Actuador Linear Para Teste De Teclas Na Indústria Automóvel

Artur Jorge de Sousa Pereira Lourenço

Mestrado em Engenharia Electrotécnica e de Computadores

2011

Dissertação apresentada ao ISEP no âmbito do Mestrado em Engenharia Electrotécnica e de Computadores.

Candidato:

Artur Jorge de Sousa Pereira Lourenço

Orientação:

Dr. André Fidalgo

Empresa:

Preh Portugal, Lda

Supervisão:

Eng. José Costa

Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização de Automação e Sistemas

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

Novembro de 2011

RESUMO

As indústrias de componentes e acessórios automóveis são um elo fundamental no processo produtivo da indústria automóvel. Neste leque industrial encontra-se a *Preh Portugal, Lda*, como empresa fornecedora de componentes electrónicos, mais concretamente, painéis de controlo de climatização.

Os painéis fornecidos pela *Preh* aos seus clientes encontram-se sujeitos a rigorosos testes de qualidade e funcionalidade. Neste sentido o teste funcional das teclas surge, relacionando o curso da tecla em função de uma força actuante. Esta relação está comprometida com uma curva característica padrão para o tipo de tecla. Para além destes compromissos, também é necessário que a tecla feche e abra o seu contacto eléctrico.

Esta tese foca-se no desenvolvimento do teste de teclas, apresentando uma alteração ao sistema actual com a introdução de um sistema embebido, no intuito de flexibilizar o sistema de teste e reduzindo custos. O sistema embebido pretende dar capacidade de processamento ao teste e, desta forma, substituir o actual computador como elemento de processamento.

A solução implementada consistiu numa mudança estrutural, através da inclusão do sistema embebido entre o computador e o sistema de deslocamento. Passando o foco central do processo de teste a residir no sistema embebido, este tem de estabelecer comunicações com os restantes elementos intervenientes no teste. Estabelece comunicações série RS-232 com o sistema de deslocamento (leitura do curso e força na tecla), Ethernet com o computador (comandos, parâmetros e resultados) e CAN com o painel de controlo de climatização (fecho/abertura do contacto eléctrico).

A concretização deste projecto resultou numa nova estrutura e aplicação, a qual é facilmente integrada na linha de produção com as vantagens de ser menos onerosa e mais flexível, conforme o pretendido.

ABSTRACT

The industries of automobile components and accessories are a fundamental link in the productive process of the automobile industry. In this range of industries we find *Preh Portugal, Lda*, as a supplier of electronic components more specifically, climate control panels.

The panels *Preh* supplies to her clients are subjected to rigorous quality and functionality tests. In this sense the functional test of keys arises relating the course of a key according to an acting force. This relationship is compromised with a standard characteristic curve pattern for the key type. Beyond these compromises it is also necessary that the key closes and opens its electrical contact.

This thesis focuses on the development of key testing, featuring a modification to the current system, with the introduction of an embedded system, aiming to ease the testing system and reduce costs. The embedded system intends to give processing capability to the test and as such, substitute the current computer as the processing element.

The implemented solution consisted in a structural change, through the inclusion of the embedded system, between the computer and the,displacement system. Having the central focus of the testing process now residing on the embedded system, it has to establish communications with the remaining elements involved in the test. It establishes serial communications RS-232 with the displacement system (reading the displacement of and force on the key), Ethernet with the computer (commands, parameters and results) and CAN with the climate control panels (closing/opening of the electric contact).

The realization of this project resulted in a new structure and application, which can be easily integrated with the production line with the advantage of being less costly and more flexible, as was intended.

AGRADECIMENTOS

Ao Eng. José Costa pela oportunidade concedida para a realização desta dissertação na *Preh*, para além da sua disponibilidade e apoio no decorrer do desenvolvimento do projecto.

Uma palavra de apreço a todos os elementos do departamento de engenharia da *Preh*, em especial ao Eng. Sílvio Granjo.

Ao Dr. André Fidalgo, orientador da dissertação, pelo seu apoio incondicional, disponibilidade e confiança depositada na minha pessoa, um sincero agradecimento.

À minha família pelo amor, apoio e compreensão demonstrado ao longo deste percurso.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

ACRÓNIMOS

A/D	*Analógico / Digital*
ASTM	*American Society for Testing and Materials*
bps	*bits por segundo*
CAN	*Controller Area Network*
CLR	*Common Language Runtime*
CPU	*Central Processing Unit*
CRC	*Cyclic Redundancy Check*
CSMA/CD	*Carrier Sense Multiple Access with Coilision Detection*
D/A	*Digital / Analógico*
DCE	*Data Communication Equipment*
DLL	*Data Link Layer*
DTE	*Data Terminal Equipment*
ECU	*Electronic Control Unit*
EIA	*Electronic Industries Association*
GPIO	*General Purpose Input/Output*
HAL	*Hardware Abstraction Layer*
I/O	*Input/Output*
I2C	*Inter-Integrated Circuit*
IEEE	*Institute of Electrical and Electronic Engineers*
IP	*Internet Protocol*
ISO	*Internacional Organization for Standardization*
LAN	*Local Area Network*
LLC	*Logic Link Control*
MAC	*Medium Acess Control*
NETMF	*.Net Micro Framework*
OSI	*Open System Interconnection*
PAL	*Platform Abstraction Layer*
PCB	*Printed Circuit Board*
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
SD/MMC	*Secure Digital / Multi Media Card*

SPI	*Serial Peripheral Interface*
TCP	*Transmission Control Protocol*
TTL	*Transistor Transistor Lolgic*
UART	*Universal Asynchronous Receiver Transmitter*
USART	*Universal Synchronous Asynchronous Receiver Transmitter*
USB	*Universal Serial Bus*
WAN	*Wide Area Network*

1. INTRODUÇÃO

1.1. MOTIVAÇÃO

No âmbito deste sector de componentes para a indústria automóvel surge o grupo industrial *Preh*, composto por um conjunto de empresas espalhadas pelo mundo. No contexto actual a sua produção contempla ECU's (*Electronic Control Unit*), controlos da condução (*driver controls*), painéis de controlo de climatização, sistemas sensoriais, etc. De toda esta produção vincadamente electrónica, a *Preh Portugal, Lda*, tem a seu encargo a produção de painéis de controlo de climatização, ocupando estes cerca de 80% da sua produção.

Estando os fornecedores da indústria automóvel sujeitos a forte empenho no cumprimento dos seus compromissos, factores como qualidade e segurança dos componentes, produção em massa, tempos de entrega baixos, custos reduzidos, entre outros, são preponderantes para o seu sucesso. De forma a cumprir os requisitos de qualidade, mais concretamente de funcionalidade, a *Preh* implementou o teste de teclas aos painéis de controlo de climatização. Assim, todos os painéis produzidos pelas *Preh* são sujeitos ao teste de teclas, o qual é feito de forma exaustiva.

A análise de uma tecla obedece a duas condicionantes: uma de carácter físico e outra de carácter eléctrico. Na vertente física é estabelecida uma relação entre a força empregue na superfície da tecla e o deslocamento da mesma, ao longo do seu curso. A vertente eléctrica procura um compromisso entre o contacto eléctrico (fecho e abertura) e o curso da tecla.

Actualmente o teste é executado numa estação de teste, composta por vários postos de operação. Cada estação possui um computador incumbido de efectuar o controlo sobre os respectivos postos de operação. Ao longo dos tempos, com o acréscimo de funcionalidades, controlos e processos em cada estação, o computador ficou cada vez mais absorvido ao nível dos seus recursos. Com solicitações de processamento a necessitarem cada vez mais

de recursos, os computadores tornam-se desadequados para o uso a que se destinam. Uma forma de aliviar esta sobrecarga na execução de tarefas por parte do computador, pode ser conseguida delegando tarefas a outros sistemas de processamento. Assim, o projecto que serviu de suporte a esta dissertação está direccionado para a substituição da actual tecnologia, utilizada nas estações de teste das teclas dos painéis, por uma mais recente recorrendo a sistemas embebidos. Com esta mudança pretende-se superar os condicionalismos no processamento até agora existentes, através da construção de uma estação controlada com base num sistema embebido.

Em resumo, deve ser encontrada uma alternativa ao actual sistema de teste das teclas, de modo a que o computador deixe de estar encarregue da execução do teste. Desta forma, outras tarefas inerentes ao sistema global de teste e, da linha de produção dos painéis de controlo de climatização, podem ser devidamente suportadas pelo computador, em virtude da partilha de recursos ter diminuído.

1.2. OBJECTIVOS

O projecto descrito nesta dissertação tem como objectivo desenvolver um actuador linear para teste de teclas em painéis de climatização. Em termos concretos pretende-se pegar na estrutura já existente e substituir o sistema de controlo actualmente em uso, atribuído a um computador, por um sistema embebido. Ao ser introduzido um sistema embebido, com mais valências ao nível do *hardware*, pretende-se melhorar a capacidade de resposta às exigências produtivas nas actuais linhas de produção. A inclusão deste sistema embebido, nas linhas de produção, deve caracterizar-se numa melhoria tecnológica, sendo um elemento que incorpore as necessidades de processamento do algoritmo de teste, comunicacionais (protocolo RS-232, Ethernet e CAN) e conversão A/D. Este elemento deve ter a capacidade de poder actuar simultaneamente em, pelo menos quatro teclas, ou seja, poderão encontrar-se em funcionamento quatro actuadores lineares em simultâneo.

Subjacente a esta temática deverá estar enquadrada uma vertente económica. A mudança tecnológica apenas terá interesse se economicamente for menos dispendiosa que o actual sistema, ou se originar um rendimento produtivo superior, permitindo recuperar o investimento mais tarde.

1.3. ESTRUTURA DA DISSERTAÇÃO

Esta dissertação encontra-se estruturada ao longo de oito capítulos, sendo que, os seis primeiros capítulos são aqueles que apresentam relevância na estrutura descritiva desta dissertação.

Este capítulo introdutório faz alusão ao enquadramento do projecto, tendo em conta a problemática existente, assim como a definição dos objectivos.

O segundo capítulo apresenta a envolvente da indústria dos componentes para automóveis. Uma abordagem ao “estado da arte” sectorial, com uma incidência ao nível dos testes realizados neste sector.

No capítulo três é abordada a evolução do processo do teste até ao modelo actual, evidenciando as suas carências. Posteriormente é apresentada a solução a implementar, com destaque para os critérios opcionais bem como as alterações estruturais.

O capítulo quatro descreve os elementos presentes neste projecto, fazendo o seu enquadramento nas vertentes do *hardware* e *software*. Também faz uma alusão teórica às comunicações de suporte.

No capítulo cinco são descritas as etapas e os moldes nos quais este projecto foi desenvolvido e aplicado. Ao terminar este capítulo é efectuado um balanço do projecto tendo em conta os resultados obtidos.

A finalizar, no capítulo seis, são apresentadas as conclusões e propostos desenvolvimentos futuros, tendo em conta os resultados obtidos e melhorias ou novas soluções a implementar.

2. INDÚSTRIA DE COMPONENTES PARA AUTOMÓVEIS

2.1.CARACTERIZAÇÃO SECTORIAL

A indústria automóvel é uma das mais importantes actividades industriais no mundo, sendo ponto de confluência de vários sectores industriais [Inteli05]. Este sector industrial tem por base grandes grupos empresariais que abarcam subsectores industriais diversificados, sendo estes parceiros ou fornecedores de componentes ou acessórios. Hoje em dia, a produção de um veículo automóvel é composta por inúmeros componentes e acessórios, desde a mais simples até à mais complexa estrutura, são originários da indústria transformadora, nomeadamente a metalomecânica, têxtil, borracha, plástica, electrónica, eléctrica, vidro, etc.

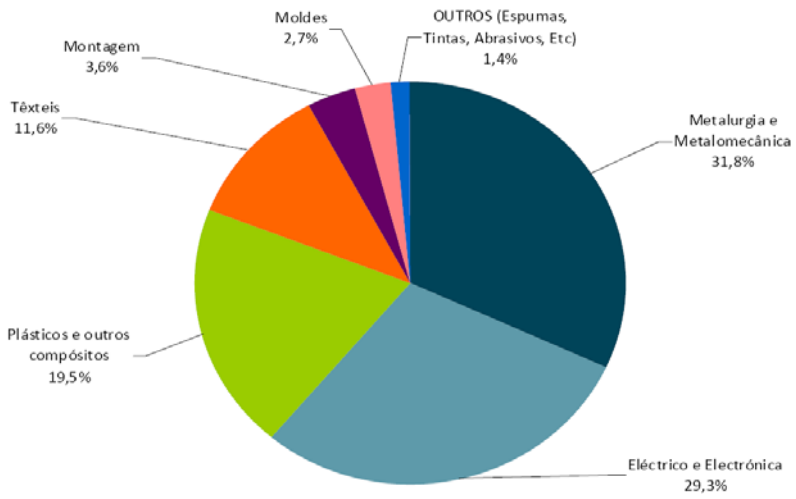


Figura 1 – Subsectores de actividade na indústria automóvel nacional [Afia11].

Os subsectores representativos da indústria nacional que fazem parte integrante da indústria automóvel encontram-se ilustrados na Figura 1. Dentro destes subsectores surge o sector eléctrico e electrónico, alvo de intervenção do projecto apresentado nesta dissertação.

Com esta abrangência de subsectores torna-se necessário que cada fornecedor integre no seu sistema produtivo a testabilidade dos produtos, sendo deste modo possível à indústria automóvel produzir em larga escala, uma vez que o componente ou acessório é fornecido tendo em conta as devidas especificações. Um outro factor de testabilidade advém das imposições regulamentares, normativas, legislativas e, também devido à complexidade do automóvel, proporcionando que um veículo automóvel, nos tempos correntes, seja mais testado do que nunca [Hurden05].

Um veículo automóvel é constituído por inúmeros componentes e acessórios e, como tal, o número de fornecedores é considerável. Uma análise à Figura 2 vislumbra a evolução do número de fornecedores, em função do custo do veículo, ao longo dos tempos. Nesta situação é fácil de depreender o caos que seria no caso de os fornecedores não serem obrigados a cumprir requisitos de qualidade rigorosos, pois dificilmente seria construído um carro. O acumular de tolerâncias fora das especificações, num conjunto de centenas de peças, teria um efeito desastroso na sua montagem. Assim, garantindo elevados padrões de qualidade, através de especificações rígidas aos produtos fornecidos à indústria automóvel, torna-se viável garantir um produto final de qualidade, isto é, um automóvel de qualidade.

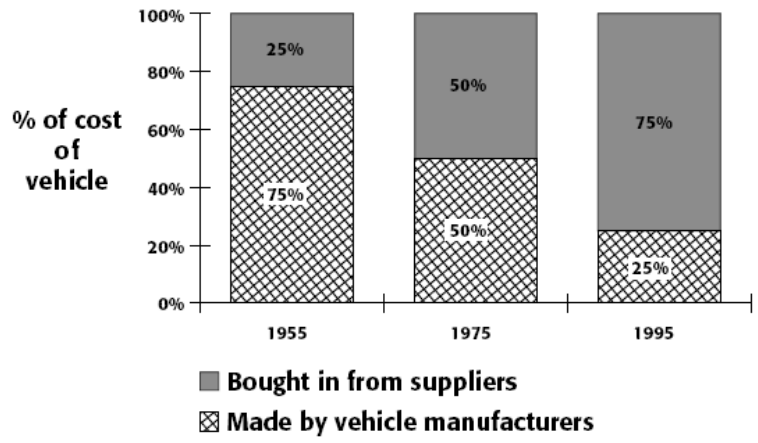


Figura 2 – Percentagem da produção própria e de fornecedores [Maxton05].

Ao exigir os testes nos fornecedores, a indústria automóvel consegue garantir a recepção de componentes conformes, elevando os padrões de qualidade dos seus fornecedores e, simultaneamente, elimina os encargos económicos dos

testes. Sendo um mercado apetecível, os fornecedores desta indústria, primam por manter elevados padrões de qualidade, através da garantia de conformidade do produto entregue ao cliente. Para além disto, uma empresa fornecedora alcança uma elevada reputação no meio industrial. Aliado a este factor, a indústria automóvel, consegue reduzir stocks na sua empresa, uma vez que o produto entregue tem garantia de conformidade do seu fornecedor.

Por outro lado, a feroz competição entre fornecedores desta indústria leva a cortes nos custos de produção, sendo a área dos testes uma delas. Assim, há necessidade de racionalizar os recursos e optimizar tempos de teste, tornando o custo do teste menos significativo.

2.2.IMPORTÂNCIA DO TESTE

Os testes na indústria automóvel desenrolam-se em diferentes fases do ciclo produtivo e abrangem áreas muito díspares, sendo algo tão difícil de os descrever, como saber o número exacto de componentes que compõem o automóvel. Fazendo parte integrante do projecto de engenharia, desenvolvimento e pesquisa, o teste fornece dados importantes sobre as características, propriedades, funcionalidades e especificidades de um determinado material, componente ou conjunto [Gedney07].

Como referido, a necessidade do teste advém logo desde a fase inicial, ou seja, do desenvolvimento do protótipo. Nesta fase os testes são focados nos novos produtos, nos novos materiais, nas ferramentas (novas ou alteradas) e em novos fornecedores. Estes últimos devem ter uma participação activa nos testes nomeadamente na definição de características críticas relativamente à funcionalidade, durabilidade, condições ambientais, simulação, segurança, entre outras. Também devem definir em consonância com o fabricante o tipo de teste, a sua realização, amostragem, a obtenção de leituras, etc. Posteriormente existe a necessidade de ensaios na pista ou mesmo em estrada, ou seja, testes reais evitando deste modo apenas os testes laboratoriais ou simulações. No entanto, o teste não fica por aqui, existindo a necessidade de um retorno por parte do cliente final ou do serviço pós venda,

sendo este o elo de ligação com as fases precedentes. Esta informação também é importantíssima, permitindo detectar anomalias a serem corrigidas ou melhorar determinados aspectos menos conseguidos e, até então não detectados.

O veículo automóvel é um produto final, podendo ser encarado como um sistema global construído à base de agrupamentos sucessivos. Começando pelos componentes e acessórios, criando subconjuntos e conjuntos, definindo funções específicas, caracterizando áreas funcionais, passando por subsistemas até ao sistema global, é definida a estruturação do veículo [Maxton05]. Uma abordagem a esta estruturação e, enquadrada neste projecto, é exemplificada através dos painéis de climatização. Este subconjunto possui a placa electrónica e os seus componentes, membrana, teclas e o corpo do painel, sendo enquadrado numa área de intervenção de controlo, com a especificação dirigida à climatização [Maxton05]. Deste exemplo é perceptível o número elevado de testes necessários ao longo das fases construtivas do veículo, desde o elemento mais simples até ao mais complexo, para que no final o sistema global possa corresponder aos objectivos traçados.

Ao ser implementado um teste a um componente ou acessório, é condição fundamental garantir que as suas propriedades ou características se encontram em conformidade com determinadas especificações provenientes de legislação, normas ou por parte dos clientes (especificações particulares). Os testes podem ser classificados por diversos factores, tendo como exemplo: as propriedades dos materiais (as mecânicas, químicas e eléctricas); as condições ambientais (térmicas, de humidade e exposição solar); em destrutivos ou não destrutivos; em dinâmicos ou estáticos.

Ao longo dos tempos, a evolução do automóvel foi acompanhada por um incremento das suas características eléctricas e electrónicas, tornando-se neste momento um elemento preponderante no seu funcionamento e desempenho. A electrónica encontra-se hoje em dia disseminada ao longo de todo o automóvel não sendo este concebido sem possuir uma boa parte integrante de sistemas electrónicos, desde as ECU's, *airbags*, sistemas de

travagem e arranque, ignição, injecção de combustível, transmissão, controlos de climatização, etc. [Auto07]. Como consequência passou a ser um produto cada vez mais complexo de projectar e desenvolver, em virtude das inúmeras funções incorporadas. Um veículo automóvel possui actualmente um elevado número de pequenos dispositivos eléctricos/electrónicos, os quais têm de funcionar como um conjunto perfeito e, tanto quanto possível de forma confiável. Uma das bases para este pleno funcionamento assenta na estrutura comunicativa entre todos os dispositivos [Baid10] [Maxton05].

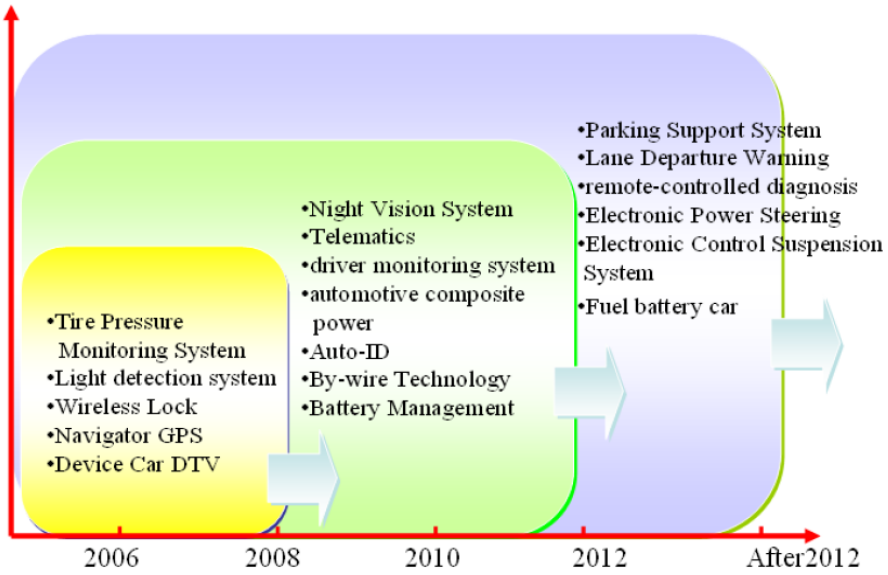


Figura 3 – Desenvolvimento tecnológico da electrónica automóvel [Auto07].

Assim, no mundo actual um automóvel está a sofrer uma transformação, passando de um sistema predominantemente baseado na engenharia mecânica para um sistema mecânico controlado pela electrónica e sistemas eléctricos [Maxton05].

Das inovações tecnológicas ao longo dos tempos, apresentadas na Figura 3, sobressai o nível de aplicações de *software* face ao *hardware* existente, denotando-se assim uma incursão mais acentuada no *software* em relação ao *hardware*. Este factor leva a que o valor dos sistemas de controlo electrónico esteja a aumentar rapidamente no custo total do veículo, sendo que em 1990 o valor era de 10%, em 2000 de 22% e em 2010 de 40% [Maxton05]. O panorama sobre as áreas onde incide maior expansão pode ser observado na Figura 4.

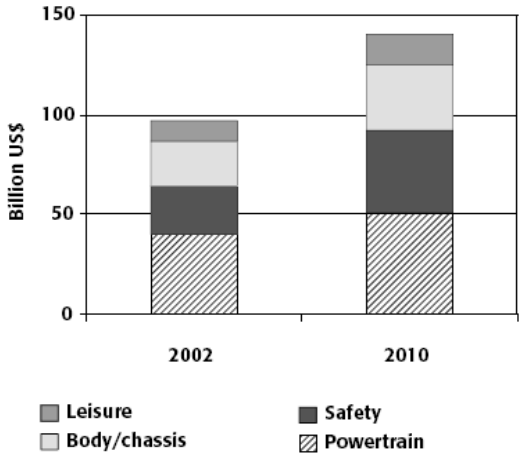


Figura 4 – Áreas de expansão da electrónica automóvel [Maxton05].

As necessidades dos testes electrónicos têm vindo a intensificar-se devido à implementação em larga escala, nos veículos modernos, de sistemas electrónicos. Mas a implementação de mais testes e medidas obriga a um aumento de custos directos e, indirectamente passa a demorar mais tempo nas linhas de produção, levando a um atraso na sua comercialização. Consequentemente, este acumular de testes com o dispêndio de mais tempo obriga a encontrar soluções alternativas, passando estas pela implementação de novas abordagens, estruturas ou recursos tecnológicos.

As soluções de teste e medida têm de ser versáteis, adaptáveis e apresentarem uma excelente compatibilidade, de modo a corresponderem à extrema variedade de requisitos na indústria automóvel, sem que afectem a integridade dos métodos de teste [Baid10]. Os veículos actuais possuem um conjunto de ECU's que podem atingir as 50 unidades, controlando diferentes subsistemas do automóvel. Estas unidades encontram-se interligadas através de barramentos de dados que permitem abranger todas as comunicações existentes no veículo [Johanson11].

O tipo de componentes electrónicos utilizados nos veículos automóveis encontra-se, por vezes, sujeitos a condições severas, tais como elevadas amplitudes térmicas, vibrações e impactos, interferências electromagnéticas. Como tal, estes componentes têm de cumprir com requisitos particularmente rigorosos de modo a garantir a confiabilidade operacional e de segurança. Para

além de um desenvolvimento meticuloso e de testabilidade do circuito, é necessário garantir a mesma qualidade no decurso da produção. A procura de defeitos não é suficiente para que os circuitos ou componentes não possuam problemas. Uma combinação de diferentes procedimentos nos testes é necessária para corresponder aos elevados níveis impostos pelas normas.

2.3. TESTES EM PAINÉIS DE CONTROLO

Um veículo automóvel, nos dias de hoje, possui um leque alargado de comandos para activar/controlar as diversas opções disponibilizadas pelo construtor. Exemplos destes comandos são encontrados em painéis para o controlo da climatização, auto-rádios, luzes, espelhos, etc. O modo em como estes comandos são actuados, isto é, a forma em que eles se encontram disponibilizados para o automobilista/utilizador pode ser variada, no entanto, o mais usual é surgirem por intermédio de uma tecla, interruptor, botão rotativo ou mais recentemente por *display* táctil. De modo a assegurar elevados padrões de qualidade, estes painéis são sujeitos a diversos testes, os quais podem ocorrer de forma exaustiva a todas as peças (a 100%), ou por amostragem onde os critérios podem ser diversos (inicio/fim da produção, por lote, num determinado número de peças).

Sendo a *Preh*, uma empresa vocacionada principalmente para a produção de painéis de climatização para a indústria automóvel e, tendo em mente a constante evolução tecnológica, bem assim como uma estratégia de melhoria continua, promove entre os seus colaboradores a necessidade de uma constante procura pela inovação de processos mais económicos e ao mesmo tempo mais fiáveis. Em consonância com estes pressupostos, o departamento de engenharia encontra-se dedicado ao desenvolvimento e aplicação de soluções na área da automação para as linhas de produção. O sistema de teste de teclas dos painéis de climatização é mais uma das áreas intervenientes deste departamento e em foco nesta dissertação.

2.3.1. TIPOLOGIA DOS TESTES

Aos painéis de controlo de climatização são efectuados diversos testes para permitirem a conformidade do produto final, com as características exigidas pelo cliente durante o seu desenvolvimento. Deste modo é aplicado um conjunto de testes para verificar as especificações, nomeadamente:

- Dimensional – onde são analisadas as dimensões do painel (absolutas e relativas). Dentro desta gama de teste, também é efectuado o teste à espessura da tinta;
- Grafismo – verificando a simbologia empregue no painel;
- Visual – analisando a iluminação do painel em ambiente nocturno;
- Comunicacional – designadamente as comunicações com os outros sistemas/periféricos;
- Electrónico – analisando o posicionamento de componentes electrónicos, injectando sinais e efectuando as suas leituras, aceitação das tolerâncias de alimentação;
- Desgaste (Vibração, calor, etc.) – permitindo uma análise do seu comportamento (uso) ao longo do tempo;
- Mecânicas/Eléctricas das teclas – permitindo desta forma analisar o painel no seu ambiente real, ou seja, se determinada tecla cumpre a sua função.

Os testes referenciados anteriormente são executados nos painéis de controlo, no entanto, no âmbito deste projecto apenas os testes de teclas serão alvo de análise.

2.4. TESTE DE TECLAS

A designação tecla é ampla e genérica, podendo ser interpretada como um interruptor que, se pressionado e fechado o seu contacto eléctrico, executa uma determinada função, à qual se encontra associado. O teste de teclas aqui descrito pretende verificar o funcionalismo mecânico (força e deslocamento) e eléctrico (contacto) das teclas de um dado painel, o qual será alvo aplicacional deste projecto. Consoante o tipo de tecla (membrana), o seu comportamento

em função da relação força/curso estabelece uma curva, a qual pode ser representada graficamente. Esta curva deve estar compreendida entre os padrões exigidos pela norma ou, neste caso, pelos clientes.

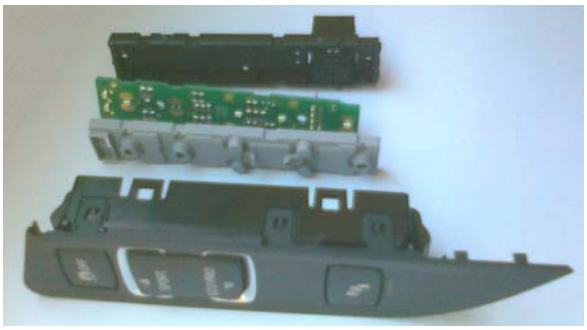


Figura 5 – Exemplo de um painel de controlo em produção na *Preh*.

Os clientes da *Preh* determinam o perfil pretendido para a tecla (força e curso), sempre em consonância com esta de modo a que a peça seja executável em conformidade com as especificações. A Figura 5 apresenta um painel de controlo simples com todos os seus elementos constituintes.

2.4.1. ASPECTOS CONSTRUTIVOS

A temática das teclas insere-se num campo diversificado de aspectos construtivos e aplicativos. No caso do projecto em causa a tipologia construtiva da tecla (ou teclado) tem por base uma membrana (reconhecida como *membrane switch*). Este é o elemento preponderante na caracterização comportamental da tecla.

Inicialmente as teclas tinham uma constituição de cariz mecânico mas com a necessidade de aplicação em produtos de uso corrente (máquinas de calcular, pequenos electrodomésticos, etc.) foi-se desenvolvendo e aplicando a tecla de membrana [Nagurka05]. Esta implantação da tecla de membrana advém das suas características que permitem desenvolver facilmente um teclado à medida das necessidades de um cliente, tais como [GDSI11] [Knitter11]:

- O seu custo é menos oneroso que os teclados com vertente mecânica;

- Permitem uma utilização em diferentes ambientes, designadamente ambientes adversos (resistentes a poeiras, humidades, salpicos de água, etc.);
- É fácil e rápida a sua aplicação;
- Utilização facilitada para o utilizador, devido à forma intuitiva na apresentação das teclas;
- Aplicável em situações de pequenas dimensões, tornando-se facilmente acessíveis.

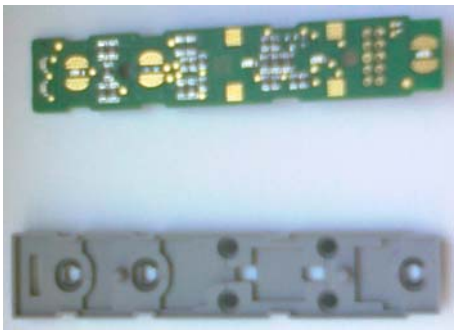


Figura 6 – Detalhe da membrana e placa do painel da Figura 5.

Na Figura 6 pode ser visualizada a membrana na parte inferior e a PCB na parte superior, utilizada num dos painéis em produção na *Preh*. Este tipo de tecla ou teclado pode divergir um pouco em função das aplicações e do fabricante, no entanto, todas elas têm como constituição básica duas camadas. A partir desta base podem surgir outras camadas, levando a que as principais sejam as seguintes:

- Gráfica – É uma fina camada, a qual pode estar incorporada na membrana, que consiste na simbologia representativa das teclas;
- Membrana – uma fina e flexível camada, torna-se imprescindível no conjunto da tecla. É o suporte de pelo menos um dos pólos do contacto;
- Táctil – Esta camada permite definir uma tecla do tipo plana ou saliente. Caso não seja adicionado nenhum elemento a tecla é plana não evidenciando a sensação de contacto. Caso seja implantado um elemento saliente (incorporação de metal ou poliéster) permite a obtenção de uma resposta táctil quando a tecla é pressionada;

- Estática – Suporta pelo menos um dos pólos do contacto. Conjuntamente com a membrana definem o contacto eléctrico;
- Rígida – Caso a camada estática seja flexível e não seja aplicável sobre uma estrutura rígida, então existe a necessidade de aplicar esta camada como suporte à camada estática.

No caso em concreto deste projecto e, tal como apresentado na Figura 5, a constituição das teclas integra um processo simples. Este processo corresponde ao padrão que englobam os seguintes elementos:

- PCB – *Printed Circuit Board* (Placa de circuito impresso);
- Membrana;
- Tecla;
- Corpo da peça (por exemplo: painel climatização).

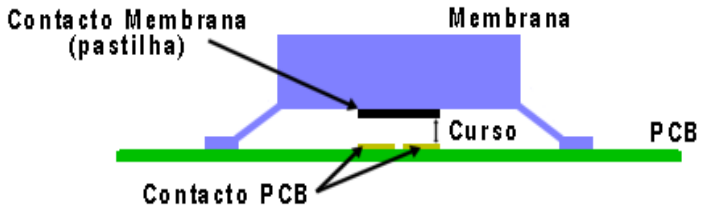


Figura 7 – Exemplo de uma estrutura interna da tecla.

A Figura 7 apresenta um corte em perfil da estrutura de uma tecla, destacando-se o seu curso e contactos. Estas teclas apresentam uma membrana (derivados de borracha/silicone) onde internamente possuem uma pastilha de grafite (carbono). Quando pressionada esta membrana e atingido o final do curso, vai concretizar o fecho do contacto aberto existente na PCB através da pastilha. As paredes laterais são o elemento primordial no tipo de actuação que a membrana vai ter, pois dependendo da altura (em relação à PCB), da sua dureza, elasticidade e do seu ângulo, são elas que caracterizam as curvas.

2.4.2. RESPOSTA DA TECLA

Em função do tipo de tecla vai ser estabelecida uma acção característica, a qual prima pela resposta da tecla. A resposta de uma tecla pode ser de dois tipos: táctil ou não táctil. Uma resposta táctil é definida como um colapso

repentino da membrana antes de ter sido efectuado o contacto eléctrico [ASTM01]. Como usualmente o utilizador necessita de obter uma percepção da acção efectuada sobre a tecla, esta pode ser obtida de forma visual, sonora ou táctil. Assim quando é necessário obter a percepção de que a tecla foi pressionada, sem a existência de meios alternativos, indicativos da activação do contacto eléctrico por parte do seu utilizador, recorre-se a uma tecla táctil. As teclas não tácteis recorrem a meios alternativos para indicarem a sua activação, tal como a utilização de um elemento visual (LED), um elemento sonoro (*beep*) ou ambos (inicio de funcionamento de um mecanismo, etc.).

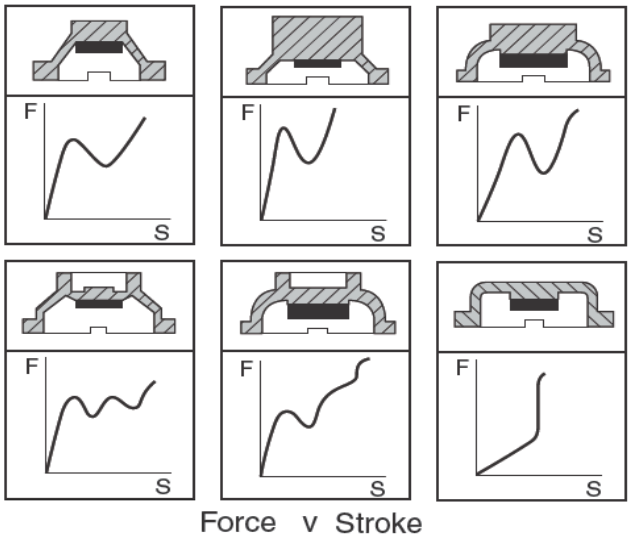


Figura 8 – Formatos e curvas característica de diferentes membranas [Knitter11].

Os diferentes formatos das membranas permitem a obtenção de diferentes características na resposta de teclas, as quais podem ter aplicações específicas. A Figura 8 reporta diversos tipos de membranas conjuntamente com a sua curva característica, onde pode ser analisada a força em função do curso [Knitter11].

2.4.3. ESPECIFICIDADES DO TESTE

O teste de tecla, em termos gerais, tem como objectivo estabelecer uma relação entre a força aplicada na sua face e o respectivo curso ocorrido na tecla (quer na fase descendente, quer na fase ascendente). A conjugação destas duas grandezas (força e deslocamento) permite construir uma curva

(gráfico), a qual deve corresponder a determinados requisitos, em consonância com o tipo de tecla (ver exemplos da Figura 8). Se a estas duas grandezas mecânicas, que caracterizam a curva, for acrescido da ocorrência de um contacto eléctrico em função do efeito força/deslocamento, teremos o resultado global do teste da tecla.

Em termos gerais não se encontram definidos parâmetros em relação ao controlo da atmosfera envolvente, isto é, temperatura, pressão, humidade ou outros parâmetros, para a realização dos testes [ASTM10]. No entanto, é conveniente que estes parâmetros sejam enquadrados em ambientes tipificados e não com temperaturas extremas, elevadas humidades, etc. Tendo em conta esta necessidade a *Preh* proporciona condições no ambiente fabril que podem ser consideradas normais e estáveis, isto é, não são condições adversas mantendo-se uma constância ao longo dos tempos. Assim, com o objectivo de manter estabilidade nas condições do ambiente fabril a *Preh* possui um sistema de climatização.

2.4.3.1. CURVA CARACTERÍSTICA

Conforme visto anteriormente, a membrana vai definir a forma de actuação da tecla. Esta forma de actuação encontra-se definida por normas da *American Society for Testing and Materials* (ASTM), as quais são enquadradas nas especificações dos clientes. A curva característica das teclas a serem utilizadas neste projecto corresponde a uma condição dada pelo cliente da *Preh*, sendo que esta deve cumprir tais requisitos.

Aquando da aplicação do teste é necessário reconhecer o tipo de curva que é expectável que surja de modo a ser enquadrada a sua análise. Desta forma a análise às teclas dos painéis em estudo neste projecto reporta à curva apresentada na Figura 9.

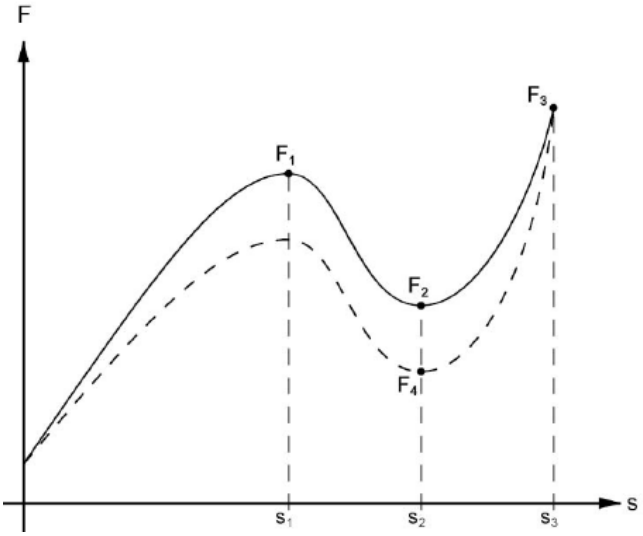


Figura 9 – Curva característica para o painel em teste.

O gráfico da Figura 9 caracteriza um determinado tipo de membrana, sendo composto por duas curvas. A primeira, a traço contínuo corresponde à fase descendente da tecla, ou seja, quando está a ser exercida uma força de modo a ultrapassar a inércia da tecla [ASTM10]. A tracejada encontra-se a curva correspondente à fase ascendente da tecla. Nesta situação a força anteriormente exercida está a ser removida ou, por outro lado, é a força exercida pela tecla para voltar à posição de repouso.

2.4.3.2. PONTOS EM ANÁLISE

Os pontos representados na Figura 9 são fundamentais para a definição gráfica da característica da tecla. Através do conhecimento quantitativo destes pontos e de combinações decorrentes do seu relacionamento é possível inferir-se sobre o estado de funcionamento da tecla (bom ou mau estado). O significado destes pontos é descrito seguidamente:

- F1 – Força máxima exercida na tecla até esta ceder, ou seja, quando acontece o "*click*" da tecla. Esta é a força de actuação correspondendo à máxima força exercida imediatamente antes ou no acontecimento do fecho do contacto (estágio de comutação), [ASTM01]. A esta força corresponde o deslocamento (S1) da tecla;

- F2 – É a força mínima exercida sobre a tecla e é atingida após o rápido avanço proporcionado pela passagem do ponto F1. Esta força encontra-se associada ao posicionamento S2;
- F3 – A força máxima aplicada na execução do teste é atingida neste ponto. Usualmente este valor encontra-se indexado ao valor obtido pela força F1. O valor S3 corresponde ao deslocamento e é definido como sendo o ponto de fixação mecânica – batente (a partir daqui podem ocorrer danos estruturais na tecla);
- F4 – Corresponde ao valor mínimo da força quando é desencadeado o movimento de retorno, levando a tecla à sua posição de repouso. Deve encontrar-se no alinhamento da força F2, sendo o deslocamento de ambas coincidentes (S2);

Snap

Uma relação directamente ligada à sensibilidade táctil, proporcionada pela tecla, é encontrada através da expressão *Snap*. Esta expressão relaciona duas forças da seguinte forma [ASTM01]:

$$Snap = \frac{F1 - F2}{F1} \times 100$$

Com esta expressão é possível inferir-se que, quanto mais elevado for o valor *snap*, maior é a sensibilidade táctil da tecla mas por outro lado, menor é o ciclo de vida. Se o valor de *snap* for baixo, o resultado é o oposto ao anterior.

2.4.3.3. ESPECIFICAÇÕES PARTICULARES

A fase de desenvolvimento de um painel de controlo de climatização é uma conjugação de esforços entre o cliente e o fornecedor (neste caso a *Preh*). O cliente inicialmente estabelece as características e especificações que pretende ver implementadas no seu produto. Posteriormente, na fase de desenvolvimento, são analisadas as especificações e negociadas com o cliente, pois nem sempre estas especificações são viáveis tecnologicamente ou economicamente. Usualmente é obtido um compromisso entre o cliente e o fornecedor de modo a garantir elevados padrões de qualidade e se possível

cumprir para além do necessário os requisitos normativos. No caso particular em análise, as especificações que a *Preh* tem de garantir ao cliente, são apresentadas em seguida:

- F1 – Obtido por leitura directa;
- F2 – Corresponde a 65% de F1, com uma tolerância de -5% a +10%;
- F3 – Maior que F1 e acrescido de 1 unidade;
- F4 – Maior que a unidade;
- S1 – Deve corresponder a 45% do valor de S3, com uma tolerância de -5% a +10%;
- S2 – Corresponde ao valor 1,7xS1, com uma tolerância de -10% a +10%;
- S3 – É obtido por leitura directa;
- *Snap* – Cuja relação é dada por 1-F2/F1, correspondente a um valor compreendido entre os 30% e os 45%.
- S1 *ratio* – Define-se como S1/S3, e o valor deve estar compreendido entre os 35% e os 50%.

Relativamente ao estado em que se encontra contacto eléctrico (aberto ou fechado) em função do curso tecla, este tem de cumprir determinadas condições, designadamente:

- Até ao deslocamento atingir S1 o contacto não pode fechar;
- Após o primeiro pico máximo de força, correspondendo ao deslocamento S1, o contacto deve fechar. O curso percorrido pela tecla é relativizado para a ocorrência do fecho.
- O fecho do contacto tem de estar estabelecido num ponto referenciado através da força, cujo valor corresponde a F1+0,5 (ponto em análise). Este ponto fica um pouco abaixo do valor máximo da força (F4);
- Na fase ascendente da tecla (retorno) tem de ocorrer a abertura do contacto eléctrico, ou seja, vão ser repostas as condições iniciais. A abertura do contacto, por norma, acontece antes de ser atingido o valor S2.

No caso de uma tecla sujeita ao teste, a conformidade da mesma, fica dependente da observância das condições anteriormente descritas.

Como iremos ver mais adiante, estas características correspondem a parametrizações assumidas na fase inicial do teste, as quais foram obtidas na fase de desenvolvimento dos testes para as teclas e ao longo de diversos ensaios.

3. PROCESSO DO TESTE

3.1. PROCESSO ACTUAL

O processo de teste actualmente empregue nas linhas de produção, é um sistema desenvolvido pela Preh, o qual necessita de melhoria contínua. Com isto, é possível retirar partido dos avanços tecnológicos ao longo dos tempos.

3.1.1. INTRODUÇÃO

Inicialmente o processo de teste recorria a sistemas pneumáticos, onde um cilindro era controlado por uma válvula proporcional. Uma determinada pressão colocada no cilindro pela válvula ia ser proporcional à força a aplicar na tecla. Esta força era quantificável em tensão eléctrica por intermédio de uma célula de carga. Nesta altura não era possível efectuar um controlo sobre o deslocamento, apenas era possível controlar a força exercida.

Mais tarde, foram desenvolvidos dispositivos que permitiam o controlo do deslocamento. Estes dispositivos eram constituídos por motores passo a passo, os quais agregavam *encoders*[1], sendo possível desta forma quantificar o deslocamento que a tecla tinha efectuado. Devido às suas dimensões, os motores passo a passo, acarretavam inúmeras dificuldades na sua implementação ao longo das estações de teste.

A situação de falta de espaço físico, devido às dimensões dos motores passo a passo, foi solucionada com a implementação de outro dispositivo motor. Este dispositivo motor é designado por *SMAC*[2], o qual possui dimensões físicas substancialmente menores, permitindo a integração em espaços mais reduzidos.

[1] Dispositivo electromecânico que proporciona impulsos eléctricos associados a deslocamentos angulares ou lineares. Desta forma é possível quantificar o deslocamento num valor digital pelo número de impulsos ocorridos.

[2] Designação oriunda da empresa fabricante: *SMAC Inc*.

3.1.2. SISTEMA ACTUAL

Actualmente um sistema de teste de teclas é, resumidamente, constituído conforme o apresentado na Figura 10. Possui um bloco designado por *SMAC*, sendo em termos gerais o elemento que efectua o deslocamento da célula de carga. Este bloco encontra-se dividido em duas áreas físicas distintas: uma é o motor e a outra é o controlador.

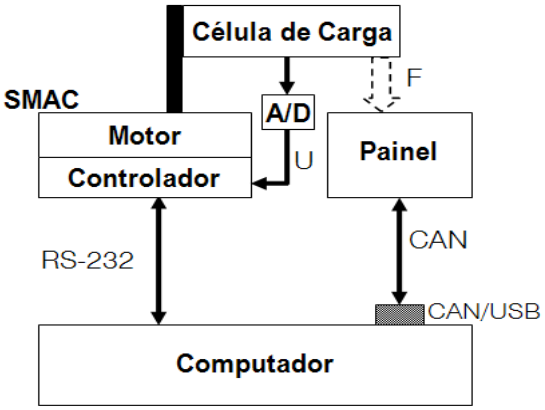


Figura 10 – Blocos constituintes do sistema actual.

O motor *SMAC* possui uma haste num dos seus extremos, proporcionando desta forma um movimento linear. Aproveitando o movimento da haste, uma célula de carga é incorporada na extremidade da mesma com o objectivo medir a força que é exercida na tecla. O mesmo movimento é indicativo do curso percorrido pela tecla, em virtude da relação directa com o deslocamento da haste do motor. A conjugação destes dois elementos (célula de carga e motor) leva à obtenção do conhecimento da força *versus* deslocamento.

O controlador *SMAC* faz a gestão do deslocamento, reconhecendo em qualquer altura o posicionamento a que se encontra a extremidade da haste. Para tal contribui a incorporação no interior do motor *SMAC* de *encoder*. Desta forma, todos os posicionamentos obtidos no decorrer do deslocamento são enviados para o computador.

A célula de carga tem uma dupla funcionalidade: em primeiro lugar obtém o valor da grandeza física – força, que é exercida na tecla; em segundo, converte a grandeza física numa grandeza eléctrica – tensão, facilitando a integração

com o controlador. A tensão eléctrica obtida aos seus terminais é amplificada antes de ser convertida num sinal digital (estágio conversor A/D – Analógico/Digital). A partir deste momento o sinal encontra-se passível de ser interpretado pelo controlador.

Na Figura 10, além dos blocos constituintes, estão apresentadas as ligações entre eles. O computador é o elemento central estabelecendo uma ligação RS-232 com o *SMAC*, para troca de comandos e dados (por exemplo: o inicio do teste, valores obtidos, etc.). Com o painel de controlo, o computador vai criar uma ligação CAN (*Controller Area Network*) mas, como este não possui uma saída deste género, vai ter de recorrer a um adaptador CAN/USB (*Universal Serial Bus*). Esta ligação serve para trocar mensagens com o painel, mais concretamente saber o estado em que determinada tecla se encontra.

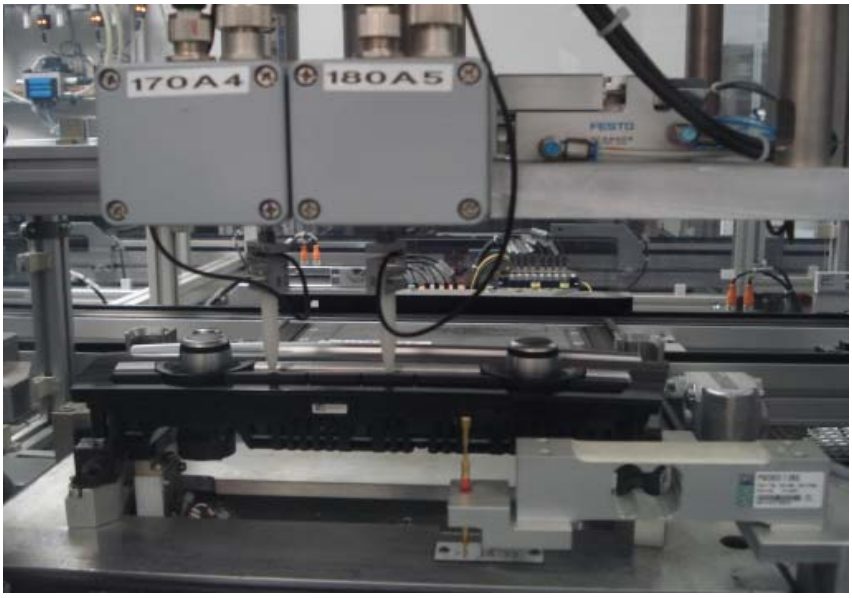


Figura 11 – Estação de teste de teclas.

O sistema de blocos apresentado na Figura 10 encontra-se implementado nas actuais estações de testes, sendo a Figura 11 exemplo dessa aplicação. Uma estação pode ser constituída um conjunto de postos de testes iguais ao da Figura 10, sendo maioritariamente este o caso.

O processo de teste que se desenrola nas linhas de produção actuais é totalmente autónomo, necessitando apenas da colocação e recolha dos

painéis. Ao longo das linhas vão surgindo estações, umas mais simples, outras mais complexas, além de diferentes actividades e funções. Uma estação complexa é associada a ter pelo menos dois postos, considerando que o posto é um local onde é exercida uma determinada tarefa sob o painel. Numa destas estações encontra-se a tarefa inerente ao teste de teclas do painel e, tendo vários postos é possível a execução à totalidade de teclas existentes no painel.

As estações, para além de outros elementos, são compostas por dispositivos de deslocamento linear (*SMAC*'s), as quais têm como denominador comum o controlo exercido por um computador. Este computador, no caso do teste de teclas, envia comandos iniciais e, posteriormente, ao longo do teste recolhe os parâmetros a fim de os processar. Convém salientar que, apenas existe um único computador no controlo de uma estação, a qual pode englobar mais que um *SMAC*, mais que um processo, mais que uma aplicação, etc. No final, o resultado do teste das teclas é sinalizado no monitor do computador por um "ok" (conforme) ou "nok" (não conforme).

Com o objectivo de criar uma base de dados de rastreabilidade, a Preh utiliza um sistema de codificação 2D (*Data Matrix*[3]) das suas peças, componentes ou conjuntos. Assim, um componente ao entrar na linha de produção leva um código associado e, à medida que vai passando pelos estações de trabalho, este código associa o componente à estação de trabalho. Transformações, configurações, resultados de testes vão sendo recolhidos e guardados com base no código atribuído. Deste modo é sempre possível verificar o historial do processo produtivo da peça, reconhecendo a linha de produção, as estações, os operadores intervenientes, resultados de testes, etc.

Numa fase posterior, qualquer anomalia que tenha sido detectada na peça, vai originar uma inspecção por parte de um técnico. Desta forma vai ser constatada, ou não, a anomalia ocorrida na execução do teste e a possibilidade de reparação da mesma. Determinadas anomalias têm um custo superior na

[3] Código bidimensional que consiste na aplicação num quadrado ou rectângulo de pequenos quadrados pretos ou brancos, contextualizando pequenas quantidades de informação.

reparação face à produção de uma nova unidade, não compensando a sua reparação.

3.1.3. CONSTRANGIMENTOS

A execução do teste de tecla acarreta um período temporal com algum significado, face a outras tarefas ou actividades que decorrem na linha de produção. Desta forma um computador que não esteja dedicado ao teste de teclas e, em determinada altura tenha de intervir noutra área ou efectuar testes simultâneos utilizando dois *SMAC*'s, torna-se um pouco lento.

Actualmente encontra-se um computador a gerir uma estação de testes e, como tal, tem diversas tarefas a executar simultaneamente. Estando o computador a funcionar com base no sistema operativo *Windows* (algo "pesado" em termos industriais), as suas performances temporais vão-se degradar à medida que são acrescentadas tarefas e funcionalidades necessárias à execução dos trabalhos das estações ou das linhas. As suas limitações são sobretudo notórias quando, em execução do teste das teclas uma recolha de valores implica uma troca constante de dados entre o computador e o *SMAC*. Esta troca origina a que os recursos do computador ficam largamente absorvidos por este periférico, originando um decréscimo nas performances das aplicações residentes no computador.

Assim, o tempo de processamento proporcionado por um computador partilhado, é um factor inibitório a um aumento da carga de trabalho nas estações/postos de teste. Por diversas vezes o ciclo de execução de tarefas é prolongado, devido à elevada quantidade de dados a manipular e processar, não tendo os computadores capacidade de resposta temporal para tal situação.

Alternativamente surgem os PLC´s (*Programmable Logic Controller*) mas a sua integração nas linhas é dificultada pelo facto de possuírem uma interface relativamente pobre, nomeadamente ao nível de visualização. Além disto, também existe alguma dificuldade de integração na tipologia do barramento CAN que a *Preh* possui.

3.2. PROCESSO A IMPLEMENTAR

Encontrando-se no mercado sistemas com processadores/microcontroladores com elevadas capacidades memória, velocidade de processamento, recursos, etc., é fácil depreender a mais-valia em passar a utilizar um sistema com esta tipologia.

3.2.1. ALTERAÇÃO ESTRUTURAL

A nova implementação pretende que o processo gestão e controlo do teste saia fora do âmbito das actividades do computador, libertando os recursos inerentes ao teste. Para tal, é necessária uma alteração na estrutura de modo a efectuar a inclusão do sistema embebido. A Figura 12 apresenta a solução encontrada, estando os seus elementos representados sob a forma de blocos (apenas se encontra apresentado um sistema *SMAC* para facilidade de interpretação). Analisando os blocos e comparando com o sistema actual (Figura 10), é perceptível a inclusão de novo *hardware*, o sistema embebido. Com este novo elemento, devido às suas características de comunicação, é possível aglomerar no próprio o controlo até quatro *SMAC's*, sendo desde logo uma mais-valia nesta nova estrutura.

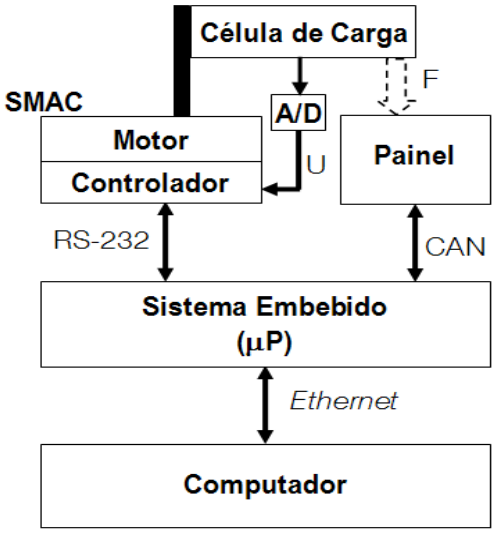


Figura 12 – Blocos constituintes do sistema a implementar.

À alteração física, vai corresponder uma mudança no procedimento de execução do teste. Em termos gerais, o computador envia configurações e comandos para o sistema embebido. Este processa os dados e manda executar o teste, recebendo valores de força, do curso e do contacto eléctrico da tecla. Com estes dados indica o sucesso ou insucesso do teste ao computador. Assim, a inclusão deste sistema como elo de ligação entre o computador e o sistema de teste, vai permitir que o computador disponha livremente do tempo durante o qual decorre o teste.

Resumindo, os elementos necessários para as implementações do novo teste são idênticos ao processo actual, acrescido do sistema embebido, contemplando então:

- *Fez-Cobra* – Sistema embebido;
- *SMAC* – Motor;
- *SMAC* – Controlador;
- Célula de carga;
- Conversor A/D;
- Computador;
- Fontes de alimentação.

3.2.2. SISTEMA EMBEBIDO

A escolha do sistema embebido recaiu sobre a placa *Fez-Cobra*, da *GHI Electronics*. Esta opção teve em consideração alguns requisitos, inicialmente estipulados, sendo eles:

- Custo reduzido;
- Processador com potencial elevado de processamento (ARM ou equivalentes);
- Suporte de comunicação série (RS-232), em número suficiente para ligar quatro *SMAC*'s;
- Suporte de comunicação CAN, permitindo ligar-se ao barramento CAN utilizado nas mensagens dos painéis;
- Ligação Ethernet de modo a assegurar comunicabilidade com o computador;

- Memória que garantisse uma margem de segurança para suportar a aplicação a desenvolver e suportar o armazenamento de alguns dados;
- Linguagem de programação corrente, evitando tempos de aprendizagem elevados e eventualmente custos de formação;
- Opcionalmente possuir conversor A/D, no mínimo de 10 bits, evitando a situação actual que recorre a um conversor A/D externo. Esta opção facilita desenvolvimentos futuros.

Como estes requisitos são satisfeitos pela *Fez-Cobra*, além de outros potencialmente interessantes (como o leitor de cartões SD/MMC – *Secure Digital / Multi Media Card*, para um armazenamento substancial de dados localmente), a escolha recaiu sobre esta placa.

Um outro componente interessante da *Fez-Cobra*, que acabou por não ter sido utilizado, foi o *display* táctil. Inicialmente teria uma aplicabilidade interessante na visualização de mensagem sobre o estado de variáveis ou resultados que, fossem analisados ou processados pela *Fez-Cobra*. Com o decorrer do desenvolvimento da aplicação verificou-se que não existia interesse na emissão de informações para o *display* da *Fez-Cobra*, podendo até ser um inconveniente devido aos recursos utilizados.

Apesar de o custo da *Fez-Cobra* não ser expressivo para as potencialidades que apresenta (valor na ordem dos 120€) e, como nesta fase do desenvolvimento nem tudo é previsível, optou-se pelo conjunto *Fez-Cobra OEM Box (3.5'' Display).* Este conjunto disponibiliza para além da placa *Fez-Cobra*, um *display* táctil, envolto numa caixa protectora (tudo o conjunto num valor na ordem dos 200€). Tendo em conta a necessidade de expansibilidade, foi necessário adquirir a placa de expansão (valor na ordem dos 16€).

Ao adquirir esta placa, também se teve em conta que, se esta fosse sobredimensionada é possível optar por uma outra, da mesma série mas de gama inferior (essencialmente a variação reside ao nível dos periféricos). As placas inferiores são modulares permitindo desta forma acrescentar os

módulos necessários e com um preçário base inferior. Com esta opção é garantido que o custo final não ultrapassa os 125€.

3.2.3. ESTRUTURA FUNCIONAL

A realização do teste pressupõe o estabelecimento de modos de funcionamento, parâmetros a configurar, resultado a obter, etc., ou seja, é necessário definir um conjunto de regras para o correcto funcionamento. Desta forma, os quatro elementos intervenientes têm papéis preponderantes no teste das teclas, sendo eles:

- Computador – Possui uma consola que permite definir modo de funcionamento do teste, configurações de parâmetros, apresentação de resultados, etc. Esta consola foi desenvolvida paralelamente com a aplicação para a *Fez-Cobra*, de modo a cobrir as necessidades de controlo e obtenção de resultados. As funções de saída de dados correspondem ao envio para a *Fez-Cobra* do modo de funcionamento, comandos, parâmetros e ao estabelecimento da ligação Ethernet. No sentido inverso recebe confirmações ou resultados;
- Placa *Fez-Cobra* – É o elo de ligação entre os diferentes elementos. A saída de dados permite enviar para o *SMAC* os valores recebidos da consola, enviar mensagens CAN para o painel e enviar resultados ou confirmações para o computador. A recepção é o conjunto de dados enviados pelos periféricos. Para além da troca de dados, tem de fazer todo o tratamento de dados decorrentes da execução do teste. Dados da força e deslocamento provenientes do *SMAC* e dados do contacto eléctrico provenientes do painel, têm de ser guardados e analisados. A análise destes dados proporciona a obtenção do resultado em termos da conformidade da tecla;
- *SMAC*'s – São os elementos que efectuam o movimento do curso da tecla para a obtenção dos valores da força e deslocamento, os quais são enviados para a *Fez-Cobra*. Podem ser no máximo de quatro *SMAC*'s ligados à *Fez-Cobra*;

- Painel de controlo – Envia diversas mensagens CAN, sendo que uma destas mensagens contém a informação sobre o estado actual das teclas. Por exemplo, se nenhuma tecla está a ser pressionada a mensagem enviada corresponde a um *Data Byte* em que todos os bits se encontram a 0, caso contrário o bit correspondente à tecla premida passa a 1.

3.2.4. MODOS DE OPERAÇÃO

Com a mudança na estrutura do sistema de teste, através da inclusão do sistema embebido, existe a necessidade de adaptar a nova estrutura para se obter a máxima eficiência. Esta adaptação à nova estrutura passa pela incorporação de dois modos de operação: o modo transparente e o modo autónomo. O modo de operação seleccionado mantém-se em vigor até que seja seleccionado o outro modo. Seleccionado um destes modos de operação, todos os *SMAC*'s (na consola um *SMAC* corresponde a um canal) ficam com o mesmo modo de operação.

3.2.4.1. MODO TRANSPARENTE

Com este modo pretende-se estabelecer uma ponte entre a consola disponibilizada no computador e o *SMAC*. Desta forma, os comandos inseridos na consola são passados integralmente para o *SMAC*. A intervenção da *Fez-Cobral,* neste modo de operação, limita-se à "conversão" do tipo de comunicação, isto é, recebe dados da consola via Ethernet e envia os mesmos dados para o *SMAC* via RS-232. Caso o comando tenha resposta, o processo é revertido e a resposta é apresentada na consola. Quando se pretende entrar neste modo de operação, tem de ser seleccionado um canal (corresponde a uma ligação *SMAC* na consola), em virtude da necessidade de estabelecer uma ligação com um *SMAC*. Ao trabalhar com este modo e, como existe uma ligação "directa" entre o computador e um *SMAC*, origina a que todos os outros estejam parados.

Este formato permite que comandos individuais possam ser executados, dando uma maior liberdade nas aplicações do *SMAC*. Uma aplicação possível deste modo é quando existir a necessidade de ajustar os actuais posicionamentos do *SMAC*, para novas posições em virtude de uma alteração dimensional do painel em teste. Enviando um comando ou conjunto de comandos individuais que, permita a obtenção da coordenada, é uma mais-valia neste sistema.

3.2.4.2. MODO AUTÓNOMO

A opção pelo modo autónomo recai quando é necessário realizar um teste à tecla e, por opção este é o modo seleccionado ao ser iniciada a consola. Em termos gerais o modo pode ser caracterizado por duas fases:

- A fase inicial diz respeito às configurações inerentes à execução do teste pretendido, isto é, definir o canal, parâmetros e opções na consola;
- O início do teste corresponde à segunda fase. Aqui é dada a ordem para se iniciar o teste através de botão na consola. A partir deste momento todo o processo é desenrolado pela *Fez-Cobra*, tendo por base todas as configurações da consola. No final do teste é apresentada uma mensagem na consola com o resultado do teste.

Após a conclusão do teste, os valores medidos na execução do mesmo, podem ser observados graficamente e textualmente. Para além desta opção, também é possível eliminar os dados.

Através deste modo, cria-se um processo de execução do teste autonomamente, indo ao encontro com um dos princípios que norteiam este projecto, ou seja, a libertação da execução do teste por parte do computador. Assim, o computador envia o comando para iniciar o teste e no final recebe a mensagem do resultado, não necessitando de disponibilizar mais recursos físicos ou temporais.

3.2.5. VANTAGENS

Através desta nova implementação no *hardware* do teste das teclas, é possível reconhecer as vantagens a usufruir deste sistema. Desde logo duas áreas vão se destacar: a área tecnológica e a económica.

TECNOLÓGICA

Ao nível tecnológico é possível encontrar um conjunto alargado de sistemas embebidos com excelentes características de processamento, com consideráveis capacidades de memória (programa e dados) e de elevada expansibilidade. Esta expansão até pode ser obtida por sistema modular permitindo, praticamente constituir um sistema à medida das necessidades. Sistemas operativos “leves”, ou seja, ocupam pouco espaço e recursos e consequentemente consegue ter tempos de execução bastantes baixos.

Versatilidade ao nível de aplicações, sendo possível numa fase posterior ser desenvolvido para outro tipo de testes (com a vantagem de já existir trabalho feito, sendo apenas uma questão de adaptação), outro tipo de aplicação, interligação com outros dispositivos, etc.

Um outro aspecto, não menos importante, prende-se com o volume ocupado por um destes dispositivos ser muito inferior a um computador, sendo possível a sua aplicação em áreas de reduzidas dimensões.

ECONÓMICA

Existe sempre o factor económico que pode influenciar o desenvolvimento e aplicação de novos sistemas, não sendo o caso ocorrido neste projecto. Ao nível do mercado existem diversos fabricantes e modelos, contudo uma placa que corresponda a estas exigências, é passível de ser obtida por menos 150€. A placa utilizada em desenvolvimento foi sobredimensionada e custa menos de 150€. O mesmo fabricante disponibiliza a placa equivalente à *Fez-Cobra* em

sistema modular e, como tal, é possível efectuar um sistema deste tipo por um valor inferior a 100€.

Tendo em consideração que um computador, dos que estão actualmente em uso nas estações, tem um custo a rondar os 500€, é fácil depreender o benefício económico directo que se obtém com esta solução. Aplicando esta solução numa dezena de estações, ainda mais significativo será o benefício económico. Indirectamente, é conseguido um aumento da produção (diminuição do ciclo de teste), originando uma redução nos custos de produção.

4. RECURSOS EMPREGUES

4.1. SISTEMA DE MOVIMENTAÇÃO

O sistema de movimentação surge da necessidade de colocar em contacto a célula de carga com a superfície do elemento em ensaio/teste, de modo a efectuar a leitura da força versus deslocamento. Para tal, recorreu-se a um motor eléctrico que incorpora um veio linear, permitindo desta forma a obtenção do desejado movimento.

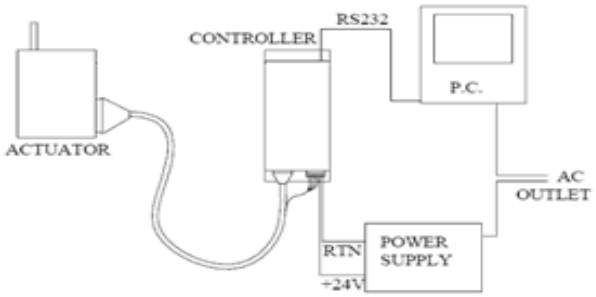


Figura 13 – Sistema genérico de movimentação.

Em termos gerais, pode-se considerar que, a unidade executante do deslocamento é um motor eléctrico. Contudo, este motor por si só, não é capaz de produzir os efeitos desejados para uma aplicação dentro deste âmbito. A Figura 13 representa o sistema genérico e a partir deste, desenvolveu-se um sistema de movimentação especifico para as necessidades do projecto, sendo este constituído por:

- Bloco motorizado – onde se encontra o motor e sistema de leitura posicional;
- Controlador – possui duas funcionalidades: uma coordena e gere todos os dados a receber e/ou a enviar para o motor; outra é ligação com uma unidade de controlo externa (sistema com capacidade de processamento);
- Célula de carga – é o elemento sensor que permite a recolha dos valores de força que estão a ser aplicados no teste da tecla, encontrando-se situada na extremidade da haste do motor;

- Amplificador/Conversor A/D – executa a conversão dos valores analógicos, obtidos pela célula de carga, em valores digitais para serem interpretados pelo controlador. Elemento que interliga a célula de carga ao controlador;
- Computador – é um elemento essencial para o desenvolvimento do código do controlo de deslocamento e leitura da força, a ser enviado para o controlador. Após este código ter sido elaborado os serviços do computador são prescindíveis neste projecto.

Convém referir que, o sistema de movimentação da *SMAC* utilizado neste projecto engloba o controlador e o motor. Como tal, a interligação entre os dois módulos é obtida por um cabo fornecido pelo fabricante.

4.1.1. MOTOR *SMAC*

Tem como principal objectivo dotar o sistema de movimento, isto é, realizar a deslocação de célula de carga, de forma a ser possível efectuar a leitura da força aplicada numa tecla. A opção recaiu sobre um determinado tipo de motor cuja designação mais usual é de actuador de bobina móvel – "*voice coil actuactor*", devido ao seu princípio de funcionamento.

A *SMAC Inc.* é uma empresa fabricante deste tipo de motores, a qual disponibiliza um leque alargado de soluções e modelos, para diferentes tipos de aplicações. Desta forma a opção recaiu sobre o motor LAL35-025, sendo a Figura 14 ilustrativa do modelo usado no decurso deste projecto.

Figura 14 – Motor *SMAC* LAL35-025.

Estes motores têm características que lhes permitem serem definidos como motores de excelência para este tipo de aplicação – teste de teclas. O seu veio é detentor de um baixo valor de massa e com auto-lubrificação, permitindo obter deste motor uma diminuição significativa na fricção/atrito. Daqui resulta a obtenção de um controlo com um baixo nível de contacto, ao nível da força, entre as superfícies. Esta função é patenteada pela *SMAC* como *SoftLand*, permitindo a eliminação de eventuais danos no posicionamento.

Em suma, estes motores são caracterizados por uma elevada fiabilidade e precisão, padrões essenciais para a execução do projecto.

4.1.1.1. CONSTITUIÇÃO

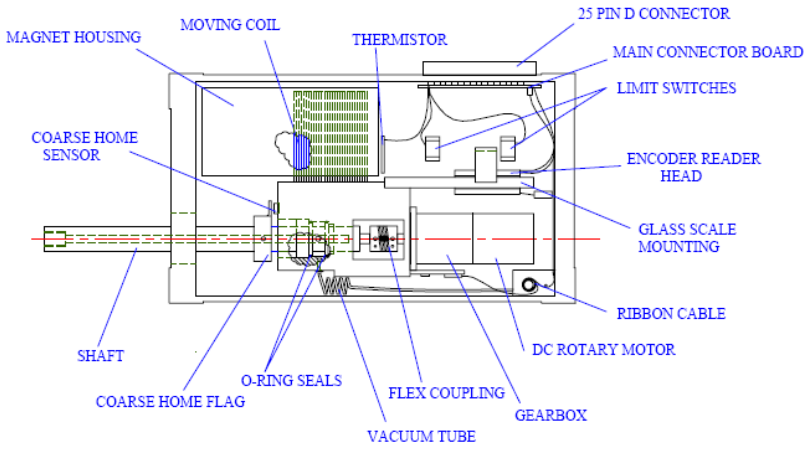


Figura 15 – Interior do motor *SMAC* [Smac07].

Como já foi referido, este motor incorpora muito mais que um simples motor. Os elementos constituintes podem ser observados na Figura 15, seguindo-se uma descrição dos elementos [Smac07]:

- Termistor – para a detecção de excesso de carga no enrolamento de bobina, originada por um aquecimento excessivo;
- Servomotor – é o elemento que permite efectuar a rotação da haste. Sendo esta rotação controlada é possível efectuar operações de aperto/desaperto de componentes, rodar um botão. Esta opção não foi utilizada neste projecto;

- Engrenagens – utilizadas pelo servomotor para efectuar o movimento rotacional;
- *Encoder* óptico linear – efectua a medição posicional, da haste sobre uma escala de vidro. Pode recorrer à escala, através de uma linha indicativa de posicionamento "*home*";
- *Encoder* óptico rotativo – encontra-se situado no topo do servomotor, indicando o posicionamento rotacional do motor;
- Interruptores de contacto – permitem fornecer um sinal eléctrico quando as extremidades da haste atingem o limite do seu curso de deslocamento;
- Elementos absorventes de choque/impacto – nas extremidades são colocados estes elementos, evitando possíveis danos por colisões mais bruscas;
- Guias lineares – guias onde se encontra assente o transportador solidário com a bobina móvel;
- Transportador – elemento que se encontra solidário com a bobina móvel, assente sobre guias metálicas, suporta a haste executando os movimentos;
- Conector – é utilizado um conector DB25 para interligar com o controlador, possibilitando desta forma a comunicação de dados.

4.1.1.2. PRINCÍPIO DE FUNCIONAMENTO

FORÇA DE DESLOCAMENTO – FORÇA DE *LORENTZ*

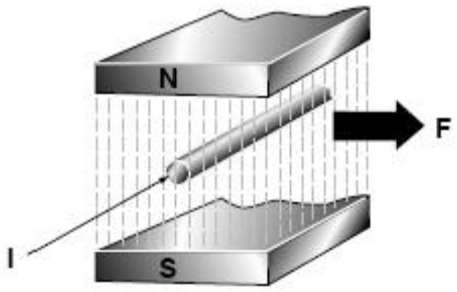


Figura 16 – Força de *Lorentz* [Smac10].

Com base na Figura 16, a força (F) originada pela passagem de uma corrente (I) num condutor inserido num campo magnético (pólo Norte – N e pólo Sul – S) é proporcional à força exercida por esse campo, pela corrente e pelo número (N) de espiras do enrolamento. Como o valor campo magnético se mantém constante, o número de espiras, também, se mantém constante, então, a força (de deslocamento) vai ser dependente da corrente que percorre o enrolamento. No caso de se mudar o sentido da corrente, esta vai influir no sentido da força, de modo a que esta força acompanha o sentido de inversão da corrente.

Em suma, a força de *Lorentz* rege-se pela seguinte expressão [Smac10]:

$$F = N \cdot I \cdot B$$

Onde,
F – força gerada
N – número de espiras do enrolamento (constante)
I – corrente que percorre o enrolamento
B – campo magnético

ACTUADOR DE BOBINA MÓVEL

Este tipo de actuador baseia-se na existência de uma bobina móvel inserida no interior de um campo magnético permanente – bloco magnético [Smac10]. Ou seja, esta bobina movimenta-se por deslizamento, através de um eixo fixo (pólo central) implementado sobre o interior deste bloco, conforme apresentado na Figura 17.

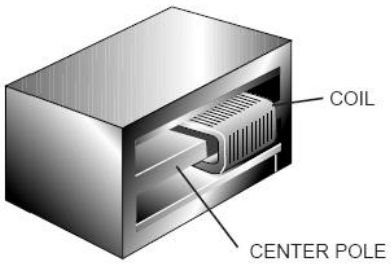


Figura 17 – Interior do motor – bobina móvel [Smac10].

O movimento é conseguido pela passagem de uma corrente eléctrica na bobina de cobre, a qual origina uma força suficiente forte para efectuar o deslizamento da bobina. Implementados num sistema realimentado permitem a obtenção de velocidades de deslocação elevadas e exactas, para além de um elevado nível de precisão, originado pelo *encoder* associado.

Em termos eléctricos este motores são de tipologia corrente contínua – motores DC, mas com elevadas vantagens ao nível da pequeníssima histerese que criam. Este factor permite-lhes uma excelente aplicabilidade em soluções que requeiram um controlo elevado do deslocamento (precisão), em ambos os sentidos.

SOFTLAND

Softland é uma rotina de *software* desenvolvida no controlador que, atribui à haste do motor a capacidade de estabelecer contacto numa superfície (neste caso a tecla do painel), com uma força programável tão baixa quanto necessário [Smac10]. Isto torna-se particularmente útil sempre que se pretende manusear componentes delicados ou de elevado valor.

Figura 18 – Actuação *softland* [Smac10].

A rotina consiste na aproximação com uma força baixa e controlada em modo de velocidade, ao mesmo tempo que o erro de posição é constantemente

monitorizado. Sendo estabelecido o contacto, o erro de posição incrementa até atingir um valor pré-programado – mantendo assim estática a posição da haste na superfície do componente. A Figura 18 faz alusão a esta rotina.

Ao algoritmo aplicado pela rotina encontra-se associado um controlador Proporcional, Integral e Derivativo (PID), de modo a assegurar uma excelente resposta no comportamento do motor, às solicitações de velocidade, aceleração e posicionamento. A parametrização do PID é configurável por *software*, existindo instruções independentes para cada um dos parâmetros (P, I e D) [Smac07].

4.1.1.3. CARACTERÍSTICAS GERAIS

Algumas características que distinguem este tipo de motores, face às restantes tipologias motorizadas existentes no mercado, podem ser evidenciadas como [Smac11]:

- Controlo absoluto sobre: força, posição, aceleração e velocidade;
- Actuador com transmissão directa, portanto com elevadíssimo grau de precisão e repetitividade face aos sistemas convencionais (motor passo a passo, por exemplo);
- *Encoder* linear integrado com escala e cabeça de leitura óptica (sem desgaste);
- Vida útil muito longa devido ao sobredimensionamento das guias lineares;
- Medida indirecta da força através da corrente;
- Canais de entradas e saídas digitais e analógicas;
- Capacidade de, a qualquer momento, comutar entre modos de operação: força, posição e velocidade;
- Aceleração e velocidade elevadas.

4.1.1.4. CARACTERÍSTICAS TÉCNICAS

Dependente do tipo de modelo, as suas características técnicas, são variáveis, no entanto, globalmente o motor *SMAC* é caracterizável por possuir [Smac10] [Smac11]:

- Sistema Linear
  - Curso até 200mm;
  - Força até 500 N;
  - Resolução da posição até 0,1 μm;
  - Força, posição, aceleração e velocidade ajustáveis.
- Sistema Rotativo
  - Binário até 1Nm (caixa de engrenagens);
  - Velocidade até 5000 rpm;
  - Resolução até 50000 incrementos por revolução;
  - Força, posição, aceleração e velocidade ajustáveis.

Dependendo do modelo da bobina seleccionado, os sistemas controlador / motor podem ser alimentados a 24Vcc ou a 48Vcc, consumindo valores de corrente que variam entre 1,5A e 6A.

O controlo da corrente da bobina providencia informação em malha fechada, que pode ser constantemente monitorizada. Esta monitorização estabelece um permanente controlo através dum *encoder* que pode ter resoluções de 5; 1; 0,5; ou 0,1 μm.

ESPECIFICAÇÕES

O modelo utilizado nesta aplicação encontra-se referenciado como LAL35-025 pela *SMAC*, apresentado anteriormente na Figura 14 e possui as seguintes características:

- Tensão de alimentação: 24V
- Corrente: 3A
- Dimensões: 135x90x35
- Curso: 25mm

- Força máxima: 12,5N
- Força contínua: 10N
- Força constante: 10N/A
- Massa (movimentação): 0,15kg
- *Encoder* Linear (resolução): 5 μm (*standard*)
- Peso: 0,95 kg

4.1.2. CONTROLADOR

4.1.2.1. ESTRUTURA

A acção do controlador vai incidir sobre o controlo do motor, isto é, quando é ordenado ao motor que inicie uma deslocação, através de *software*, vai ser definida (calculada) essa movimentação. O posicionamento do transportador (associado à bobina móvel) do motor *SMAC* vai ser monitorizado pelo controlador, ao longo do tempo, assim como, a força que está a ser desenvolvida. Desta forma, pretende-se que o posicionamento actual coincida com a movimentação definida inicialmente. Esta diferença entre o posicionamento real e o desejado é referenciado como erro posicional, o qual tende a ser zero por acção do controlador. Todo este controlo posicional é possível devido à malha de realimentação, que envia o posicionamento actual da haste do motor, obtido pela leitura do *encoder* óptico [Smac07].

Figura 19 – Controlador *SMAC*, LAC-1.

É possível utilizar, este mesmo sistema, em malha aberta. Para tal, o controlador vai gerar uma corrente constante, aplicável na bobina, que irá desenvolver uma força constante na extremidade da haste. Caso não exista

resistência a opor-se ao movimento, a haste ganha movimento devido à força desenvolvida pela passagem da corrente na bobina.

O modelo de controlador utilizado, também, é fornecido pela *SMAC* possibilitando desta forma uma melhor interacção entre os dois elementos – motor e controlador. Este modelo é designado por LAC–1 e encontra-se apresentado na Figura 19.

4.1.2.2. *HARDWARE*

Neste controlador LAC-1, de todo o seu *hardware*, destaca-se o possuir um sistema integrado para o controlo/*driver* do motor de um eixo. As instruções estão implementadas sob o formato de mnemónicas que podem ser activadas via interface RS-232 [Smac97] [Smac07]. Estas instruções podem ser executadas directamente ou usados para criar macros conjuntamente com outras instruções que são guardadas em memória interna RAM (*Random Access Memory*) não volátil (NVRAM).

A estrutura física do controlador permite dispor nos topos interfaces com diferentes funcionalidades. Sendo cinco o total de interfaces disponíveis, estas são apresentadas na Figura 20, sendo elas:

- *Power*;
- *Servo*;
- User I/O;
- I/O Exp;
- RS-232.

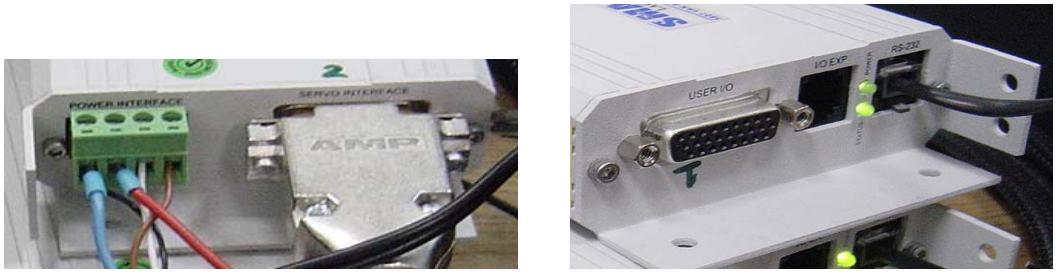


Figura 20 – Interfaces do controlador.

FONTE DE ALIMENTAÇÃO

Este conector de 4 terminais permite a entrada da alimentação para o controlador e a saída da alimentação para o motor. As características de alimentação comportam valores compreendidos entre os +11 e os +50VDC na entrada do controlador, sendo desta forma obtida a alimentação necessária para o motor (saída do controlador). Como o controlador foi desenvolvido para aplicação dos motores *SMAC*, então as características de alimentação deste deve garantir as condições de alimentação dos motores, cujos valores se encontram tipificados nos +24 ou +48VDC. Assim, alimentando a +24 ou +48VDC o controlador, estão garantidas as condições para alimentação de qualquer tipo de motor da *SMAC*, no entanto, a saída do controlador para alimentar um motor pode variar desde os +12 aos +48VDC.

DRIVER DE POTÊNCIA

Internamente, este controlador possui um *driver* que permite a ligação directa a um motor DC normal ou a este tipo de motor. Este elemento de potência é baseado num amplificador de potência, comutado através de um PWM (*Pulse Width Modulation*). Tem uma capacidade para fornecer até 3A de corrente contínua ou 6A de corrente de pico (num máximo de 200ms), a uma frequência de comutação de 19 531kHz.

Este *driver* encontra-se protegido contra sobreaquecimento, ficando inactivo quando for atingida uma temperatura superior a 140ºC, accionando a respectiva *flag* para controlo no *software*.

INTERFACE DO MOTOR – SERVO INTERFACE

Como o próprio nome faz referência, esta interface vai conectar o controlador ao motor através de um conector do tipo DB, 15 pinos fêmea. Os sinais presentes neste conector permitem:

- utilizar um *encoder* em quadratura;

- dispor de sinais auxiliares para controlo do sistema, designadamente os sinais de *limit*+, *limit*-, *home* e *fault* (estas entradas dedicadas são de tipologia TTL[4]).

➢ ENCODER

Este controlador possui um canal, com duas fases (*single ended* e diferencial), disponível para receber sinais de um *encoder* externo, com sinal em quadratura. Também tem a capacidade para receber um sinal de *index*, cujo intuito deste sinal é focalizar um determinado acontecimento (por exemplo a posição *home*, determinada fase em que se encontram os sinais, etc.). Na Figura 21 são apresentadas duas tipologias para *encoder*, sendo a da esquerda *single ended* e a da direita diferencial.

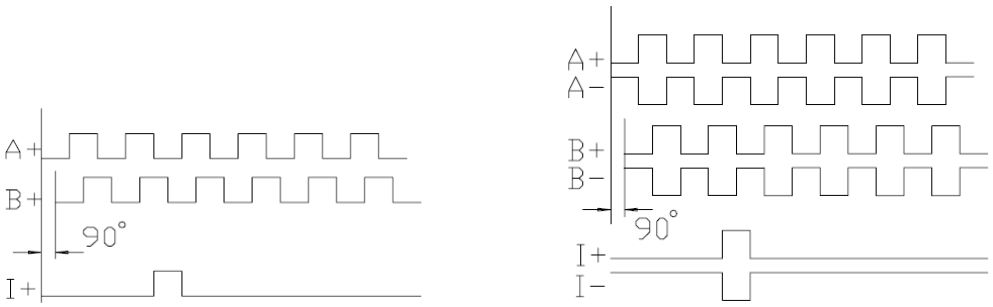


Figura 21 – *Encoder* com sinal em quadratura e sinal *índex* [Smac07].

Como já referido anteriormente, este canal possui duas fases que possibilitam do controlador a interagir com *encoders* cuja as saídas são do tipo *open collector* ou *totem pole*, conforme na Figura 22. As fases do canal, assim como o sinal *índex*, são passíveis de serem alterados por meio de instruções.

[4] TTL – *Transistor Transistor Logic*, são circuitos integrados cuja etapa de entrada e saída recorrem ao transístor como elemento de interligação com o exterior (existem diferentes configurações para estas etapas) possuindo valores lógicos de tensão de 0V e +5V.

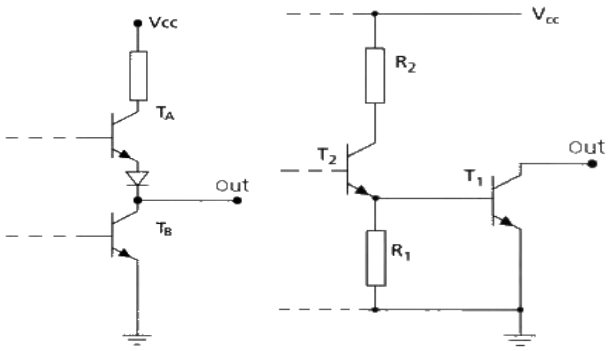


Figura 22 – Tipologias de saída em *totem pole* e *open collector*.

O *encoder* deve ser testado para calibração, antes do motor começar a movimentar-se. Caso não exista esta precaução pode originar um erro de posicionamento, com consequências potencialmente danosas para o sistema onde se encontra implementado. Desta forma o fabricante apresenta uma rotina, em *software*, a qual deve ser incluída e se necessário adaptada, que permite testar o *encoder* [Smac07]. No geral esta rotina consiste em partir da posição de *home* da haste, efectuar retracção da mesma a uma determinada força e, verificar se o número de "impulsos" dados pelo *encoder* está enquadrado com o esperado. Alternativamente, e caso a retracção da haste não esteja enquadrada com os parâmetros desejáveis, é efectuado o mesmo processo mas com a extensão da haste (sentido contrário).

- ENTRADAS DIGITAIS DEDICADAS

Este tipo de entrada tem a particularidade de possuírem uma função específica, ou seja, definirem o *limit*+, *limit*-, *home* e *fault*. A sua estrutura é esquematizada na Figura 23, sendo cada uma destas entradas activa num nível lógico baixo e possuidoras de resistência de *pull-up*, cujo valor é de 2,7kΩ.

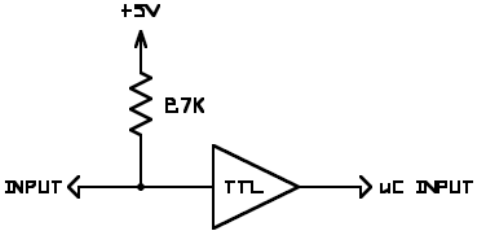


Figura 23 – Entrada digital dedicada [Smac97].

O objectivo das entradas *limit*+ e *limit*- é sinalizar que um eixo atingiu o limite do seu deslocamento. A entrada *home* tem como finalidade a detecção de alguma indicação de posição através de sensor, podendo ser obtida com recurso ao *encoder*. A utilização da entrada *fault* proporciona que um dispositivo externo possa sinalizar uma falha (por exemplo: excesso de temperatura de um motor). Este último sinal acumula funções, pois encontra-se ligado internamente à detecção de excesso de temperatura do *driver*. Assim, o sinal de falha pode ser interno ou externo, dependendo da fonte que emite o sinal.

ENTRADAS/SAÍDAS

Este conector providência nos seus terminais, para uma utilização indiscriminada mas condicionada às suas capacidades, os seguintes sinais:

- 8 canais digitais de entrada do tipo HCT[5], de uso geral;
- 8 canais digitais de saída do tipo HCT, de uso geral;
- 3 canais de entrada, conversores AD com uma resolução de 10 bits.

A obtenção destes sinais é conseguida com recurso a um conector DB, 26 pinos, HD fêmea.

➢ ENTRADAS/SAÍDAS DE USOS GERAL

Todas as entradas/saídas têm como etapa inicial/final um *buffer*, que disponibiliza um certo isolamento entre o *hardware* externo e interno, estando este *buffer* a cargo do 74HCT541. Uma interface de expansão surge como uma opção, que permite atingir os 64 canais de I/O (*Input/Output*).

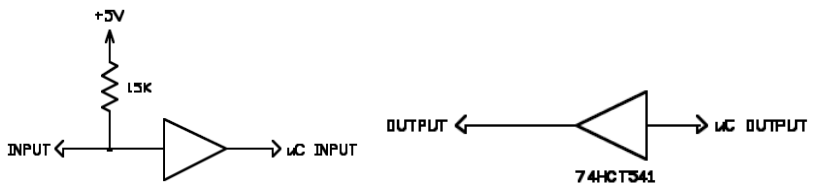


Figura 24 – Entrada e saída digital de uso geral [Smac97].

[5] HCT – *High-speed CMOS with TTL*.

Na Figura 24 está apresentada uma entrada de uso geral. Cada entrada deste tipo é constituída por uma resistência de *pull-up*, cujo valor é 15kΩ. Igualmente encontra-se esquematizada uma saída digital de uso geral, a qual apenas recorre a um *buffer* de saída.

- ESTADO DAS ENTRADAS/SAÍDAS

As entradas e saídas podem ser definidas em função das necessidades, para tal existem várias instruções que lidam com as elas. Como tal a entrada ou saída pode ser definida como activa alta ou baixa e a partir daqui, apenas é necessário atribuir e/ou reconhecer o estado ON/OFF. Para uma análise mais detalhada encontra-se definida na Figura 25 que ilustra as condições/estados que definem uma I/O.

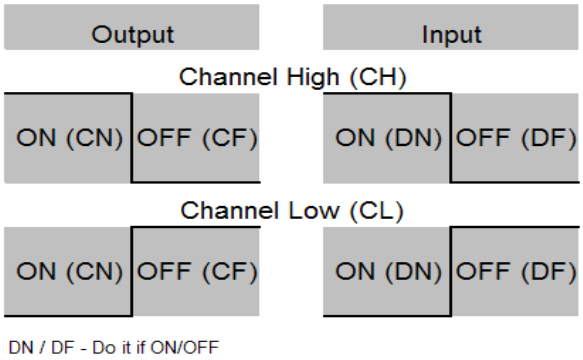


Figura 25 – Estados das entradas e saídas.

- CONVERSOR A/D

O número de canais de entrada A/D que o controlador possui são 10, mantendo desta forma compatibilidade com outras versões, no entanto, apenas 4 se encontram disponíveis. Deste 4 canais um deles está com funções dedicadas no controlo do sobreaquecimento do *driver* de potência, os restantes 3 encontram-se disponíveis para uso geral e têm como tensão de referência +5VDC.

Sempre que é executado a instrução para ser efectuada a conversão A/D, o canal específico efectua essa mesma conversão e é apresentado ou armazenado em registo o valor convertido.

EXPANSÃO DE ENTRADAS/SAÍDAS

Encontrando-se disponível mas não funcional (prevista para aplicações futuras) esta expansão de entradas e saídas tem por base a comunicação série RS-422, aplicável num conector RJ25. Esta expansão encontra-se limitada a 64 canais.

RS-232

Esta porta de comunicação RS-232, obtida por intermédio de um conector RJ25, permite interligar o controlador LAC-1 com um computador, permitindo o envio de comandos. Tem possibilidade de variação de *baudrate* sendo que a trama da mensagem é fixa e constituída por: 8 bits de comprimento, um stop bit e sem paridade. O controlo de fluxo é obtido por *software* e do tipo *Xon/Xoff*, não sendo possível efectuar o mesmo através de *hardware* [Smac97].

Esta porta é fundamental para o funcionamento do sistema, pois é a partir dela que são enviados os comandos e são recebidos os dados, da movimentação efectuada pelo motor.

4.1.2.3. *SOFTWARE*

Os controladores da *SMAC* utilizam uma linguagem de programação semelhante à estrutura de um código em *assembler*, sendo este código mantido numa memória não volátil. As instruções ficam definidas por duas letras seguidas de um valor numérico. Os programas são assentes em linhas de código denominadas de “Macros”, as quais são numeradas e o programa pode executar diferentes macros, chamar subrotinas e efectuar saltos, condicionais ou não, para diferentes locais do programa [Smac97] [Smac07].

Este controlador disponibiliza o recurso a 256 registos de 32 bits para programação, em memória NVRAM. Assim, variáveis, resultados, operações intermédias podem ser guardadas para posterior uso. Só como nota, ao registo 0 é-lhe atribuído o funcionalismo de acumulador.

No decorrer da programação o motor pode movimentar segundo três modos, isto é, a sua deslocação pode ser através do modo de força, o modo de velocidade e o modo de posição.

FORMATO DAS INSTRUÇÕES

De modo a elucidar um pouco como são compostas as linhas de código e suas instruções, ficam de seguida alguns exemplos e comentários.

- AL10000,AR220
  - *Accumulator Load value 10000 (*AL10000*)*, *Accumulator to Register 220 (*AR220*)*
- RA50,AA1,AR50
  - *Register to Accumulator 50 (*RA50*), Accumulator Add 1 (*AA1*), Accumulator to Register 50* (AR50)
- MD10,QM,MN,SQ10000,MJ20
  - *Macro Definition 10* (MD10), *force mode* (QM), *Motor oN* (MN), *set force 10000* (SQ10000), *Macro Jump 20* (MJ20)

4.1.3. CÉLULA DE CARGA

Com a movimentação do sistema obtém-se o valor do deslocamento da haste do motor mas falta a outra grandeza, não menos importante, a força que a haste vai exercer sobre determinado objecto (neste caso serão teclas). Para alcançar tal objectivo é colocado na extremidade da haste do motor, um sensor de força designado por célula de carga. A célula de carga apresentada na Figura 26 corresponde ao modelo utilizado neste projecto.

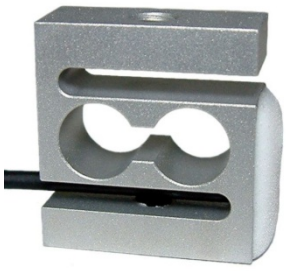


Figura 26 – Célula de carga KD24s da ME-Meßsysteme [ME07].

No essencial, a funcionalidade inerente à célula de carga é converter uma força mecânica exercida pelo motor, a que esta se encontra sujeita num determinado momento, num sinal eléctrico passível de ser quantificado.

Em complemento com a célula de carga surge o amplificador de sinal. Por si só o sinal obtido do transdutor pode não ser o ideal para uma utilização numa fase subsequente, como tal, o recurso a um amplificador de sinal vai permitir que este se torne mais forte e imune a interferências.

4.1.3.1. PRINCÍPIO DE FUNCIONAMENTO

Uma célula de carga, em termos gerais, serve para medir uma força. Estando esta grandeza aplicada num objecto (de um dado material), vai originar uma alteração dimensional do mesmo. Partindo desta relação entre a força e a dimensão, é possível quantificar essa força, necessitando para tal de ser conhecida diferença dimensional do objecto. Convém salientar que o material sujeito a esta alteração dimensional, ou seja sob medição, tem de ser elástico.

A força é uma grandeza abrangente do ponto de vista físico (mecânica), logo é possível relacioná-la com um leque alargado de propriedades dos materiais. Assim, em sistemas electromecânicos existe o interesse em conhecer (quantificar) algumas das propriedades, como por exemplo tensão, extensão, compressão, flexão e deformação.

As células de carga podem ser classificadas quanto ao tipo de sinal que se obtém na saída, ou seja, sinal hidráulico, pneumático ou eléctrico. Sendo este um projecto com tipologia eléctrica, é esta a área aplicável da célula de carga. Usualmente o tipo de força aplicada ao material vai definir dois modos de funcionamento: em tensão ou compressão. Em tensão, a estrutura da célula vê aumentada a sua estrutura ao longo do mesmo eixo onde é exercida essa força, enquanto sob compressão, a sua estrutura é vai diminuir nesse eixo [NI11] [Webster99]. Na Figura 27 está representado o fenómeno de extensão do material, onde a forma inicial é a correspondente à linha contínua e, consequentemente, com uma força aplicada o material toma o formato

correspondente à linha tracejada. A compressão corresponde ao fenómeno inverso, onde acontece uma redução do comprimento e um aumento da largura.

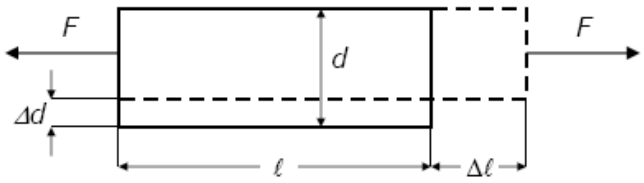


Figura 27 – Material sujeito a forças de tracção.

A partir deste momento, relacionando este deslocamento físico com uma grandeza eléctrica, é possível quantificar esse deslocamento. A grandeza utilizada para o efeito é a resistência eléctrica, pois o seu valor resistivo é obtido pela seguinte expressão,

$$R = \rho \frac{l}{s}$$

Sendo constante o valor da resistividade (ρ), logo por mais pequena que seja a alteração no comprimento (l) ou na secção (s), o valor da resistência (R) vai variar [Webster99] [Bentley05]. Com tudo isto, uma corrente que atravesse a resistência ou uma queda de tensão obtida aos terminais da resistência, são grandezas dependentes do valor resistivo dessa resistência.

Em suma, a leitura obtida por uma célula de carga baseia-se numa variação da resistência eléctrica, que se encontra incorporada na estrutura da célula, a qual é proporcional à variação da dimensão da estrutura da célula. Para traduzir esta grandeza física em eléctricas existe no mercado um transdutor bastante usual nestas aplicações, cuja designação é de extensómetro (*strain gauge* ou *strain gauges*) [Bentley05] [NI11] [Ricelake07].

EXTENSÓMETROS

Elementos que permitem obter aos seus terminais um valor resistivo variável em função da deformação à qual são sujeitos. A deformação pode surgir por

intermédio de um deslocamento, usualmente associado a uma força, mas também pode ser por pressão, torção uma carga [Ricelake07]. Conforme a Figura 28 um extensómetro possui uma constituição bastante simples englobam um fio condutor de secção muito reduzida, aplicado numa fina camada de material elástico e dispondo de dois terminais de acesso ao fio condutor (resistência variável).

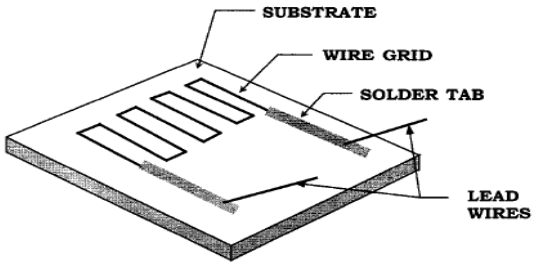


Figura 28 – Estrutura de um extensómetro.

Aplicado numa qualquer estrutura elástica, de forma solidária com o deslocamento desta, permite relacionar as variações da estrutura com a variação da sua resistência. Devido à sua estrutura, são transdutores vocacionados para a obtenção de pequenos deslocamentos. De formatos variados, o mais usual no mercado, encontra-se representado na Figura 29.

Figura 29 – Extensómetro, modelo produzido por SHOWA [Shoma11].

Na Figura 30 está um leque alargado de formas e conjugação de vários extensómetros, permite abranger uma vasta área na obtenção leituras respeitante a forças (e suas grandezas adjacentes).

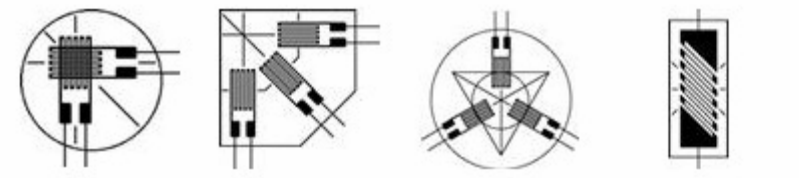


Figura 30 – Tipologias com extensómetros [Shoma11].

IMPLEMENTAÇÃO

Para uma correcta obtenção da variação a que vai ser sujeito o material em análise, o extensómetro deve ficar localizado num ponto onde essas variações são mais fidedignas, sendo exemplo de aplicação o da Figura 31

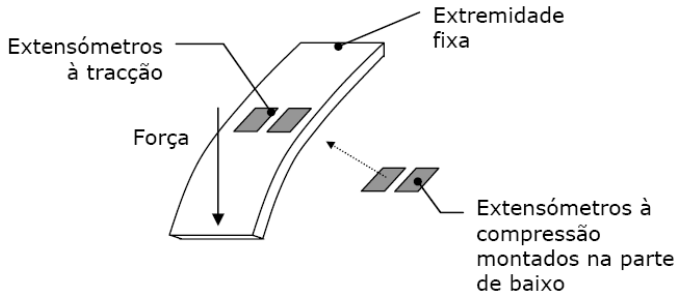


Figura 31 – Colocação de extensómetros.

Devido às variações no deslocamento, e consequentemente ao valor emergido nos terminais do extensómetro, serem extremamente diminutas existe a necessidade de implementar um circuito adicional. A ponte de *Wheatstone* é um circuito com elevado potencial neste tipo de aplicação, pois é um circuito ideal para a medição de pequenas variações no valor da resistência [NI11].

O circuito em ponte de *Wheatstone* é formado por quatro ramos resistivos R1 a R4, alimentados por uma tensão externa *Vin*. A tensão de saída – *Vout* é obtida em conformidade com os valores que se encontram nos ramos. A configuração de ponte de *Wheatstone* e as relações entre os seus elementos estão explícitos na figura seguinte.

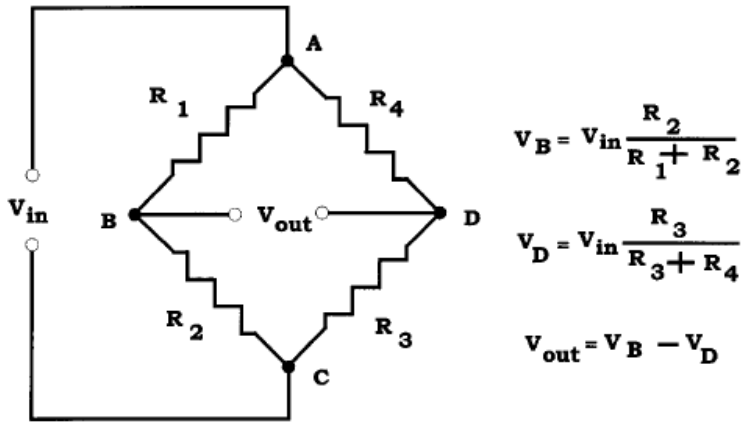


Figura 32 – Ponte de *Wheatstone* [Webster99].

Do circuito da Figura 32, quando a tensão se saída (*Vout*) tem o valor 0V, diz-se que a ponte está equilibrada. Nesta situação a tensão no ponto B é igual à tensão no ponto D.

Todos os ramos da ponte de *Wheatstone* podem ser formados por extensómetros e apenas um deles estar sujeito à deformação do material, sendo este ramo designado por activo. É possível implementar a ponte de *Wheatstone* com diferentes configurações, em que os ramos activos variam em número e disposição [Ricelake07].

4.1.3.2. APLICAÇÕES

A célula de carga pode assumir diferentes formatos tendo em conta factores diversos como:

- O fabricante;
- A aplicação pretendida;
- O ambiente onde vai ser utilizada;
- A carga a que vai ser sujeita;
- A precisão.

Na Figura 33 são apresentados alguns formatos básicos, encontrando-se representadas as forças que podem ser aplicadas. É de referir que todos estes modelos são susceptíveis de variações e derivações, levando em conta os factores abordados anteriormente.

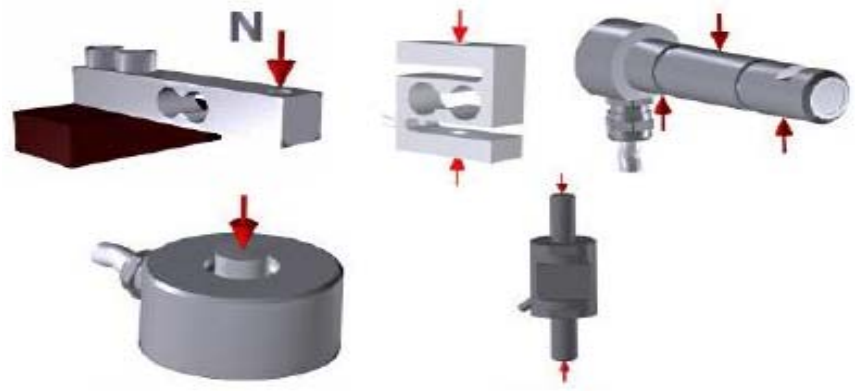


Figura 33 – Modelos de células de carga da ME-Meßsysteme.

4.1.3.3. CARACTERÍSTICAS

A célula de carga utilizada neste projecto é um modelo compacto, em forma de "S", vocacionado para aplicações de testabilidade funcional, em diferentes áreas, representada na Figura 26. Internamente ou externamente a determinação da força/pressão a que se encontra sujeita é centralizada, originado um deslocamento paralelo do suporte [ME07]. A gama de forças máximas dentro deste modelo comporta cargas que vão para além da sua força nominal. Nas gamas mais pequenas, como é o caso desta célula, essa força pode atingir quatro vezes o seu valor nominal, enquanto que para gamas mais elevadas essa força fica pela duas vezes.

Os valores obtidos por esta célula de carga, no que diz respeito à exactidão são de 0,1% para uma tensão na saída de 0,5mV/V. De seguida são apresentadas algumas das características do sensor [ME07]:

- Força: Tensão/Compressão;
- Construção: *Double bending beam*;
- Material: Alumínio;
- Força nominal: 10N;
- Deflexão nominal: 0,05mm;
- Tensão de alimentação: 10V;
- Saída nominal (tensão): 0,5±10%mV/V;
- Cablagem: 2 condutores para alimentação e 2 condutores de saída do sinal.

4.1.3.4. AMPLIFICADOR

Devido às características do sinal obtido na célula de carga (na ponte de medida), uma etapa de amplificação torna-se imprescindível. Assim, a amplificação permite dispor de um sinal em melhores condições para que este seja usado de uma forma mais eficiente na etapa seguinte, ou seja, no controlador. O sinal à saída da célula de carga apresenta pequeníssimos

valores de tensão os quais têm de ser amplificados para darem entrada no controlador

Figura 34 – Amplificador de sinal GSV-1 da ME-Meßsysteme [ME11].

O amplificador utilizado neste projecto é o da Figura 34, sendo o complemento da célula de carga. Esta opção vai de encontro com as especificações do fabricante e nesse sentido é potencializado o sinal adquirido pela célula de carga.

Terminals	
1	$-U_D$: negative differential input
2	$+U_D$: positive differential input
3	$+U_S$: positive bridge supply (5V)
4	$-U_S$: negative bridge supply (GND)
5	GND : ground
6	$+U_A$: analog output
7	$+U_B$: voltage supply (12V or 24V)
8	T: control input, zero balancing

Figura 35 – Terminais do amplificador anterior [ME11].

Como pode ser observável na Figura 35, os pinos de 1 a 4 permitem ligar as células de carga, e se estas forem do mesmo fabricante, está garantida uma fiabilidade superior do sinal obtido à saída (o pino 7). Se ligado à tensão de alimentação, o pino 8, obtém-se o acerto a 0.

CARACTERÍSTICAS

Este modelo de amplificador encontra-se especificamente vocacionado para trabalhar em ponte, mais concretamente quando estas são preenchidas por extensómetros. Possui um conjunto diversificado de caixas externas e de opções que vão desde amplificação configurável por *jumpers*, guardar valor de

pico, diferentes formas de ligação (conectores), etc. Em seguida são descriminadas algumas características deste amplificador [ME11]:

- Tensão de alimentação de +12V ou+24V;
- Sensibilidade na entrada de 2mV/V;
- Frequência máxima de 250Hz;
- Tensão para alimentação da ponte +5V;
- Impedância de entrada de 20MΩ;
- Classe de exactidão de 0,1;
- Saída em tensão de ±5V, ±10V ou em corrente de 4-20mA;
- Amplificação variável de 1, 2, 4 ou 10 vezes (por *jumpers*);
- Alimenta até quatro células de carga em paralelo (máximo de 350Ω por ponte).

4.2. PLACA DE DESENVOLVIMENTO *FEZ-COBRA*

Figura 36 – Aspecto exterior do *kit* de desenvolvimento.

O elemento fulcral deste projecto reside na placa de desenvolvimento pois é centro nevrálgico de todo o processo operativo. De modo a corresponder às expectativas criadas para o projecto, a escolha recaiu sobre uma placa do fabricante *GHI Electronics, LLC*, denominada por *Fez-Cobra*. Esta placa foi adquirida num *kit* de desenvolvimento designado por *Fez-Cobra OEM Box (3.5'' Display)*. Este *kit*, para além da placa já referenciada, é composto por uma caixa e por um *display* TFT táctil de 3,5'' de cores, conforme apresentado na Figura 36 e na Figura 37.

Figura 37 – Interior do kit de desenvolvimento.

Com esta placa o potencial de desenvolvimento e abrangência de aplicações para sistemas embebidos é colocado num patamar elevado, isto em virtude das características técnicas apresentadas pela *Fez-Cobra* serem excelentes, no seu segmento.

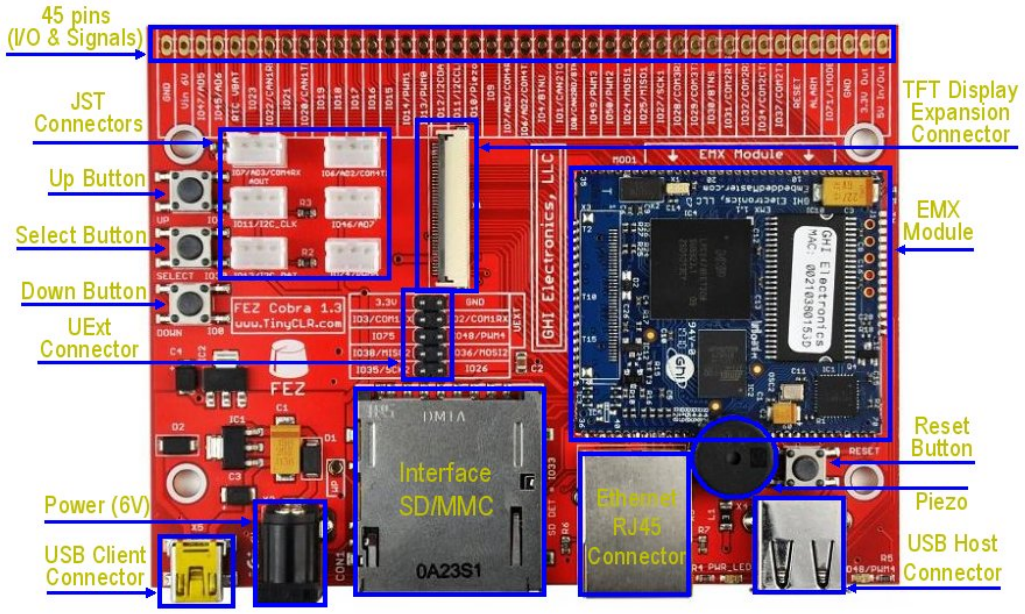


Figura 38 – Placa *Fez-Cobra* [GHI10a].

Como pode ser observado na Figura 38, um leque alargado de opções encontram-se disponíveis para ligação directa aos respectivos conectores ou para acesso aos pinos.

4.2.1. CARACTERÍSTICAS GERAIS

A placa de desenvolvimento *Fez-Cobra*, utiliza um módulo de processamento designado por EMX (Figura 39), cujo núcleo central é constituído pelo processador ARM 7, onde corre o *.Net Micro Framework* (NETMF), da Microsoft. Este recurso permite o desenvolvimento de código eficiente num ambiente "*user friendly*", com recurso à linguagem de programação C#, sob a plataforma do *Visual Studio Express Edition 2010*.

Figura 39 – Módulo EMX da *GHI*, aplicado na placa *Fez-Cobra*.[GHI11]

Em resumo a placa *Fez-Cobra* possui como principais características [GHI10a] [GHI10c] [GHI11] [NPX09]:

- Processador ARM7, 32 bits a 72MHz, inserido no módulo EMX;
- Memória: 16MB RAM e 4,5MB do tipo *Flash*. A memória disponível (livre) situa-se nos 12MB de RAM e nos 3MB de *Flash*;
- Conector de expansão para *display* TFT a cores;
- Conector Ethernet, tipo RJ45 (10/100Mbps);
- Entradas e saídas de uso geral, algumas têm disponíveis interrupções (*interrupts*);
- Uma saída analógica (possibilidade de áudio);
- Duas linhas de barramento SPI (*Serial Peripheral Interface*);
- Interface I2C (*Inter Integrated Circuit*);
- Sete entradas analógicas para conversão digital, duas delas dedicadas ao *display* táctil;

- Seis sinais de PWM;
- Comunicação série constituída por quatro UART's (*Universal Asynchronous Receiver Transmitter*);
- Dois canais para comunicação CAN;
- Conector de cartão SD/MMC;
- Conectores USB para ligação *Host* e *Client;*
- Expansibilidade por módulos: WiFi, GPS, Bluetooth, XBee entre outros;
- Possui RTC (*Real Time Clock*) interno;
- Alimentação por USB ou fonte independente;
- *Debug Interface*: USB, Série ou Ethernet.

4.2.1.1. MÓDULO EMX

Este módulo é uma combinação de *hardware* e *software* inserido numa placa de reduzidas dimensões, a qual contempla a maioria das características apresentadas anteriormente. No seu núcleo encontra-se o microcontrolador LPC2478 do fabricante *NXP Semiconductors*, cujo CPU (*Central Processing Unit*) é um ARM7 a 72MHz, de 32 bits.

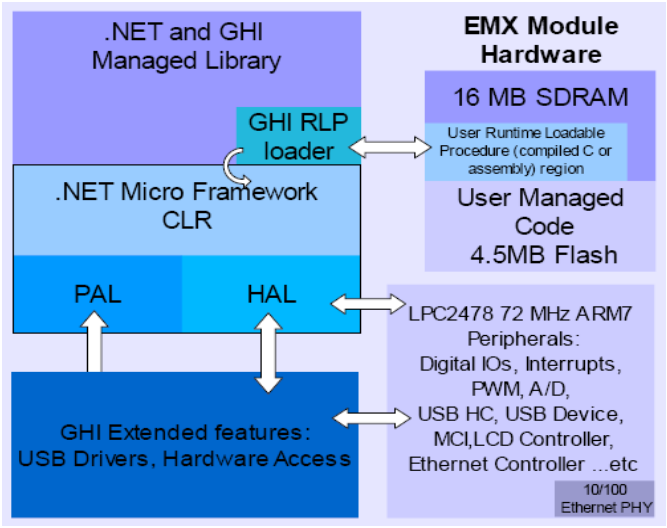


Figura 40 – Diagrama de blocos do módulo EMX [GHI11].

Ao nível de memória este módulo proporciona uma RAM de 16MB (dos quais aproximadamente 12MB são memória livre) e uma memória flash de 4,5MB. Da memória flash 4MB são externos, dos quais 1MB são utilizados para efectuar o

boot loader, ficheiros do sistema (*system assemblies*) e outras fontes da *GHI*. Enquanto os restantes 3MB são para uso da aplicação desenvolvida. Os 0,5MB de flash são internos e usados para a execução da *Common Language Runtime* (CLR) do NETMF. A Figura 40 apresenta sobre forma de blocos esta abordagem.

Uma característica interessante do módulo EMX é o *Runtime Loadable Procedure* [GHI11], a qual permite ao utilizador carregar o seu código compilado em formato nativo (C ou *assembly*) e correr através do NETMF. Outras potencialidades deste módulo permitem dispor um conjunto de opções que vão desde o TCP/IP, SSL, FAT, USB host, PPP, Ethernet, CAN, Conversor A/D entre outras.

MICROCONTROLADOR

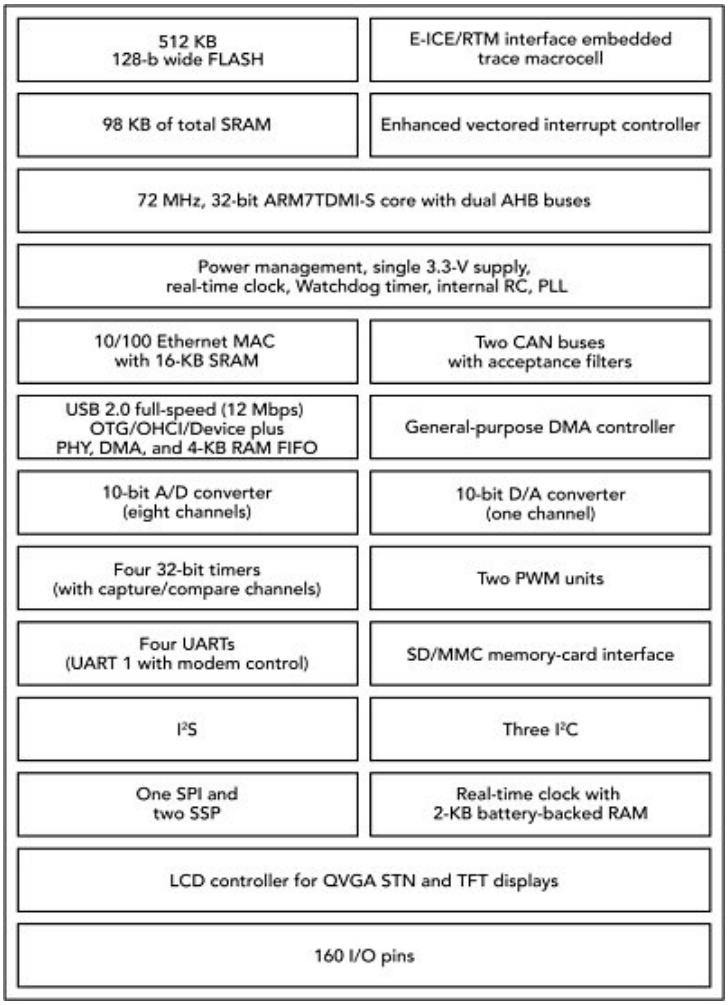


Figura 41 – Principais características do LPC2478, da NXP [NPX11].

A NXP Semicondutor desenvolveu a família de microcontroladores LPC24xx de modo a garantir uma gama de aplicações alargada, onde a diversidade nas comunicações fosse um factor essencial. Deste modo o processador LPC2478 é a base de suporte nas características apresentadas pela *Fez-Cobra*. Na Figura 41 encontram-se as principais características que se podem obter deste microcontrolador.

SEQUÊNCIA DE ARRANQUE

O módulo EMX possui três estruturas de *software* embebido para efectuar o arranque da *Fez-Cobra*, contemplando [GHI11]:

- *GHI boot loader* – inicia a memória Flash e RAM, procurando um *TinyBooter* válido. Permite efectuar a actualização do *TinyBooter* (vem definido de fabrica e não é passível de actualização);
- *TinyBooter* – é o *software* desenvolvido pela Microsoft, o qual executa o reconhecimento do *hardware* preparando as fontes deste *hardware* para serem tratadas pelo *firmware*. Também serve de base para a actualização do *firmware*;
- EMX *firmware* – é a principal área onde corre o *software* embebido NETMF e a aplicação.

A sequência de arranque pode ser alterada pela mudança dos estados em que se encontram determinadas entradas (neste caso correspondem aos interruptores disponibilizados na *Fez-Cobra*).

4.2.1.2. ALIMENTAÇÃO

A alimentação da *Fez-Cobra* pode ser efectuada por uma das duas opções disponíveis, as quais são:

- USB (mini A) – serve para alimentar directamente a placa recorrendo a uma porta USB de um computador. Deste modo consegue obter os +5V necessários à sua alimentação;

- *Power Jack* (PJ-102A) – Conector de alimentação independente através do qual é possível alimentar a *Fez-Cobra* até +12V contínuos.

Dois reguladores de tensão fazem parte da *Fez-Cobra*, permitindo a obtenção das tensões para a alimentação dos circuitos internos da mesma. Um primeiro bloco de regulação de +5V é utilizado quando a alimentação é independente, ou seja, para tensões até +12V. Contudo, é aconselhado pelo fabricante, a não utilização de tensões de alimentação superiores a +6V, evitando desta forma um aquecimento excessivo do regulador. O segundo bloco de regulação é utilizado para a obtenção de +3,3V, fornecendo a tensão de alimentação para o módulo EMX, por exemplo.

Os consumos de corrente apresentados pelo fabricante contemplam três áreas de actividade da *Fez-Cobra*, assim os modos são:

- *Active Mode*;
- *Idle Mode*;
- *Hibernate Mode*.

4.2.1.3. ENTRADAS E SAÍDAS

O microcontrolador possui 208 pinos, e destes 160 pinos de entrada ou saída de uso geral (*General Purpose Input/Output* – GPIO), passíveis de configuração com resistência de *pull-up/down*. Uma grande parte destes pinos de uso geral, têm associadas outras funções, as quais podem ser seleccionadas através de configuração. Por exemplo, pino 0 do porto0 (P0[0]) é um GPIO digital, ou é recepção do canal um do CAN (RD1), ou é transmissão da UART3 (TXD3), ou é recepção/transmissão do I2C1 (SDA1).

As entradas digitais devem ter como limite máximo (nível alto) +3,3V, no entanto, a *Fez-Cobra* permite que estas entradas sejam tolerantes até +5V. Deve-se tomar algum cuidado na forma como se configura o pino e tentar manter sempre a mesma definição, pois a utilização de um pino simultaneamente como saída e entrada pode danificar o processador.

Existem quatro áreas para se aceder aos pinos, que são disponibilizados pela *Fez-Cobra*, sendo que uma destas ligações é exclusiva para o *display*, assim:

- Barramento de I/O – É composto por 45 pinos, aos quais se tem acesso através de uma linha perfurada na placa (*pitch*[6] 0,1'');
- Conector *UEXT (Universal Expansion Connector)* – este conector possui um formato de 2x5, ou seja 10 pinos (pitch 0,1''), onde usualmente contempla as comunicações I2C, SPI e RS-232, mais os pinos de alimentação. Neste caso a comunicação I2C não está presente, além da linha SS (*Slave Select* ou *Chip Select*) respeitante ao SPI. Como tal, encontram outras linhas disponíveis no local destas últimas, permitindo deste modo a compatibilidade com as extensões produzidas pela *GHI*, para opções como leitor de MP3, GPS, Acelerómetro entre outros.
- Seis conectores JST, constituídos por três pinos, disponibilizando algumas entradas e saídas. Também são utilizados para aplicação de módulos para expansão produzidos pela *GHI*, como por exemplo o *XBee*, o *Bluetooth*, e outros módulos de aplicações sensoriais.
- Conector FPC (*Flexible Print Circuit*) para ligação aos pinos de comunicação com o *display*.

4.2.1.4. INTERRUPÇÕES

As fontes de interrupções externas são compostas por quatro entradas dedicadas. Para além destas fontes, o porto GPIO0 e o GPIO2 disponibilizam os seus pinos para a obtenção de interrupções, as quais são mapeadas pela interrupção externa 3. Internamente, todos os periféricos possuem linha de interrupção (eventualmente até podem possuir mais que uma linha). O vector de interrupção do processador permite estabelecer um máximo de 32 tipos de interrupção.

[6] Representa o espaço entre dois furos na placa de circuito impresso.

4.2.1.5. COMUNICAÇÕES

RS-232

A comunicação série RS-232 é constituída por quatro unidades, com 16 bytes de *buffer* de tipologia FIFO (*First In First Out*), na recepção e transmissão. Possuem gerador de *baud rate* interno configurável. Duas destas unidades possuem características particulares:

- UART1 encontra-se disponível com linhas de controlo para modem: CTS, DCD, DTS, DTR, RI, DTS. Por *hardware* e *software* é possível a implementação de controlo de fluxo;
- UART3 encontra-se provida para dar suporte a comunicações IrDA (*Infrared Data Association*);

Como as UART's são saídas directas dos pinos do microcontrolador, logo existe a necessidade de efectuar a conversão dos níveis lógicos deste para os níveis RS-232.

ETHERNET

A *Fez-Cobra* disponibiliza um conector RJ45 para ligação desta em rede Ethernet, cujo controlador está inserido no LPC2478. Este bloco está dotado tecnologicamente para se interligar externamente com dispositivos controladores de 10 ou 100 Mbps (10Base-T, 100Base-TX, 100Base-FX e 100Base-T4. Totalmente compatível com o norma IEEE 802.3 e com o IEEE 802.3x (*full duplex* e controlo de fluxo), suporte de tramas VLAN, gestão da memória com *buffer* de entrada e saída independentes, filtro de recepção são algumas das características deste controlador.

Em complemento à rede Ethernet a *Fez-Cobra* permite implementar: o protocolo TCP/IP (*Transmission Control Protocol / Internet Protocol*) através de *sockets*, sendo o número máximo de 128 *sockets* que se podem estabelecer simultaneamente; estabelecimento de ligação ponto a ponto, ou seja, PPP (*Point-to-Point Protocol*) permitindo o uso de redes 3G/GPRS; SSL (*Secure*

Socket Layer); *Wireless* LAN (WiFi, IEEE 802.11b); HTTP.(*Hipertext Transfer Protocol*). Além disto, existe a possibilidade de alteração do MAC (*Medium Acess Control*) por *software* e suporte de endereçamento IP estático ou dinâmico (DHCP – *Dynamic Host Configuration Protocol*) [GHI11].

CAN

Para dar suporte a comunicações CAN a *Fez-Cobra* está provida de dois canais cujo, controlador encontra-se de acordo com as especificações CAN versão 2.0B. A estrutura CAN compreende dois blocos: um é o controlador e outro é o filtro de aceitação. Contempla um *buffer* duplo na recepção e um *buffer* triplo na transmissão, recebendo as próprias mensagens. Todos os registos e acessos à RAM são em palavras de 32 bits. Permite o recurso a identificadores de 11 bits ou 29 bits e taxas de transmissão programáveis, até 1Mbps.

USB

Neste tipo de comunicação utiliza a norma USB 2.0 *full-speed* (12Mbps) que através de dois conectores permite dispor de uma ligação *host* e *device*. Um conector normalizado tipo A é utilizado para a ligação *host* e, um conector mini-A estabelece a ligação *device*. Esta última serviu para efectuar o *debug* ao programa assim como alimentar a *Fez-Cobra*.

I2C

A *Fez-Cobra* disponibiliza um barramento para comunicações I2C, provido de resistências de *pull-up* nas duas linhas do barramento, permitindo desta forma a interligação imediata com qualquer dispositivo. A Figura 42 é ilustrativa desta comunicação.

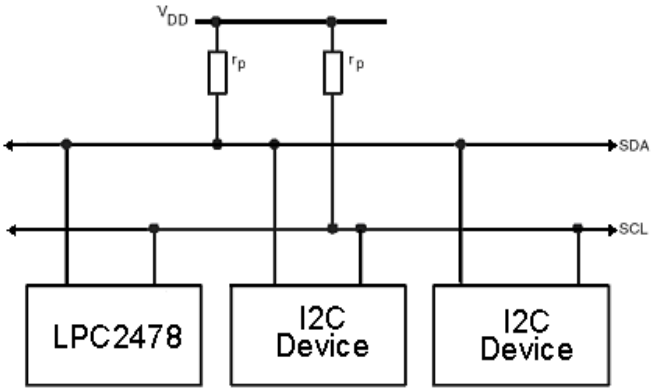


Figura 42 – Comunicação I2C.

Sistema originalmente desenvolvido pela *Philips Semiconductors* (actualmente *NXP Semiconductors*), para fácil e rápido estabelecimento de comunicações bidireccionais, entre componentes da mesma placa de circuito impresso, requerendo apenas duas linhas para composição do barramento. Uma linha é o relógio, designada por SCL (*Serial Clock Line*) e outra é para os dados, designada por SDA (*Serial Data Line*). A transmissão série de 8 bits pode compreender transferências de dados a taxas de: 100 kbps (*Standard* mode), 400 kbps (*Fast mode*), 1Mbps (*Fast mode Plus*) ou 3,4 Mbps (*High-speed mode*) [NPX07].

Cada dispositivo colocado no barramento é endereçável por *software* correspondendo a um único endereço, tornando uma relação entre os dispositivos do tipo *master/slave*. Desta forma o I2C é um sistema *multi-master* possuindo arbitragem no acesso e detecção de colisão no barramento.

SPI

São duas as interfaces (SPI0 e SPI1) para utilizar esta comunicação de dados série, síncrona e em modo *full duplex*, entre dispositivos periféricos. Na *Fez-Cobra*, por imposição do NETMF, a transferência de dados apenas pode acontecer com 8 ou 16 bits e a sua taxa de transmissão máxima tem um valor de 1/8 da frequência do relógio

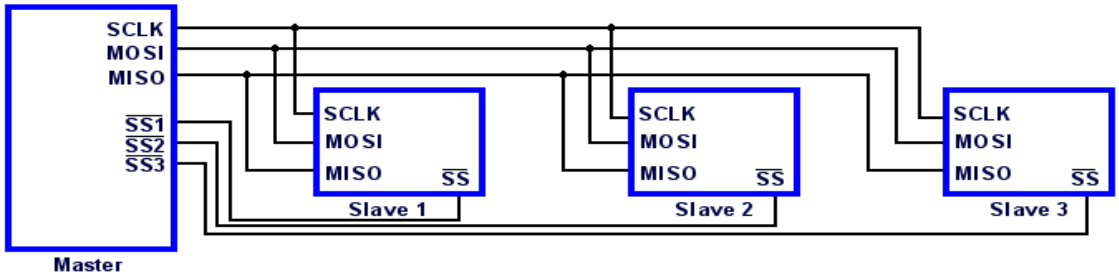


Figura 43 – Comunicação SPI.

Originalmente desenvolvido pela Motorola, a comunicação SPI assenta numa relação *Master/Slave*, onde pode coexistir um *master* e, um ou mais *slave* (é possível constituir um sistema com mais que um *master*, mas só pode existir um em cada momento). É composto por um barramento de quatro linhas sendo elas:

- MOSI – *Master Out / Slave In*;
- MISO – *Master In / Slave Out*;
- SCLK – *Serial Clock*;
- SS – *Slave Select* (ou CS – *Chip Select*).

A linha SS vai ter tantas ligações quanto o número de dispositivos colocados no barramento (ver exemplo apresentado na Figura 43). Apenas um único *master* e *slave* podem estabelecer comunicação para troca de dados, sendo sempre o *master* a iniciar comunicação e o *slave* responde sempre com um byte. Com transferência de dados de comprimento variável entre 8 e 16 bits pode compreender diferentes velocidades de transmissão (dependentes do *clock* do sistema) [Freescale02].

1–*WIRE*

Qualquer pino digital de I/O pode desempenhar funções de comunicação no formato *1–Wire*, tal como no exemplo apresentado na Figura 44. Isto é possível devido ao NETMF ter desenvolvido uma biblioteca (classe), a qual pode ser chamada para actuar num pino qualquer de I/O.

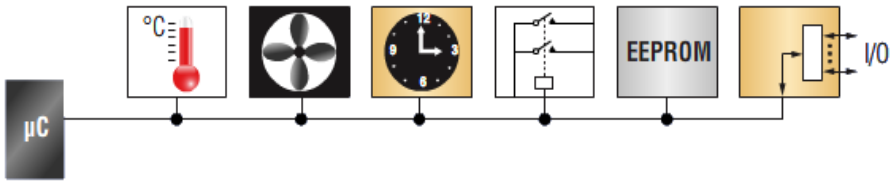


Figura 44 – Comunicação *1-Wire* [Maxim11].

Este sistema foi desenvolvido pela *Dallas Semicondutor* (actualmente *Maxim*), que permite obter transmissão de dados série através de uma única linha *bidireccional* e em modo *half-duplex*. Possui um único *master* com a possibilidade de ligar múltiplos *slaves* a uma baixa taxa de transmissão (15,4kbps – *standard* ou 125kbps – *overdrive*) e limitada na distância. Tem como aplicação dispositivos electrónicos (sensores de temperatura, temporizadores, memórias, etc.) sendo estes possuidores de um único ID (*Identifier*). Deste modo o *master* reconhece os dispositivos que tem ligado a si, sendo a comunicação iniciada por este [Maxim11].

4.2.1.6. CONVERSOR A/D E D/A

Os conversores de sinal analógico/digital e digital/analógico (D/A) encontram-se inseridos no interior do microcontrolador. São ambos de 10 bits e possuem a mesma tensão de referência externa, sendo neste caso de +3,3V. Em seguida são apresentadas algumas características dos conversores:

- Analógico /Digital:
  - São 7 os canais disponíveis na placa (multiplexados internamente), mas 2 deles estão direccionados para o uso do *display* táctil;
  - A arquitectura utilizada pelo conversor A/D é de aproximações sucessivas;
  - Tempo de conversão mínimo de 2,44µs, no microcontrolador.
- Digital/Analógico:
  - Apenas existe uma única saída analógica (o pino é partilhado pela entrada analógica 3);
  - Arquitectura *resistor string*;

- Settling time: máximo de 1μs (Ibias = 700μA) ou 2,5μs (Ibias = 350μA), no microcontrolador.

4.2.1.7. PWM

O funcionamento do PWM é obtido por característica do processador, ou seja, é constituído por *hardware*. Ao nível do *hardware* existem dois PWM's, gerando seis sinais de saída, aplicados em diferentes pinos. Assim, os seis sinais de PWM não partilham o mesmo temporizador (*timer*), pois apenas existem dois *timers*. Daqui resulta:

- PWM0 e PWM2 partilham um *timer*;
- PWM1, PWM3, PWM4 e PWM5 partilham outro *timer*.

4.2.1.8. INTERFACE SD/MMC

Este interface permite o armazenamento de dados em cartões de memória SD/MMC, utilizando velocidades elevadas de 25MHz para SD e 20MHz para MMC. Igualmente, existe diferença entre o número de bit na transferência, sendo de 4bits quando é cartão SD e de 1 bit quando é cartão MMC.

4.2.1.9. *DISPLAY* TÁCTIL

O módulo EMX é caracterizado por suportar um *display* TFT de 16 bits de cor. A sua resolução é de 320x640 num *display* de 3,5'' podendo atingir os 800x600, sendo passíveis de serem visualizadas imagens do tipo WPF, BMP, GIF e JPG. A parte táctil é assegurada por quatro linhas resistivas permitindo desdobramento em *Yup*, *Ydown*, *Rleft* e *Rright*.

4.2.2. .NET MICRO FRAMEWORK

4.2.2.1. INTRODUÇÃO

A ferramenta *.Net*, da Microsoft, é uma tecnologia que surgiu com o intuito potencializar a produtividade no desenvolvimento de aplicações para computadores e servidores. Nesta mesma linha surge o *.Net Micro Framework*, como uma pequena e eficiente plataforma do *.NET Framework*, para ser uma mais valia no desenvolvimento de aplicações em pequenos dispositivos (sistemas embebidos), tendo em conta os seus recursos limitados de *hardware* não necessitando de sistema operativo [Thompson07] [Kuhner09] [Miles07].

Através do *Visual Studio* (no projecto foi utilizado o *Visual Studio 2010 Express Edition*) e do *.Net Micro Framework 4.0* é disponibilizado um conjunto de ferramentas, entre as quais se incluem o suporte no desenvolvimento de aplicações em C# e o *debugging* por simulação (*software*) ou no próprio dispositivo (*hardware*) [GHI10b].

Com o NETMF o acesso ao *hardware* é obtido através da construção de livrarias e das suas respectivas classes, passando este a ser tratado como um objecto. Assim, o *hardware* passa para um nível abstracto, onde o acesso ao mesmo é garantido pelo recurso a objectos orientados na fase de desenvolvimento do programa. Deste modo, apenas é necessário configurar as propriedades de um determinado objecto, deixando de existir a necessidade de configurar o *hardware* (registos, bits, etc.). Este formato permite que determinada aplicação seja independente da plataforma onde é aplicada [Freescale09] [Miles07] [Thompson07].

Com novos processadores de 32 bits a tornarem-se mais competitivos devido a serem menos onerosos, com menores consumos energéticos, a superarem performances de microcontroladores, velocidade de funcionamento mais rápidas e, por outro lado, as memórias a aumentarem de capacidade e a reduzirem tamanho e preço, permitiu incorporar o *managed code* nos sistemas embebidos [Freescale09] [Miles07].

O NETMF a correr directamente sobre o *hardware* requer menos 100kB de RAM. O exemplo surge da placa utilizada neste projecto, cujo processador é um ARM7 a 72MHZ, com 96kB de RAM e 512kB de memória *flash*. Como termo de comparação, uma aplicação do *.NET Compact Framework* requer no mínimo 12MB, aplicação num dispositivo do tipo telemóvel [Kuhner09].

4.2.2.2. ARQUITECTURA

O NETMF encontra-se disponível para uma diversidade de plataformas de *hardware* devido a possuir uma arquitectura flexível e adaptável. A sua arquitectura está representada na Figura 45.

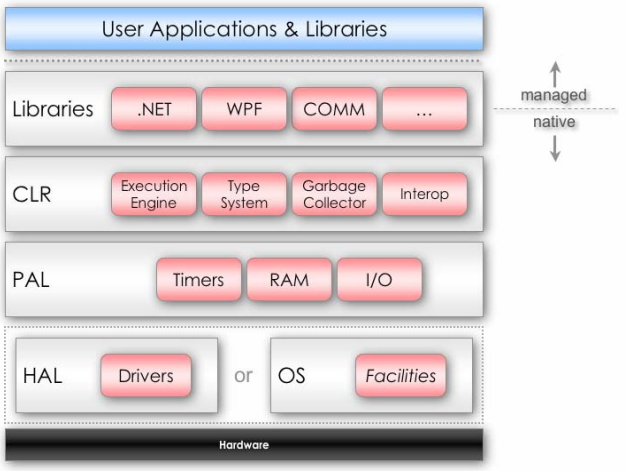


Figura 45 – Arquitectura do NETMF [Thompson07].

HARDWARE LAYER

Esta é a camada que possui o processador e os circuitos auxiliares e/ou complementares, que constituem a plataforma do *hardware* do dispositivo.

RUNTIME COMPONENT LAYER

- CLR

A CLR é o "motor" de execução da plataforma *.Net*, o qual manuseia o código durante o tempo de execução. No caso do NETMF versão reduzida da CLR do

.NET é adaptada e aplicada ao *hardware* do dispositivo em uso [Thompson07]. Este pequeno trecho de *software*, designado por TinyCLR no NETMF, é responsável pela análise e execução da assemblagem do código, além da gestão dos recursos do sistema [Kuhner09]. No caso da *Fez-Cobra* este *software* é aplicado no módulo EMX.

A compilação do código fonte dá origem ao código em linguagem intermédia, ou seja, *Intermediate Language* (IL), a qual também pode ser designado por *Microsoft Intermediate Language* (MSIL), sendo carregado para o CLR. Este código é um meio-termo entre o código fonte e o de código máquina, ou seja, foi compilado mas não no código máquina (código nativo).

O código que vai ficar sujeito à gestão e execução por parte do CLR, é designado por *managed code*. Ao código que não é afectado pelo CLR, ou seja, não depende da CLR é designado como sendo código nativo (ou *unmanaged code*).

Ao suportar o *managed code* o CLR passa a estar incumbido de compilar este código em código nativo e efectuar a sua execução. Como características inerentes a este processo há a salientar as seguintes [Kuhner09]:

- Gestão da memória recorrendo ao do *garbage collector*, permitindo deste modo a libertação da memória não utilizada mas retida;
- Gestão e sincronização de *threads*, designadamente partilhando tempos (*time slice*) para os threads e métodos de sincronização, para assegurar o acesso a recursos partilhados;
- Referências aos objectos são tratadas pelo CLR, uma vez que a execução do *managed code* encontra-se sobre a sua alçada, deixando de existir a preocupação dos apontadores inseguros;
- Reforça a segurança no tipo de dados, impedindo execução de código nativo não seguro e não pertencente ao sistema;
- Redução nas possibilidades de ocorrência de erros, recorrendo ao processo de *debugging*.

- PAL

O objectivo da PAL (*Platform Abstraction Layer*) é estabelecer uma ligação entre o CLR e o HAL, através de um conjunto de serviços disponibilizados por *software*, sempre que o CLR necessitar de funcionalidades do *hardware*. As funcionalidades do PAL são independentes do *hardware* [Freescale09]. Assim o PAL inclui serviços como a gestão de memória, temporizadores, *debugging*, eventos, chamadas a procedimentos assíncronos.

- HAL

Esta camada, designada por *Hardware Abstraction Layer* (HAL), é composta por um conjunto de primitivas que disponibilizam o acesso ao *hardware* [Freescale09]. Quando não existe sistema operativo é através desta camada que é possível estabelecer uma estrutura de arranque do *hardware*. Assim, o *bootstrap*[7] contém o código que permite iniciar o *hardware* do sistema (nível baixo), quando o dispositivo é ligado. Posteriormente o CLR inicia o sistema ao nível mais elevado. O código de *bootstrap* não tem qualquer interacção com os níveis superiores da sua arquitectura, antes do começo do CLR.

CLASS LIBRARY LAYER

Nesta camada estão localizadas as livrarias de apoio à plataforma *.NET*, neste caso ao NETMF. Esta camada proporciona todas as funcionalidades do NETMF, incluindo as livrarias de interface gráfica, rede (*networking*), comunicações, encriptação, etc.

APPLICATION LAYER

É a camada superior, onde se encontra a aplicação desenvolvida para ser aplicável aos dispositivos que compõem o *hardware*. A aplicação neste projecto foi desenvolvida em C#.

[7] Processo de inicialização do CPU e do seu hardware (periféricos).

4.3. PROTOCOLOS DE COMUNICAÇÃO

4.3.1. CAN

4.3.1.1. INTRODUÇÃO

Com a necessidade de diminuir as cablagens nos sistemas eléctricos automóveis e das ligações ponto a ponto de uma rede de comunicações tradicional, em virtude do elevado número de dispositivos electrónicos (Figura 46), a empresa *Robert Bosch* desenvolveu em meados dos anos 80 um sistema de interligação de dispositivos através de uma rede de comunicações [Dominique07] [Göhner06].

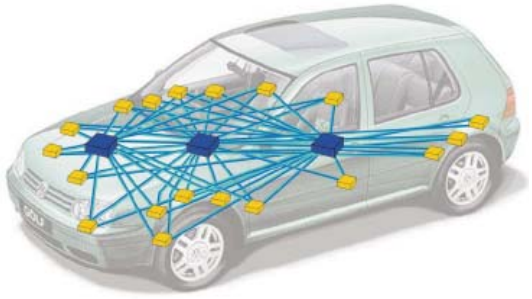


Figura 46 – Automóvel com três unidades de controlo e ligações ponto a ponto aos dispositivos [Volkswagen01].

O aparecimento desta rede de comunicação que utiliza a ligação dos sistemas de controlo (controladores, actuadores e sensores) a um único barramento, designado de CAN. O protocolo de comunicação CAN foi apresentado no congresso da *Society Automotive Engineers* (SAE), em 1986 pela Bosch, tornando-se um sistema aberto à indústria automóvel [Dominique07] [Göhner06] [Herald11].

Assim, os princípios que norteiam esta rede advêm de requisitos temporais para as comunicações dos dispositivos electrónicos, ou seja, transferência de dados rápidas e fiáveis. Isto significa o estabelecimento de um elevado número de comunicações, entre os diversos dispositivos inteligentes, mas com uma

pequena troca de dados. Desta forma as principais características que se pretende obter são [Bosh91] [Göhner06]:

- Minimizar a cablagem;
- Sistema robusto quanto aos erros;
- Tempos de latência baixos;
- Ligação de vários dispositivos receptores distribuídos ao longo do barramento;
- Boa expansão;
- Prioridade nas mensagens;
- Baixo custo.

Com algumas empresas de topo na área tecnológica a mostrarem interesse, desde logo se verificou que as potencialidades e aplicabilidades da rede CAN iam muito para além dos sistemas automóveis. A indústria é um sector que emprega bastante esta tipologia de rede, pois são propícias a funcionarem em ambientes hostis à transmissão de dados (ruído, variações de alimentação são exemplos que evidenciam esta problemática). Actualmente, desde os pequenos electrodomésticos até às grandes unidades de produção industrial, o CAN é um modelo de rede utilizável. Hoje em dia todos os automóveis fabricados trazem implementada, pelo menos uma rede CAN, demonstrada na Figura 47.

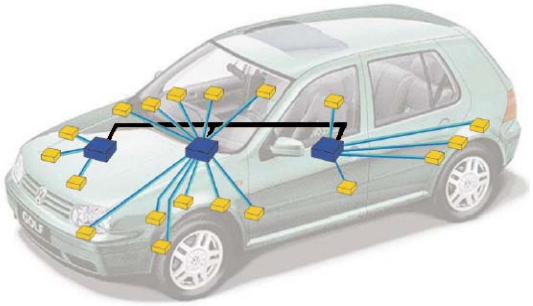


Figura 47 – Automóvel com três unidades de controlo e diversos dispositivos num único barramento CAN [Volkswagen01].

A especificação de protocolo CAN de Robert Bosch, na sua versão 2.0, comporta duas versões: - A (original) e a B. A versão 2.0A, apresenta 11 bits para identificação das mensagem, enquanto a versão 2.0B utiliza 29 bits para este campo, sendo esta a diferenciação entre as versões [Bosch91].

Estas especificações foram submetidas para norma internacional no início da década de 90, tendo-se publicado em Novembro de 1993 a norma ISO (*Internacional Organization for Standardization*) 11898 referente às especificações do Protocolo CAN. Esta norma estabelece a arquitectura da rede CAN referente as camadas do modelo OSI (*Open System Interconnection*), especificando as características das camadas: – física e de ligação; utilizadas por este protocolo. A versão ISO 11898-2 contém especificações sobre a rede CAN – *High Speed* (até 1Mbps), enquanto a versão ISO 11898-3 estabelece a características da CAN – *Low Speed* (até aos 125kbps) [Bosch11] [CiA11] [Keith99].

Naturalmente a rede CAN tem evoluído, surgindo desta forma variações, designadamente novos protocolos de nível superior que correm sobre o CAN. Exemplos desta evolução são [CiA11]:

- CANOpen e CAL, especificações da CiA (*CAN in Automation*);
- Can Kingdom – especificações KVASER;
- DeviceNet – especificações Allen Brandley – Rockwell;
- SDS (Smart Distributed Systems) – especificações Honeywell;
- OSEK/VDX – especificações OSEK;
- J 1939 – especificações SAE.

Deste leque é fácil constatar que a rede CAN é bastante mais abrangente que simplesmente a industria automóvel.

4.3.1.2. ESTRUTURA

O protocolo CAN é baseado num sistema *multimaster*, onde diversos nós podem tentar a transmissão em qualquer ponto da rede e ao mesmo tempo (o acesso ao meio garante a exclusividade para um dos nós). A informação transmitida é orientada à mensagem, pelo que não há definição dos nós nem dos endereços, apenas à mensagem. Estas mensagens são identificadas pelo uso de um identificador, o qual será único em toda a rede, assim como a definição de prioridade que permite a atribuição do barramento. Em função do

identificador da mensagem enviada para a rede, e consequentemente para todos os nós, cabe a cada nó decidir sobre a sua utilidade ou não [CiA11].

Esta característica da informação transmitida com orientação à mensagem, permite gerir uma rede com um nível de flexibilidade bastante elevado, pois torna-se simples efectuar alterações físicas. O acrescentar ou retirar nós não carece de modificações no sistema, ou seja, em termos de *hardware* e *software* as condições mantêm-se inalteráveis, permitindo o estabelecimento de velocidades de transmissão até 1Mbps.

Tal como já apresentado anteriormente, o protocolo da rede CAN é um *standard* ISO para comunicações série. Desta forma, a estrutura do protocolo CAN encontra-se definido no modelo OSI (Figura 48), pelo recurso às duas camadas inferiores: - camada 1 e 2, ou seja, a camada física (*physical layer*) e a camada de ligação de dados (*data link layer*), respectivamente.

Figura 48 – Camadas do modelo OSI aplicáveis no CAN.

Enquanto as camadas agora referidas têm normas para aplicação da rede CAN, as camadas superiores não as têm. O recurso a camadas de nível superior, designadamente a camada aplicação – nível 7 (*application layer*), não está normalizado para o estabelecimento de ligação entre os níveis inferiores. Sem bases normativas, ficam condicionadas ao desenvolvimento de aplicações com especificações definidas pelo fabricante ou grupos de trabalho, como é o caso do grupo de fabricantes que aderiram ao CiA.

4.3.1.3. CAMADA FÍSICA

Do modelo OSI referente à camada física, esta incorpora os aspectos da ligação física entre os nós, assim como os sinais exigíveis para a transmissão. Desta forma a camada física pode ser dividida em três subcamadas, sendo [Dominique07] [Steve08]:

- *Physical signaling* (PLS) – implementada no controlador CAN
  - Codificação/descodificação de bits
  - *Timing* dos bits
  - Sincronização entre nós
- *Physical Medium Attachment* (PMA)
  - Características do *transceiver*
- *Medium Dependent Interface* (MDI)
  - Características do meio de transmissão (cabo, fichas, etc.)

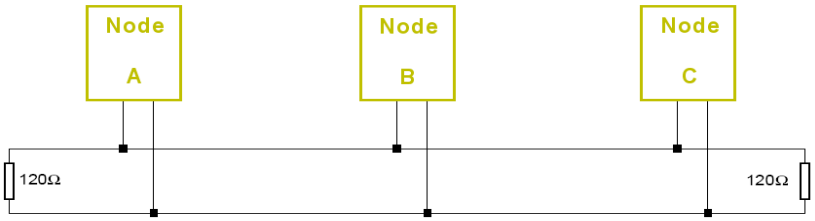


Figura 49 – Estrutura do barramento CAN [Steve08].

Na Figura 49 está representado o meio físico utilizado nas comunicações CAN. É definido por um barramento com dois fios, terminando em ambas as extremidades com a ligação dos dois condutores, através de uma resistência de 120Ω (terminador). O comprimento máximo da rede (barramento) está dependente da tipologia do cabo assim como da taxa de transmissão empregue.

A Figura 50 apresenta a relação entre estas duas variáveis: comprimento e taxa de transmissão. Através desta figura pode-se observar que para uma taxa de 1Mbps (máxima), utilizando um cabo entrelaçado, (par de fios), esta situa-se em redor dos 40m.

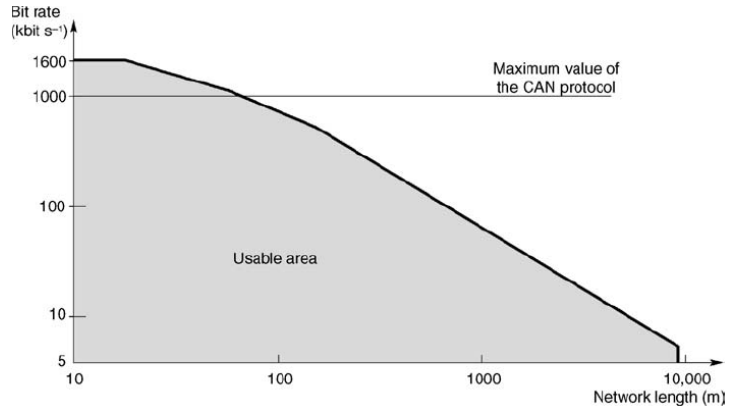


Figura 50 – Relação entre o comprimento do barramento e a taxa de transmissão [Dominique07].

BIT DOMINANTE/RECESSIVO

No barramento CAN, por especificações da norma, são introduzidos os conceitos de bit dominante e bit recessivo. Esta definição é o resultado da tensão que vai surgir entre as duas linhas do barramento, sendo que uma linha é possuidora da uma tensão fixa mais elevada que a outra. As linhas do barramento são designadas de CAN_H e de CAN_L. A designação usualmente atribuída aos bits de nível 0 ou nível 1 deixa de fazer sentido numa rede CAN, pois aqui correspondem aos bits dominantes ou aos bits recessivos, respectivamente.

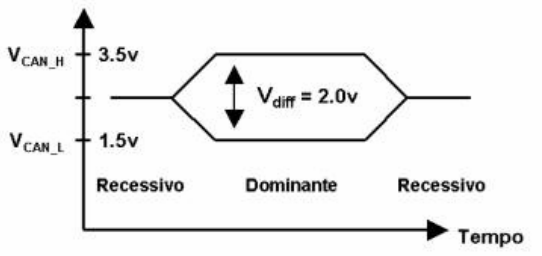


Figura 51 – Níveis lógicos presentes numa rede CAN.

Para se obter os estados dos bits, o barramento utiliza o mecanismo designado por *wired-AND*, onde os bits dominantes são equivalentes a um nível lógico 0 e, sobrepõem-se aos bits recessivos, equivalentes ao nível lógico 1 [CiA11]. A Figura 51 ilustra os níveis de tensão numa rede CAN, assim como os bits dominantes e recessivos.

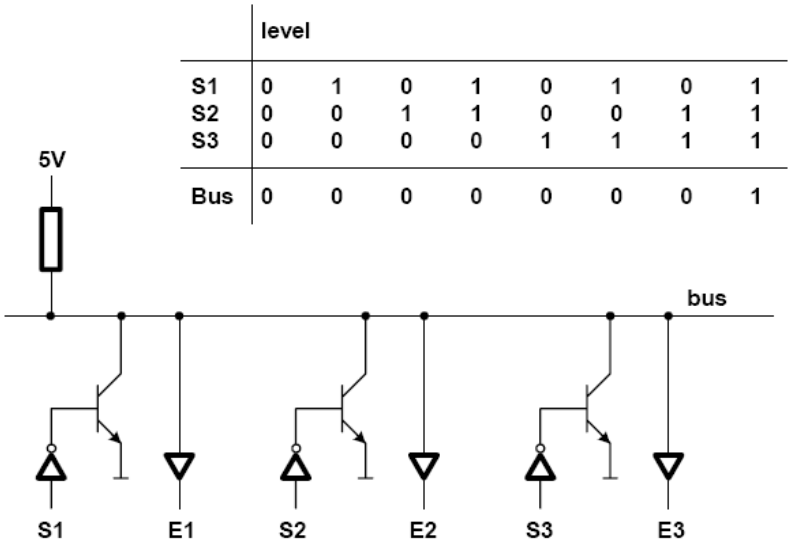


Figura 52 – Ligação do tipo *wired-AND* e a respectiva tabela de verdade [Göhner06].

No barramento CAN, para a existência de um bit recessivo (“1”), todos os nós têm de se encontrar a transmitir na mesma altura esse bit. Doutro modo, existindo pelo menos um nó transmissor que emita “0”, vai gerar o bit dominante. Esta topologia é designada por *wired-AND*, uma vez que o “0” (dominante) prevalece face ao “1” (recessivo), encontrando-se representada na Figura 52.

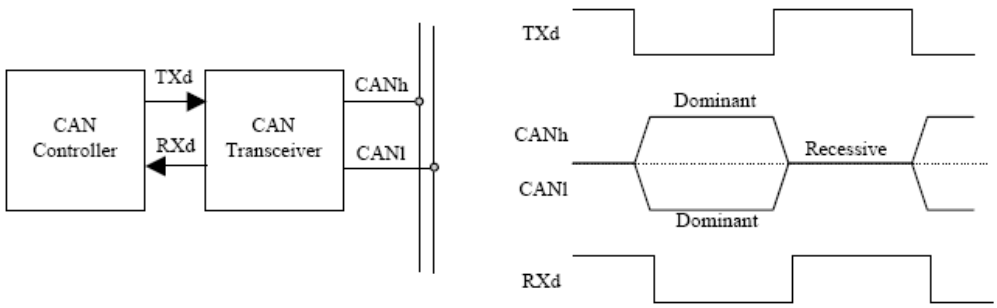


Figura 53 – Interligação entre controlador, *transceiver* e barramento CAN [Atmel04].

Os dados enviados através da rede são interpretados pela análise da diferença de potencial entre os fios CAN_H e CAN_L, cuja análise pode ser observada na Figura 53. Por isso, o barramento CAN é classificado como par de fios entrançado diferencial. Este conceito atenua fortemente os efeitos causados por interferências electromagnéticas, uma vez que qualquer acção sobre um dos fios será sentida também pelo outro, causando flutuação em ambos os sinais para o mesmo sentido e com a mesma intensidade. Como o que vale

para os módulos que recebem as mensagens é a diferença de potencial entre os condutores CAN_H e CAN_L (e esta permanecerá inalterada), a comunicação não é prejudicada.

CODIFICAÇÃO DO BIT – BIT STUFFING

O CAN utiliza uma codificação NRZ – *Non Return to Zero*, que permite identificar o "1" como um sinal de nível alto e o "0" como um sinal de nível baixo. No entanto, para situações em que os bits se encontrem sucessivamente no mesmo estado, a norma CAN adoptou uma técnica designada por bit *stuffing*, representada na Figura 54.

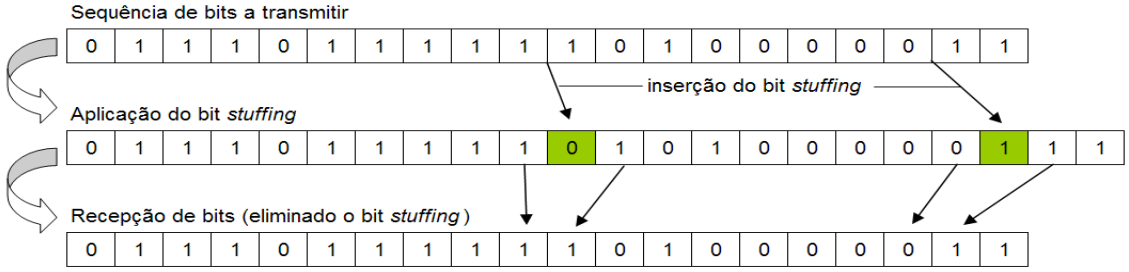


Figura 54 – Processo bit *stuffing*.

O bit *stuffing* no protocolo CAN consiste em cada cinco bits consecutivos de um mesmo valor lógico, colocar um bit de sinal oposto. Esta colocação é realizada pelo nó transmissor e incluída na codificação, pelo que os nós receptores podem eliminar estes bits e assim apresentar a mensagem correctamente. Desta forma o bit *stuffing* assegura a sincronização da rede, uma vez que se passa a ter transições ascendentes.

Outra vertente desta técnica é o reconhecimento por parte de todos os nós da presença de um erro. Se uma sequência de seis bits dominantes consecutivos surgir numa trama de dados, esta encontra-se a violar a regra de bit *stuffing*. Esta violação da regra origina o aparecimento de uma trama de erro gerada pelos próprios nós.

TEMPO DO BIT

Um dos requisitos para o bom funcionamento do protocolo CAN é a existência de tempos de bit condizentes nos diferentes nós. Como garante deste requisito os controladores CAN devem compreender uma gestão dos tempos no barramento, sobretudo devido à necessidade de sincronização dos nós presentes na rede, compensação dos atrasos na propagação do sinal e definição do ponto de amostragem.

Com efeito é definido por *bit time* (tempo do bit) o período de tempo efectivo no qual o bit está presente no barramento. Para que exista coerência nos diferentes nós, cujos controladores podem funcionar a diferentes frequências internas de relógio, surge o *nominal bit time* (tempo nominal do bit). Este tempo apenas pode ser considerado como ideal e teórico, pois devido à sua natureza variável, não é possível o estabelecer um valor fixo. O inverso do *nominal bit time* é apelidado de *nominal bit rate* (taxa de transferência de bits), representando o número de bits transmitidos por segundo, ou seja, o *baudrate* pretendido na transmissão [Robb04].

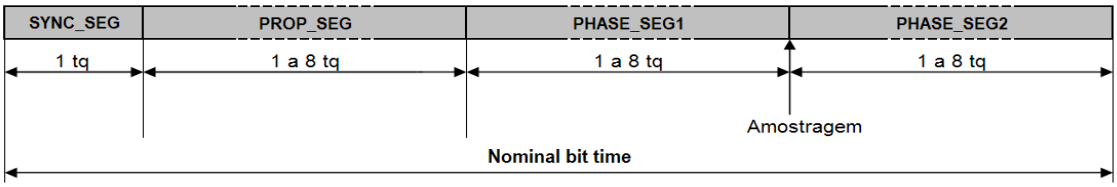


Figura 55 – Segmentos do bit (*nominal bit time*).

Estabelecido nas especificações, o *nominal bit time*, é constituído por quatro segmentos não sobrepostos. Cada um destes segmentos é construído por um múltiplo inteiro de um *time Quantum* (tq). Este tempo é definido como a porção de tempo discreta mais pequena usada por um nó. O seu comprimento é gerado por um divisor programável oriundo da frequência de oscilação dos nós CAN, ou seja, é o controlador CAN que fica encarregue de gerar o número de *time Quantum* nos diferentes segmentos. Para bit *time* é possível obter um valor entre os 8 e 25 vezes do *time Quantum*, e como a consequência desta

variação, é possível tornar um novo valor do *nominal bit time*, resultando num *baudrate* configurável.

Os quatro segmentos que representam o bit encontram-se na Figura 55, sendo os seguintes [Bosh91] [Robb04]:

- SYNC_SEG – é usado para sincronizar os vários nós existentes no barramento. Tem o comprimento unitário cujo valor é equivalente a um *time Quantum*;
- PROP_SEG – esta parte do segmento é usada para compensar os atrasos temporais, devidas às características físicas da rede. O seu comprimento é variável, podendo atingir um valor máximo de 8 *time Quantum*;
- PHASE_SEG1, PHASE_SEG2 – são segmentos utilizados para compensarem erros, variações ou alterações na fase e/ou posicionamento da linha de sinal. Estes segmentos servem para aumentar ou diminuir o seu comprimento através do mecanismo da *re-synchronization*. O segmento PHASE_SEG1 tem um comprimento máximo 8 *time Quantum* e o PHASE_SEG2 é o máximo do segmento PHASE_SEG1 acrescido do ponto de amostragem.

O tempo decorrido por estes 4 segmentos corresponde ao *nominal bit time*, sendo conveniente salientar mais um aspecto deste tempo – o ponto de amostragem. Este ponto situa-se no final do segmento PHASE_SEG1 e estabelece um tempo de processamento da informação, cujo valor é menor ou igual a 2 *time quantum*.

SINCRONIZAÇÃO

Manter o sincronismo entre os intervenientes na rede CAN obriga à existência de processos para tal, surgindo duas tipologias [Atmel04] [Bosh91]:

- *Re-synchronization* – permite efectuar o sincronismo no acontecimento de uma transição, a qual pode ser no seguimento da

transmissão de um bit ou de um conjunto de bits (recurso à técnica do bit *stuffing*). A Figura 56 ilustra este processo;

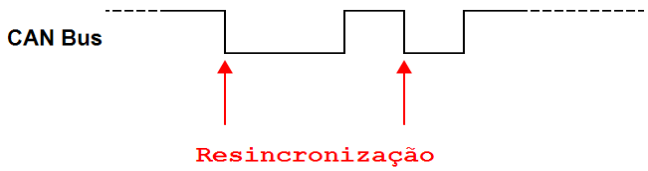


Figura 56 – *Re-synchronization*.

- *Hard-synchronization* – no início da transmissão de uma trama é feito o sincronismo de toda a rede (nós). A Figura 57 ilustra este processo;

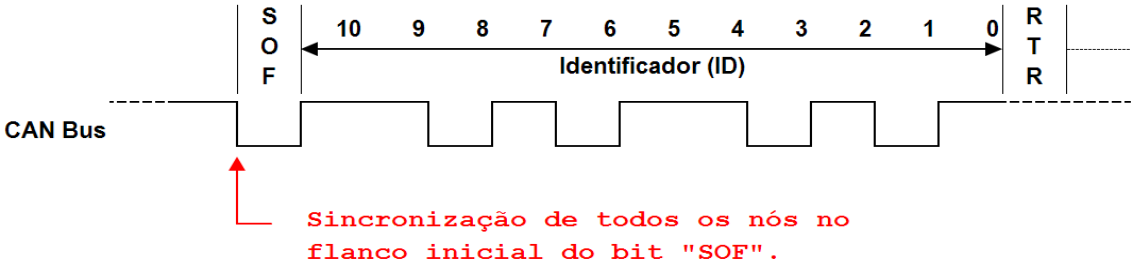


Figura 57 – *Hard-synchronization*.

4.3.1.4. CAMADA LIGAÇÃO DE DADOS

A camada ligação, genericamente, estabelece o formato das mensagens que são enviadas para o barramento, dispondo de meios para a detecção e sinalização de erros. Esta camada encontra-se dividida em duas subcamadas [Dominique07]:

- MAC – *Media Access Control*, que representa o núcleo do protocolo CAN. Tem como função a recepção das mensagens vinda da subcamada LLC e apresentar mensagens para a transmissão à mesma camada (LLC) É responsável pelas tramas de dados, pelo campo de arbitragem e confirmação, detecção e sinalização de erros;
- LLC – *Logic Link Control*, tem como objectivo a filtragem das mensagens (como as mensagens são definidas por um identificador, ela não é dirigida especificamente a um nó, ou seja,

todos os nós recebem a mensagem simultaneamente e é aqui que ocorre essa identificação/filtragem da mensagem que seja relevante), notificação de sobrecarga e procedimentos para recuperação de erros.

ACESSO AO MEIO

O protocolo CAN aplica o sistema CSMA/CA – *Carrier Sense Multiple Acess/Collision Avoidance* na gestão dos acessos ao barramento, com arbitragem na prioridade da mensagem. Este conceito de arbitragem evita colisões em mensagens cujos nós iniciaram simultaneamente a transmissão e certifica-se de que a mensagem mais importante é transmitida em primeiro lugar sem perdas significativas de tempo. Assim, cada nó verifica, primeiramente, o estado em que se encontra o barramento (se este se encontra sem actividade - *idle*) por um período de tempo (CS – *Carrier Sense*). A partir deste tempo todos os nós têm oportunidade igual de acesso ao barramento (MA – *Multiple Acess*). Para evitar a colisão (CA – *Collision Avoidence*) no barramento, com o envio simultâneo de dados de mais que um nó, é usada a arbitragem do bit. Esta arbitragem baseia-se num processo de exclusão do acesso ao meio por parte de nós, cuja mensagem seja de prioridade inferior, designado o processo de *bitwise* [Dominique07] [Keith99]. Este processo encontra-se representado na Figura 58.

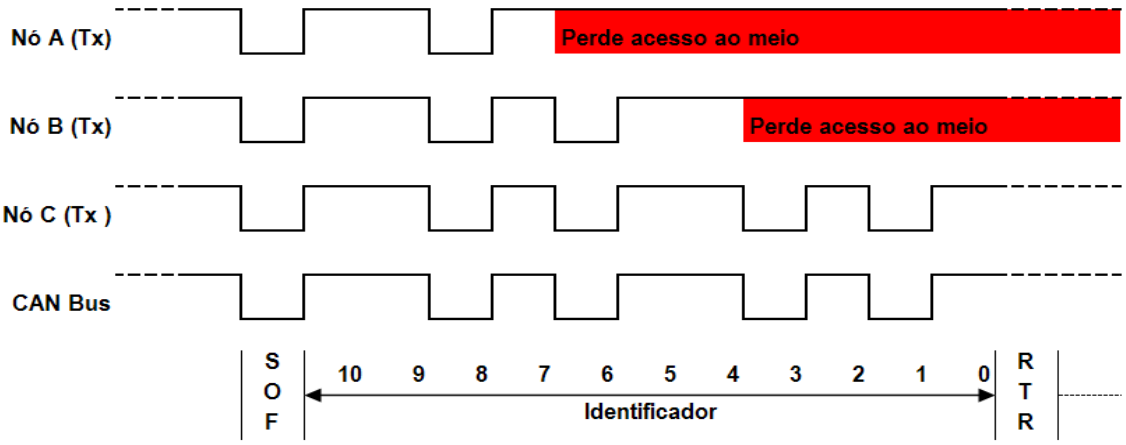


Figura 58 – Acesso ao barramento e sua arbitragem.

Assim, *bitwise* consiste em que cada nó envia os bits do identificador e fica à escuta dos níveis do barramento [CiA11]. O suporte para esta arbitragem assenta na definição de um estado lógico como dominante ou recessivo e, um nó transmissor deve verificar o barramento para garantir que o estado lógico que vai transmitir aparece no barramento. Daqui surge o conceito de bit dominante e bit recessivo, em que quem envia um bit dominante tem prioridade sobre o bit recessivo [Keith99].

FILTRAGEM DAS MENSAGENS

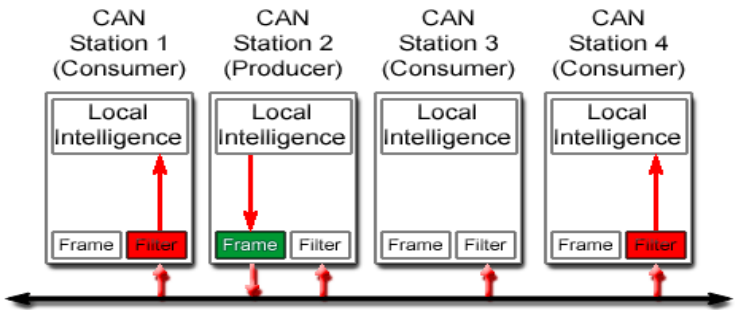


Figura 59 – Aplicação de filtro às mensagens CAN [CiA11].

Ao nível do protocolo CAN não se encontram estabelecidas especificações para o processo de filtragem de mensagens, no entanto, os controladores permitem o recurso a um filtro de aceitação de mensagens. Como as mensagens são possuidoras de um identificador, ela não dirigida especificamente para um nó, ou seja, todos os nós recebem a mensagem. Esta mensagem é então recolhida e tratada por todos os nós onde existe o interesse na recepção da mensagem (na Figura 59 é a estação 1 e a estação 4 que aceitam a mensagem). Nos restantes nós onde a mensagem não tem interesse, isto é, o identificador não é reconhecido, esta mensagem é desprezada (tal como acontece na estação 3 e 2 da Figura 59, sendo que o próprio emissor recebe a mensagem e rejeita-a).

FORMATO DAS TRAMAS

Existem quatro tipos de tramas no protocolo CAN, sendo elas [Bosh91] [Steve08]:

- Trama de Dados (*Data Frames*) – efectua o transporte de dados;

- Tramas Remotas (*Remote Frames*) – pede a transmissão de uma trama de dados com o mesmo identificador;
- Trama de Erros (*Error Frames*) – é transmitida por um nó que detecte um erro na rede;
- Tramas de Sobrecargas (*Overload Frames*) – um nó só pode utilizar esta trama para atrasar a transmissão de tramas de dados ou remotas, por parte de outro nó, quando não está pronto a recebê-los.

➢ TRAMA DE DADOS

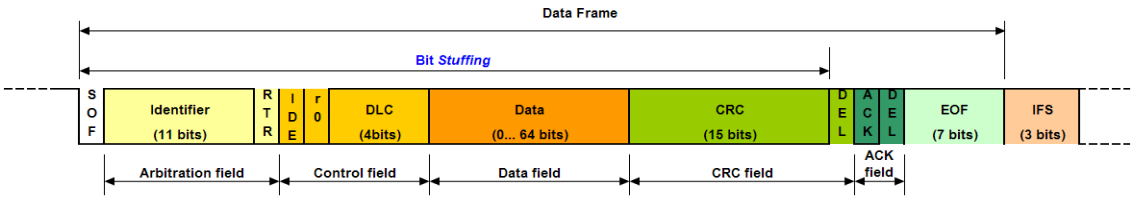


Figura 60 – Trama de dados *standard*.

Como já foi referenciado existem duas versões do protocolo CAN, a versão A e a versão B. Assim, as tramas são ligeiramente diferentes pois a versão A possui 11 bits de identificação, enquanto a versão B possui 29 bits de identificação, sendo esta a principal diferença [Bosh91] [Steve08].

Formato *standard*

Uma mensagem no protocolo CAN, em formato *standard* conforme a Figura 60, é composta pelos bits:

- Início de Trama (SOF – *Start of Frame*);
- Campo de Arbitragem (*Arbitration Field*), sendo este campo composto através do:
  - Identificador (*Identifier*), estabelecendo a prioridade e a sua identificação;
  - Pedido Remoto de Transmissão (RTR – Remote Transmition Request), sendo este o bit que distinguirá um pedido remoto de transmissão de um pedido e envio de dados.

- Campo de Controlo (*Control Field*) contém o bit:
  - Extensão do Identificador (IDE – *IDentifier Extension*), o qual serve para efectuar a distinção entre a trama do formato *standard* (2.0 A) e a trama de formato estendido (2.0 B);
  - Código do Comprimento de Dados (DLC – *Data Lenght Code*), o qual indica a quantidade de bytes que o Campo de Dados (*Data Field*) contém. Nos casos em que a mensagem é um pedido remoto, este campo (DLC) indicará a quantidade de dados pedida;
  - r0 bit reservado para possível uso no futuro.
- O campo de dados pode conter até 8 bytes de dados;
- Verificação Cíclica de Redundância – CRC (*Cyclic Redundancy Check*), verifica integridade dos dados, composto por 16 bits (15 bits+bit delimitador);
- Campo de Confirmação (ACK – *Acknowledge Field*) este campo é transmitido no estado recessivo. Os nós que o recebem impõem o estado dominante nesta posição e, deste modo, o transmissor é informado da correcta recepção da trama. Este campo é delimitado por um bit recessivo;
- Fim de Trama (EOF – *End of Frame*), ou seja, indicação de fim de mensagem correspondente a 7 bits recessivos consecutivos;
- Espaço Entre Tramas (IFS – *InterFrame Space*), é o último bit o qual indica o espaço entre duas tramas de mensagens consecutivas. Este campo inclui 3 bits obrigatoriamente recessivos designados de Intermissão (*Intermission*), durante os quais não pode haver nenhuma trama, excepto se estiver sinalizado como sendo de erro. Se não ocorrer nenhuma mensagem a seguir, o barramento manter-se-á em estado livre (idle).

Formato estendido

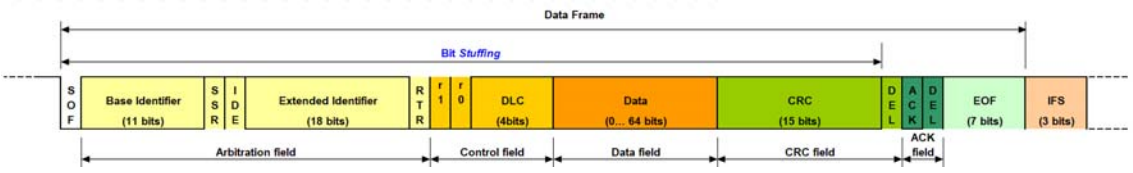


Figura 61 – Trama de dados estendida (Identificador de 29 bits).

Como pode ser observado pela Figura 61, este formato possui 29 bit distribuídos por 11 bits na mesma posição que a trama do formato *standard* e 18 bits de extensão situados depois do bit IDE, o qual indica a presença ou não desta extensão. São acrescentados os seguintes bits:

- Pedido Remoto Substituto (SRR – *Substitute Remote Request*) que substitui o bit RTR, surgindo este depois da extensão do identificador;
- r1 que vem no seguimento do anterior r0 (para utilização futura).

➢ TRAMA REMOTA

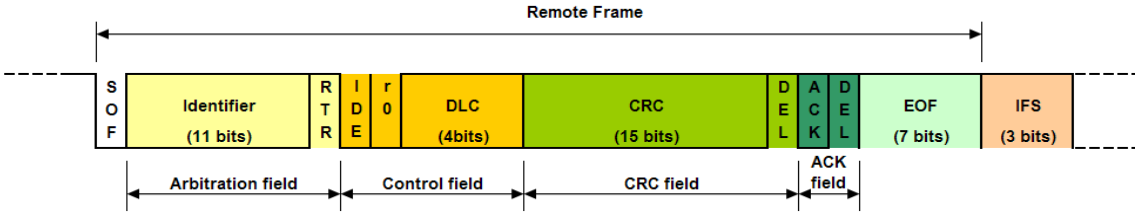


Figura 62 – Trama remota.

É possível um nó de destino requerer os dados do nó fonte, enviando um pedido remoto de dados, surgindo desta forma esta trama remota. A Figura 62 representa este tipo de trama, sendo esta semelhante à anterior mas com duas diferenças fundamentais:

- Nesta trama o bit RTR é transmitido como recessivo, ao invés da trama de dados, o qual é dominante;
- A trama de pedido remoto não possui um campo de dados propriamente dito, isto é, neste campo apenas é especificado a quantidade de dados pedida.

➢ TRAMA DE ERRO

Esta trama é gerada sempre que, um dos nós do barramento, verifique a existência de erro, sendo composta por dois campos:

- *Error Flag* - é constituído por seis bits consecutivos do mesmo estado:
  - Se o nó se encontra no estado *Active Error*, os seis bits são dominantes;

- Se o nó se encontrar no estado *Passive Error* os seis bits são recessivos.

♦ O campo *Error Delimiter* é constituído por oito bits recessivos consecutivos.

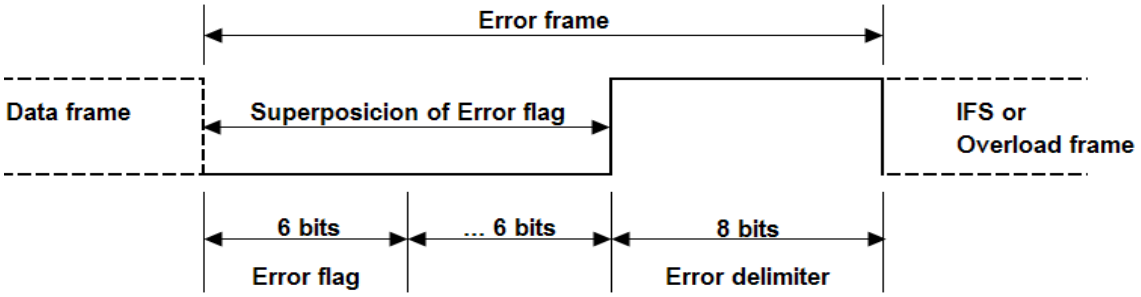


Figura 63 – Trama de Erro.

Este tipo de trama viola as regras de formatação das mensagens CAN, pois é transmitida quando um nó detecta um erro de mensagem, causando que todos os outros nós da rede também enviem uma trama de erro (*superposition*). De seguida o transmissor original automaticamente retransmite a mensagem. Este processo está representado na Figura 63.

➢ TRAMA DE SOBRECARGA

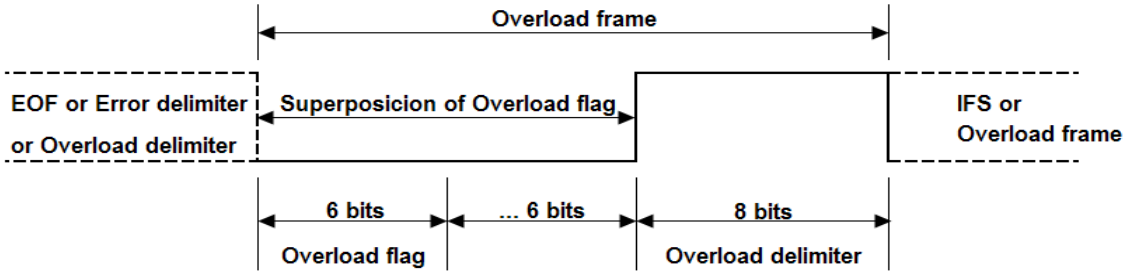


Figura 64 – Trama de sobrecarga.

É semelhante à trama de erro, sendo transmitida por um nó que se encontra demasiado ocupado, sendo gerada durante o espaço entre tramas. É constituída por dois campos: *Overload flag* – cuja duração corresponde a 6 bits; e *Overload delimiter* – duração corresponde 8 bits. Esta trama está representada na Figura 64.

GESTÃO DE ERROS

A detecção de erros é tornada pública a todos os nós através da trama de erro ou de bits de erros. A Figura 65 apresenta o estado dos nós, sendo que eles podem estar num dos três estados de erro seguintes [Bosh91] [Dominique07] [Keith99]:

- Erro activo – é o estado normal em que se encontra um nó. Consegue enviar todo o tipo de tramas, incluindo tramas de erro. Se um nó se encontra neste estado e detecta um erro no barramento, o nó interrompe a transmissão da mensagem actual e gera uma *flag* de erro. Após a conclusão da trama de erro, o barramento retoma o seu funcionamento normal, e o nó sujeito à interrupção tenta reenviar a mensagem cancelada;
- Erro passivo – o nó consegue transmitir e receber todas as tramas, e quando é detectado erro é enviada uma *Passive Error Flags*;
- *Bus off* – o nó fica isolado do barramento.

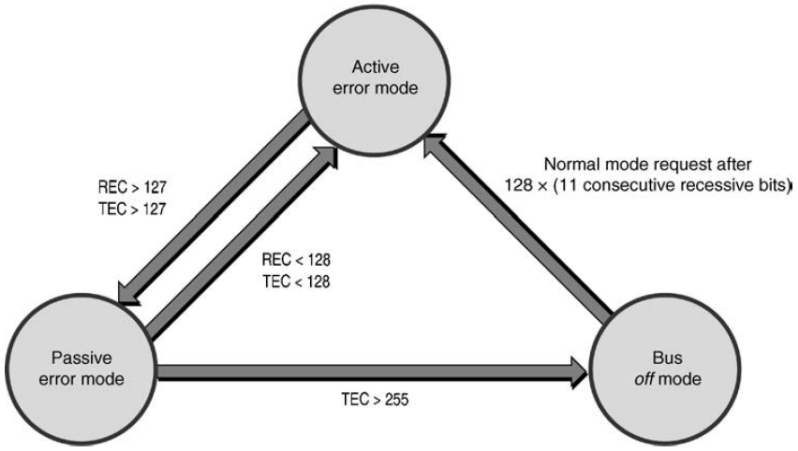


Figura 65 – Estado dos nós em função dos erros [Dominique07].

O modo de definição do estado do nó é obtido pela leitura de dois contadores de erros, que se encontram no próprio nó (*hardware* de controlo / controlador). Um contador (TEC – *Transmit Error Counter*) quantifica os erros de transmissão, enquanto um outro contador (REC – Receive Error Counter) quantifica os erros recebidos. Uma operação com erro origina um incremento de 8 unidades, por sua vez, uma operação bem sucedida diminui o contador em uma unidade.

Caso os contadores possuam valores inferiores a 128, o nó encontra-se no estado normal – erro activo. Quando os valores dos contadores TEC e REC superam os 127, o nó passa a um estado de erro passivo. Se o contador TEC exceder o valor de 255 origina à entrada do estado *bus-off*, por parte do nó Desta forma é prevenido o bloqueio dos nós pelas falhas geradas.

DETECÇÃO DE ERROS

O protocolo CAN possui diversas formas para a detecção e sinalização de erros. Abaixo são apresentadas estas formas de detectar, sendo que os três primeiros dizem respeito aos erros na trama, enquanto que os dois últimos são mecanismos para detectar erro ao nível do bit [Bosh91] [Dominique07] [Keith99].

- CRC

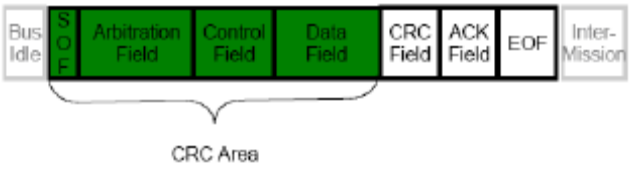


Figura 66 – Campos de aplicação do CRC.

O transmissor efectua um cálculo que compreende vários bits, ou seja, é aplicado desde o bit inicial (SOF) até ao fim do campo de dados. Esta sequência é convertida num polinómio (coeficientes correspondentes à sequência binária) o qual fica sujeito à divisão por um polinómio gerador (previamente definido). O resto desta divisão é então incluído neste campo CRC, por parte do emissor. No lado oposto, o nó receptor efectua a divisão da trama recebida pelo polinómio gerador e se este resultado for nulo é indicativo da não detecção de erros. Caso seja detectado erro, deita fora a trama e gera uma mensagem de erro. Na Figura 66 encontram-se definidos os campos que fazem parte aplicativa do CRC.

- ACK

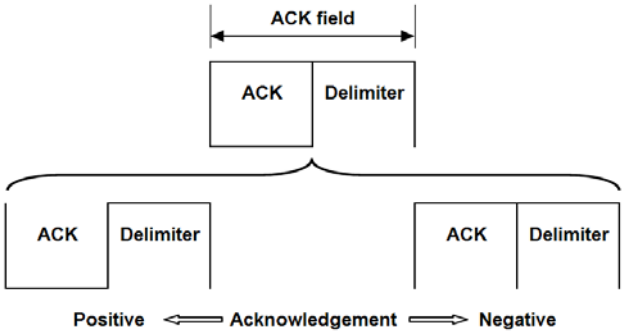


Figura 67 – Sinalização do ACK.

Na troca de tramas, sempre que um receptor recebe uma trama vai enviar um sinal de confirmação de recepção, colocando o bit ACK em dominante. Se o transmissor não receber este sinal, origina um erro ACK. Esta situação está representada na Figura 67.

- FORMATO

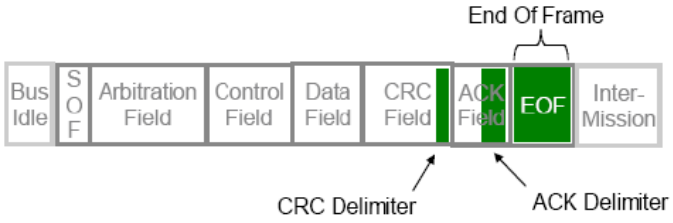


Figura 68 – Erro de formato.

Um bit dominante nos campos CRC, ACK, EOF ou IFS (representados na Figura 68), detectado no nó transmissor origina um erro de formato, sendo gerada uma trama de erro (levando à repetição da transmissão). Ou seja, o transmissor amostra bits destes campos com polaridades erradas.

- MONITORIZAÇÃO

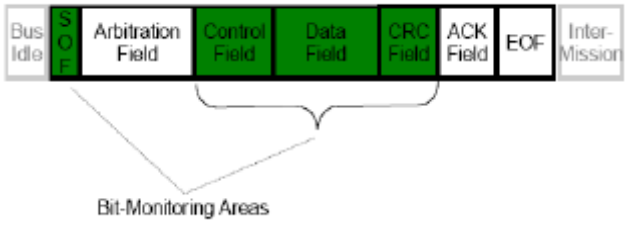


Figura 69 – Áreas de monitorização do bit.

Advém da possibilidade do transmissor monitorizar os sinais no barramento, isto é, se o transmissor envia um bit dominante mas detecta no barramento um bit recessivo ou vice-versa. Desta forma gera-se uma trama de erro e a mensagem é repetida. O bit e os campos alvos desta monitorização encontram-se representados na Figura 69.

- BIT STUFFING

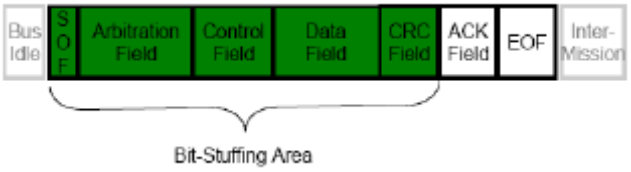


Figura 70 – Áreas de actuação do bit *stuffing*.

Como já visto anteriormente, numa sequência superior a 5 bits consecutivos de valor lógico igual, é introduzido um bit de sinal oposto. A ausência deste bit vai dar origem a erro. Os campos e o bit sujeito à técnica de bit *stuffing* encontram-se representados anteriormente na Figura 70.

TRANSCEIVER

É usual que cada nó CAN deve dispor de um elemento conversor de sinais digitais gerados pelo controlador CAN, os quais sejam capazes de serem transmitidos sobre o barramento CAN (saída diferencial). Em conjugação, permite servir de *buffer* de isolamento entre o controlador CAN e barramento CAN, de modo a evitar os picos de tensão criados no barramento por fontes externas.

4.3.2. ETHERNET

4.3.2.1. INTRODUÇÃO

A Ethernet é uma arquitectura de rede para partilha de dados, que surgiu nos anos 80 por intermédio de três empresas: Digital, Intel e Xerox. Anteriormente já estavam a ser desenrolados esforços para a interligação de estações de

trabalho e impressoras, com objectivo da troca de dados. A versão inicial e experimental da Ethernet foi actualizada, melhorando a sua velocidade de transmissão, que passou a ser de 10Mbps (inicialmente não atingia os 3Mbps, através de cabo coaxial), sendo apelidada de Ethernet II [Beasley09] [Kaplan01] [McQuerry08] [Tanembaum03].

Contudo, através do *Institute of Electrical and Electronic Engineers* (IEEE), em Fevereiro de 1980, um grupo de trabalho sob a égide "*Local Área Network Standards*", começou a delinear a norma designada por IEEE 802. Este grupo de trabalho apresentou em 1985, uma nova norma para a Ethernet designada por IEEE 802.3. Apesar da nomenclatura correcta da ser o IEEE 802.3, é bastante mais comum ser referenciada como Ethernet [McQuerry08] [Tanembaum03].

A partir desta norma e devido às suas características com taxas de transmissão significativas, baixo custo e de fácil instalação, tornou-se na rede preferível ao nível local, isto é, uma *Local Area Network* (LAN). Ao longo dos tempos a norma Ethernet foi evoluindo em função dos novos requisitos e desenvolvimentos tecnológicos, e com naturalidade surgiram alterações e derivações, criando um leque alargado de normas.

4.3.2.2. NORMAS IEEE

2,94Mbps	Ethernet Experimental, 1973	cabo coaxial (tipologia de rede em barramento)
10Mbps	Ethernet II (DIX v2.0), 1982	cabo coaxial grosso (*thick*), as tramas passam a dispor de um campo "Type"
	IEEE 802.3, 1983	cabo coaxial grosso (10BASE5)
	IEEE 802.3a, 1985	cabo coaxial fino (thin) (10BASE2)
	IEEE 802.3d, 1987	ligação em fibra óptica (Fiber Optic Inter-Repeater Link – FOIRL)
	IEEE 802.3i, 1990	par entrançado (10BASE-T)
	IEEE 802.3j, 1993	fibra óptica (10BASE-F)(F...P/...B/...L)
100Mbps	IEEE 802.3u, 1995	Faster Ethernet par entrançado(100BASE-T)
	IEEE 802.3x, 1997	full-duplex e controlo de fluxo, 100BASE-T2
1000Mbps (1Gbps)	IEEE 802.3z, 1998	Gigabit Ethernet sobre fibra óptica (1000BASE-X)
	IEEE 802.3ab, 1999	par entrançado (1000BASE-T)
	IEEE 802.3ac, 1998	extensão da trama com etiqueta "VLAN Tag"
10Gbps	IEEE 802.3ae, 2002	10Gbps sobre fibra óptica (10GBASE-...)
	IEEE 802.3ak, 2004	cabo em cobre "twinax" (10GBASE-CX4)
	IEEE 802.3an, 2006	par entrançado (10GBASE-T)
	IEEE 802.3ap, 2007	backplanes em placa de circuito impresso (PCI)
	IEEE 802.3as, 2006	extensão da trama
40 e 100Gbps	IEEE 802.3ba, 2010	backplane, cobre ("twinax"), fibra (40GBASE-.../100GBASE-...)

Tabela 1 – Normas IEEE e algumas características.

Uma apreciação às normas assentes no IEEE 802.3, ao longo dos tempos, caracteriza a sua evolução, nomeadamente ao nível da velocidade de transmissão. Com este intuito está apresentado na tabela anterior um resumo sobre essas normas.

Fazendo uma abordagem desde o inicio, até à mais recente aprovação (IEEE 802.3bg), surge em 1995 a variante à norma inicial, superando a velocidade de transmissão dos 10Mbps, designado por IEEE 802.3u. Esta norma eleva para os 100 Mbps a velocidade de transmissão sobre cabo de cobre ou fibra óptica apelidando-se de *Fast Ethernet* (10/100 Mbps). Apesar desta mudança os dispositivos de rede tinham possibilidade de comutar automaticamente entre as duas velocidades, mantendo a compatibilidade com a norma anterior.

Numa forma evolutiva, surge em 2002, a norma IEEE 802.3ae, tornando a velocidade de transmissão nos 10Gbps, designando-se por 10Gigabit Ethernet, suportando apenas o modo *full-duplex*. Mais recentemente foram atingidos os 100Gbps, ou seja, mais uma etapa na evolução das velocidades de transmissão.

4.3.2.3. ESTRUTURA

A rede Ethernet, cuja norma é o IEEE802.3, encontra nas duas camadas inferiores do modelo OSI os alicerces para a sua estrutura. Assim, incorpora a camada 1 – Física (*Physical Layer* – PHY) e a camada 2 – Ligação (*Data Link Layer* – DLL), sendo a definição destas duas camadas o objectivo da norma IEEE 802.

Na camada física é definido o sinal (características eléctricas, velocidades de transmissão, codificação), meio de transmissão (cabos e conectores) e tipologias de redes. Por outro lado, na camada ligação (DLL) é definida a forma em como é estabelecida a comunicação sobre o meio, nomeadamente as características seguintes [Stallings06]:

- Determinam o acesso ao meio de comunicação;

- Estruturam as mensagens (de transmissão e recepção) disponibilizando campos para o endereçamento e detecção de erros;
- Estabelecem o elo de ligação com as camadas superiores e desempenham o controlo do fluxo e dos erros na comunicação.

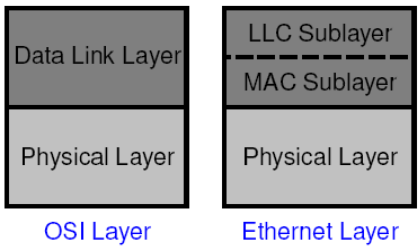


Figura 71 – Comparação do modelo OSI com a Ethernet.

Na Figura 71 é possível verificar que ao formato típico do modelo OSI, designadamente à camada ligação, surge uma alteração passando esta a ser dividida em duas, surgindo a:

- Subcamada *Medium Acess Control* (MAC);
- Subcamada *Logic Link Control* (LLC).

A subcamada MAC vai garantir os dois primeiros pontos expostos anteriormente para a camada DLL. Desta forma vai lidar com acesso físico ao meio no qual inclui o iniciar da trama de dados e, a recuperação na falha por transmissão. Todos os dispositivos passam a ter um endereço MAC, o qual é único e mantém uma tabela com todos os endereços MAC dos dispositivos na sua rede [McQuerry08].

A subcamada LLC permite que parte da camada DLL funcione independentemente da tecnologia existente, ou seja, compatibilidade tecnológica [Tanembaum03]. A versatilidade desta camada permite criar uma interface entre, o endereçamento proporcionado pela subcamada MAC e camada física, com as camadas superiores. Ainda faz parte integrante no processo de encapsulamento.

4.3.2.4. ACESSO AO MEIO

HALF-DUPLEX

Quando a norma Ethernet (IEEE 802.3) foi estabelecida, o acesso ao meio ficou garantido através do CSMA/CD, visto que duas ou mais estações vão partilhar o meio de transmissão.

Com o intuito de atribuir o acesso ao meio e evitar colisões no meio por parte dos dispositivos colocados na rede, o CSMA/CD estipula regras de como os nós (estações) numa rede vão responder, quando dois deles tentam aceder ao meio em simultâneo, ou seja, evitar a colisão de dados no meio físico. A Figura 72 esquematiza o processo de acesso ao meio.

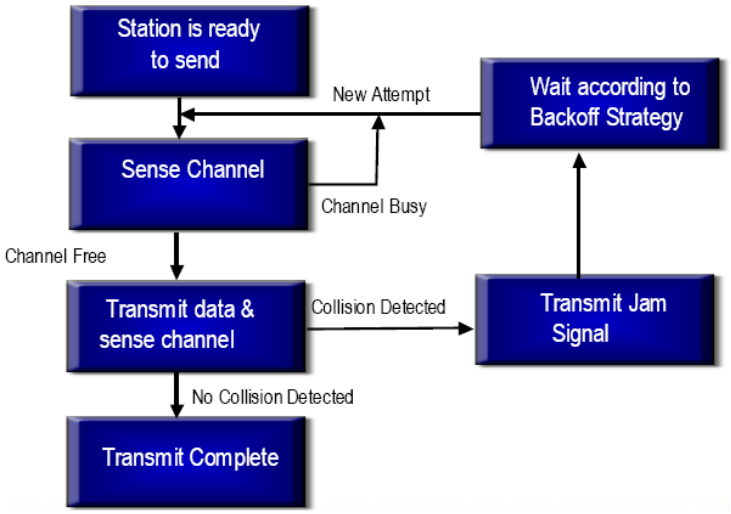


Figura 72 – Acesso ao meio e transmissão de dados [Xilinx11].

Qualquer estação colocada numa rede Ethernet pode usufruir do meio para o envio de dados, desde que ele não esteja ocupado (*Multiple Access*). Uma constante verificação do estado do meio permite determinar quando este está livre (*Carrier Sense*). Se o meio está livre, ou seja, ausência de portadora (*No Carrier*), então é despoletada a transmissão dos dados. Caso contrário, estando o meio ocupado, a estação espera um tempo para voltar a tentar transmitir.

Encontrando-se as estações à escuta do meio, verificando a sua disponibilidade, duas delas podem iniciar a transmissão simultaneamente, originando uma colisão na transmissão. Enquanto as estações se encontram a transmitir ficam a analisar o meio para uma possível ocorrência de colisão. Se acontecer tal situação, as estações transmissoras param de enviar dados e enviam uma sequência de dados (*Jam*), para identificar a ocorrência de erro (*Coilision Detection*). Após a detecção, em cada uma das estações é gerado um tempo aleatório de espera (processo *Backoff*), para voltarem a retransmitir.

FULL-DUPLEX

Com o evoluir da Ethernet, surgiu a versão IEEE 802.3x definindo um segundo do modo de operação designado de *full-duplex*. Este modo permite que duas estações possam simultaneamente trocar dados numa ligação ponto a ponto, prevalecendo independentes do meio de transmissão e de recepção. Simultaneamente pode ocorrer uma troca de dados entre duas estações, acarretando este facto uma duplicação da largura de banda. Assim, uma estação a operar a 10 Mbps passa a dispor de uma largura de banda de 20 Mbps.

Com o funcionalismo do modo de operação em *full-duplex*, o estabelecimento das ligações entre os dispositivos na rede, deve compadecer de:

- Ambas as estações têm de se encontrar habilitadas para trabalhar em modo *full-duplex*.
- Os requisitos do meio físico têm de possuir capacidades para o suporte simultâneo de transmissão e recepção, sem que haja interferência (cablagens).

Para além dos requisitos anteriores, a implementação do modo *full-duplex* leva a algumas considerações, como:

- Deixa de existir o meio físico como um recurso partilhado, pois as ligações passam a ser ponto a ponto. Daqui resulta uma duplicação da largura de banda;

- O protocolo CSMA/CD deixa de ser necessário, em virtude do meio físico não ser partilhado e, consequentemente, da ausência de colisões. A eficiência na transmissão é melhorada;
- As mensagens podem ser enviadas sempre que necessário, não existindo a espera de acesso ao meio, podendo apenas ter de esperar pelo espaço temporal entre mensagens (*interframe gap*);
- As ligações passam a ser estabelecidas ponto a ponto;
- A limitação no comprimento dos segmentos deixa de ser baseado nos requisitos temporais;
- Tipologias de 10BASE5, 10BASE2, 10BASE-FP, 10BASE-FB e 100BASE-T4, não suportam este modo.

➢ CONTROLO DE FLUXO

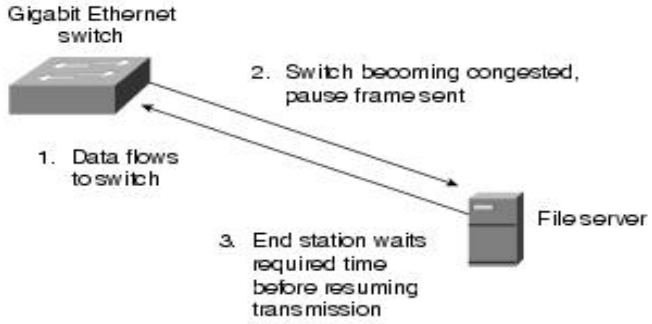


Figura 73 – Sequência de acontecimentos no controlo de fluxo [McQuerry08].

Com a introdução do modo de operação em *full-duplex* na norma 802.3x, ficou disponível a possibilidade de se efectuar o controlo do fluxo do tráfego na rede, tal como apresenta a Figura 73. Basicamente este controlo consiste em efectuar uma pausa no fluxo do tráfego, devido às estações receptoras se encontrarem momentaneamente congestionadas. Isto acontece quando uma estação está a receber dados e, a partir de determinado momento fica congestionada (por exemplo, o *buffer* de entrada fica sem espaço). Nesta situação vai enviar um pedido à estação emissora que suspenda temporariamente o envio de dados. Este pedido é gerado automaticamente por parte da subcamada MAC do receptor e dirigida para a mesma subcamada do emissor, correspondendo a uma trama específica para este acontecimento, cuja designação é *Pause*. Esta trama possui um campo onde especifica o tempo de suspensão pretendido pelo receptor.

Se o congestionamento for aliviado antes do tempo de espera ter terminado, uma segunda trama *Pause* pode ser enviada com o valor de espera nulo. De outro modo passado o tempo a transmissão é restabelecida.

4.3.2.5. TRAMA

A troca de dados na rede Ethernet, com base na norma IEEE 802.3, possui um formato que corresponde à designada trama básica de dados, composta por sete campos, conforme apresentada na Figura 74. Retirando os dois campos iniciais (*Preamble* e SFD) surge a verdadeira trama, ou seja, a trama MAC cujo comprimento vai desde os 64 bytes até aos 1518 bytes. Os primeiros dois campos permitem fazer a sincronização dos sinais na linha. Alterações impostas a esta trama permitem disponibilizar outras opções, outras características para diferentes aplicações.

Figura 74 – Trama básica Ethernet.

Os campos apresentados na Figura 74 têm as seguintes características [McQuerry08] [Rodriguez01] [Stallings06]:

- *Preamble* – é uma sequência alternada de zeros e uns, num total de 56 bits (7 bytes), permitindo efectuar a sincronização entre os dispositivos em comunicação antes se iniciar a transmissão da trama;
- *Star of Frame Delimitier* (SFD) – é um byte composto alternadamente por zeros e uns e que termina com dois bits consecutivos a um (10101011). A partir daqui será a trama propriamente dita;
- *Destination Address* – são seis bytes que identificam a quem se destina a trama, pois cada dispositivo colocado em rede possui um único endereço de acesso ao meio (MAC). Consoante o valor

colocado neste campo, existem três tipos de endereços possíveis, sendo:

- Endereço individual – *Unicast address* (é composto pelo número que vem agregado ao dispositivo colocado em rede);
- Endereço de grupo – *Multicast address*;
- Endereço público – *Broadcast address* (tem a particularidade de ser composto exclusivamente por uns);
- *Source Address* – idêntico ao campo anterior, também de 6 bytes, mas agora diz respeito à estação de onde é oriunda a trama e corresponde sempre a um endereço individual;
- *Lenght / Type* – este campo possui dois bytes que quantificam o número de bytes existentes no campo de dados (*Data*), se este valor for inferior a 1500. No caso de este número ser igual ou superior a 1536 passa a indicar o tipo de formato dos dados (corresponde a uma trama da Ethernet II, ou seja, compatibilidade entre tramas);

♦ *Data* – contém os dados a serem transmitidos desde a fonte até ao destino, de tamanho variável, mas com um valor compreendido entre um mínimo de 46 bytes até um máximo de 1500 bytes. Definido na norma IEEE 802.2

♦ *Pad* – este campo é utilizado para complementar o campo de dados (*Data*) caso este não consiga atingir o número mínimo de 46 bytes necessários para a transmissão. Assim, se o campo de dados for inferior a 46 bytes, serão inseridos neste campo os restantes bytes até atingir os 46 bytes. É de referir que o comprimento mínimo da trama Ethernet é de 64 bytes, excluindo os dois primeiros campos, resultando daqui a necessidade deste campo;

♦ *Frame Check Sequence* (FCS) – constituído por 4 bytes este campo permite a verificação de erros através do CRC. Na origem é calculado o valor CRC, sobre a trama a enviar, incluindo esse valor neste campo. No destino é calculado o valor CRC, sobre a trama recebida, comparando-se de seguida os dois valores.

PAUSE

Como visto no *full-duplex*, existe a possibilidade do receptor controlar o fluxo de dados. Esta possibilidade advém do facto de este enviar uma trama ao emissor a pedir que este efectue uma pausa. O pedido de pausa é uma trama designada por Pause, cuja estrutura é apresentada Figura 75 [McQuerry08].

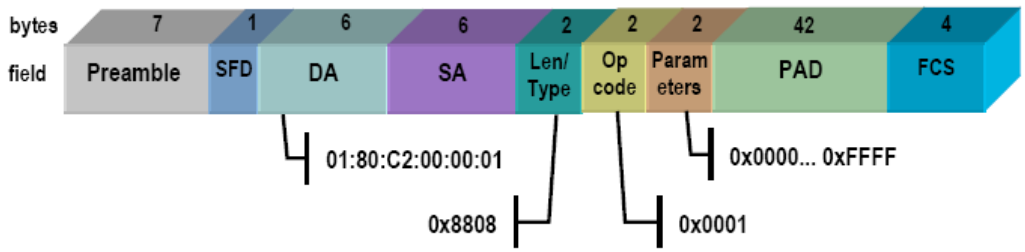


Figura 75 – Formato da trama *Pause*.

A trama continua a ser uma trama Ethernet mas com algumas particularidades, designadamente:

- O endereço de destino (DA) pode ser:
  - *Unicast*, indicando apenas a estação que deve efectuar a suspensão na transmissão;
  - *Multicast*, (01:80:C2:00:00:01) sendo este um endereço reservado que permite o reconhecimento de uma trama *Pause*.
- O campo *Type* assume o valor de 0x8808, sendo este indicativo de uma trama de controlo MAC.
- O campo MAC *Control Opcode* tem como objectivo indicar o tipo de trama que está a ser utilizada, isto é, dentro das tramas de controlo define o tipo de controlo. Sendo esta uma trama *Pause* assume o valor 0x0001;
- Em seguida surge o campo MAC *Control Parameters*, tendo como finalidade a definição do tempo de duração da pausa. São permitidos valores de 0x0000 até 0xFFFF (2 bytes), em unidades de 512 tempos de bit. Encontrando-se em execução um tempo de pausa e chegando uma nova trama de *Pause*, sucede uma nova

pausa com o novo tempo. Se esta nova pausa contiver um tempo 0, então a pausa termina, voltando ao seu funcionamento normal.

- Como do campo *Data* da trama Ethernet apenas 4 bytes são necessário, o campo *Pad* (42 bytes) vai ser preenchido com zeros, perfazendo os 46 bytes necessários para a trama.

Q TAG (VLAN TAGGED)

É uma trama Ethernet à qual é acrescentada um novo campo designado por TAG. Foi desenvolvida para ser aplicável numa rede *Virtual LAN* (VLAN), necessitando para tal de preencher o tal campo TAG. A sua definição encontra-se nas normas 802.3ac e 802.1q. Em relação à trama básica, há um aumento em 4 bytes, pois é o tamanho deste novo campo, passando no total a trama a ter no máximo um tamanho de 1522 bytes [Tanembaum03].

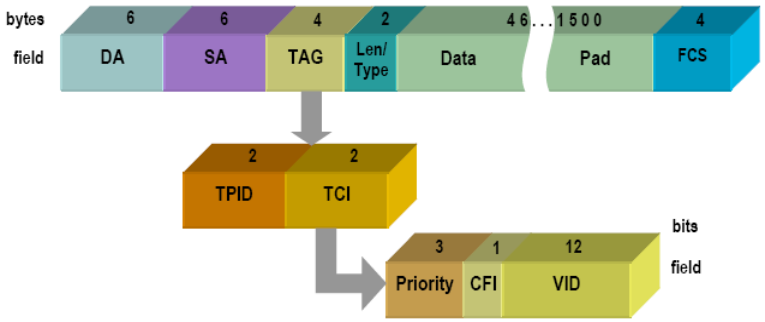


Figura 76 – Trama Q-TAG.

A inserção dos novos campos situa-se entre o *Source Address* e *Lenght/Type*, conforme se pode observar na Figura 76. Então, tendo campo *TAG* um comprimento de 4 bytes, estes correspondem a dois grupos de 2 bytes sendo eles:

- TPID (*Tag Protocol Identifier*) – Serve para identificar a trama assumindo, este campo, o valor de 0x8100;
- TCI (*Tag Control Information*) – Contém informação de controlo, sendo constituído por 3 campos, sendo:
  - *Priority* – este campo é composto por 3 bits, correspondendo à atribuição de um nível de prioridade à trama;

- CFI (*Canonical Format Identifier*) – é um campo apenas constituído por um bit, e se este estiver a 1 indica a presença de *Routing Information Field* (RIF);
- VID (*VLAN Identifier*) – é colocado neste campo a identificação da VLAN, à qual pertence a trama. Tem um comprimento de 12 bits permitindo atribuir 4096 VLAN.

EXTENDIDA

Figura 77 – Formato extendido da trama.

Esta trama é semelhante à trama básica da Ethernet, apenas sendo-lhe acrescentada no final um campo, designado por extensão (Figura 77). Com a chegada do *Gigabit Ethernet* (IEEE 802.3z), foi necessário garantir que as tramas teriam o comprimento suficiente para que, em caso de colisão, esta se iria propagar a todas as estações na rede. O comprimento mínimo para a transmissão da trama é garantido com 512 bytes. Assim, se a trama básica não atingir este valor (o valor mínimo para ser transmitida são 64 bytes) é colocada uma extensão no final para compensar os bytes em falta, perfazendo no total os tais 512 bytes. A informação contida na extensão é irrelevante pois o objectivo é atingir o comprimento mínimo da trama. Esta extensão apenas faz sentido no modo de operação em *half-duplex*, onde podem ocorrer colisões [McQuerry08].

BURSTING

Com o objectivo de melhorar a eficiência da rede quando uma estação está a transmitir pequenas tramas, abaixo dos 512 bytes, foi introduzido este formato. Definida pela norma IEEE 802.3z encontra-se vocacionada para trabalhar com redes *Gigabit Ethernet* ou superiores e que consiste em manter uma estação a transmitir pequenas tramas sem deixar o meio [McQuerry08].

Este processo desenrola-se com o envio normal de uma trama, usualmente uma trama estendida e garantido o sucesso do primeiro envio, seguem as restantes tramas, até que seja atingido o limite de 65536 tempos de bit. Entre cada uma destas tramas é incluído um espaço – *interframe gap*, permitindo a separação das tramas. Mas ao invés de permitir a libertação do meio, a estação emissora preenche este espaço com bits, mantendo assim a posse sobre o meio. Na recepção estes bits são distinguíveis daqueles que pertencem aos dados, não sofrendo alteração a mensagem.

Aquando da primeira trama existe a possibilidade de ocorrência de colisão, daí a necessidade da extensão caso a trama seja inferior aos 512 bytes. Nas tramas seguintes, como o meio está ocupado, não existe a possibilidade de colisão, logo dá-se a supressão da extensão.

JUMBO

Esta trama não está definida por norma, apenas é promovida por um grupo de fabricante cujo objectivo é dotar de um campo de dados até 9000 bytes. Com o aumento das tramas pretende-se elevar a eficiência da comunicação aliando uma diminuição na escala de processamento. O modo de operação é em *full-duplex*.

TEMPOS

- *SLOT TIME*

Um *slot time* numa rede Ethernet, vai definir uma temporização correspondente a um tempo mínimo que uma trama demora a ser transmitida. Assim, um *slot time* deve ser o tempo necessário a que cada transmissor possa assegurar a detecção de colisões e, simultaneamente, o tempo requerido para que as colisões sejam propagadas a todas as estações. Este tempo corresponde a 512 tempos de bit, para redes Ethernet 10 e 100 Mbps, e 4096 tempos de bit para redes *Ethernet Gigabit*. Com o *slot time* é garantido que uma estação não consegue terminar a transmissão de uma trama sem antes ter detectado a

ocorrência de uma colisão, ou seja, o tempo de transmissão de uma trama não pode ser menor que o *slot time*.

É de referir que este *slot time* apenas é aplicável em transmissões com suporte em *half-duplex*. No caso contrário, Ethernet com suporte em *full-duplex*, este tempo não faz qualquer sentido.

- *BACKOFF*

O processo *backoff* consiste, tal como descrito atrás, em gerar um tempo de espera aleatório por parte de uma estação, após a detecção de uma colisão, para voltar a retransmitir a trama envolvida. Deste modo é evitada nova colisão, caso as duas estações voltassem a transmitir simultaneamente com base num tempo previamente estipulado e igual para todas as estações.

- *INTERFRAME GAP (IFG)*

Entre cada trama deve existir um espaço temporal mínimo que se compadeça com o reconhecimento da separação de duas tramas, permitindo aos dispositivos restabelecerem condições para a recepção da próxima trama. Este valor é definido em 96 tempos de bit.

- *JAM*

As estações enquanto transmitem encontram-se a monitorizar o meio reconhecendo uma colisão, através de um excesso de corrente que se gera no meio. A partir deste momento pára de transmitir a trama e passa a transmitir uma sequência de 32 bits. O propósito desta sequência é assegurar que todas as estações vão receber este sinal, obrigando as estações que estavam a transmitir a pararem e que recebiam descartam-se da trama que estavam a receber.

4.3.2.6. TOPOLOGIA DA REDE

Com o objectivo da troca de dados são constituídas estruturas físicas de interligação entre os diferentes dispositivos. Estas estruturas podem tomar diferentes formas e formatos criando uma topologia característica. As topologias mais usuais encontram-se descritas nos próximos pontos, permitindo a partir destas, criar novas topologias de rede, agregando um pouco de cada uma destas.

BARRAMENTO (*BUS*)

Todos os dispositivos de rede estão ligados ao longo de um cabo central, designado de barramento (*Bus*) e são reconhecidos pelos seus endereços físicos. Nas extremidades do cabo existe uma carga adaptada (terminador), para absorção dos sinais eléctricos que percorrem o barramento. A Figura 78 representa uma estrutura deste tipo.

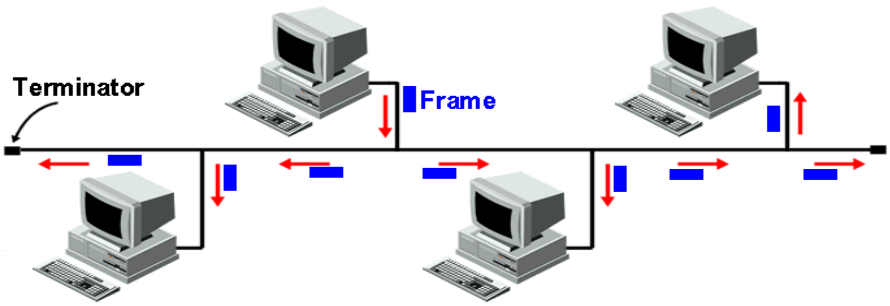


Figura 78 – Rede em tipologia barramento.

Nesta topologia, cada trama enviada para o barramento é escutada pelas restantes estações, ou seja, é uma transmissão *broadcast*. Cada estação analisa a trama e se esta é dirigida a si, a estação recolhe a trama, senão a trama é descartada. Deste modo, cada estação pode responder aos dados enviados para ela e ignorar os dados enviados para as outras estações.

Como o meio é partilhado pelos dispositivos colocados ao longo do cabo, estamos perante uma comunicação *half-duplex*, onde é necessário definir condições para o acesso ao meio. Existindo um corte no barramento, toda a

estrutura da rede deixa de funcionar, nem mesmo as partes divididas (inexistência de terminador nos pontos de quebra).

Foi a topologia base para a implementação da Ethernet com o cabo coaxial, no entanto, foi sendo substituída pela tipologia em estrela, apesar de ainda existirem aplicações com base nesta tipologia. Fácil de instalar e implementar para pequenas redes, de baixo custo (em termos de material e comparativamente com outras tipologias), detecção facilitada nas anomalias mas acarreta um comprimento e número de estações limitado, assim como, uma taxa de transmissão moderada.

ANEL (*RING*)

Todos os dispositivos de rede encontram-se ligados entre si, criando um formato físico de anel. Com isto cada dispositivo vai estabelecer ligações com outros dois dispositivos (um de cada lado, formando o tal anel).

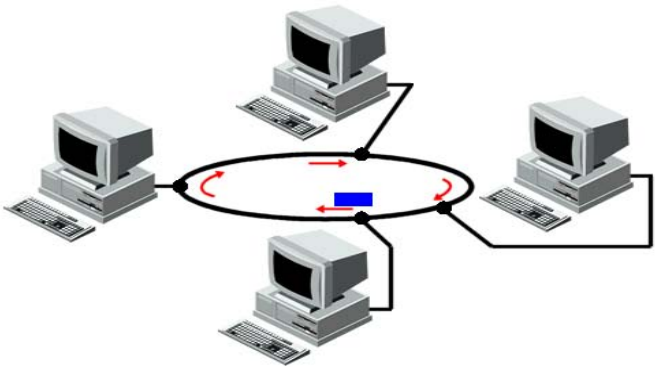


Figura 79 – Rede em tipologia anel.

O método de acesso ao meio consiste na passagem de um testemunho (*token*), que periodicamente é enviado de estação em estação, permitindo desta forma o envio e recepção de dados e regenerando o sinal. A Figura 79 representa esta estrutura onde, uma mensagem colocada no testemunho é retirada do anel pela mesma estação que a colocou lá, garantindo a passagem por todas as estações. No caso de uma estação não possuir dados para transmitir, apenas dá seguimento ao testemunho. Como cada estação tem acesso ao meio em períodos de tempo regulares esta é uma rede que, por este facto, assume a designação de determinística. Uma estação que não esteja a

funcionar nas devidas condições pode colocar em causa toda a rede e uma falha ou quebra de ligação deixa a rede inoperacional. Consegue superar a largura da banda da tipologia em barramento, podendo abranger longas distâncias. A sua manutenção não é fácil, pois a substituição de uma estação coloca a rede fora de uso

O aparecimento da fibra óptica através da *Fiber Distributed Data Interface* (FDDI) levou a uma utilização deste tipo de rede, por proporcionar uma eficiência elevada neste recurso.

ESTRELA (*STAR*)

Neste tipo de estrutura da rede pressupõe a existência de um ponto central, onde todos os dispositivos pertencentes à rede vão estar ligados. A servir de ponto central encontram-se dois dispositivos:

- *Hub* – faz a distribuição das tramas recebidas, regenerando o sinal e difundindo pelas suas saídas (portos). Com isto, todos os dispositivos a ele ligado vão receber as mesmas tramas, ou seja, executa o *braodcast* da trama, instituindo assim uma rede em modo de operação *half-duplex*. A Figura 80 é representativa desta configuração;
- *Switch* – é um dispositivo similar ao anterior diferenciando-se no modo em como efectua a distribuição. Neste equipamento a trama recebida é distribuída exclusivamente para o porto associado ao endereço de destino, abrangendo desta forma o modo de operação em *full-duplex*. Assim, o *switch* diferencia-se do *hub* por operar ao nível MAC, não se limitando à camada física. A Figura 81 é representativa desta configuração

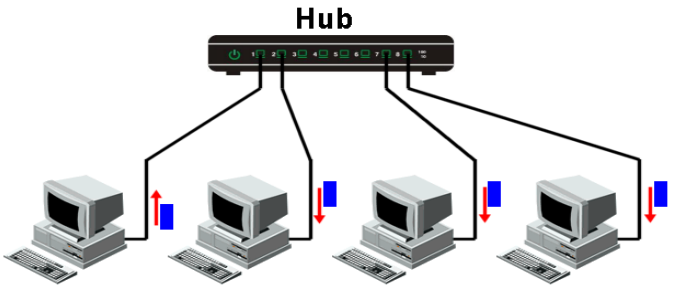


Figura 80 – Rede em tipologia estrela utilizando *hub*.

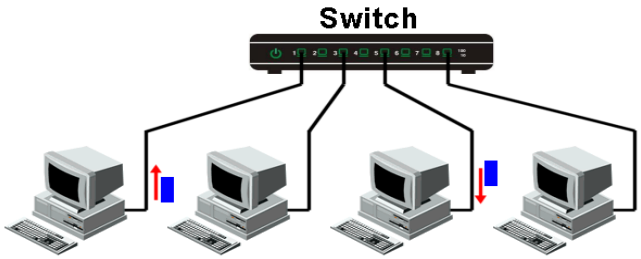


Figura 81 – Rede em tipologia estrela utilizando *switch*.

Esta tipologia é actualmente a que se encontra implementada na maioria das LAN, pois é simples de estruturar, permite mudança de *half-duplex* para *full-duplex* sem que haja alteração da rede (apenas de dispositivo central), aliando uma elevada expansibilidade. Cada uma das estações opera independentemente das restantes, facilitando a manutenção da rede. Existe uma forte dependência do ponto central, pois no caso de se encontrar inoperacional toda a rede fica em baixo e, por outro lado limita o número de dispositivos ao número de portos que possui. Convém referir que este tipo de estrutura possibilita que o ponto central possa sofrer excesso de tráfego e, caso este não tenha capacidades para gerir o tráfego, pode comprometer o rendimento da rede (troca de dados).

4.3.2.7. ENDEREÇAMENTO

Uma trama possui sempre, pelo menos, um destino e a forma em como é enquadrado esse destino no âmbito do seu alcance, tem uma designação específica. A designação reside em três opções: *Unicast*, *Multicast* ou *Broadcast*. Na Figura 82 são apresentadas as formas de endereçamento.

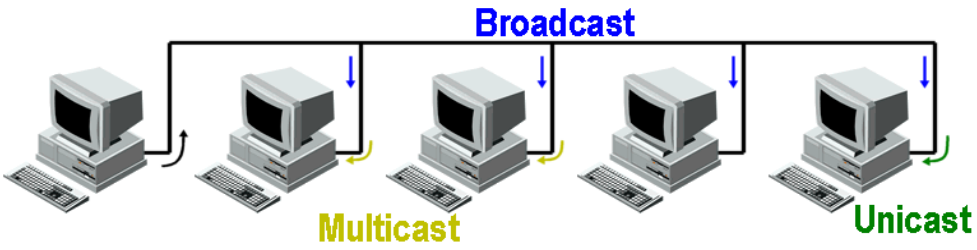


Figura 82 – Tipos de comunicações Ethernet.

Abordando cada um dos tipos de comunicação temos:

- *Unicast* – Os dados têm como destino uma única estação, ou seja, existe apenas uma estação fonte e uma estação receptora, independente do número de estações que existam na rede;
- *Multicast* – Os dados são enviadas para um grupo de estações receptoras, por uma estação fonte. Os grupos encontram-se definidos na rede, permitindo que a especificação do endereço atingia um determinado grupo (em principio será composto por mais que uma estação);
- *Broadcast* – O universo de dispositivos que se encontram na rede, são receptores de uma mensagem enviada por uma estação fonte.

ENDEREÇOS

Cada dispositivo de rede, usualmente reconhecido como NIC – *Network Interface Card*, tem associado a si um endereço único de acesso ao meio (MAC *Address*), permitindo-lhe desta forma ser reconhecido na rede onde se encontra inserido. Sendo o endereço MAC único à saída do fabricante, este também é reconhecido como *Burn In Address* (BIA) [Beasley09]. Este endereço é composto por 6 bytes, tal como já visto anteriormente na trama Ethernet, correspondente aos campos *Destination Address* e *Source Address*. A composição do endereço corresponde ao formato apresentado na Figura 83.

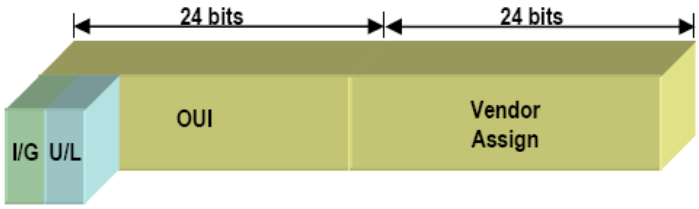


Figura 83 – Composição do MAC *Address*.

Os 6 bytes do MAC *Address* são divididos em dois grupos de 3 bytes, sendo estes:

- OUI (*Organizationally Unique Identifier*) – os primeiros 24 bits são atribuídos pelo IEEE aos fabricantes, ou seja, cada fabricante é referenciado, pelo menos, por um código. Neste campo surgem

dois bits cujo significado apenas tem sentido quando o endereço é utilizado como destino. Estes bits são:

- I/G (*Individual/Group*) – é um bit que identifica o endereço de destino da trama. Se o destino for uma única estação (*unicast*) o bit é colocado a 0 ou se for para um grupo de estações (*multicast* ou *broadcast*) o bit é colocado a 1;
- U/L (*Universally/Locally*) – este bit indica se o endereço é administrado a nível universal (se igual a 0) ou a nível local (se igual a 1). Normalmente este bit é de índole universal, contudo localmente pode ser alterado e neste caso os bits OUI deixam de ter significado.

♦ *Vendor Assign* – são 24 bits de uso exclusivo do fabricante, ou seja, cabe ao fabricante atribuir um número ao dispositivo de rede por si fabricado.

Os endereços MAC são normalmente apresentados num conjunto de 12 algarismos, associado dois a dois (representando os bytes) e separado por dois pontos ou travessão, exemplificando: 00:80:C8:10:FA:C1.

4.3.2.8. TECNOLOGIA

A designação técnica das redes Ethernet tem por base três parâmetros, sendo a sua representação genérica condizente com a seguinte estrutura:

[Parâmetro 1] Parâmetro 2 – [Parâmetro 3]

Onde,

- ♦ Parâmetro 1 – Velocidade de transmissão;
- ♦ Parâmetro 2 – Corresponde ao sinal do tipo *Base* ou *Broad*;
- ♦ Parâmetro 3 – Cablagem.

Com o andar dos anos a evolução tecnológica aplicada a este tipo de rede permitiu alargar as opções disponíveis no mercado. Assim, nos pontos seguintes são apresentadas sucintamente estas opções de modo a clarificar um pouco mais a rede Ethernet. As Tabelas 2, 3, 4 e 5 representam as diferentes tecnologias da rede Ethernet.

Designação	Descrição
10BASE5	10 – velocidade de transmissão:10 Mbps 5 – comprimento máximo do cabo: 500m Rede em barramento sobre cabo coaxial grosso (*thick*)
10BASE2	2 – comprimento máximo do cabo: supostamente deveria atingir os 200m mas o limite situa-se nos 185m Rede em barramento sobre cabo coaxial grosso (*thin*)
10BASE-T	T – Par entrançado não blindado - UTP (*Unshield Twisted Pair*), com comprimento máximo de 100m
10BASE-F variantes:	F – Fibra óptica (*optical Fiber*) Utiliza a fibra óptica, atinge comprimentos desde 500m até 2000m
10BASE-FL	Ligação em fibra óptica (*Optical Fiber Link*)
10BASE-FP	Fibra óptica passiva (*Optical Fiber Passive*)
10BASE-FB	Fibra óptica "*backbone*" (*Optical Fiber Backbone*)
10BROAD36	Banda larga com modulação, de comprimento 3600m

Tabela 2 – Tecnologia Ethernet.

Designação	Descrição
100BASE-TX	2 pares entrançados (UTP ou STP), cat.5, até 100m,
100BASE-FX	2 fibras multimodo, distância até 2000m
100BASE-T2	2 pares entrançados (UTP ou STP), cat.3, até 100m
100BASE-T4	4 pares entrançados (UTP ou STP) cat.3, 4 ou 5

Tabela 3 – Tecnologia *Fast Ethernet*.

Designação	Descrição
1000BASE-SX	Fibra multimodo (S – Short wavelenght = 770 a 860nm), distância até 550m
1000BASE-LX	Fibra (L – Long wavelenght = 1270 a 1355nm) monomodo e multimodo, com distâncias de 5000m e 500m, respectivamente
1000BASE-T	4 pares entrançados UTP, comprimento de 100m
1000BASE-CX	Twinax, comprimento de 25m

Tabela 4 – Tecnologia *Gigabit Ethernet*.

Designação	Características
10GBASE-SR	Fibra óptica monomodo e multimodo (850 nm LAN)
10GBASE-LR	Fibra óptica monomodo e multimodo (1310 nm LAN)
10GBASE-ER	Fibra óptica monomodo e multimodo (1550 nm LAN)
10BASE-SW	Fibra óptica monomodo e multimodo (850 nm WAN)

10GBASE-LW	Fibra óptica monomodo e multimodo (1310 nm WAN)
10GBASE-EW	Fibra óptica monomodo e multimodo (1550 nm WAN)
10GBASE-LX4	Fibra óptica monomodo e multimodo (4fontes emissoras)
10GBASE-CX4	4 cabos twinax
10BASE-LR	Fibra óptica de longo alcance (*Long Reach*)
10BASE-LRM	Fibra óptica multimodo de longo alcance (*Long Reach Multimode*)
10GBASE-T	4 pares entrançados

Tabela 5 – Tecnologia *10Gigabit Ethernet*.

4.3.3. PROTOCOLO TCP/IP

4.3.3.1. INTRODUÇÃO

A base que se encontra na origem do TCP/IP remonta à pesquisa desenvolvida pelo departamento norte-americano ARPA (*Advance Research Projects Agency*), com o intuito de criar uma rede comutada por pacotes de dados. Com isto, um determinado pacote de dados poderia assumir qualquer caminho entre origem e destino, os quais são identificados por um único endereço de rede. Como resultado deste esforço surgiu a rede designada por ARPANet, a qual tinha por objectivos (assumidamente governamentais):

- Resistir a um ataque nuclear;
- Permitir que diferentes sistemas computorizados pudessem facilmente comunicar entre eles;
- A necessidade de interligar sistemas a longas distâncias

A rede ARPANet necessitava de protocolos que assegurassem tais objectivos, sendo fiável e de fácil implementação, como tal surge a estrutura TCP/IP. O crescimento da rede ARPANet deu origem ao aparecimento da Internet [Beasley09] [Tanembaum03].

O TCP/IP na sua essência incorpora dois protocolos que se interligam, onde o TCP corresponde ao protocolo de controlo da transmissão e IP é o protocolo sobre a Internet. A aplicabilidade deste protocolo é largamente abrangente, começando pelas redes locais LAN, passando pelas grandes redes WAN (*Wide*

Area Network) até à rede global Internet. Também assume um suporte alargado no campo dos sistemas operativos permitindo deste modo que dispositivos com diferentes estruturas internas possam estar ligados na mesma rede.

Assim, o TCP é responsável pela verificação da correcta entrega dos dados desde a fonte emissora até ao receptor. Acondiciona uma estrutura que lhe permite detectar erros ou perda de dados, despoletando a retransmissão dos dados enquanto estes não estiverem ausentes de erros ou não estiverem completos. Já o IP assume o papel de movimentar os pacotes de dados entre nós, tendo em conta o endereço de destino (IP *address*). A organização destes endereços permite seccionar toda a rede em sub-redes (países, organizações, empresas, etc.), num processo em cadeia, levando os dados de encontro à máquina de destino [Kozierok05].

4.3.3.2. CAMADAS DO TCP/IP

Baseando nas sete camadas do modelo OSI, o protocolo TCP/IP surge com quatro camadas (ver Figura 84) sendo estas designadas de: Aplicação, Transporte, Internet e Interface de Rede [Beasley09] [Oracle11] [Rodriguez01] [Stevens94] [Tanembaum03].

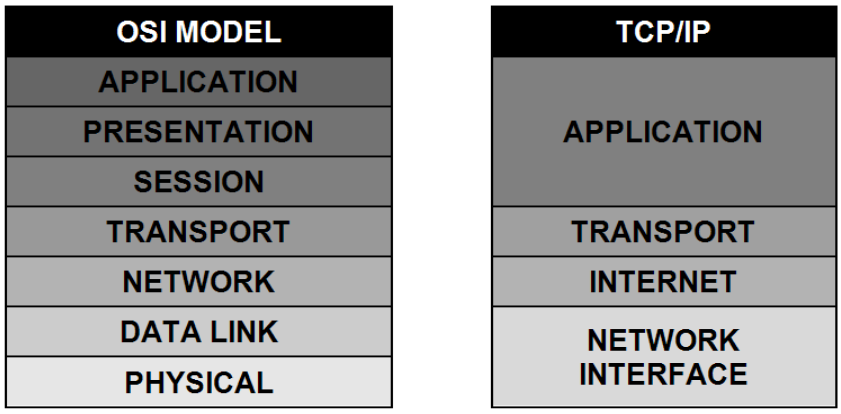


Figura 84 – Modelo OSI e camadas no TCP/IP.

Estas camadas têm como finalidade:

- Camada Aplicação: é responsável pela ligação com a camada Transporte. Incorpora as aplicações usuais da rede (*Telnet*, SMTP

– *Simple Mail Transfer Protocol*, FTP – *File Transfer Protocol*, DNS – *Domain Name System*, entre outros);

- Camada Transporte: assegura o encaminhamento dos dados entre a origem e o destino, assim como o estabelecimento e finalização das ligações. São dois os protocolos a actuar nesta camada o TCP e UDP – *User Datagram Protocol*. A grande diferença entre estes dois protocolos reside no facto de o TCP garantir a entrega dos dados (orientada às ligações) enquanto o UDP não garante a entrega dos dados (não estabelece ligação logo ligações não fiáveis). Este último é muito aplicado em situações de vídeo e áudio, onde impera a rapidez ao invés da ausência de erros;
- Camada Internet: define o protocolo usado para o endereçamento e encaminhamento dos pacotes. Protocolos típicos desta camada são: IP, ARP – *Address Resolution Protocol*, RARP – *Reverse Address Resolution Protocol*; ICMP – *Internet Control Message Protocol*; IGMP – *Internet Group Management Protocol*;
- Camada Interface de Rede: assegura a forma como o dispositivo (máquina) se liga à rede, podendo esta ser Ethernet, *Token Ring* ou outras.

4.3.3.3. ENCAPSULAMENTO

O encapsulamento consiste num processo de acrescento de informação aos dados que se pretendem transmitir numa rede [Oracle11]. À medida que os dados transitam entre camadas superiores para inferiores, existe a necessidade de os caracterizar em função dos protocolos que utilizam. Em cada camada é acrescentada informação no início, ou seja, no cabeçalho (*header*) e algumas vezes no final (*trailer*) [Stevens94]. Este processo encontra-se representado na Figura 85.

À informação que transita entre camadas de uma rede como uma unidade, a qual pode conter os elementos de endereçamento, informação de controlo ou dados é designada de PDU (*Protocol Data Unit*).

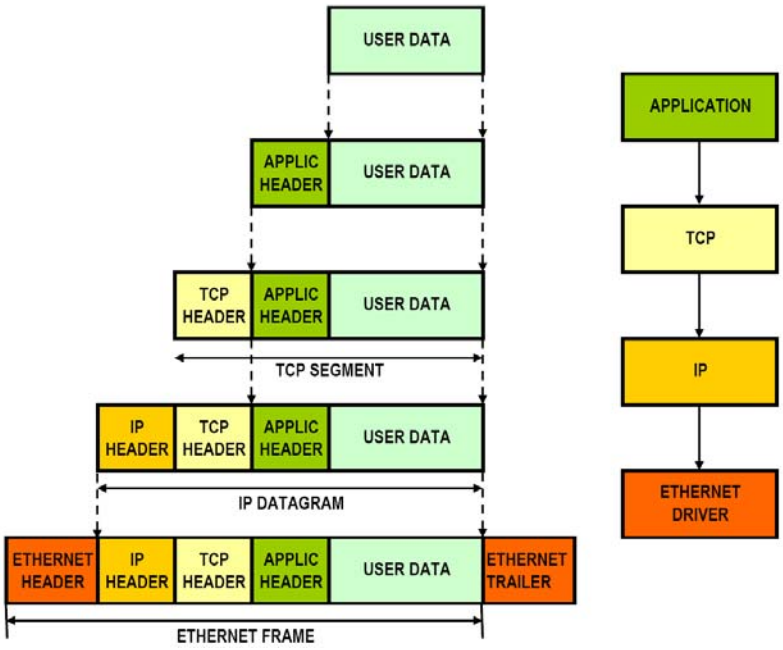


Figura 85 – Processo de encapsulamento.

Na recepção, quando a transição entre camadas é ascendente, o processo é o inverso e designa-se por desencapsulamento.

4.3.3.4. CABEÇALHO TCP

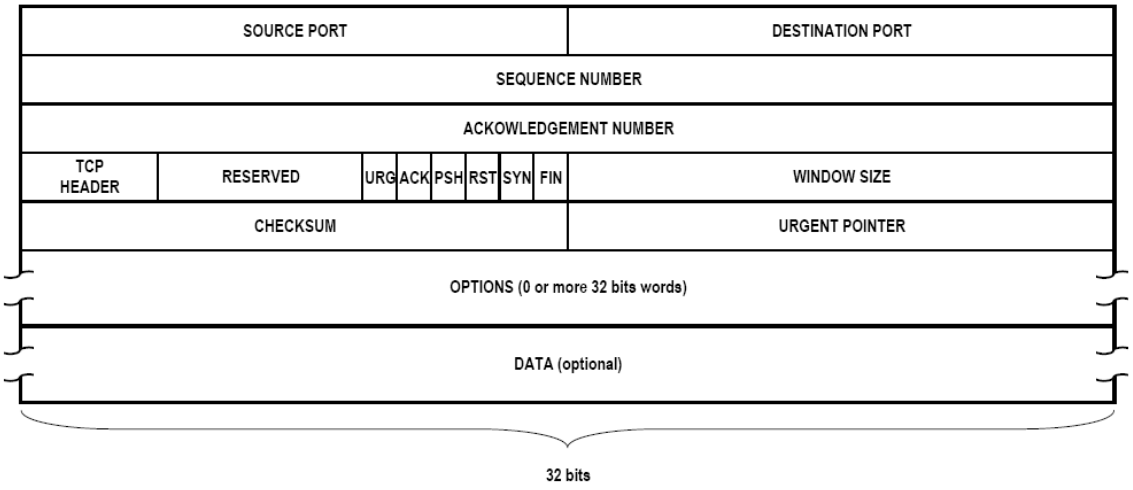


Figura 86 – Formato do cabeçalho TCP.

Para dar seguimento aos dados, a camada transporte, vai acrescentar o cabeçalho TCP o qual dá origem ao Segmento TCP (TCP *Segment*). Este

cabeçalho é composto pelos campos apresentados na Figura 86, sendo eles [IETF81b] [Stevens94] [Tanembaum03]:

- *Source Port* (16 bits) – identifica o porto de origem;
- *Destination Port* (16 bits) – identifica o porto de destino;
- *Sequence Number* (32 bits) – como todos os bytes enviados são numerados sequencialmente pelo TCP, este campo indica a posição do 1º byte de dados do pacote nos dados globais;
- *Acknowledgement Number* (32 bits) – indica o próximo número da sequência dos dados que o emissor do ACK espera receber, ou seja, representa o número do último byte correctamente recebido acrescido de um;
- TCP *Header Lenght* (4 bits) – este campo representa o *offset*, em palavras de 32 bits, no início do campo de dados. Indirectamente está a indicar o tamanho do cabeçalho;
- *Reserved* (6 bits) – campo não utilizado;
- FLAGS
  - URG – dados urgentes;
  - ACK – reconhecimento dos dados como ok;
  - PSH (*Push*) – o *buffer* de dados deve ser entregue na aplicação;
  - RST – *reset* da ligação (devido a algum erro).
  - SYN – serve para o estabelecimento da ligação na fase inicial;
  - FIN – utilizada para terminar uma ligação.
- *Window Size* (16 bits) – define o tamanho da janela utilizada pelo mecanismo da janela deslizante;
- *Checksum* (16 bits) – serve para controlar os erros no pacote (cabeçalho e dados);
- *Urgent Pointer* (16 bits) – indica onde termina o byte dos dados urgentes, caso a flag URG esteja a 1;
- *Options* – o comprimento deste campo encontra-se definido pelo TCP *Header Lenght* e serve para indicar ao outro dispositivo o tamanho máximo dos dados que suporta.

PORTOS

Porto (*port*) é tipicamente um endereço que serve para direccionar dados para uma aplicação específica no destino. Existem 65536 portos divididos em três categorias e que são definidos pelo ICANN (*Internet Corporation for Assigned Names and Numbers*) do seguinte modo [Beasley09]:

- 0 a 1023 são portos reservado (*well-know ports*) pelo, no qual se encontram os comummente utilizados (ex.: porto 21 = FTP; porto 80 = http; etc.);
- 1024 a 49151 encontram-se os portos reservados
- 49152 a 65535 são os portos dinâmicos e/ou privados.

4.3.3.5. CABEÇALHO IP

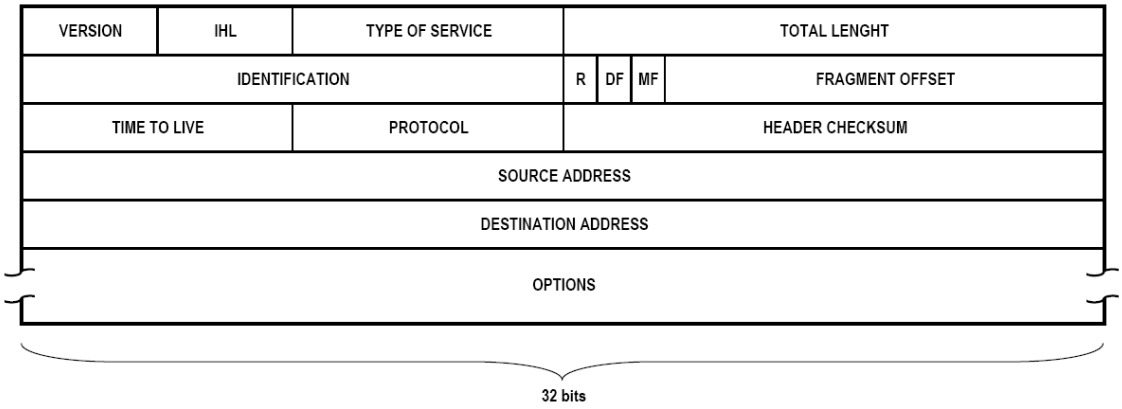


Figura 87 – Formato do cabeçalho IP.

Aos pacotes de dados vindos das camadas superiores é-lhes acrescentado um cabeçalho IP, permitindo desta forma a sua circulação e entregue através da rede (por exemplo Ethernet). O seu cabeçalho possui a estrutura representada na Figura 87 onde [IETF81a] [Stallings06] [Stevens94]:

- *Version* (4 bits) – identifica a versão do protocolo;
- IHL, *Internet Header Lenght* (4bits) – em palavras de 32 bits define o tamanho do cabeçalho situando entre o mínimo de 5 e máximo de 15;
- *Type of Service* (8 bits) – é um campo pouco utilizado e tem como objectivo reconhecer tipos de dados (ex. áudio, vídeo, ftp, etc) e consequentes performances;

- *Total Lenght* (16 bits) – corresponde ao tamanho total, ou seja, inclui cabeçalho mais informação;
- *Identification* (16 bits) – identifica o fragmento com o correspondente datagrama inicial (todos os fragmentos de um datagrama possuem o mesmo valor neste campo);
- FLAGS – campo com 3 bits que correspondem:
  - R,Reservada
  - DF, *Don´t Fragment* – quando o destino não consegue reassemblar;
  - MF, *More Fragment* – todos os fragmentos, à excepção do último levam este bit a 1;
- *Fragment Offset* (13 bits) – indica a posição do fragmento no datagrama;
- TTL, *Time to Live* (8 bits) – limita o tempo de vida de um pacote nos meandros da rede;
- *Protocol* (8 bits) – protocolo utilizado na camada superior;
- *Header Checksum* (16 bits) – verificação de erros no cabeçalho;
- *Source Address* (32 bits) – identifica a origem do pacote;
- *Destination Address* (32 bits) – identifica o receptor do pacote;
- *Option* (variável) – permite adicionar informação extra mas é pouco utilizado.

4.3.3.6. ENDEREÇAMENTO

Ao nível do protocolo IP existem duas versões: o IPv4 e o IPv6. A versão que actualmente se encontra em uso é o IPv4, o qual tende a ser substituído num futuro pela versão IPv6, devido à escassez de endereços. Os endereços IPv4 encontram-se divididos em cinco classes: A, B, C, D, e E. As classes A, B e C são as que definem os endereços primários sobre a rede TCP/IP. A classe D representa endereços para utilização em situações *multicasting*. A classe E encontra-se por definir pois está reservada para aplicações futuras [Beasley09] [Rodriguez01] [Stallings06] [Stevens94].

Classe	Gama de endereços
A	0.0.0.0 a 127.255.255.255
B	128.0.0.0 a 191.255.255.255
C	192.0.0.0 a 223.255.255.255
D	224.0.0.0 a 239.255.255.255
E	240.0.0.0.a 254.255.255.255

Tabela 6 – Classes do endereçamento IP.

Um endereço IPv4 consiste em 4 bytes dispondo assim de 32 bits. Estes bits, apesar de identificarem unicamente um endereço, também conseguem apresentar um endereço de rede correspondente aos bits iniciais e um endereço de máquina, correspondente aos últimos bits.

Classe	Bits da Rede	Bits da Máquina
A	8	24
B	16	16
C	24	8
32 bits ⇒ Endereço de Rede + Endereço de Máquina		

Tabela 7 – Número de bits da rede e máquina.

Cada um destes bytes aparece com uma notação decimal compreendendo valores desde os 0 até aos 255, separados por pontos. Na Figura 88 aparece a representação decimal (mais usual) e a representação em binário.

Decimal: 90 . 195 . 1 . 100

Binário: 01011010 . 11000011 . 00000001 . 01100100

Figura 88 – Exemplo de representação do endereço IPv4.

ENDEREÇOS ESPECIAIS

Da gama de endereços apresentada anteriormente existem algumas particularidades, que originam endereços especiais tais como [Beasley09]:

- 0.0.0.0 – Esta máquina/dispositivo
- 255.255.255.255 – *Broadcast* nesta rede
- 0.x.x.x – Máquina nesta rede
- x.255.255.255 – *Broadcast* em rede remota
- 127.x.x.x – *Loopback*

4.3.4. RS-232 (SÉRIE)

4.3.4.1. INTRODUÇÃO

A necessidade de interligar dois equipamentos, para troca de dados (ligação ponto-a-ponto), levou ao surgimento da comunicação série nos anos 60. Como tal, foi estabelecido por intermédio da *Electronic Industries Association* (EIA) a norma na transmissão de dados série [Hazen03] [Texas02]. Esta norma comporta especificações ao nível das tensões e dos tempos dos sinais, protocolo da troca de dados e características da ligação dos conectores.

Nos dias de hoje a designação correcta é EIA/TIA-232F, evidenciando a origem da norma (EIA/TIA) e as suas especificações (232F), no entanto é mais usual o seu reconhecimento pela designação RS-232. O índice F é resultado da evolução do RS-232 ao longo dos anos, ou seja, indica a sua actualização [Texas02].

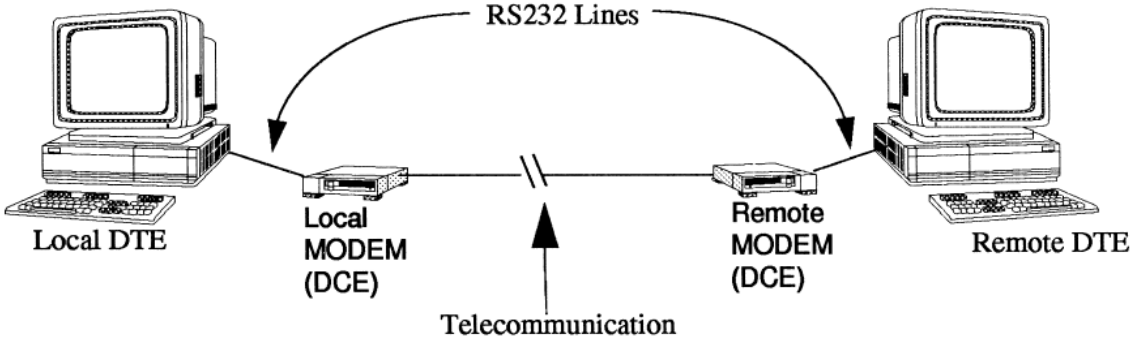


Figura 89 – Linhas de comunicação série RS-232 [Huq93].

4.3.4.2. ESTRUTURA

A base da comunicação série assenta numa estrutura em que os dados são enviados bit a bit sobre um meio transmissor, interligando emissor e receptor, exemplificado na Figura 89. Este meio pode ser um simples cabo condutor, onde transitam os sinais inerentes a esta comunicação mas, prevalecendo uma distância comportável com os padrões RS-232, ou seja, um valor de referência de 15m. Se a distância for longa a opção recai sobre o uso do *modem*,

permitindo desta forma recorrer às linhas telefónicas, transformando o sinal digital em analógico.

Daqui resultam duas terminologias: – *Data Terminal Equipment* (DTE) e *Data Communication Equipment* (DCE). O DTE é usualmente o dispositivo computorizado, enquanto o DCE incorpora os dispositivos de comunicação série, como é o caso do *modem* [Taltech11]. Na Figura 90 está representada a ligação entre um computador (DTE) e um receptor, neste caso modem (DCE).

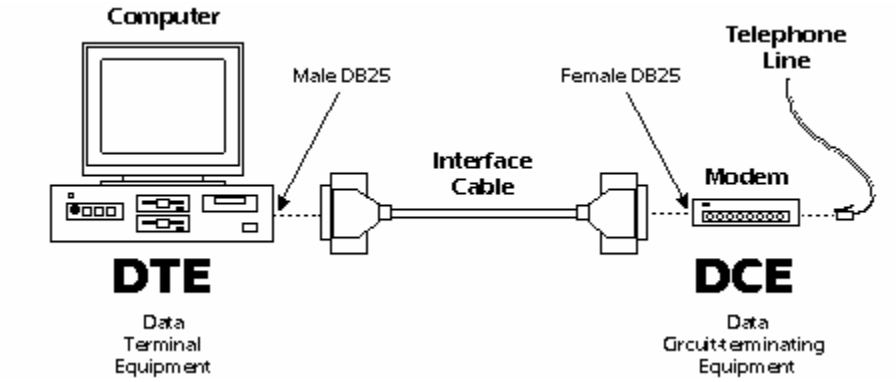
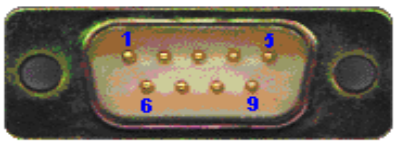


Figura 90 – Ligação entre DTE e DCE [Cami11].

4.3.4.3. SINAIS

A norma RS-232 contempla a existência de 25 sinais eléctricos na sua versão completa, entre os dispositivos DTE e DCE. Se quisermos ser mais precisos, destes 25 sinais eléctricos apenas 22 são utilizados, pois três sinais eléctricos não são usados. Este formato permite tirar todo o partido de uma ligação série, quando o DCE é um *modem* (ou similar), pois todos os sinais inerentes ao protocolo estão presentes.



Pino	Sinal	Descrição	Função
1	DCD	*Data Carrier Detect*	Controlo
2	RD	*Receive Data*	Dados
3	TD	*Transmit Data*	Dados
4	DTR	*Data Terminal*	Controlo
5	GND	*Ground*	Comum
6	DSR	*Data Set Ready*	Controlo
7	RTS	*Request To Send*	Controlo
8	CTS	*Clear To Send*	Controlo
9	RI	*Ring Indicator*	Indicador

Figura 91 – Ficha DB9 com os respectivos sinais.

No caso em que o DCE deixa de ser um modem, deixa de fazer sentido a existência de 22 sinais eléctricos, pois em termos práticos raramente os dispositivos DTE (usualmente computadores ou sistemas similares) necessitam mais que nove destes sinais [Huq93] [Taltech11] [Texas02]. Os nove sinais básicos estão apresentados na Figura 91.

Destes noves sinais, apenas três deles são cruciais para a implementação de uma comunicação série, sendo eles: - RD; - TD e GND. Com este três sinais é possível estabelecer uma comunicação série, assíncrona e *full-duplex*.

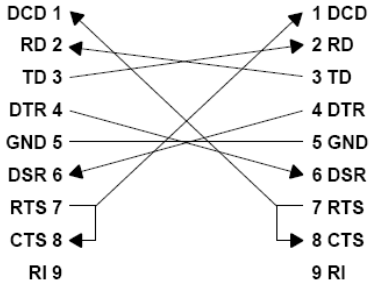


Figura 92 – Ligações de um cabo *null-modem* [Texas02].

Na Figura 92 representa as ligações necessárias paras se dois dispositivos DTE. Recorrendo a um cabo designado de *cross-over* (ou *null-modem*) é possível criar uma ligação, como se fosse um conjunto composto por um DTE e um DCE. Esta situação é muito recorrente, pois permite ligar dois computadores através de uma ligação série RS-232, mantendo as linhas de controlo, desde que a distância física contemple os requisitos máximos da cablagem.

4.3.4.4. CARACTERÍSTICAS

MECÂNICAS

As especificações originais para a cablagem RS-232 recomendam 15m como o comprimento máximo do cabo, no entanto, este é um valor de referência. Pois estas especificações foram revistas e passaram a caracterizar o comprimento do cabo através da sua carga capacitiva, verificada aos extremos do cabo. A obtenção do comprimento rege-se por expressão que, para além de outros

factores inerentes ao sistema, há outros externos com influência na definição do comprimento. Factores como: a taxa de transmissão a utilizar pelo sistema, a utilização, ou não, de malha de protecção no cabo, especificações técnicas dos fabricantes são condicionantes no comprimento do cabo [Taltech11].

Como o número de condutores que perfazem o cabo condutor pode variar, também os seus conectores têm algumas variações. Desde as três ligações mínimas para esta comunicação até às 22 ligações são estabelecidos diferentes conectores, sendo as tipologias mais usuais o DB-25 e DB-9. Outros conectores como: - RJ-11, RJ-12, RJ-45, DIN-8 pinos são passíveis de serem encontrados.

ELÉCTRICAS

As características eléctricas têm a particularidade de terem sido estabelecidas antes da existência dos níveis TTL, resultando em níveis lógicos diferentes dos tradicionais +5V e 0V (massa). Na Figura 93 encontram-se os níveis lógicos sendo que, para o nível lógico alto o valor da tensão pode variar entre +5V a +15V e, para o nível baixo o valor da tensão pode estar compreendido entre -5V a -15V. Os níveis de recepção foram estabelecidos de modo a permitir existência de quedas ao longo da linha, podendo estas atingirem os 2V. Assim, para a entrada o nível alto corresponde a valores dentro da gama +3V até aos +15V, enquanto para o nível baixo estes valores vão de -3V aos -15V. O nível lógico “0” enquadra-se nas tensões definidas de +5V a +15V, enquanto o valor lógico “1” representa valor das tensões compreendidas entre -5V e -15V. Isto obriga a uma inversão do sinal que transita no meio de comunicação, por intermédio de um circuito conversor. Tensões entre os +3V e os -3V não definem qualquer tipo de valor lógico. Os elementos receptores (*receivers*) devem ser concebidos para suportarem tensões até aos $\pm$25V, sem que para tal fiquem danificados.

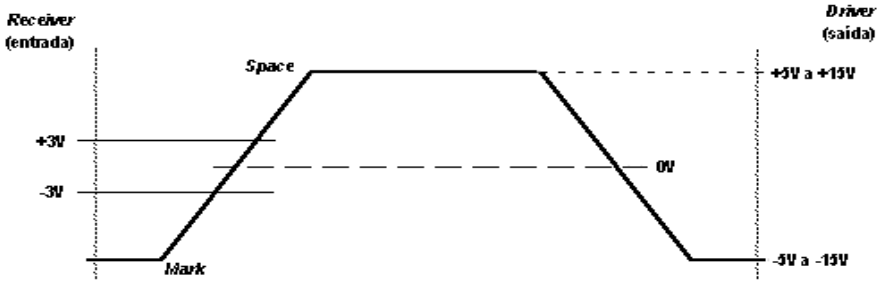


Figura 93 – Especificações dos níveis lógicos RS-232.

A impedância característica entre o *driver* de saída e o receptor está definido como sendo uma carga de valor compreendido entre os 3kΩ e os 7kΩ, por cada linha [Texas02].

Uma outra característica importante é o limite máximo do *slew rate*, à saída do *driver*. Esta limitação tem efeitos na redução do ruído de *crosstalk* entre os condutores do cabo de transmissão, pois quanto mais rápidas as transições do sinal mais ruído é gerado. O valor máximo permitido para o *slew rate* está definido como sendo de 30V/μs [Texas02]. Em consonância com o exposto, a norma RS-232 estabelece que a taxa máxima para a transmissão de dados tenha um valor de 20 kbps [Texas02].

4.3.4.5. DRIVERS/RECEIVERS

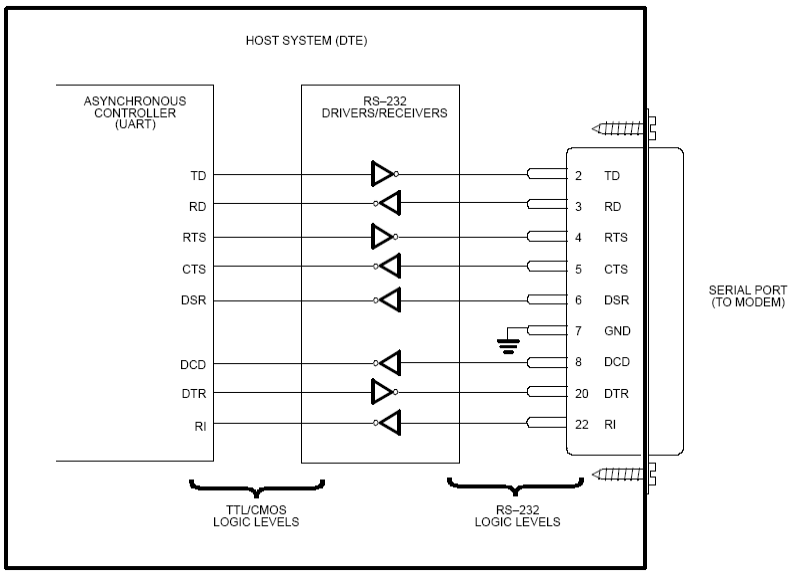


Figura 94 – Aplicação do conversor de níveis RS-232 / TTL [Dallas01].

Hoje em dia, uma larga maioria dos sistemas que utilizam o RS-232 não recorre aos níveis lógicos desta norma. Sendo os níveis lógicos RS-232 caracterizados por valores de tensão elevados, comparativamente aos sistemas onde estão inseridos, estes tornam-se inaplicáveis aos circuitos de transmissão (emissor/receptor). Assim, existe a necessidade de recorrer a determinados circuitos integrados (*drivers/receivers*) cuja função é efectuar uma translação dos níveis de tensão, com inversão do sinal (ver Figura 94). Com esta translação pretende-se que sejam exequíveis as trocas de dados entre os dispositivos emissores e receptores – DTE e DCE, através do meio transmissor que os une – cabos condutores. Quanto à lógica dos sinais convém salientar que a inversão da mesma (lógica negativa) só ocorre para as linhas de TD e RD. No caso de uma comunicação que utilize os 9 sinais, isto significa, que todos os restantes sinais (controlo) encontram-se implementados com lógica positiva, ou seja, uma tensão positiva na linha de controlo pressupõe um valor verdadeiro.

4.3.4.6. TRAMA

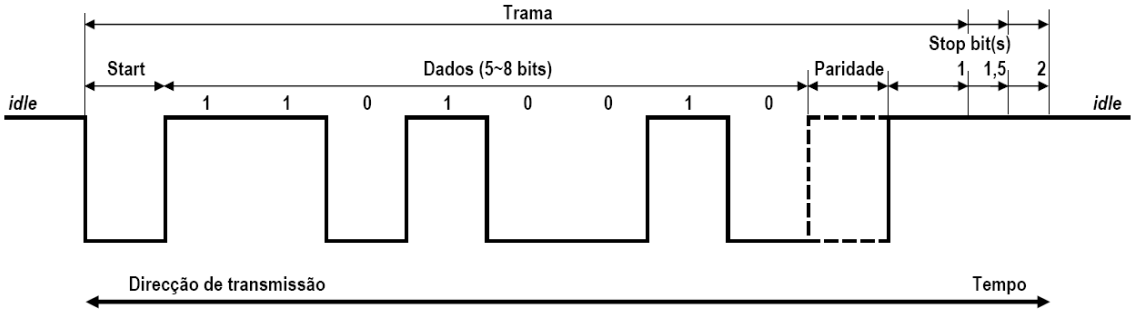


Figura 95 – Composição de uma trama série ao nível do bit.

A transmissão série tem por base agrupar um conjunto de bits, os quais formam a trama de dados série, conforme pode ser verificado na Figura 95. Descrevendo esta trama e por ordem temporal (direcção da transmissão), temos [Lava02]:

- Numa fase inicial a linha encontra-se num estado parado (*idle state*) ou a recepcionar o stop bit da última trama. Em ambos os casos o seu valor lógico encontra-se num nível alto;

- O início da comunicação surge através do aparecimento do *start* bit, que mais não é do que a passagem da linha ao nível lógico baixo;
- Os dados em si que podem ter em tamanho variável de 5, 6, 7 ou 8 bits (o mais usual são os 7 ou 8 bits), sendo primeiramente transmitido o bit menos significativo;
- A paridade dos dados como um bit opcional. Este bit de paridade, quando opção válida, insere à transmissão algum tipo de controlo de erros. Caso este bit (paridade) esteja activo, ele pode ser do tipo: par ou ímpar. Se o número de uns dos dados for par e a opção tiver recaído sobre a paridade par, significa que o bit paridade vai ser colocado a zero. Na paridade impar o processo é análogo salientando a vertente impar;
- A fase final é definida pelo *stop* bit, cuja duração temporal do bit pode ser variável. Esta duração pode ser: 1, 1,5 ou 2 stop bit. No final deste tempo a linha pode manter-se num estado *idle* ou iniciar a recepção do próximo *start* bit.

Para uma correcta leitura dos bits a sua amostragem é efectuada aproximadamente a meio da sua duração temporal.

No RS-232, como já foi verificado, a tensão na linha pode ter dois estados: - *mark* e *space*. Ao estado *mark* é atribuído o valor lógico alto e ao estado *space* é atribuído o valor lógico baixo [Hazen03]. A linha estando num estado *idle*, é equivalente ao estado *mark*.

4.3.4.7. SINCRONISMO

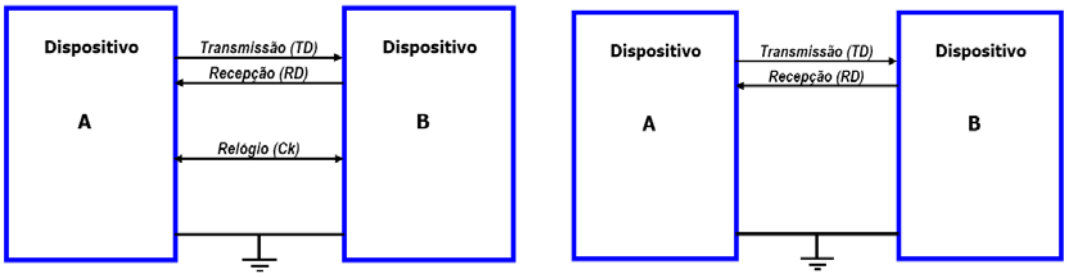


Figura 96 – Representação do modo síncrono e modo assíncrono.

A Figura 96 apresenta dois modos de operação, para a comunicação série RS-232, sendo estes: síncrono e assíncrono [Lava02]. Em modo síncrono os sinais são sincronizados pelo relógio, que caracteriza o tempo em que cada bit é enviado. Esta temporização do bit é conseguida por intermédio de uma linha de relógio que une os dois lados da comunicação, assim cada lado é conhecedor do próximo byte de dados.

No caso oposto, ou seja, em modo assíncrono o relógio do emissor é independente do relógio do receptor. Os tempos dos sinais são caracterizados pelas transições de tensões, as quais vão identificar o início e final de uma trama de dados série. No receptor e com base no seu relógio, a trama fica exposta a uma amostragem temporal, para o reconhecimento dos bits que são esperados por parte deste.

4.3.4.8. PORTA SÉRIE

Os dispositivos DCE e DTE, externamente, possuem pelo menos um conector que permite ligar o cabo que efectua a ligação entre ambos. Este conector é reconhecido como porta (ou porto) série. Então, este é o local físico onde os sinais fazem a transição do dispositivo para o cabo (interior/exterior) e vice-versa, o qual é composto por várias entradas e saídas (dependendo da configuração nem todas as entradas/saídas são utilizáveis)

UART

Internamente, a porta série, encontra-se associada a uma unidade gestora da comunicação série, a qual pode ser um simples integrado ou estar agregada a uma outra unidade (microcontrolador ou processador). Esta unidade é designada por USART (*Universal Synchronous Asynchronous Receiver Transmitter*), ou também designada de UART, quando apenas possui modo assíncrono [Lava02]. Assim, esta unidade produz e recebe os sinais eléctricos que se encontram definidos pelo protocolo RS-232. É conveniente referir que algumas destas unidades não possuem todos os sinais de controlo, na versão

mais simplista da comunicação série (3 ligações) os sinais de controlo são prescindíveis.

Sendo a UART a unidade que gere a recepção e, envio de dados (bytes), é usual estas encontrarem-se providas de uma memória auxiliar (*buffer*), possibilitando o armazenamento de alguns bytes que se encontram em trânsito. Este método permite obter uma melhoria significativa na eficiência entre o CPU e a porta série. Também minimiza o número de erros que podem ocorrer quando um novo byte chega na porta série, antes de o byte anterior ter deixado a UART (*overrun error*).

Uma outra funcionalidade da UART prende-se com o facto de esta possuir um conversor de dados (byte) paralelo/série e vice-versa. Na transmissão, a entrada de um byte na UART obriga à sua decomposição em bits, os quais são colocados na linha série. Na recepção executa a função inversa, isto é, recebe os bits individuais e trata de os aglomerar num byte, de modo a estarem à disposição do CPU.

4.3.4.9. TAXA DE TRANSMISSÃO

Devido às limitações técnicas da norma RS-232, já analisadas anteriormente, a máxima taxa de transmissão surge com um valor de 20 kbps. Apesar deste valor, na prática, são largamente utilizadas taxas de transmissão superiores, as quais podem atingir valores de 1 Mbps (comprometendo características da norma RS-232).

Como a taxa de transmissão máxima que pode ocorrer numa comunicação série RS-232 está intimamente ligada à USART e, não sendo esta unidade o factor decisivo, é um factor preponderante. As suas características são normalmente um factor limitativo na taxa de transmissão.

Dependendo do fabricante, os valores apresentados para a taxa de transmissão podem variar desde os 1200bps até aos 115200bps, usualmente em múltiplos da taxa inicial.

4.3.4.10. CONTROLO DE FLUXO

O controlo de fluxo permite verificar o estado da ligação entre o dispositivo DTE e o DCE, antes de ser transmitido qualquer dado. Este controlo de fluxo surge opcionalmente sob formato de *hardware* ou *software*, sendo que o primeiro recorre às linhas CTS/RTS (já apresentadas anteriormente). Por *software* a designação usual é o *Xon/Xoff*, aplicando dados de controlo nas linhas de transmissão/recepção (TD/RD). O controlo por *hardware* é preferível pois permite um melhor desempenho das comunicações. Caso não exista necessidade de ser efectuado o controlo de fluxo, este pode ser desactivado.

5. IMPLEMENTAÇÃO DO SISTEMA DE TESTE

5.1. ARQUITECTURA

A inserção da *Fez-Cobra* no sistema actual, permite alargar o campo de opções e aplicações ao nível do *hardware*, ou seja, é dado um salto qualitativo no desempenho do *hardware*. Contudo, algumas das opções disponibilizadas pela *Fez-Cobra* e necessárias neste projecto carecem de *hardware* suplementar, o qual foi suprimido com a sua implementação na placa de expansão.

Em complemento com este *hardware*, foi desenvolvida uma aplicação (*software*) que possa corresponder às necessidades decorrentes da execução de um teste. De igual forma foi necessário desenvolver uma outra aplicação, para o computador, que servisse de interface com o utilizador.

Na fase inicial apenas foi utilizado um *SMAC*, permitindo desta forma construir as bases da aplicação. Mais tarde, nas fases de ensaio do conjunto global, foram utilizados dois *SMAC*'s em funcionamento simultâneo. Apenas por uma questão logística não foi possível dispor de quatro *SMAC*'s. Convém ressalvar que, todo o desenvolvimento da aplicação teve em conta a utilização de quatro *SMAC*'s a funcionarem simultaneamente.

5.1.1. PLACA *FEZ-COBRA*

Figura 97 – Ligações estabelecidas com a *Fez-Cobra*.

A implementação da *Fez-Cobra* neste projecto vai proporcionar a ligação com os diferentes sistemas envolvidos no teste. As quatros ligações externas que fazem parte deste projecto são apresentadas na Figura 97, sendo:

- USB + Power – Estabelece a ligação ao *debbuger* e simultaneamente serve alimentação à *Fez-Cobra*;
- Ethernet – Ligação da *Fez-Cobra* ao computador por modo a interagir com a consola de utilizador;
- CAN – É o meio de comunicação com o painel de controlo. Através da recepção de mensagens por parte da *Fez-Cobra* vai ser possível saber o estado do contacto eléctrico das teclas;
- RS-232 – Ligação aos controladores *SMAC*'s, permitindo a troca de dados para a execução do teste e leitura de resultados do teste.

5.1.1.1. *SOFTWARE* DE DESENVOLVIMENTO

Antes de dar inicio aos trabalhos foi necessário efectuar a instalação do NETMF 4.0 e do GHI NET SDK (*Software Development Kit*), no computador onde iria ligar a *Fez-Cobra*. Para verificar a conectividade da *Fez-Cobra* ao computador, o NETMF SDK vem provido de uma pequena aplicação – MFDeploy, a qual permite efectuar o *ping* à placa. Deste modo é possível analisar imediatamente se a *Fez-Cobra* está a ser reconhecida pelo computador.

Nesta fase e, face ao desconhecimento das características que envolvem a programação da *Fez-Cobra*, assim como do ambiente de trabalho, foi necessário desenvolver pequenos programas, com o objectivo de proporcionar uma progressiva integração, quer no ambiente NETMF, quer no ambiente da placa *Fez-Cobra*.

Os programas iniciais serviram para lidar com as ferramentas que o NETMF dispõem, como a criação de um projecto, a interface em uso (consola ou emulador), dispor de janelas (*output*, *watch*, etc.) ou recorrer ao *debugger*. Simultaneamente permitiam um contacto com os elementos da *Fez-Cobra*,

designadamente a utilização das entradas e saídas, o uso do conversor A/D e o *display*.

Uma ferramenta muito útil ao longo do desenvolvimento da aplicação foi o *debugging*. Utilizando a porta USB para estabelecer comunicação entre a *Fez-Cobra* e o computador, é então possível recorrer ao *debugging*, o qual permite efectuar o controlo do programa executando-o passo a passo, receber dados, reconhecer valores de variáveis, entre outras opções.

Para correr um programa pode ser necessário incluir determinado tipo de *assemblies*[8], os quais vêm incluídos no NETMF SDK. Uma inclusão necessária de um *assembly* é aquela que identifica a *Fez-Cobra*, disponibilizando desta forma algumas funcionalidades da placa. Outras funcionalidades, como o uso da Ethernet ou o CAN estão incorporadas noutras *assemblies*, as quais também, necessitam de ser adicionadas.

5.1.1.2. ACTUALIZAÇÃO DO *FIRMWARE*

Aquando da aquisição da *Fez-Cobra*, a *GHI Electronics*, já anunciava uma actualização para breve do *firmware*. Contudo existia alguma incerteza na sua utilização, já que esta poderia não estar concluída antes do término deste projecto, inviabilizando desta forma o seu uso. Tal não se verificou e uma nova versão de *firmware* foi aplicada na *Fez-Cobra*.

Havia um interesse acrescido e, também, alguma curiosidade nesta nova versão para actualização, pois era apanágio da *GHI* o lançamento de um novo *driver* para as comunicações CAN. O *driver* actual já se encontrava apelidado de obsoleto pela *GHI*, com um conteúdo empobrecido, pelo que se afigurava uma tarefa árdua a sua utilização.

[8] Determinadas características de *hardware*/*software* tornam-se acessíveis através do *managed code*, recorrendo a métodos, propriedades ou variáveis das classes do C#.

Em Março de 2011 foi lançada a nova versão do *firmware*, a qual foi aplicada na *Fez-Cobra* através da sua actualização, não tendo surgido complicações neste processo. Todo o *software* da aplicação desenvolvido até à altura comportou-se de igual forma, ou seja, as mudanças de *firmware* não trouxeram alterações no funcionamento. Por vezes determinadas actualizações provocam algumas alterações nas aplicações mais antigas, e existia algum receio neste ponto. O novo *driver* vem substancialmente reforçado, com novos membros da classe e com novas características. Uma destas características, de uso quase imprescindível para a aplicação desenvolvida, é a possibilidade de utilização de eventos, até então não disponível.

5.1.2. PLACA DE EXPANSÃO

Apesar da placa *Fez-Cobra* englobar, intrinsecamente um conjunto de periféricos disponíveis para uso imediato, outros há que necessitam de *hardware* suplementar. Prevendo esta situação, foi adquirida uma placa de expansão que permite efectuar a expansibilidade da placa *Fez-Cobra*.

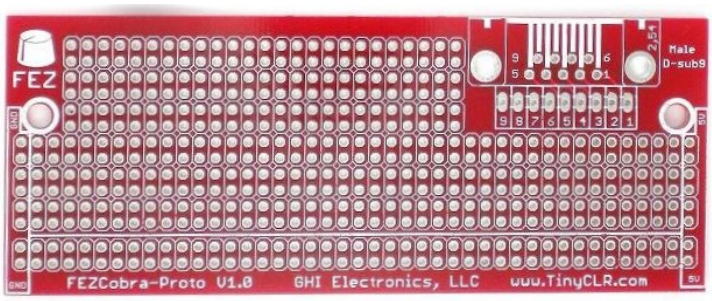


Figura 98 – Placa de expansão para acesso ao barramento I/O.

Esta placa, apresentada na Figura 98, encontra-se provida de uma linha perfurada coincidente com uma outra linha existente na placa *Fez-Cobra*, providenciando desta forma uma interligação das duas placas. Com esta ligação é conseguido, externamente, o acesso aos pinos do módulo EMX e dos seus periféricos. Para além deste factor, esta placa de expansão permite a implementação de pequenos circuitos auxiliares, como foi o caso deste projecto.

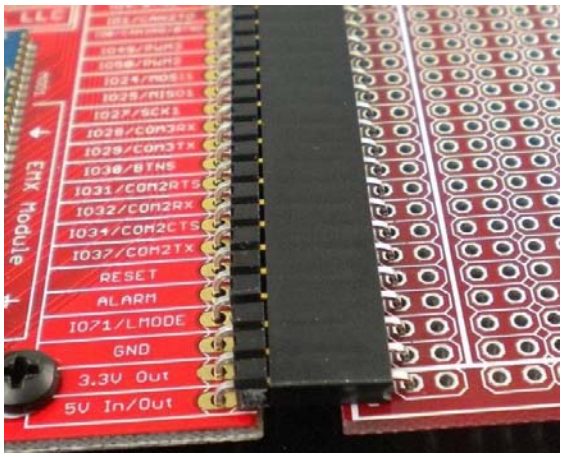


Figura 99 – Ligação entre a placa *Fez-Cobra* e a placa de expansão.

Unir esta placa com a placa *Fez-Cobra* é relativamente simples desde que, se utilize uma barra de pinos, macho e fêmea. A visualização desta situação é apresentada através da Figura 99.

CONECTORES DB9

Analisando as características físicas da *Fez-Cobra* constata-se que esta não possui conectores de ligação para as comunicações RS-232 e CAN. Assim foi necessário adicionar conectores DB9, aplicados na placa de expansão, para permitir ligar ao exterior estes dois meios de comunicação – RS-232 e CAN.

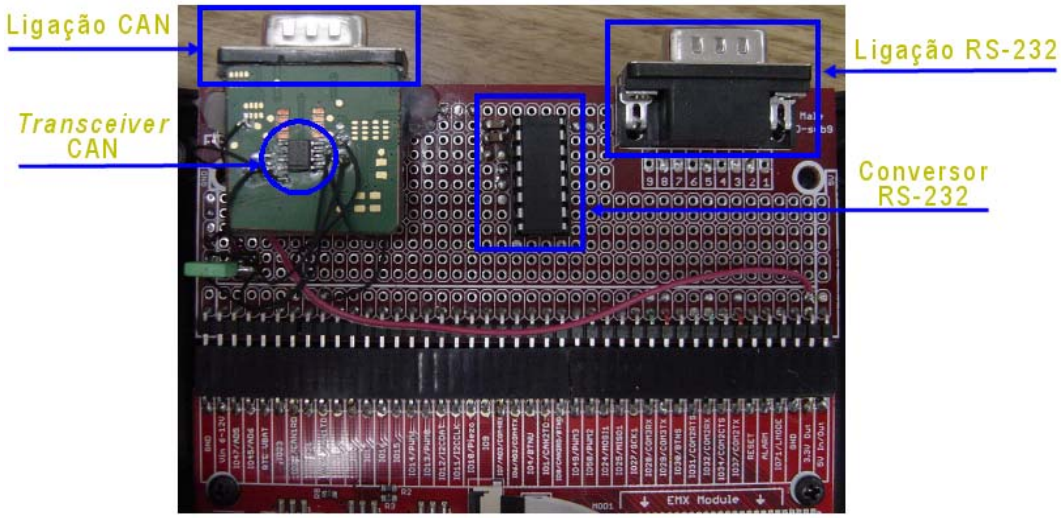


Figura 100 – Incorporação de componentes na placa de expansão.

A placa de expansão trás de origem uma localização especifica para o conector DB9, assim como o próprio conector, para o estabelecimentos de comunicação série RS-232. Com este tipo de comunicação podem ser suprimidas o número de linhas, sendo apenas necessárias duas linhas, uma para transmissão e outra para recepção de dados, acrescida da massa. Tendo em conta esta possibilidade, optou-se por aproveitar para ligar duas saídas das portas série da *Fez-Cobra* a este conector. Desta forma, o pino 2 e 3 correspondem ao TD e RD da COM1 e os pinos 7 e 8 correspondem ao TD e RD da COM2. Estas alterações, com a inclusão de *hardware*, estão apresentadas na Figura 100.

CONVERSOR RS-232

As saídas série da *Fez-Cobra* não vêm providas de conversores para níveis TTL. Para o efeito foi necessário acrescentar um conversor, sendo eleito o circuito integrado da *MAXIM* designado por *MAX3232*.

TRANSCEIVER CAN

Na comunicação CAN, também, foi necessário adicionar *hardware* suplementar, mais concretamente o *transceiver*. O fabricante da placa sugere a colocação do *transceiver* da *Texas Instruments*, cuja referência é SN65HVD230, sendo esta a opção empregue. Ainda há a acrescentar que o barramento CAN foi provido de resistências de terminação de 56Ω.

5.1.3. CONVERSORES CAN

LOW SPEED CAN

A *Fez-Cobra* é portadora de controlador CAN *High Speed* mas, os painéis de climatização utilizados neste teste e na indústria automóvel possuem controladores CAN *Low Speed*. Estas características impossibilitaram a ligação directa destes dois controladores ao barramento CAN.

Figura 101 – Conversor CAN *High Speed* para *Low Speed* [PEAK06].

A solução encontrada foi a introdução de um conversor de velocidade, que permitisse baixar a *High Speed* para *Low Speed*. Esta solução recaiu sobre o conversor designado por PCAN-TJA1054, da *Peak-System* [PCAN], apresentado na Figura 101.

A Figura 97 apresenta a ligação CAN à saída da *Fez-Cobra*, onde se encontra colocado o conversor, fazendo de interface entre esta (*High Speed*) e o barramento (*Low Speed*). De fácil aplicação, pois é comercializado com conectores DB9, o que permitiu a sua rápida integração no sistema.

USB / CAN

Aquando da fase de desenvolvimento da aplicação, que englobava o reconhecimento e tratamento de mensagens CAN, foi utilizado um conversor CAN / USB.

Figura 102 – Conversor CAN / USB [PEAK11].

Este conversor facilitou a compreensão inicial da troca de mensagens que, ocorria entre a *Fez-Cobra* e o painel de controlo, através de *software* de monitorização incluído com o conversor. Na Figura 102 está apresentado o conversor utilizado.

5.1.4. PROGRAMA *SMAC*

O controlador *SMAC* estabelece comunicação externa através de uma ligação RS-232. Esta ligação permite duas funcionalidades: uma é a inserção de um programa, para ser executado; outra é a troca de dados com o exterior. Assim, primeiramente é necessário enviar um programa, o qual é possível com recurso ao *Hyperterminal* do *Windows* (com os respectivos parâmetros de comunicação configurados). O programa é um ficheiro de texto com as menmónicas da linguagem do *SMAC*. Utilizando o mesmo meio é possível enviar comandos isoladamente para o *SMAC*, obtendo resposta caso exista.

Inicialmente foram executados alguns comandos e, posteriormente, alguns programas simples resultando num conhecimento mais pormenorizado sobre a estrutura de comandos e do modo de operação do *SMAC*.

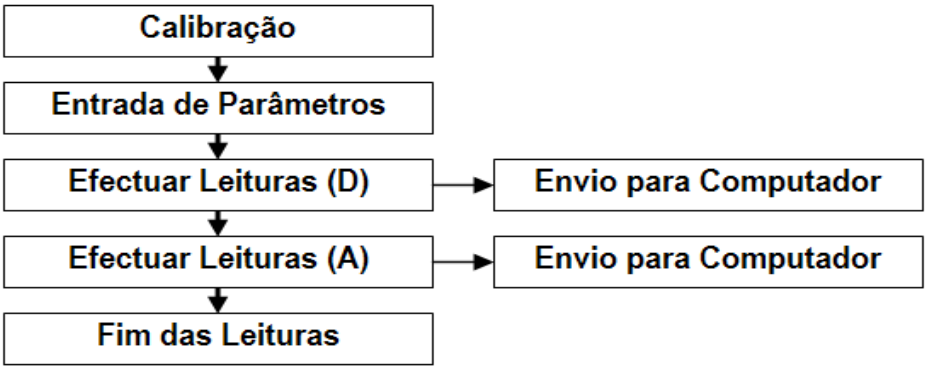


Figura 103 – Estrutura do *software* implementado no *SMAC*.

O programa implementado no controlador *SMAC*, com esboço representado na Figura 103, proporciona que, no arranque do sistema o *SMAC* efectue a calibração do seu sistema de movimentação (haste), através de rotinas próprias para o efeito. No final dessa fase envia uma mensagem "*Start?*", via RS-232 para a *Fez-Cobra*, indicando o sucesso da calibração e simultaneamente que o *SMAC* se encontra na posição *home*. Em caso de resultado negativo é enviada a mensagem correspondente ao erro.

Ao enviar a mensagem "*Start?*", o *SMAC* fica a aguardar que a *Fez-Cobra* envie os parâmetros de configuração do teste. Os parâmetros são os seguintes:

- "*Rapid Pos*" – Corresponde à posição rápida, originando que o *SMAC* se movimente rapidamente, até atingir a coordenada referenciada neste parâmetro. Neste teste, a coordenada localiza-se imediatamente antes do contacto com a tecla. As características rápidas são intrínsecas ao programa inserido no *SMAC*;
- "*Acceleration*" – Quando inicia o movimento o *SMAC* é provido de uma aceleração definida por este parâmetro;
- "*Velocity*" – Igual ao anterior mas neste caso referente à velocidade;
- "*Test Force*" – Indica a força máxima, aplicada ao teste, com a qual o *SMAC* efectua o movimento;
- "*Max Force*" – É um parâmetro de controlo de força máxima, servindo para detectar o final do curso da tecla. A partir deste momento o *SMAC* inverte o movimento. A *força máxima* tem de ser sempre inferior à força do teste, garantindo desta forma que este parâmetro é atingido.
- "*Test Increment*" – Corresponde ao incremento que é efectuado na deslocação, ou seja, é a distância ou avanço entre dois pontos consecutivos do movimento;
- "*Wait Increment*" – O tempo que permeia entre os incrementos no teste.

Após a recepção do último parâmetro o *SMAC* inicia a leitura da força e deslocamento. Numa primeira fase no sentido descendente (D) e de seguida no sentido ascendente (A). O sentido descendente é interrompido quando a força ultrapassa o valor do parâmetro máxima força ("*Max Force*") e, no sentido ascendente quando atinge o valor da posição rápida ("*Rapid Pos*"). Por cada leitura efectuada é transmitido, via porta série, o resultado (força e deslocamento) para a *Fez-Cobra*, cabendo a esta guardar os valores num *array* (unidimensional) de força e num outro de deslocamento. Por cada par de dados (força e deslocamento) transmitidos para a *Fez-Cobra* pelo *SMAC*, este fica a

aguardar por um tipo de *acknowledge*, o qual não é mais do que um *carriage return* (“\r”).

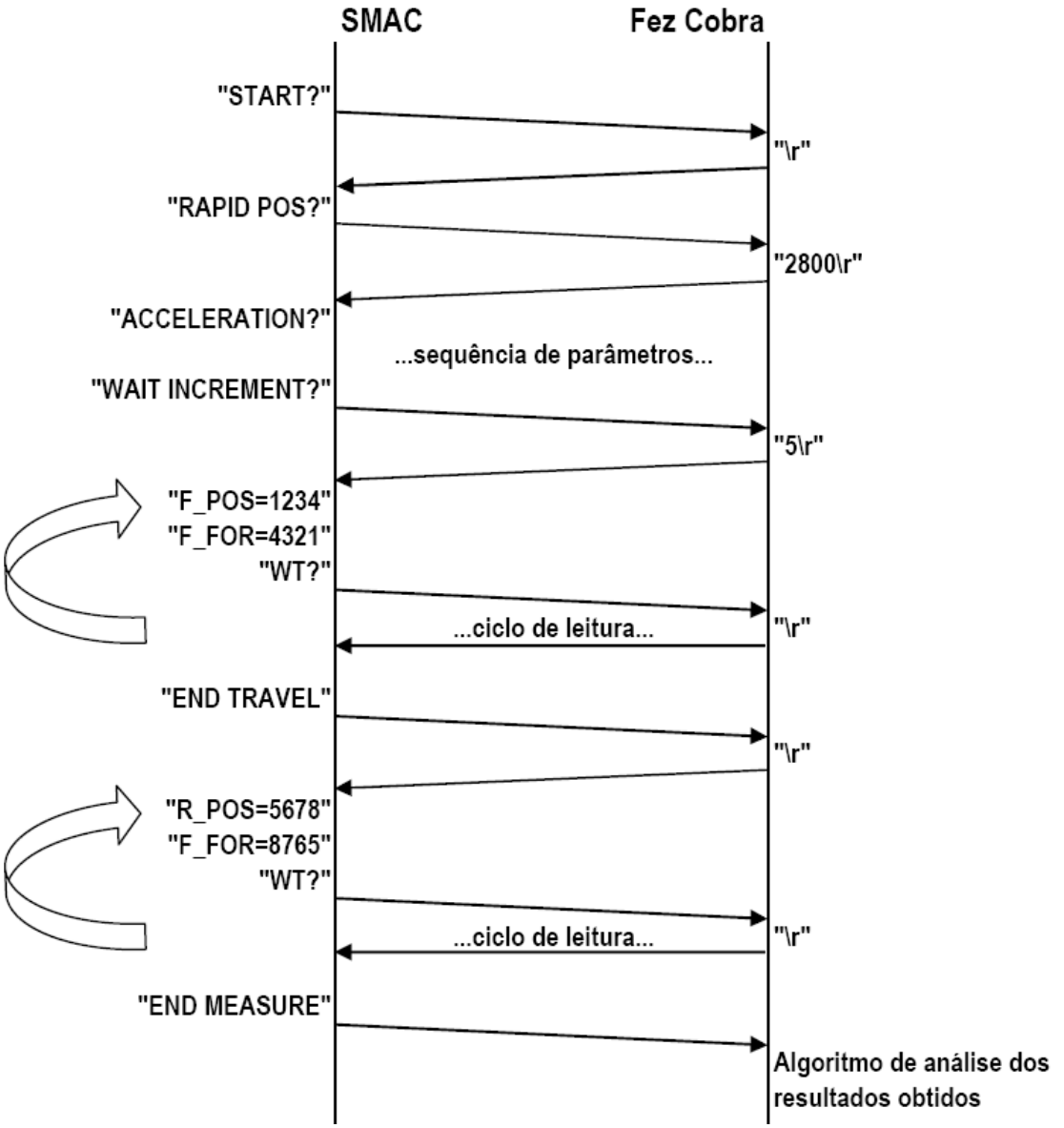


Figura 104 – Sequência na troca de dados entre o *SMAC* e a *Fez-Cobra*.

A finalização do teste na fase descendente é indicada à *Fez-Cobra* através do envio da mensagem “*End Travel*”, assim como a fase ascendente. No entanto, a finalização completa do teste é correspondida com a transmissão da mensagem “*End Measure*”, para a *Fez-Cobra*. A partir deste momento o *SMAC* posiciona-se em *home*, repetindo a sequência iniciada com o envio do comando “*Start?*”. A Figura 104 representa de forma resumida a sequência da troca de dados entre os dois elementos, *SMAC* e *Fez-Cobra*.

5.1.5. DESENVOLVIMENTO DA APLICAÇÃO

Estruturado o formato do *hardware* a aplicar neste projecto, o passo seguinte foi o desenvolvimento da aplicação. No início, para este desenvolvimento foram abordadas algumas considerações, nomeadamente:

- A estrutura da aplicação deve ser versátil de modo a poder corresponder a novas variantes. Com isto pretende-se que, futuramente esta aplicação possa ser enquadrada numa outra situação de teste, sem que seja necessário efectuar alterações profundas;
- A concepção da aplicação deve evitar acontecimentos propícios ao bloqueio do sistema de teste, pois a ocorrência de um erro não deve colocar o sistema num estado de bloqueio, por tempo indefinido mas apenas temporariamente. Em alternativa, aplicar situações de excepção, podendo estas lidar com determinados tipos de erros e em função deles assumir tarefas variadas. Uma situação de bloqueio obriga à paragem da linha de produção, de modo a ser efectuado o *reset* na *Fez-Cobra* (iniciar a aplicação). Esta situação acarreta consideráveis consequências, ou seja, diminuição da produção e consequentemente aumentos de custos. Um exemplo simples de erro e, que esporadicamente pode ocorrer, é a falha de comunicação no envio de dados para o computador (cabo desconectado). Evitar esta situação é criar alternativas de saída (*escape*);
- Recurso a *threads*, é uma das medidas que permite tornar versátil e não bloqueante um sistema. Isto é, o processo de execução do teste para um determinado *SMAC*, não pode aguardar que um outro *SMAC* termine o seu teste. Lançando *threads* sob determinados mecanismo (por exemplo: funções) permite a execução simultânea;
- Utilizar as potencialidades da linguagem C#. Conforme já referido anteriormente, a *Fez-Cobra* utiliza o .NET Micro Framework, com recurso da linguagem de programação C#. Esta linguagem acrescenta uma mais-valia, num sistema que se quer dinâmico e

versátil, através do uso de classes. Inicialmente foram estipuladas as classes base para uma optimização do código. Estas classes uma vez instanciadas passam a definir o código dinamicamente.

5.1.5.1. ETAPAS

Um factor preponderante, considerando o desconhecimento inicial no desenvolvimento de aplicações para a *Fez-Cobra*, foi o estabelecimento de etapas no desenvolvimento da aplicação para o projecto. Com isto, pretendeu-se garantir a executabilidade da fase actual para que, a fase ulterior pudesse decorrer sem retrocessos. De seguida são apresentadas as cinco etapas, consideradas fundamentais e três menos relevantes sendo que, todas elas foram os objectivos parciais do desenvolvimento da aplicação.

- *Software* da *Fez-Cobra*;
- Comunicação RS-232;
  - Recepção e manipulação de dados;
- Comunicação Ethernet;
  - Transferência de dados e comandos
- Consola do utilizador;
- Comunicação CAN;
  - Análise de mensagens CAN.

Das etapas anteriores, a consola do utilizador é a única que não pertence ao desenvolvimento da aplicação para a *Fez-Cobra*. Esta foi uma etapa que foi sendo desenvolvida paralelamente, face às evoluções surgidas na aplicação da *Fez-Cobra*.

5.1.5.2. ESTRUTURA

Convergindo as etapas anteriores para uma das características do C#, as classes, foi o ponto de arranque no desenvolvimento do código na *Fez-Cobra*. Assim a estrutura base encontra-se alicerçada em classes, sendo que três

delas são fundamentais para assegurar o regular funcionamento da aplicação. Estas classes são:

- Série – Tem como ponto central o estabelecimento da comunicação série com o *SMAC* e a *Fez-Cobra*, para a recepção e transmissão de dados entre ambos. Inerente a esta actividade surge o tratamento das leituras recebidas. O resultado final do teste é gerado em função do resultado das leituras e da leitura das mensagens CAN (contacto eléctrico). No código surge definida como *SerialCom*.
- Ethernet – Classe que coloca em comunicação o computador com a *Fez-Cobra*, ou seja, trata da comunicação via Ethernet. Esta classe pretende ser uma interface de comunicação entre o computador e o *SMAC* na troca de dados. A sua identificação no código é de *HandleClients*.
- CAN – Uma outra classe que estabeleça a ligação entre a *Fez-Cobra* e o painel. Desta forma toda a estrutura de leitura e envio de mensagens CAN é aqui tratada. Em coordenação com a classe série, é enviada para esta, o resultado da leitura da mensagem CAN referente à tecla em teste. Esta classe encontra-se designada no código por *CANCom*.

Para além destas três classes aqui apresentadas existem mais duas classes. Uma destas classes surge por inerência do C#, ou seja, é aquela que contém o *main()*, designado por *Program*. A outra classe foi criada com o objectivo de conter parâmetros e valores globais ao programa, isto é, acessíveis em qualquer parte do programa. Esta classe está designada como *Global*.

A classe *SerialCom*, como se encontra agregada às ligações estabelecidas com os *SMAC*'s tem de ser dinâmica, isto é, o mesmo formato de código tem de servir para trabalhar com qualquer uma das portas série (num máximo de quatro portas devido às limitações da *Fez-Cobra*). Uma das ferramentas do C# é instanciar objectos independentes mas referentes à mesma classe. Esta potencialidade foi usada como a solução encontrada para se conseguir que, o mesmo código sirva as diferentes portas série. Mais adiante a referência à

porta série deixa de ser contextualizada como tal, passando a ser reconhecida como canal, ou seja, um *SMAC* vai estar ligado a um canal da *Fez-Cobra*.

Em paralelo com este desenvolvimento e, devido à necessidade do utilizador interagir com a *Fez-Cobra* (configuração e execução do teste), com os *SMAC*'s, obter resultados e apresentar dados graficamente, foi necessário desenvolver uma consola de trabalho no computador.

5.1.5.3. COMUNICAÇÃO RS-232

A comunicação série começou por ser estabelecida entre o computador e a *Fez-Cobra*. Esta ligação inicial tinha como objectivo retirar conhecimentos do modo em como a *Fez-Cobra* se comportava na comunicação série. Assim, partindo da vantagem do computador ter disponível o meio físico (porta série) e de possuir uma aplicação para comunicação série (*Hyperterminal*, do *Windows*), era possível da parte deste estabelecer a comunicação série. Desenvolvendo uma pequena aplicação para a *Fez-Cobra* em que, simplesmente recebia os dados e fazia o "eco" dos mesmos, tornou a tarefa de análise da comunicação série bastante facilitada. A partir deste momento e, sabendo-se que a comunicação série estava operacional, foi iniciado o desenvolvimento do programa para colocar a *Fez-Cobra* a comunicar com o controlador *SMAC*. Posteriormente este programa seria inserido, com as adaptações necessárias, na aplicação da *Fez-Cobra*

ESTRUTURA GERAL

Após o sucesso no estabelecimento da comunicação série entre a *Fez-Cobra* e o computador avançou-se para a ligação ao *SMAC*. Em virtude dos controladores *SMAC*'s já estarem a efectuar testes nas teclas e, com isto, já possuírem um programa a funcionar, a adaptação à comunicação série recaiu sobre a *Fez-Cobra*.

Toda a estrutura do código inerente à comunicação série, pelo menos a fundamental, está implementada na classe *SerialCom*, tal como já foi referido.

Começando desde logo pelos parâmetros de configuração da porta série, assim como da sua abertura. Como o controlador *SMAC*, já se encontra com os parâmetros da comunicação definidos, e não existindo necessidade de alteração, levou à *Fez-Cobra* ser configurável de igual forma.

Um aspecto essencial, para o bom desempenho da comunicação, está assente numa *thread* que foi criada para uma função desta classe. A sua tarefa é trabalhar sob um ciclo infinito, com base na sequência em que o teste é realizável. Assim: desde o envio dos parâmetros do teste para o *SMAC*, à leitura dos dados na porta série (até à finalização dos mesmos), ao tratamento dos valores das leituras, é efectuado dentro desta função a correr sobre uma *thread*.

A recepção dos dados na porta série ocorre por evento, libertando deste modo a necessidade de se efectuar o *polling* constante à porta, o qual ocuparia tempo e recurso da aplicação.

DADOS E COMANDOS

Ao ser conseguida a comunicação entre a *Fez-Cobra* e o controlador *SMAC*, havia necessidade de dar continuidade à estrutura existente no programa implementado no *SMAC*, desenvolvendo uma aplicação condizente para a *Fez-Cobra*. Esta aplicação deve ser enquadrada, tendo em conta o funcionamento do programa do *SMAC* apresentado anteriormente. O ponto 5.1.4 aborda a estrutura do programa que corre no controlador *SMAC*, onde resumidamente:

- Recebe os parâmetros antes do início do teste;
- Dá continuidade à ordem de início de teste, com a consequente execução do mesmo;
- Envia os valores das leituras obtidas;
- Envia a finalização do teste e, seguidamente, fica apto para executar um novo ciclo de teste.

A aplicação da *Fez-Cobra* deve estar preparada para que, no atendimento de uma ordem de início do teste vinda da consola, verifique se possui no *buffer* de

entrada uma mensagem *"Start?*". Esta mensagem chega à *Fez-Cobra* vinda do *SMAC* e, basicamente corresponde à informação de prontidão por parte do *SMAC*. A partir do momento em que lê esta mensagem pode enviar os parâmetros de configuração do teste (posição rápida, aceleração, etc.), à medida que eles vão sendo solicitados por parte do *SMAC*. Todas estas trocas de dados estão limitadas temporalmente, desta forma é evitado o bloqueio do sistema pela ocorrência de alguma falha neste processo.

Iniciado o teste, os valores das medições vão ser recepcionados e guardados num *array* na *Fez-Cobra*. A recepção tem condicionantes originadas pela composição da mensagem, pois um valor é constituído por vários bytes (um valor *double* corresponde a 8 bytes). A forma encontrada para diferenciar a recepção de um pedido (ex: "*Start?*", "*Rapid Pos?*, etc.") na *Fez-Cobra*, de um valor de leitura ou final de leitura (ex.: "*End Measure*"), está na finalização da mensagem. O último carácter dos pedidos termina com um ponto de interrogação ("?"), ao invés, valores ou final de leituras possuem como último carácter um *line feed*("\n").

TRATAMENTO DAS LEITURAS

Conforme o apresentado no ponto 5.1.4, a aplicação da *Fez-Cobra* efectua a recolha dos valores das leituras, enviados pelo *SMAC*. Antes de serem guardados estes valores, da força e deslocamento, são sujeitos a conversão. O armazenamento destes valores corresponde a ter dois *arrays* unidimensionais de igual comprimento: um para a força e outro para o deslocamento. Os valores contidos nos dois *arrays* têm correspondência posicional.

Aos valores do deslocamento versus força guardados nos *arrays*, é imprescindível aplicar um filtro de modo a serem reduzidos aos elementos essenciais. Nem todos os valores recolhidos são tidos em conta na análise do teste, pois na fase inicial e na fase final existem alguns valores que não têm interesse.

Ao iniciar-se um teste, o *SMAC* executa um movimento rápido para uma posição próxima da tecla. A partir desta posição começa a recolher valores mas, tais valores não têm significado, pois não se exerce força alguma sobre a tecla. Os valores obtidos nesta fase têm de ser eliminados. Na Figura 105 está ilustrada esta situação, ocorrendo entre o posicionamento inicial (Si) e o posicionamento considerado de contacto (So).

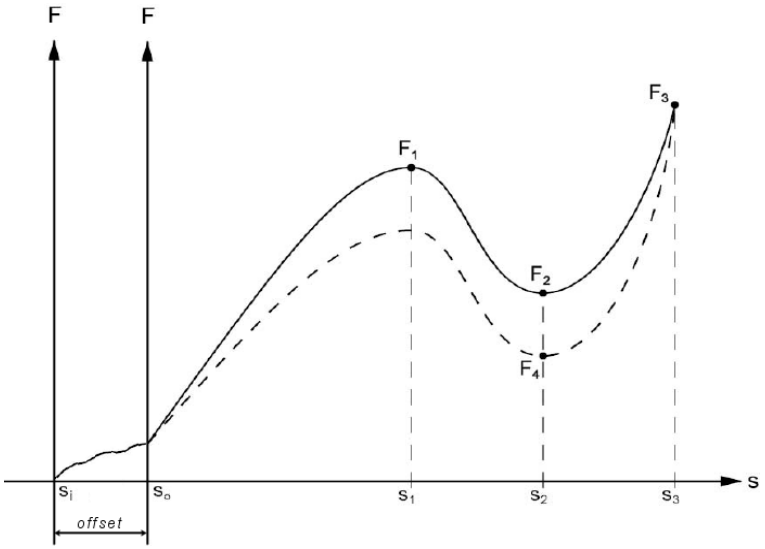


Figura 105 – *Offset* no deslocamento.

A força exercida na tecla, a partir da qual deve ser considerada como válida, tem de possuir um valor igual ou superior a 0,1N, para o teste em causa. Como é óbvio, todos os valores anteriores não são considerados, passando este par de valores força e deslocamento a serem os valores iniciais. Se esta posição for considerada a origem posicional (So) do curso da tecla, então as restantes posições têm de ajustadas por um valor de offset correspondente So-Si. Convém salientar que a posição cujo valor é 0 corresponde à posição de repouso do *SMAC* (posição *home*).

Na fase final vai acontecer uma situação semelhante mas agora o factor eliminatório é o posicionamento. Tendo sido aplicado o offset a todos os valores do deslocamento, aqueles que ficarem acima do contacto com a tecla vão ter valores negativos. Estes deslocamentos negativos são valores que não têm interesse.

O valor da força máxima (F1) é obtido por comparação entre dois valores consecutivos. Quando o valor actual for inferior em 0,25N, então foi encontrado o valor F1. O valor F3 já não necessita de um factor diferenciado, bastando que o valor actual seja inferior ao último valor.

O formato para a obtenção do valor mínimo é idêntico ao anterior porém, neste caso para serem detectados ambos os valores das forças F2 e F4, o valor actual da força tem de ser superior em 0,35N o valor anterior.

5.1.5.4. COMUNICAÇÃO ETHERNET

Criar uma ligação cliente/servidor entre a *Fez-Cobra* e o computador foi o ponto de partida para o estabelecimento da comunicação Ethernet. Numa primeira fase foi utilizado um *browser* e, mais tarde, com o desenvolvimento de uma pequena aplicação a servir de interface, permitiu retirar as potencialidades necessárias para o uso da Ethernet na *Fez-Cobra*.

O código da comunicação Ethernet, desenvolvido na aplicação *Fez-Cobra* encontra-se praticamente todo implementado numa classe designada por *HandleClients*, à excepção de uma pequena estrutura integrada no *main*. O código encontra-se a correr como servidor, permitindo o estabelecimento de ligação por parte de diferentes computadores (clientes). Daqui resulta a implementação de um pequeno trecho de código no *main*, onde se encontram as configurações da porta Ethernet para a abertura do *socket*, seguindo-se a escuta na porta para o estabelecimento de ligação. Do lado do cliente apenas é necessário conhecer as características da porta Ethernet da *Fez-Cobra*, para se estabelecer a comunicação. Como foi desenvolvida uma consola de interface para interagir directamente com a aplicação da *Fez-Cobra*, esta já inclui as especificidades da porta Ethernet quando se executa a ligação.

Como já vimos anteriormente, o funcionamento da aplicação assenta em dois modos distintos de operação: o modo transparente e o modo autónomo (abordado no ponto 3.2.4). Deste modo a primeira troca de dados, após

abertura do *socket*, surge na definição do modo de operação. Por defeito encontra-se no modo autónomo, canal 2.

Ao ser estabelecida uma ligação Ethernet, vai ser criada uma *thread* para lidar com toda a troca de dados resultantes da porta Ethernet. O corpo central desta *thread* tem como objectivo distinguir se, os dados recebidos correspondem a algum dos do modo de operação ou outro tipo de dados. Consonante os dados recebidos pela *Fez-Cobra*, através da porta Ethernet, estes vão desencadear duas formas de tratamento da informação. Assim, se os dados recebidos correspondem a uma alteração no modo de operação, leva a que esta informação seja actualizada de imediato no *SMAC*. No caso dos dados reportarem a outro tipo de informações (por exemplo os parâmetros), esta é retida na *Fez-Cobra*. Ao ser dada ordem para se iniciar o teste, toda a informação retida na *Fez-Cobra* é transmitida para o *SMAC*. A causa de apenas a mudança de modo ser enviada de imediato para o *SMAC*, advém de um problema surgido no desenvolvimento e abordado adiante.

Quando é dada ordem para a execução de um teste é despoletada uma outra *thread*, cujo objectivo é permitir que outros testes dos diferentes canais possam ocorrer em simultâneo. Nesta situação é evitada a exclusividade de recursos através da partilha dos mesmos.

Encontrando-se várias *threads* a correr e com a necessidade de passagem de parâmetros entre classes, originaram algumas dificuldades na executabilidade do código. No entanto, utilizando alguns conceitos inerentes ao C# foi possível ultrapassá-los.

TRANSIÇÃO MODO TRANSPARENTE PARA AUTÓNOMO

A mudança de modos de operação não pressuponha nenhum problema mas de facto originou erro. Passando do modo transparente para o modo autónomo, sucedia a ocorrência de erro no movimento do *SMAC*. Estando o *SMAC* em modo transparente, qualquer tipo de comando pode ser executado manualmente. Dependente do último comando executado, este pode levar o

SMAC à posição *home* ou não. Caso não leve à posição *home* e, posteriormente seja mudado o modo para autónomo, vai dar origem a erro na execução do teste. No modo autónomo o *SMAC* tem de estar na posição *home* para que as leituras sejam correctas, caso contrário o início e o offset encontram-se incorrectos, originando erros nas leituras.

PROTOCOLO DA COMUNICAÇÃO

O processo de troca de dados, via Ethernet, entre a consola e a *Fez-Cobra* está definida no seguinte protocolo:

- Modos de operação: iniciam pelo carácter "%". Exemplificando:
  - "%MT" – modo transparente;
  - "%MA,*x*" – modo autónomo, em que o *x* representa o canal onde se pretende actuar (ligação ao *SMAC*). O *x* pode tomar os valores 1, 2, 3 ou 4.
- Comandos e parâmetros: iniciam pelo carácter ">". Exemplificando:
  - ">MasterReset" – executa o *reset* de valores e parâmetros em todos os canais;
  - ">TestResul, *x*" – obtém o resultado do teste referente ao canal *x*;
  - ">StartTest, *x*" – dá inicio ao teste no canal *x*;
  - ">Param, *x*, *y*, *z*" – configuração de parâmetros em que *x* indica o canal, *y* é o nome do parâmetro (por exemplo: *Rapid Pos, Accelaration*) e *z* o valor do parâmetro.

Sempre que sejam enviados dados para a *Fez-Cobra*, esta retorna uma resposta em função desses mesmos dados, sendo que:

- Modos de operação: é enviada a mensagem recebida acrescida do carácter "%". Exemplificando:
  - "%MT%"; "%MA,*x*%".
- Comandos e parâmetros: neste caso é dependente do tipo de mensagem recebida mas termina sempre no sinal oposto ao do envio "<". Se existir algum comando que careça de indicação do sucesso (ou não) da operação, o símbolo anterior é precedido de 1 para sucesso ou 0 para insucesso. Exemplificando:

- ">MasterReset<", ">Rst, *x*<" – não carece de indicação de sucesso, como tal faz retorno da mensagem acrescida do sinal final;
- ">TestResul, *x*, *n*<"– carece de indicação de sucesso. Se o resultado for positivo n assume o valor 1, caso contrário n assume o valor 0;
- ">StartTest, *x*, *n*<" – idêntica ao anterior mas neste caso aplicável à execução do teste;
- ">Param, *x*, *y*, *z*<" – não carece de indicação de sucesso, como tal faz retorno da mensagem acrescida do sinal final.

5.1.5.5. COMUNICAÇÃO CAN

Utilizando o mesmo método de desenvolvimento das comunicações anteriores, começou-se por criar uma pequena aplicação de ensaio, antes de implementar uma estrutura para comunicação CAN na aplicação deste projecto. Uma dificuldade acrescida na implementação deste pequeno código para ensaio, residia na inexistência de um meio externo que, pudesse analisar as mensagens no barramento CAN. Com esta análise pretendia-se constatar que as mensagens lidas pela *Fez-Cobra* eram as que circulavam no barramento.

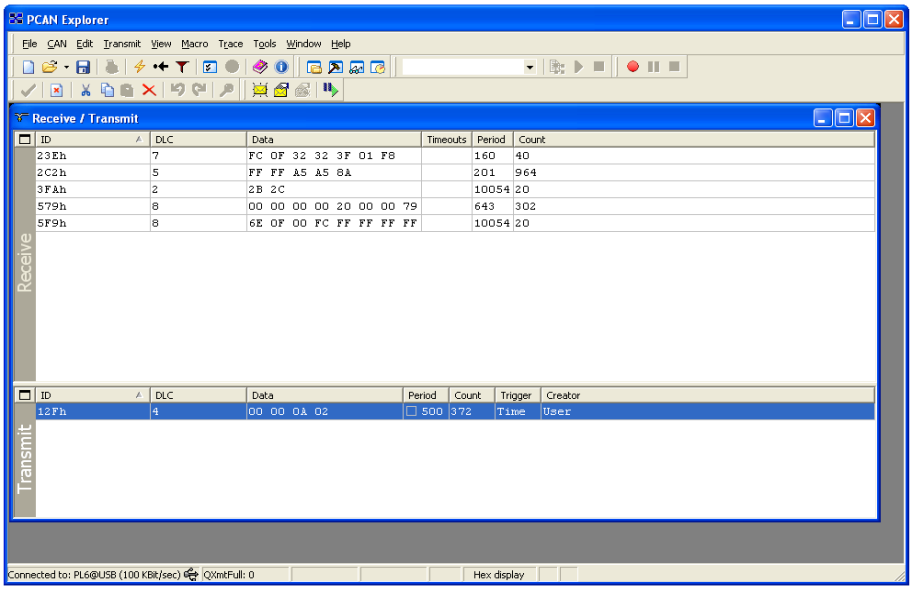


Figura 106 – Monitorização de mensagens CAN através do PCAN [PEAK11].

De modo a ultrapassar esta dificuldade recorreu-se a um analisador de tráfego do barramento CAN. Este analisador é composto por duas vertentes:

- *Hardware* – A sua implementação levou à necessidade de criar uma ligação do barramento CAN com o computador. Não estando provido, o computador, de um controlador (porta) CAN, houve necessidade de efectuar a conversão de comunicação. Tal facto recaiu sobre um dispositivo conversor que, possibilitou a ligação do barramento CAN ao computador via porta USB, ou seja, um conversor CAN/USB (Figura 102);
- *Software* – Uma aplicação fornecida pelo fabricante do conversor (Figura 106), onde são visualizadas no monitor do computador as mensagens que circulam no barramento. Para além disto, o utilizador dispõem de um conjunto de ferramentas que lhe permite compor e parametrizar as mensagens a enviar.

Montado este conjunto, foi possível verificar que, as mensagens lidas pela *Fez-Cobra* eram coincidentes com as que estavam a circular no barramento, dissipando qualquer dúvida que existisse, quanto à leitura das mensagens por parte da *Fez-Cobra*.

Estabelecido o código base nas comunicações CAN entre a *Fez-Cobra* e o painel, foi o ponto de partida para a adaptação deste código à aplicação em desenvolvimento. A partir deste momento, apenas seria necessário garantir que a aplicação incorporasse o protocolo das mensagens CAN do painel, para o correcto funcionamento nas trocas de dados.

PROTOCOLO DA COMUNICAÇÃO

O protocolo deste código é relativamente simples bastando enviar duas mensagens e aguardar por uma mensagem de resposta. A primeira mensagem a enviar tem a intenção de “acordar” o painel, uma vez que, estando o painel

sem actividade este entra em *sleep mode*[9]. Posteriormente é enviada uma mensagem, cujo conteúdo é reconhecido pelo painel, como sendo um pedido do estado das teclas. Como resposta a este pedido, é enviada uma mensagem composta pelos *Data Bytes* que contêm esta informação. Consoante o estado dos bits que compõem os *Data Bytes*, poder-se-á aferir sobre o estado das teclas. A verificação do estado da tecla num painel vai ser resumida ao seu contacto eléctrico.

Control Panel	Button	Data Byte	Bit
	A TASTE_AUTO	0	0
	B TASTE_LUFTVERTEILUNG	0	1
	C TASTE_GBL_MINUS	0	2
	D TASTE_GBL_PLUS	0	3
	E TASTE_MAX_AC	0	4
	F TASTE_SH_LI	1	0
	G TASTE_SH_RE	1	1
	H TASTE_SL_LI	1	2
	I TASTE_SL_RE	1	3

Figura 107 – Correspondência entre teclas e bit/byte do CAN.

Num estado de repouso, todas as teclas possuem os seus *Data Bits* no nível lógico 0. Neste caso o contacto eléctrico encontra-se aberto. Se a tecla está pressionada e fechou o contacto eléctrico, o bit correspondente a essa tecla passa ao estado alto. Em função do número de teclas a resposta pode incorporar mais que um *Data Byte*, ou eventualmente ser complexa, ao nível de utilizar mais que uma trama para a obtenção da resposta. Na Figura 107 encontra-se apresentado o painel utilizado na execução do teste deste projecto e a correspondência entre as teclas e os *Data Byte*, acompanhado dos respectivos bits.

Aquando da recepção são despoletados dois tipos de eventos: um para a recepção de mensagens e outro no caso de acontecer erro. No caso dos erros, estes estão qualificados em três tipos, sendo as suas designações:

- *"OverRun"* – Sobreposição de mensagens e consequente perda das mesmas;

[9] Estado em que determinado dispositivo se encontra, onde as suas funcionalidades gerais são desligadas, permitindo economizar energia. Proporciona um retorno célere à actividade normal ("*acordar*"), após o despoletar de um sinal específico.

- *"RxOver"* – *Buffer* interno completo, com perda de mensagens posteriores;
- *"BusOff"* – Erro no barramento ou no controlador CAN, provocando a inoperacionalidade.

"ACORDAR" PAINEL

Durante o processo inicial de "acordar" o painel, do seu estado de *sleep mode* é necessário enviar uma mensagem. Ao enviar esta mensagem constatou-se que em determinadas alturas esta originava erro, colocando as comunicações CAN fora de serviço. Foi possível de verificar que a ocorrência deste erro acontecia sempre que o painel não se encontrava ligado à *Fez-Cobra*. A explicação da causa do aparecimento do erro era difícil de descortinar, pois não existia razão aparente.

Após alguma análise das circunstâncias em que ocorria o erro, chegou-se à conclusão que a causa estava no não reconhecimento da mensagem CAN no barramento, por uma qualquer estação. Ora, não existindo mais nenhum elemento ligado no barramento CAN, a mensagem enviada pela *Fez-Cobra* para "acordar" o painel, não vai gerar o *acknowledge* (ack). A não ser captada por uma estação, a mensagem vai dar origem a erro.

Com o evento de detecção de erros a sinalizar a ocorrência de erro do tipo *"BusOff"*, este colocava fora de serviço o controlador CAN da *Fez-Cobra*. Sendo este erro originado no início da comunicação, a solução passou por criar um mecanismo que ignore este erro inicial (apenas este erro). Passado este erro inicial por inoperância do barramento, todas as restantes mensagens relativas ao "acordar", são bem interpretadas.

Na prática, ou seja, no automóvel isto não sucede, em virtude de existirem vários dispositivos CAN ligados ao barramento e, deste modo, qualquer um deles pode fazer o *acknowledge* (ack) da mensagem. Com o desenrolar do teste da tecla é necessário ir mantendo activo o painel, isto é, não permitir que ele entre em *sleep mode*. Para este efeito é enviada, com periodicidade de um

segundo uma mensagem para manter activo o painel (mensagem “acordar”). Usualmente esta técnica é designada por *refresh*.

5.1.6. DESENVOLVIMENTO DA CONSOLA

Em complemento com a aplicação desenvolvida para a *Fez-Cobra*, tornou-se imprescindível criar uma outra aplicação, desta feita para o computador. Esta aplicação, designada por consola, é uma interface que permite disponibilizar um conjunto de opções que, vão condicionar o modo de funcionamento do teste a executar.

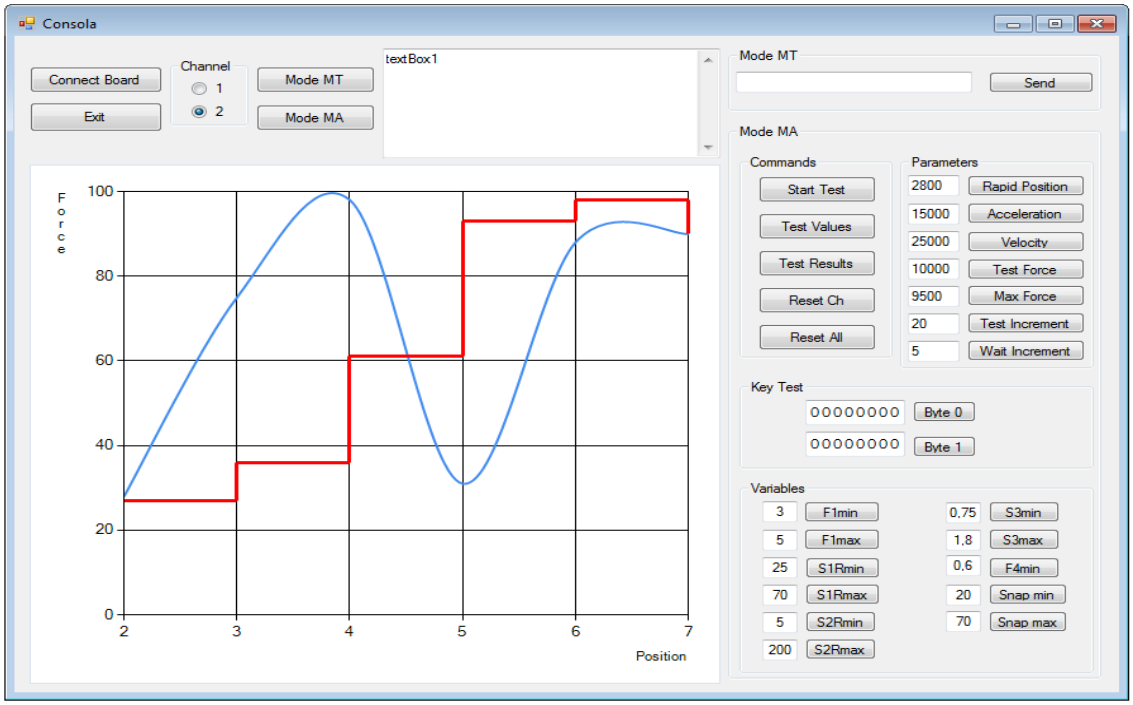


Figura 108 – Consola de trabalho no computador.

Fundamentalmente estas opções caracterizam-se pelas configurações que os testes podem ter, podendo ser divididas em dois grandes grupos. Em função do modo de funcionamento, a consola disponibiliza os comandos e/ou opções desse mesmo modo. Recordemos que os modos de funcionamento são: Transparente e Autónomo. A Figura 108 apresenta o *template* gráfico utilizado na consola de trabalho, o qual é constituído pelos seguintes elementos:

- “Connect Board” – Ligação do computador à *Fez-Cobra*, utilizando a ligação Ethernet;

- "Exit" – Permite encerrar a interface e todas as suas ligações;
- "Channel" – Identifica o canal (*SMAC*) com o qual se vai trabalhar. Em termos logísticos não foi possível obter mais *SMAC*'s, logo só existem dois canais;
- "Mode MT" – O Modo Transparente disponibiliza a inserção de comandos (um da cada vez), os quais são executados após o seu envio. Quando seleccionado este modo ficam disponíveis os comandos sob a alçada do grupo "Mode MT", assim como o comando "Mode MA", o qual permite a mudança de modo. A Figura 109 apresenta um exemplo da consola neste modo de operação;

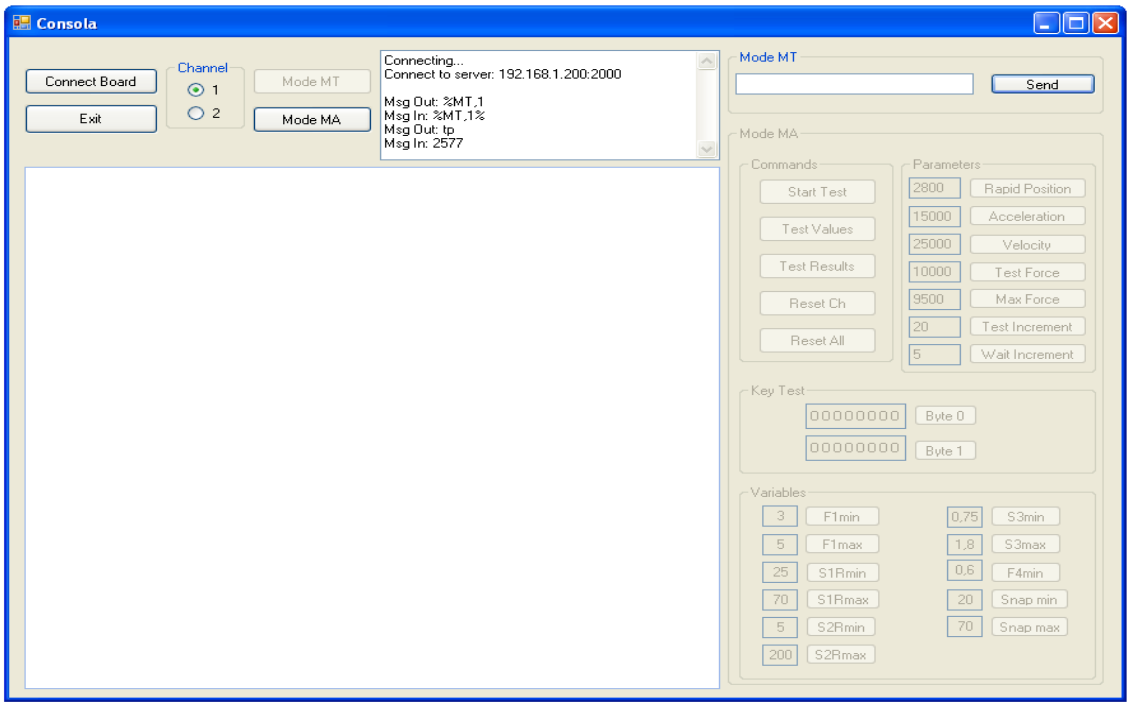


Figura 109 – Exemplo da consola em modo transparente ("Mode MT").

- "Mode MA" – O Modo Autónomo permite executar um ciclo de teste à tecla com as parametrizações pretendidas, as quais se encontram no grupo "Mode MA". A Figura 110 representa a consola em modo autónomo e após a execução de um teste, com obtenção dos seus valores e apresentação gráfica;

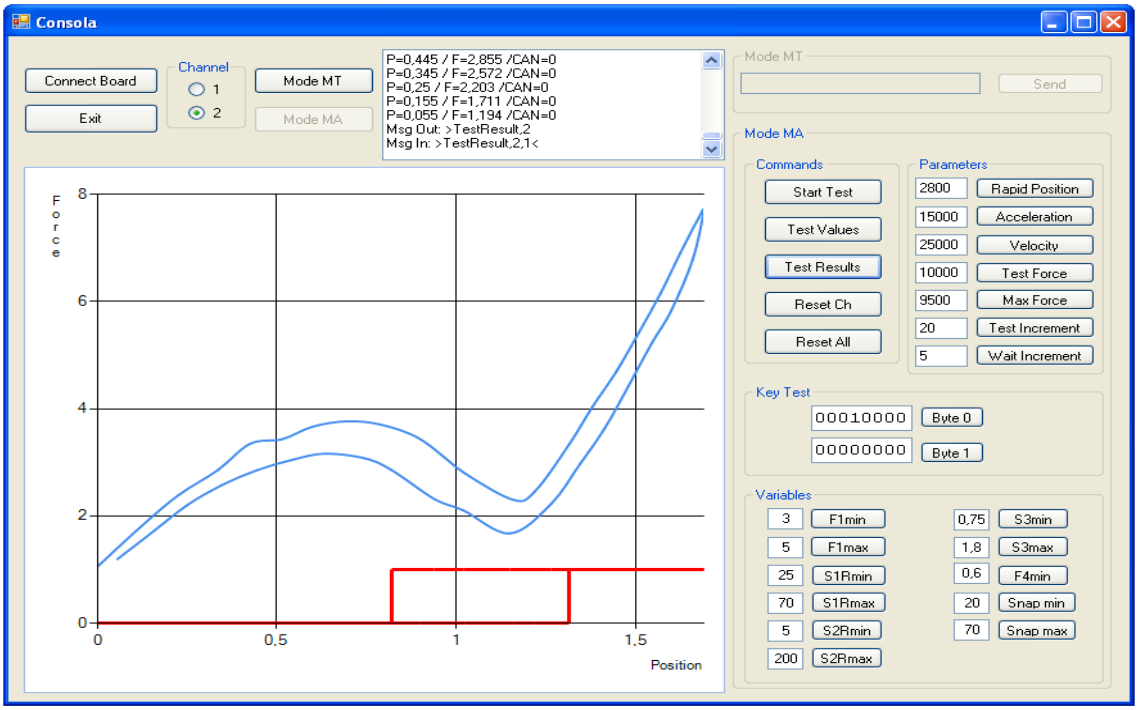


Figura 110 – Exemplo da consola em modo autónomo (“Mode MA”).

- “textbox1” – Representa a caixa de texto onde são apresentadas as mensagens, como por exemplo os comandos inseridos, respostas a comandos, alterações das configurações, etc.;
- Grupo “Mode MT” – É composto por uma linha de texto, para inserção de comandos do *SMAC* e, pelo botão “Send”, que envia o comando para o *SMAC*. Caso o comando seja passível de obtenção de resposta, esta é visualizada no campo “textbox1”;
- Grupo “Mode MA” – Conjunto de elementos que servem de configuração à execução do teste em modo autónomo. Este grupo é composto por quatro subgrupos: “Commands”, “Parameters”, Key Test” e “Variables”;
- Grupo “Commands” – Possui os comandos a executar, sendo:
  - “Start Test” – Envia a ordem para iniciar o teste da tecla;
  - “Test Values” – Vai recolher os valores obtidos pela execução do teste. Apresenta os valores na caixa de texto e, simultaneamente, elabora o gráfico do teste efectuado;
  - “Test Results” – Apresenta o resultado na caixa de texto do teste, sob a forma de validado ou não (ver ponto 5.1.5.4);

  - "Reset Ch" – Apaga os valores do teste referentes ao canal seleccionado;
  - "Reset All" – Semelhante ao anterior mas neste caso apaga os valores de todos os canais.

- Grupo "Parameters" – É composto pelos parâmetros do teste, já analisados anteriormente. Possui uma caixa de texto para inserir o valor do parâmetro e o botão que, quando pressionado, envia o parâmetro para a *Fez-Cobra*;
- Grupo "Key Test" – Indica a tecla que vai estar sujeita ao teste, através da indicação do byte e do respectivo bit, inserido na caixa de texto. O botão lateral permite enviar a configuração do byte para a *Fez-Cobra*. Este valor fica sujeito à análise da leitura da mensagem CAN recebida do painel;
- Grupo "Variables" – São um conjunto de variáveis que permitem ajustar a análise do teste. Como nem todas as teclas têm a mesma forma de resposta, foi necessário colocar este parâmetros, de forma a tornar versátil o campo de aplicação do teste. Os parâmetros base e relações descritos no ponto 2.4.3.3 encontram-se susceptíveis a adaptações através da inserção de novos valores neste grupo. Uma alteração de valor é inserida na caixa de texto e posteriormente enviada para a *Fez-Cobra*, pressionando o respectivo botão;
- Gráfico – Local onde é apresentado graficamente os valores obtidos da leitura do teste. A curva do gráfico apresentada a azul corresponde à força *versus* deslocamento, enquanto a vermelho é apresentada a curva (em formato degrau) correspondente ao contacto eléctrico da tecla. Este elemento apenas é utilizado no modo autónomo e quando é feita a importação dos resultados da *Fez-Cobra*, através do comando "Test Results".

5.2. ANÁLISE DOS RESULTADOS

No cômputo geral a análise dos resultados deste projecto recai sobre duas áreas temáticas: uma de cariz técnica e uma outra de cariz operacional.

Inevitavelmente, era preponderante que a vertente técnica tivesse sucesso, pois sem o cumprimento das funcionalidades técnicas do projecto a sua operacionalidade não fazia sentido.

5.2.1. RESULTADOS TECNOLÓGICOS

Na vertente técnica os resultados obtidos têm de ser analisados em função do modo de operação do novo sistema. Assim, no modo transparente, o qual é o modo de funcionamento auxiliar, duas funcionalidades eram essenciais para o correcto desempenho deste modo: garantir o estabelecimento das comunicações entre os diferentes elementos (conversão Ethernet/RS-232 e vice versa) e em seguida, o comando inserido na consola tinha de ser executado por parte do *SMAC* (implicava uma correcta recepção/envio de dados). Garantidas estas duas funcionalidades, a dependência do sistema de teste face ao computador passou a ser diminuta. A ligação directa, via porta série, entre o computador e o *SMAC* deixou de ser necessária para o envio de comandos individuais.

As funcionalidades adstritas ao modo transparente tiveram sucesso pois, as comunicações implementadas neste projecto que, envolviam diferentes elementos, provaram estar à altura das necessidades. Não foram reportados erros, nem evidenciados atrasos nas propagações das comunicações e, tal era plausível que pudesse acontecer em virtude da conversão Ethernet/RS-232, no processo de envio e o inverso no processo de recepção. Não sendo este um factor preponderante não seria agradável esperar 2 ou 3 segundos pelo envio de comandos e mais 2 ou 3 segundo na recepção de dados, caso houvesse resposta.

Neste modo transparente, em que o comando a ser executado é inserido na consola, foi possível verificar a sua funcionalidade, por intermédio de duas situações: uma ocorrida pela movimentação do *SMAC* e a outra pela resposta obtida na consola.

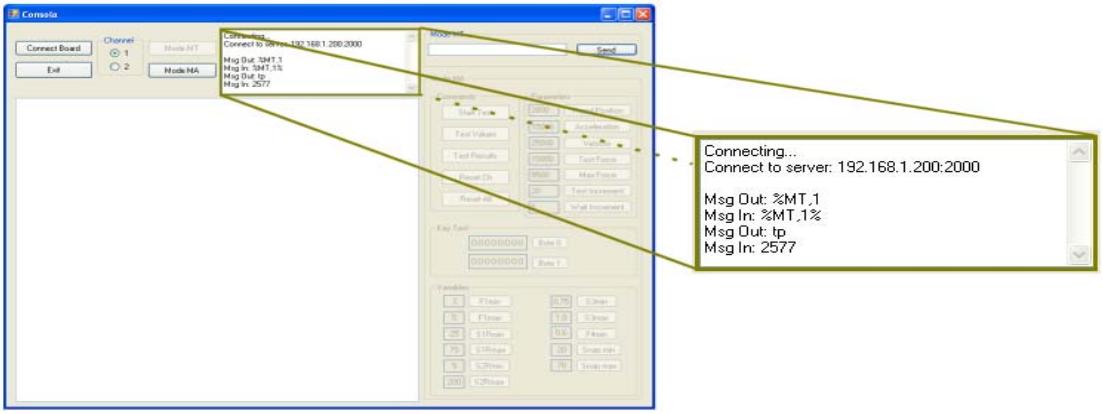


Figura 111 – Exemplo de funcionamento em modo transparente.

Na Figura 111 encontra-se exemplificada a consola de trabalho quando esta é seleccionada para o funcionamento em modo transparente. A interacção com o sistema e os resultados esperados são apresentados na caixa de texto, conforme se pode observar na Figura 111, no lado direito em destaque. Neste exemplo, após o estabelecimento da ligação Ethernet (primeiras duas linhas) e o envio com resposta do modo de operação – MT (duas linhas seguintes), surge o comando pretendido. O comando enviado para o *SMAC* foi o "tp" (*tell position*), cuja resposta foi "2577". Ou seja, o comando vai solicitar o posicionamento actual do *SMAC*, permitindo desta forma o reconhecimento de uma coordenada.

Verdadeiramente, o modo autónomo é aquele que enquadra na essência deste projecto, ou seja, a análise do estado das teclas. Em virtude do computador estar incumbido da gestão global do teste, no actual processo do teste, era necessário transferir para o domínio da plataforma *Fez-Cobra*, toda essa gestão. Assim, com a implementação de um novo sistema de *hardware*, o desenvolvimento de uma nova aplicação para a *Fez-Cobra* e outra para a consola foram os passos dados na introdução das alterações necessárias. Em complemento existia a necessidade de adequar estas alterações aos resultados obtidos no decorrer do teste.

A passagem da gestão do teste do computador para a *Fez-Cobra* resulta numa maior disponibilidade de recursos por parte do computador. Ou seja, os recursos absorvidos ao computador (ao nível da comunicação e

processamento) no decorrer do teste à tecla passaram a estar disponíveis. Tendo em conta que o tempo médio na execução do teste é de 6,8 segundos[10], foi este o tempo suprimido na actividade do computador. Este tempo é susceptível de ser reduzido na linha de produção, em virtude de uma optimização do teste e dos seus parâmetros.

A rapidez de execução do teste é dependente do *SMAC*, em virtude de este ser o elemento mais lento do sistema. Contudo em determinadas situações de elevada actividade nos recursos do computador, não estando quantificável, verifica-se um aumento no tempo de execução do teste. Minimizando o uso do computador, a rapidez de execução é da responsabilidade do *SMAC*.

A consola apresentada na Figura 110, para além de ser novidade, disponibiliza um conjunto de funcionalidades até agora inexistentes. Como destaque destas funcionalidades surgem os parâmetros (“Parameters”) os quais são passíveis de serem configurados antes do teste iniciar, permitindo versatilidade na sua aplicação. A actual aplicação define estes valores no código e, como não possui consola, a alteração dos valores é efectuada com uma intervenção no próprio código da aplicação. De igual forma, encontram-se nesta situação os valores do “Key Test” e das “Variables”.

Uma outra característica nova, a qual corresponde a um dos requisitos, prende-se com a selecção do canal (“Channel”). Com este novo sistema são disponibilizados quatro canais que correspondem à utilização de quatro *SMAC*’s. A selecção de um canal vai desencadear todo o processo inerente ao teste, em função desse mesmo canal.

Atestar da validade dos resultados era um factor importante pois, a resposta obtida na execução de um teste a uma tecla, não seria garantia do bom funcionamento do sistema. Uma primeira análise e, face ao conhecimento da *Preh* no comportamento das teclas em função do teste, foi possível validar os

[10] Valor obtido no decurso do desenvolvimento da aplicação *Fez-Cobra*, através de um temporizador interno. Convém referir que este valor apenas serve de referência e tendo por base os ensaios realizados.

resultados obtidos pela visualização gráfica. Uma segunda análise foi obtida através da inconformidade da tecla, ou seja, o teste tem de corresponder às especificações referenciadas no ponto 2.4.3.3. Uma falha destes critérios é fácil de simular, bastando para tal alterar os valores do grupo “Variables” existente na consola. A simulação, obrigando à ocorrência de erros, aconteceu com estreitamento das tolerâncias, mais concretamente, com a modificação dos valores do grupo “Variables”.

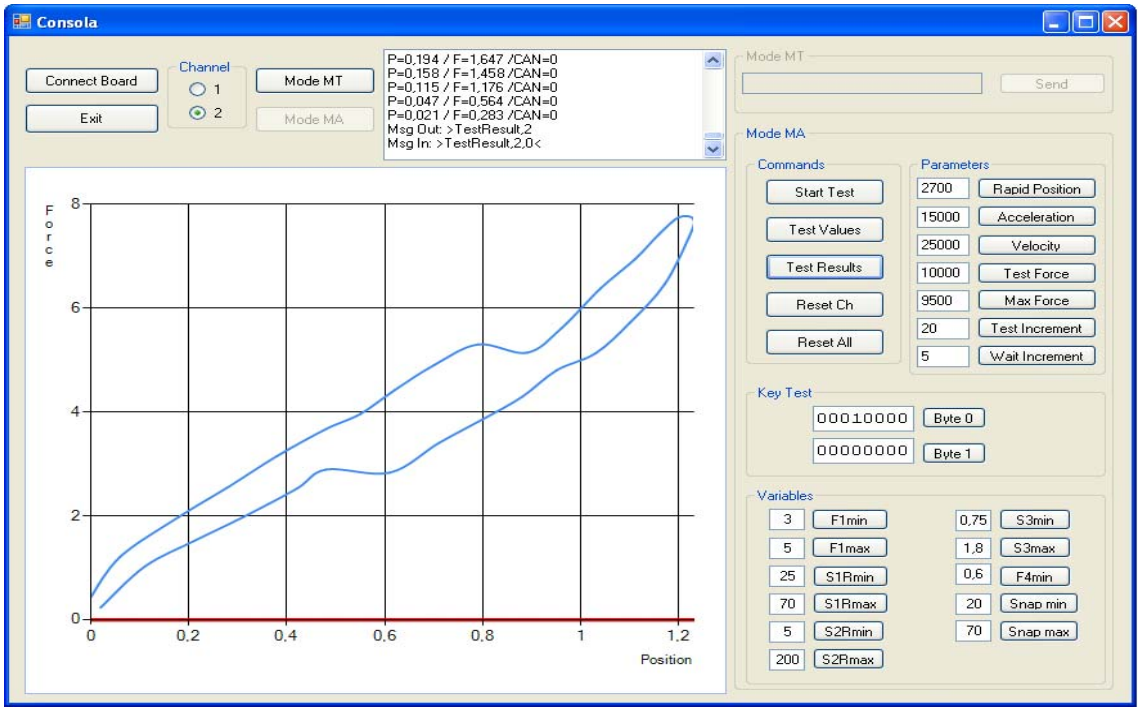


Figura 112 – Falha devido ao comportamento da tecla.

Outras falhas foram analisadas para confirmar que as mesmas eram detectadas no sistema tais como: alteração no comportamento da tecla, membrana danificada ou mal posicionada, encravamento (no início ou final do curso). Uma destas falhas pode ser observada na Figura 112, reportando esta situação à ocorrência de erro com origem num comportamento não condizente com as especificações. Pela análise gráfica é notória a diferença de comportamento da curva obtida no teste, face à curva padrão do teste em causa. Para além deste facto, também ocorre uma falha no contacto eléctrico, em virtude deste painel não possuir comunicação CAN. Na Figura 113 é apresentada outro tipo de falha, ocorrida ao ser executado um teste numa tecla. Neste exemplo a tecla encontra-se encravada e logo na fase inicial do

curso da mesma. O deslocamento efectuado pelo *SMAC* é curto pois rapidamente a força máxima é atingida, fazendo este o retorno às condições iniciais. Como a tecla não atingiu a PCB, então o contacto eléctrico não se realizou.

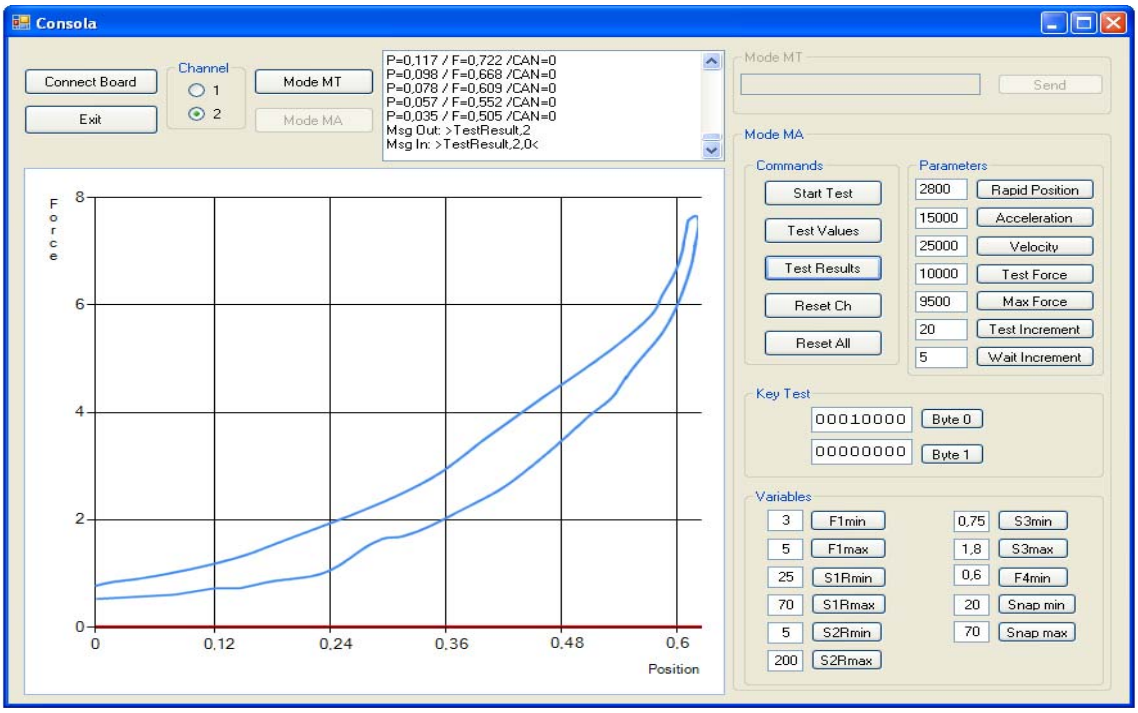


Figura 113 – Falha devido ao encravamento da tecla.

Uma outra área em apreciação prende-se com a utilização simultânea dos *SMAC*'s, ou seja, a execução simultânea de teste de teclas. Devido a questões logísticas não foi possível implementar quatro *SMAC*'s nos ensaios decorridos, sendo limitando o teste prático a dois *SMAC*'s. No entanto, a estrutura da aplicação empregue na *Fez-Cobra* foi desenvolvida tendo em conta a utilização de quatro *SMAC*'s, não se encontrando deste modo limitada. Apenas a consola está de momento limitada à utilização de dois canais (dois *SMAC*'s) mas facilmente é alterada para ser utilizada em quatro canais.

Convém referir que ao nível individual os *SMAC's* demonstraram sempre excelente comportamento nas tarefas que lhes foram atribuídas, apesar das suas limitações temporais impostas pela comunicação série, nunca comprometeram o sistema de teste. No funcionamento simultâneo de dois *SMAC*'s foi registado um ligeiro acréscimo no tempo de execução do teste.

Este facto já era expectável e, deve-se à partilha de recursos que acontece na *Fez-Cobra*, mais concretamente devido ao uso de duas portas série simultâneas, à análise do barramento CAN e duas estruturas de análise de resultados. O atraso temporal foi quantificado na ordem dos 0,45s, isto é, o tempo do teste individual é acrescido deste valor.

Uma área, na qual existiu algum enfoque no desenvolvimento da aplicação, foi a das comunicações. Era objectivo que estas, em caso de falha, não comprometessem o funcionamento do sistema provocando nesta situação o bloqueio do sistema. Através da simulação de falhas nas comunicações, provocadas pela a interrupção física do suporte de comunicação (desligando os cabos de comunicação), foi analisado o comportamento do sistema. Desta forma não foram detectadas situações anómalas que originassem interferência no normal comportamento do sistema. No entanto, existe uma situação onde se torna necessário ter alguma atenção e intervenção, mais concretamente, quando existe uma falha de comunicação (série) entre o *SMAC* e a *Fez-Cobra* no decorrer do teste. Neste caso em particular, o *SMAC* fica a aguardar por uma resposta da *Fez-Cobra* a qual nunca irá surgir, provocando uma paragem na execução do código no *SMAC*. Esta situação pode ser contornável com alteração do código existente no *SMAC* ou, pelo envio de um comando pela consola (utilizando o modo transparente) obrigando-o a regressar à posição inicial (*home position*).

5.2.2. RESULTADOS OPERACIONAIS

Numa perspectiva operacional, os resultados deste projecto são perfeitamente enquadráveis com a realidade, vislumbrando-se uma forte implementação no sector produtivo da *Preh*. Para isto contribuem a análise positiva dos resultados tecnológicos (abordados anteriormente) aliados a factores económicos e materiais favoráveis.

O compromisso económico era um factor decisório na implementação deste projecto e, como nenhuma alteração substancial foi englobada ao projecto inicial então, as condicionantes económicas foram cumpridas. Comparando o

preço médio de um computador, pelo preço médio de um sistema embebido é perceptível a diferença de custos. Para além deste custo directo existe o benefício indirecto, onde o sistema embebido pode absorver mais e diferentes tipos de periféricos. Este benefício advém da implementação, até um máximo, de quatro sistemas de deslocação (*SMAC*'s) por sistema embebido (*Fez-Cobra*).

Ao nível material, apesar de não ter sido possível efectuar uma implementação no verdadeiro meio de actuação, a linha de produção, é evidente a operacionalidade deste projecto, quanto mais não seja, pelo facto de toda a sua estrutura "externa" do teste se ter mantido inalterável. Isto é, um operador / técnico que exerça funções numa estação, não vai notar qualquer tipo de alterações físicas com este novo sistema, uma vez que, a área de execução de teste permanecer inalterada. Internamente, no armário que serve de suporte aos componentes eléctricos, apenas é necessário dispor de um pequeno espaço para incluir a *Fez-Cobra*. Desta forma nenhum custo vai ser acrescentado ao projecto por necessidades de mudança na estrutura física.

A implementação deste projecto tem em conta duas vertentes na área de produção, ou seja, a criação de novas linhas de produção e a manutenção das actuais linhas de produção. Ao ser criada uma nova linha de produção, para um novo produto, vai incluir esta nova versão tecnológica da execução do teste. Actualmente encontra-se em desenvolvimento na *Preh*, em parceria com um cliente, a produção de um novo painel que permitirá a inclusão deste sistema de teste, com base na *Fez-Cobra*. Na vertente de produção actual, a introdução desta tecnologia vai ser efectuada de forma gradual e ao longo dos tempos, sobretudo tendo em conta as necessidades computacionais.

Tendo em conta a melhoria contínua na sua produção, a *Preh*, passa a dispor de uma solução tecnológica mais interessante ao nível produtivo e económico, para ser implementada ao longo das suas linhas de produção actuais ou futuras linhas de produção, mantendo a operacionalidade actual.

6. CONSIDERACÕES FINAIS

6.1. CONCLUSÕES

O projecto desenvolvido na *Preh* e descrito nesta dissertação, parece demonstrar que é possível substituir a tecnologia existente, na execução do teste das teclas, por uma mais recente. Nesta mudança é apanágio “fazer mais com menos”, isto significa que novas funcionalidades e novas potencialidades foram introduzidas, reduzindo o custo comparativamente ao sistema actual.

O elemento chave neste projecto é a placa *Fez-Cobra* e, como tal, era imperativo que as características apresentadas pelo fabricante pudessem colmatar as necessidades reais. Assim, face às expectativas criadas em redor da *Fez-Cobra*, estas não foram goradas pois nunca comprometeu o funcionamento do novo sistema. Os tempos de execução dos testes corresponderam às necessidades, pois este seria o factor crítico para a *Fez-Cobra*, em virtude da multiplicidade e simultaneidade no processamento dos resultados dos testes. Este factor foi relevante no sucesso do projecto, pois permite dispor de um sistema embebido a controlar quatro *SMAC*’s, sem qualquer processamento adicional. O algoritmo correspondente ao processamento dos resultados obtidos, em função da curva característica, foi enquadrado na *Fez-Cobra*, libertando o computador deste tipo de processamento. Ao dotar o sistema de teste de uma ligação Ethernet, este torna-se passível de ser intervido externamente, independentemente da sua localização, desde que exista uma ligação à rede Ethernet.

Inicialmente, a utilização das comunicações série RS-232 e CAN na placa Fez-Cobra é condicionada pela falta de hardware, pois esta não vem provida com suporte físico para interligação ao exterior. Como tal, teve de ser implementado hardware suplementar para que estas duas tipologias de comunicação de dados estivessem acessíveis externamente. O recurso ao protocolo CAN obrigou a uma actualização do *firmware* na *Fez-Cobra* que adicionou novas e melhoradas funcionalidades. A linguagem de programação C# vem reduzida

nas suas funções e métodos, obrigando a criar processos alternativos para a execução de funções mais elaboradas.

A implementação da comunicação CAN entre o painel de controlo e a *Fez-Cobra* recorreu a um adaptador devido às diferenças na velocidade de transmissão. Também foi necessário recorrer a *software* e *hardware* que permitisse uma análise das mensagens no barramento.

Em complemento, o desenvolvimento da consola de trabalho permitiu tornar todo este sistema mais versátil. Uma vertente de configuração e uma outra vertente de análise encontram-se disponíveis para o técnico/operador poder usufruir das características da aplicação implementada na *Fez-Cobra*. Foram criados dois modos de trabalho, garantindo desta forma uma abrangência alargada no controlo do teste e nos próprios *SMAC*'s, até agora indisponível.

No cômputo geral este projecto cumpriu os requisitos inicialmente estipulados, sendo certo que tem uma grande margem de progressão. Ficou também evidenciado que um sistema embebido pode suprimir, em determinadas situações, sistemas computorizados dedicados, sem colocar em causa as suas funcionalidades. Não é um sistema estanque, deixando em aberto um leque alargado de potenciais aplicações na linha produção, com todas as vantagens que daí advêm (custo reduzido, facilidade de aplicação – sistema modular, dimensões reduzidas, etc.).

6.2. DESENVOLVIMENTOS FUTUROS

Como já referido anteriormente, este tipo de projecto tem uma margem de progressão bastante promissora pois, capacidades ao nível do *hardware* e do *software* na *Fez-Cobra* estão longe de estarem esgotadas. Este projecto teve em consideração um tipo de curva características mas a análise desta curva, pode facilmente ser alargada a outro tipo de tecla. O *software* desenvolvido tem o intuito de poder ser adaptável a novas formas, novos parâmetros. Logo uma reformulação do algoritmo leva a que, novos painéis possam ser incluídos na linha de produção, onde se encontra implementado o sistema em projecto.

Com este tipo de produto passa a ser possível a implementação deste sistema ao longo da linha de produção, mais concretamente nas zonas de acção e /ou transformação. Isto é, sempre que existe uma acção sobre uma peça ou componente na linha de produção, nem sempre é acompanhado por teste. Nestas situações o teste surge mais tarde (duas, três estações à frente), sendo nesta altura detectada a falha da peça, ou seja, o tempo que vai desde a falha até à sua detecção está a ser desperdiçado. Um caso que ocorre actualmente na fase de colocação das teclas é, estas não são testadas de imediato mas apenas na próxima estação, quando o painel já se encontra montado. Este é um exemplo prático em como a implementação de dispositivos baseados nesta tecnologia podem surgir em postos/fases intermédias. Com esta solução e, tendo este caso como exemplo, o tempo que decorre entre a inserção e a detecção deixa de ser um desperdício temporal.

Ao longo das linhas de produção da *Preh*, devido à necessidade de processamento, existem muitas situações onde são usados computadores, em vez do equipamento mais propício, o PLC. Com um sistema baseado na *Fez-Cobra*, torna-se possível dotar todo o posto, estação com capacidade de processamento, apenas necessitando de receber ordens e enviar respostas ao PLC associado. Se num PLC forem agregadas várias estações, como por exemplo o projecto aqui apresentado, todos os computadores inerentes a estas estações podem ser suprimidos.

A aplicação da *Fez-Cobra* já se encontra preparada para, num futuro, ser implementado um modo de funcionamento designado por diagnóstico. Com este modo pretende-se desenvolver um teste padrão e recolher resultados sobre esse teste, ou seja, uma situação de auto aprendizagem. Outras opções em aberto podem fazer recolha de elementos estatísticos (tempos de teste, ocorrência e descrição de erros, etc), trocar mensagens CAN, etc., ou seja, poder implementar novas necessidades adjacentes ao teste.

De modo a alargar o campo de aplicação do teste de teclas nos painéis de climatização, tem de ser adicionada à aplicação agora desenvolvida a opção *multiframe*. Esta opção deriva do facto de determinados painéis, usualmente

aqueles que possuem um elevado número de teclas, terem a necessidade de sinalizarem a tecla (contacto eléctrico) em duas tramas de dados CAN.

Uma outra área de interesse futuro é a integração do conversor A/D na *Fez-Cobra*, uma vez que, esta placa já incorpora este tipo de conversor. Esta implementação vai implicar adicionar *hardware*, nomeadamente a etapa de amplificação do sinal. Com esta integração, as condições reais do teste não são substancialmente alteradas, a não ser a eliminação do bloco conversor A/D que deixa de ser um bloco autónomo. Isto torna o sistema mais compacto.

De cariz mais complexo e, possivelmente com necessidade de parceria com a *SMAC Inc*., um desenvolvimento futuro seria a integração do controlador do motor *SMAC* na *Fez-Cobra.* A parceria a realizar advém do facto do código fonte do controlador estar protegido pela *SMAC Inc.*, e como tal, esta implementação é dependente de terceiros. No entanto, este desenvolvimento acarretava melhorias significativas, pois a comunicação série passava a ser dispensável e o controlador tornava-se interno. Diminuição nos tempos de realização dos testes seria algo concretizável nesta situação. Para além disto, o *hardware* seria reduzido, diminuindo as dimensões do sistema e, consequentemente, os custos diminuiriam sem a aquisição do controlador (aqui voltamos novamente à parceria com a *SMAC Inc.*).

Algumas aplicações futuras não necessitam de exigências/características como a da *Fez-Cobra*. Sendo esta uma placa de topo, dentro desta gama da *GHI*, é possível recorrer a outras inferiores e consequentemente com um custo inferior.

7. REFERÊNCIAS

[Afia11] Indústria de componentes para automóveis, AFIA – Associação de Fabricantes para a Indústria Automóvel; Fevereiro 2011.

[ASTM01] ASTM F1570-01, Standard Test Method for Determining Tactile Ratio of a Membrane Switch; American Society for Testing and Materials; 2001.

[ASTM10] ASTM F2592-10, Standard Test Method for Measuring the Force-Displacement of a Membrane Switch; American Society for Testing and Materials; 2010.

[Atmel04] Can Tutorial; Atmel Microcontrollers; 2004.

[Auto07] Automobile Components & Auto Electronics Industry: Analysis & Investment Opportunities, Ministry of Economic Affairs; December 2007.

[Baid10] Shweta Dhadiwal Baid; Electronics for You; December 2010.

[Beasley09] Jeffrey S. Beasley, Networking 2nd Edition; Prentice Hall; 2009.

[Bentley05] John P. Bentley; Principles of Measurement Systems, 4th Edition; Prentice Hall; 2005.

[Bosch11] CAN – Controller Area Network; Robert Bosch GmbH; 2011. http://www.semiconductors.bosch.de; 2011.

[Bosch91] CAN Specification; Version 2; Robert Bosch GmbH; September 1991.

[Cami11]The RS232 Standard; CAMI Research Inc.; 2011. http://www.camiresearch.com

[CiA11] Controller Area Network; CiA; 2011. http://www.can-cia.org

[Dallas01] Fundamentals of RS-232 Serial Communications; Application Note 83; Dallas Semicondutor; 2001.

[Dominique07] Dominique Paret; Multiplexed Networs for Embedded Systems; WILEY; 2007.

[Freescale02] Using the SPI to Communicate Between Multiple Microcomputers; Application Note 991/D, Rev. 1; Freescale Semiconductor, Inc; 2002.

[Freescale09] Development with Microsoft® .NET Micro Framework 2.0; Application Note 3887, Rev.0; Freescale; 2009.

[GDSI11] Membrane Switches; GDSI; 2011.
http://www.gdsiswitches.com

[Gedney07] Richard Gedney; Automotive Materials – Measurement Erros in Mechanical Testing; Advanced Materials & Process; April 2007.

[GHI10a] Fez Cobra Board Brochure, GHI Electronics, 2010.

[GHI10b] Beginners guide to C# and .Net Micro Framework, Rev.1.06; GHI Electronics; 2010.

[GHI10c] Fez Cobra Board Schematic 1.2; GHI Electronics; 2010.

[GHI11] EMX User Manual, Rev. 1.3; GHI Electronics; 2011

[Göhner06] P. Göhner; Theory: Can-Bus; Universität Stuttgart; Institute of Industrial Automation and Software Engineering; April 2006.

[Hazen03] Mark E. Hazen; Understanding Some Basic Recommended Standards for Serial Data Communications; Intersil Corporation; 2003.

[Herald11] Controller Area Network interface in embedded systems; E.E. Herald; 2011.
http://www.eeherald.com

[Huq93] Syed Huq; Understanding Power Requirements in RS-232 Applications; Application Note 914; National Semiconductor; 1993.

[Hurden05] Jonathan Hurden, SAE100 Future Look; May 2005.

[IETF81a] RFC 791, Internet Protocol; Internet Engineering Task Force; September 1981.

[IETF81b] RFC 793, Transmission Control Protocol; Internet Engineering Task Force; September 1981.

[Inteli05] Diagnóstico da Industria Automóvel em Portugal; Inteli; Abril 2005.

[Johanson11] Mathias Johanson; Information and Communication Support for Automotive Testing and Validation; 2011.

[Kaplan01] Hadriel Kaplan, Robert Noseworthy, The Ethernet Evolution From 10 Meg to 10 Gig; Networld + Interop; Atlanta; 2001

[Keith99] Keith Pazul; Controller Area Network Basics; Application Note 713; Microchip; 1999.

[Knitter11] Input Systems Overview; Knitter-Switch; 2011.

[Kozierok05] Charles M. Kozierok; The TCP/IP Guide Version 3.0; September 2005.

[Kuhner09] Jens Kuhner; Expert .NET Micro Framework, 2nd Edition; Apres; 2009.

[Lava02] RS-232: Serial Ports; Lava Computer MFG Inc; 2002.

[Maxim11] 1-Wire Tutorial; Maxim; 2011.
http://www.maxim-ic.com

[Maxton05] Graeme P.Maxton, John Wormald, Time for a Model Change, Cambridge University Press; 2004.

[McQuerry08] Steve McQuerry; Interconnecting Cisco Network Devices Part-1; 2nd Edition; Cisco Press; 2008.

[ME07] Data Sheet KD24s; ME-Meßsysteme GmbH; 2007.

[ME11] Data Sheet: Strain Gage Measuring Amplifier GSV-1; ME-Meßsysteme GmbH; 2011.

[Miles07] Donald Thompson, Rob S. Miles; Embedded Programming with the Microsoft .NET Micro Framework; Microsoft Press; 2007.

[Nagurka05] Mark Nagurka, Richard Marklin; Measurement of Stiffness and Damping Characteristics of Computer Keyboard Keys; Journal of Dynamic Systems, Measurement, and Control; June 2005.

[NI11] Load Cells ans Pressure Transducers – Overview of Operating Principles, Tutorial 7138; National Instruments; 2011

[NPX07] I2C-Bus specification and user manual, Rev.03;, NXP Semiconductors; 2007.

[NPX09] LPC24XX User Manual, Rev. 04; NXP Semiconductors; 2009

[NPX11] http://ics.nxp.com/products/lpc2000/lpc24xx/

[Oracle11] System Administration Guide Volume 3 – Overview of TCP/IP; Oracle; 2011
http://download.oracle.com

[PEAK06] PCAN-TJA1054, User Manual; PEAK-System Technik GmbH; 2006.

[PEAK11] USB to CAN Interface, User Manual V2.0.5; PEAK-System Technik GmbH; 2011.

[Ricelake07] Load Cell Handbook; Rice Lake; 2007.

[Robb04] Stuart Robb; Can Bit Timing Requirements; Application Note 1798; Freescale Semicondutor, Inc; 2004.

[Rodriguez01] Adolfo Rodriguez et al; TCP/IP Tutorial and Technical Overview; IBM; 2001.

[Shoma11] Strain Gages; SHOWA – Measuring Instruments Inc.; 2011
http://www.showa-sokki.co.jp

[Smac07] Smac Actuators User Manual, Revision 1.8; SMAC Inc; 2007.

[Smac10] Catalog; Smac Inc.; 2010.

[Smac11] Quality Control – Sample Application; SMAC Inc.; 2011.

[Smac97] LAC-1, Technical Reference Manual, Revision 1.0; SMAC Inc; 1997.

[Stallings06] William Stallings; Data and Computer Communications; 8th Edition; Prentice Hall; 2006.

[Steve08] Steve Corrigan; Introduction to the Controller Area Network (CAN); Application Report – SLOA101A; Texas Instruments; July 2008.

[Stevens94] W. Richard Stevens; TCP/IP Illustrated Volume 1: The Protocols; Addison Wesley; 1994.

[Taltech11] Introdution to Serial Communications; TAL Technologies Inc.; 2011.
http://www.taltech.com

[Tanembaum03] Andrew S. Tanembaum; Computer Networks; 4th Edition; Prentice Hall; 2003.

[Texas02] Interface Circuits for TIA/EIA-232-F; Design Notes; Texas Instruments; 2002.

[Thompson07] Donald Thompson, Colin Miller; Introducing the .NET Micro Framework, Microsoft Corporation; 2007.

[Volkswagen01] Data Exchange on the Can Bus I; Volkswagen AG, October 2001.

[Webster99] John G. Webster; The Measurement Instrumentation And Sensors Handbook; CRC Press LLC; 1999.

[Xilinx11] Ethernet; Xilinx Inc.; 2011.
http://www.xilinx.com

8. ANEXOS

Faz parte integrante desta dissertação o código desenvolvido para as aplicações da *Fez-Cobra* e da consola, o qual se encontra em formato digital no CD anexo ao documento. O CD é composto por duas pasta que incluem todo o software gerado pela ferramenta *Visual Studio Express Edition 2010*, o qual é necessário para trabalhar nesta mesma ferramenta. As pastas são as seguintes:

- *FezCobra* – Inclui todo o software desenvolvido e que foi aplicado na *Fez-Cobra*. Esta pasta é composta por diversas subpastas geradas pela abertura de um novo projecto no *Visual Studio* e pela necessidade de incluir as *assemblies* da placa *Fez-Cobra*.
- Consola – Abrange toda a estrutura criada no desenvolvimento da consola. Também inclui subpastas, mas agora, apenas devidas à abertura de um novo projecto.

Nas pastas anteriormente referidas, os ficheiros com extensão “sln” e “suo” são representativos do projecto, enquanto os ficheiros com extensão “cs” representam o código desenvolvido em C#.