit* está a 0 não há transmissão de nenhum sinal deste tipo.

Habilitação da Interrupção por Erro (EIE) - Quando este *bit* está a 1 o CPU recebe um sinal de interrupção por erro se o estado do erro ou o estado do barramento (Registo de Estado) mudarem. No caso contrário o CPU não recebe este sinal.

Habilitação da Interrupção por Atropelamento (OIE) - Se este *bit* estiver a 1 e o *bit* de Atropelamento de Dados for colocado em 1, o CPU recebe um sinal de Interrupção por Atropelamento de Dados. No caso contrário, não é enviado este sinal.

Referência Activa (RA) - Quando este pino está a 1, o pino REF é uma saída, com uma tensão de referência $AV_{DD}/2$. Quando este pino está a 0, a tensão de referência é uma entrada. Este *bit* só pode ser modificado quando o *bit* RR estiver a 1 (reinicialização).

Sincronização (S) - Se este *bit* estiver a 1, a resincronização é feita em dois tipos de transições: transições dominante-recessivo e recessivo-dominante. No caso contrário, a resincronização é feita apenas num tipo de transição: a transição recessivo-dominante. Este *bit* só pode ser modificado quando o *bit* RR estiver a 1.

Modo de Teste (TM) - Quando este *bit* está a 1, o controlador CAN entra no Modo de Teste, que impossibilita o funcionamento normal. Se estiver a 0, o controlador CAN encontra-se no modo normal de funcionamento (este modo endereça os testes do produtor e não os do consumidor).

Registo de Comando

Um *bit* de comando inicializa uma acção na camada de transferência do controlador CAN. Este registo está descrito na tabela 4.3 e aparece ao CPU como uma memória de leitura/escrita, à excepção dos *bit* CMR.0 a CMR.3, que retornam 1 quando lidos.

Bit	Símbolo	Nome em Inglês	Designação adoptada
CMR.0	TR	<i>Transmission Request</i>	Pedido de Transmissão
CMR.1	AT	<i>Abort Transmission</i>	Abortar Transmissão
CMR.2	RRB	<i>Release Receive Buffer</i>	Libertar o <i>Buffer de Recepção</i>
CMR.3	COS	<i>Clear Overrun</i>	Eliminar o Atropelamento
CMR.4	SLP	<i>Sleep</i>	Adormecido
CMR.5	WUM	<i>Wake-up Mode</i>	Modo de Acordar
CMR.6	RX1A	RX1 Active	RX1 Activo
CMR.7	RX0A	RX0 Active	RX0 Activo

Tabela 4.3: Conteúdo do Registo de Comando

Pedido de Transmissão (TR) - É transmitida uma mensagem quando está a 1. Se este *bit* tiver sido posto a 1 num comando anterior, a transmissão só pode ser cancelada quando o *bit* Abortar Transmissão a 1.

Abortar Transmissão (AT) - A colocação deste *bit* em 1 pelo CPU cancela um Pedido de Transmissão pendente, se este ainda não estiver em progresso. O *bit* AT é usado pelo CPU quando surge a necessidade de transmitir uma mensagem urgente. Como uma transmissão em progresso não é cancelada, é preciso ver o *bit* Estado da Transmissão Completa (Registo de Estado), para averiguar se a mensagem foi transmitida ou abortada.

Libertar Buffer de Recepção (RRB) - Depois do conteúdo do *buffer* RB0 ou RB1 ter sido lido, o CPU põe este *bit* a 1 para libertar o *Buffer de Recepção*.

Limpar o Estado de Atropelamento (COS) - Este *bit* de comando é utilizado para reconhecer a condição de atropelamento de dados, sinalizada pelo *bit* Atropelamento de Dados do Registo de Estado. Se este *bit* estiver a 1, então o *bit* Atropelamento de Dados é posto a 0.

Adormecido (SLP) - Se este *bit* estiver a 1, se não houver actividade no barramento e se não houver nenhuma interrupção pendente, o controlador CAN entra no modo Adormecido. O CAN acorda quando este *bit* for posto a 0 ou quando surgir actividade no barramento. Ao acordar, é gerado um sinal de Interrupção ao Acordar. Um controlador CAN que é acordado por actividade no barramento (uma mensagem) não pode receber esta mensagem até detectar um sinal de barramento livre. Quando se pretende saber o estado do controlador CAN basta ler este *bit*.

Modo de Acordar (WUM) - Este *bit* deve ser escrito ao mesmo tempo que o *bit* SLP. O Modo de Acordar diferencial (*bit* a 0) é útil quando ambos os cabos do barramento estão completamente operacionais, uma vez que minimiza o acordar devido a ruído. O Modo de Acordar un-

lateral (*bit* a 1) é recomendado quando se pretende que seja possível acordar o controlador CAN, mesmo que um dos cabos não esteja nas melhores condições.

RX1 Activo e RX0 Activo (RX1A e RX0A) - Estes *bit* identificam o estado dos respectivos interruptores. Recomenda-se mudar o estado deste *bit* apenas durante a reinicialização.

A figura 4.22 apresenta o esquema do receptor CAN. Este receptor é configurável através dos *bit* de comando RX0A e RX1A, que modificam o estado dos interruptores S0 e S1. O interruptor S2 é controlado pelo *bit* que define o Modo de Acordar (WUM).

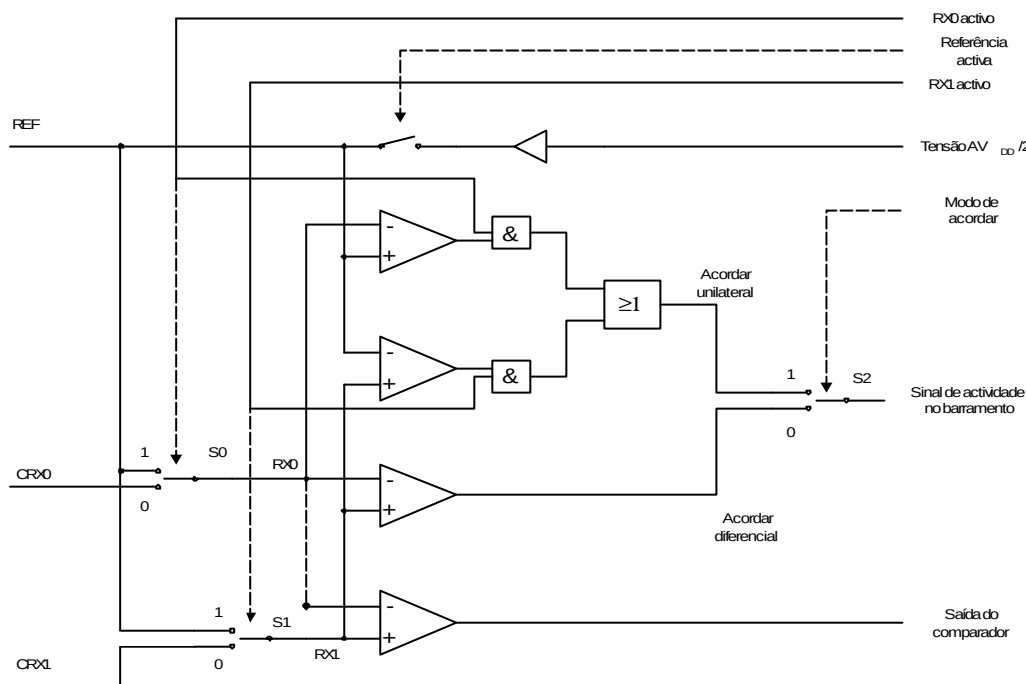


Figura 4.22: Receptor CAN configurável

A tabela 4.4 apresenta o efeito que os *bit* RX0A e RX1A têm sobre o estado das entradas do controlador CAN. Dependendo da combinação destes *bit*, as entradas do controlador CAN registam a sequência de *bit* do barramento ou a tensão de referência.

RX0A	RX1A	RX0	RX1
1	1	CRX0	CRX1
1	0	CRX0	AV _{DD} /2
0	1	AV _{DD} /2	CRX1
0	0	sem acção	sem acção

Tabela 4.4: Combinação dos *bit* RX0A e RX1A

Registo de Estado

Este registo está descrito na tabela 4.5 e reflecte o estado do controlador CAN, aparecendo ao CPU como uma memória de leitura. Ao consultar este registo, o CPU pode determinar se existe

uma nova mensagem no *Buffer de Recepção* ou se já pode escrever uma nova mensagem no *Buffer de Transmissão*. Pode igualmente averiguar se o controlador está a transmitir uma mensagem, a receber uma mensagem, ou se não está envolvido em nenhuma actividade no barramento.

Bit	Símbolo	Nome em Inglês	Designação adoptada
CS.0	RBS	<i>Receive Buffer Status</i>	Estado do <i>Buffer de Recepção</i>
CS.1	DO	<i>Data Overrun</i>	Atropelamento de Dados
CS.2	TBS	<i>Transmit Buffer Status</i>	Estado do <i>Buffer de Transmissão</i>
CS.3	TCS	<i>Transmit Complete Status</i>	Estado da Transmissão completa
CS.4	RS	<i>Receive Status</i>	Estado da Recepção
CS.5	TS	<i>Transmit Status</i>	Estado da Transmissão
CS.6	ES	<i>Error Status</i>	Estado do Erro
CS.7	BS	<i>Bus Status</i>	Estado do Barramento

Tabela 4.5: Conteúdo do Registo de Estado

Estado do Buffer de Recepção (RBS) - Este *bit* é posto a 1 quando uma nova mensagem está disponível (este *bit* a 0 quer dizer que o *Buffer de Recepção* está vazio). Se o *bit* de comando RRB é posto a 1 pelo CPU, este *bit* é posto a 0 pelo IML. Quando uma nova mensagem for armazenada num dos *Buffers de Recepção* este *bit* é novamente posto a 1.

Atropelamento de Dados (DO) - Este *bit* é posto a 1 quando ambos os *Buffers de Recepção* estão preenchidos e se tenta escrever o primeiro *byte* da próxima mensagem. Quando está 0 quer dizer que não ocorreu nenhum Atropelamento de Dados, desde que o comando de Limpar o Atropelamento de Dados foi emitido.

Acesso ao Buffer de Transmissão (TBS) - Se o CPU tentar escrever no *Buffer de Transmissão* quando este *buffer* estiver a 0, os *bytes* escritos não serão aceites e serão perdidos, sem que o facto seja sinalizado. Se este *bit* estiver a 1, o CPU pode escrever no *Buffer de Transmissão*.

Estado da Transmissão Completa (TCS) - Um 1 neste *bit* indica que a última transmissão solicitada foi completada com sucesso. Se *bit* estiver a 0 a transmissão ainda não foi completada. Este *bit* está a 0 sempre que o *bit* Pedido de Transmissão (Registo de Comando) estiver a 1. Se for emitido um comando de Abortar Transmissão (Registo de Comando), *Buffer de Transmissão* será libertado. Se a mensagem, que foi solicitada e depois abortada, não foi transmitida, este *bit* permanece a 0.

Estado da Recepção (RS) - Quando este *bit* estiver a 1, o controlador CAN está a receber uma mensagem.

Estado da Transmissão (TS) - Se este *bit* estiver a 1, o controlador CAN está a transmitir uma mensagem. Se este *bit* e o anterior estiverem a 0, o barramento está inactivo.

Estado do Erro (ES) – Um 1 neste *bit* (erro) indica que pelo menos um dos Contadores de Erros chegou ao limite de aviso do CPU. Se estiver a 0, quer dizer que ambos os contadores de erros estão aquém do limite de aviso.

Estado do Barramento (BS) - Quando este *bit* é posto a 1, o controlador CAN vai pôr o *bit* Pedido de Reinicialização (Registo de Controle) a 1. Este *bit* só será posto a 0 depois que o CPU mudar o *bit* de Pedido de Reinicialização para 0 e depois de esperar 128 ocorrências do sinal barramento livre. Nesta altura, o *bit* de Estado do Erro tem que ser posto a 0 e os Contadores de Erros têm que ser limpos.

Registo de Interrupção

Este registo está descrito na tabela 4.6 e permite a identificação de uma fonte de interrupção. Quando um ou mais *bits* deste registo são postos a 1, é gerada uma interrupção para o CPU por parte do CAN. Depois de os *bits* terem sido lidos pelo CPU, este registo é reinicializado. Este registo aparece ao CPU como uma memória apenas de leitura.

Bit	Símbolo	Nome em Inglês	Designação adoptada
IR.0	RI	<i>Receive Interrupt</i>	Interrupção de Recepção
IR.1	TI	<i>Transmit Interrupt</i>	Interrupção de Transmissão
IR.2	EI	<i>Error Interrupt</i>	Interrupção de Erro
IR.3	OI	<i>Overrun Interrupt</i>	Interrupção por Atropelamento
IR.4	WUI	<i>Wake-up Interrupt</i>	Interrupção ao Acordar
IR.5	-	-	Reservado
IR.6	-	-	Reservado
IR.7	-	-	Reservado

Tabela 4.6: Conteúdo do Registo de Interrupção

Interrupção de Recepção (RI) - Este *bit* é posto a 1 quando existe uma mensagem no *Buffer de Recepção* e o *bit* habilitação da Interrupção por Recepção (RIE - Registo de Controlo) está ver igualmente a 1. Este *bit* é posto automaticamente a 0 depois de um acesso de leitura do CPU ao Registo de Interrupção. O mesmo se passará para os *bits* seguintes.

Interrupção de Transmissão (TI) - Este *bit* é posto a 1 quando é detectada a transição de 0 para 1 no *bit* Acesso ao Buffer de Transmissão (TBS - Registo de Estado), se o *bit* Habilitação da Interrupção de Transmissão (TIE - Registo de Controlo) estiver a 1.

Interrupção por erro (EI) - Se o *bit* Habilitação da Interrupção por Erro (EIE - Registo de Controlo) estiver a 1, este *bit* fica a 1 quando é detectada uma mudança em um dos *bits* Estado do Erro ou Estado do Barramento (Registo de Estado).

³ Se um dos Contadores de Erros exceder o limite de aviso, que é de 96 pontos de erro, este facto é sinalizado através de uma interrupção por erro.

Interrupção por Atropelamento(OI) - Este *bit* é posto a 1 se ambos os *Buffers de Recepção* contiverem uma mensagem, o primeiro *byte* de outra mensagem tiver que ser guardado e o *bit Habilitação da Interrupção por Atropelamento(OIE* do Registo de Controlo) estiver a 1.

Interrupção ao Acordar(WUI) - Este *bit* é posto a 1 quando o modo de Adormecido (SLP do Registo de Comando) é abandonado.

Registo do código de aceitação

Este registo faz parte do filtro de aceitação do controlador CAN e só pode ser lido ou escrito se o *bit* Pedido de Reinicialização estiver a 1. Quando é recebida uma mensagem e esta passa o teste de aceitação, existindo um *Buffer de Recepção* vazio, o descritor e o Campo de Dados da mensagem são guardados sequencialmente no *buffer* vazio. Caso este não esteja disponível, o *bit Atropelamento de Dados* é posto a 1. Após a recepção da totalidade da mensagem, o *bit Estado do Buffer de Recepção* é posto a 1. Se o *bit Habilitação da Interrupção de Recepção* estiver a 1, o *bit Interrupção por Recepção* é posto igualmente a 1.

Os *bit* do Código de Aceitação (AC.7-AC.0) e os *bit* mais significativos do Identificador de Mensagem têm que ser iguais (ID.10-ID.3), naqueles *bit* que a Máscara de Aceitação assinala como relevantes (AM.7-AM.0). Se a seguinte equação for satisfeita, a mensagem é aceite pela estação:

$$[(ID.10...ID.3) \text{ igual a } (AC.7...AC.0)] \quad \text{ou} \quad (AM.7...AM.0) = 1111 \ 1111_B$$

Registo da Máscara de Aceitação

Este registo faz parte do filtro de aceitação do controlador CAN e qualifica quais os *bit* do código de aceitação são relevantes (0) ou não tem interesse (1), para a aceitação de uma mensagem. Este registo só pode ser acedido (lido/escrito) quando o *bit* Pedido de Reinicialização estiver a 1.

Registo 0 de Regulação do Tempo do Barramento

O conteúdo deste registo está descrito na tabela 4.7 e define o valor do pré-escalador do *baud rate* (BRP) e da largura do salto de sincronização (SJW). Este registo só pode ser acedido (lido/escrito) quando o *bit* de Pedido de Reinicialização estiver a 1.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0

Tabela 4.7: Conteúdo do registo 0 de regulação do tempo do barramento

O Pré-escalador do *Baud Rate* programa o período do relógio do sistema e determina o tempo do *bit* individual, sendo aquele calculado pela seguinte expressão:

$$t_{scf} = 2 * t_{clk} * (32 * BRP.5 + 16 * BRP.4 + 8 * BRP.3 + 4 * BRP.2 + 2 * BRP.1 + BRP.0 + 1)$$

(onde t_{clk} = período de tempo do relógio do controlador CAN)

A Largura do Salto de Sincronização pretende compensar os desvios de fase dos osciladores dos relógios dos diferentes controladores CAN, que tornam necessária a resincronização por qualquer transição relevante da transmissão. Os dois *bits* da Largura do Salto de Sincronização definem o número máximo de ciclos de relógio que *um bit* pode ser aumentado ou diminuído, aquando da resincronização:

$$T_{sjw} = t_{scf} * (2 * SJW.1 + SJW.0 + 1)$$

Registo 1 de Regulação do Tempo de Barramento

O conteúdo deste registo está descrito na tabela 4.8 e define o comprimento do período *bit* e a localização do ponto de amostragem (TSEG1 e TSEG2), bem como quantas amostras devem ser tiradas para cada ponto de amostragem. Este registo só pode ser acedido (lido/escrito) se o *bit* Pedido de Reinicialização estiver a 1.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SAM	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0

Tabela 4.8: Conteúdo do registo 1 de regulação do tempo do barramento

Quando a Amostragem (SAM) estiver a 1 são tiradas três amostras, sendo tirada apenas uma quando estiver a 0. O primeiro tipo de amostragem é recomendado para um barramento de alta velocidade, sendo o outro para um barramento de baixa e média velocidade, onde a filtragem de picos no barramento é importante. A determinação destes parâmetros é feita de acordo com as seguintes desigualdades:

$$TSEG1 = t_{CL} * (8 * TSEG1.3 + 4 * TSEG1.2 + 2 * TSEG1.1 + TSEG1.0 + 1)$$

$$TSEG2 = t_{CL} * (4 * TSEG2.2 + 2 * TSEG2.1 + TSEG2.0 + 1)$$

Registo de Controlo da saída

Este registo está descrito na tabela 4.9 e permite controlar por *software* a configuração dos *drivers* de saída. O acesso a este registo só é possível se *bit* Pedido de Reinicialização estiver a 1. Se o controlador estiver no modo Adormecido (SLP=1), um nível recessivo é colocado nos pinos CTX0 e CTX1. Se estiver no Estado de Reinicialização (RR = 1), os *drivers* de saída estarão em alta impedância.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCMODE1	OCMODE0

Tabela 4.9: Conteúdo do registo de controlo da saída

A figura 4.23 ilustra a relação entre os *bits* do registo de controlo e a estrutura dos andares de saída, que suportam quatro modos de funcionamento: Modo Normal, Modo de Relógio, Modo Bifase e Modo de Teste.

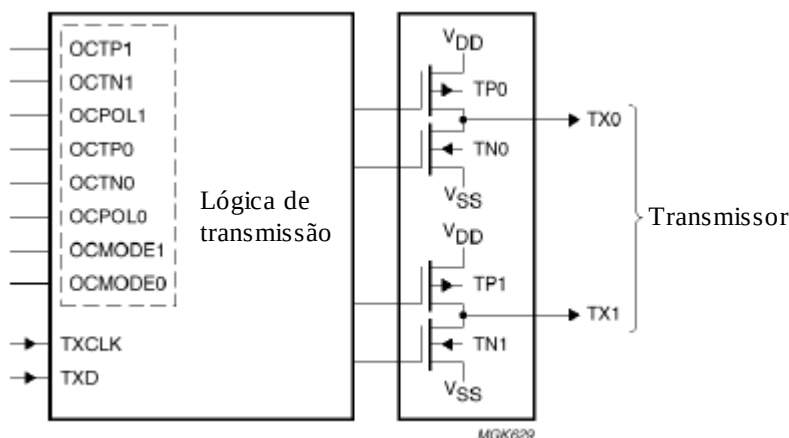


Figura 4.23: Registo de controlo da saída e andar de saída

Modo Normal - Neste modo a sequência *debt* (TXD) é enviada através de CTX0 e CTX1. Os níveis de tensão nesses pinos dependem das características dos *drivers*, programadas por OCTPx⁴ e por OCTNx (*float, pull-up, pull-down, push-pull*), e da polaridade de saída, programada por OCPOLx.

Modo de Relógio - Para o pino CTX0, é o mesmo que no modo normal. No entanto, para o pino CTX1, os dados são substituídos pelo relógio de transmissão (TXCLK). A transição a cendente do relógio de transmissão marca o início do período *debt*. A largura do impulso de relógio é dada por t_{cl} .

Modo Bifase - Neste modo, a representação dos *bits* é variável com o tempo e alternada. Se os controladores do barramento estiverem isolados galvanicamente por um transformador, a sequência de *bit* não pode ter uma componente contínua. Isto é conseguido da seguinte forma: durante os *bit* recessivos todas as saídas são desactivadas, ficando a flutuar (alta impedância). Os *bit* dominantes são enviados alternadamente por CTX0 e CTX1, ou seja, o primeiro *bit* dominante é enviado por CTX0, o segundo é enviado por CTX1, o terceiro é enviado por CTX0, etc.

⁴ O x nesta designação *debt* do Registo de Controlo da Saída tem o significado de um 0 ou um 1, ou seja, o valor dos dois *bit* com o mesmo nome e terminação diferente é o mesmo.

Modo de Teste - Este modo serve apenas para medir o tempo de atraso do emissor e receptor. Para isso liga a entrada do driver de saída CTX1 com a saída do comparador de entrada (COMP OUT). Para o pino CTX0 é equivalente ao Modo Normal.

Os *bit* OCMODE0 e OCMODE1 controlam o modo de saída. A tabela 4.10 apresenta a relação entre o valor destes *bit* e as duas saídas série (CTX0 e CTX1) ligadas ao barramento.

OCMODE1	OCMODE0	Descrição
1	0	Modo normal; CTX0, CTX1: sequência de <i>bit</i>
1	1	Modo de relógio; CTX0: sequência de <i>bit</i> ; CTX1: relógio do barramento
0	0	Modo bifase
0	1	Modo de Teste; CTX0: sequência de <i>bit</i> , CTX1: saída do comparador

Tabela 4.10: Efeito dos *bit* OCMODE1 e OCMODE0

Registo de Teste

O Registo de Teste é apenas usado para os testes de produção e o significado dos *bit* que o compõem está descrito na tabela 4.11:

Bit	Símbolo	Nome em Inglês	Designação adoptada
TR.0,1	-	-	Reservado
TR.2	MIR	Map Internal Register	Mapear Registo Interno
TR.3	CRB0	Connect RX Buffer 0 CPU	Ligar o Buffer 0 RX ao CPU
TR.4	CTBC	Connect TX Buffer CPU	Ligar o Buffer TX ao CPU
TR.5	AIB	Access Internal Bus	Aceder Barramento Interno
TR.6	NRC	Normal RAM Connect	Ligação Normal da RAM
TR.7	FOD	Float Output Driver	Driver de Saída a Flutuar

Tabela 4.11: Conteúdo do registo de teste

Buffer de Transmissão

O Buffer de Transmissão serve para armazenar uma mensagem a ser transmitida pelo controlador CAN e está dividido em Descritor e Campo de Dados. O Buffer de Transmissão está representado na tabela 4.12 e pode ser lido e escrito pelo CPU.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Endereço 10: Identificador (ID Identifier)							
ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
Endereço 11: DLC, Código de Comprimento de Dados (DLC Data Length Code)							
ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
Endereços 12 até 19: Bytes 1 até 8							
Dados	Dados	Dados	Dados	Dados	Dados	Dados	Dados

Tabela 4.12: Buffer de Transmissão

Identificador (ID) - é composto pelos *bits* ID.0 a ID.9 contidos nos descritores 1 e 2. O *bit* ID.9 é o mais significativo, sendo transmitido em primeiro lugar no processo de arbitragem. O identificador actua como um nome de mensagem, usado num receptor para filtragem de aceitação e determinando, igualmente, a prioridade de acesso ao barramento durante o processo de arbitragem. Quanto mais baixo for o valor binário do identificador, maior é a sua prioridade.

Bit de Pedido de Transmissão remota - quando o *bit* RTR está a 1, é transmitida uma Trama Remota pelo controlador CAN. Quando o *bit* RTR está a 0, é transmitida uma Trama de Dados.

Código de Comprimento de Dados (DLC) - o número de *bytes* (*Data Byte Count*), no Campo de Dados da mensagem, é codificado pelo Código de Comprimento de Dados. No início da transmissão de uma Trama Remota o DLC não é considerado, uma vez que o *bit* RTR está a 1, o que força a que o número de *bytes* de dados transmitidos ou recebidos seja 0. No entanto, o DLC deve ser especificado correctamente para evitar erros no barramento, prevendo o caso de dois controladores CAN começarem a transmissão de uma Trama Remota simultaneamente. O alcance do Contador de *Bytes* de Dados (DBC, *Data Byte Count*) é de 0 a 8 *bytes*, de acordo com a seguinte expressão:

$$DBC = 8 * DLC.3 + 4 * DLC.2 + 2 * DLC.1 + DLC.0$$

Por razões de compatibilidade, só deverão ser usados os valores 0,1,2,... e 8, para o Contador de *Bytes* de Dados.

Campo de Dados - o número de *bytes* de dados transferidos é determinado pelo Código de Comprimento de Dados (o primeiro *bit* transferido é o *bit* mais significativo do *byte* 1).

Buffer de Recepção

Existem dois *Buffers de Recepção* com uma disposição de *bytes* individuais correspondentes ao *Buffer de Transmissão* tal como se ilustra na tabela 4.13:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Endereço 20: Identificador (ID, <i>Identifier</i>)							
ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
Endereço 21: DLC, Código de Comprimento de Dados (DLC, <i>Data Length Code</i>)							
ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
Endereços 22 até 29: <i>Bytes</i> 1 até 8							
Dados	Dados	Dados	Dados	Dados	Dados	Dados	Dados

Tabela 4.13: Composição dos *Buffers de Recepção*

Cálculos de inicialização do controlador CAN

A Lógica do Tempo de Bit (BTL, ver figura 4.20) do controlador CAN monitoriza o barramento através do comparador de entrada, executando as seguintes funções:

- Monitoriza o nível do barramento

- Amostra o nível do barramento
- Sincronização:
 - Sincronização no início de uma mensagem.
 - Ressincronização durante a transmissão da mensagem.

A configuração deste bloco é feita durante a fase de inicialização do controlador CAN, através dos seguintes registos:

- Registo de Controlo (*Synch*)
- Registo 0 de Regulação do Tempo do Barramento
- Registo 1 de Regulação do Tempo do Barramento

Regulação do Período do *Bit*

O Período de um *Bit* (*Bit Period*) é o resultado da soma dos segmentos programáveis TSEG1, TSEG2 e do segmento geral SYNCSEG, como se ilustra na figura 4.24.

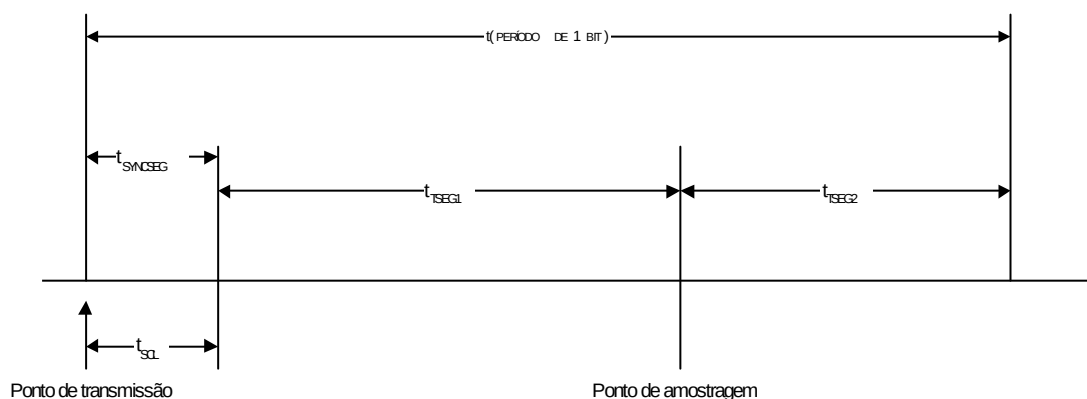


Figura 4.24: Período do *bit*

O intervalo SYNCSEG corresponde a um ciclo de relógio (t_L) e é durante este estado que se espera a transição associada a um *bit*.

O segmento de tempo 1 (t_{TSEG1}) determina a posição do ponto de amostragem dentro do Período do *Bit*, sendo programável de 1 a 16 ciclos de relógio. A localização correcta do ponto de amostragem é essencial para o funcionamento correcto da transmissão, sendo preciso ter em atenção três pontos:

- Campo de Início de Trama obriga todos os controladores a fazerem uma sincronização na primeira transição recessivo-dominante. No entanto, durante o Campo de Arbitragem, vários controladores podem começar a transmitir simultaneamente, sendo necessário um atraso até que o barramento fique estável: o Tempo de Atraso de Propagação (t_{PROP} , *Propagation Delay Time*)⁵.

⁵ $t_{PROP} = 2 * (\text{atraso do barramento físico} + \text{atraso do comparador de entrada} + \text{atraso driver de saída})$, arredondado ao múltiplo mais próximo de t_{SCL} .

- Para evitar a amostragem num ponto incorrecto é necessário incluir um espaço de sincronização em ambos os lados do ponto de amostragem. As principais razões para uma amostragem errada são:
 - Sincronização incorrecta devido a picos no barramento.
 - Variações na frequência dos osciladores dos vários controladores, o que resulta num erro de fase.

Sendo assim, o TSEG1 é constituído por um segmento de compensação dos atrasos de propagação e por um espaço de sincronização, antes do ponto de amostragem.

O segmento de tempo 2 é programável de 1 a 8 ciclos de relógio e proporciona:

- Um tempo adicional no ponto de amostragem, para o cálculo dos níveis *bit* seguintes.
- Um espaço de sincronização depois do ponto de amostragem.

Largura do Salto de Sincronização

A largura do salto de sincronização (SJW) define o número máximo de ciclos de relógio que um *bit* pode ser aumentado ou diminuído, aquando da resincronização. O SJW pode ser programado de 1 a 4 ciclos do relógio do sistema.

Restrições à regulação do período do *bit*

Para calcular os tempos t_{SEG1} e t_{SEG2} é necessário ter em conta certas restrições baseadas no processamento interno, nomeadamente as que são expressas pelas seguintes desigualdades:

$$t_{SEG2} \geq 2 * t_{SCL}$$

$$t_{SEG2} \geq t_{SJW}$$

$$t_{SEG1} \geq t_{TSEG2}$$

$$t_{SEG1} \geq t_{SJW} + t_{PROP}$$

Quando se considera o SAM = 1, ou seja, quando o controlador tira 3 amostras, o barramento sofre um atraso de um ciclo de relógio, sendo necessário modificar duas restrições:

$$t_{SEG2} \geq 3 * t_{SCL}$$

$$t_{SEG1} \geq t_{SJW} + t_{PROP} + 2 * t_{SCL}$$

Sincronização

Aquando da resincronização pode acontecer um de dois fenómenos: t_{SEG1} pode ser aumentado até ao máximo de t_{SJW} , ou t_{SEG2} pode ser diminuído até ao máximo de t_{SJW} . Sendo assim, é necessário ter em atenção as seguintes restrições:

$$t_{SEG1} \leq t_{SCL} * [(TSEG1+1)+(SJW+1)]$$

$$t_{SEG1} \geq t_{SCL} * [(TSEG2+1)-(SJW+1)]$$

(sendo TSEG1, TSEG2 e SJW os valores numéricos programados)

Cálculos dos parâmetros

Para calcular os parâmetros de transmissão do CAN é necessário começar por decidir qual vai ser a velocidade de transmissão do controlador. Considerando que:

$$baud\ rate = 1 / t_{(período\ de\ umbit)}$$

e que:

$$t_{(período\ de\ umbit)} = t_{SCL} + t_{SEG1} + t_{SEG2}$$

e atendendo às restrições enunciadas anteriormente, é possível calcular o valor dos registos BTR0 e BTR1.

Ligação Transceptor PCA82C250

Para proteger o controlador CAN, usa-se um transceptor PCA82C250 como interface entre o controlador e o barramento físico. O 82C250 é um dispositivo de interface que tem capacidade de transmissão diferencial para o barramento e capacidade de recepção diferencial para o controlador CAN. Um circuito limitador de corrente protege a saída do transmissor contra curto-circuitos com a alimentação, evitando a destruição do andar de saída. Este dispositivo pode funcionar em três modos diferentes: Alta Velocidade, Controlo de Rampa (*slope*) e Apoio (*standby*).

No modo de Alta Velocidade, os transístores de saída do transmissor comutam o mais rapidamente possível e são tiradas medidas por forma a limitar a inclinação da subida e da descida da rampa. Deve usar-se um cabo blindado, para evitar problemas de interferência de rádio-frequência (RFI, *Radio Frequency Interference*). Este modo é seleccionado ligando o pino RS à massa.

Para velocidades menores ou para comprimentos do barramento inferiores pode usar-se um cabo não blindado de par entrançado ou um par de cabos paralelos para o barramento. Para reduzir a interferência por rádio-frequência, a inclinação da subida e da descida da rampa deve ser limitada, podendo ser programadas com uma resistência ligada ao pino RS.

Se for aplicado um nível alto no pino RS, o controlador entra num Modo de Apoio, em que o transmissor é desligado e o receptor fica num modo de baixa corrente.

Na figura 4.25 apresenta-se o diagrama de blocos do PCA82C250, cujos pinos podem ser descritos da seguinte forma:

TXD - entrada de dados a transmitir

V_{CC} - alimentação

RXD - saída de dados recebidos

V_{ref} - tensão de referência da saída

CANL - entrada/saída do controlador CAN de baixo nível de tensão

CANH - entrada/saída do controlador CAN de alto nível de tensão

RS - entrada da resistência de controlo da rampa

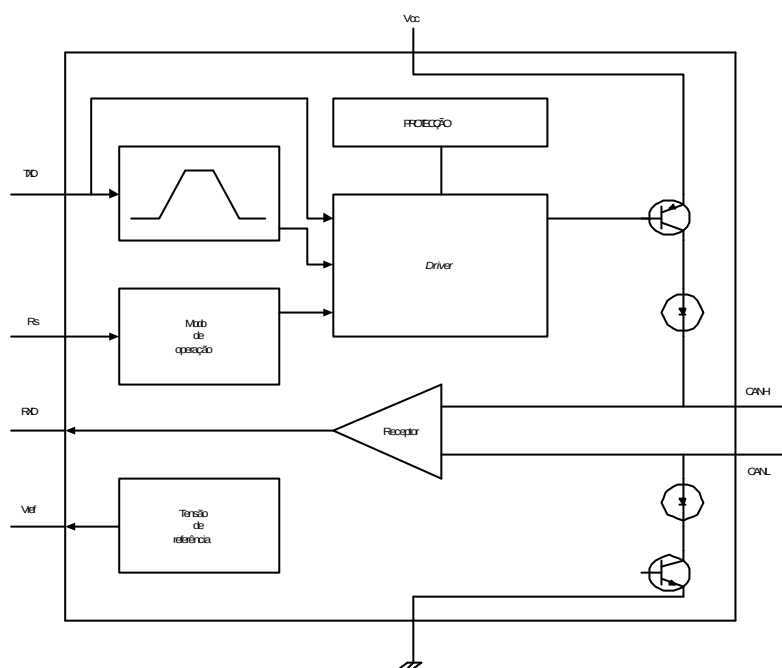


Figura 4.25: Diagrama de blocos do PCA82C250

Na figura 4.26 apresenta-se o esquema de ligação de um controlador CAN (como o PCA82C200) ao barramento, através do transceptor PCA82C250. Este tipo de ligação é recomendado para aplicações de alta velocidade (até 1 Mbps) na indústria automóvel.

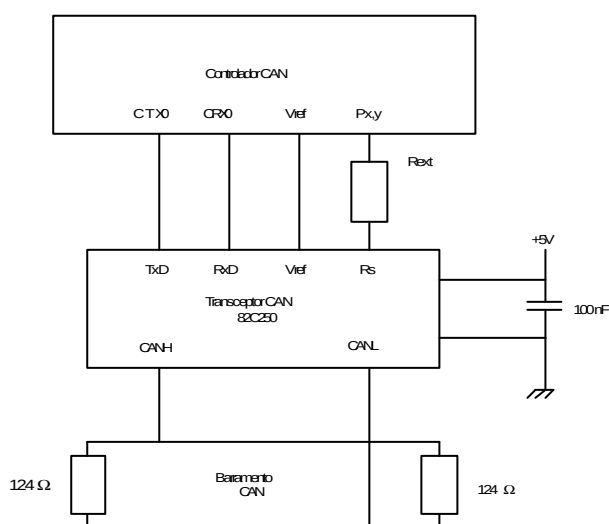


Figura 4.26: Ligação do transceptor CAN

4.5.3 Software da aplicação

Dentro de um nó CAN existem duas camadas de *software* independentes, representadas na figura 4.27, sendo uma responsável pela gestão das comunicações e a outra pelo controlo da aplicação. Existem mecanismos bem definidos para a comunicação entre a camada de gestão do CAN e o seu *hardware* de suporte (controlador PCA82C200), bem como a um nível mais elevado, entre a camada de aplicação e a camada de gestão CAN.

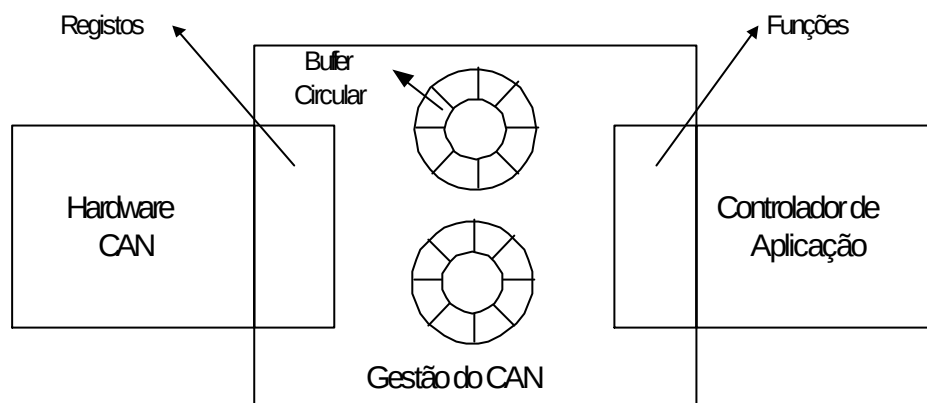


Figura 4.27: Código presente em cada nó CAN

Nos parágrafos seguintes abordaremos o *software* de gestão da comunicação e as regras de construção de uma aplicação de controlo num nó CAN.

Gestão da comunicação CAN

Esta camada de *software* é composta por doze funções, divididas em quatro categorias principais:

- Acesso ao *hardware*
- Inicialização
- Troca de mensagens
- Monitorização

Acesso ao *hardware*

O controlador PCA82C200 está ligado ao microcontrolador 87C51FA através da porta P0, usando o modo de endereçamento indirecto por registo, para não colocar bits de endereço na porta 2, que é usada para controlo do barramento de E/S. Este modo de endereçamento corresponde em *assembly* às instruções **MOVX A,@R_i** e **MOVX @R_i,A**, mas na linguagem de programação C é necessário construir uma função ou uma macro com esta funcionalidade. **write (char pdata * addr, char value)** Esta função permite escrever no endereço *addr* o valor especificado por *value*. Trata-se portanto de uma função muito simples, cujo fluxograma está representado na figura 4.28.

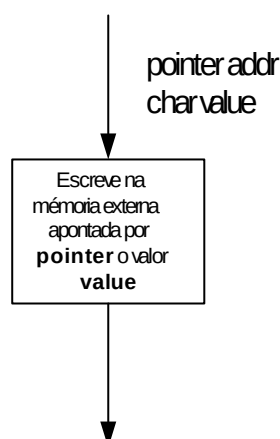


Figura 4.28: Fluxograma da função **Pwrite**

A função **char Pread (char pdata *addr)** por sua vez, retorna o valor contido na posição de memória *addr*. Esta função é complementar da anterior e pode ser representada pelo fluxograma ilustrado na figura 4.29.

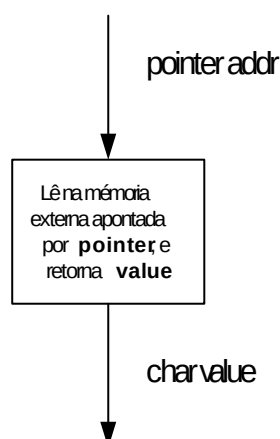


Figura 4.29: Fluxograma da função **Pread**

O atendimento da interrupção 0 do microcontrolador está associado à função **can_int(void)** que efectua a recepção e transmissão de mensagens a pedido do controlador CAN. Existe ainda a possibilidade de associar esta função ao tratamento de erros reportados pelo controlador CAN através do disparo de interrupções, mas a implementação actual não contém código para este efeito. A figura 4.30 apresenta o fluxograma desta função, onde se descreve o tipo de acção a executar, em função da informação contida no registo de interrupção do controlador CAN.

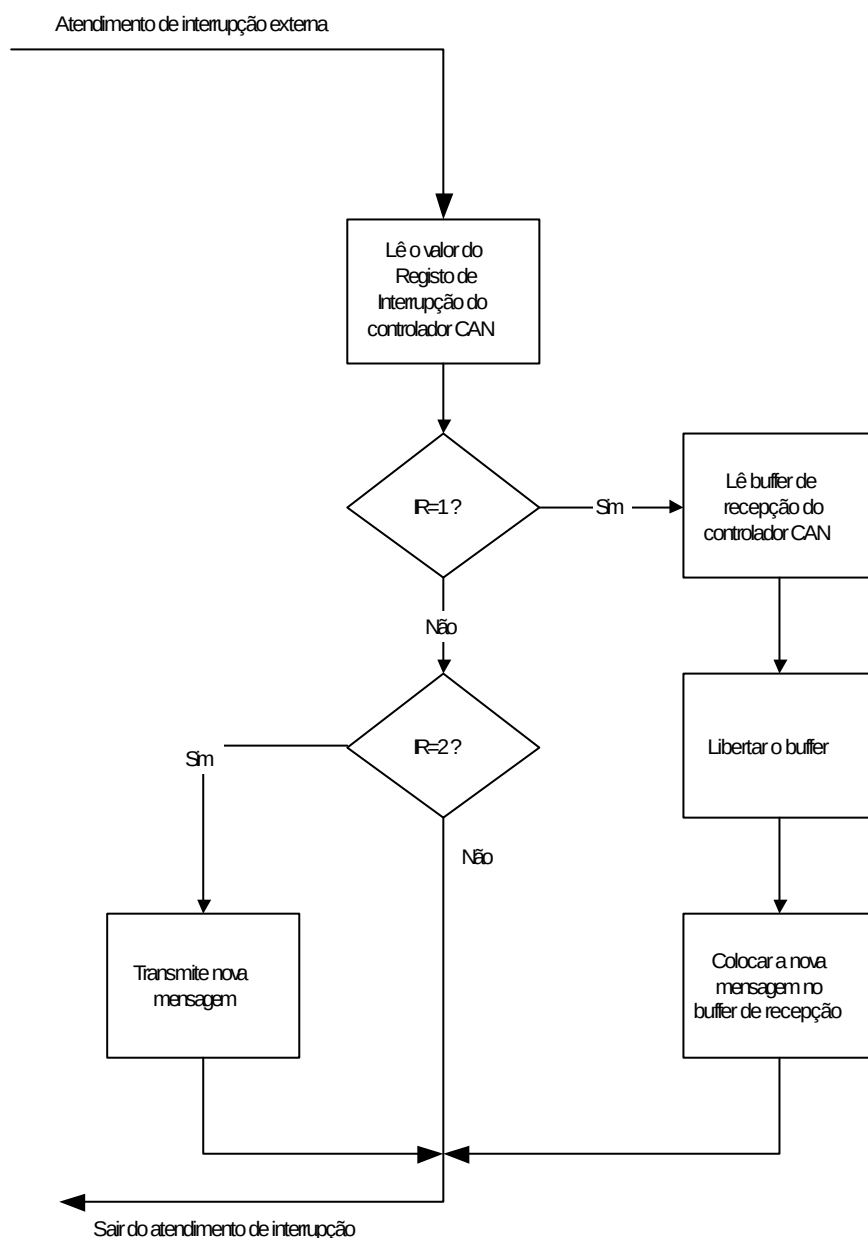


Figura 4.30: Fluxograma de **can_int**, para o atendimento de interrupções

Troca de mensagens

Estas funções estão encarregadas de realizar a interface para troca de mensagens entre a aplicação de controlo e o controlador CAN. A função **clear_can_buffer ()** limpa o *buffer* circular de recepção de mensagens, correspondendo-lhe o fluxograma ilustrado na figura 4.31.

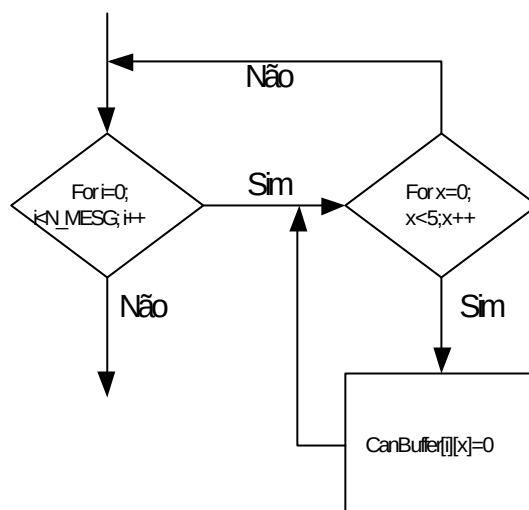


Figura 4.31: Fluxograma da função **clear_can_buffer**

A função **get_can_message (void)** remove uma mensagem do *buffer* circular e preenche as variáveis globais T_TYPE, T_FROM, T_TO, T_COM e T_VAL, com os valores destas mensagens. A figura 4.32 representa o fluxograma desta função, no qual é importante notar o mecanismo de apontadores de início e fim do *buffer* circular (CB_RP e CB_WP).

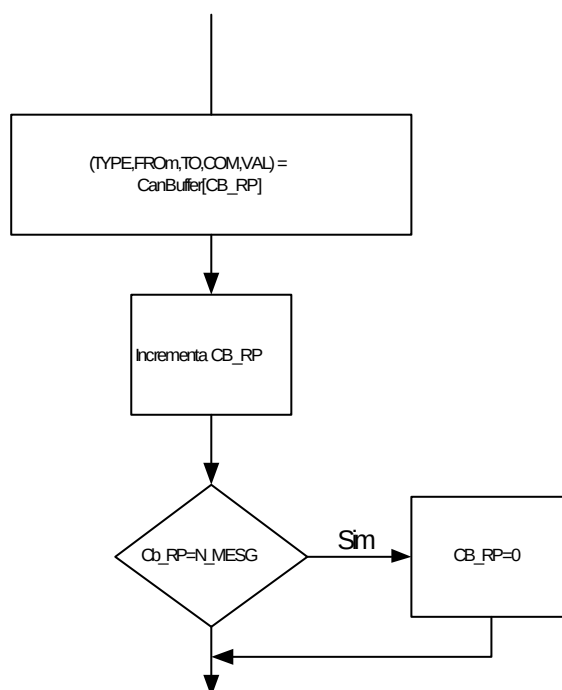


Figura 4.32: Fluxograma da função **get_can_message**

A função **send_can_message()** envia uma mensagem ao controlador CAN, com os valores das variáveis globais T_TYPE, T_TD, T_TO, T_COM e T_VAL. O fluxograma representado na figura 4.33 descreve os passos dados por esta função, nomeadamente “pooling” ao estado do

buffer de transmissão do controlador CAN, através da leitura do Registo de Estado, esperando que este fique livre para iniciar o processo de escrita da mensagem, seguido por um comando de início de transmissão.

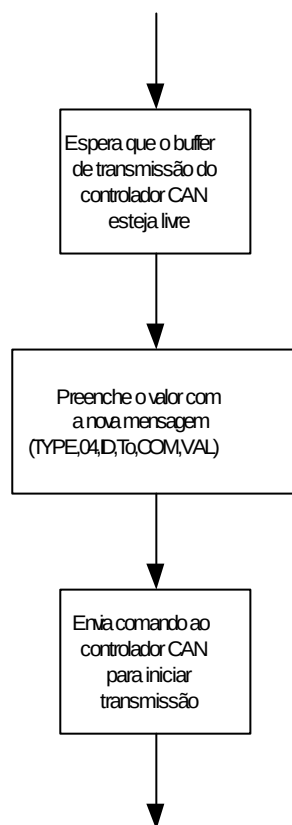


Figura 4.33: Fluxograma da função **send_can_message**

Monitorização

As funções que se seguem são do tipo booleano e servem para obter indicações sobre o estado de funcionamento do controlador CAN. Trata-se de funções muito simples, que apenas verificam o estado dos bits do registo de estado (SR), retornando o valor *verdadeiro* ou *falso*. A função **can_receive ()** indica que o controlador CAN se encontra no modo de recepção de mensagens. A figura 4.34 ilustra o fluxograma desta função, cujo resultado depende do quinto bit do registo de estado (SR).

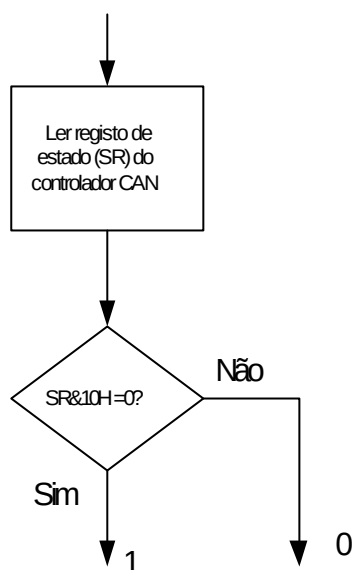


Figura 4.34: Fluxograma da função **can_receive**

A função **can_transmit ()** indica que o controlador CAN se encontra a transmitir uma mensagem. A figura 4.35 ilustra o fluxograma desta função, cujo resultado depende do sexto bit do registo de estado (SR).

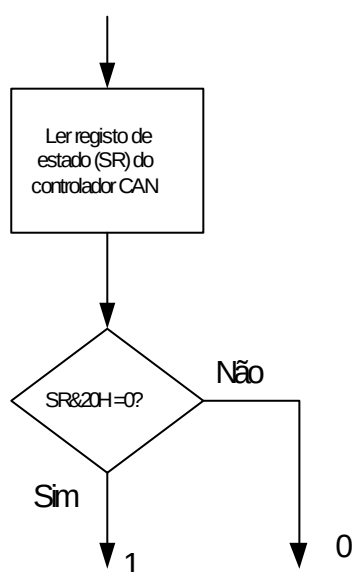


Figura 4.35: Fluxograma da função **can_transmit**

A função **can_error ()** indica que foi detectado um erro no controlador CAN. A figura 4.36 ilustra o fluxograma desta função, cujo resultado depende do sétimo bit do registo de estado (SR).

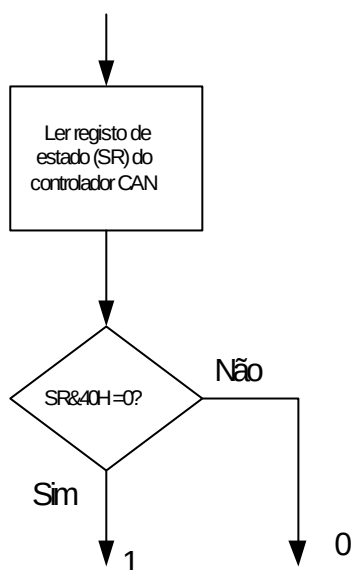


Figura 4.36: Fluxograma da função **can_error**

A função **can_busy ()** indica que ambos os *buffers* de recepção se encontram cheios e que o controlador não pode receber mais mensagens. A figura 4.37 ilustra o fluxograma desta função, cujo resultado depende do oitavo bit do registo de estado (SR).

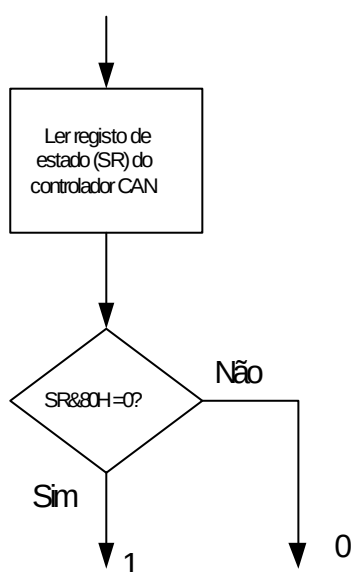


Figura 4.37: Fluxograma da função **can_busy**

A função **new_can_message ()** é funcionalmente idêntica às anteriores, mas indica o estado dos *buffers* circulares de recepção, nomeadamente no que respeita à existência ou não de **alg** ma mensagem disponível para leitura. A figura 4.38 apresenta o fluxograma desta função, cujo resultado é verdadeiro sempre que o apontador de início do *buffer* difere do apontador de fim.

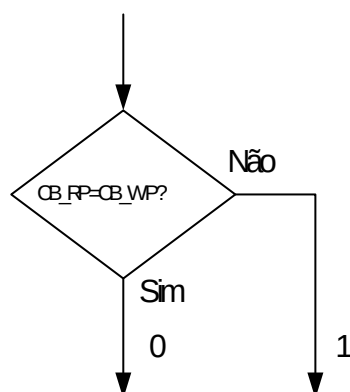


Figura 4.38: Fluxograma da função **new_can_message**

Configuração

A função **can_setup ()** é usada uma vez para configurar o controlador CAN com *baudrate* desejado e para configurar as portas de recepção e transmissão, de acordo com a configuração física da ligação ao PCA820C50. A figura 4.39 descreve os passos efectuados por esta função.

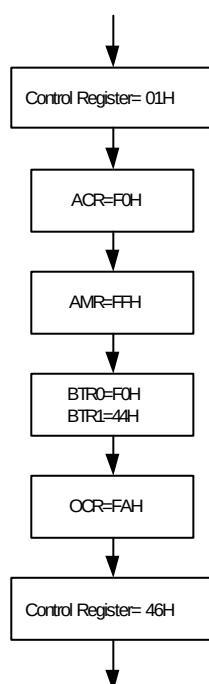


Figura 4.39: Fluxograma da função **can_setup**

Software de controlo de aplicação

Este *software* constitui a verdadeira camada de aplicação do código interno do CAN, contendo o algoritmo de controlo que caracteriza o comportamento de um nó. A diferença entre um nó CAN para controlo de iluminação, face a outro para o controlo da abertura e fecho de portas,

reside no algoritmo de controlo, que utiliza os recursos disponíveis de forma diferente (estes recursos dizem respeito à comunicação e às linhas das portas de E/S que estão ligadas a sensores e a actuadores). A camada de aplicação, representada na figura 4.40, pode ser dividida em cinco blocos funcionais, correspondentes à inicialização, à leitura de entradas, ao algoritmo de controlo, à actualização de saídas e à leitura de mensagens CAN.

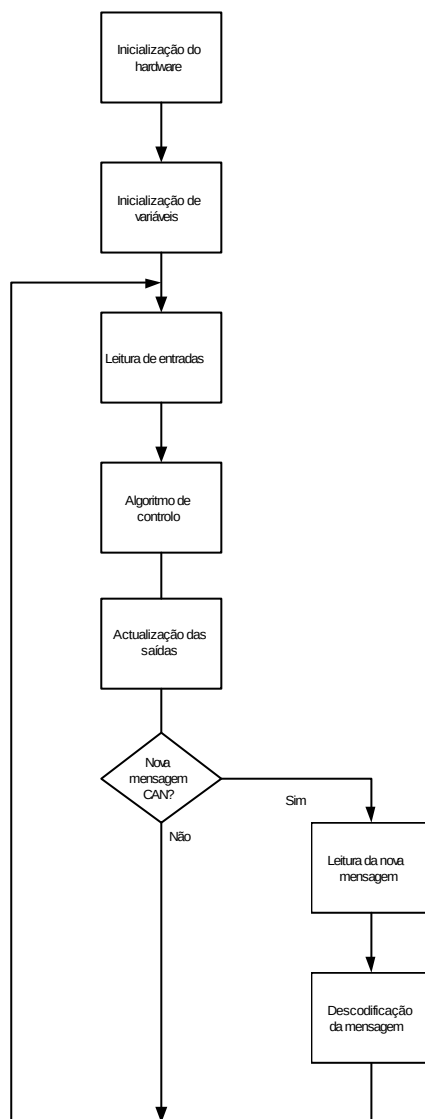


Figura 4.40: Fluxograma da camada de aplicação

Inicialização do hardware

O código para a inicialização efectua a leitura do Identificador da estação, a partir do interruptor DIP de selecção, por forma a configurar o controlador CAN e os timers do microcontrolador, e permitir o disparo de interrupções. Em relação aos timers, existe uma variável de 16 bit chamada *Timer* e outra chamada *Tempo*, que são decrementadas em intervalos regulares de 10

ms, até atingirem o valor 0. Estas variáveis são usadas para efeitos de temporização não bloqueante.

Inicialização de variáveis

Após a inicialização do hardware, a primeira operação realizada consiste em colocar a máquina de estados, que efectua o controlo, num estado conhecido (Estado Inicial). O nó CAN é colocado no modo local e a máquina de estados no modo normal, sendo as saídas para drivers externos colocadas no valor inicial. Nos pinos de entrada é escrito o valor lógico 1, por forma a permitir a leitura.

Leitura de sensores

A leitura de sensores desencadeia acções sempre que detecta uma mudança no estado de um sensor. No entanto, foi introduzido um mecanismo de histerese na leitura de transições consecutivas, para prevenir a ocorrência de “bouncing” nos contactos do sensor. Existe assim um intervalo de 300 ms entre leituras de transição de estado, implementado com um mecanismo de contagem de tempo não bloqueante, baseado no hardware do microcontrolador.

Após a detecção de um evento, é verificado se o nó CAN se encontra em modo local ou remoto. O modo local significa que os eventos nos sensores serão avaliados localmente e reflectidos nas saídas. O modo remoto permite reconfigurar o algoritmo de controlo de uma carta, acelerar a avaliação sobre uma estação remota ligada ao barramento de comunicações CAN, que depois actua sobre as saídas locais e sobre as de outras cartas, através do barramento.

Algoritmo de controlo

Este algoritmo é implementado por uma máquina de estados, que realiza a gestão de variáveis de estado para um dado dispositivo e dos valores indicados por sensores e temporizadores. No caso de um nó CAN para iluminação, por exemplo, existe uma variável de estado chamada “LightState”, que indica o estado desejado para a iluminação. A operação da máquina de estados verifica se a lâmpada se encontra no mesmo estado que o indicado pela variável e, em caso negativo, desencadeia uma acção nesse sentido.

A máquina de estados tem um modo de funcionamento especial para a depuração, que divulga para o barramento todas as mudanças de estado numa saída. Este modo pode ser ligado/desligado através de mensagens de gestão vindas do barramento.

Actualização de saídas

A actualização das saídas do nó CAN serve para sincronizar o estado lógico das variáveis de estado da máquina de controlo, com o estado físico do dispositivo a controlar. A título de exemplo, podemos referir que uma variável de estado pode conter o valor 1 para representar o

estado ligado, mas o nível lógico a aplicar na porta de saída, para obter o estado desejado no actuador, pode ser 0.

Leitura de mensagens CAN

Até este momento, foi descrita a capacidade de controlo de um nó CAN, através de sensores locais. Estes nós podem no entanto ser totalmente controlados, monitorizados e reconfigurados remotamente, através de mensagens CAN. Este processo remoto começa por verificar se existem mensagens nos buffers circulares, e em caso afirmativo procede à leitura de uma mensagem. As mensagens são divulgadas a todas as estações CAN, que depois fazem a sua deslocação para determinar se a mensagem tem significado local. No caso negativo é automaticamente descartada e o algoritmo de controlo prossegue normalmente. A mensagem tem o seguinte formato:

TYPE	FROM	TO	COM	VAL
------	------	----	-----	-----

O significado dos campos apresentados é o seguinte:

TYPE – representa o tipo de formato

FROM – identificador da estação que gerou a mensagem

TO – identificador da estação a quem se destina a mensagem
(255 se for uma mensagem de difusão –*broadcast*)

COM – comando local a executar

VAL – parâmetro associado ao comando

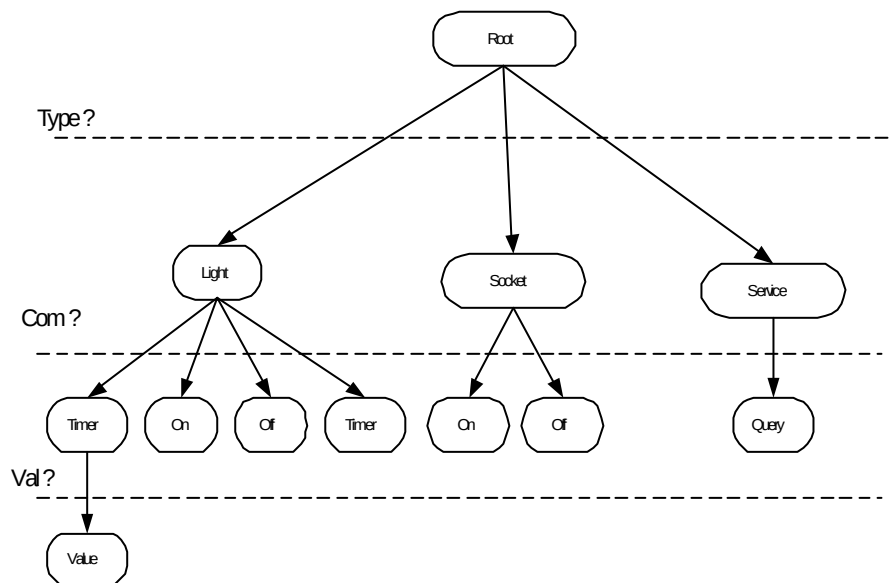


Figura 4.41: Árvore de decodificação de mensagens

A decodificação é realizada percorrendo uma árvore, representada na figura 4.41, à procura da informação contida nas folhas. Quando se encontra uma mensagem com significado, um comando local é executado. Caso contrário, a mensagem é descartada por falta de significado.

CAN sobre RS-232

Para transportar mensagens CAN sobre RS-232, foi desenvolvido um pacote de *software* estruturalmente semelhante ao descrito para as mensagens CAN, mas que recorre agora a um encapsulamento por trama de bytes, no sentido de garantir a sua integridade perante fontes de interferência. As funções de alto nível desenvolvidas são as seguintes:

A função **can_2_rs (void)** lê uma mensagem CAN nos buffers de recepção, encapsula a trama e envia-a sobre RS-232. A figura 4.42 ilustra o fluxograma desta função.

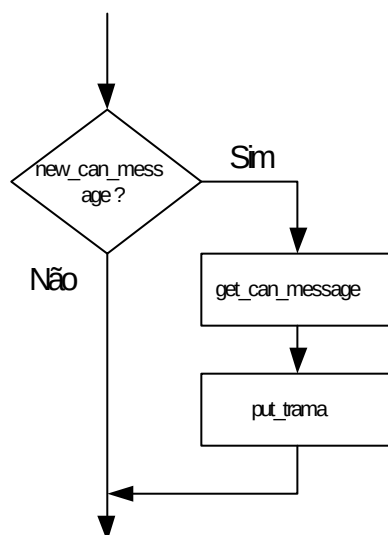


Figura 4.42: Fluxograma da função **can_2_rs**

A função **rs_2_can (void)** recebe uma trama via RS-232, verifica a sua integridade e transmite-a sobre o barramento CAN.

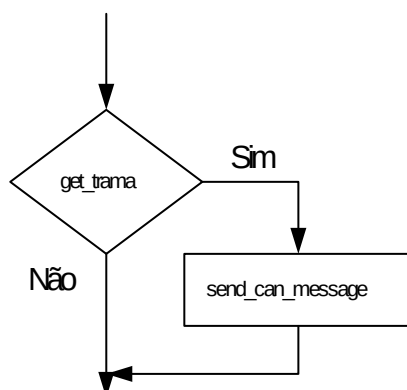


Figura 4.43: Fluxograma da função **rs_2_can**

A mais baixo nível, encontramos as seguintes funções:

A função **put_trama ()** envia uma trama sobre RS-232, encapsulando-a com um *header* = 05, H seguido da mensagem CAN e de um *trailer* igual ao CRC ($CRC = \sum \text{Trama} \% 254$). A figura 4.44 ilustra o fluxograma desta função.

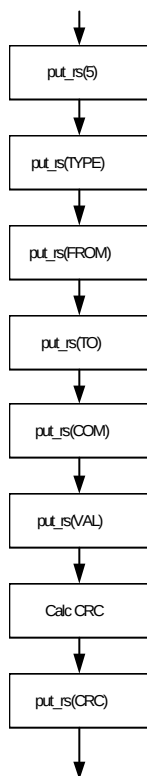


Figura 4.44: Fluxograma da função **put_trama**

A função **get_trama ()** corresponde ao fluxograma ilustrado na figura 4.45, no qual a primeira operação consiste em receber um *byte* via RS-232 e verificar se se trata do indicador de *header* da trama, seguido da mensagem CAN e do respectivo *trailer*, que deve conter um valor igual ao do CRC calculado.

A nível do *hardware* da UART, foram desenvolvidas quatro funções para facilitar a escrita do código para cada nó CAN. A simplicidade destas funções não justifica a apresentação dos respectivos fluxogramas, pelo que apresentaremos aqui apenas uma descrição sucinta da respectiva funcionalidade.

setup (): programa a UART para 9600 *baud*, 8 *bit*, sem paridade e com um *stop bit*.

new_rs (): indica a recepção de um novo carácter via RS-232.

put_rs (char valor): envia um carácter via RS-232.

get_rs (): retorna um carácter via RS-232.

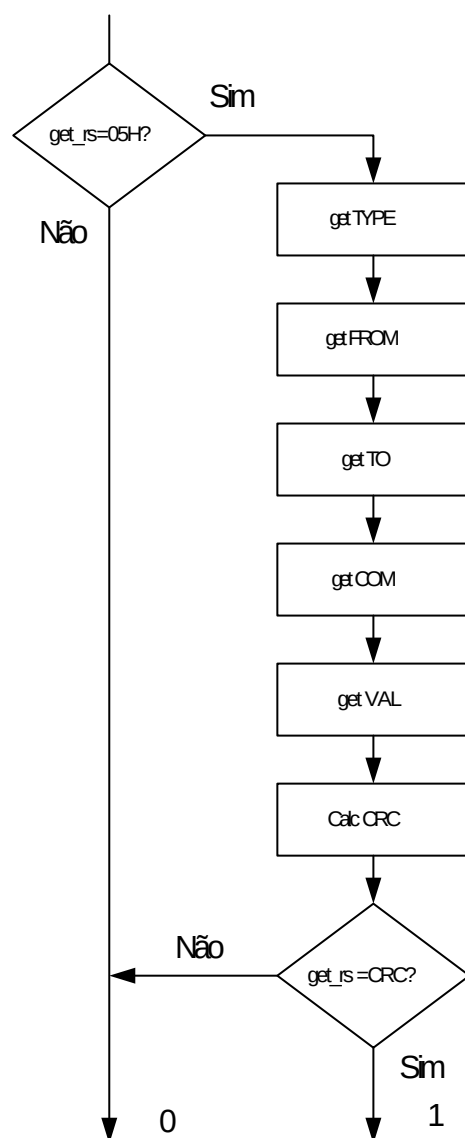


Figura 4.45: Fluxograma da função **get_trama**

Concluída a apresentação das características principais da arquitectura física do sistema desenvolvido, passaremos no próximo capítulo para a segunda das duas componentes principais que correspondem ao trabalho realizado pelo autor: a aplicação de interface com o utilizador.

Capítulo 5

Interface com o utilizador

Este capítulo apresenta a disposição dos elementos da interface com o utilizador e dos mecanismos de varrimento suportados, passando daí para a descrição da estrutura *software* que implementa as funcionalidades descritas no capítulo 3. A metodologia necessária à criação da interface adaptada a cada utilizador é então apresentada, a que se segue a consideração dos aspectos específicos de cada uma das vertentes suportadas (controlo ambiental, comunicação oral e telefónica, cenários, lançamento e utilização de aplicações Windows). Finalmente, é apresentado um exemplo concreto, usado para demonstrar as potencialidades da aplicação.

5.1 Elementos da Interface

A interface é constituída por sete elementos principais, ilustrados na figura 5.1:

- Menu Windows para personalizar a aplicação
- Fundo
- Botões de navegação
- Botão seleccionado (com 5% de ampliação face aos restantes)
- Sombra de selecção
- Imagem dos botões
- Texto indicador da acção do botão

A disposição ilustrada na figura 5.1 permite um máximo de 24 botões, dos quais dois são utilizados para navegação entre quadros: o primeiro botão, com nome “Raiz” (*Root*), permite saltar para o primeiro quadro da hierarquia a partir de qualquer outro, enquanto que o segundo, com o nome “Anterior”, salta apenas para o quadro “Pai” (*Parent*). Os quadros de selecção podem ser organizados numa árvore, como se ilustra na figura 5.2.

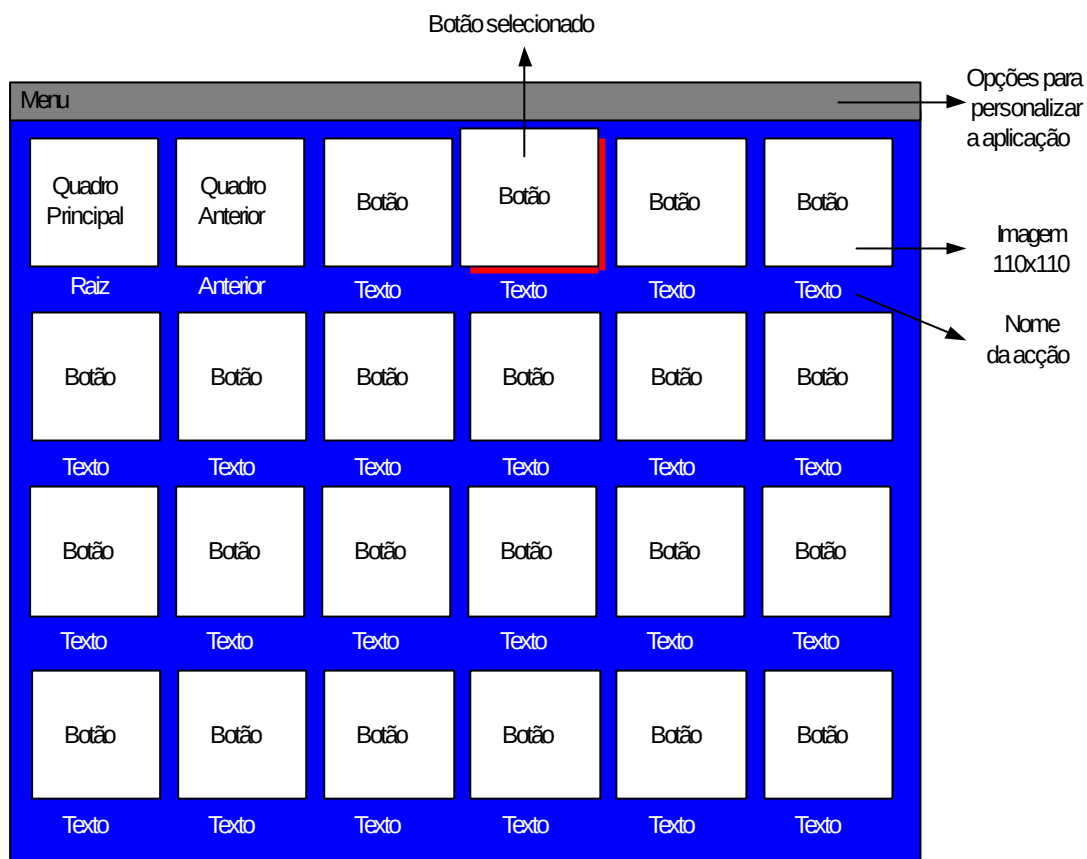


Figura 5.1: Elementos da interface

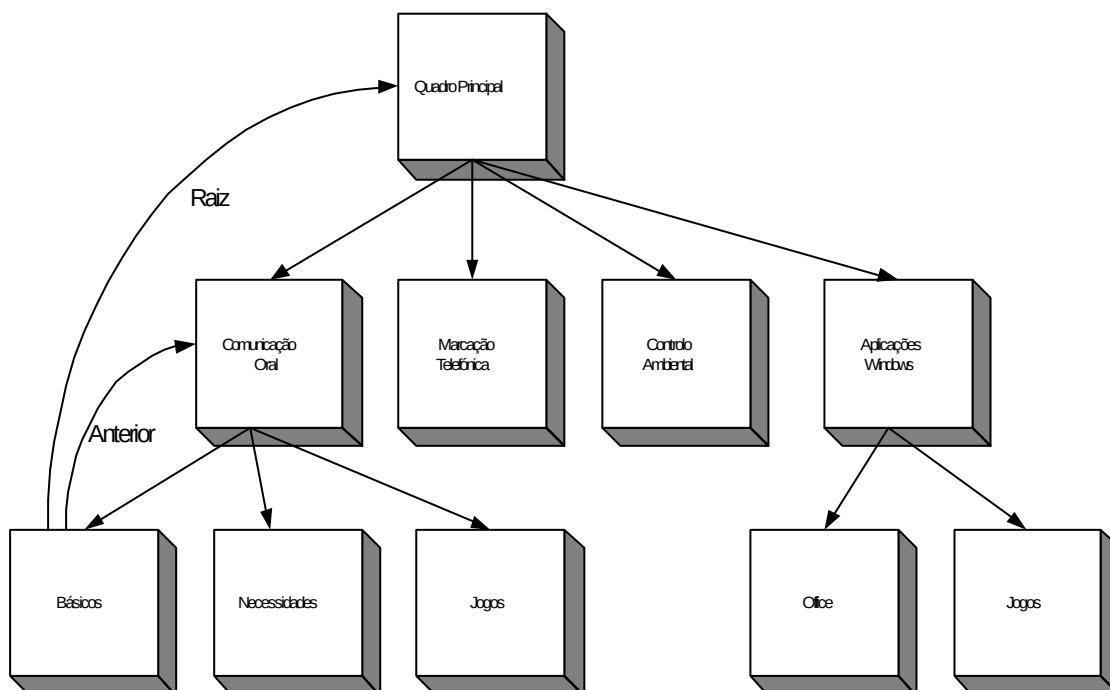


Figura 5.2: Organização hierárquica dos quadros da interface com o utilizador

A selecção dos botões é realizada através de um mecanismo de varrimento sequencial, em função de uma base de tempo escolhida para cada utilizador. Esta selecção é representada graficamente por um aumento do botão em 5% e pelo acrescento de uma sombra colorida nas linhas do direito e inferior, sugerindo a elevação do botão. Os mecanismos de varrimento implementados são os seguintes:

- O varrimento linear, em que cada botão vai sendo seleccionado da esquerda para a direita, linha a linha, como se ilustra na figura 5.3.
- O varrimento linha-coluna, mais eficiente, onde se começa por seleccionar a coluna e depois a linha, correspondendo a respectiva intercepção ao botão pretendido, como se ilustra na figura 5.4.

Com o segundo destes métodos consegue-se efectuar a selecção em menos de 10 intervalos de tempo, enquanto que o primeiro apresenta um máximo de 24 intervalos de selecção, quando o quadro se encontra cheio.

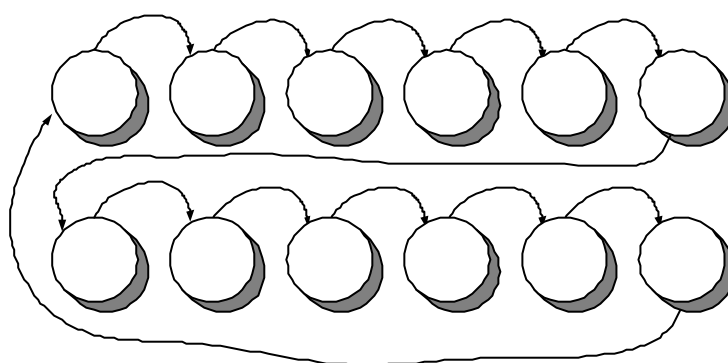


Figura 5.3: Varrimento linear

O esquema de varrimento é acompanhado de vários tipos de *feedback* ao utilizador, nomeadamente sonoro e visual, tais como frases identificando a acção, a correspondente descrição textual, uma sombra colorida em contraste com o fundo e aumento das dimensões do botão. Também o tempo entre selecções não contíguas é duplicado, por forma a permitir ao utilizador voltar a focar o objecto seleccionado, após um salto para o início. O mesmo acontece após a selecção da acção de um botão, já que o varrimento salta para o primeiro botão e permanece durante dois intervalos de selecção.

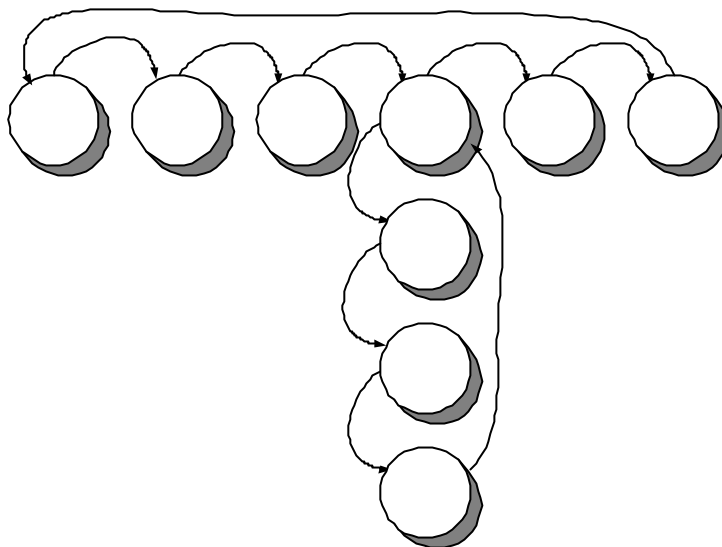


Figura 5.4: Varrimento linha-coluna

Após esta breve descrição funcional, passaremos à consideração da estrutura de software que serve de base à aplicação desenvolvida.

5.2 Descrição da implementação

A interface com o ambiente Windows é conseguida através de três objectos principais, nomeadamente o *TscanApplication*, *TscanWindow* e o *TPointerWindow* que actuando em conjunto tratam e respondem às mensagens vindas do sistema operativo.

A estrutura de dados formada pela aplicação pode ser descrita pelo esquema ilustrado na figura 5.5, em que as ligações representadas estão implementadas sob a forma de apontadores. Existem listas de objectos derivados de duas classes principais: *Botão* e *MiniBotão*, que implementam as acções de navegação e interface gráfica.

5.2.1 Classe TScanApplication

Esta classe, derivada da classe *TApplication* do *ObjectWindowsLibrary* é responsável por inicializar a aplicação no ambiente Windows. Como dados, esta classe acrescenta à sua base uma *string* de 255 caracteres, com o nome *StartName*, que se destina a guardar o nome do ficheiro *script* que contém a informação sobre os menus a criar. Como funções, apresenta *InitMainWindow* e *InitInstance*:

- *InitMainWindow* é usada para criar a estrutura da janela que vai conter a aplicação e registar o seu valor na propriedade *MainWindow* da classe *TApplication*. É igualmente responsável pela acção de leitura e inicialização dos menus, colocando de seguida o dispositivo apontador no meio da área da janela da aplicação.

- *InitInstance* é usada para informar a aplicação que deve aceitar os ícones arrastados sobre a área da respectiva janela, como uma operação de leitura de um ficheiro de menu (a esta operação o sistema Windows dá o nome de *DragFiles*).

Após estas operações de inicialização, a classe *TScanApplication* fica à espera de receber mensagens vindas do Windows, às quais dá por omissão o tratamento definido pela classe *TApplication* da *ObjectWindowsLibrary*. Finalmente, esta classe é responsável pela comutação entre janelas visíveis, *TScanWindow* e *TpointerWindow*, que implementam respectivamente as funções de selecção por quadros e as funções de emulação de rato e teclado.

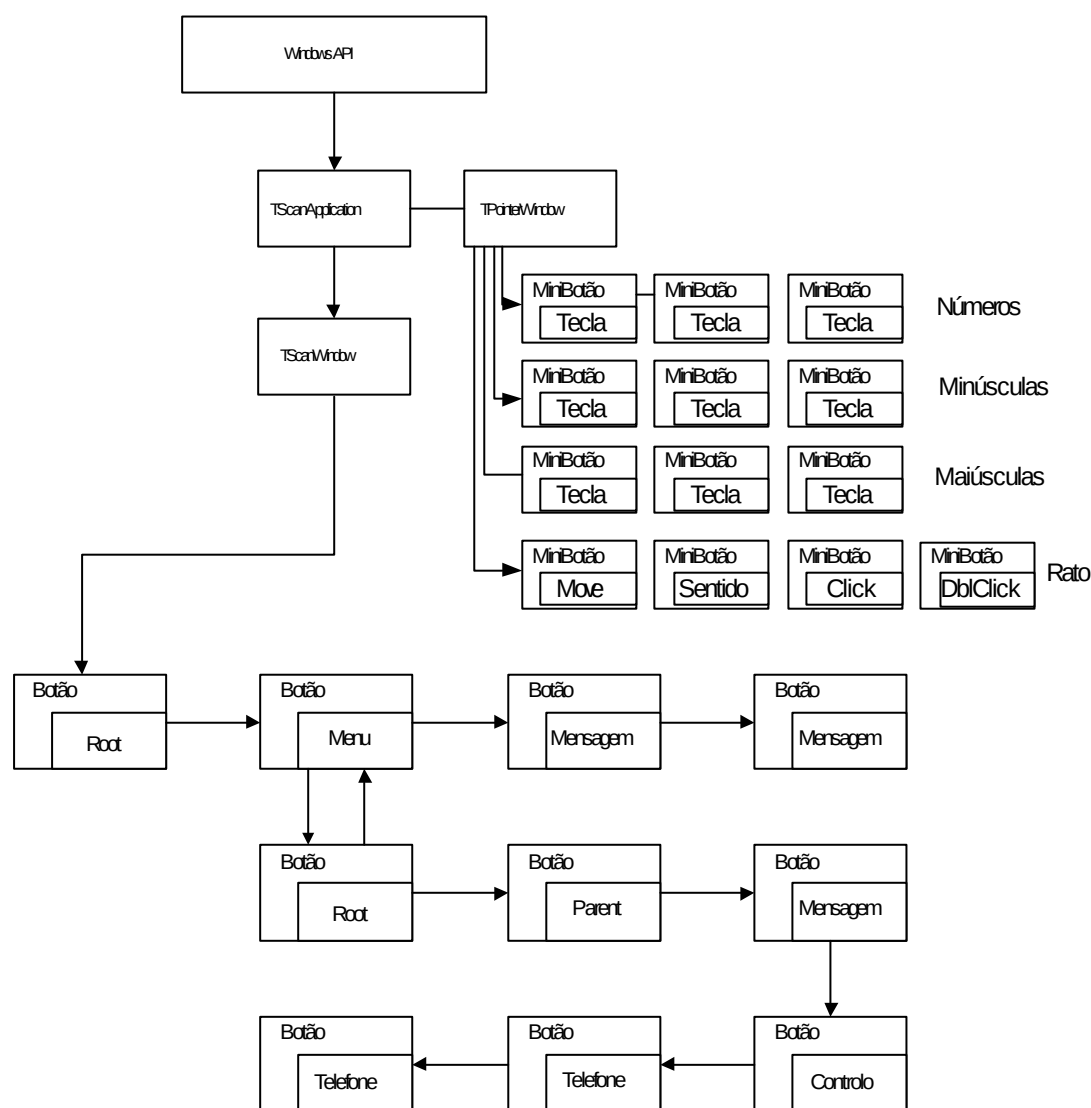


Figura 5.5: Representação da estrutura de dados

5.2.2 Classe TScanWindow

Esta classe é derivada da classe *TFrameWindow* da *ObjectWindowsLibrary* e é responsável por criar a *frame*, os menus, a área cliente da aplicação e por responder a todas as mensagens provenientes do *Windows* para a janela. Para além das funções de tratamento herdadas da classe base, tem funções para o tratamento das mensagens provenientes dos menus, *timer*, do dispositivo apontador dos *DragFile* e do próprio *Windows*.

Os dados guardados por esta classe permitem alterar as características da aplicação e adaptá-las da melhor forma ao utilizador em causa, o que cria a necessidade de se definirem as seguintes variáveis:

Tempo	- Tempo entre duas operações de realce (de botões) consecutivas
Scanmode	- Varrimento linha-a-linha ou linha-coluna.
SoundSW	- Liga ou desliga o som de realce de um objecto.
Altera	- Indica que o conjunto dos menus que formam a aplicação foi alterado.
FileTitle	- Nome do ficheiro <i>script</i> que descreve os menus.
Botão *Root	- Aponta para a raiz da hierarquia de menus.
Botão *Objectos	- Aponta para o menu actualmente activo

Em resposta às mensagens dos menus, a aplicação possui um conjunto de funções que se destinam a responder às acções pedidas pelo utilizador, que passamos a enumerar:

- **CmStep** - Activa a operação de varrimento dos menus da aplicação, sendo o tipo de varrimento seleccionável entre linha-a-linha ou linha-coluna. Para isso, esta função activa o temporizador com o valor de tempo seleccionado e altera o menu da janela para um menu contendo apenas a função STOP.
- **CmSmart** - Comuta entre os modos linha-a-linha e linha-coluna. O modo linha-a-linha é também designado por varrimento sequencial, já que os objectos vão sendo realçados um a um.
- **CheckSound** – Repõe o estado dos menus da janela, após a operação de STOP do varrimento.
- **CmTempo** - Altera o tempo entre duas operações de realce consecutivas, para o que é aberta uma caixa de diálogo onde o utilizador pode seleccionar o tempo que pretende, entre 0.2 s e 5 s.
- **CmFileNew** - Apaga os apontadores dos menus actuais e põe a aplicação no estado inicial, sem menus.
- **CmFileOpen** - Abre um ficheiro *script* e os vários objectos que formam os menus da aplicação.

- **CmFileSave** - Grava a aplicação corrente, usando o nome definido. Caso a aplicação nunca tenha sido gravada antes, invoca a função CmFileSaveAs.
- **CmFileSaveAs** - Pergunta ao utilizador o nome a dar à aplicação e envia ao objecto root a mensagem desave, que é propagada a toda a estrutura de dados.
- **CmInsertData** - Cria um novo menu do tipo de comando.
- **CmInsertDir** - Cria um sub-menu.
- **CmInsertSom** - Cria um objecto de comunicação assistida.
- **CmInsertTelf** - Cria um objecto de marcação de chamadas.
- **CmApagar** - Apaga o objecto realçado.
- **CmEditar** - Modifica as características do objecto realçado.

As restantes funções disponíveis no interior do objecto realizam procedimentos necessários à gestão da aplicação e em resposta a mensagens vindas do sistema operativo, mas não directamente relacionadas com os menus das janelas:

- **SetTitle** - Altera o título apresentado na frame da janela, de modo a mostrar o nome do ícone dos menus que estão carregados no momento.
- **EvTimer** - Responde à mensagem proveniente do timer, indicando a passagem do intervalo de tempo entre duas operações de realce consecutivas. Quando esta mensagem é recebida, o menu activo é notificado de que deve proceder à operação de realce do próximo objecto.
- **EvRButtonDown** - Indica que o objecto realçado foi seleccionado e que se deve executar a acção por ele especificada. Para isso, o menu activo é informado de que deve proceder à operação de selecção do objecto activo, que neste caso será o objecto realçado.
- **EVLButtonDblClick** - Quando o utilizador faz um duplo clique sobre o objecto da aplicação, a janela recebe uma mensagem contendo as coordenadas (X,Y) onde esta ocorrência teve lugar. A operação executada por esta função consiste em enviar uma mensagem ao menu activo, indicando que deve realçar o objecto, cuja área se contém em (X,Y).
- **SetList** - Especifica um novo menu activo, no caso dos saltos para um sub-menu, para o menu anterior ou para o menu principal.
- **SetRoot** - Especifica o objecto que se encontra na raiz da hierarquia de menus.
- **FileOpen** - Interpreta o ficheiro e cria os objectos descritos pelo menu de uma forma recursiva.
- **CanClose** - Verifica se o trabalho está gravado e em caso negativo pergunta ao utilizador se o deseja fazer, tomando as providências necessárias nesse sentido.
- **Paint** - Ordena ao Windows que desenhe o interior da janela.

5.2.3 Classe “TPointerWindow”

A gestão dos mini-botões para emulação do teclado e do rato é realizada por esta classe, que mantém quatro listas para efectuar as funções de emulação de rato, emulação de teclas mín

culas, de teclas maiúsculas, e de teclas com números e símbolos. Do ponto de vista da gestão de mensagens do Windows, é mais simples do que a classe anterior, uma vez que não existem menus de seleção. Existe apenas a necessidade de se efectuar o atendimento de mensagens provenientes do *Timer* e das mensagens do tipo `CM_EXIT`, indicadoras de que a aplicação terminou.

5.2.4 Classe "Botão"

Como base para a estrutura de dados da aplicação, foi criado um objecto com a totalidade da informação e procedimentos comuns a todos os objectos, formando assim uma estrutura de dados com os objectos de base "Botão". Este objecto é responsável pelos mecanismos de gestão que formam a estrutura de dados usada, multinível, à qual chamaremos hierarquia de menus, onde cada menu corresponde a uma lista ligada de objectos. Esta lista é auto-gerida, pelo que a realização de operações sobre ela requer apenas o envio de uma mensagem à cabeça da lista, que se encarregará de a propagar até aos objectos a que se destina. Com esta técnica conseguimos o encapsulamento das listas, com um acesso controlado aos seus objectos, o que representa maior segurança da informação e a consequente contenção de erros no programa.

Os dados contidos no objecto Botão permitem especificar características tal como o nome, a imagem, o som associado, o estado e as posições espaciais na janela do objecto. Aos objectos derivados desta classe pode ser acrescentada mais informação que seja necessária para executar a tarefa em causa. Entre as funções disponibilizadas por este objecto destacamos as seguintes:

Insert	- Inserir um objecto no fim da lista.
Delete	- Apagar o n-ésimo objecto da lista.
Count	- Retornar o número de objectos da lista.
Selected	- Usado pelo varrimento linha a linha para realçar o próximo objecto.
SetXY	- Para realçar o objecto que contém as coordenadas (X,Y) da janela.
CleanStatus	- Destina-se a limpar as <i>flags</i> de realce de todos os objectos
Positions	- Serve para atribuir posições espaciais dentro da janela a cada objecto da lista.
Run	- Especifica a operação a executar quando o botão é seleccionado.
Any	- Retorna <i>verdadeira</i> se existe um objecto realçado.
PlaySound	- Som que especifica o objecto realçado.
SetType	- Número inteiro que especifica o tipo de objecto em causa.
GetType	- Retorna um inteiro que identifica o tipo do objecto.
Save	- Salva o objecto num ficheiro <i>script</i> .
Edit	- Serve para alterar as características de um objecto.
Paint	- Serve para dar forma gráfica ao objecto.

Entre as funções apresentadas pelo objecto de base Botão, encontramos três funções que foram declaradas como virtuais, ou seja, que devem ser declaradas nas classes derivadas, por forma a obter-se uma função específica de cada classe, mas considerando-se como se se tratasse de uma função da classe base (ou seja, as classes derivadas alteram o funcionamento destas funções, adaptando-o às suas necessidades). As funções virtuais deste objecto são as seguintes: *Run*, *Save* e *Edit*.

Serão apresentados dois conjuntos de classes, uma com funções associadas aos menus e a outra com funções associadas a acções, sendo ambas apresentadas de seguida.

Classes associadas aos menus

- **Classe "Menu Principal"**: Este objecto pertence ao grupo dos objectos que mudam o apontador do menu activo. Quando este objecto é seleccionado, o apontador do menu activo passa a ser o primeiro menu da hierarquia, permitindo assim voltar directamente ao início da hierarquia, a partir de qualquer ponto da estrutura.
- **Classe "Menu Anterior"**: Este objecto contém um apontador para a lista de objectos que formam o menu anterior. A selecção deste objecto faz com que a aplicação mude o apontador de menu, para o menu que se situa um nível hierárquico superior.
- **Classe "Sub-Menu"**: Este objecto é análogo ao objecto "Menu Anterior", contendo um apontador para um menu de objectos. Quando for seleccionado, a aplicação muda o apontador do menu activo para um sub-menu.

Classes associadas às acções

- **Classe "Mensagens"**: As funções de comunicação assistida são asseguradas por este objecto, que quando é seleccionado utiliza a placa de áudio para reproduzir o som contido no respectivo ficheiro de som. Este objecto, para além das propriedades herdadas do objecto base "Botão", acrescenta uma nova propriedade *FileScan[]*, que se destina a guardar o nome do ficheiro de som que lhe foi associado quando foi criado.
- **Classe "Telefone"**: Este objecto realiza a marcação automática de números de telefone, através de um *Modem*. Para guardar o número de telefone, este objecto acrescenta uma nova característica ao objecto base, correspondente a um vector de caracteres que representa o número a marcar.
- **Classe "Comando"**: O objectivo deste objecto é controlar uma estrutura domótica através de comandos enviados pela porta série. Estes comandos estão contidos num ficheiro binário e são definidos pela estrutura em causa, que é independente do tratamento que lhe é dado pela aplicação. Para guardar a informação referente ao nome do ficheiro de comandos, este objecto acrescenta uma nova característica ao objecto de base, designada por *FileDados* e representada por um vector de 255 caracteres.

5.2.5 Classe “Mini-Botão”

Como se ilustra no diagrama da estrutura de dados, representado na figura 5.5, foi criado um objecto designado por “mini-botão”, com a totalidade da informação e os procedimentos comuns a todos os objectos que implementam as funcionalidades de emulação. Este objecto é funcionalmente equivalente à classe botão, mas apresenta como característica inovadora o facto de os objectos dele derivados serem inseridos “*on-the-fly*” numa janela de apresentação, que não recebe mensagens de eventos Windows, gerando-os antes para este sistema operativo, através de funções de síntese de eventos. Entre as funções disponibilizadas por este objecto destacamos as seguintes:

Insert	-	Inserir um objecto no fim da lista.
Count	-	Retorna o número de objectos da lista.
Selected	-	Usado pelo Varrimento linha a linha para realçar o próximo objecto.
Run	-	Especifica a operação a executar quando o botão é seleccionado.
Paint	-	Serve para dar forma gráfica ao objecto.
TimeTick	-	Propaga um evento do <i>Timer</i> pela lista de objectos

Entre as funções apresentadas pelo objecto de base Botão, encontramos três funções que foram declaradas como virtuais, ou seja, que devem ser declaradas nas classes derivadas, por forma a obter-se uma função específica de cada classe, mas considerando-se como se se tratasse de uma função da classe base (ou seja, as classes derivadas alteram o funcionamento destas funções, adaptando-o às suas necessidades). As funções virtuais deste objecto são as seguintes: *Run*, *Paint* e *TimeTick*.

5.3 Descrição da aplicação "Fernão Magalhães"

A designação dada à aplicação desenvolvida para efectuar o interface com o utilizador ~~for-~~
~~não de Magalhães~~ tendo sido testada com sucesso nos sistemas operativos ~~Windows~~ 3.11 com
extensão 32 bit e Windows 95/98. O *hardware* necessário consiste num computador com in
croprocessador 80386 ou superior, placa SVGA com resolução 800x600 e mais de 65535
cores, placa de som e um ~~Modem~~. Para iniciar a aplicação existem vários métodos:

- Duplo *click* no ícone da aplicação.
- Duplo *click* num documento do tipo menu
- Arrastar um documento do tipo menu sobre o ícone da aplicação
- Depositar um documento do tipo menu no directório ~~startup~~ (isto permite que a aplicação se inicie automaticamente ao iniciar o Windows).

5.3.1 Adaptação a cada utilizador

Para criar interfaces dedicadas, o programa disponibiliza um menu de janela com várias funções, que permitem adicionar ou retirar objectos aos menus, acrescentando novos menus e alterando as características de varrimento. No seu estado inicial, a interface apresenta-se como se ilustra na figura 5.6, onde se constata a presença de um objecto com o nome "Root". Este objecto é adicionado automaticamente a todos os menus criados e tem por função disponibilizar um atalho para a raiz, a partir de qualquer ponto da hierarquia de menus.

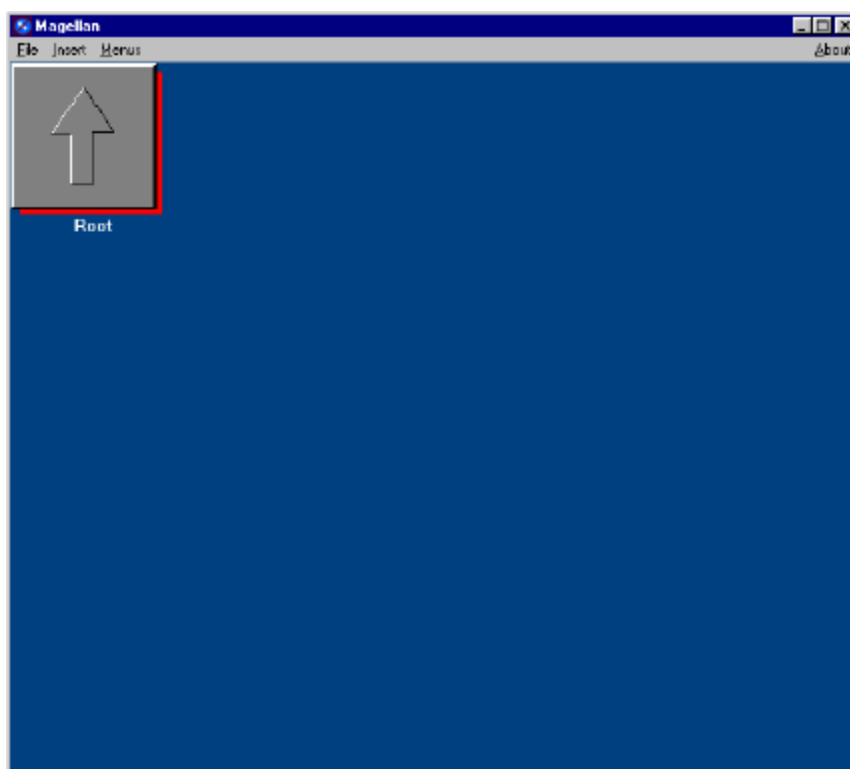


Figura 5.6: Estado inicial da interface, antes de criados os menus

Quando se cria um novo sub-menu, para além do objecto "Root", é também adicionado automaticamente um objecto do tipo "Parent", representado na figura 5.7, que permite retornar ao menu imediatamente acima deste sub-menu.

Nesta interface, para realçar um objecto, basta dar um duplo toque com o botão esquerdo do rato sobre a respectiva área. Para efectuar a selecção do objecto basta um toque sobre o botão direito do rato. Para criar a interface propriamente dita dispomos de três conjuntos de opções, agrupadas sob as designações de *File*, *Insert* e *Menus*, também presentes na figura 5.7.

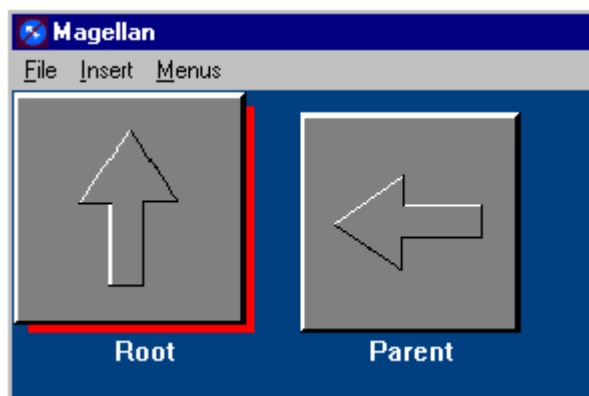


Figura 5.7: Botões iniciais de um sub-menu

O menu "File"

O menu "*File*" faz a gestão de ficheiros da aplicação e integra as cinco opções representadas na figura 5.8.



Figura 5.8: O menu "*File*"

A funcionalidades destas opções pode ser descrita da seguinte forma:

- **New** - Permite limpar o conteúdo da memória, apagando a interface actual e colocando a interface no estado inicial. Se a interface carregada tiver sido alterada durante a sessão actual, esta opção pergunta ao utilizador se deseja guardar as alterações, antes de proceder à reposição do estado inicial. A correspondente janela de diálogo está representada na figura 5.9.
- **Open** - Permite carregar um ficheiro do tipo menu já existente, apagando a interface actual e carregando o estado guardado no ficheiro. Se a interface carregada tiver sido alterada durante a sessão actual, esta opção pergunta ao utilizador se deseja guardar as alterações antes de proceder à reposição do estado inicial, após o que abre o *File Requester* representado na figura 5.10.

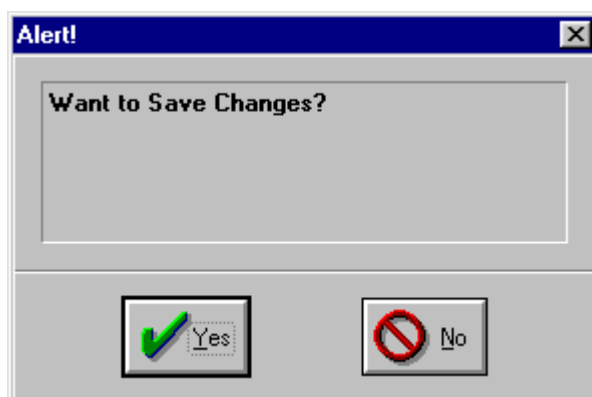


Figura 5.9: Mensagem de alerta quando se quer abandonar uma interface alterada

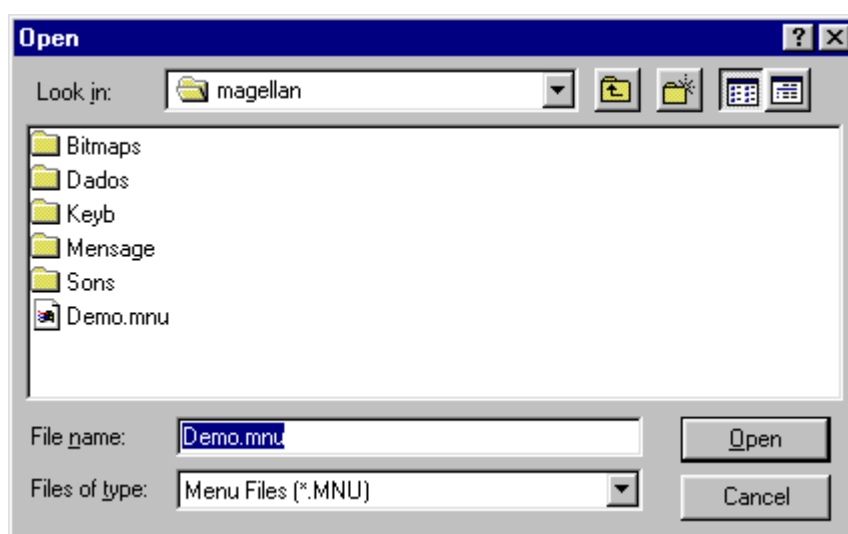


Figura 5.10: *File Requester* correspondente à opção *Open*

- **Save** - Permite guardar a interface actual, desde que não seja a primeira vez que esta opção é executada. Se a interface actual nunca foi gravada, chama a função seguinte.
- **Save As** - Guarda a interface actual com o nome indicado pelo utilizador. Pode ser seleccionada pelo utilizador ou pode ser chamada pela função anterior. Esta função é utilizada para gravar interfaces que não foram gravadas antes, ou para mudar o nome da interface ou a sua localização. Quando esta função é seleccionada, aparece um *File Requester*, representado na figura 5.11, onde o utilizador pode especificar o nome e a localização pretendidos para a interface.
- **Sair** - Abandona o programa, apagando a interface actual e limpando o conteúdo da memória. Se a interface carregada tiver sido alterada durante a sessão actual, esta opção pergunta ao utilizador se deseja guardar as alterações antes de abandonar a aplicação.

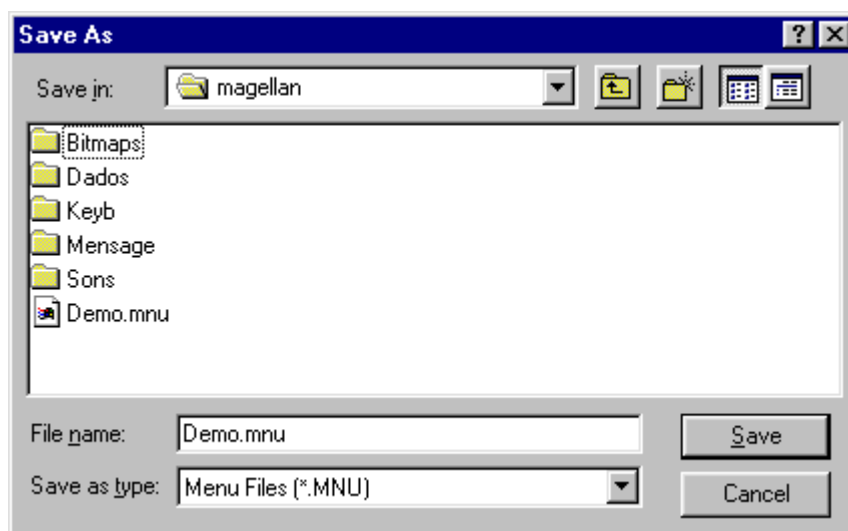


Figura 5.11: *File Requester* correspondente à opção *Save As*

O menu "Insert"

O menu "*Insert*" disponibiliza as opções representadas na figura 5.12.

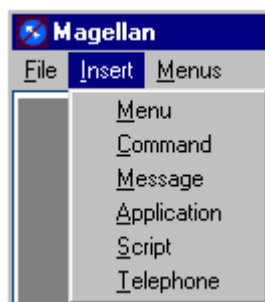
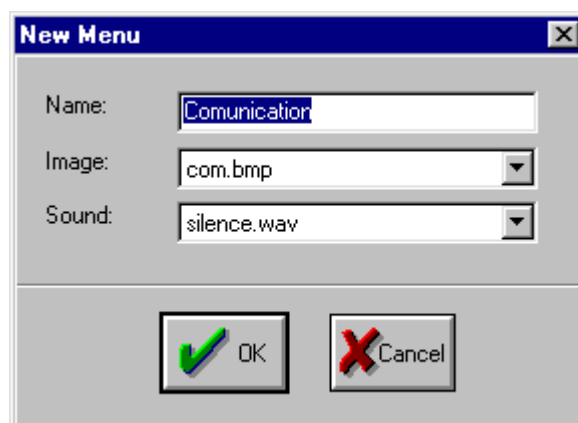
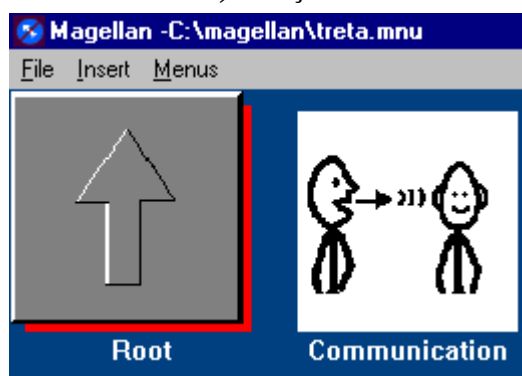


Figura 5.12: Opções do menu "*Insert*"

- **Menu** - Permite incluir na interface um objecto que possibilita a passagem para outro menu (um sub-menu). Este objecto tem um nome, uma imagem e um som associados, como se ilustra na figura 5.13.a, que apresenta a caixa de diálogo usada para a sua construção.
- **Command** - Permite incluir na interface um objecto que comanda um dispositivo *ns-e* estrutura domótica. Este tipo de objecto tem um nome, uma imagem, um som e um ficheiro de dados associado, como se ilustra na figura 5.14.a, que apresenta a caixa de diálogo usada para a sua construção. Este ficheiro é composto pelo código de uma acção a executar, que será enviado para a estrutura domótica através da porta série, quando este objecto for selecionado.



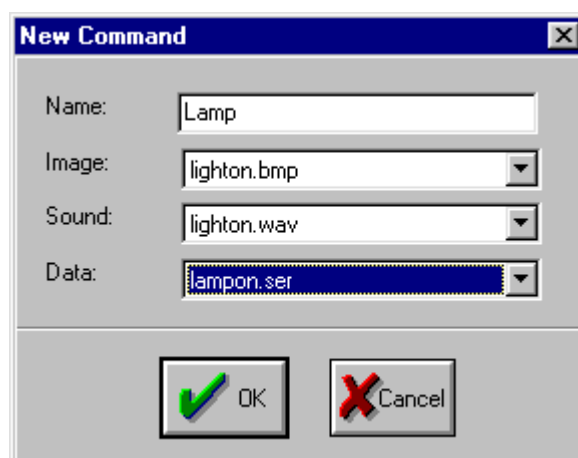
a) Criação



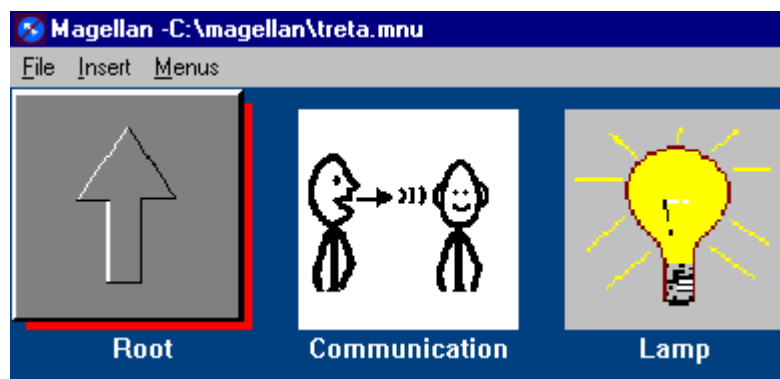
b) O objecto depois de criado

Figura 5.13: Criação do objecto Menu

- **Message** – Esta opção diz respeito aos objectos de comunicação assistida. A figura 5.15.a ilustra a caixa de diálogo para a construção de um objecto deste tipo, que inclui um nome, uma imagem, um som e uma mensagem associada, que será enviada para um dispositivo reproduzidor de som quando este objecto for seleccionado.



a) Criação

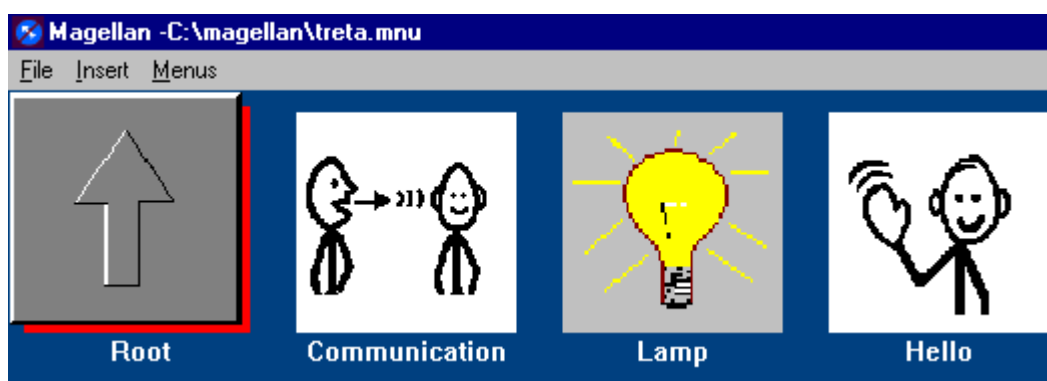


b) O objecto depois de criado

Figura 5.14: Criação do objecto Comando



a) Criação



b) O objecto depois de criado

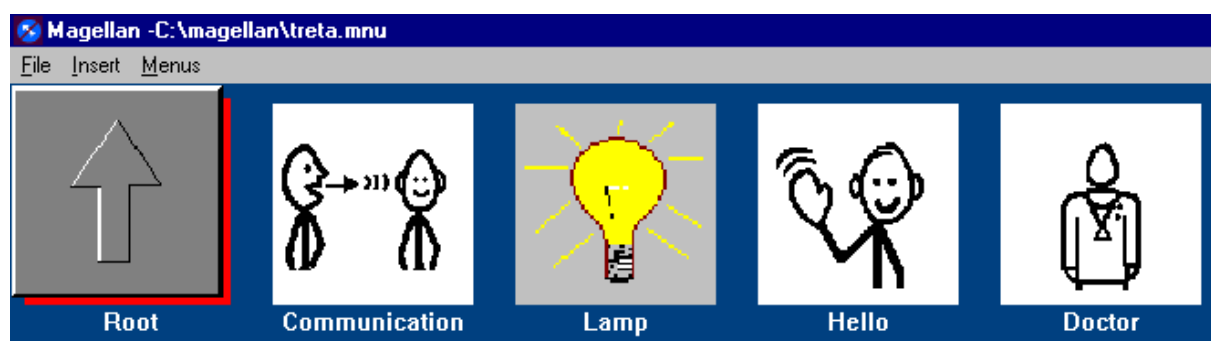
Figura 5.15: Criação do objecto Mensagem

- **Telephone** – Esta opção permite o estabelecimento de ligações telefónicas. A figura 5.16.a ilustra a caixa de diálogo para a construção de um objecto deste tipo, que inclui um nome, uma imagem, um som e um número de telefone associado. Quando esta opção é selecionada,

nada, a marcação faz-se através de um *modem* e a chamada é passada para um telefone do tipo mãos livres.



a) Criação

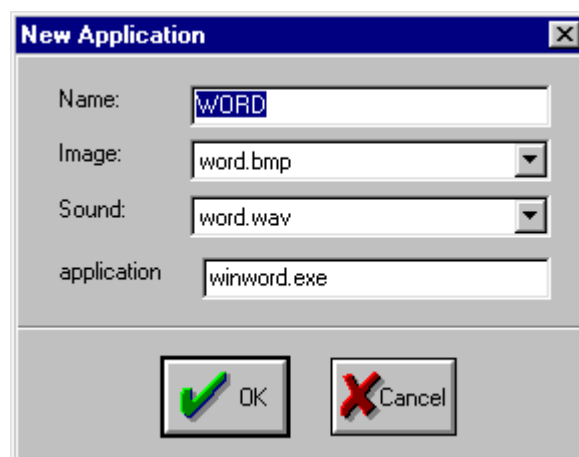


b) O objecto depois de criado

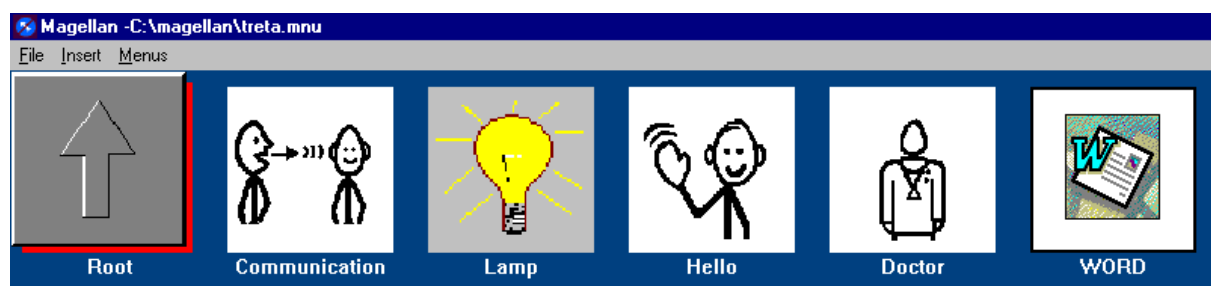
Figura 5.16: Criação do objecto Telefone

- **Application** – Esta aplicação permite o lançamento de aplicações Windows. A figura 5.17.a ilustra a caixa de diálogo para a construção de um objecto deste tipo, que inclui um nome, uma imagem, um som e o caminho para o ficheiro da aplicação associada. Quando esta opção é seleccionada, a aplicação referida é iniciada e tem lugar automaticamente a passagem para o modo de emulação de rato.

Quando qualquer uma das opções descritas é seleccionada, aparece uma caixa de diálogo onde se pedem os valores dos campos a ela associados. A aplicação só cria novos objectos cujos campos, na respectiva caixa de diálogo, estejam completamente preenchidos.



a) Criação

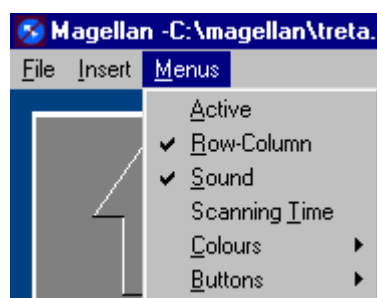


b) O objecto depois de criado

Figura 5.17: Criação do objecto Aplicação

O menu "Menus"

O menu "*Menus*" é composto por seis opções e serve para mudar o tipo de varrimento, mudar o tempo de varrimento, inibir o som de cada objecto, personalizar o esquema de cores usados, editar os campos dos objectos já criados, apagar objectos ou pôr a interface a funcionar.

Figura 5.18: Menu "*Menus*"

- **Active** – Esta opção coloca a aplicação em funcionamento, iniciando-se o varrimento dos objectos existentes na interface, de acordo com os parâmetros indicados pelas três opções

consideradas a seguir. Esta opção passa então **Stop**, que permite parar o varrimento dos objectos e voltar ao modo de edição da interface.

- **Row-column** – Esta opção permite seleccionar o tipo de varrimento que a aplicação irá efectuar quando em funcionamento: linha-a-linha ou linha-coluna. Quando se selecciona esta opção aparece ou desaparece uma marca à sua frente. Se a marca estiver presente está-se no varrimento linha-coluna e se estiver ausente no modo linha-a-linha.
- **Sound** - Esta opção permite habilitar/inibir o som do objecto, quando o varrimento o afecta. É do mesmo tipo da opção anterior. Quando a marca está presente o som está activo, estando inibido no caso contrário.
- **Scanning Time** - Esta opção permite modificar o tempo de varrimento (e.g. o tempo entre duas operações de realce consecutivas). Aparece então a caixa de diálogo ilustrada na figura 5.19, cujo cursor permite seleccionar a velocidade do movimento, entre os 0.2 s e os 5 s.

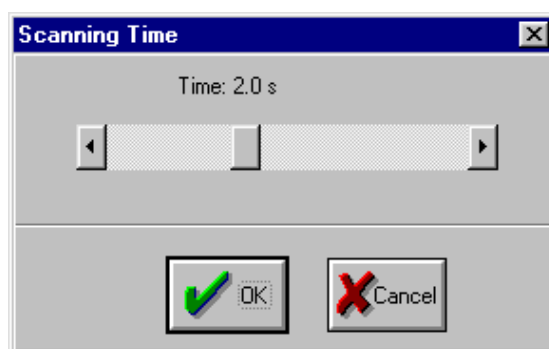


Figura 5.19: Caixa de diálogo para especificar o tempo de varrimento

- **Colours** – Esta opção permite seleccionar o esquema cromático da interface, no que diz respeito às cores usadas para o fundo, para o texto e para a sombra de selecção. A selecção de cores é conseguida pela caixa de diálogo ilustrada na figura 5.20, que permite escolher qualquer cor de um universo com 16 milhões de alternativas.

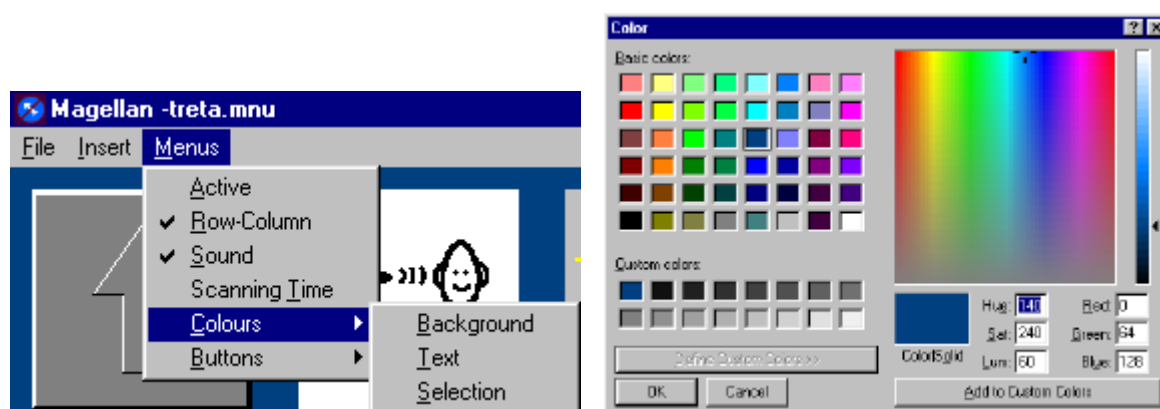


Figura 5.20: Menu para selecção de cores

- **Buttons** - Esta opção é do tipo *pop-up* e tem duas sub-opções, que servem para apagar ou editar os objectos existentes, como se ilustra na figura 5.21.

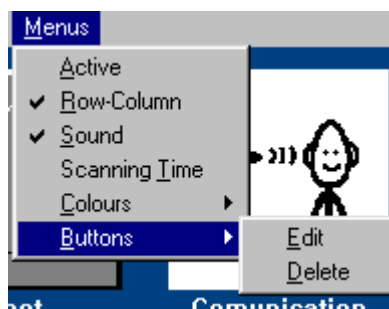


Figura 5.21: Menu de edição de Botões

- **Editar** - permite modificar qualquer um dos campos do objecto correntemente seleccionado. Esta opção abre uma caixa de diálogo que é diferente para cada tipo de objecto e onde o utilizador pode modificar os campos pretendidos.
- **Apagar** - permite apagar o objecto seleccionado, após uma mensagem de confirmação.

5.3.2 Controlo Ambiental

Para criar um objecto de controlo ambiental é necessário ir ao menu *Inserir* e seleccionar a opção *Comando*. Surge então uma caixa de diálogo com quatro campos para preencher:

- O primeiro campo diz respeito ao nome do objecto, que deve ser escolhido por forma a ilustrar a sua utilização. Apesar de poder ter 255 caracteres, o nome não deve ocupar mais do que a imagem do objecto.
- O segundo campo diz respeito à imagem do objecto e deve ser preenchido com o nome de um ficheiro *bitmap* do Windows (.bmp). Os *bitmaps* usados são imagens de 256 cores, 110x110, que podem ser feitas na aplicação *Paint* e devem ser gravadas no directório *Horizon/Bitmaps*.
- O terceiro campo tem que ser preenchido com um ficheiro de som digitalizado (.wav). Esta possibilidade permite aos utilizadores invisuais identificar quando o objecto é realçado, pelo que o som deve identificar a acção a executar. Este ficheiro é gravado com *sound recorder* ou com o *SuperSonic*.
- O quarto campo deve ser preenchido com a identificação de um ficheiro de dados (.ser), que contém a sequência de valores a enviar pela porta série, para controlar uma interface externa. Estes valores podem ir desde um carácter até uma sequência de milhares de caracteres, gerada por uma aplicação compatível com o protocolo de comunicações usado. Este ficheiro pode ser feito com o *Notepad* e tem que ser gravado no directório *Horizon/Dados*.

5.3.3 Comunicação Assistida

Um objecto de comunicação assistida serve para comunicar com outra pessoa através de um dispositivo reproduzidor de som e pode ser criado através da opção *Mensagem*, no menu *Inserir*. Aparece então uma caixa de diálogo com quatro campos a preencher, sendo os três primeiros preenchidos da mesma forma apresentada para o objecto anterior (e que, como foi já referido, devem permitir uma identificação intuitiva da função a que se destinam). O quarto campo deve ser preenchido com a identificação do ficheiro de som (.wav) que contém a mensagem a ser transmitida para o dispositivo reproduzidor de som. Este ficheiro é também gravado como *sound recorder*, mas deve ser gravado no directório *Horizon/Mensagens*.

5.3.4 Estabelecimento de ligações telefónicas

Este objecto destina-se ao estabelecimento de ligações telefónicas com o auxílio de um *modem* e de um telefone do tipo mãos livres. Para criar um objecto deste tipo deve seleccionar-se a opção *Telefone*, no menu *Inserir*. Os três primeiros campos são idênticos aos anteriores, uma vez que dizem respeito a características que são comuns a todos os menus. O quarto campo deve ser preenchido com o número de telefone que será marcado por um *modem*, quando este objecto for seleccionado, após o que a chamada deve ser passada para um telefone mãos livres.

5.3.5 Definição de cenários

A definição de cenários baseia-se numa linguagem de programação, que permite realizar tarefas complexas a partir de um único objecto.

Os comandos suportados são os seguintes:

Play(mensagem)	-	Comunicação Oral, reproduzindo uma mensagem.
Send(command)	-	Envia uma trama de comando para o sistema doméstico.
Phone(Numero)	-	Inicia a marcação de um número telefónico.
Run(Application)	-	Lança uma aplicação Windows.
Pointer()	-	Entra em modo de emulação de rato e teclado.
Wait(tempo)	-	Espera durante o tempo especificado.
Switch()	-	Espera até um novo evento <i>switch</i> de selecção.

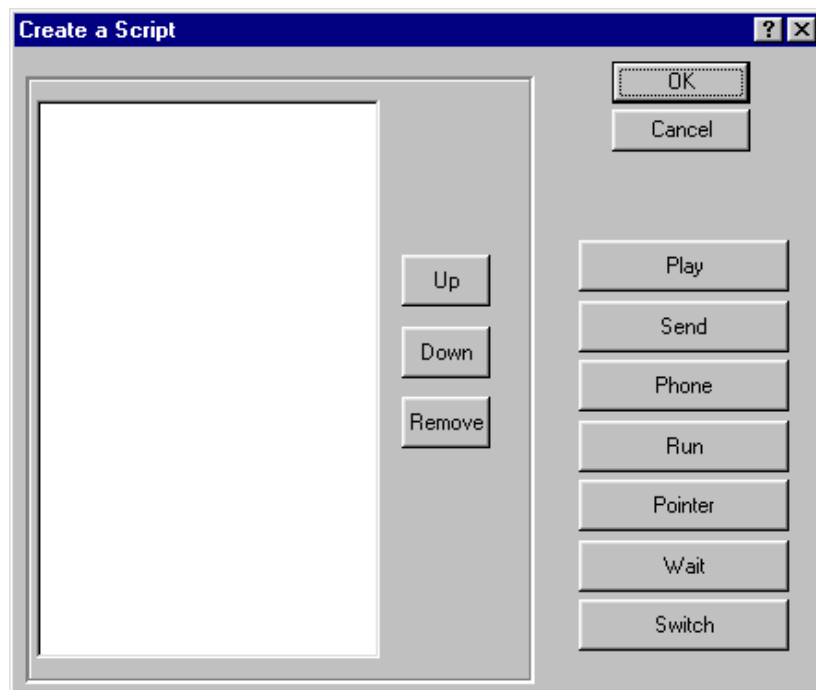


Figura 5.22: Editor de cenários

5.3.6 Lançamento de aplicações Windows

Quando uma aplicação Windows é lançada a partir da aplicação Fernão de Magalhães, esta comuta automaticamente para o modo de emulação de rato e teclado. Na figura 5.23 está representada a régua com os botões para a emulação de rato.

Os botões que integram esta régua têm o seguinte significado:



Voltar ao sistema de quadros de selecção.



Movimentar o cursor na direcção da seta (representada no botão seguinte).



Definir um sentido de movimentação para o cursor. Quando este botão é seleccionado, a sua seta gira no sentido dos ponteiros do relógio.



Simula o premir do botão esquerdo do rato.



Simula premir duas vezes o botão esquerdo do rato, para seleccionar ícones do Windows.



Entra no modo de emulação de teclado. Este modo altera a janela de emulação por forma a conter 39 botões de representação de caracteres maiúsculos, minúsculos, números e símbolos, como se ilustra na figura 5.24.

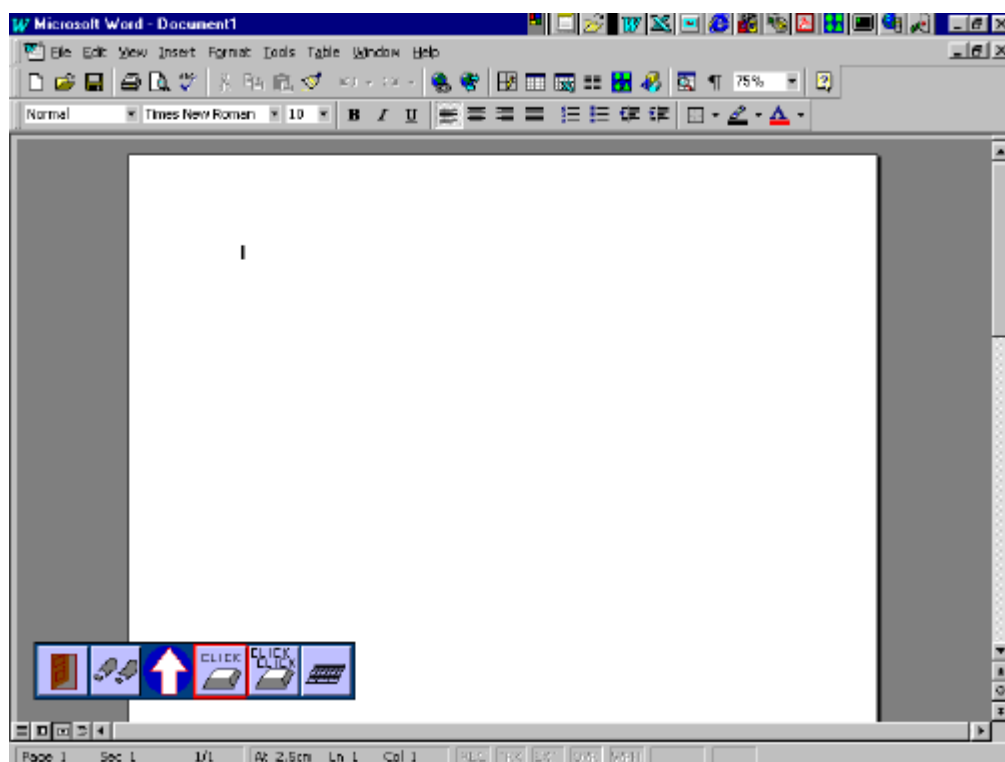


Figura 5.23: Emulação de rato

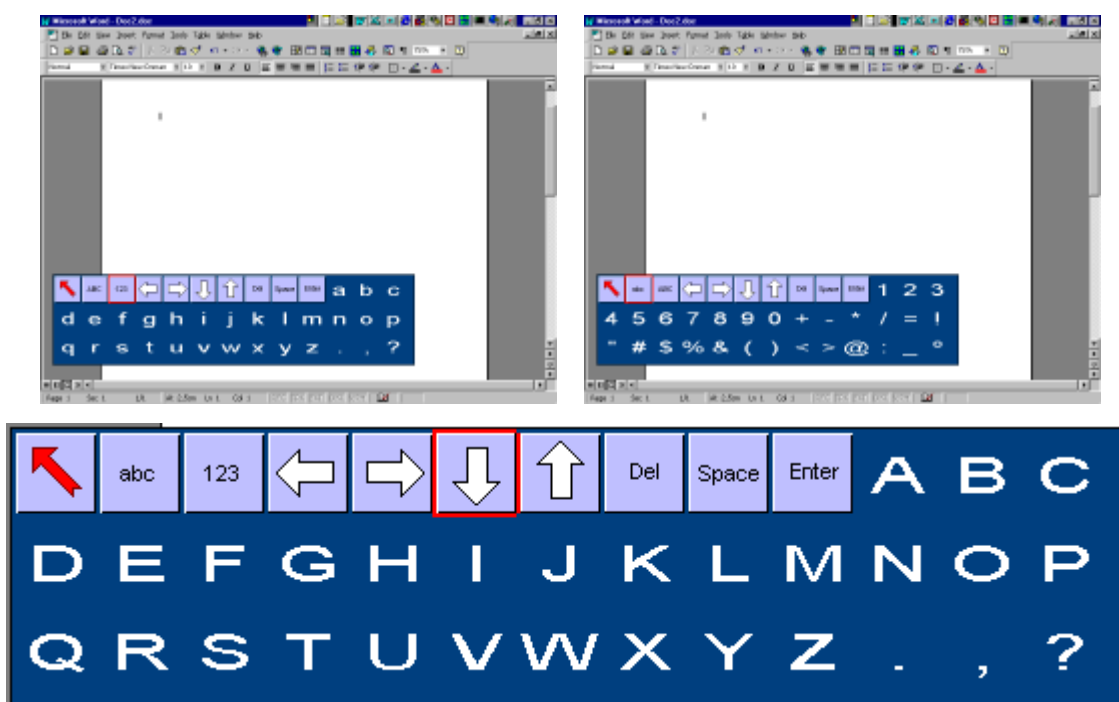


Figura 5.24: Emulação de teclado

5.4 Exemplo de aplicação

Como exemplo de aplicação, está incluído no CD-ROM em anexo um ficheiro de demonstração da aplicação de Fernão de Magalhães, com o nome “demo.mnu”. O aspecto inicial desta interface é o representado na figura 5.25.



Figura 5.25: Interface de demonstração

Esta interface está organizada em dois níveis:

- O nível da Raiz, que contém apenas botões do tipo Menu, que apontam para as várias áreas de intervenção presentes.
- O nível dos quadros de selecção, que implementam as funções de comunicação oral assistida, controlo ambiental, apoio ao estabelecimento de chamadas telefónicas e o lançamento de aplicações Windows.

O quadro de selecção para apoio à comunicação oral, representado na figura 5.26, é composto por desenhos e imagens Bliss representativas da mensagem associada a cada objecto, existindo ainda um campo textual com a mesma informação e um som *feedback* de selecção, que é emitido sempre que um novo objecto é realçado e se encontra pronto para ser seleccionado. Este quadro poderia conter imagens do tipo fotográfico e ser reorganizado em mais sub-menus, vocacionados para situações específicas do dia-a-dia (e.g. um quadro de alimentação, um quadro de sentimentos, um quadro de desejos, um quadro de jogos, etc.).

O quadro de controlo ambiental, representado na figura 5.27, ilustra um conjunto de dispositivos que estiveram presentes num painel de demonstração, permitindo controlar a iluminação, tomadas eléctricas, portas, janelas e persianas.

O quadro de apoio ao estabelecimento de chamadas telefónicas contém dois tipos de botões, um para o estabelecimento de chamadas e o outro para a comunicação oral, por forma a permitir interagir rapidamente com o interlocutor remoto e avisá-lo do tipo de diálogo que será estabelecido. Este quadro está representado na figura 5.28.

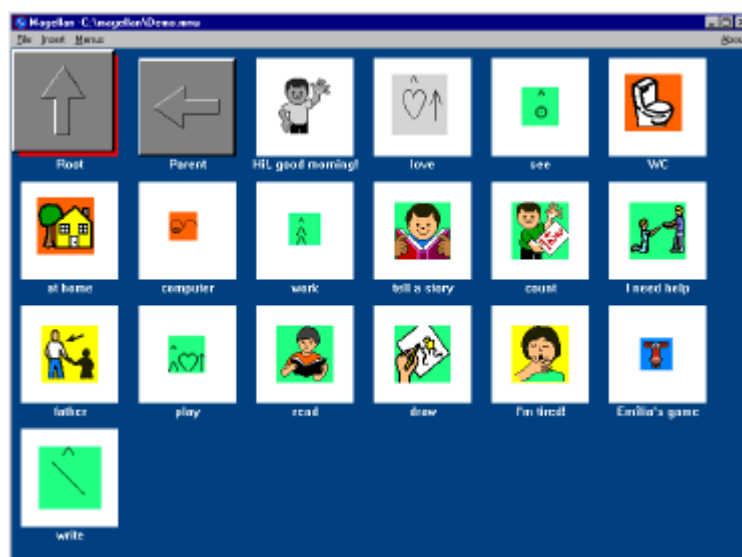


Figura 5.26: Quadro de apoio à comunicação oral

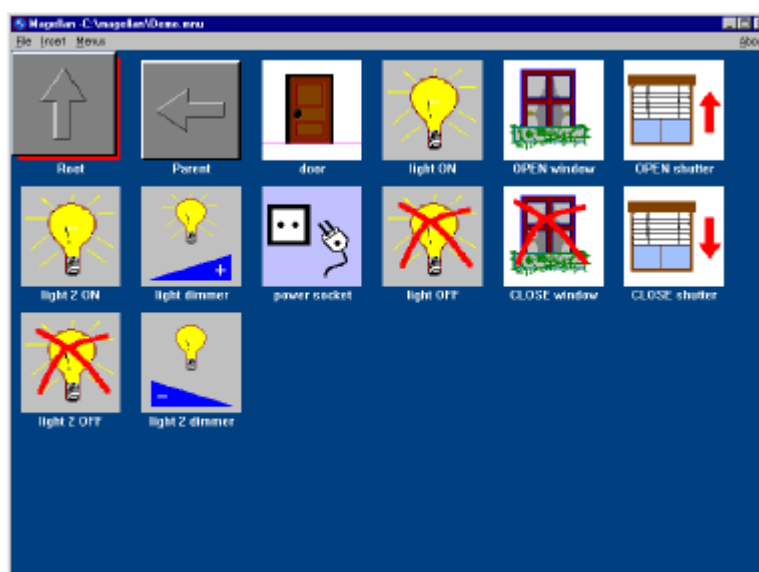


Figura 5.27: Quadro de controlo ambiental

O lançamento de aplicações Windows corresponde ao quadro ilustrado na figura 5.29 e integra neste caso ferramentas da família Microsoft Office, que se encontram representadas pelos seus ícones habituais. Repare-se na existência de um botão com o nome "Pointer", que permite passar ao modo de emulação de rato e teclado, sem ter lançado nenhuma aplicação do menu.

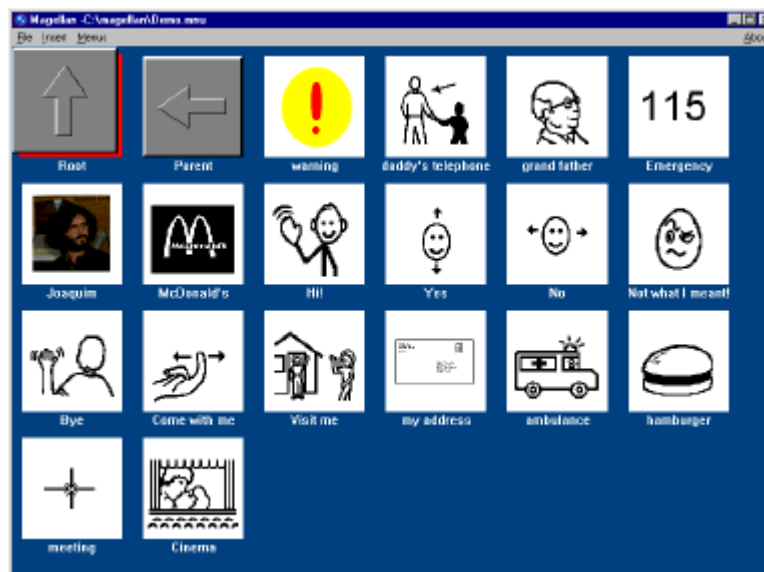


Figura 5.28: Quadro de apoio ao estabelecimento de chamadas telefónicas

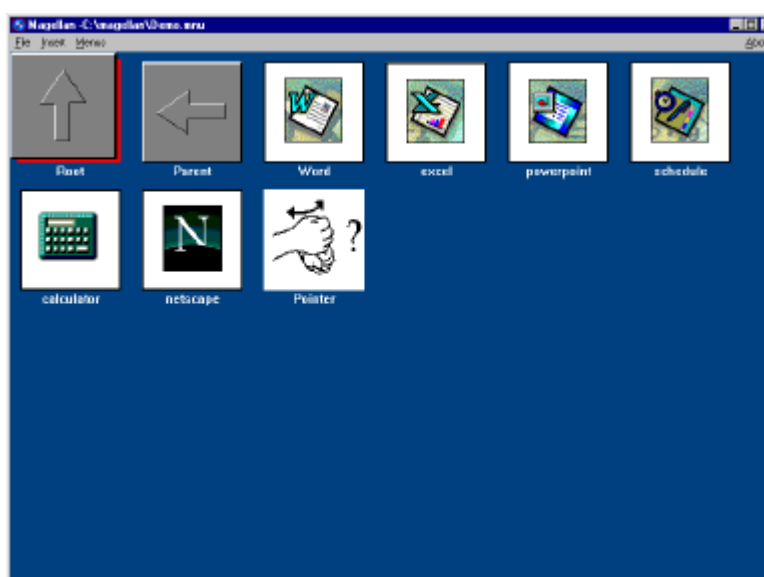


Figura 5.29: Quadro de lançamento de aplicações Windows

A descrição da aplicação Fernão de Magalhães e das questões técnicas associadas à sua implementação conclui a apresentação do trabalho de desenvolvimento realizado, pelo que é chegada a altura de se passar de forma sumária à instalação que foi efectuada na Casa da Maceda do Núcleo Regional do Norte da Associação Portuguesa de Paralisia Cerebral.

Capítulo 6

Experimentação prática: A instalação na APPC

Apesar de terem sido realizados diversos testes e implementações de laboratório que confirmaram o bom funcionamento do sistema, a validação junto dos utilizadores só poderia ter lugar através da sua implementação num ambiente real, para o que se seleccionou o edifício da APPC na Travessa da Maceda. Neste capítulo serão apresentados os resultados desta instalação, começando-se com uma descrição das características deste edifício, a que se seguirá a descrição do sistema que lá foi instalado (recursos físicos e interface com o utilizador).

6.1 Características do edifício na Travessa da Maceda

A Casa da Maceda é um edifício do Núcleo Regional do Norte da APPC, localizado na periferia da cidade do Porto, que serve como residência temporária de pessoas com Paralisia Cerebral. A sua lotação máxima é de 18 utentes em simultâneo, mas a população total é muito superior, uma vez que existe um grande número de utentes que a frequentam como centro de dia. Uma fotografia da fachada deste edifício, que possui dois andares, está representada na figura 6.1.

A planta do rés-do-chão está ilustrada na figura 6.2 e compreende nove áreas principais: uma cantina, uma cozinha, duas casas de banho, arrumador de entrada, uma recepção, uma sala de trabalhos e uma sala de reuniões. No primeiro andar, cuja planta se apresenta na figura 6.3, encontramos onze áreas principais: cinco quartos, três casas de banho, uma sala de TV, corredor e uma varanda.



Figura 6.1: A Casa da Maceda

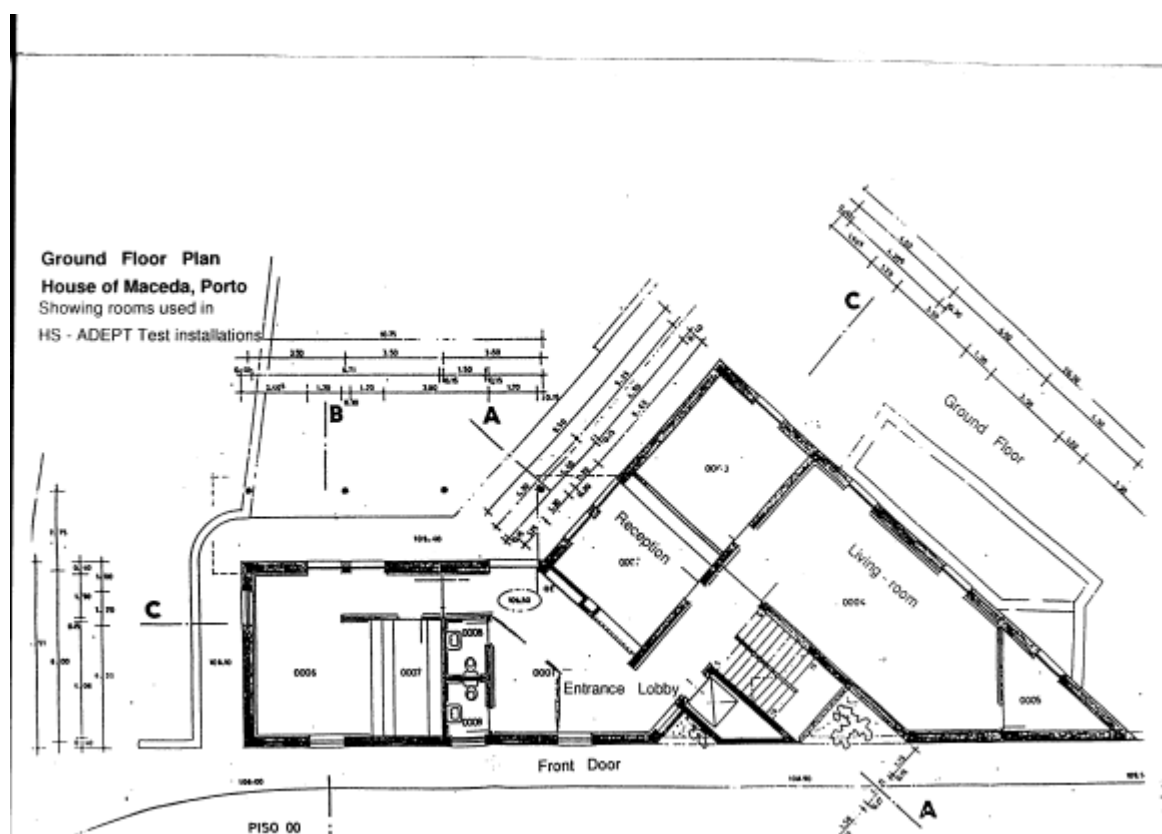


Figura 6.2: Casa da Maceda - Planta do R/C

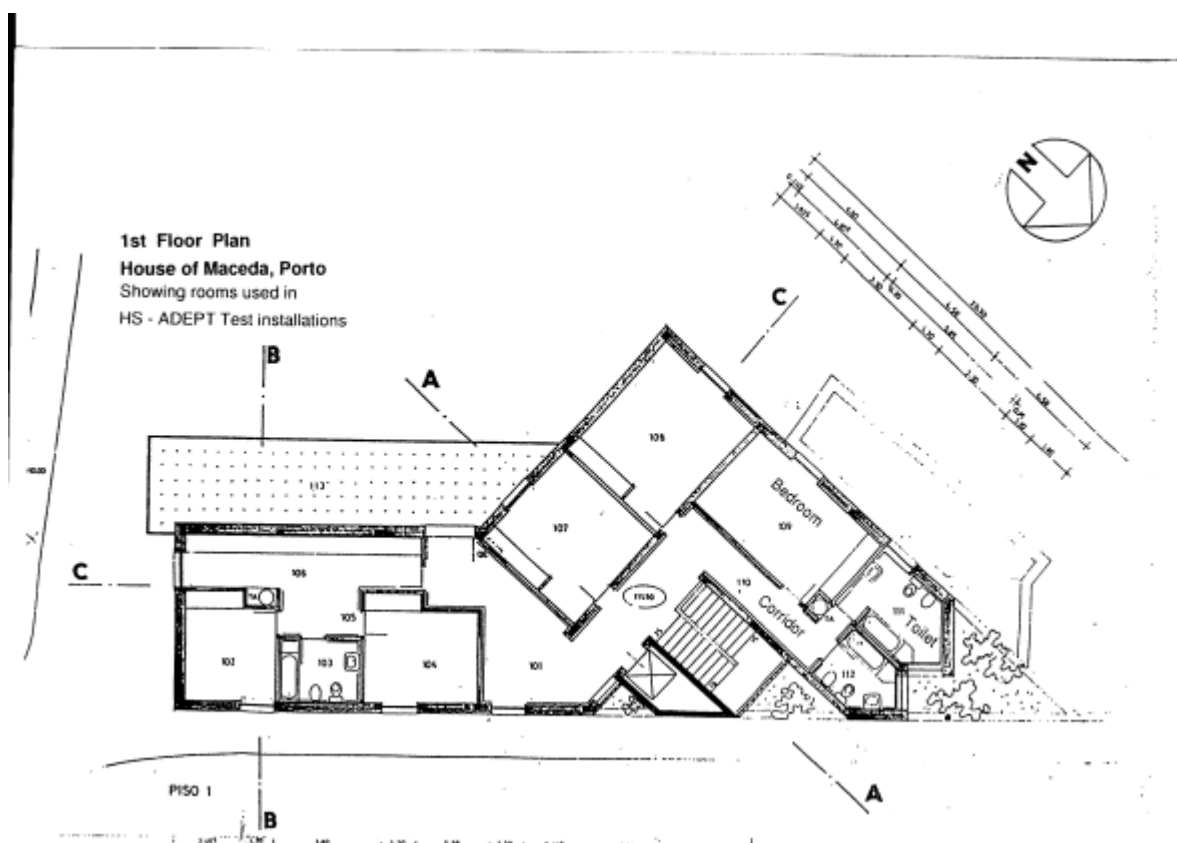


Figura 6.3: Casa da Maceda - Planta do 1º Andar

6.2 Instalação do sistema de controlo ambiental

A instalação realizada teve como objectivo principal a substituição das cartas EHS (European Home Systems), instaladas durante o projecto piloto TIDE HS-ADEPT [Coop93], por cartas CAN com a mesma funcionalidade, para aproveitar o investimento realizado em dispositivos actuadores e respectivos *drivers* de potência. Desta forma não foi necessário implementar uma nova instalação de energia, havendo um reaproveitamento total da infra-estrutura existente. A instalação de comando sofreu algumas alterações, uma vez que a filosofia do sistema domótico CAN integra a parte de comando local com o controlo de actuadores locais, e na filosofia implementada em EHS havia dois nós instalados, um para os comandos locais e o outro para controlo de actuadores.

A distribuição dos módulos CAN instalados nas várias áreas que integram os dois pisos da Casa da Maceda será agora apresentada na secção seguinte.

6.2.1 Hall de entrada

- **Módulo de controlo da porta exterior:** a partir de dois interruptores de comando, permite a abertura de um trinco e o controlo de um actuador de porta. Este conjunto está ilustrado na figura 6.4.



Figura 6.4: Módulo de controlo da Porta exterior

- **Módulo de iluminação com intensidade variável (*dimmer*):** ilustrado na figura 6.5, este módulo usa um interruptor para controlar uma lâmpada da seguinte forma:
 - Se o tempo durante o qual o interruptor está premido é curto, o controlo é do tipo liga / desliga.
 - Se mantivermos o interruptor premido, a intensidade da luz variará entre 0 e 100%.



Figura 6.5: Módulo de controlo lâmpada

6.2.2 Recepção

- **Fonte de alimentação** este módulo é responsável pela alimentação do barramento CAN, sendo indispensável para o funcionamento dos outros módulos. A tensão de distribuição é de 24 V, com uma corrente máxima de 10 A. A figura 6.6 ilustra esta fonte de alimentação.



Figura 6.6: Fonte de alimentação

- **Módulo RS232-CAN para controlo por computador** este módulo encontra-se localizado junto ao computador de controlo da instalação, como se ilustra na figura 6.7, permitindo controlar a rede domótica CAN a partir da porta RS-232C.



Figura 6.7: Computador de controlo da instalação

- **Módulo de tomada:** este módulo permite ligar e desligar a energia de uma tomada, sendo particularmente útil para comandar dispositivos como candeeiros, rádios, etc. Na figura 6.8 ilustra-se uma tomada controlada e o seu interruptor associado.



Figura 6.8: Tomada de energia controlada

Nesta área existe ainda um módulo de controlo de iluminação, como o que foi descrito para o *hall* de entrada.

6.2.3 Corredor junto às escadas

Nesta área foi instalado igualmente um módulo de iluminação, tal como o que existe *hall* de entrada.

6.2.4 Sala de trabalhos

- **Módulo de controlo de porta interior:** permite controlar o fecho e abertura de uma porta interior a partir de dois interruptores (do tipo *toggle*). Na figura 6.9 podemos ver o actuador da porta associado a este módulo.



Figura 6.9: Porta interior e respectivo actuador

Nesta área existem ainda dois módulos de iluminação e um módulo de tomada, com as mesmas características já descritas anteriormente.

6.3 Configuração do interface com o utilizador

O computador para controlo da instalação ficou instalado na recepção, com uma ligação directa por cabo ao módulo adaptador RS232-CAN. Este computador executa a aplicação Fernão de Magalhães com quadros para comunicação oral, lançamento de aplicações Windows e o controlo ambiental. Por razões internas da APPC, o estabelecimento de chamadas telefónicas não foi implementado.

A comunicação oral assistida tem sido usada principalmente para avaliação dos utilizadores e da sua capacidade de manipulação da interface, no que respeita aos mecanismos de selecção. O primeiro quadro de comunicação usado contém apenas cinco botões, que correspondem às mensagens *Olá*, *Adeus*, *Sim*, *Não* e *Gosto de ti*. No entanto, quadros mais complexos vão sendo criados para cada um dos utilizadores, em função das suas necessidades específicas.

As funções de lançamento de aplicações Windows estão contidas num quadro especial com botões para os programas do pacote Microsoft Office, acessórios Windows, Netscape e Vasco da Gama (programa de correio electrónico com varrimento). A emulação de teclado é a função mais utilizada, uma vez que a maioria dos utilizadores consegue trabalhar com o rato de alguma forma.

O controlo ambiental contém botões para todos dispositivos CAN instalados, mas as funções de controlo representadas são as mais simples, tais como ligar e desligar luzes, ou abrir e fechar portas. Mesmo os dispositivos reguláveis, como *dimmers*, são usados em modo liga / desliga.

Concluída a descrição da instalação na Casa da Maceda, resta para encerrar esta dissertação apresentar as principais conclusões decorrentes dos resultados obtidos e apontar direcções de trabalho futuro, que constituem o tema do capítulo final.

Capítulo 7

Conclusão

Este capítulo final começa por apresentar um conjunto de comentários referentes aos resultados obtidos, a que se seguem algumas considerações relativas a direcções possíveis para continuar o trabalho aqui apresentado.

7.1 Os resultados obtidos

Em termos gerais, o autor considera que o sistema desenvolvido constituiu um sucesso, quer por ter merecido a aprovação dos utentes e dos terapeutas com quem teve oportunidade de contactar na APPC, quer pelas reacções recolhidas nos locais onde foi apresentado [CSUN97, ENDI97, AAATE97]. O objectivo de se produzir um sistema de baixo custo foi satisfeito, quer em resultado de se ter (sempre que possível) optado por soluções normalizadas e generalizadamente acessíveis, quer pelo recurso ao CAN e pelo desenvolvimento de cartas muito simples, descritas no capítulo 4. Importa no entanto assinalar que a escolha destas cartas tem associado a necessidade de se proceder à respectiva produção, o que é menos prático do que adquirir unidades disponíveis no mercado. A escolha de outros protocolos / produtos para o controlo ambiental, e.g. o EIB, embora proporcionasse uma solução mais fácil de manter e reproduzir, implicaria em contrapartida custos substancialmente superiores. Repare-se no entanto que esta solução não está colocada de lado, já que o processo de construção de interfaces com o utilizador, descrito no capítulo 5, possibilita a criação dos ficheiros de comando para qualquer rede domótica cujo protocolo seja conhecido, ou para o qual estejam disponíveis bibliotecas que possibilitem o comando via RS-232C.

A independência entre a aplicação de interface com o utilizador e os restantes módulos instalados constitui uma vantagem que merece destaque adicional, já que possibilita uma implementação modular, facilitando a adaptação às necessidades especiais de cada utilizador, quer em termos das suas capacidades cognitivas, quer no que respeita aos tipos de ajudas a implementar (controlo ambiental, comunicação local assistida, comunicação telefónica assistida, trabalho em ambiente Windows). A melhor prova deste facto constitui a utilização continuada do Fe

não de Magalhães, tendo inclusivamente a APPC passado a dar apoio a esta aplicação e por seguido com o respectivo desenvolvimento.

7.2 Futuro da instalação

O futuro da instalação na Casa da Maceda passa pelo controlo do resto do edifício, nomeadamente dos quartos do primeiro andar, da sala de TV e das casas de banho. Para isso será necessário implementar uma instalação de distribuição de energia e uma cablagem de comando com respectivos interruptores.

O controlo da instalação através de uma cadeira de rodas, dispondo de computador portátil com ligação RF, permitirá que qualquer utilizador possa comandar a iluminação e as portas sem qualquer esforço, aumentando a sua liberdade de movimentos dentro do edifício. O controlo remoto por infravermelhos constitui outra das necessidades identificadas pelos utilizadores. Sob o ponto de vista do conforto, a inclusão dos módulos de controlo de electrodomésticos por infravermelhos permitirá a criação de cenários de funcionamento das salas de trabalho e de convívio, independentes da intervenção do pessoal de auxiliar.

Finalmente, a interligação do sistema de segurança e detecção de incêndios, através do desenvolvimento de uma carta adaptadora Central/CAN, constitui outra área de intervenção com interesse, uma vez que toda a infra-estrutura de comunicação com os sensores de fumo e de presença já se encontra desenvolvida nos módulos CAN de controlo de iluminação.

7.3 Perspectivas de futuro

Apesar de o seu percurso profissional não lhe permitir actualmente manter uma actividade neste domínio, o autor considera ainda assim útil apresentar aqui algumas breves reflexões, relativamente a dois tópicos principais, que poderão constituir percursos de continuidade do trabalho realizado:

- O desenvolvimento de bibliotecas de ficheiros que permitam expandir as opções pré-programadas de mensagens digitalizadas e de comandos para a rede domótica constitui naturalmente uma área de continuidade que deveria ser levada a cabo, no sentido de permitir à aplicação de interface com o utilizador dar uma resposta imediata às necessidades de um maior grupo de possíveis utilizadores. Apesar das facilidades apresentadas no capítulo 5, para a criação de interfaces adaptadas às necessidades particulares de cada indivíduo, o processo de criação de mensagens digitalizadas requer tempo e alguns conhecimentos técnicos, passando-se o mesmo no que respeita ao comando dos dispositivos presentes na rede domótica, pelo que é desejável que o leque de bibliotecas disponíveis para este efeito seja tão alargado quanto possível.

- A expansão para suportar outros protocolos usados em redes domóticas constitui outra área de intervenção principal, já que o acentuado crescimento da oferta de dispositivos EIB é tão vantajoso que se disponha da possibilidade de utilizar a aplicação Fernão de Magalhães com este tipo de produtos. O seu custo é ainda relativamente elevado, mas as perspectivas de crescimento deste mercado permitem prever uma redução dos preços e uma maior acessibilidade, pelo que seria desejável que o sistema suportasse directamente a integração de dispositivos compatíveis com esta especificação.

A utilização continuada das soluções desenvolvidas permitirá com certeza identificar outras áreas de intervenção desejáveis, pelo que seria importante passar a uma fase de testes mais sistémicos, com o objectivo de recolher informações que permitissem formular de maneira mais precisa a evolução a dar às soluções desenvolvidas e as eventuais correcções que se revelarem necessárias. As opiniões recolhidas até ao momento são suficientes para o autor se sentir satisfeito pelo trabalho realizado, mas a sua satisfação maior seria naturalmente constatar que os resultados apresentados justificaram que alguma das direcções de futuro referidas venha a ser implementada, no sentido de proporcionar aos utilizadores com paralisia cerebral um conjunto de ajudas técnicas que aumentem efectivamente a sua autonomia em ambiente doméstico.

Referências

- [AAATE97] J. M. Ferreira, R. M. Brito, M. A. Simões, M. L. Lourenço, "Integrated Access and Control in a Tele-Working Environment," *Proc. of the 12th Annual Conference Technology and Persons with Disabilities*, Los Angeles, EUA, 1997.
- [Apple] Apple Disability Resources <http://www.apple.com/education/k12/disability/>
- [Bati] Batibus for "on-line" buildings, the reference std for intelligent buildings, <http://www.invitro.com/batibus>
- [Blvd] The Boulevard - Environmental Control Units <http://www.blvd.com/ecu.html>
- [CEBus] CE Bus, <http://www.cebus.org>
- [CiA] CAN in Automation (CiA), <http://www.can-cia.de>
- [CMC] Children Medical Center: Cerebral Palsy, <http://www.med.virginia.edu/cmc/tutorials/cp>
- [Coop93] M. Cooper, A. Teixeira, J. M. Ferreira, "Implications of the Emerging Home System Technologies on Rehabilitation and the HS-ADEPT Project," *Proc. of the 2^d TIDE Congress*, Paris, 1995, pp. 409-412.
- [Cost98] L. J. Costa, *Mobilidade Virtual*, dissertação de mestrado submetida à FEUP, 1998.
- [CSUN97] R. M. Brito, M. S. Santiago, M. L. Lourenço, "A Home Office for Users with Special Needs," *Proc. of the AAATE 1997*, pp. 217-222.
- [Defnet] Paralisias Cerebrais, <http://www.defnet.org.br/paralis.htm>
- [Eche] Echelon Corporation, <http://www.echelon.com>
- [Eiba] EIBA, <http://www.eiba.org>
- [Ekbe95] I. Ekberg, R. Rontio, P. Kylänpää, "Smart Homes for Supporting Independent Living," *Proc. of the 2^d TIDE Congress*, Paris, 1995, pp. 383-386.
- [ENDI97] J. M. Ferreira, R. M. Brito, M. S. Santiago, M. L. Lourenço, "Um Posto de Tele-Trabalho para Pessoas com Necessidades Especiais," *trabalho apresentado ao Prémio de Inovação e Criatividade ENDIEL 97*.
- [Gall94] J. Gallant, "Speech-Synthesis and -recognition chips personalize consumer products," *EDN Europe*, October 20, 1994, pp. 51-57.
- [Gard95] L. Gardezabal, J. G-Abascal, N. Garay, A. Postigo, S. Urigoitia, M. J. Morentín, "Design and Integration of Alternative and Augmentative Interfaces under Windows Environment," *Proc. of the 2^d TIDE Congress*, Paris, 1995, pp. 336-339.
- [Khoh95] S.B. Khoh & R.T. McLaughlin, "Controller Area Network (CAN) for Industrial Applications", 1995, <http://www.csv.warwick.ac.uk/~esrjo/indus.html>

- [Miln93] M. Milner, "Rehabilitation Technology: Exploitation of R&D and Current Technologies," *Proc. of the 1st TIDE Congress*, Bruxelas, 1993, pp. 3-12.
- [Mins] Prentke Romich Company, *What is Minspeak?* 1993.
- [Nune96] R. Nunes, C. Sêro, "Edifícios Inteligentes: Conceitos e serviços," *Técnica - Revista de Engenharia* n.º 3, Setembro de 1996, pp. 5-15.
- [Philips] Controller Area Network (CAN), <http://www-us.semiconductors.philips.com/can/>
- [Poss] Possum Controls Limited <http://www.possum.co.uk>
- [Sant98] R. Santos, *Sistema Domótico EHS Suportado na Rede dos 220V*, dissertação de mestrado submetida à FEUP, 1998.
- [Sche93] M. Scherer, *Living in the State of Stuck*, Brookline Books, 1993, ISBN 0-914797-84-0.
- [Schofield96] M.J. Schofield, "Controller Area Network - CAN Application Layer", 1996, <http://www.omegas.co.uk/CAN/osimodel.html>
- [Smart] SmartHome, <http://www.smarhome.com>
- [Soed93] M. Soede, P. Röben, H. Knops, "Start of Creating an Overview of the RT Market," *Proc. of the 1st TIDE Congress*, Bruxelas, 1993, pp. 151-156.
- [Step97] C. Stephanidis, A. Savidis, D. Akoumianakis, "Unified Interface Development: A Step Towards User Interfaces for All," *Proc. of the AAATE*, 1997, pp. 24-28.
- [Teix95] A. J. Teixeira, *Casas Inteligentes: Adaptação desta Tecnologia para Deficientes e Idosos*, dissertação de mestrado submetida à FEUP, 1995.
- [Tetz93] S. von Tetzchner, "Aided Communication for Handicapped Children," A. Mital e W. Karwowski (editores), *Ergonomics in Rehabilitation*, New York: Taylor & Francis, 1993.
- [Tind95] Ken Tindell & Thor Kristmundsson, "CAN: Controller Area Network", 1995, <http://www.docs.uu.se/Eken/CAN.html>
- [Tull94] J. Tullman, M. F. Mitchell, H. C. Shave, "Impact of Augmentative and Alternative Communication on Natural Speech Production: Clinician's Perspective," *ISAAC Conference*, 1994 (poster).
- [User96] D. Poulson, M. Ashby, S. Richardson (editors), *A Practical Handbook on User-Centred Design for Assistive Technology*, 1996.
- [Wai] Web Accessibility Initiative <http://www.w3.org/WAI/>
- [WHO] World Health Organization: The International Classification of Impairments, Disabilities, and Handicaps, <http://www.who.int/whosis/icidh/icidh.html>
- [X10] X10 FAQ, <http://web.cs.ualberta.ca/~wade/HyperHome/Faq/faqoutline.html>

Anexo

O anexo desta dissertação é constituído por duas partes:

- Uma em formato impresso, que ocupa as páginas seguintes (código dos nós CAN e dossier de fabrico destas cartas de circuito impresso).
- Outra em formato electrónico, armazenada no CR-ROM que se encontra na bolsa fixada no verso da contracapa.

Código Fonte da Gestão de Comunicações CAN

```

/*****

    Arquitectura de Comunicacoes CAN

*****/

#pragma debug objectextend code
#include <reg51.h>

#define N_MESG 4

#define f_send_can_message

extern char MyID;

/*****
    Quando se usa o 82C200 CanController, este e'
    mapeado como memoria,usando os enderecos de
    0 ate 31
*****/

#define C_ControlRegister 0
#define C_CommandRegister 1
#define C_SR 2
#define C_IR 3
#define C_ACR 4
#define C_AMR 5
#define C_BTR0 6
#define C_BTR1 7
#define C_OCR 8
#define C_TR 9
#define C_TransmitBuffer 10
#define C_ReceiveBuffer 20
#define C_CDR 31

/*****
Acesso ao Controlador CAN 82C200 em paged mode
*****/
Pwrite(char pdata *addr,char value)
{
(*addr)=value;
}

Pread(char pdata *addr)
{
return(*addr);
}

/*****
Variaveis associadas ao buffer de leitura
*****/

```

```

static unsigned char CanBuffer[N_MESG][5]; /* Fila Circular */
static unsigned char CB_RP,CB_WP;          /* Apontador de escrita*/
                                           /* Apontador de leitura*/

char T_TYPE,T_FROM,T_TO,T_COM,T_VAL;
char i;
/*****
CAN INTERRUPT SERVICE
*****/
char pdata *CanAddress;

void can_int() interrupt 0
{
CanAddress=C_IR;
switch(Pread(CanAddress)&0x03)
{
case 1:
CanAddress=C_ReceiveBuffer; /* Apontador para o 1o Byte da Mensagem */
CanBuffer[CB_WP][0]=Pread(CanAddress++); /* ID */
CanAddress++;
CanBuffer[CB_WP][1]=Pread(CanAddress++); /* FROM */
CanBuffer[CB_WP][2]=Pread(CanAddress++); /* TO */
CanBuffer[CB_WP][3]=Pread(CanAddress++); /* MODO */
CanBuffer[CB_WP][4]=Pread(CanAddress++); /* VAL */
CanAddress=C_CommandRegister;
Pwrite(CanAddress,0x04); /* Release Buffer */
CB_WP++;
if(CB_WP==N_MESG)
CB_WP=0;
break;
case 2: /* Possibilidade de implementar*/
break; /* Escrita escondida no CAN */
}
}

/*****
CONTROL AREA NETWORK TRANSFER LAYER
*****/

void clear_can_buffer(void)
{
for(i=0;i<N_MESG;i++)
{
CanBuffer[i][0]=0;
CanBuffer[i][1]=0;
CanBuffer[i][2]=0;
CanBuffer[i][3]=0;
CanBuffer[i][4]=0;
}
}

bit new_can_message(void)
{
if(CB_RP==CB_WP)
return(0);
return(1);
}

```

```
void get_can_message(void)
{
    T_TYPE=CanBuffer[CB_RP][0];
    T_FROM=CanBuffer[CB_RP][1];
    T_TO=CanBuffer[CB_RP][2];
    T_COM=CanBuffer[CB_RP][3];
    T_VAL=CanBuffer[CB_RP][4];
    CB_RP++;
    if(CB_RP==N_MESG)
        CB_RP=0x00;
}

#ifdef f_send_can_message
void send_can_message(void)
{
    CanAddress=C_SR;
    while((Pread(CanAddress)&0x04)==0);
    CanAddress=C_TransmitBuffer;
    Pwrite(CanAddress++,T_TYPE);
    Pwrite(CanAddress++,0x04);
    Pwrite(CanAddress++,MyID);
    Pwrite(CanAddress++,T_TO);
    Pwrite(CanAddress++,T_COM);
    Pwrite(CanAddress++,T_VAL);
    CanAddress=C_CommandRegister;
    Pwrite(CanAddress,0x01);
}
#endif

#ifdef f_can_receive
bit can_receive(void)
{
    CanAddress=C_SR;
    if((Pread(CanAddress)&0x10)!=0x00)
        return(1);
    return(0);
}
#endif

#ifdef f_can_transmit
bit can_transmit(void)
{
    CanAddress=C_SR;
    if((Pread(CanAddress)&0x20)!=0x00)
        return(1);
    return(0);
}
#endif

#ifdef f_can_error
bit can_error(void)
{
    CanAddress=C_SR;
    if((Pread(CanAddress)&0x40)!=0x00)
        return(1);
    return(0);
}
```

```

}
#endif

#ifdef f_can_busy
bit can_busy(void)
{
CanAddress=C_SR;
if((Pread(CanAddress)&0x80)!=0x00)
    return(1);
return(0);
}
#endif

/*****
                        SETUP
*****/

void can_setup(void)
{
clear_can_buffer();
CB_WP=0;CB_RP=0;
/* setup CAN CONTROLLER */

CanAddress=C_ControlRegister;
Pwrite(CanAddress,0x01); /*ResetRequest*/

CanAddress=C_ACR;
Pwrite(CanAddress,0xf0);

CanAddress=C_AMR;
Pwrite(CanAddress,0xff);

CanAddress=C_BTR0;
Pwrite(CanAddress,0x83);

CanAddress=C_BTR1;
Pwrite(CanAddress,0x44);

CanAddress=C_OCR;
Pwrite(CanAddress,0xfa); /* antes usei 0xda */

CanAddress=C_ControlRegister;
Pwrite(CanAddress,0x46); /* Reference active + TI + RI */

IT0=1;
EX0=1;
}

```

Código Fonte do CAN sobre RS-232

```
#pragma code debug
#include <reg51.h>
#include <stdio.h>
#include "can.h51"

extern unsigned char T_TYPE,T_FROM,T_TO,T_COM,T_VAL;

char RS_DATA;

void rs_setup(void)
{
PCON=(PCON&0x7F);
SCON=0x52;
TMOD=0x20;
TCON=0x40;
TH1=0xfd;
TL1=0xfd;
TR1=1;
TI=1;
ES=0;
}

bit new_rs(void)
{
return(RI);
}

void put_rs(char valor)
{
while(!TI);
TI=0;
SBUF=valor;
}

char get_rs(void)
{
while(!RI);
RS_DATA=SBUF;
RI=0;
return(RS_DATA);
}

bit get_trama(void)
{
int Xsum;
char start;
if(get_rs()==0x05)
{
T_TYPE=get_rs();
T_FROM=get_rs();
```

```
        T_T0=get_rs();
        T_COM=get_rs();
        T_VAL=get_rs();
        Xsum=5+T_TYPE+T_FROM+T_T0+T_COM+T_VAL;
        Xsum%=254;

    if(Xsum==get_rs())
        return(1);
    }
    else put_rs(RS_DATA);

return(0);
}

void put_trama(void)
{
    int Xsum;
    put_rs(5);
    put_rs(T_TYPE);
    put_rs(T_FROM);
    put_rs(T_T0);
    put_rs(T_COM);
    put_rs(T_VAL);
    Xsum=5+T_TYPE+T_FROM+T_T0+T_COM+T_VAL;
    Xsum%=254;
    put_rs((char)Xsum);
}

void rs_2_can(void)
{
    if(get_trama()==1);
        send_can_message();
}

void can_2_rs(void)
{
    if(new_can_message())
    {
        get_can_message();
        put_trama();
    }
}
```

Código Fonte de um Nó CAN do tipo Iluminação

```

/*****
Programa:      Light Can Node
Data:         25/10/97
Notes:        Client Server Version
              Motion detector client
              Extended Command Set

*****/

#pragma debug objectextend code
#include <reg51.h>
#include "can.h51"
#include "..\protocol.h51"

#define TIME_MOVING 500

#define LIGHT_ON 0x01
#define LIGHT_OFF 0x02

#define MODE_NORMAL 0x01
#define MODE_TIMER 0x02
#define MODE_DELAY 0x03

#define LOCAL 0x00
#define REMOTE 0x01

#define DM_UNKNOWN 0x00
#define DM_DAY 0x01
#define DM_NIGHT 0x02

#define P_ON 0x00

#define SMOKE_OFF 0x01
#define SMOKE_ON 0x00

#define FALSE 0
#define TRUE 1
extern char T_TYPE, T_FROM, T_TO, T_COM, T_VAL;
int Timer;
int Tempo;
char LightState;
char MachineMode;
char Command_Mode;
char Day_Mode;
char MyID;

sbit S1 = P2^7;
sbit S2 = P2^6;
sbit SW1 = P2^5;
sbit SW2 = P2^4;
sbit Smoke = P2^1;
sbit Present = P2^0;

```

```

/*****

```

``` SETUP CAN CONTROLLER ```

```

*****/

```

```

void setup(void)
{
    EA=0; /* Disable all interrupts while setting up the system */
    S1=1;

    can_setup();
    TMOD|=0x01; /* Ligar Modo 16bits timer 0 */
    TH0=0xd8; /* Contar 10.000 us */
    TL0=0xf0;
    ET0=1; /* Enable Overflow Interrupt */
    TR0=1; /* Arranque do Timer0 */

    EA=1;
}

```

```

void TimerInt() interrupt 1
{
    TH0=0xd8;
    TL0=0xf0;

    if (Timer)
        Timer--;
    if (Tempo)
        Tempo--;
}

```

```

void main(void)
{
    bit lastSW1, lastSmoke, Send_Status;
    char LastLightState;
    MyID=P1;
    setup();

    MachineMode=MODE_NORMAL;
    Command_Mode=LOCAL;
    LightState=LIGHT_OFF;
    Timer=0;
    Tempo=0;
    lastSW1=SW1;
    lastSmoke=Smoke;
    Send_Status=FALSE;
    LastLightState=LightState;

    while(1)
    {

        if((SW1!=lastSW1)&&(Timer==0))
        {
            if(Command_Mode==REMOTE)

```

```

        {
            T_TYPE=TYPE_LAMP;
            T_FROM=MyID;
            T_TO=255;
            T_COM=LAMP_SW1;
            T_VAL=(char)SW1;
            send_can_message();
            Timer=30;
        }

    else if(SW1==0)
    {
        MachineMode=MODE_NORMAL;
        if(LightState==LIGHT_ON)
        {
            LightState=LIGHT_OFF;
        }
        else LightState=LIGHT_ON;
        Timer=30;
    }
}

lastSW1=SW1;

if((Smoke!=lastSmoke)&&(Timer==0))
{
    T_TYPE=TYPE_SMOKE;
    T_FROM=MyID;
    T_TO=255;
    if(lastSmoke==1)
        T_COM=SMOKE_DETECTED;
    else T_COM=SMOKE_UNDETECTED;
    T_VAL=0;
    send_can_message();
    Timer=30;
}
lastSmoke=Smoke;

switch(MachineMode)
{
    case MODE_NORMAL:

        if((LightState==LIGHT_ON)||((Day_Mode==DM_NIGHT)&&(Present==P_ON)))
            S1=0;
            else S1=1;
                break;

        case MODE_TIMER:
            if(Tempo!=0)
            {
                S1=0;
            }
            else
            {
                MachineMode=MODE_NORMAL;

```

```

                                LightState=LIGHT_OFF;
                                }
                                break;
        case MODE_DELAY:
            if(Tempo!=0)
            {
                S1=1;
            }
            else {
                MachineMode=MODE_NORMAL;
                LightState=LIGHT_ON;
            }
            break;
    }

    if(LightState!=LastLightState)
        Send_Status=TRUE;

    LastLightState=LightState;

    if((Send_Status==TRUE)&&(Timer==0))
    {
        Send_Status=FALSE;
        T_TYPE=TYPE_STATE;
        T_FROM=MyID;
        T_TO=255;
        T_COM=TYPE_LAMP;
        T_VAL=LightState;
        send_can_message();
        Timer=30;
    }

    if(new_can_message())
    {
        get_can_message();
        switch(T_TYPE)
        {
            case TYPE_LAMP:
                if((T_TO==MyID)|| (T_TO==255))
                switch(T_COM)
                {
                    case LAMP_ON:
                        MachineMode=MODE_NORMAL;
                        LightState=LIGHT_ON;
                        break;

                    case LAMP_OFF:
                        MachineMode=MODE_NORMAL;
                        LightState=LIGHT_OFF;
                        break;

                    case LAMP_TOGGLE:
                        MachineMode=MODE_NORMAL;
                        if(LightState==LIGHT_ON)
                            LightState=LIGHT_OFF;
                        else LightState=LIGHT_ON;
                }
            }
    }

```

```

        break;
    case LAMP_TIMER:
        MachineMode=MODE_TIMER;
        Tempo=T_VAL*100;
        break;
    case LAMP_DELAY:
        MachineMode=MODE_DELAY;
        Tempo=T_VAL*100;
        break;
    case LAMP_REMOTE:
        if(T_VAL==1)
            Command_Mode=REMOTE;
        else Command_Mode=LOCAL;

    }
    break;

case TYPE_LIGHTDETECTOR:
    if((T_TO==MyID)|| (T_TO==255))
    switch(T_COM)
    {
    case LIGHTDETECTOR_DAY:
        Day_Mode=DM_DAY;
        break;
    case LIGHTDETECTOR_NIGHT:
        Day_Mode=DM_NIGHT;
        break;
    default:
        Day_Mode=DM_UNKNOWN;
        break;
    }
    break;

case TYPE_SERVICE:
    if((T_TO==MyID)|| (T_TO==255))
    switch(T_COM)
    {
    case SERVICE_QUERY:
        T_TYPE=TYPE_STATE;
        T_FROM=MyID;
        T_TO=255;
        T_COM=TYPE_LAMP;
        T_VAL=LightState;
        send_can_message();
        break;
    }
    break;
    }
}
}
}

```

Código Fonte de um Nó CAN do tipo Dimmer

```

/*****
Programa:      Dimmer
Data:         10/11/96
Notes:

*****/

#pragma debug objectextend code
#include <reg51.h>
#include "can.h51"
#include "..\protocol.h51"

extern char T_TYPE,T_FROM,T_TO,T_COM,T_VAL;
unsigned int Timer,TimeRun;
char DimmerState;
char MyID;

sbit S1=P2^0;
sbit S2=P2^1;
sbit S3=P2^2;
sbit S4=P2^3;
sbit LED=P2^4;
sbit SW1=P2^7;

/*****

                SETUP CAN CONTROLLER

*****/

void setup(void)
{
EA=0; /* Disable all interrupts while seting up the system */
DimmerState=0;
Timer=0;
TimeRun=1;
S1=0;
S2=0;
S3=0;
S4=0;
SW1=1;
can_setup();
TMOD|=0x01; /* Ligar Modo 16bits timer 0*/
TH0=0xd8; /* Contar 10.000 us */
TL0=0xf0;
ET0=1; /* Enable OverFlow Interrup*/
TR0=1; /* Arranque do Timer0*/

EA=1;
}

void TimerInt() interrupt 1
{

```

```
TH0=0xd8;
TL0=0xf0;

if (Timer)
    Timer--;
TimeRun++;
}

void main(void)
{
bit lastSW1,SW1Down;
char lastState;
MyID=P1;
Timer=0;
setup();
lastSW1=SW1;
SW1Down=0;
while(1)
{
    if((Timer==0)&&(SW1==0)&&(lastSW1!=SW1))
    {
        SW1Down=1;
        TimeRun=1;
        Timer=10;
        lastState=DimmerState;
    }
    lastSW1=SW1;

    if((Timer==0)&&(SW1==1)&&(SW1Down==1))
    {
        SW1Down=0;
        Timer=10;
        if(TimeRun<100)
        {
            if(lastState==0)
                DimmerState=15;
            else DimmerState=0;
        }
    }

    if((SW1Down==1)&&(TimeRun<1510))
    {
        LED=0;
        DimmerState=TimeRun/100;
    }
    else LED=1;

    if(TimeRun>1505)
        TimeRun=1505;

    if(DimmerState&0x01)
        S1=1;
    else S1=0;
}
```

```

        if(DimmerState&0x02)
            S2=1;
        else S2=0;

        if(DimmerState&0x04)
            S3=1;
        else S3=0;

        if(DimmerState&0x08)
            S4=1;
        else S4=0;

    if(new_can_message())
    {
        get_can_message();
        switch(T_TYPE)
        {
            case TYPE_DIMMER:
                if((T_T0==MyID)|| (T_T0==255))
                    switch(T_COM)
                    {
                        case DIMMER_ON:
                            DimmerState=15;
                            break;
                        case DIMMER_OFF:
                            DimmerState=0;
                            break;
                        case DIMMER_TOGGLE:
                            if(DimmerState==0)
                                DimmerState=15;
                            else DimmerState=0;
                            break;
                        case DIMMER_INC:
                            if(DimmerState<15)
                                DimmerState++;
                            break;
                        case DIMMER_DEC:
                            if(DimmerState>0)
                                DimmerState--;
                            break;
                        case DIMMER_SET:
                            if((T_VAL>=0)|| (T_VAL<16))
                                DimmerState=T_VAL;
                            break;
                    }
                break;
        }
    }
}

```

Código Fonte de um Nó CAN do tipo IntDoor

```

/*****
Programa:      Internal door
Data:         17/03/98
Notes:        Client Server
              Remote

*****/

#pragma debug objectextend code
#include <reg51.h>
#include "can.h51"
#include "..\protocol.h51"

#define TIME_MOVING 1500

#define LOCAL 0x00
#define REMOTE 0x01

extern char T_TYPE, T_FROM, T_TO, T_COM, T_VAL;
int Timer;
char MyID;
char Command_Mode;
char DoorStatus;

sbit S1=P2^7;
sbit S2=P2^6;
sbit SW1=P2^5;
sbit SW2=P2^4;

/*****

                SETUP CAN CONTROLLER

*****/

void setup(void)
{
EA=0; /* Disable all interrupts while seting up the system */
Timer=0;
S1=1;
can_setup();
TMOD|=0x01; /* Ligar Modo 16bits timer 0*/
TH0=0xd8; /* Contar 10.000 us */
TL0=0xf0;
ET0=1; /* Enable OverFlow Interrup*/
TR0=1; /* Arranque do Timer0*/

EA=1;
}

void TimerInt() interrupt 1
{
TH0=0xd8;

```

```

TL0=0xf0;

if (Timer)
    Timer--;
}

void main(void)
{
bit lastSW1, lastSW2;
DoorStatus=DOOR_CLOSE;
Command_Mode=LOCAL;
setup();
MyID=P1;
lastSW1=SW1;
lastSW2=SW2;

while(1)
{
if(((SW1!=lastSW1)||(SW2!=lastSW2))&&(Timer==0))
{
if(Command_Mode==REMOTE)
{
T_TYPE=TYPE_INTDOOR;
T_FROM=MyID;
T_T0=255;
T_COM=DOOR_SW;
T_VAL=(char)(SW1&SW2);
send_can_message();
Timer=20;
}
else if((SW1==0)||(SW2==0))
{
if(DoorStatus==DOOR_OPEN)
    DoorStatus=DOOR_CLOSE;
else DoorStatus=DOOR_OPEN;
Timer=20;
}
}
lastSW1=SW1;
lastSW2=SW2;

if(DoorStatus==DOOR_OPEN)
    S1=0;
else S1=1;

if(new_can_message())
{
get_can_message();
switch(T_TYPE)
{
case TYPE_INTDOOR:
    if((T_T0==MyID)||(T_T0==255))
        switch(T_COM)
        {
case DOOR_OPEN:
DoorStatus=DOOR_OPEN;

```

```

        break;
    case DOOR_CLOSE:
        DoorSts=DOOR_CLOSE;
        break;
    case DOOR_TOGGLE:
        if(DoorStatus==DOOR_OPEN)
            DoorSts=DOOR_CLOSE;
        else DoorSts=DOOR_OPEN;
        break;
    case DOOR_REMOTE:
        if(T_VAL==1)
            Command_Mode=REMOTE;
        else Command_Mode=LOCAL;

    }
    break;
case TYPE_SERVICE:
    if((T_TO==MyID)||(T_TO==255))
    switch(T_COM)
    {
    case SERVICE_QUERY:
        T_TYPE=TYPE_STATE;
        T_FROM=MyID;
        T_TO=255;
        T_COM=TYPE_INTDOOR;
        T_VAL=DoorStatus;
        send_can_message();
        break;
    }
    break;

    }

}
}
}

```

Código Fonte de um Nó CAN do tipo Tomada

```

/*****
Programa:      power socket
Data:         17/03/98
Notes:
              remote version
              Client Server Version

*****/

#pragma debug objectextend code
#include <reg51.h>
#include "can.h51"
#include "..\protocol.h51"

#define TIME_MOVING  500

#define LOCAL 0x00
#define REMOTE 0x01

extern char T_TYPE,T_FROM,T_TO,T_COM,T_VAL;
int Timer;
char MyID;
char Command_Mode;

sbit S1=P2^7;
sbit S2=P2^6;
sbit SW1=P2^5;
sbit SW2=P2^4;

/*****

                SETUP CAN CONTROLLER

*****/

void setup(void)
{
EA=0; /* Disable all interrupts while seting up the system */
Timer=0;
S1=1;
can_setup();
TMOD|=0x01; /* Ligar Modo 16bits timer 0*/
TH0=0xd8; /* Contar 10.000 us */
TL0=0xf0;
ET0=1; /* Enable OverFlow Interrup*/
TR0=1; /* Arranque do Timer0*/

EA=1;
}

void TimerInt() interrupt 1
{

```

```

TH0=0xd8;
TL0=0xf0;

if (Timer)
    Timer--;
}

void main(void)
{
    bit lastSW1;
    char SocketStatus;
    SocketStatus=SOCKET_OFF;
    Command_Mode=LOCAL;
    setup();
    MyID=P1;
    lastSW1=SW1;

    while(1)
    {

if((SW1!=lastSW1)&&(Timer==0))
    {
        if(Command_Mode==REMOTE)
        {
            T_TYPE=TYPE_POWER;
            T_FROM=MyID;
            T_TO=255;
            T_COM=SOCKET_SW1;
            T_VAL=(char)SW1;
            send_can_message();
            Timer=20;
        }
        else if(SW1==0)
        {
            if(SocketStatus==SOCKET_ON)
                SocketStatus=SOCKET_OFF;
            else SocketStatus=SOCKET_ON;
            Timer=20;
        }
    }
    lastSW1=SW1;

    if(SocketStatus==SOCKET_ON)
        S1=0;
    else S1=1;

    if(new_can_message())
    {
        get_can_message();
        switch(T_TYPE)
        {
            case TYPE_POWER:
                if((T_TO==MyID)|| (T_TO==255))
                    switch(T_COM)
                    {
                        case SOCKET_ON:

```

```

        SocketStatus=SOCKET_ON;
        break;
    case SOCKET_OFF:
        SocketStatus=SOCKET_OFF;
        break;
    case SOCKET_TOGGLE:
        if(SocketStatus==SOCKET_ON)
            SocketStatus=SOCKET_OFF;
        else SocketStatus=SOCKET_ON;
        break;
    case SOCKET_REMOTE:
        if(T_VAL==1)
            Command_Mode=REMOTE;
        else Command_Mode=LOCAL;

    }
    break;
case TYPE_SERVICE:
    if((T_TO==MyID)||(T_TO==255))
        switch(T_COM)
        {
        case SERVICE_QUERY:
            T_TYPE=TYPE_STATE;
            T_FROM=MyID;
            T_TO=255;
            T_COM=TYPE_POWER;
            T_VAL=SocketStatus;
            send_can_message();

            break;
        }

    break;

}

}

}

```

Código Fonte de um Nó CAN do tipo Persiana

```

/*****

        shutter
        Data: 15/8/96

        Fila Leitura: FIFO

*****/

#pragma debug objectextend code
#include <reg51.h>
#include "can.h51"
#include "..\protocol.h51"

#define DIR_OPEN 1
#define DIR_CLOSE 2
#define TIME_MOVING 1500

extern char T_TYPE,T_FROM,T_TO,T_COM,T_VAL;
int TimerMove;
int Timer;
char MoveDIR;
char MyID;

sbit S1=P2^0;
sbit S2=P2^1;
sbit SW1=P2^2;
sbit SW2=P2^3;

/*****

        SETUP CAN CONTROLLER

*****/

void setup(void)
{
EA=0; /* Disable all interrupts while seting up the system */
TimerMove=0;
Timer=0;
S1=1;
S2=1;
SW1=1;
SW2=1;
can_setup();
TMOD|=0x01; /* Ligar Modo 16bits timer 0*/
TH0=0xd8; /* Contar 10.000 us */
TL0=0xf0;
ET0=1; /* Enable OverFlow Interrup*/
TR0=1; /* Arranque do Timer0*/

EA=1;

```

```

}

void TimerInt() interrupt 1
{
  TH0=0xd8;
  TL0=0xf0;

  if (TimerMove)
    TimerMove--;
  if(Timer)
    Timer--;
}

void main(void)
{
  bit lastSW1,lastSW2;
  setup();
  lastSW1=SW1;
  lastSW2=SW2;
  MyID=P1;
  while(1)
  {
    if(Timer==0)
      if((SW1!=lastSW1)&&(SW1==0)&&(SW2==1))
      {
        Timer=40;
        if(TimerMove)
          TimerMove=0;
        else {
          MoveDIR=DIR_OPEN;
          TimerMove=TIME_MOVING;
        }
      }
    if(Timer==0)
      if((SW2!=lastSW2)&&(SW2==0))
      {
        Timer=40;
        if(TimerMove)
          TimerMove=0;
        else {
          MoveDIR=DIR_CLOSE;
          TimerMove=TIME_MOVING;
        }
      }
    lastSW1=SW1;
    lastSW2=SW2;

    if(TimerMove)
    {
      switch(MoveDIR)
      {
        case DIR_CLOSE:
          S1=0;
          S2=1;
          break;

        case DIR_OPEN:

```

```

        S1=1;
        S2=0;
        break;
    }
}
else {
    S1=1;
    S2=1;
}

if(new_can_message())
{
    get_can_message();
    switch(T_TYPE)
    {
        case TYPE_SHUTTER:
            if((T_T0==MyID)|| (T_T0==255))
                switch(T_COM)
                {
                    case SHUTTER_OPEN:
                        if(TimerMove)
                            TimerMove=0;
                        else {
                            DIR=DIR_OPEN;
                            TimerMove=TIME_MOVING;
                        }
                        break;
                    case SHUTTER_CLOSE:
                        if(TimerMove)
                            TimerMove=0;
                        else {
                            DIR=DIR_CLOSE;
                            TimerMove=TIME_MOVING;
                        }
                        break;
                }
            break;
    }
}
}
}

```

Código Fonte de um Nó CAN do tipo Janela

```

/*****

        Janela
        Data: 15/8.06

        Fila Leitura: FIFO

*****/

#pragma debug objectextend code
#include <reg51.h>
#include "can.h51"
#include "..\protocol.h51"

#define DIR_OPEN 1
#define DIR_CLOSE 2
#define TIME_MOVING 1500

extern char T_TYPE, T_FROM, T_TO, T_COM, T_VAL;
int TimerMove;
int Timer;
char MoveDIR;
char MyID;

sbit S1=P2^0;
sbit S2=P2^1;
sbit SW1=P2^2;
sbit SW2=P2^3;
sbit Sensor=P2^4;

/*****

        SETUP CAN CONTROLLER

*****/

void setup(void)
{
    EA=0; /* Disable all interrupts while seting up the system */
    TimerMove=0;
    Timer=0;
    SW1=1;
    SW2=1;

    can_setup();
    TMOD|=0x01; /* Ligar Modo 16bits timer 0 */
    TH0=0xd8; /* Contar 10.000 us */
    TL0=0xf0;
    ET0=1; /* Enable OverFlow Interrup */
    TR0=1; /* Arranque do Timer0 */

    EA=1;

```

```

}

void TimerInt() interrupt 1
{
  TH0=0xd8;
  TL0=0xf0;

  if (TimerMove)
    TimerMove--;
  if(Timer)
    Timer--;
}

void main(void)
{
  bit lastSW1,lastSW2;
  setup();
  lastSW1=SW1;
  lastSW2=SW2;
  MyID=P1;
  while(1)
  {
    if(Timer==0)
      if((SW1==0)&&(lastSW1==1)&&(SW2==1))
      {
        if(TimerMove)
          TimerMove=0;
        else
        {
          MoveDIR=DIR_OPEN;
          TimerMove=TIME_MOVING;
        }
        Timer=40;
      }
    if(Timer==0)
      if((SW2==0)&&(lastSW2==1))
      {
        if(TimerMove)
          TimerMove=0;
        else
        {
          MoveDIR=DIR_CLOSE;
          TimerMove=TIME_MOVING;
        }
        Timer=40;
      }

    lastSW1=SW1;
    lastSW2=SW2;

    if(TimerMove)
    {
      switch(MoveDIR)
      {
        case DIR_CLOSE:
          S1=0;

```

```

                                S2=1;
                                break;
                                case DIR_OPEN:
                                    S1=1;
                                    S2=0;
                                    break;
                                }
                            }
                        else {
                            S1=1;
                            S2=1;
                        }
                    if(new_can_message())
                    {
                        get_can_message();
                        switch(T_TYPE)
                        {
                            case TYPE_WINDOW:
                                if((T_TO==MyID)|| (T_TO==255))
                                    switch(T_COM)
                                    {
                                        case WINDOW_OPEN:
                                            if(TimerMove)
                                                TimerMove=0;
                                            else
                                            {
                                                DIR=DIR_OPEN;
                                                TimerMove=TIME_MOVING;
                                            }
                                            break;
                                        case WINDOW_CLOSE:
                                            if(TimerMove)
                                                TimerMove=0;
                                            else
                                            {
                                                DIR=DIR_CLOSE;
                                                TimerMove=TIME_MOVING;
                                            }
                                            break;
                                    }
                                break;
                            }
                        }
                    }
                }
            }
        }
    }

```

