



Universidade do Porto
Faculdade de Engenharia
FEUP

Aplicações Multimédia para Ambientes Industriais

Bertil Maria Pires Marques

Dissertação de Mestrado, Tecnologias Multimédia

Junho 2006

Orientação: Prof. Eduardo Tovar

Sumário

Desde há alguns anos, *gadgets* como assistentes pessoais digitais (*Personal Digital Assistants* – PDA), telemóveis e câmaras digitais têm vindo a ganhar uma crescente popularidade e adesão, mesmo no universo de utilizadores menos familiarizados com as novas tecnologias de informação e comunicação (TIC). Contudo, e por diversas razões, os ambientes fabris, e em particular os sistemas de fabrico automatizado, continuam quase herméticos a esta nova realidade.

Uma das principais justificações prende-se com a natureza pouco evolutiva do equipamento de automação industrial, que tipicamente tem um tempo de vida espectacularmente longo. Por outro lado, as TIC vocacionadas para os ambientes de automação industrial têm de cumprir requisitos apertados de confiança no funcionamento, de robustez, de funcionamento em tempo real e de segurança. Por isso, normalmente, baseiam-se em sistemas computacionais com especificações e funcionalidades diversas das TIC vocacionadas para outros ambientes de aplicação, nos quais os requisitos atrás referidos têm eventualmente níveis de preocupação mais baixos.

Acredita-se, porém, que as novas abordagens de interacção pessoa-máquina, incluindo tecnologias multimédia e as técnicas de realidade aumentada, combinadas com a crescente disponibilidade de tecnologias de computação ubíqua, permitirão modificar, para melhor, a forma como os operários fabris interagem com as máquinas e com os sistemas de informação da planta fabril.

Nas décadas de 80 e 90, o advento das redes do tipo *fieldbus* quebrou com a tradicional filosofia de controlo centralizado e comunicações ponto-a-ponto vigente até então nos sistemas de automação de fabrico. Nos anos mais recentes, as pressões do mercado e a crescente disponibilidade de técnicas e tecnologias relacionadas com as modernas TIC têm levado os fabricantes a fazer evoluir as soluções *fieldbus* existentes de forma a torná-las compatíveis com uma realidade mais ubíqua e interactiva do ambiente fabril. Para isso, duas funcionalidades cruciais (adicionais) são essenciais às redes do tipo *fieldbus*: suporte de mobilidade de dispositivos, e por isso a necessidade de integrar capacidades de comunicações sem fios (*wireless*) adequadas ao ambiente industrial; suporte de tráfego multimédia, e por isso a eventual necessidade de integrar protocolos e funcionalidades normalmente encontrados noutro tipo de tecnologias de comunicação, como é o caso do conjunto de protocolos associados à *Internet*.

A tecnologia RFieldbus disponibiliza estas extensões. Esta tecnologia será descrita nesta dissertação, bem como uma aplicação industrial de demonstração levada a cabo no ISEP. O objectivo desta tese é o de ilustrar e concretizar o desenvolvimento de aplicações de *software*, demonstrando, por essa via, a hipótese de que, com estas novas tecnologias de informação e comunicação industrial é viável uma perspectiva diferente da forma como os operários fabris podem interagir com as máquinas e com os sistemas de informação ao nível do controlo de operações de fabrico.

Abstract

Since a few years, gadgets such as Personal Digital Assistants (PDA), mobile phones or digital cameras have been gaining a crescent popularity even among users less familiar with the emerging information and communication technologies (ICT). However, and for several reasons, the factory-floor has remained almost hermetic to this revolution.

One of the main justifications deals with the less evolving nature of the industrial automation equipment, which has typically a relatively high life expectation. On the other hand, the ICT devoted to the automation of factory-floor operations must fulfil strict requirements concerning timeliness, reliability or dependability. Therefore, these rely on computing systems with specifications and functionalities that are different from those ICT devoted to other types of applications where the above mentioned requirements have a less stringent importance.

It is however believed that the new approaches for human-machine interaction, including multimedia technologies and augmented reality combined with the crescent availability of ubiquitous computing technologies will enable to modify and improve the way operators interact with the machines and the information systems at the factory-floor.

In the 80s and 90s, fieldbus networks revolutionised industrial communications at the factory-floor. In the more recent years, the fieldbus manufactures started to evolve commercial-off-the-shelf (COTS) solutions in order to make them suitable for the new reality of a more ubiquitous and interactively-rich factory-floor.

It was in this context that the RFieldbus (High Performance Wireless Fieldbus in Industrial-Related Multimedia Environment) European project evolved. The main target of that project was precisely the extension of the PROFIBUS international fieldbus standard towards supporting device mobility (via radio communications) and multimedia traffic (via IP encapsulation through PROFIBUS frames).

In this thesis we address the RFieldbus technologies and how they can be exploited to support innovative applications for the interaction between the operators and the machines and the information systems at the factory-floor.

Agradecimentos

Quero manifestar o meu agradecimento a todos os que no Grupo de I&D IPP-HURRAY do ISEP estiveram, de uma forma ou de outra, envolvidos nas actividades relacionadas com o projecto Europeu RFieldbus. Sendo que nesse contexto a lista seria extensa, gostaria de realçar o Filipe Pacheco, pela sua enorme contribuição aquando do desenvolvimento das primeiras versões das aplicações Servidor e Cliente *Intranet*. Essas foram as versões utilizadas na demonstração do Piloto do ISEP em 2003. Não sendo essas as aplicações principais para a demonstração da tecnologia RFieldbus, os seus protótipos eram por isso relativamente preliminares. Foi no entanto um excelente ponto de partida para as diversas iterações subsequentes, quer a nível de funcionalidades, quer a nível das interfaces dessas aplicações. Nas diversas versões entretanto elaboradas muito contribuíram as ajudas e o apoio do Filipe Pacheco e do meu orientador, o Eduardo Tovar, a quem gostaria também de transmitir o meu apreço pelo enorme empenho no esforço de escrita desta dissertação.

Gostaria por fim de agradecer a todos os colegas e/ou amigos que durante este longo período demonstraram incentivo, paciência, solidariedade, ensinamentos, disponibilidade e colaboração, contribuindo assim de alguma forma para elevar os meus níveis de motivação.

Porto, Junho 2006

Bertil Maria Pires Marques

“[...]Não tenhais medo das novas tecnologias! Elas incluem-se "entre as coisas maravilhosas" "*inter mirifica*" que Deus pôs à nossa disposição para as descobrirmos, usarmos, fazer conhecer a verdade, também a verdade acerca do nosso destino de filhos seus, e herdeiros do seu Reino eterno.[...] ”

Vaticano, 24 de Janeiro de 2005

Karol Wojtyła

Aos meus Pais;
Aos **verdadeiros AMIGOS**
Sempre presentes quando necessário....
...e a si AVÔ.



Índice

Capítulo 1 – Panorâmica Global da Dissertação.....	1
1.1. Motivação.....	1
1.2. Multimédia em Ambiente Industrial: Aplicações Prospectivas.....	2
1.3. As Redes de Comunicação Industriais	3
1.4. A Iniciativa RFieldbus.....	4
1.5. Objectivos da Dissertação e Hipótese.....	5
1.6. Abordagem	5
1.7. Organização da Dissertação.....	6
Capítulo 2 – Aspectos Relevantes do PROFIBUS/RFieldbus	7
2.1. Introdução.....	7
2.2. Características Fundamentais do PROFIBUS	8
2.2.1. Camada de Aplicação	8
2.2.2. Camada de Ligação de Dados.....	9
2.2.3. Camada Física.....	10
2.3. Extensões para o Suporte de Mobilidade no PROFIBUS.....	11
2.3.1. Adaptação de Meios Físicos por Inserção de Tempos de Inactividade.....	12
2.3.2. Suporte de Mobilidade entre Células	12
2.4. Extensões para o Suporte Multimédia no PROFIBUS	13
2.4.1. A Sub-camada IP/DP <i>Dispatcher</i>	14
2.4.2. A Sub-camada <i>Admission Control and Scheduling</i> (ACS)	16
2.4.3. A Sub-camada <i>IP-Mapper</i>	18
Capítulo 3 – O SFF e o Suporte Computacional Distribuído Associado	21
3.1. Introdução.....	21
3.2. Descrição Genérica da Aplicação Industrial.....	22
3.3. A Infra-estrutura de Comunicações e Computacional	24
3.3.1. Fluxos de Informação Horizontal	24
3.3.2. Fluxos de Informação Vertical.....	26
3.4. Detalhes dos Fluxos de Informação Horizontal.....	27
3.4.1. Tráfego Cíclico DP (DPH).....	28
3.4.2. Tráfego Não Cíclico DP (DPL)	31
3.4.3. Tráfego TCP/IP Encapsulado em DP (IPH & IPBE).....	32
3.5. Alguns de Detalhes Fluxo de Informação Vertical.....	36
Capítulo 4 – Cooperação entre Processos em Ambiente <i>Windows</i>.....	39
4.1. Introdução.....	39
4.2. Mensagens <i>Windows</i> – Conceitos Genéricos.....	40
4.3. Aspectos da Implementação do Mecanismo de Mensagens	41
4.3.1. Alguns Detalhes da Implementação.....	43
4.3.2. Referência Resumida	44
4.4. <i>Windows Sockets</i>	46
4.4.1. Alguns Detalhes de Utilização.....	47
4.4.2. Referência Resumida	47

Capítulo 5 – Clientes <i>Intranet</i>	51
5.1. Introdução	51
5.2. Aspectos Genéricos da Aplicação	52
5.3. Súmula das Comunicações com Outras Aplicações	59
5.4. Funcionalidades Suportadas: Monitorização do SFF	61
5.4.1. Monitorização dos Tapetes, <i>Buffers</i> e Robôs	61
5.4.2. Monitorização do SFF Através das Câmaras	65
5.4.3. Outros Aspectos de Monitorização do SFF	70
5.5. Funcionalidades Suportadas: Configuração do SFF e de Clientes de <i>Output</i>	70
5.5.1. Configuração de Parâmetros do SFF	71
5.5.2. Configuração de Clientes de <i>Output</i> e da <i>Gateway</i> SMS	72
5.6. Funcionalidades Suportadas: Comando de Dispositivos do SFF	74
5.7. Funcionalidades Suportadas: Geração e Visualização de Alarmes/Eventos	75
Capítulo 6 – Suporte da Interação entre Clientes <i>Intranet</i> e o SFF	77
6.1. Introdução	77
6.2. Suporte à Monitorização do SFF	79
6.2.1. Monitorização dos Tapetes, <i>Buffers</i> e Robôs	79
6.2.2. Monitorização do SFF Através das Câmaras	87
6.4. Suporte à Configuração do SFF e de Clientes de <i>Output</i>	92
6.4.1. Configuração de Parâmetros do SFF	92
6.4.2. Configuração de Clientes de <i>Output</i> e da <i>Gateway</i> SMS	94
6.5. Suporte ao Comando de Dispositivos do SFF	97
6.6. Suporte à Geração e Visualização de Alarmes/Eventos	98
Capítulo 7 – Conclusões	101
7.1. Revisão dos Objectivos e Contribuições da Dissertação	101
7.2. Trabalho Futuro	103
Referências	105
Anexo A – Detalhes VB6.0 da Aplicação Cliente <i>Intranet</i>	109
Anexo B – Detalhes VB6.0 da Aplicação Servidor <i>Intranet</i>	161
Anexo C – Detalhes VB6.0 da Aplicação RF - HMI	177

Lista de Figuras

Figura 2.1: Anel lógico de passagem de <i>token</i> entre mestres; e transacções mestre/escravo em redes industriais PROFIBUS.....	8
Figura 2.2: Topologia típica de uma rede RFieldbus	11
Figura 2.3: Protocolos para os nós <i>wired</i> e <i>wireless</i>	11
Figura 2.4: Diagrama temporal simplificado do mecanismo de gestão de mobilidade	13
Figura 2.5: Integração de aplicações TCP/IP na arquitectura protocolar PROFIBUS/RFieldbus	14
Figura 2.6: Alocações de tipos de tráfego numa rede RFieldbus com dois mestres	15
Figura 2.7: Exemplo de escalonamento ao nível do ACS	17
Figura 2.8: Funcionalidades do <i>IP-Mapper</i>	19
Figura 3.1: Diagrama esquemático do sistema de fabrico flexível.....	22
Figura 3.2: Vista parcial do SFF tal como está no Laboratório de Sistema Computacionais Críticos do ISEP (LASCRI).....	23
Figura 3.3: Diagrama esquemático da infra-estrutura de comunicações e computacional	24
Figura 3.4: Ilustração da sequência de passagem de <i>token</i>	28
Figura 3.5: A interface do operador com a aplicação RF-SCADA (módulo RF-HMI).....	30
Figura 3.6: A interface do operador com a aplicação RF-SCADA (módulo RF-CON), janela principal.....	31
Figura 3.7: A interface do operador com a aplicação RF-SCADA (módulo RF-CON), janela de configuração de parâmetros do mestre.....	31
Figura 3.8: Diagrama esquemático da infra-estrutura de comunicações e computacional, com particular relevo para os 8 fluxos de tráfego IP (MM# multimédia) encapsulado em tramas DP	32
Figura 3.9: Interface da aplicação de captura e processamento de imagem no PC6 (b) e de visualização e resultado da classificação no PC1 (a)	35
Figura 3.10: a) Contexto de utilização do <i>NetMeeting</i> numa plataforma RFieldbus; b) interface da aplicação desenvolvida em <i>ActiveX</i>	35
Figura 3.11: Aplicações que contribuem para a prestação de serviços aos clientes na <i>Intranet</i>	37
Figura 4.1: Fluxos de eventos/mensagens numa plataforma <i>Windows</i>	40
Figura 5.1: Janela de entrada (<i>Entry_Form</i>) da aplicação Cliente <i>Intranet</i>	52
Figura 5.2: As quatro janelas principais acessíveis a partir da <i>Main_Form</i> (a): <i>Monitor_Form</i> (b); <i>Control_Form</i> (c); <i>Config_Form</i> (d); <i>Gen_Events_Form</i> (e).....	54
Figura 5.3: Imagem do Compac iPAQ 3870 executando uma versão Cliente <i>Intranet</i> em eMVB3.0 (visualização da janela equivalente à <i>Config_Devices_Form</i> da versão para plataforma PC – Figura 5.8c deste capítulo)	55
Figura 5.4: A janela acessível a partir da <i>Main_Form</i> quando se selecciona a opção Monitor: <i>Monitor_Form</i> (b). A partir desta a opção <i>Belts & Robots</i> permite a visualização da janela <i>Monitor_BR_Form</i> (c). Por sua vez, ao seleccionar-se um dos equipamentos, é feita a visualização da janela <i>BR_Info_Form</i> (d – contexto RA1; e – contexto QUEUES; f – contexto AGV1).....	62

Figura 5.5: A aplicação RF-HMI e as definições de <i>Queues</i> e <i>Buffers</i>	63
Figura 5.6: A partir de <i>Monitor_Form</i> (b) pode seleccionar-se o Select_CAM_Info_Form (c). Deste, pode seleccionar-se, View_CAM_Form (d, e, f) uma câmara para visualizar (ou activar / desactivar a recepção de imagens dela provenientes). No exemplo, a recepção das CAM 1 e 3 está activada, enquanto a CAM 2 está desactivada. Na visualização de CAM 1 vê-se uma peça preta (B).	66
Figura 5.7: A partir de <i>Config_Form</i> (b) pode seleccionar-se o Config_Buffers_Form (c).....	71
Figura 5.8: A partir de <i>Config_Form</i> (b) pode seleccionar-se o Config_Devices_Form (c).....	73
Figura 5.9: Show_Events_Form da aplicação cliente.....	75
Figura 6.1: Aplicações de <i>software</i> que executam no PC1.....	78
Figura 6.2: Tipologia de Clientes <i>Intranet</i>	79
Figura 6.3: Símbolos utilizados para denotar o início (a) e uma acção intermédia (b) da interacção descrita.....	80
Figura 6.4: Interação despoletada com o evento <i>Main_Form.Form_Load</i> de uma aplicação Cliente <i>Intranet</i>	81
Figura 6.5: Interação despoletada com o evento <i>BR_Monitor_Form.Form_Load</i> de uma aplicação Cliente <i>Intranet</i>	82
Figura 6.6: Armazenamento na aplicação RF-HMI da informação contida em algumas mensagens <i>Windows</i> recebidas de RF-CON	84
Figura 6.7: Envio periódico (<i>Timer_Send_Serv_Timer</i>) aos Clientes <i>Intranet</i> de informação relativa aos <i>Buffers</i> , às <i>Queues</i> , aos AGVs e aos RAs	85
Figura 6.8: Interações resultantes do envio de mensagens <i>wmRF_OUTP</i> e <i>wmRF_MM</i> do RF-CON para o RF-HMI	86
Figura 6.9: Interação relativa ao pedido de um Cliente <i>Intranet</i> para subscrever um determinado fluxo de vídeo de uma câmara específica do SFF (caso em que esse fluxo de vídeo não tem ainda nenhuma subscrição).....	87
Figura 6.10: Interação relativa ao pedido de um Cliente <i>Intranet</i> para subscrever um determinado fluxo de vídeo de uma câmara específica do SFF (caso em que esse fluxo de vídeo já tem pelo menos uma subscrição)	88
Figura 6.11: Interação relativa ao pedido de um Cliente <i>Intranet</i> para indicar o fim de subscrição (caso em que esse fluxo de vídeo passa a não ter subscrições).....	89
Figura 6.12: Interação relativa ao pedido de um Cliente <i>Intranet</i> para indicar o fim de subscrição (caso em que esse fluxo de vídeo continua a ter outras subscrições).....	90
Figura 6.13: Acção de desactivação de um fluxo de vídeo (<i>chk_CLI_Cam_Click()</i>) utilizando o interface do Servidor <i>Intranet</i>	90
Figura 6.14: Envio de fragmentos (mensagens UDP) de imagens por parte de IMG- CENTRE	91
Figura 6.15: Pedido de alteração do tamanho máximo feiro pelo Cliente C1	92
Figura 6.16: Interações resultantes da mensagem <i>wmRF_BUFF_CNF</i> enviada pelo RF- CON ao RF-HMI	93
Figura 6.17: Exemplo de configuração do Cliente HMD (enviada pelo Cliente C1).....	94
Figura 6.18: Exemplo de configuração da <i>Gateway SMS</i> (enviada pelo Cliente C1)	95
Figura 6.19: Exemplo de configuração do Cliente HMD (enviada pelo Servidor <i>Intranet</i>)	96
Figura 6.20: Exemplo de configuração da <i>Gateway SMS</i> (enviada pelo Servidor <i>Intranet</i>).....	96

Figura 6.21: Interações resultantes do envio de comandos ao SFF	97
Figura 6.22: Interações na recepção de informação de actuação por parte do RF - CON.....	98
Figura 6.23: Interações no caso da geração de um alarme/evento no Cliente C1	98
Figura 6.24: Interações no caso da geração de um alarme/evento no Servidor <i>Intranet</i>	99
Figura 6.25: Interações resultantes do evento/alarme relativo ao SFF (<i>buffer full</i>)	100
Figura A.1: <i>Layout</i> da Entry_Form.....	112
Figura A.2: <i>Layout</i> da Main_Form.....	113
Figura A.3: <i>Layout</i> da Monitor_Form.....	125
Figura A.4: <i>Layout</i> da Monitor_BR_Form.....	127
Figura A.5: <i>Layout</i> da BR_Info_Form.....	130
Figura A.6: <i>Layout</i> da Select_CAM_Info_Form.....	133
Figura A.7: <i>Layout</i> da CAM_View_Form.....	135
Figura A.8: <i>Layout</i> da Control_Form.....	139
Figura A.9: <i>Layout</i> da Config_Form das aplicações cliente <i>Intranet</i>	142
Figura A.10: <i>Layout</i> da Config_Buffers_Form	144
Figura A.11: <i>Layout</i> da Config_Devices_Form	148
Figura A.12: <i>Layout</i> da Gen_Events_Form.....	152
Figura A.13: <i>Layout</i> da Show_Events_Form	154
Figura A.14: Imagens utilizadas para a propriedade <code>picture</code> das forms	156
Figura A.15: Outras imagens utilizadas para a propriedade <code>picture</code> das forms.....	157
Figura A.16: Imagens utilizadas para a propriedade <code>picture</code> do objecto <code>Picture1</code> da form <code>Show_Events_Form</code>	158
Figura B.1: <i>Layout</i> da Server_Form e objectos relacionados com a monitorização do estado do SFF	161
Figura B.2: <i>Layout</i> da Server_Form e objectos relacionados com a monitorização e gestão da subscrição (por parte de Clientes <i>Intranet</i>) e comandos de activação de vídeo <i>streams</i> UDP por parte do <i>Image Center</i>	162
Figura B.3: <i>Layout</i> da Server_Form e objectos relacionados com a geração de alarmes/eventos para os Clientes <i>Intranet</i>	162
Figura B.4: <i>Layout</i> da Server_Form e objectos relacionados com a geração a configuração de Clientes <i>Intranet</i> de capacidade limitada (HMD/PK1/PK2) e com a configuração da <i>Gateway</i> SMS.....	163
Figura C.1: <i>Layout</i> da HMI_Form	177

Lista de Tabelas

Tabela 2.1: Exemplo e caracterização de 5 fluxos IPH	17
Tabela 2.2: Tabela de Escalonamento para o exemplo	18
Tabela 3.1: Características dos dispositivos ligados à rede PROFIBUS/RFieldbus	25
Tabela 3.2: Características dos dispositivos ligados à <i>Intranet</i>	27
Tabela 3.3: Funcionalidades suportadas pelo tráfego cíclico PROFIBUS/RFieldbus DP	29
Tabela 3.4: Funcionalidades, protocolos de transporte e classes de serviço RFieldbus utilizadas nos fluxos de tráfego TCP/IP encapsulado em DP	34
Tabela 3.5: <i>Hardware</i> e <i>software</i> para suportar as funcionalidades previstas para os fluxos de tráfego TCP/IP encapsulado em DP	36
Tabela 5.1: Arrays de <code>CommandButton</code> definidos em <code>Globals.bas</code> e sua associação lógica aos diversos <code>forms</code> da aplicação Cliente <i>Intranet</i>	57
Tabela 5.2: Tipos de pacotes UDP enviados pelo Cliente <i>Intranet</i>	60
Tabela 5.3: Tipos de mensagens UDP recebidos no Cliente <i>Intranet</i>	61
Tabela 5.4: Exemplo de <code>Info_Monitor(id_dev)</code> no procedimento <code>Arrived_Monitor_Update</code>	64
Tabela 5.5: Mensagens UDP recebidas com o primeiro carácter a “<”, e processadas em <code>Main_Form.Arrived_Video_Info</code>	67
Tabela 5.6: Mensagens recebidas sendo o primeiro carácter um “#” e processadas em <code>Arrived_Control_Update</code> da <code>Main_Form</code>	70
Tabela 5.7: Mensagens enviadas com o primeiro carácter “#” na <code>Control_Form</code>	74
Tabela 6.1: Mensagens enviadas de RF-CON para RF-HMI	83
Tabela A.1: Relação de objectos relevantes da <code>Entry_Form</code>	112
Tabela A.2: Relação de objectos relevantes da <code>Main_Form</code>	114
Tabela A.3: Relação de objectos relevantes da <code>Monitor_Form</code>	125
Tabela A.4: Relação de objectos relevantes da <code>Monitor_BR_Form</code>	127
Tabela A.5: Relação de objectos relevantes da <code>BR_Info_Form</code>	130
Tabela A.6: Relação de objectos relevantes da <code>Select_CAM_Info_Form</code>	133
Tabela A.7: Relação de objectos relevantes da <code>CAM_View_Form</code>	136
Tabela A.8: Relação de objectos relevantes da <code>Control_Form</code>	139
Tabela A.9: Relação de objectos relevantes da <code>Config_Form</code>	142
Tabela A.10: Relação de objectos relevantes da <code>Config_Buffers_Form</code>	145
Tabela A.11: Relação de objectos relevantes da <code>Config_Devices_Form</code>	148
Tabela A.12: Relação de objectos relevantes da <code>Gen_Events_Form</code>	152
Tabela A.13: Relação de objectos relevantes da <code>Show_Events_Form</code>	154
Tabela B.1: Relação de objectos relevantes da <code>Server_Form</code>	164

Capítulo 1

Panorâmica Global da Dissertação

O principal objectivo desta dissertação é o de demonstrar a exequibilidade do desenvolvimento e da utilização de aplicações multimédia de uso genérico em sistemas de informação industriais associados à automatização de processos de fabrico. Este capítulo fornece uma visão geral do domínio tecnológico, e estabelece o contexto e os objectivos deste trabalho.

1.1. Motivação

Desde há alguns anos, *gadgets*¹ como assistentes pessoais digitais (*Personal Digital Assistants* – PDA), telemóveis e câmaras digitais têm vindo a ganhar uma crescente popularidade e adesão, mesmo no universo de utilizadores menos familiarizados com as novas tecnologias de informação e comunicação (TIC). Contudo, e por diversas razões, os ambiente fabris, e em particular os sistemas de fabrico automatizado, continuam quase herméticos a esta nova realidade.

Uma das principais justificações prende-se com a natureza pouco evolutiva do equipamento de automação industrial, que tipicamente tem um tempo de vida espectável consideravelmente longo. Por outro lado, as TIC vocacionadas para os ambientes de automação industrial têm de cumprir requisitos apertados de confiança no funcionamento, de robustez, de funcionamento em tempo real e de segurança. Por isso, normalmente, baseiam-se em sistemas computacionais com especificações e funcionalidades diversas das TIC vocacionadas para outros ambientes de aplicação, nos quais os requisitos atrás referidos têm eventualmente níveis de preocupação mais baixos.

Acredita-se, porém, que as novas abordagens de interacção pessoa-máquina, incluindo tecnologias multimédia e as técnicas de realidade aumentada, combinadas com a crescente disponibilidade de tecnologias de computação ubíqua, permitirão modificar, para melhor, a forma como os operários fabris interagem com as máquinas e com os sistemas de informação da planta fabril (Pacheco e Tovar, 2002).

¹ Termo inglês utilizado para designar um tipo específico de máquinas – “A *gadget* is a **cleverly** designed **small** machine, often one that does a complicated action, and is usually **modern**” (Longman, 2005).

1.2. Multimédia em Ambiente Industrial: Aplicações Prospectivas

O termo multimédia é usado para referir a multiplicidade de diferentes tipos e formatos de informação (texto, gráficos, imagem, som, animação, vídeo, etc.) utilizados para um determinado objectivo comum. No âmbito das TIC, o conceito multimédia começou a ser badalado no início dos anos 90, e a ganhar um impacto crescente com a vulgarização da *Internet*.

A profusão de aplicações multimédia deveu-se à crescente capacidade das TIC em processar, armazenar e disponibilizar formatos de informação ricos, que exigem, quando comparados com os rudimentares formatos texto/caracteres, maior largura de banda por parte dos sistemas de comunicação, maior capacidade de processamento por parte dos processadores, e maior capacidade de armazenamento por parte dos sistemas de armazenamento de dados.

A tecnologia multimédia e as modernas TIC permitem novas filosofias de interacção entre os utilizadores e os sistemas de informação. Importa, por isso, sistematizar alguns conceitos, em paralelo com a referência a potenciais aplicações para ambientes fabris.

Do ponto de vista das infra-estruturas de computação, a realidade crescente é a da proliferação de microprocessadores por todo o lado. Hoje em dia, as plataformas computacionais não são só os PCs (ou outras plataformas tradicionais), mas são também os telemóveis, os PDAs e as plataformas de jogos, só para referir algumas soluções menos “tradicionais”. Os microprocessadores aparecem em quantidade dentro de veículos de transporte, ligados em rede, e com capacidades de transmissão de dados multimédia (Most, 2004), nos mais insuspeitos electrodomésticos e máquinas de venda automática, em edifícios (domótica), etc.

Esta nova realidade de microprocessadores em todo o lado (omnipresença), configura o paradigma a que se convencionou chamar de computação ubíqua (do inglês, *ubiquitous computing*). Ubiquidade é o dom de alguém (ou de algo) poder estar ao mesmo tempo em vários lugares. Neste contexto de ubiquidade, a eficácia e flexibilidade no processamento de informação resulta da cooperação entre diversas (eventualmente pequenas) unidades de computação. Daí o conceito de *ubiquitous computing* aparecer, na literatura, como sinónimo de *disappearing computer* (Schenker, 2000).

A mobilidade (e as comunicações *wireless*) e a crescente portabilidade dos dispositivos computacionais tornam praticável a existência de *wearable computers* (computadores “vestíveis”).

Imagine-se um operário fabril com um PDA dotado de interface de comunicações *wireless*, associado a um monóculo contendo um pequeno ecrã (*Head Mounted Display* – HMD), e a um conjunto auscultador/microfone áudio. A panóplia de aplicações que se pode imaginar é enorme. Imagine-se que o sistema de informação fabril, com a qual o PDA interage, detecta a localização do operário numa área específica da fábrica. Ao ser detectado nessa área, o operário pode automaticamente receber informações no seu monóculo (texto, sinópticos, etc.) ou via auscultador (som), sobre a necessidade de intervir numa determinada máquina desse sector da fábrica.

Interacções potenciadas pela capacidade de localização física da unidade de computação (*location awareness*) (Jonker e Caarls, 2003) são só um exemplo dos novos paradigmas de computação centrados no utilizador (e na interacção com o utilizador). Podemos continuar com mais alguns exemplos ilustrativos.

Imagine que, associado ao HMD há uma tecnologia de seguimento do movimento dos olhos (*eye tracking*), que permite que interações sejam despoletadas em função do foco de visão do operário. Se o HMD permitir visão humana através dele (*see-through*), pode ser disponibilizada ao operário uma funcionalidade de realidade aumentada (*augmented reality*): ele poderá ver uma realidade física enriquecida com informação (gráfica, textual, som, etc.) adicional sobre essa realidade física. Por exemplo, ao olhar para uma determinada máquina, pode aparecer um sinóptico no monóculo que indica a necessidade da sua intervenção para substituir uma ferramenta da máquina. O operador poderá então recorrer aos esquemáticos de processo (típicos dos sistemas de supervisão, controlo e aquisição de dados – SCADA) e, a partir do seu PDA, através de hiperligações, consultar de forma rápida e eficiente documentação e outra informação multimédia disponíveis em rede. Ou então, usar áudio e vídeo para comunicação inter-pessoal com outros funcionários da fábrica.

1.3. As Redes de Comunicação Industriais

Os sistemas de automação fabril utilizam, tipicamente, arquitecturas distribuídas de computação. São, por isso, sistemas distribuídos de controlo computadorizado (*Distributed Computer-Controlled Systems* – DCCS) (Tovar, 1999). Neste tipo de sistemas, diversos módulos de *software* e *hardware* cooperam com vista a um objectivo comum. A abordagem usual é suportar essa cooperação por recurso a uma infra-estrutura de comunicações especialmente vocacionada para o cumprimento dos requisitos específicos dos sistemas de fabrico automatizado. Este tipo de redes de comunicação industrial é usualmente conhecido rede de campo (*fieldbus*).

As redes do tipo *fieldbus* são redes de comunicação industriais capazes de satisfazer, entre outros, os requisitos de correcto funcionamento temporal exigido pelas aplicações distribuídas. Permitem, por isso, um suporte adequado a sistemas distribuídos de tempo-real. Note-se que num sistema computacional de tempo-real, o correcto funcionamento não depende só da correcção lógica do resultado da computação, mas também do instante de tempo no qual o resultado é produzido (Stankovic, 1988).

As redes do tipo *fieldbus* têm outras especificidades, que vão desde as características físicas da cablagem e da transmissão de dados, até aos serviços específicos da camada de aplicação (AL), passando pela forma como são concretizadas em termos de plataformas computacionais (normalmente os nós são sistemas computacionais embebidos, relativamente complexos, e com interfaces de sensorização e actuação).

Existem inúmeras soluções comerciais de redes do tipo *fieldbus*, quase todas baseadas em normas internacionais de especificação independentes dos fabricantes (normas abertas). Uma das soluções comerciais mais importantes, quer pela sua penetração no mercado, quer pelo número de empresas que a suportam (entre outras a Siemens), baseia-se na norma internacional PROFIBUS (acrónimo de PROcess FIeld BUS).

Uma rede PROFIBUS (EN 50170, 1996; Tovar e Vasques, 1999) baseia-se no modelo de camadas OSI (*Open Systems Interconnection*) da ISO (*International Standards Organization*), apesar de só três camadas serem suportadas: camada física (*Physical Layer* – PhL); camada de ligação de dados (*Data Link Layer* – DLL) e camada de aplicação (*Application Layer* – AL). Utiliza por isso o modelo colapsado (Cardoso e

Tovar, 1994) do modelo de sete camadas OSI. Trata-se de uma rede com fios (*wired*), na qual, do ponto de vista da partilha do meio de comunicação (um barramento de comunicações série multiponto – *bus*), existem dois tipos de nós de rede: mestres e escravos. O PROFIBUS especifica um mecanismo de passagem de testemunho (*token*), semelhante ao *token-passing bus* da norma IEEE 802.4 (IEEE802.4, 1985). Este mecanismo é utilizado pelos mestres para aceder, em partilha temporal, ao meio de comunicação. Os nós escravo não têm iniciativa de comunicação, podendo apenas responder a pedidos feitos por mestres.

1.4. A Iniciativa RFieldbus

Nas décadas de 80 e 90, o advento das redes do tipo *fieldbus* quebrou com a tradicional filosofia de controlo centralizado e comunicações ponto-a-ponto vigente até então nos sistemas de automação de fabrico. Nos anos mais recentes, as pressões do mercado e a crescente disponibilidade de técnicas e tecnologias relacionadas com as modernas TIC e com os *gadgets*, têm levado os fabricantes a fazer evoluir as soluções existentes de forma a torná-las compatíveis com uma realidade mais ubíqua e interactiva do ambiente fabril. Para isso, duas funcionalidades cruciais (adicionais) são essenciais às redes do tipo *fieldbus*.

1. Suporte de mobilidade de dispositivos, e por isso a necessidade de integrar capacidades de comunicações sem fios (*wireless*) adequadas ao ambiente industrial.
2. Suporte de tráfego multimédia, e por isso a eventual necessidade de integrar protocolos e funcionalidades normalmente encontrados noutro tipo de tecnologias de comunicação, como é o caso do conjunto de protocolos associados à *Internet*.

O requisito *wireless* é intuitivamente óbvio. O requisito de compatibilidade com os protocolos associados à *Internet* resulta do facto de aplicações como monitorização associada a câmaras e microfones digitais, de acesso remoto a bases de dados de manutenção e de guias de operações de fabrico (incluindo gráficos, vídeos e som), entre outras, serem aplicações largamente difundidas e suportadas pela pilha de protocolos de comunicações TCP/IP (*Transmission Control Protocol / Internet Protocol*) (Rodriguez *et al.*, 2001), respectivamente protocolos de camada de transporte e de camada de rede. Tipicamente, as redes de *fieldbus* não especificam nem suportam protocolos de camada de rede (*Network Layer* – NL) nem de camada de transporte (*Transport Layer* – TL). Daí os esforços no sentido de incorporar mecanismos de suporte a comunicações sem fios (Alves, 2003), e no sentido de incorporar o suporte a tráfego e aplicações TCP/IP (Pacheco *et al.*, 2001; Pereira *et al.*, 2002; Tovar *et al.*, 2003).

Estes trabalhos enquadraram-se num importante projecto Europeu: o RFieldbus (*High Performance Wireless Fieldbus in Industrial-Related Multimedia Environment*), o projecto IST-1999-11316 do subprograma IST (*Information Society Technologies*) do quinto programa-quadro de I&D da União Europeia. O objectivo essencial do projecto (Rauchhaupt, 2002) foi o de estender a norma PROFIBUS de forma a ser possível suportar nós móveis (via comunicações rádio) e tráfego multimédia (via encapsulamento de fragmentos IP em tramas PROFIBUS).

O projecto RFieldbus desenvolveu-se entre Janeiro de 2000 e Março de 2003, envolvendo três parceiros fornecedores de tecnologia, a Siemens (D), a Softing (D) e a Temex ST2E (F), e três instituições de I&D, o ISEP (através do grupo de I&D IPP-HURRAY), o ifak (D) e o ISI (GR). As fases de projecto, especificação e implementação das tecnologias conduziram a uma fase final de demonstração, através de duas instalações piloto: uma ligada ao controlo de processos petroquímicos (na empresa Lube Processing Corporation – LPC, Atenas, Grécia); outra ligada a sistemas de fabrico de componentes discretos (Piloto, 2003; Tovar *et al.*, 2003) desenvolvida no Laboratório de Sistemas Críticos do ISEP (LASCRI).

1.5. Objectivos da Dissertação e Hipótese

A tecnologia RFieldbus disponibiliza as funcionalidades de uma rede de *fieldbus* típica: o PROFIBUS. É, porém, uma tecnologia enriquecida com a capacidade de interligação a nós móveis via rádio, e com a capacidade de transferir tráfego multimédia normalizado encapsulado.

O sistema de fabrico implementado no piloto de demonstração do ISEP utiliza essas funcionalidades nos fluxos de informação directamente relacionados com os diversos controladores que comandam as diferentes operações do equipamento industrial. Designamos o tráfego relacionado com estes fluxos de informação por fluxo horizontal de informação, tipicamente correspondendo a operações de computação distribuídas com requisitos temporais críticos. No Piloto, mesmo ao nível do fluxo horizontal, é utilizado tráfego IP para suportar funções de controlo (imagens digitais utilizadas na decisão de manipulação).

A “integração” do conjunto de protocolos TCP/IP no PROFIBUS abre todo um conjunto de possibilidades de novas formas de interacção entre operadores e o sistema de fabrico, como algumas das que foram mencionadas na Secção 1.2.

O objectivo desta tese é o de ilustrar e concretizar o desenvolvimento de aplicações de *software*, demonstrando, por essa via, a hipótese de que, com as novas tecnologias de informação e comunicação, é viável uma perspectiva diferente da forma como os operários fabris podem interagir com as máquinas e com os sistemas de informação ao nível do controlo de operações de fabrico.

1.6. Abordagem

A tecnologia RFieldbus foi desenvolvida, na sua versão PC, para plataformas *WindowsNT*. A implementação visada neste trabalho passa, por isso, pelo desenvolvimento de aplicações *Windows*. Uma, designada por **Servidor Intranet**, interage com as aplicações de controlo, supervisão e aquisição de dados (SCADA) desenvolvidas para o controlo do sistema de fabrico (fluxo de informação horizontal). A aplicação Servidor *Intranet* permite o fluxo de comunicações verticais com **Clientes Intranet** e clientes residentes em plataformas computacionais “exteriores” ao sistema distribuído de controlo RFieldbus. Esta dissertação aborda com um foco especial a especificação e o desenvolvimento das aplicações Cliente *Intranet*. Estas aplicações

cliente permitem, como se verá, formas de interação avançadas entre os operadores e o sistema de automação industrial. A aplicação Servidor *Intranet* é alojada na mesma plataforma computacional utilizada para alojar as aplicações SCADA, pelo que estas interagem entre si utilizando *Windows Messages* e *Windows Sockets*.

Esta dissertação abordará, em particular, as especificidades do Servidor *Intranet*, das componentes das aplicações SCADA para a interação com esta, e das aplicações desenvolvidas para os Clientes *Intranet*.

1.7. Organização da Dissertação

O foco e as contribuições principais desta dissertação estão relacionados com o fluxo de comunicações vertical, e em particular com as aplicações de *software* referidas na Secção 1.6. É, contudo, necessária uma contextualização detalhada relativamente às funcionalidades do sistema de fabrico industrial utilizado como demonstração, o qual, por sua vez, exige a compreensão de algumas das especificidades da tecnologia de comunicações PROFIBUS/RFieldbus.

Por isso, a dissertação está organizada do seguinte modo. O Capítulo 2 é dedicado às tecnologias PROFIBUS e RFieldbus. Os conceitos fundamentais no contexto desta tese são descritos. O Capítulo 3 descreve os aspectos essenciais quer do sistema de fabrico automatizado, quer das infra-estruturas de comunicação e das aplicações utilizadas para o controlar. Dado o seu relevo em alguns aspectos da implementação das aplicações envolvidas no suporte ao fluxo vertical, é feita, no Capítulo 4, uma descrição dos mecanismos de comunicações entre essas aplicações, quer os que têm por base as mensagens da API (*Application Programming Interface*) do Windows, quer os que têm por base os *Sockets*, em ambos os casos, num contexto de programação *Visual Basic 6.0*. Os Capítulos 5 e 6 são capítulos chave desta dissertação. O Capítulo 5 é dedicado às aplicações cliente *Intranet*. O Capítulo 6 descreve os aspectos fundamentais do suporte ao fluxo de comunicações verticais, incluindo os detalhes de interação entre as diferentes aplicações em apreço: SCADA, Servidor *Intranet*, Clientes *Intranet* e outras aplicações de suporte. Finalmente, no Capítulo 7, são apresentadas conclusões e perspectivas de desenvolvimento futuro.

Capítulo 2

Aspectos Relevantes do PROFIBUS/RFieldbus

Neste capítulo são descritos alguns dos aspectos mais relevantes da tecnologia PROFIBUS e das extensões entretanto desenvolvidas no âmbito do projecto Europeu RFieldbus. O nível de detalhe utilizado é o adequado à contextualização dos restantes capítulos desta dissertação.

2.1. Introdução

As redes de comunicação industriais têm, desde há uns anos, vindo a ganhar uma grande popularidade no suporte ao controlo de sistemas automatizados de fabrico. Este tipo de redes permite que dispositivos de campo (sensores, actuadores e controladores), sejam interligados a baixo custo, utilizando uma menor quantidade de cablagem, e por via disso necessitando de menos manutenção do que as soluções de controlo centralizadas baseadas em ligações ponto a ponto (Lenhart, 1993). Para além destas vantagens, a utilização de redes de comunicação para o suporte de sistemas industriais controlados por unidades computacionais vai ao encontro da tendência generalizada de descentralização de tarefas de controlo, de medida, e de actuação.

As redes locais de computadores (*Local Area Networks* – LANs) utilizadas para a interligação de sensores, actuadores e controladores são, como já referido no capítulo anterior, usualmente designadas por redes de campo (*fieldbus*).

À semelhança de outras LANs, as redes *fieldbus* baseiam-se numa estrutura de camadas protocolares derivada do modelo de referência OSI de sete camadas (Day e Zimmermann, 1983). No entanto, e para fazer face a requisitos específicos, a estrutura protocolar das redes do tipo *fieldbus* é simplificada e optimizada, pelo que normalmente apenas três das sete camadas do modelo de referência OSI são consideradas: a camada de aplicação, a camada de ligação de dados e a camada física.

Desde o aparecimento das primeiras propostas comerciais de redes *fieldbus*, o seu âmbito de aplicação e as suas funcionalidades tem vindo a ser alargado (Thomesse, 1997). Nos últimos anos, dois dos aspectos de desenvolvimento que mais atenções têm recebido, dizem respeito à capacidade das redes *fieldbus* em suportarem nós de rede com mobilidade e em suportarem aplicações e tráfego multimédia. Daí os esforços no sentido de incorporar mecanismos de suporte a comunicações sem fios (Alves, 2003), e no sentido de incorporar o suporte a tráfego e aplicações TCP/IP (Tovar *et al.*, 2003).

Este capítulo descreve as características fundamentais de uma solução *fieldbus*, o PROFIBUS, e dos desenvolvimentos efectuados no âmbito do projecto Europeu

RFieldbus (Rauchhaupt, 2002), no sentido de dotar o PROFIBUS com as funcionalidades anteriormente referidas.

2.2. Características Fundamentais do PROFIBUS

À semelhança de outras redes do tipo *fieldbus*, a arquitectura protocolar do PROFIBUS baseia-se no modelo OSI, mas considerando um modelo colapsado só com três camadas: a camada física (*Physical Layer* – PhL), a camada de ligação de dados (*Data Link Layer* – DLL) e a camada de aplicação (*Application Layer* – AL).

Em PROFIBUS existem dois tipos de nós de rede: mestres e escravos (Figura 2.1). Trata-se por isso de uma rede do tipo múltiplo mestre, na qual o controlo do acesso ao meio é gerido através de um protocolo de passagem de testemunho (*token*) temporizado entre mestres (estações activas). Este protocolo é uma versão simplificada do *Timed Token Protocol* (TTP) proposto por Grow (Grow, 1982), e cujas variantes são utilizadas noutras normas de redes de comunicação, como por exemplo o *IEEE802.4 Token Bus* (IEEE 802.4, 1985) ou a *Fiber Distributed Data Interface* (FDDI) (ISO 9314-2, 1989). O TTP confere um controlo de acesso distribuído aos mestres PROFIBUS. Enquanto está na posse do *token*, um mestre PROFIBUS pode executar transacções de mensagens com outros nós da rede, quer sejam mestres ou escravos. Os escravos não têm iniciativa de comunicação, sendo que apenas respondem a pedidos efectuados pelos mestres. Por essa razão, os escravos são também designados por estações passivas.

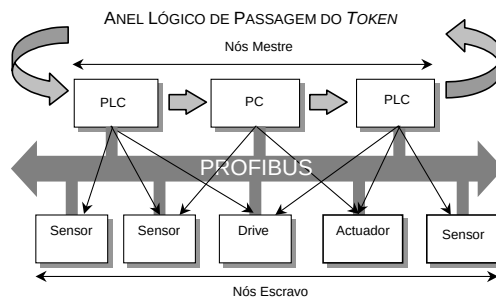


Figura 2.1: Anel lógico de passagem de *token* entre mestres; e transacções mestre/escravo em redes industriais PROFIBUS

Nas subsecções seguintes serão descritos alguns aspectos relevantes relacionados com a arquitectura protocolar do PROFIBUS.

2.2.1. Camada de Aplicação

O PROFIBUS considera três perfis de aplicação diferentes: FMS, DP e PA. O PROFIBUS FMS (*Field Message Specification*) é a solução universal, fornecendo uma gama alargada de funcionalidades e serviços de aplicação. O PROFIBUS DP (*Decentralised Periphery*) é a versão do PROFIBUS mais adequada a aplicações com

características de tempo-real, fornecendo uma extensão à norma EN 50170 (EN 50170, 1996) no que respeita aos requisitos específicos destas aplicações. O PROFIBUS PA (*Process Automation*), tal como o nome sugere, especifica funcionalidades adequadas a aplicações na área do controlo de processos contínuos (por exemplo, indústria petroquímica).

Pelas suas características, a versão DP do PROFIBUS é a utilizada no âmbito deste trabalho.

2.2.2. Camada de Ligação de Dados

A Camada de Ligação de Dados (DLL) define os protocolos de controlo de acesso ao meio de comunicação (*Medium Access Control* – MAC), e os diferentes tipos de serviços de ligação de dados, entre outros aspectos.

No PROFIBUS, o acesso ao meio de comunicação é conferido pela posse do *token*. O *token* (uma mensagem específica PROFIBUS) circula entre os mestres da rede por ordem ascendente: do mestre com o endereço MAC mais baixo até ao mestre com o endereço MAC mais alto, e deste último para o de endereço MAC mais baixo novamente, formando-se assim um anel lógico de rotação do *token* entre as estações activas da rede.

Este esquema implica que cada mestre conheça, para além do seu próprio endereço, os endereços das estações anterior e posterior no anel lógico. O PROFIBUS considera diversos mecanismos de gestão do anel lógico. Estes permitem suportar alterações da composição do anel (entrada/saída de estações activas) ou fazer face a situações de erro (perda do *token* ou duplicação do *token*).

Aquando da recepção do *token*, um mestre PROFIBUS pode executar transacções. Tipicamente, uma transacção corresponde a um envio de um pedido (*request*) por parte do mestre e à recepção de uma resposta (*response*) proveniente de um escravo (ou de um outro mestre). No PROFIBUS são considerados dois tipos de transacções: as de baixa prioridade e as de alta prioridade. Durante a posse do *token*, estas transacções são processadas de acordo com o tempo ainda disponível de posse do *token*, que é gerido da forma que, sucintamente, se explica a seguir.

Quando um mestre recebe o *token*, pode iniciar transacções com outras estações da rede durante um período de tempo designado por T_{TH} (*Token Holding Time*). O T_{TH} é obtido dinamicamente num mestre, em cada rotação do *token*, da seguinte forma:

$$T_{TH} = T_{TR} - T_{RR} \quad (2.1)$$

onde T_{TR} significa *Target token Rotation Time*, e T_{RR} significa *Real token Rotation Time*. O valor obtido para T_{TH} é utilizado para carregar um temporizador que controla o número de transacções processadas durante a posse do *token*. Após expirar o tempo T_{TH} , o mestre tem de passar o *token* ao seu sucessor no anel lógico.

O tempo de rotação do *token* (T_{RR}) é medido por cada mestre, em cada chegada do *token*, e corresponde ao tempo decorrido entre o instante dessa chegada e o da chegada na visita anterior.

O T_{TR} é um parâmetro que é definido nos mestres (terá de ser igual em todos os mestres da rede) antes da rede entrar em funcionamento. A forma de determinar o seu

valor entra em consideração com diversos parâmetros, entre os quais o número de mestres da rede ou número e o tipo de transacções a serem efectuadas pelos mestres. Algumas dessas metodologias são descritas em (Tovar e Vasques, 1999; Tovar e Vasques, 1999b), e visam, no essencial, garantir o cumprimento de requisitos temporais das transacções entre as estações da rede.

Como já foi referido anteriormente, o tempo de posse do *token* (T_{TH}) é obtido dinamicamente em cada rotação. Pode acontecer que, numa determinada rotação, a diferença entre o T_{TR} e o T_{RR} seja negativa ($T_{TH} < 0$). Nesse caso o *token* “está em atraso”, e a estação que o recebe só poderá processar uma transacção correspondente a uma mensagem de alta prioridade, tendo depois de passar o *token* à sua estação sucessora.

As mensagens de alta prioridade correspondem, normalmente, a tráfego de controlo com requisitos temporais apertados. Durante o tempo de posse do *token*, uma estação executa todo o tráfego de alta prioridade (*DP High* – DPH) pendente, e só depois as mensagens de baixa prioridade (*DP Low* – DPL).

Por fim, uma referência aos serviços da camada de ligação de dados. A transferência de dados entre as estações é suportada pelos seguintes tipos de serviços da DLL.

SDA – Send Data with Acknowledge. Permite o envio de dados para uma única estação remota (*unicast*). No caso da ocorrência de um erro, a transacção deve ser repetida.

SDN – Send Data with No acknowledge. Permite o envio de informação para uma única estação, para múltiplas estações (*multicast*) ou para todas as estações da rede (*broadcast*), ao mesmo tempo, sem confirmação.

SRD – Send and Request Data. Permite o envio de informação para uma única estação e, ao mesmo tempo, requerer informação à mesma estação. Se ocorrer algum erro, deve haver uma retransmissão.

CSRD – Cyclic Send and Request Data. O serviço CSRD permite enviar mensagens SRD ciclicamente para um conjunto de estações definidas numa lista a que vulgarmente se designa por *Poll List*.

2.2.3. Camada Física

A implementação mais vulgar do PROFIBUS contém apenas uma estação mestre, que actua como a unidade central de processamento de todo o sistema, embora sejam permitidos até um máximo de 32 mestres na rede. São suportadas várias velocidades de transmissão que dependem do meio físico implementado. No caso do PROFIBUS DP, com meio físico baseado na norma RS-485, a velocidade de transmissão máxima típica é de 1.5 Mbps.

Uma rede PROFIBUS pode ter no máximo 127 estações, incluindo mestres, escravos e repetidores, mas o número de estações por segmento físico (ou entre repetidores) é limitado a 32.

2.3. Extensões para o Suporte de Mobilidade no PROFIBUS

Uma rede *fieldbus* consiste, tipicamente, num conjunto de nós fisicamente ligados por um barramento de comunicações série (*wired*). Consequentemente, e tendo em consideração a penetração das soluções *wired* no mercado, a incorporação de nós *wireless* favorece uma solução híbrida *wired/wireless*, onde os nós móveis sem fio podem interoperar com os nós “tradicionais” (*wired*). No âmbito do Projecto Europeu RFieldbus, foi especificada, implementada e validada uma arquitectura deste tipo (Alves, 2003). A Figura 2.2 ilustra uma topologia RFieldbus típica (M – Master; S – Slave).

Os nós *wired* e os nós móveis comunicam entre si através dispositivos de interligação (IS – *Intermediate Systems*), que transferem tramas da camada física (PhL – *Physical Layer*) entre os domínios *wired* (segmentos de rede PROFIBUS) e *wireless* (células rádio).

No RFieldbus, foram definidos três tipos de ISs: *Link Stations* (LSs), *Base Stations* (BSs) e *Link Base Stations* (LBSs). As LSs apenas transferem tramas entre os domínios *wired* e *wireless*. As BSs estruturam células rádio, recebendo tramas num canal rádio (*uplink*) e retransmitindo-as noutra canal (*downlink*). As LBSs integram simultaneamente as funcionalidades associadas às LSs e LBSs.

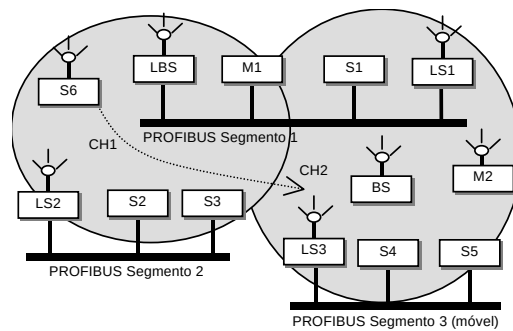


Figura 2.2: Topologia típica de uma rede RFieldbus

Os ISs transferem tramas ao nível da camada física, funcionando como repetidores (do tipo *cut-through*). Desta forma, os domínios *wireless* podem ser considerados como simples extensões dos domínios *wired* (segmentos PROFIBUS), onde todos os nós recebem todas as tramas. Os protocolos adoptados para as várias camadas estão ilustrados na Figura 2.3.

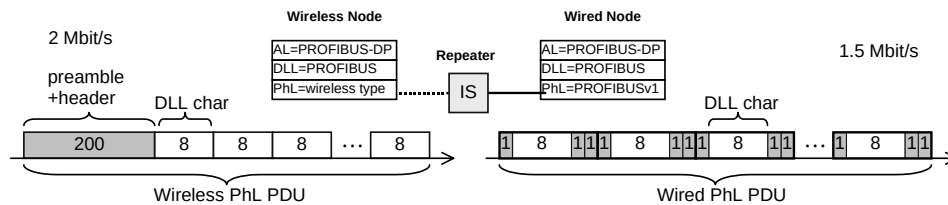


Figura 2.3: Protocolos para os nós *wired* e *wireless*

A camada de aplicação (AL) e a camada de ligação de dados (DLL) adoptadas são o PROFIBUS-DP e o PROFIBUS, respectivamente. Contudo, a camada física utilizada pelos nós *wired* é o PROFIBUS PhL v1 (RS-485, assíncrono, 1.5 Mbps), enquanto que para os nós *wireless* é utilizado um protocolo baseado na norma IEEE802.11b (IEEE802.11b, 1999) a 2 Mbps.

Passemos agora à descrição de alguns detalhes mais relativos à especificação e implementação do RFieldbus.

2.3.1. Adaptação de Meios Físicos por Inserção de Tempos de Inactividade

No PROFIBUS, a resposta a um pedido tem de ser recebida dentro de um determinado intervalo de tempo – o *Slot Time* (T_{SL}), um parâmetro definido nas estações. Se este tempo for excedido (medido após o envio do pedido ou do *token*), o mestre repete o pedido (ou *token*), ou aborta a transacção. Numa rede híbrida como o RFieldbus, existem dois factores que afectam a duração das transacções: as latências adicionais devido à transferência de tramas nos ISs, e os atrasos potencialmente ilimitados nas filas dos ISs.

O problema é que, não é possível garantir, neste contexto de funcionamento, um limite superior para a duração de uma transacção. A necessidade de conhecer esse limite superior é crucial para o *setup* da rede, nomeadamente através da configuração do parâmetro T_{SL} do PROFIBUS.

Uma solução para este problema foi proposta em (Alves *et al.*, 2001a). Trata-se de um método vulgar de adaptação de meios físicos, mas com aspectos temporais complexos. Na prática, os nós mestre são configurados para inserirem um tempo morto (*idle time*) adicional entre transacções consecutivas, recorrendo, para o efeito, aos parâmetros *Idle Time* do PROFIBUS (T_{ID1} e T_{ID2}).

Obviamente, a inserção de tempos mortos adicionais reduz o número de transacções por unidade de tempo. No entanto, a vantagem de eliminar os atrasos crescentes nas filas dos ISs é significativa, pois leva a um tempo de reacção a erros de transmissão (que é função do T_{SL}) mais curto, e a transacções entre nós pertencentes a domínios diferentes com tempos de resposta muito inferiores.

2.3.2. Suporte de Mobilidade entre Células

O RFieldbus contempla um mecanismo relativamente simples, e com garantias temporais para a gestão de mobilidade de nós entre células rádio. A solução proposta em (Alves *et al.*, 2002; Alves, 2003) possibilita uma mudança de célula *wireless* (*handoff*) transparente para qualquer tipo de nó móvel (mestre, escravo ou LS).

Um mestre específico (o *mobility master*), por exemplo o M1 da Figura 2.2, despoleta o procedimento através da difusão de uma mensagem específica designada por *Beacon Trigger* (BT). Esta trama é propagada para todos os nós da rede, e com algum relevo para as BSs/LBSs. Estas começam então a enviar *beacons* no seu canal (para o exemplo da Figura 2.2, a LBS no CH1 e a BS no CH2), e todos os nós móveis (e.g. S6) devem avaliar a qualidade dos diferentes canais e comutar para o de melhor qualidade (para o CH2, assumindo que S6 se move para essa célula, tal como reflectido pelo percurso a tracejado na Figura 2.2. A Figura 2.4 ilustra este procedimento.

Dado que o caminho entre o *mobility master* e cada BS/LBS pode ser diferente, cada BS/LBS poderá receber a trama BT em diferentes instantes. Consequentemente, o número de *beacons* que cada BS/LBS tem de transmitir também poderá ser diferente. Esta situação acontece na topologia apresentada na Figura 2.2, e assume-se, no exemplo da Figura 2.4, que a LBS tem de transmitir 4 *beacons* e a BS tem de transmitir 3 *beacons*.

Para assegurar a compatibilidade deste mecanismo com o MAC do PROFIBUS, depois do *mobility master* despoletar o mecanismo de gestão de mobilidade, este deve inserir um tempo morto adequado (T_{ID2} , representado na Figura 2.4) antes de passar o *token*. De facto, a rede só poderá voltar ao seu funcionamento normal (troca de mensagens PROFIBUS) quando todos os nós móveis terminarem o seu procedimento de *handoff*. Por isso, o tempo morto inserido pelo *mobility master* (depois deste enviar a trama BT) corresponde à pior duração do procedimento de gestão de mobilidade nos diferentes nós móveis.

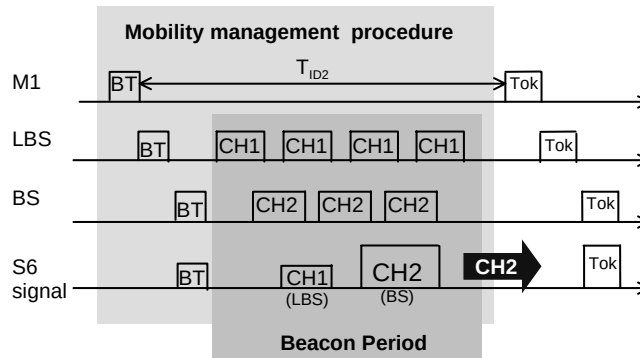


Figura 2.4: Diagrama temporal simplificado do mecanismo de gestão de mobilidade

A metodologia de cálculo de T_{ID2} bem como do número de *beacons* que têm de ser transmitidos por cada BS ou LBS no sistema não é abordada nesta dissertação. O leitor pode consultar (Alves, 2003) para mais detalhes.

2.4. Extensões para o Suporte Multimédia no PROFIBUS

Como foi referido no Capítulo 1, grande parte das aplicações multimédia executadas em rede são suportadas por protocolos TCP/IP. Este “requisito” é contemplado na solução RFieldbus, que considera uma dupla pilha de comunicações nas camadas superiores à camada de ligação de dados.

Arquitectura protocolar de pilha dupla é ilustrada na Figura 2.5, adaptada de (Ferreira *et al.*, 2001). No essencial, trata-se de uma arquitectura baseada no PROFIBUS DP. Por um lado, permite manter as características de tempo crítico para as aplicações “nativas” (tráfego PROFIBUS DP). Por outro lado, garante a qualidade de serviço necessária ao bom funcionamento das aplicações multimédia.

As camadas de aplicação do TCP/IP e do PROFIBUS DP coexistem. As camadas física e de ligação de dados são PROFIBUS/RFieldbus. Entre o IP e a DLL existem três sub-camadas com características específicas, e que sustentam a integração dos dois conjuntos: a *DP/IP Dispatcher*, a *Admission Control and Scheduling (ACS)* e a *IP-Mapper*. As funcionalidades e os mecanismos considerados nestas três sub-camadas vão agora ser sucintamente descritas.

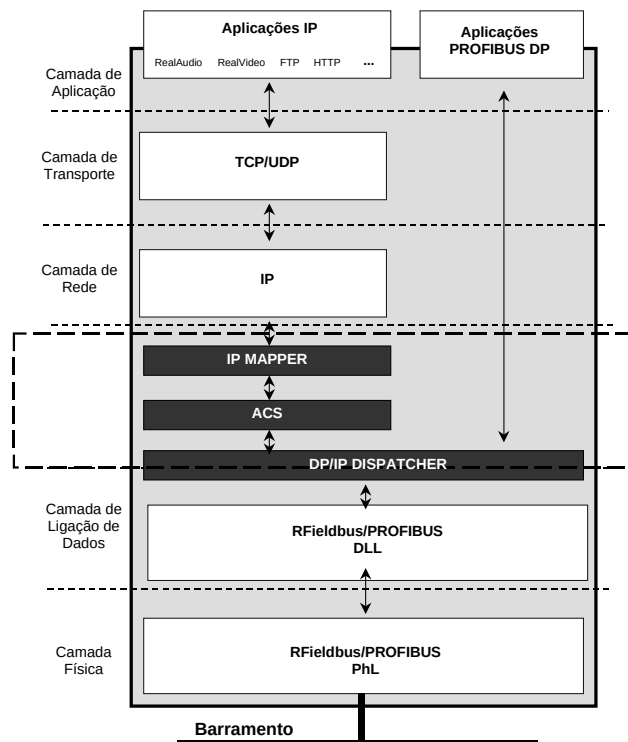


Figura 2.5: Integração de aplicações TCP/IP na arquitectura protocolar PROFIBUS/RFieldbus

2.4.1. A Sub-camada *IP/DP Dispatcher*

Como já foi dito atrás, no PROFIBUS, o número de transacções de alta prioridade (DPH) que podem ser executadas numa visita do *token* a um mestre depende do número de transacções efectuadas pelos mestres durante a rotação anterior do *token*. No limite, o *token* poderá chegar em atraso ao mestre ($T_{RR} > T_{TR}$), pelo que este só poderá executar uma única transacção de alta prioridade, tendo depois de passar o *token* ao seu sucessor.

Em (Tovar e Vasques, 1999b), os autores propõem duas abordagens distintas para a utilização de redes PROFIBUS em aplicações distribuídas de tempo real.

1. Uma abordagem na qual o tráfego de baixa prioridade não é restringido. Garantias temporais só são dadas ao tráfego de alta prioridade, assumindo o pior caso de uma só transacção de alta prioridade por visita do *token*.
2. Uma abordagem na qual o tráfego de baixa prioridade é restringido, pelo que numa visita do *token* é garantido que todo o tráfego de alta e baixa prioridade pendente nas filas da DLL é processado.

O princípio inerente à segunda das duas abordagens referidas é o utilizado no RFieldbus. Assim, e de acordo com os requisitos das aplicações, para o sistema é definido um T_{TR} mínimo, e para cada mestre é imposto um T_{TH} máximo, designado por T_{MA} (*Master Allocation*). Adicionalmente, em cada mestre são definidas as alocações de tempo para cada tipo de tráfego: T_{DPH} (para o tráfego DP de alta prioridade); T_{DPL} (para o tráfego DP de baixa prioridade); T_{IPH} (para os fragmentos IP relativos a aplicações que usam IP que exigem qualidade de serviço) e T_{BE} (para tráfego DP ou fragmentos IP sem exigências de qualidade de serviço). Esta alocação de tempo é ilustrada na Figura 2.6.

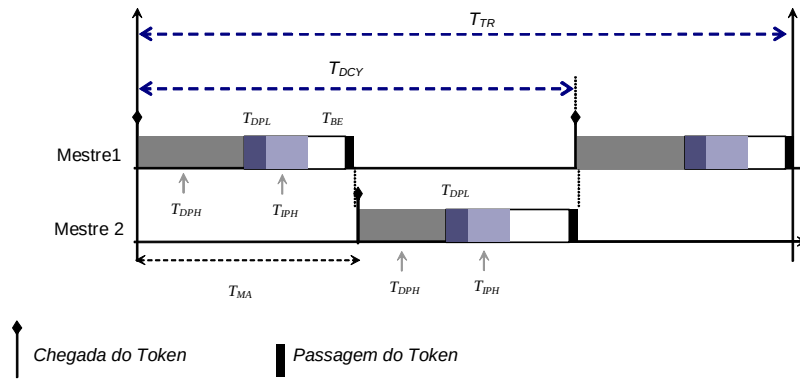


Figura 2.6: Alocações de tipos de tráfego numa rede RFieldbus com dois mestres

O *IP/DP Dispatcher* contém quatro filas do tipo *first-come-first-served* (FCFS) para pedidos relativos a cada um dos quatro tipos de tráfego. O *IP/DP Dispatcher* implementa um algoritmo de serviço a essas filas (Tovar *et al.*, 2001), que garante que o tempo de serviço a cada uma dessas filas não é superior à alocação de tempo prevista para cada tipo de tráfego em cada visita do *token*. Como é óbvio, o *IP/DP Dispatcher* tem de ter informação sobre a duração das transacções envolvidas. O algoritmo de despacho (servir as filas) é executado ciclicamente, com uma periodicidade que corresponde ao pior caso de rotação do *token* (pior T_{RR} admitido para a rede), e na Figura 2.6 representado por T_{DCY} (*dispatcher cycle*), também um parâmetro do *IP/DP Dispatcher*. O algoritmo prevê ainda, um T_{BE} dinâmico, isto é, o tempo das alocações para tráfego DPH, DPL e IPH não utilizado pode ser reutilizado para tráfego BE (*Best Effort*), quer seja DP, quer sejam fragmentos IP.

De notar a utilização do termo “fragmento IP”. Tipicamente os pacotes IP têm uma quantidade de bytes superior ao comprimento máximo do campo de dados de uma trama PROFIBUS, que é de 244 bytes. O tráfego IP é fragmentado pelo *IP-Mapper*. Estes fragmentos circulam na rede encapsulados em tramas PROFIBUS.

Por fim, de referir que o tráfego DPL, IPH e BE é mapeado nas fila de tráfego de baixa prioridade da DLL do PROFIBUS. Este aspecto não é muito relevante, uma vez que a existência do *IP/DP Dispatcher* torna irrelevante, do ponto de vista da qualidade de serviço, esta distinção original da DLL do PROFIBUS.

As metodologias e critérios mais detalhados para a obtenção dos parâmetros do *IP/DP Dispatcher* não são abordados neste capítulo. O leitor pode consultar (Tovar *et al.*, 2001) para mais detalhes.

2.4.2. A Sub-camada *Admission Control and Scheduling* (ACS)

A alocação prevista para o tráfego IPH (fragmentos IP para os quais é preciso dar garantias de qualidade de serviço), T_{IPH} , é partilhada por um conjunto de fluxos IP, que podem apresentar requisitos de qualidade de serviço (QoS) diferentes, nomeadamente a nível de largura de banda.

O ACS, implementa um escalonamento do tipo multi-ciclo, idêntico ao utilizado em outras redes de tempo real como é o caso do WorldFIP (EN 50170, 1996), onde é definida uma tabela de escalonamento, designada *Bus Arbitrator Table* (BAT) (Tovar e Vasques, 2000).

O conceito de multi-ciclo implica a definição de micro-ciclos e de macro-ciclos. No RFieldbus, o micro-ciclo corresponde à periodicidade (T_{DCY}) de execução do algoritmo do *IP/DP Dispatcher*, que, como já foi dito, corresponde ao maior tempo de rotação admissível para o *token*. Em cada micro-ciclo são escalonados fragmentos IP. O macro-ciclo define um conjunto de micro-ciclos que se repete continuamente.

De forma a dar uma melhor intuição, vamos ilustrar o conceito com um exemplo.

Imagine-se que para uma rede RFieldbus específica, $T_{DCY} = 10\text{ms}$ e $T_{IPH} = 2\text{ms}$ para um determinado mestre (ver Figura 2.6). (Como se obtêm estes parâmetros a partir dos requisitos das aplicações não será objecto de análise deste capítulo).

Admita-se agora que existem cinco fluxos (*streamed data*) de dados IP para os quais se pretende garantir qualidade de serviço. Cada um desses cinco fragmentos IP exige uma periodicidade e corresponde a transacções com durações como as indicados na Tabela 2.1. Note-se que as durações diferentes resultam de se tratarem de transacções com “percursos” entre nós que podem passar por um número diferente de ISs da rede RFieldbus. Admita-se ainda que em cada transacção são trocados 200 *bytes* (1600 *bits*) de informação multimédia. O número de *bits* por segundo (bps) que se consegue para cada um dos fluxos é o dado pela Equação (2.2), na qual *bpf* corresponde a bits úteis por fragmento.

$$\frac{1}{T_{IPH_i}} \times bpf = \frac{1}{k_i \times T_{DCY}} \times bpf = \frac{1}{k_i \times 0.01} \times 1600 = \frac{100}{k_i} \times 1600 \text{ bps} \quad (2.2)$$

Tabela 2.1: Exemplo e caracterização de 5 fluxos IPH

	Periodicidade requerida ($k_i = T_{IPH_i} / T_{DCY}$)	Duração da Transacção (μs)	Transferência de dados multimédia (bps)
IPH_1	1	100	$100 \times 1600 = 160000$
IPH_2	3	200	$33,3(3) \times 1600 \approx 26667$
IPH_3	3	200	$33,3(3) \times 1600 \approx 26667$
IPH_4	4	400	$25 \times 1600 = 40000$
IPH_5	4	1000	$25 \times 1600 = 40000$

O mínimo múltiplo comum das periodicidades é $12 \times T_{DCY} = 120\text{ ms}$, pelo que um macro-ciclo é composto por 12 micro-ciclos (Figura 2.7).

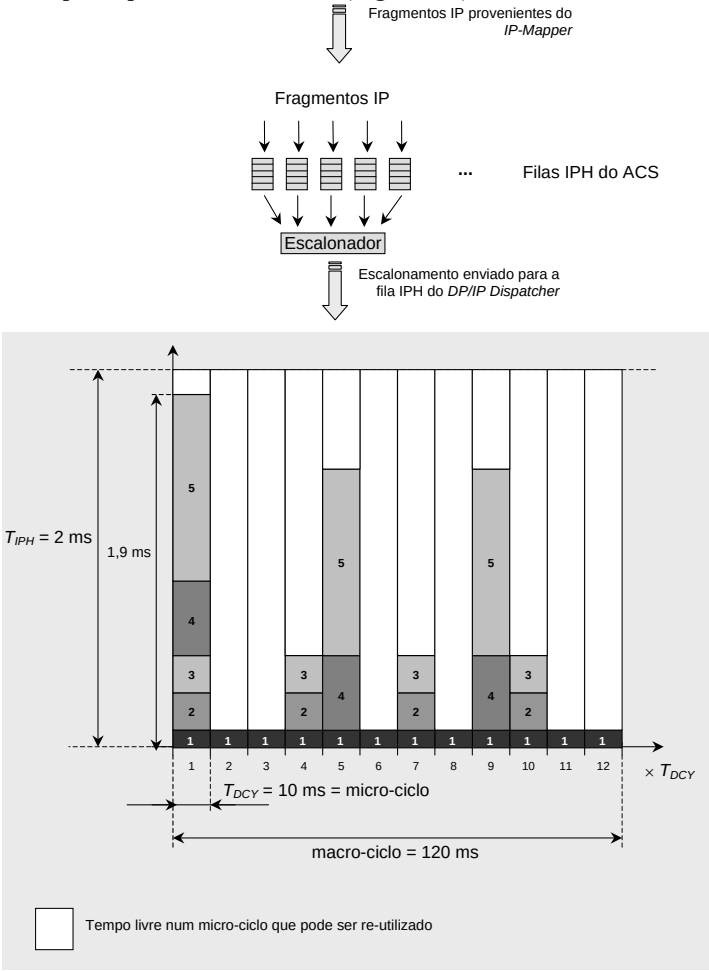


Figura 2.7: Exemplo de escalonamento ao nível do ACS

A Figura 2.7 ilustra um escalonamento possível para suportar os requisitos de largura de banda dos fluxos IPH constantes da Tabela 2.1. Note-se que os requisitos de *delay* e de *jitter* reduzido estão implicitamente cobertos dada a natureza do escalonamento (fragmentos IP escalonados de forma distribuída no tempo).

Obviamente que outras tabelas de escalonamento (a Tabela 2.2 corresponde ao escalonamento resultante na Figura 2.7) podem ser configuradas no mestre, obtendo sensivelmente o mesmo efeito. Em (Ferreira *et al.*, 2001) são descritos vários algoritmos para a construção da tabela.

Tabela 2.2: Tabela de Escalonamento para o exemplo

Micro-ciclo	1	2	3	4	5	6	7	8	9	10	11	12
IPH_1	1	1	1	1	1	1	1	1	1	1	1	1
IPH_2	1			1			1			1		
IPH_3	1			1			1			1		
IPH_4	1				1				1			
IPH_5	1				1				1			

Esses algoritmos são válidos para a obtenção, *on-line*, do escalonamento para cada micro-ciclo, ou para a construção de uma tabela de escalonamento fornecida ao mestre como parâmetro. Entre outros aspectos, os algoritmos descritos tentam otimizar a utilização do T_{IPH} . Por exemplo, uma distribuição mais equitativa dos fragmentos pelos micro-ciclos, permite, se o sistema exigir (por causa dos outros tipos de tráfego), um valor menor para o T_{IPH} , mantendo o T_{DCY} . Outro aspecto importante considerado nos algoritmos de escalonamento é o de efectuar compensações resultantes do facto (devido ao funcionamento do *IP-Mapper* e às próprias aplicações multimédia) de fragmentos não estarem disponíveis nas filas do ACS quando seria previsto passá-lo para o *IP/DP Dispatcher*. Estes detalhes, assim como outros associados ao ACS, não são descritos neste capítulo. Para mais detalhes, o leitor é remetido para (Ferreira *et al.*, 2001).

2.4.3. A Sub-camada *IP-Mapper*

O *IP-Mapper* é responsável pela conversão de pacotes IP em fragmentos que são encapsulados em unidades de protocolo da DLL do PROFIBUS, e vice-versa. É também responsável por suportar o fluxo de tráfego IP entre quaisquer duas estações da rede RFieldbus, independentemente de se tratarem de nós escravos ou nós mestre da rede PROFIBUS. Em termos de funcionalidades e arquitectura, trata-se de uma sub-camada que apresenta alguma complexidade, e cujos detalhes (Pacheco *et al.*, 2001) não são abordados nesta dissertação.

No essencial, o *IP-Mapper* contém cinco entidades (Figura 2.8): a entidade de fragmentação; a entidade de re-assemblagem; a entidade de identificação e encaminhamento; a entidade de comutação; e a entidade de geração de IDs.

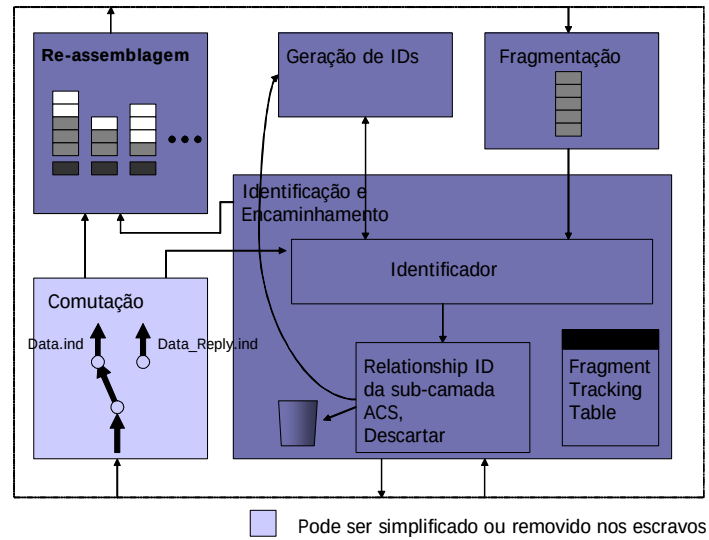


Figura 2.8: Funcionalidades do IP-Mapper

Capítulo 3

O Sistema de Fabrico Flexível e o Suporte Computacional Distribuído Associado

Neste capítulo será feita uma breve descrição do sistema de fabrico flexível (SFF) implementado no LASCRI no âmbito do projecto RFieldbus, incluindo a descrição dos seus aspectos funcionais e com algum relevo as aplicações distribuídas que recorrem ao TCP/IP para suporte multimédia.

3.1. Introdução

No âmbito do projecto Europeu RFieldbus, foram especificadas e implementadas duas aplicações industriais que visavam, no essencial, a validação da tecnologia RFieldbus, também desenvolvida no âmbito desse projecto.

Uma das aplicações industriais foi especificada, desenvolvida e implementada na LPC (empresa ligada ao fabrico de produtos derivados do petróleo), Grécia, e tinha por objectivo essencial a validação de aspectos relacionados com a qualidade da transmissão rádio. Por isso foi uma aplicação industrial mais relacionada com a validação dos aspectos da camada física da tecnologia RFieldbus.

A outra aplicação foi especificada, desenvolvida e implementada no laboratório de sistemas críticos do ISEP, o LASCRI. Ao contrário da aplicação levada a cabo na LPC, uma típica indústria de processos contínuos, a aplicação desenvolvida no ISEP (Alves *et al.*, 2001b; Tovar *et al.*, 2003) relaciona-se com o fabrico de componentes discretos, e teve por objectivo a validação dos aspectos da tecnologia relacionados com:

1. o cumprimento de prazos temporais em aplicações de controlo distribuído de tempo real;
2. o suporte da mobilidade de dispositivos entre células rádio;
3. o suporte a aplicações multimédia distribuídas com qualidade de serviço.

Neste capítulo será feita uma breve descrição do sistema de fabrico flexível implementado no LASCRI no âmbito do projecto RFieldbus, incluindo a descrição dos seus aspectos funcionais, e sendo dado algum relevo às aplicações distribuídas que recorrem ao TCP/IP para suporte multimédia. No contexto desta dissertação, o acrónimo TCP/IP é utilizado para referir genericamente o conjunto de protocolos de rede IP, incluindo o suporte dos serviços de transporte TCP e UDP (*User Datagram Protocol*). TCP e UDP, quando usados isoladamente, referem-se aos serviços da Camada de Transporte da referida pilha, em particular ao serviço com ligação e sem ligação (datagrama), respectivamente. O acrónimo IP, quando usado isoladamente, refere-se ao serviço base da Camada de Rede.

3.2. Descrição Genérica da Aplicação Industrial

O Sistema Flexível de Fabrico (SFF) especificado e implementado foi um sistema que se pode encontrar em diversas etapas de uma linha de produção de alguns tipos de indústria.

Pode-se admitir que este SFF é uma etapa (um subsistema) de uma linha de produção de um sistema industrial mais vasto. Cada peça que chega ao subsistema deve ser classificada de acordo com critérios específicos (por exemplo, cor, formato, etc.) e posteriormente distribuída por *buffers* de armazenamento intermédio que fazem a interface com outros subsistemas a jusante deste. Um exemplo de um subsistema a montante poderia ser um sistema que realizasse uma operação de maquinação (por exemplo, perfuração) num dos tipos de peças.

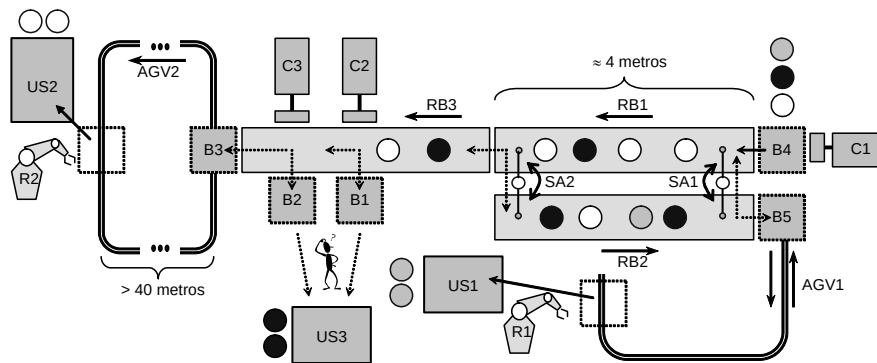


Figura 3.1: Diagrama esquemático do sistema de fabrico flexível

O SFF implementado está ilustrado na Figura 3.1, e a descrição que se apresenta a seguir reporta a essa figura. Do ponto de vista mecânico, o sistema é constituído por:

- 5 *buffers* de armazenamento intermédio: B1, B2, B3, B4, B5;
- 3 tapetes rolantes (*roller belts*): RB1, RB2, RB3;
- 3 cilindros pneumáticos: C1, C2, C3;
- 2 braços rotativos com cilindros pneumáticos (*swivel arms*): SA1, SA2;
- 2 veículos de transporte automáticos (*automatic guided vehicles*): AGV1 (autónomo), AGV2 (guiado por linha).

Estes dois veículos automáticos realizam operações de transporte entre estações que podem representar dois subsistemas de fabrico a jusante. Estão por isso representadas também na figura:

- 3 estações de descarga (*Unload Stations*): US1, US2 e US3;
- 2 robôs de manipulação associados às estações: R1 e R2.

No subsistema, e nas suas interfaces, circulam 3 tipos de peças: B (preto), W (banco), G (cinza – defeituosas).

B4 é um *buffer* de entrada no subsistema. Os outros quatro são *buffers* de saída. A título de exemplo, B3 é um *buffer* de interface com um subsistema a jusante (*Unload Station 2*).

A chegada de peças é simulada, por razões práticas, usando um *buffer* cilíndrico de armazenamento vertical (B4). Também por razões de realização prática do sistema, as peças têm um formato circular. Um cilindro pneumático (C1) alimenta o subsistema, peça a peça (por gravidade em B4), com uma cadência programável. As peças vão sendo colocadas no tapete rolante RB1. Também por razões práticas, os tipos distinguem-se pela cor: preta, cinzenta ou branca. As peças cinzentas têm como destino final o subsistema a jusante ilustrado na Figura 3.1 por *Unload Station 1* (US1). As peças pretas têm como destino final o subsistema *Unload Station 2* (US2). Finalmente, as peças brancas têm como destino final o subsistema representado na figura como *Unload Station 3* (US3). Para efeitos de classificação, são utilizadas duas câmaras de vídeo colocadas por cima dos tapetes rolantes RB1 e RB2. De acordo com o resultado da informação delas provenientes, os braços pneumáticos SA1 e SA2 fazem a transferência entre os tapetes rolantes RB2 e RB1 ou entre RB1 e RB2, respectivamente. As peças pretas e brancas que seguem para o tapete RB3 irão para B1/B1 (as pretas) ou B3 (as brancas), por actuação (ou não) dos cilindros pneumáticos C2 e C3.

Os *buffers* B1, B1, B3 e B5 têm capacidade programável. Quando atingem a sua capacidade limite, B3 e B5 são transferidos para os subsistemas a jusante por dois carros de transporte automáticos, AGV2 e AGV1, respectivamente. B1 e B2 são descarregados manualmente por intervenção de um operador. Enquanto o *buffer* B4 não está disponível, ou porque atingiu o limite da sua capacidade, ou porque está momentaneamente ausente, as peças brancas deverão permanecer em circulação em RB1/RB2. O mesmo se passa relativamente ao conjunto dos dois *buffers* B1 e B2 para as peças pretas, e para *buffer* B5 para as peças cinzentas. A Figura 3.2 ilustra, de forma parcial, o SFF real implementado no LASCRI do ISEP.



Figura 3.2: Vista parcial do SFF tal como está no Laboratório de Sistema Computacionais Críticos do ISEP (LASCRI), <http://www.hurray.isep.ipp.pt/rfpilot>

3.3. A Infra-estrutura de Comunicações e Computacional

O SFF descrito na secção anterior é controlado por um sistema computacional de controlo distribuído, ilustrado na Figura 3.3.

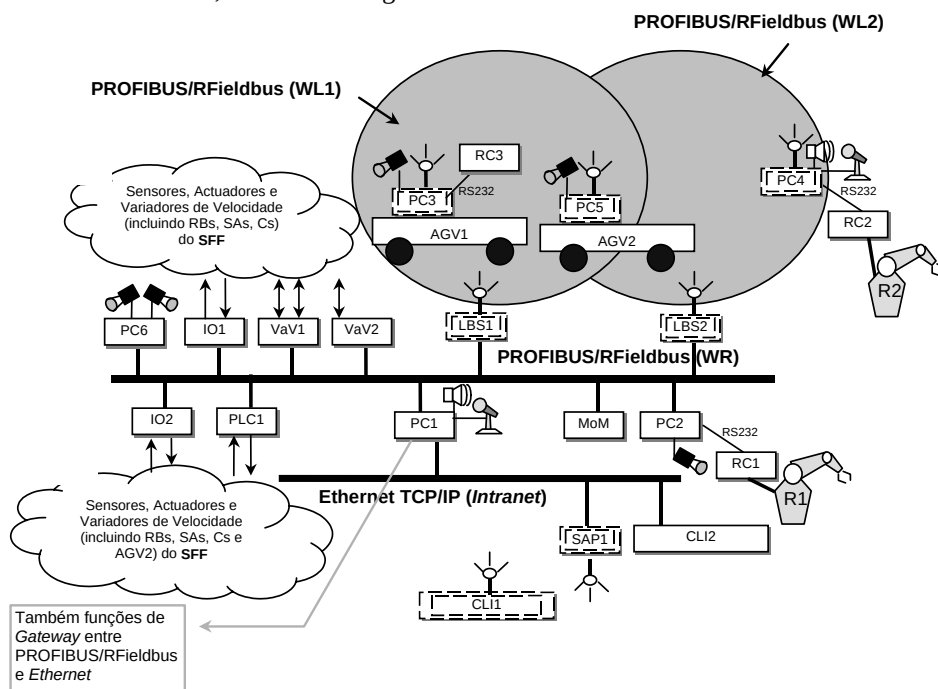


Figura 3.3: Diagrama esquemático da infra-estrutura de comunicações e computacional

A distribuição é suportada por uma rede de comunicações industrial PROFIBUS/RFieldbus e uma *Gateway* (implementada no PC1) que permite a interligação entre a rede PROFIBUS/RFieldbus e uma *Intranet* baseada em *Ethernet*. Na rede *Ethernet*, existe um *Wireless Access Point* (WAP), um Compaq WL410 (Compaq, 2001), pelo que do lado da *Intranet* é possível também a existência de nós móveis sem fios compatíveis com a norma IEEE802.11b, por exemplo PDAs.

Vamos agora descrever mais alguns detalhes do sistema de comunicações e computacional ilustrado na Figura 3.3. No essencial, este sistema computacional distribuído configura dois tipos essenciais de fluxos de informação:

1. fluxos de informação horizontal;
2. e fluxos de informação vertical.

3.3.1. Fluxos de Informação Horizontal

Os fluxos de informação horizontal correspondem às comunicações de dados relacionadas com as operações mais críticas de controlo do SFF. Estas comunicações são implementadas ao nível da rede PROFIBUS/RFieldbus, e correspondem tipicamente às

transacções cíclicas PROFIBUS DP entre o mestre PROFIBUS/RFieldbus (PC1) e os dez nós escravo PROFIBUS/RFieldbus descritos na Tabela 3.1.

Tabela 3.1: Características dos dispositivos ligados à rede PROFIBUS/RFieldbus

Designação	Endereço PROFIBUS	Endereço IP	Suporte Computacional	Comunicações	Utilização de Extensões RFieldbus	
					Wireless	Multimédia
Mestres						
PC1	1	10.0.0.1	PC/WindowsNT	Carta PROFIBUS - IFAK_isPRO ISA	–	√
MoM	8	–	PC/WindowsNT	Carta PROFIBUS - IFAK_isPRO ISA	√	–
Escravos						
PLC1	65	–	PLC S7200	PROFIBUS DP Embebido	–	–
VaV1	10	–	MicroMaster MM420	PROFIBUS DP Embebido	–	–
VaV2	11	–	MicroMaster MM420	PROFIBUS DP Embebido	–	–
IO1	64	–	Concentrador ET200	PROFIBUS DP Embebido	–	–
IO2	66	–	Concentrador ET200	PROFIBUS DP Embebido	–	–
PC2	2	10.0.0.2	PC/WindowsNT	Carta PROFIBUS - IFAK_isPRO ISA	–	√
PC3	3	10.0.0.3	PC/WindowsNT	Carta PROFIBUS - IFAK_isPRO PCMCIA	√	√
PC4	4	10.0.0.4	PC/WindowsNT	Carta PROFIBUS - IFAK_isPRO ISA	√	√
PC5	5	10.0.0.5	PC/WindowsNT	Carta PROFIBUS - IFAK_isPRO PCMCIA	√	√
PC6	6	10.0.0.6	PC/WindowsNT	Carta PROFIBUS - IFAK_isPRO ISA	–	√
Interligação						
LBS1	–	–	RFIL/LBS	–	√	–
LBS2	–	–	RFIL/LBS	–	√	–

Note-se, no entanto, que existe também um conjunto de fluxos horizontais, relacionados com o controlo do SFF, que recorrem a encapsulamento de dados TCP/IP em transacções PROFIBUS DP. Essas transacções PROFIBUS DP são transferidas nas janelas IP do tempo de posse de *token* do mestre PROFIBUS associado ao PC1,

conforme o escalonamento previsto para o *Dispatcher RFieldbus* (alguns detalhes foram dados na Secção 2.4.1 desta dissertação). São disso exemplos o fluxo de voz bidireccional existente entre o PC1 e o PC4, ou os fluxos de imagens (entre o PC1 e o PC6) para a classificação das peças.

No segmento com fios da rede PROFIBUS/RFieldbus (segmento *wired* – WR) estão ligados os seguintes elementos:

- 2 nós PROFIBUS/RFieldbus do tipo mestre (*master*): PC1 e MoM;
- 2 nós PROFIBUS/RFieldbus do tipo escravo (*slave*): PC2² e PC6;
- 5 nós PROFIBUS do tipo escravo (*slave*): IO1, IO2, VaV1, VaV2 e PLC1;
- 2 dispositivos de interligação híbridos (*wired/wireless*) estruturantes (*link base stations*): LBS1 e LBS2.

Estes dois últimos dispositivos de interligação híbridos estruturam duas células de rádio PROFIBUS/RFieldbus independentes: a célula de rádio WL1 e a célula de rádio WL2. Associados a estas duas células de rádio, existem:

- 3 nós PROFIBUS/RFieldbus do tipo escravo (*slave*): PC3, PC4 e PC5.

O PC4 é um nó sem fios (*wireless*) PROFIBUS/RFieldbus sem mobilidade, que estará sempre associado à célula rádio WL2, definida por LSB2. Ao contrário, PC3 e PC5 são dois nós sem fios PROFIBUS/RFieldbus com mobilidade. No entanto, PC3 é um nó cuja área de movimentação faz com que esteja sempre associado à célula WL1 (definida por LSB1). Ao contrário, PC5 tem uma área de movimentação que faz com que possa estar em determinadas alturas associado à célula WL1, e em outras alturas associado à célula WL2, de acordo com os mecanismos de suporte à mobilidade entre células implementados pela tecnologia RFieldbus, e já descritos sucintamente na Secção 2.3.2 desta dissertação.

Relativamente aos dois nós mestre, é de referir que o mestre MoM se limita a implementar, periodicamente, o mecanismo de gestão de mobilidade entre células rádio (Secção 2.3.2).

A Tabela 3.1 resume algumas das características dos dispositivos ligados à rede PROFIBUS/RFieldbus.

3.3.2. Fluxos de Informação Vertical

O fluxo de informação vertical corresponde a comunicações (*Ethernet* TCP/IP) entre o PC1 e os nós associados à *Intranet*. Como já foi referido anteriormente, o PC1 implementa funcionalidades de *gateway* entre os sistemas computacionais ligados à rede PROFIBUS/RFieldbus e os sistemas computacionais ligados à *Intranet*.

Do lado da *Intranet*, consideram-se os seguintes dispositivos (as aplicações por eles suportadas são representativas de outros que poderiam coexistir em simultâneo):

- 1 nó correspondente ao *proxy* PROFIBUS/RFieldbus: PC1;
- 1 nó correspondente a um cliente *wired*: CLI2;
- 1 ponto de acesso (*service access point*) IEEE802.1: SAP1.

Este dispositivo de interligação híbrido estrutura uma célula de rádio IEEE802.11b, na qual se considera:

² Na realidade o PC2 é uma plataforma *WindowsNT* com duas cartas de comunicação PROFIBUS/RFieldbus, sendo que a segunda carta corresponde ao mestre MoM.

- 1 nó correspondente a um *Personal Digital Assistant* (PDA): CLI1.
- A Tabela 3.2 resume algumas das características dos dispositivos ligados à *Intranet*.

Tabela 3.2: Características dos dispositivos ligados à *Intranet*

<i>Designação</i>	<i>Endereço MAC</i>	<i>Endereço IP</i>	<i>Suporte Computacional</i>	<i>Comunicações</i>
PC1	00-05-5D-00-F6-01	10.0.8.1	PC/WindowsNT	Carta D-Link DFE-530TX PCI Fast Ethernet Adapter
SAP1	00-02-2D-15-58-8E	10.0.8.100	Compaq WL410	–
CLI1	00-02-A5-6F-1E-7A	10.0.8.99	PDA iPAQ 3870/Windows PocketPC 2002	Carta PCMCIA Compaq WL110 11Mbps Wireless LAN
CLI2	00-B0-D0-15-6A-1E	10.0.8.98	PC/Windows NT	Carta 3Com Etherlink 10/100 PCI

3.4. Detalhes dos Fluxos de Informação Horizontal

O componente computacional mais importante de todo o sistema é o PC1. O PC1 é um computador pessoal *WindowsNT* com duas cartas de comunicação: uma carta PROFIBUS DP a funcionar como mestre, e uma carta *Ethernet* que permite a interoperabilidade entre o sistema computacional suportado pela rede PROFIBUS/RFieldbus e os dispositivos computacionais associados à *Intranet* (ver Figura 3.3).

Nesta unidade computacional residem várias aplicações, das quais se destacam, pela sua importância, a aplicação de supervisão, controlo e aquisição de dados (*Supervision Control and Data Acquisition*) – **RF-SCADA**.

A aplicação RF-SCADA é na realidade composta por dois módulos cooperantes: um módulo de interface com o operador (módulo **RF-HMI**), e um módulo que implementa a máquina de estados relacionada com o controlo do SFF (o módulo **RF-CON**). Estes dois módulos, por terem requisitos computacionais muito distintos, foram concebidos como dois componentes independentes, que interagem via mecanismos de comunicação entre processos oferecido pelo sistema operativo (as *Windows Messages* da API – detalhes no Capítulo 4). A implementação do módulo RF-CON foi concretizada em C++ (Pereira e Pacheco, 2002), tendo em conta, principalmente, os seus requisitos temporais e a necessidade de interagir com a carta PROFIBUS DP instalada no computador, que apresenta um interface de programação compatível. Por outro lado, o módulo RF-HMI (Pereira e Pacheco, 2002) foi desenvolvido em *VisualBasic 6.0*, dada a facilidade proporcionada por este ambiente no desenvolvimento da aplicação em causa.

Nas restantes subsecções desta secção iremos descrever em detalhe os fluxos de informação existentes no SFF no âmbito da rede PROFIBUS/RFieldbus:

1. tráfego cíclico DP na rede PROFIBUS/RFieldbus (DPH);
2. tráfego não cíclico DP na rede PROFIBUS/RFieldbus (DPL);
3. tráfego IP encapsulado em tráfego DP, com garantias de qualidade de serviço, na rede PROFIBUS/RFieldbus (IPH);
4. tráfego IP encapsulado em tráfego DP, sem garantias de qualidade de serviço, na rede PROFIBUS/RFieldbus (IPBE).

Como se verá, do ponto de vista operacional, o relevo é dado ao tráfego DPH e IPH.

3.4.1. Tráfego Cíclico DP (DPH)

Antes de dar mais alguns detalhes sobre estes dois módulos, e em particular no que se refere ao módulo RF-CON, é conveniente falar dos objectos distribuídos PROFIBUS DP que são manipuláveis pelo PC1 via carta de comunicações PROFIBUS/RFieldbus.

Num mestre PROFIBUS/RFieldbus, o tempo de posse de *token* é distribuído de acordo com uma parameterização feita *à priori* (Tovar *et al.*, 2003). A Figura 3.4 ilustra um cenário que considera os mestres PC1 e MoM. Considera ainda que, no período de tempo ilustrado, na visita do *token* ao mestre MoM é despoletado o mecanismo de suporte à mobilidade de nós entre células rádio (Alves *et al.*, 2002). Mas, o mais importante a destacar na figura é o facto de que em cada visita do *token* ao PC1, este executar uma transacção com cada um dos 10 escravos.

De facto, no PC1 existem objectos PROFIBUS DP associados a cada escravo que são manipulados (leitura/escrita) pelo RF-SCADA (especificamente pelo módulo RF-CON). Em cada transacção cíclica com cada escravo, objectos de leitura do RF-CON são actualizados com informação vinda do escravo e objectos escritos pelo RF-CON são actualizados remotamente no escravo.

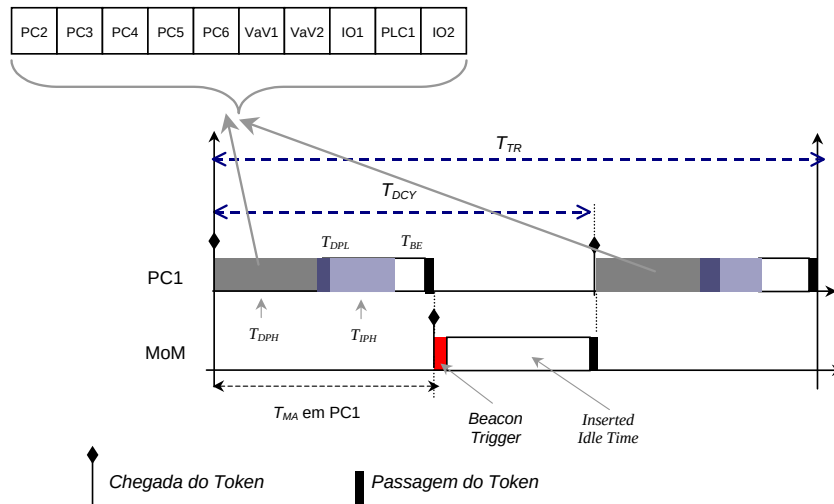


Figura 3.4: Ilustração da sequência de passagem de *token*

A Tabela 3.3 descreve, sucintamente, as funcionalidades suportadas pelo tráfego cíclico DP, bem como a relação com outras unidades de computação. Algumas considerações adicionais são apresentadas a seguir.

Por razões que se prendem com a criticalidade do seu funcionamento (nomeadamente exigências temporais), todos os sensores e actuadores relacionados com os braços pneumáticos (SA1 e SA2) são controlados e coordenados por um PLC dedicado, o nó escravo PROFIBUS/RFieldbus PLC1. Este PLC é um SIMATIC S7-200 (Siemens, 1998) com um módulo de comunicação PROFIBUS DP. Foram desenvolvidos diversos programas em lógica *Ladder* para o controlo dos braços pneumáticos (Pacheco, 2002a).

Os tapetes rolantes são controlados pelos variadores de velocidade VaV1 (RB1 e RB2 em paralelo), VaV2 (RB3). Estes variadores de velocidade são dois MicroMaster 420 (Siemens, 2001) com módulo de comunicação PROFIBUS DP.

Tabela 3.3: Funcionalidades suportadas pelo tráfego cíclico PROFIBUS/RFieldbus DP

<i>Escravo</i>	<i>O que controla</i>	<i>O que PC1 manipula</i>	<i>Ligação a elementos computacionais</i>	<i>Aplicações residentes</i>	<i>Manipulação Directa de IO</i>
PLC1	SA1, SA2	Início/fim de programas específicos residentes no PLC	–	√	–
VaV1	RB1, RB2	Velocidade, parar e arrancar	–	√	–
VaV2	RB3	Velocidade, parar e arrancar	–	√	–
IO1	Cilindros e Sensores	Cilindros, sensores de detecção de presença de peças	–	–	√
IO2	Cilindros, Sensores e AGV2	Cilindros, sensores de detecção de presença de peças e manipulação do I/O de uma estação do AGV2	–	–	√
PC2	R1	Início/fim de programas específicos residentes em RC1	a RC1 via porta série RS232	√ (em RC1)	–
PC3	AGV1	Início/fim de programas específicos residentes em RC3	a RC3 via porta série RS232	√ (em RC3)	–
PC4	R2 e AGV2	Início/fim de programas específicos residentes em RC2 e manipulação do I/O de uma outra estação do AGV2 (via RC2)	a RC2 via porta série RS232	√ (em RC2)	√ (via RC2)
PC5	–	–	–	–	–
PC6	Som	Aplicação de sons de alarme	–	√	–

Os robôs R1 e R2 são dois robôs da Eshed Robotec do tipo SCORBOT ER-VII (Eshed Robotec, 1991). As unidades de controlo desses robôs comunicam com os respectivos PCs (PC2 com RC1 e PC4 com RC2) via porta série (interface RS-232). Essencialmente, existem programas específicos armazenados nos respectivos controladores desenvolvidos na linguagem proprietária ACL (*Advanced Control Language*) (Eshed Robotec, 1994; Marques, 2002a). Note-se que o PC4 tem associado objectos PROFIBUS/RFieldbus que permitem não só que o PC1 despolete e monitorize

a execução dos programas específicos do manipulador robótico R2 (residentes em RC2), como também controlar, via RC2, o estado de uma das estações de paragem do AGV2 (ver (Alves *et al.*, 1998) para mais detalhes sobre esta tecnologia). Da mesma forma, IO2 é não só utilizado na manipulação do estado de cilindros pneumáticos, como também para controlar o estado de uma outra das estações de paragem do AGV2.

Quer IO1 quer IO2 correspondem a dispositivos SIMATIC ET 200M (Siemens, 2000) com um módulo de comunicação PROFIBUS DP.

À semelhança do que acontece com os robôs R1 e R2, o AGV1 tem também associado uma unidade de computação programável (designada na Figura 3.3 por RC3). O AGV1 é uma unidade robótica móvel autónoma do tipo Robuter II (Robosoft, 1997) com programas dedicados específicos desenvolvidos para essa plataforma (Pereira, 2002).

Não existe nenhuma funcionalidade implementada em PC5 (dada a especificidade do AGV2) mapeada em tráfego cíclico PROFIBUS/RFieldbus DP.

De referir ainda o PC6. Existe um objecto DP manipulado em PC6 que diz respeito à escolha de sons de aviso que soam em PC6 para sinalizar em áudio eventos particulares.

Por fim, e antes de passar à descrição dos outros tipos de fluxo de informação horizontais, cabe voltar outra vez à aplicação RF-SCADA, e em particular às suas interfaces com o operador.

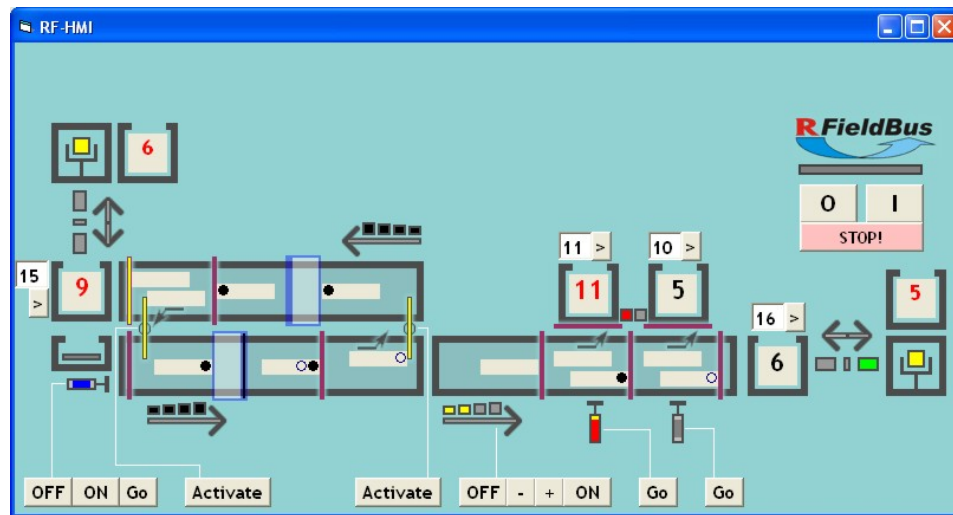


Figura 3.5: A interface do operador com a aplicação RF-SCADA (módulo RF-HMI)

A Figura 3.5 ilustra a interface pessoa-máquina principal (correspondente ao módulo RF-HMI). Note-se que é possível ao operador manipular o tamanho permitido para os *buffers*, manipular a velocidade dos tapetes, pedir a execução de operações específicas (actuar cilindros, braços pneumáticos, etc.). Em geral, estas operações são feitas antes do funcionamento normal do sistema, pelo que em *run-time* esta interface é de monitorização e supervisão. Durante o funcionamento do sistema, a sinóptica permite visualizar qual o estado de todos os sensores (incluindo câmaras de vídeo), estado dos *buffers*, estado dos braços de transferência, estado dos AGVs e o estado dos Robôs.

Também é visível a velocidade actual dos tapetes, assim como actuações automáticas nos cilindros pneumáticos. Em situações de erro (por exemplo, erro num braço de transferência) a área afectada é assinalada no sinóptico. O sistema também permite dois modos de paragem: a paragem normal em que é suspensa a entrada de novas peças e o sistema continua em funcionamento até todas as peças terem sido colocadas em *buffers* de saída; e a paragem de emergência que pára os tapetes e todos os actuadores.

A aplicação RF-CON tem também interfaces com o operador que permitem visualizar os valores dos objectos DP trocados com os 10 escravos (Figura 3.6), bem como configurar alguns dos parâmetros relacionados com o RFieldbus (Figura 3.7), e mencionados do Capítulo 2 desta dissertação.

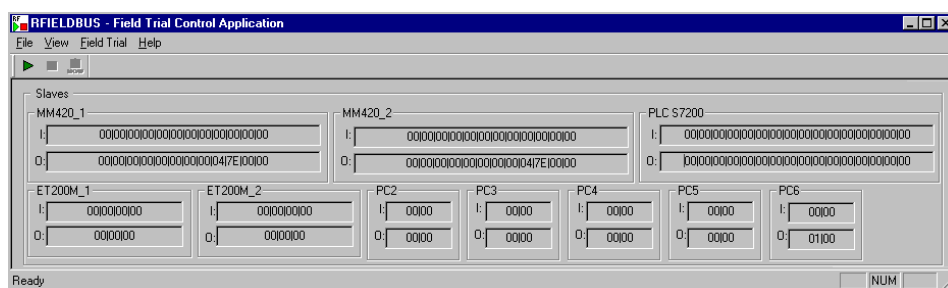


Figura 3.6: A interface do operador com a aplicação RF-SCADA (módulo RF-CON), janela principal

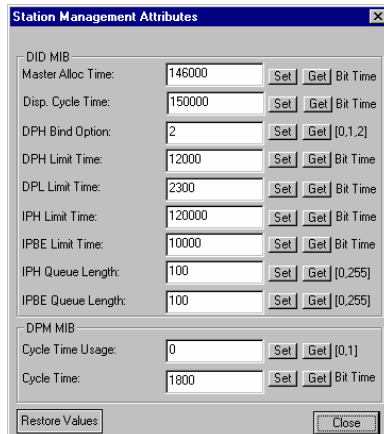


Figura 3.7: A interface do operador com a aplicação RF-SCADA (módulo RF-CON), janela de configuração de parâmetros do mestre

3.4.2. Tráfego Não Cíclico DP (DPL)

Existe tráfego não cíclico DP na rede PROFIBUS/RFieldbus. No entanto este não é relevante do ponto de vista das funcionalidades do SFF, tendo mais a ver com os

mecanismos de gestão do anel lógico (*logical ring maintenance*) e de gestão dos nós da rede (*live list* e *DP Status*). Na configuração do sistema (Figura 3.7) é definida uma alocação de tempo máxima (T_{DPL} da Figura 3.4) para o *Dispatcher* RFieldbus, que permita ao mestre (neste caso associado ao PC1) processar um número adequado de transacções deste tipo.

3.4.3. Tráfego TCP/IP Encapsulado em DP (IPH & IPBE)

A nível de desenvolvimento tecnológico, um dos objectivos do projecto RFieldbus era incorporar os mecanismos adequados ao suporte de aplicações multimédia numa rede de campo (*fieldbus*) normalizada, particularmente no PROFIBUS.

A solução passou pela especificação e implementação de uma arquitectura que considerou uma pilha protocolar dupla e dos mecanismos adequados a veicular pacotes IP encapsulado em tramas PROFIBUS DP (ver Secção 2.4 desta dissertação).

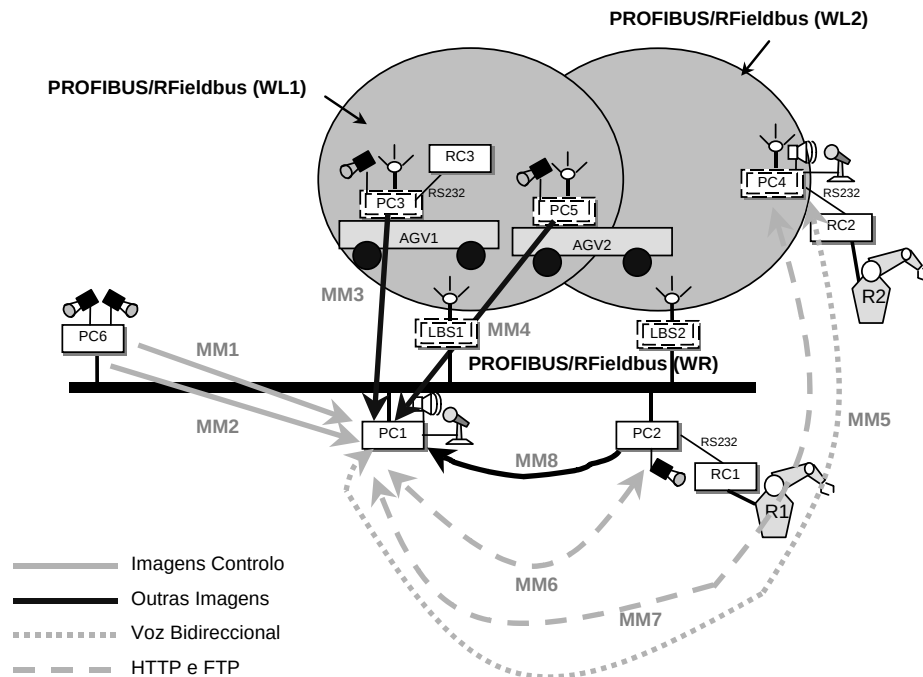


Figura 3.8: Diagrama esquemático da infra-estrutura de comunicações e computacional, com particular relevo para os 8 fluxos de tráfego IP (MM# – multimédia) encapsulado em tramas DP

Esta possibilidade de suportar simultaneamente o tráfego PROFIBUS DP “nativo” e tráfego IP encapsulado foi explorada no piloto de demonstração do ISEP, através de várias aplicações, incluindo algumas ligadas ao controlo do SFF descrito já descrito atrás. Nesta secção será feita uma descrição, mais ou menos sucinta, dessas funcionalidades.

A Figura 3.8 destaca os oito fluxos de tráfego TCP/IP encapsulado que suportam funcionalidades distribuídas ao nível do SFF.

Em primeiro lugar, convém lembrar que o *dispatcher* de um mestre PROFIBUS/RFieldbus considera a existência de dois tipos de tráfego TCP/IP encapsulado em tramas DP:

1. tráfego TCP/IP encapsulado com garantias de qualidade de serviço (IPH);
2. tráfego TCP/IP encapsulado sem garantias de qualidade de serviço (IPBE), em que BE significa *best effort*.

Essencialmente, e no que diz respeito ao tráfego IPH, pretende-se garantir qualidade de serviço, quer a nível de largura de banda mínima, quer a nível de *jitter* (variabilidade temporal na entrega de pacotes IP). Estas garantias permitem implementar fluxos de tráfego multimédia entre nós PROFIBUS/RFieldbus, mesmo para os relacionados com partes críticas do controlo do SFF. É disso exemplo o fluxo de imagens (MM1 e MM2 na Figura 3.8) das peças captadas pelo nó PC6, enviadas para o nó PC1 e utilizadas por este para actuação adequada dos braços pneumáticos (SA1 e SA2) em função da classificação obtida (no caso vertente a cor das peças). É também disso exemplo o fluxo de voz bidireccional (MM5 na Figura 3.8) entre a estação de descarga “remota” US2 (associada ao PC 4) e a estação de controlo (PC1), permitindo assim a dois operadores manterem uma comunicação de voz bidireccional utilizando para o efeito a infraestrutura de comunicações PROFIBUS/RFieldbus.

Já em relação ao tráfego IPBE, as suas garantias de entrega são condicionadas pelo tempo disponível em cada ciclo do *dispatcher* do mestre PROFIBUS/RFieldbus, pelo que implementam fluxos de informação horizontais não críticos ao (bom) funcionamento do SFF. São disso exemplo os fluxos MM6 e MM7 ilustrados na Figura 3.8, que permitem ao PC1 fazer, por exemplo, descarga de programas via FTP para os controladores robóticos RC1 e RC2.

A Tabela 3.4 sintetiza as funcionalidades, os protocolos de transporte (TCP ou UDP) e classes de serviço RFieldbus (IPH ou IPBE) utilizadas na implementação dos fluxos de tráfego TCP/IP encapsulado em DP.

Cabe agora referir sucintamente alguns aspectos relativos à implementação das funcionalidades previstas serem suportadas por estes fluxos de informação.

Os fluxos designados por MM1 e MM2 dizem respeito à classificação remota de peças em PC1. Existem duas câmaras *Winnov VideumCam Traveller* (Winnov, 2001) de captura de imagem com interface digital, associadas a uma placa de aquisição *Winnov* instalada no PC6. As câmaras estão protegidas por uma caixa opaca com iluminação própria, sendo por isso proporcionado um ambiente mais ou menos controlado de captura de imagem. As câmaras estão posicionadas por forma a captarem secções adequadas dos tapetes rolantes RB1 e RB2. As capturas de imagem são feitas a uma cadência *quasi*-contínua. Sempre que for detectada uma peça, essa imagem é enviada para PC1. Para reduzir as necessidades de largura de banda (e dado o tipo de classificação que é feita em PC1), a imagem é diminuída em termos de resolução (*down-sample* para 160×120 *pixels*) e comprimida num formato JPEG, antes de ser enviada via TCP/IP encapsulado para PC1. Daí resultam imagens que têm um tamanho de 2-3KBytes. Este requisito, conjugado com o da cadência máxima de entrada de peças no sistema (uma em cada 3 segundos), permite uma parameterização adequada da tabela de escalonamento do tráfego IPH no mestre PROFIBUS/RFieldbus PC1 (ver alguns detalhes na Secção 2.4.2). Para sincronização com a aplicação de controlo, sempre que

uma peça deixa de ser detectada é enviada a PC1 uma mensagem de texto com um tamanho de 4 bytes. Foram desenvolvidas duas aplicações específicas, uma para o lado do “cliente” (PC1) e outra para o lado do “servidor”, cujas interfaces se lustram na Figura 3.9, para o suporte desta funcionalidade (Marques, 2002b).

Tabela 3.4: Funcionalidades, protocolos de transporte e classes de serviço RFieldbus utilizadas nos fluxos de tráfego TCP/IP encapsulado em DP

<i>Fluxo</i>	<i>Nós</i>	<i>Descrição</i>	<i>Protocolo de Transporte</i>	<i>Classe de Serviço RFieldbus</i>
MM1	PC6 ↔ PC1	Classificação remota (em PC1) das peças que circulam to tapete rolante RB1	TCP	IPH
MM2	PC6 ↔ PC1	Classificação remota (em PC1) das peças que circulam to tapete rolante RB2	TCP	IPH
MM3	PC3 ↔ PC1	Imagens vídeo captadas pela câmara a bordo do AGV1	UDP	IPH
MM4	PC5 ↔ PC1	Imagens vídeo captadas pela câmara a bordo do AGV2	UDP	IPH
MM5	PC4 ↔ PC1	Fluxo de voz bidireccional	TCP/UDP	IPH
MM6	PC2 ↔ PC1	Manipulação remota de programas (em RC1) do Controlador Robótico de R1	TCP	IPBE
MM7	PC4 ↔ PC1	Manipulação remota de programas (em RC2) do Controlador Robótico de R2	TCP	IPBE
MM8	PC2 ↔ PC1	Detecção de posição do cesto de descarga	TCP	IPH

Os fluxos designados por MM3 e MM4 são utilizados para monitorizar, em PC1, o campo de visão frontal de AGV1 e AGV2, respectivamente. A cada segundo é enviada uma imagem captada por câmaras *Winnov Traveller*, associadas a uma carta de interface PCMCIA instaladas quer em PC3 quer em PC5. À semelhança do que acontece com MM1 e MM2, as imagens são comprimidas para um formato JPEG (resultado de 2-3 Kbytes), antes de enviadas via UDP encapsulado em PROFIBUS DP para o PC1. De todo o modo, é garantida qualidade de serviço adequada aos requisitos temporais mencionados, mapeando também este fluxo na classe de serviço IPH do RFieldbus.

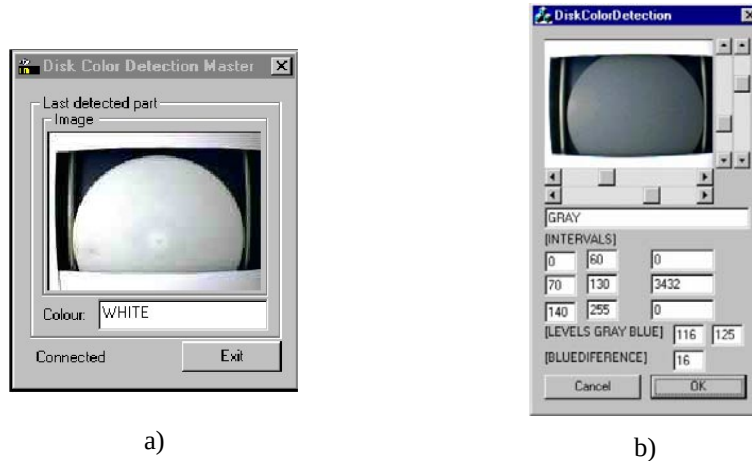


Figura 3.9: Interface da aplicação de captura e processamento de imagem no PC6 (b) e de visualização e resultado da classificação no PC1 (a)

O fluxo de tráfego MM5 é um fluxo bidireccional de voz. É utilizado o *Microsoft NetMeeting* (NetMeeting, 2001), quer do lado do PC1, quer do lado do PC4. O *Microsoft NetMeeting* usa ligações TCP e UDP em diferentes fases da comunicação. Esta aplicação é esporádica e simétrica, ou seja, pode a qualquer momento ser iniciada quer pelo PC1 quer pelo PC4, consoante a necessidade de comunicação dos operadores aí localizados. Existe uma aplicação (Figura 3.10b) desenvolvida em *Visual Basic 6.0* (Pacheco, 2002b) que interage com o *Microsoft NetMeeting* via *ActiveX*. A aplicação é simples, sendo disponibilizados ao operador os comandos “ligar” e “desligar”.

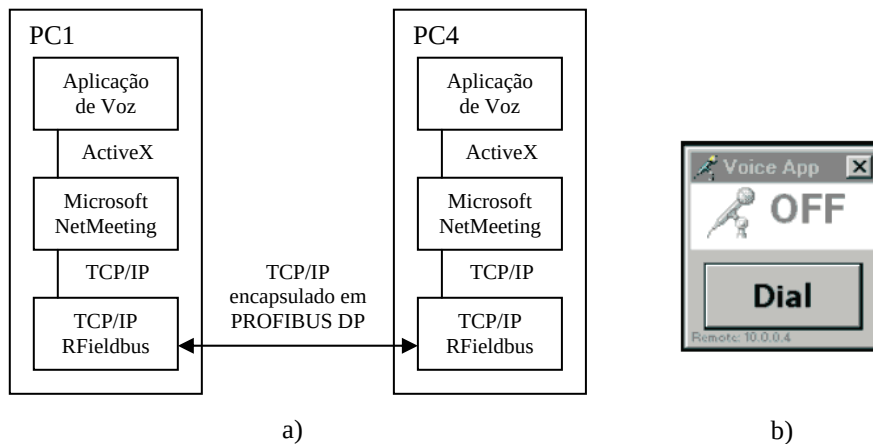


Figura 3.10: a) Contexto de utilização do *NetMeeting* numa plataforma RFieldbus; b) interface da aplicação desenvolvida em *ActiveX*

Os fluxos de tráfego MM6 e MM7 permitem a transferência e a manutenção de programas para os robôs R1 e R2, utilizando o efeito FTP. Do lado do PC2 e do PC4 é utilizado o serviço FTP do IIS (*Internet Information Server*), devidamente configurado. Do lado do PC1, são exploradas as capacidades de FTP do *Internet Explorer* ou de qualquer outro programa FTP cliente para *WindowsNT*.

A Tabela 3.5 sintetiza as aplicações de *software* e o *hardware* necessário à implementação das funcionalidades suportadas pelos fluxos de tráfego descritos na Tabela 3.4.

Tabela 3.5: Hardware e software para suportar as funcionalidades previstas para os fluxos de tráfego TCP/IP encapsulado em DP

<i>Fluxo</i>	<i>Nós</i>	<i>Hardware específico no PC1</i>	<i>Hardware específico no escravo</i>	<i>Software específico no PC1</i>	<i>Software específico no escravo</i>
MM1	PC6 ↔ PC1	–	Sistema Winnov Desktop PCI: Câmara Digital + carta de aquisição PCI	Disk Color Detection – Master; Image Center	Disk Color Detection – Slave
MM2	PC6 ↔ PC1	–	Sistema Winnov Desktop PCI: Câmara Digital + carta de aquisição PCI	Disk Color Detection – Master Image Center	Disk Color Detection - Slave
MM3	PC3 ↔ PC1	-	Sistema Winnov Traveler: Câmara Digital + carta de aquisição PCMCIA	Image Center	Image Client
MM4	PC5 ↔ PC1	-	Sistema Winnov Traveler: Câmara Digital + carta de aquisição PCMCIA	Image Center	Image Client
MM5	PC4 ↔ PC1	Colunas e altifalante	Colunas e altifalante	Netmeeting – Voice Application Browser; FTP Client	Netmeeting – Voice Application Web Server; FTP Server
MM6	PC2 ↔ PC1	-	-	Browser; FTP Client	Web Server; FTP Server
MM7	PC4 ↔ PC1	-	-	Browser; FTP Client	Web Server; FTP Server
MM8	PC2 ↔ PC1	-	Sistema Winnov Desktop PCI: Câmara Digital + carta de aquisição PCI	Robot Arm Operator - Master	Robot Arm Operator - Slave

3.5. Alguns Detalhes do Fluxo de Informação Vertical

Antes de finalizar este capítulo, vamos fazer uma descrição genérica da arquitectura utilizada para suportar o fluxo de informação vertical, isto é, o fluxo que existe entre a rede PROFIBUS/RFieldbus e a *Intranet*.

O fluxo de informação entre a rede PROFIBUS/RFieldbus e a *Intranet* é facultado através do PC1, que possui capacidade para manipular directamente os objectos de

comunicação da rede PROFIBUS/RFieldbus. É da responsabilidade do PC1 servir de “ponte” entre o SFF e a *Intranet*, disponibilizando informação sobre o estado do sistema, facilitando a manipulação propriedades/estado do sistema, e distribuindo os diversos fluxos de imagens. De forma a alcançar as funcionalidades pretendidas, foram desenvolvidas diversas aplicações que interagem entre si, facilitando diversos serviços para os possíveis clientes que se encontrarem na *Intranet*.

Na Figura 3.11, ilustram-se as aplicações que contribuem para a prestação de serviços aos clientes da *Intranet*, bem com as interações entre essas aplicações.

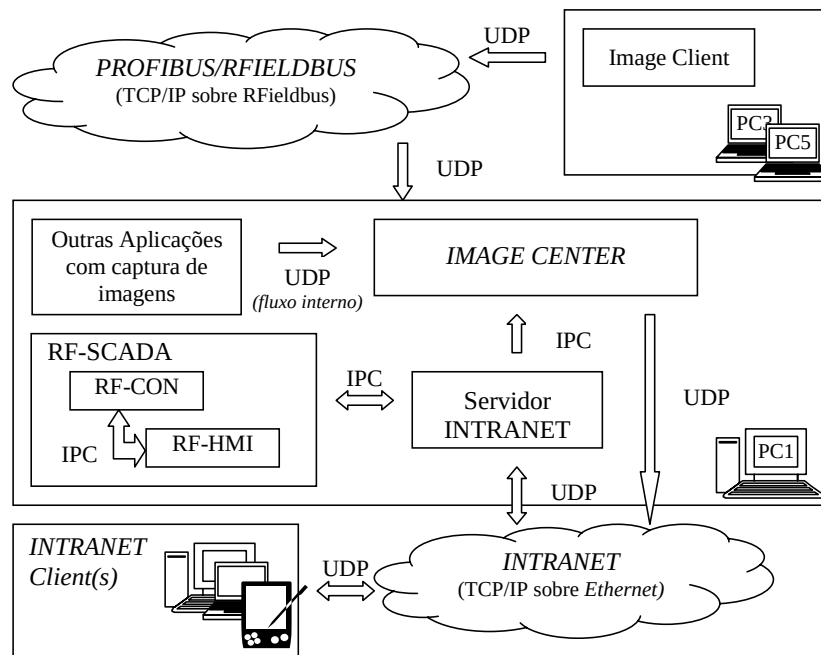


Figura 3.11: Aplicações que contribuem para a prestação de serviços aos clientes na *Intranet*.

A aplicação com o papel principal é a **Servidor Intranet**. Esta aplicação tem como responsabilidade difundir informação sobre o estado do SFF e permitir a manipulação de propriedades/estado do SFF por parte dos clientes da *Intranet*. A manutenção das propriedades do sistema e a modificação do estado deste é uma atribuição do módulo **RF-CON** da aplicação **RF-SCADA**. No entanto, a interação com outras aplicações é uma tarefa do módulo **RF-HMI**. Assim, utilizando um mecanismo de comunicação entre processos (**IPC – Inter-Process Communication**) baseado em mensagens da API *Windows*, o **Servidor Intranet** interage com o módulo **RF-CON**, através do módulo **RF-HMI**, de forma a, por um lado obter a informação sobre o SFF que irá difundir pela *Intranet* e, por outro lado, modificar as propriedades/estado do sistema.

Outro serviço prestado aos clientes *Intranet* é a distribuição dos vários fluxos de imagens capturadas no SFF. O controlo da distribuição destes fluxos é efectuado pelo

Servidor Intranet. Este mantém o número de clientes que estão a consumir um determinado fluxo, e pode desligar a difusão de fluxos que não possuem consumidores. No entanto, a distribuição propriamente dita é efectuada pela aplicação denominada *Image Center*. O *Image Center* funciona como um núcleo agregador de todos os fluxos de imagens existentes no sistema. Ele recebe as imagens provenientes de aplicações clientes encarregues da captura destas e de aplicações que façam uso de imagens para o seu funcionamento, como é o caso das aplicações responsáveis pela classificação de peças e do posicionamento do cesto de descarga. Esta aplicação recebe comandos do *Servidor Intranet* para ligar/desligar a distribuição de determinados fluxos via IPC, e difunde as imagens recebidas através de pacotes UDP enviados em *broadcast* para a *Intranet*.

Capítulo 4

Cooperação entre Processos em Ambiente *Windows*

Conforme mencionado no final Capítulo 3, o suporte a clientes “exteriores” ao SFF, por nós designados clientes *Intranet*, baseia-se num conjunto de aplicações cooperantes. Estas aplicações, desenvolvidas em *VisualBasic* 6.0, cooperam entre elas através de dois tipos distintos de mecanismos: *API Windows Messages* e *Windows Sockets*. Este capítulo aborda os aspectos essenciais destas duas abordagens distintas para comunicação entre aplicações em ambiente *Windows*.

4.1. Introdução

Na plataforma *Windows*, as aplicações são essencialmente guiadas por eventos. Tipicamente as aplicações não fazem explicitamente chamadas a funções para obter informação de entrada. Pelo contrário, as aplicações em *Windows* esperam que o sistema operativo lhes forneça informação *event driven*.

Por exemplo, quando o sistema operativo detecta que o utilizador pressionou alguma tecla do teclado, notifica as aplicações de destino (normalmente apenas a aplicação que estiver activa no momento) da ocorrência desse evento. O mesmo acontece quando o rato é movido, ou um botão do mesmo é pressionado, entre outros exemplos que podiam ser mencionados.

O *Windows* comunica estes eventos às aplicações através de mensagens (*Windows Messages*). Não é contudo só o sistema operativo que pode utilizar esta forma de comunicação. O mesmo mecanismo de comunicação pode ser também utilizado pelas aplicações, quer internamente (por exemplo comunicação entre janelas), quer entre aplicações distintas.

O mecanismo de mensagens da plataforma *Windows* é pouco exigente em termos de recursos e é bastante robusto. É por isso um mecanismo muito interessante para a comunicação de eventos entre aplicações *Windows*.

Este é o mecanismo utilizado pelas aplicações cooperantes referidas no Capítulo 3, e ilustradas na Figura 3.11 (fluxos de comunicação IPC). Na Secção 4.2 deste capítulo, é feita uma descrição mais detalhada do mecanismo. A Secção 4.3 aborda a implementação de uma classe auxiliar que adapta o módulo da *API Windows* ao modelo de eventos do *VisualBasic* 6.0, desenvolvida especificamente para as aplicações em apreço. Finalmente, na Secção 4.4, é feita uma descrição da utilização (mais trivial) de *sockets* (universo TCP/IP) para comunicações entre aplicações desenvolvidas em *VisualBasic*. Esta maior trivialidade resulta por um lado da simplicidade do conceito, por

outro lado do facto de o *VisualBasic* disponibilizar um componente específico para o efeito: o *Microsoft Winsock Control 6.0*.

4.2. Mensagens Windows – Conceitos Genéricos

Como já foi referido, para difundir informação sobre praticamente todo o tipo de eventos, o sistema operativo *Windows* utiliza um mecanismo específico de mensagens. Por outro lado, as próprias aplicações também podem gerar mensagens *Windows*.

A Figura 4.1 ilustra os conceitos fundamentais associados. Os eventos gerados (por exemplo por dispositivos ou aplicações) são recebidos pelo sistema operativo que, por sua vez, os processa e encaminha, sob a forma de mensagens, para os receptores apropriados. As mensagens podem ser enviadas explicitamente para um receptor, ou podem ser difundidas para todos. Os receptores, por sua vez, recebem as mensagens na sua fila de mensagens.

É importante notar que as mensagens não têm sempre de passar pela fila de mensagens do receptor. O emissor das mensagens pode escolher um envio síncrono ou assíncrono das mensagens. As mensagens assíncronas passam pela fila de mensagens, e são processadas pela ordem de chegada (FCFS) por uma rotina de *handling* das mensagens. As mensagens síncronas são directamente entregues à rotina de tratamento de mensagens, e o envio da mensagem (no “transmissor”) só termina quando esta for processada no “receptor”.

A Figura 4.1 contém uma série de exemplos ilustrativos. Por exemplo, a possibilidade de aplicações (APP4 e APP5) enviarem mensagens quer para a fila de uma ou de todas as aplicações, quer a possibilidade de serem enviadas mensagens directas entre aplicações (APP4 para APP3).

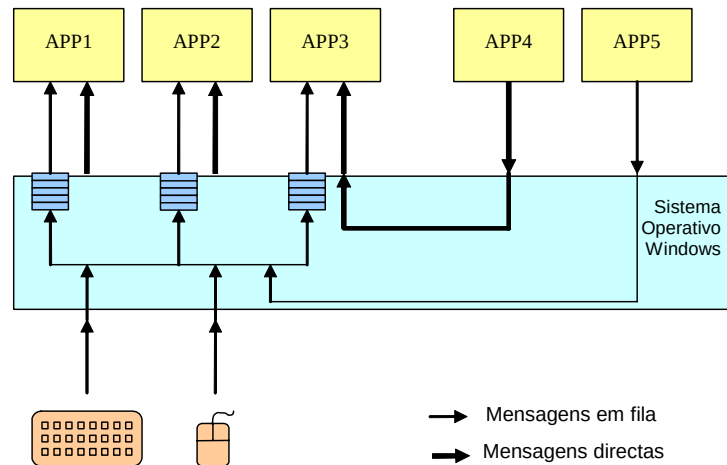


Figura 4.1: Fluxos de eventos/mensagens numa plataforma *Windows*

Cada mensagem *Windows* tem associado um conjunto de quatro atributos: um endereço do destinatário; um identificador da mensagem; e dois valores que são designados de parâmetros da mensagem.

O endereço do destinatário é utilizado para encaminhar a mensagem para a fila ou rotina de tratamento de mensagens apropriada.

O identificador da mensagem é uma constante numérica que identifica o significado de uma mensagem. Quando a rotina de processamento de mensagens recebe uma nova mensagem, utiliza este atributo para determinar a forma como deve processar a mensagem.

Os parâmetros da mensagem especificam os dados da mensagem. O valor e significado destes dependem da mensagem em particular. Os parâmetros da mensagem podem conter valores numéricos, valores binários codificados, apontadores para estruturas de dados, etc. Para saber como interpretar estes parâmetros, a aplicação tem de reconhecer o identificador da mensagem.

Para além de as mensagens poderem ser assíncronas ou síncronas, podem também ter uma génese de sistema ou de aplicação.

As mensagens definidas pelo sistema caracterizam-se por ser um conjunto de mensagens com uma interpretação predefinida. Estas mensagens são utilizadas pelo sistema operativo para comunicar à aplicação (ou aplicações) informação que pretende que estas processem. Nada proíbe, no entanto, as aplicações comuns de enviarem este tipo de mensagens. Exemplos de mensagens de sistema incluem comandos para a aplicação fechar, redesenhar a janela, envio de teclas ou atalhos activados, encerramento do sistema, etc.

As mensagens definidas pela aplicação são mensagens específicas criadas pelas aplicações, e portanto não possuem uma interpretação predefinida no sistema operativo. Por esta razão, as aplicações que trocam este tipo de mensagem têm de ser cooperantes, no sentido em que devem ter um entendimento comum sobre o significado das mensagens. A API de mensagens do *Windows* tem um mecanismo que permite fazer a identificação destas mensagens através de *strings*, o que simplifica a utilização das mesmas e evita o problema de distribuição de identificadores numéricos entre as diferentes aplicações.

4.3. Aspectos da Implementação do Mecanismo de Mensagens

Para facilitar a utilização de mensagens no *VisualBasic*, foi usada uma pequena classe auxiliar que adapta o módulo da API *Windows* para o modelo de eventos do *VisualBasic* 6.0. (Pacheco 2002e).

Os objectos da classe devem ser declarados com o prefixo `WithEvents`, como no exemplo seguinte:

```
Public WithEvents wm1 As WindowsMessages
```

Para enviar mensagens para filas são utilizados os métodos `PostMessageLong` e `PostMessageString`. A diferença entre os dois está na forma como é passado o segundo parâmetro da API de mensagens do *Windows*. `PostMessageLong` utiliza um

tipo `Long` do *VisualBasic*. `PostMessageString` utiliza um tipo `String`. No caso das mensagens predefinidas do *Windows*, a definição das constantes respectivas está feita na declaração da classe, pelo que basta passar apenas o identificador respectivo (como por exemplo `WM_SETFOCUS`). Para mensagens específicas entre aplicações é necessário invocar antes o `GetMessageID`, indicando o descritor textual respectivo.

Para mensagens directas são utilizados os métodos `SendMessageLong` e `SendMessageString`. A principal diferença entre estes métodos e os apresentados no parágrafo anterior é que, no primeiro caso a mensagem é colocada numa fila e o método retorna imediatamente, enquanto que no segundo caso a mensagem é entregue directamente à aplicação destino e o método só retorna quando esse processamento estiver terminado, devolvendo o resultado respectivo. Os métodos `SendMessageLong` e `SendMessageString` devem ser utilizados com cuidado, pois podem bloquear a execução de uma ou mais aplicações.

Para receber mensagens o processo é mais complexo.

Em primeiro lugar, é preciso registar as mensagens que gerarão eventos no objecto `WindowsMessage`, utilizando os métodos `RegisterMessage` e `RegisterStdMessage`. No primeiro caso, o parâmetro que identifica a mensagem é um descritor textual. No segundo caso são utilizados os identificadores predefinidos do *Windows*.

É preciso depois activar o processamento de mensagens pelo objecto `WindowsMessage`. Para o efeito utilizado o método `HOOK`. Este método substitui a função de processamento de mensagens *Windows* do *VisualBasic* por uma função interna do objecto. As mensagens registadas irão originar eventos `ReceivedLong` e `ReceivedString`. Para as mensagens que não foram registadas, as mesmas são passadas para a função original do *VisualBasic*. Para desactivar o processamento de mensagens pelo objecto usa-se o método `Unhook`.

A primeira chamada a `RegisterMessage` ou a `GetMessageID` (ou a `RegisterWindowMessage` no caso de utilização directa da API *Windows*) retorna um novo identificador único para cada descritor textual utilizado. As chamadas seguintes aos métodos, em qualquer aplicação no sistema utilizando o mesmo descritor textual, irá retornar o identificador da primeira chamada.

Uma limitação da classe `WindowsMessage` é que não é possível devolver valores quando se processa uma mensagem recebida, por ser utilizado o paradigma de eventos do *VisualBasic*. Outra limitação da classe é que entre a chamada a `HOOK` e a chamada a `Unhook` não é possível fazer *debug* do programa *VisualBasic*. Qualquer erro durante a execução do programa irá bloquear o sistema de desenvolvimento devido à função de processamento de mensagens inesperadas.

4.3.1. Alguns Detalhes da Implementação

O seguinte exemplo de código que se segue diz respeito à inicialização do objecto *wm1*.

```
Set wm1 = New WindowsMessages
wm1.ClearRegisteredMessages

wmRF_Msg_IR = wm1.RegisterMessage("RF_MSG_IR", wmtLong)
wmRF_Msg_MM = wm1.RegisterMessage("RF_MSG_MM", wmtLong)

wmRF_Msg_CMD = wm1.GetMessageID("RF_MSG_CMD")

wm1.Hook Me.hwnd
```

As variáveis *wmRF_Msg_IR* e *wmRF_Msg_MM* guardam os identificadores de duas mensagens que irão ser recebidas (com parâmetro *Long*), e *wmRF_Msg_CMD* guarda o identificador de uma mensagem que irá ser enviada.

O exemplo seguinte diz respeito ao código respeitante ao processamento de eventos do tipo *Long*.

```
Private Sub wm1_ReceivedLong(ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long)
    On Error Resume Next
    Select Case Msg
    Case wmRF_Msg_IR
        Select Case lParam
        Case 0
            IR(wParam).Visible = False
        Case 1
            IR(wParam).BorderColor = &HFFFF
            IR(wParam).Visible = True
        Case 2
            IR(wParam).BorderColor = &HFF&
            IR(wParam).Visible = True
        End Select
    Case wmRF_Msg_MM
        ' ....
        ' ....
        ' ....
    End Select
End Sub
```

No caso da mensagem *wmRF_Msg_IR*, o parâmetro *wParam* é utilizado para identificar um elemento de um vector de controlos, e o parâmetro *lParam* é utilizado para identificar uma operação sobre esse elemento.

As mensagens de difusão das aplicações *VisualBasic* são recebidas não pelas janelas (**form**) propriamente ditas mas sim por uma janela mãe invisível. O exemplo seguinte apresenta o código necessário para activar a recepção:

```
Public Const GW_OWNER = 4
Public Declare Function GetWindow Lib "user32" (ByVal hwnd As Long, ByVal wCmd As Long) As Long

' ...

hwnd2 = GetWindow(Me.hwnd, GW_OWNER)
wm2.Hook hwnd2
```

4.3.2. Referência Resumida

A constante definida no código seguinte é utilizada para enviar mensagens de difusão (*broadcast*).

```
Public Enum WindowsMessagesHWND
    HWND_BROADCAST = &HFFFF&
End Enum
```

As constantes seguintes são utilizadas para identificar directamente as mensagens predefinidas do *Windows*.

```
Public Enum WindowsMessagesID
    WM_NULL = &H0
    WM_CREATE = &H1
    WM_DESTROY = &H2
    WM_MOVE = &H3
    WM_SIZE = &H5
    WM_SETFOCUS = &H7
    ' ...
    ' ...
    ' ...
    WM_HOTKEY = &H312
    WM_PENWINFIRST = &H380
    WM_PENWINLAST = &H38F
    WM_USER = &H400
End Enum
```

O método seguinte permite limpar a lista de mensagens registadas no objecto.

```
object.ClearRegisteredMessages
```

Função seguinte retorna o identificador de mensagem correspondente ao descritor textual *MessageIDString*.

```
object.GetMessageID(MessageIDString)
```

O seguinte método activa o processamento de eventos do objecto, ligando-o à janela `hwnd`. Para desactivar o processamento, utiliza-se, como já referido, o método `UnHook`.

```
object.Hook hwnd
```

A função seguinte permite enviar uma mensagem (`Msg`) para a fila da janela `hwnd`, utilizando os parâmetros `wParam (Long)` e `lParam (Long)`. A função retorna imediatamente, devolvendo `True` se o envio foi bem sucedido, ou devolvendo `False` nas outras situações.

```
object.PostMessageLong(hwnd, Msg, wParam, lParam)
```

A função seguinte permite enviar uma mensagem (`Msg`) para a fila da janela `hwnd`, utilizando os parâmetros `wParam (Long)` e `lParam (String)`. A função retorna imediatamente, devolvendo `True` se o envio foi bem sucedido, ou devolvendo `False` nas outras situações.

```
object.PostMessageString(hwnd, Msg, wParam, lParam)
```

Os eventos são activados sempre que uma mensagem do tipo `Long` é recebida:

```
object_ReceivedLong(Msg As Long, wParam As Long, lParam As Long)
```

ou uma do tipo `String` é recebida:

```
object_ReceivedString(Msg As Long, wParam As Long, lParam As String)
```

A função seguinte permite registar a mensagem identificada pelo descritor textual `MessageIDString` para geração de eventos de recepção. Se `MessageType` for `wmtLong` então o evento `ReceivedLong` é chamado; caso seja `wmtString` então o evento `ReceivedString` é chamado. É responsabilidade do programador garantir que o parâmetro `lParam` pode ser utilizado como `Long` ou `String` para uma determinada mensagem. Não é possível registar o mesmo descritor textual `MessageIDString` novamente antes de fazer uma chamada a `UnregisterMessage` ou `ClearRegisteredMessage`. A função devolve 0 em caso de erro, ou o identificador da mensagem nos outros casos.

```
object.RegisterMessage(MessageIDString, MessageType)
```

A função seguinte permite registar a mensagem identificada pelo identificador `MessageID` para geração de eventos de recepção. Se `MessageType` for `wmtLong` então o evento `ReceivedLong` é chamado, caso seja `wmtString` então o evento `ReceivedString` é chamado. É responsabilidade do programador garantir que o parâmetro `lParam` pode ser utilizado como `Long` ou `String` para uma determinada mensagem. Não é possível registar o mesmo identificador `MessageID` novamente

antes de fazer uma chamada a `UnregisterMessage` ou `ClearRegisteredMessage`. A função devolve `True` se a mensagem tiver sido correctamente registada, ou `False` para os outros casos.

```
object.RegisterStdMessage(MessageID, MessageType)
```

Função seguinte permite enviar directamente a mensagem `Msg` para a janela `hwnd` utilizando os parâmetros `wParam (Long)` e `lParam (Long)`. A função só retorna quando a mensagem tiver sido processada pela aplicação destino, devolvendo o resultado respectivo.

```
object.SendMessageLong(hwnd, Msg, wParam, lParam)
```

Função seguinte permite enviar directamente a mensagem `Msg` para a janela `hwnd` utilizando os parâmetros `wParam (Long)` e `lParam (String)`. A função só retorna quando a mensagem tiver sido processada pela aplicação destino devolvendo o resultado respectivo.

```
object.SendMessageString(hwnd, Msg, wParam, lParam)
```

Método que desactiva o processamento de mensagens deste objecto. No caso de múltiplos objectos na mesma janela *VisualBasic* as chamadas a `UnHook` dos diversos objectos devem ser feitas pela ordem inversa das chamadas a `Hook`.

```
object.UnHook
```

Finalmente, o método seguinte permite remover a mensagem com identificador `MsgID` da fila de mensagens processadas por este objecto. Para remover uma mensagem utilizando o descritor textual, tem de se utilizar primeiro `GetMessageID` para obter o identificador respectivo.

```
object.UnregisterMessage MsgID
```

4.4. Windows Sockets

A forma mais directa de implementar *Sockets* em *VisualBasic* é utilizando o componente fornecido *Winsock* (também identificado por “*Microsoft Winsock Control 6.0*” ou pelo ficheiro respectivo: “*MSWINSCK.OCX*”). Este componente disponibiliza serviços de rede TCP e UDP.

O protocolo TCP é utilizado para ligações fiáveis ponto a ponto. O protocolo garante o estabelecimento da ligação entre as duas aplicações, controla a manutenção da ligação, a transmissão fiável e ordenada dos dados e, quando necessário, termina a ligação nas duas aplicações. O protocolo TCP utiliza o paradigma Cliente/Servidor: há uma aplicação Servidor que fica à espera de ligações de um ou mais Clientes.

4.4.1. Alguns Detalhes de Utilização

Para escrever uma aplicação Cliente TCP é necessário saber o endereço ou nome do servidor (propriedade `RemoteHost` do objecto `Winsock`), assim como o porto (`RemotePort`) que estabelece correspondente ao à “ligação” à aplicação. Com estes dados basta invocar o método `Connect` para iniciar a ligação.

Do lado da aplicação Servidor, o primeiro passo é definir o porto de ligação (`LocalPort`) onde o servidor fica “à escuta” e invocar o método `Listen`. Quando é recebido um pedido de ligação de um Cliente, é activado um evento `ConnectionRequest` no Servidor. Para completar a ligação, o Servidor tem de chamar o método `Accept` no processamento do evento.

Assim que a ligação estiver estabelecida, qualquer uma das aplicações pode enviar ou receber dados. Os dados são enviados utilizando o método `SendData`. Quando são recebidos dados, o evento `DataArrival` é activado. No processamento do evento, é utilizado o método `GetData` para obter os dados recebidos.

Para terminar a ligação, qualquer uma das aplicações pode chamar o método `Close`.

O protocolo UDP é, ao contrário do TCP, um protocolo sem ligação. Os dados podem ser enviados a qualquer momento, por qualquer aplicação, e não é garantido pelo protocolo a correcta entrega dos dados nem a sua ordenação.

O porto de recepção de dados é definido utilizando a propriedade `LocalPort`. A recepção de dados neste porto origina eventos `DataArrival`, em cujo *handler* deve ser chamado o método `GetData` (tal como no TCP).

Para enviar dados é necessário saber o endereço remoto e porto do computador destino (`RemoteHost` e `RemotePort`), e invocar o método `SendData`.

4.4.2. Referência Resumida

O método seguinte é utilizado em aplicações Servidor TCP para aceitar ligações remotas. É utilizado no evento `ConnectionRequest` (que tem o parâmetro `requestID`).

object.Accept requestID

O método seguinte é utilizado em aplicações Servidor TCP em computadores com múltiplos endereços IP, para especificar qual o adaptador que irá ser utilizado por este componente.

object.Bind LocalPort, LocalIP

O método seguinte é utilizado em aplicações TCP para terminar a ligação.

object.Close

Quando a ligação é terminada pela aplicação remota, é gerado, em aplicações TCP, o seguinte evento:

object.Close()

O seguinte método é utilizado em aplicações Cliente TCP para iniciar uma ligação a um servidor.

object.Connect remoteHost, remotePort

O seguinte evento é activado nas aplicações Servidor TCP quando um cliente inicia uma ligação. As propriedades *RemoteHostIP* e *RemotePort* identificam a aplicação cliente.

object.ConnectionRequest (requestID As Long)

O evento seguinte é activado quando a aplicação recebe dados. Normalmente é utilizado o método *GetData* para obter os dados.

object.DataArrival (bytesTotal As Long)

O evento seguinte é activado em situações de erro (*timeout*, utilização incorrecta do componente, etc.).

object.Error(number As Integer, Description As String, Scode As Long, Source As String, HelpFile as String, HelpContext As Long, CancelDisplay As Boolean)

O método seguinte é utilizado para reaver os dados recebidos. O parâmetro *data* é uma variável do tipo *variant* que vai receber os dados. O parâmetro *type* permite especificar qual o tipo de dados a guardar na variável. O parâmetro *maxLen* indica qual o comprimento máximo de dados a reaver. Os dados adicionais serão descartados.

object.GetData data, [type,] [maxLen]

Método seguinte cria um *socket* para Aplicações Servidor TCP e coloca-o em modo “à escuta”. Quando um cliente se ligar a este *socket* será activado o evento *ConnectionRequest*.

object.Listen

A propriedade seguinte identifica a porta local.

object.LocalPort [= long]

enquanto que a propriedade seguinte identifica qual o serviço (protocolo) de transporte a ser utilizado. Para protocolo TCP (por defeito) deve utilizar-se *sckTCPProtocol*. Para protocolo UDP deve utilizar-se *sckUDPProtocol*.

```
object.Protocol [= protocol]
```

A propriedade seguinte identifica o computador da aplicação remota (que será utilizado para enviar dados ou de onde os dados serão recebidos).

```
object.RemoteHost [= string]
```

A propriedade seguinte identifica a porta remota (que será utilizado para enviar dados ou de onde os dados serão recebidos).

```
object.RemotePort [= port]
```

O método seguinte é utilizado para enviar dados num socket.

```
object.SendData data
```


Capítulo 5

Clientes *Intranet*

Neste capítulo são descritas as funcionalidades, os aspectos de usabilidade e as opções de implementação da aplicação cliente *Intranet*. É essencialmente focada a versão para plataforma PC *Microsoft Windows*, desenvolvida em *Visual Basic 6.0* (VB6.0). No entanto, algumas das opções de desenho da aplicação tiveram em conta a sua fácil adaptação para plataformas *Microsoft Windows Pocket PC 2002* (para PDAs). Nestas últimas, a aplicação tem de ser desenvolvida em *eMbedded Visual Basic 3.0* (eMVB3.0). O eMVB3.0 apresenta algumas restrições no desenvolvimento da aplicação. Algumas destas restrições serão também referidas, quando adequado, neste capítulo.

5.1. Introdução

Como já foi mencionado nos Capítulos 3 e 4, optou-se por fazer o desenvolvimento da aplicação cliente em *Microsoft Visual Basic 6.0* (VB6.0) (Balena, 1999). Apesar de actualmente não ser o ambiente de desenvolvimento predilecto de muitos programadores, ele supre os requisitos de facilidade de desenvolvimento e de usabilidade necessários aos objectivos da aplicação Cliente *Intranet*.

Neste capítulo serão descritos os aspectos fundamentais da aplicação. Como já foi referido em capítulos anteriores, o objectivo principal desta aplicação Cliente *Intranet* é o de poder disponibilizar em *Laptops*, PDAs ou PCs com interface de comunicações *Ethernet-like* (versão *wired*, ou versão *wireless* IEEE 802.11x) ligados à *Intranet* a capacidade de monitorizar, configurar (alguns parâmetros) e controlar (algumas funcionalidades) o Sistema Flexível de Fabrico (SFF) descrito no Capítulo 3.

Este capítulo está organizado da seguinte forma. Na Secção 5.2 serão analisadas algumas opções genéricas relativas a aspectos de visualização e navegação das e entre as diferentes janelas da aplicação. São discutidas opções à luz das limitações impostas pelo requisito de a aplicação ser facilmente adaptável para plataformas *Microsoft Windows Pocket PC 2002*. Na Secção 5.3 são sintetizadas as mensagens UDP trocadas entre a aplicação Cliente *Intranet* e as outras entidades: Servidor *Intranet*, *Image Center*, *Gateway SMS* e outros Clientes *Intranet*. Estas mensagens são depois contextualizadas nas secções restantes deste capítulo a propósito da descrição das várias funcionalidades suportadas pela aplicação Cliente *Intranet*: de monitorização do SFF (Secção 5.4); de configuração de parâmetros do SFF, de Clientes *Intranet* de capacidade reduzida de e da *Gateway SMS* (Secção 5.5); de controlo de alguns dos dispositivos do SFF (Secção 5.6); e finalmente de geração de eventos/alarmes a difundir para outros Clientes *Intranet* e para a *Gateway SMS* (Secção 5.7).

5.2. Aspectos Genéricos da Aplicação

A aplicação é constituída por treze janelas (forms), dois módulos (modules) e vários ficheiros áudio. Os seus detalhes constam do Anexo A desta dissertação.

Ao executar a aplicação (executável UDPCliente.exe), é visualizada temporariamente uma janela de entrada, a *Entry_Form* (ver Anexo A.3), cuja aparência é a que se ilustra na Figura 5.1.



Figura 5.1: Janela de entrada (*Entry_Form*) da aplicação Cliente *Intranet*

A Figura 5.1 corresponde sensivelmente ao “tamanho real” da janela. Note-se que na versão da aplicação para PC, a que se descreve nesta dissertação, o tamanho escolhido para as janelas é também o adequado ao visor dos PDAs iPAQ 3870 (referidos na Tabela 3.2 do Capítulo 3). Estes utilizam a plataforma *Microsoft Windows Pocket PC 2002* e a versão Cliente *Intranet* para eles foi desenvolvida em *Microsoft eMbedded Visual Basic 3.0* (eMVB3.0) (Tacke e Basset, 2002).

Quando é executado o evento *Form_Load* (Secção A.3.3) relativo à janela *Entry_Form*, é lido um ficheiro de texto (o ficheiro *config.txt*) que contém o endereço IP do Servidor *Intranet*. Caso este ficheiro não exista, é utilizado o endereço IP de *loopback* (o 127.0.0.1), no pressuposto de que se trata de uma aplicação Cliente *Intranet* a executar no mesmo PC onde está o Servidor *Intranet* (o PC1 descrito no Capítulo 3). No sistema, e conforme consta da Tabela 3.2 do Capítulo 3, o endereço IP do PC1 (e por isso do Servidor *Intranet*) é o 10.0.8.1 (endereço privado). Por isso, e será este o endereço a utilizar no ficheiro *config.txt* para um Cliente *Intranet* “normal”;

isto é, a executar numa plataforma computacional que não a do PC1. Os Clientes *Intranet* terão de ter endereços da mesma rede IP da do PC1 (ver os endereços seleccionados alguns dos clientes específicos na mesma Tabela 3.2).

Voltando a alguns dos aspectos genéricos de desenho da aplicação Cliente *Intranet*. Repare-se que, ao contrário do que acontece noutras janelas da aplicação, na janela *Entry_Form*, o aspecto gráfico do fundo da janela (*background*) é definido com um objecto do tipo *Image* (o objecto *Image1*), e não por manipulação da propriedade *Picture* da própria *form*. Serve esta ressalva para fazer notar uma das restrições encontradas no desenvolvimento da versão em eMVB3.0. De facto, em eMVB3.0, os objectos do tipo *form* não possuem a propriedade *Picture* ou uma outra equivalente que possa ser manipulada para efeitos de manipulação do aspecto gráfico do fundo da *form*. Assim, a janela *Entry_Form* exemplifica, para a versão PC do Cliente *Intranet*, uma solução alternativa compatível com a versão para *Microsoft Windows Pocket PC 2002*.

Voltando ao código e aos objectos VB6.0 contemplados na *Entry_Form* (Secção A.3 do Anexo A desta dissertação). Nesta *form* existe um objecto do tipo *Timer*: o objecto *Timer1*. O evento *Timer1_Timer* é utilizado com dois objectivos. Por um lado fazer piscar (por manipulação da propriedade *Visible* do objecto *Image2* – ver Figura A.1) o objecto gráfico que na Figura 5.1 aparece por baixo de “IPP-HURRAY!”. Por outro lado, abrir a janela principal (a *Main_Form*) ao fim de algumas execuções do evento *Timer1_Timer*.

A *Main_Form*, ilustrada na Figura 5.2a (de notar que, tal como nas restantes figuras deste capítulo, as capturas das imagens do cliente são reduzidas a 60% do tamanho real), e descrita em detalhe na Secção A.4, contém, entre outros objectos, cinco objectos do tipo *CommandButton* (botões de acção) e um objecto do tipo *Winsock*, o *Wsk1*. Este último é um dos componentes mais importantes (o coração) da aplicação Cliente *Intranet*, sendo de particular relevo o evento *Wsk1_DataArrival* descrito na Secção A.4.3. Mas vamos, por enquanto, fixar a nossa atenção nos botões cuja propriedade *Caption* corresponde a *Monitor* (botão *B_Monitor*), *Config* (botão *B_Config*), *Control* (botão *B_Control*) e *Gen. Event* (botão *B_Event*).

Estes quatro botões no essencial dão acesso a outras tantas classes de funcionalidades da aplicação Cliente *Intranet*. O botão *B_Monitor* abre a janela *Monitor_Form* (Figura 5.2b). O botão *B_Config* abre a janela *Config_Form* (Figura 5.2d). O botão *B_Control* abre a janela *Control_Form* (Figura 5.2c). Finalmente, o botão *B_Event* abre a janela *Gen_Events_Form*.

Em qualquer uma destas quatro outras janelas existe um botão *B_Back* que permite retornar à janela *Main_Form*. O botão *B_Exit* da *Main_Form* termina a execução da aplicação cliente (ver o evento *Main_Form.B_Exit_Click* na Secção A.4.3).

A descrição das funcionalidades oferecidas por estas quatro opções será feita mais tarde neste capítulo. Vamos agora referir alguns aspectos relacionados com a navegação entre botões nas janelas da aplicação Cliente *Intranet*.

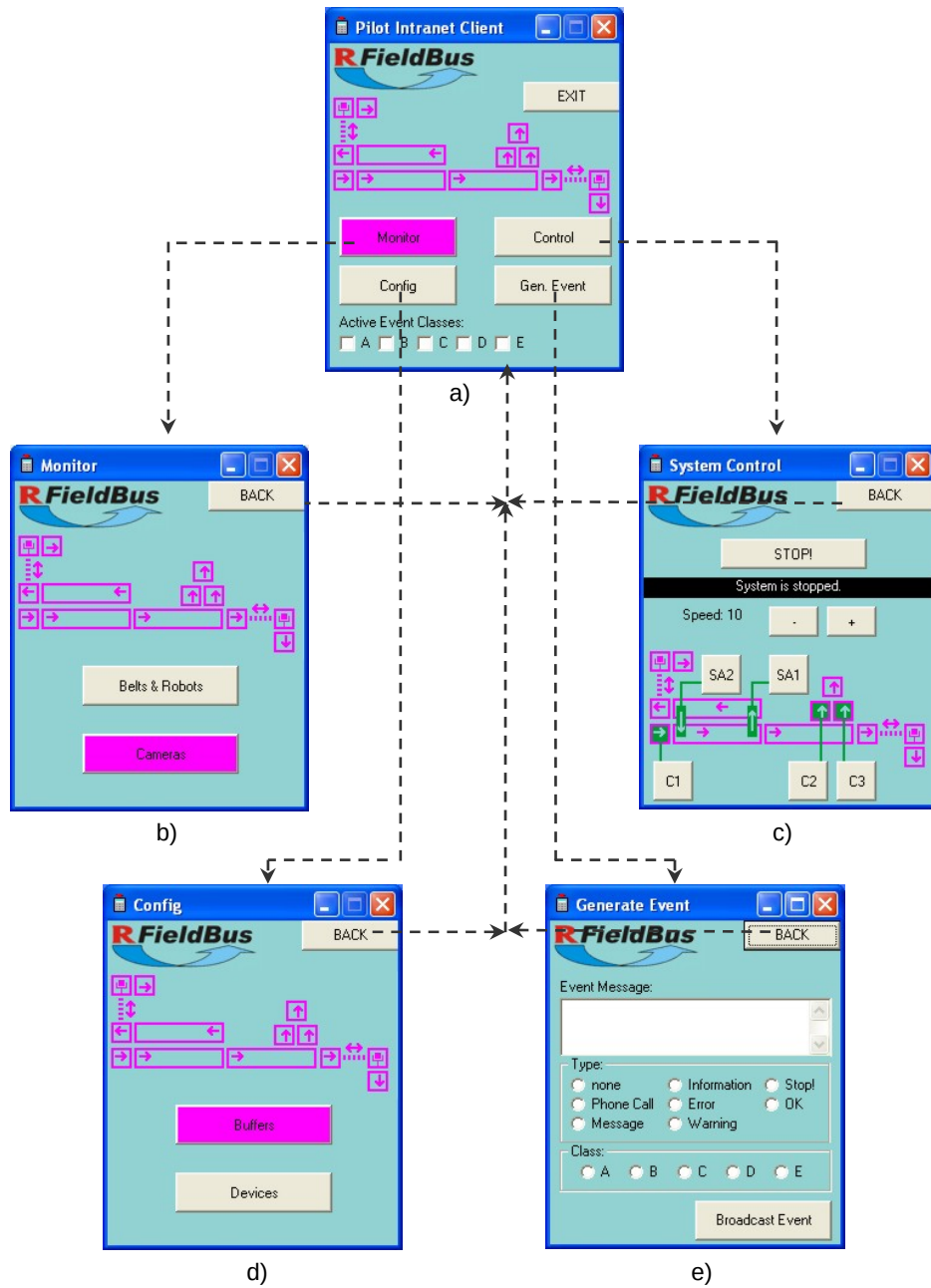


Figura 5.2: As quatro janelas principais acessíveis a partir da Main_Form (a): Monitor_Form (b); Control_Form (c); Config_Form (d); Gen_Events_Form (e)

Na versão para plataformas PC, o utilizador pode seleccionar uma das quatro janelas atrás referidas utilizando para o efeito o botão esquerdo do rato e posicionando o cursor em cima de um dos cinco botões existentes na *Main_Form* (Figura 5.2a).

Obviamente, e também para a versão plataforma PC, o utilizador pode alternativamente utilizar a tecla <TAB> do teclado para ir mudando o foco entre os objectos da form cuja propriedade *Enabled* tenha o valor *True*. No caso da *Main_Form*, esta alternativa permite ao utilizador uma “navegação” tipo round-robin entre os seis objectos (os cinco botões já referidos e os objectos do tipo *CheckBox* *chkAlarm(i)*) visíveis e *Enabled* contemplados na *Form*. Esta navegação “nativa” suportada pelo VB6.0 é ainda complementada com as teclas <LEFT>, <RIGHT>, <UP>, <DOWN> do teclado para navegar num array de objectos (como é o caso do objecto *Main_Form.chkAlarm(i)*) e pelas teclas <SPACE> e <ENTER> para seleccionar uma *CheckBox* ou para executar o código associado a um botão (evento *_Click* de um objecto do tipo *CommandButton*), respectivamente.

Estas capacidades *built-in* do VB6.0 permitem a utilização total da aplicação sem recurso ao rato. No entanto, em muitos PDAs a utilização de teclado é limitada. Concretamente, o iPAQ 3870 (Figura 5.3) oferece apenas as funcionalidades <LEFT>, <RIGHT>, <UP>, <DOWN>, <ENTER> no “*Joystick*” central e quatro botões adicionais para funções de utilização genérica (estes não possíveis de configurar para *keystrokes* associados a uma aplicação eMVB3.0).



Figura 5.3: Imagem do Compaq iPAQ 3870 executando uma versão Cliente *Intranet* em eMVB3.0 (visualização da janela equivalente à *Config_Devices_Form* da versão para plataforma PC – Figura 5.8c deste capítulo)

Na maioria dos PDAs, a função equivalente ao rato é suprida por um *touch screen*. No entanto, não é possível utilizar a navegação “nativa” por teclado existente no VB6.0 como descrita anteriormente.

Desse modo, e para facilitar a adaptação do código para as plataformas *Microsoft Windows Pocket PC 2002*, optou-se por incluir na versão Cliente Intranet para PC uma funcionalidade de navegação via teclado alternativa. Desta forma, e indirectamente, será feita uma referência à versão desenvolvida em eMVB3.0.

Esta versão alternativa (que por norma utiliza apenas as teclas <LEFT>, <RIGHT> e <ENTER> tem ainda algumas vantagens adicionais sobre a versão “nativa” do VB6.0. Uma, é que permite manipular (em termos de visibilidade e por isso usabilidade) as cores dos botões mais facilmente (a alternativa seria acrescentar eventos `_GotFocus` e `_LostFocus` para ir manipulando a cor dos botões que vão ganhando o foco com o a à medida que se prime a tecla <TAB>). Outra ainda é que permite implementar mais facilmente dependências de contexto entre janelas, principalmente quando uma mesma janela é utilizada para visualização de múltiplos dados. Outra ainda, é que permite limitar a navegação entre um sub conjunto de objectos `Enabled` na `form` (por exemplo só os objectos do tipo `CommandButton`).

Vamos agora descrever, genericamente, as diferentes soluções contempladas na versão para PC descrita em detalhe nesta dissertação.

Assim, na maioria das janelas da aplicação Cliente *Intranet*, é disponibilizada uma funcionalidade de navegação por teclas, utilizando um objecto “auxiliar” do tipo `TextBox` (o `Text1`), e o evento `Text1_KeyDown` a ele associado. Tipicamente o código contemplado no evento consiste simplesmente numa chamada ao `Form_KeyDown` existente na `form`. Nas janelas que contemplam esta funcionalidade de navegação “alternativa”, o objecto `Text1` não está na área visível da `form`. A necessidade de utilizar este objecto do tipo `TextBox` prende-se com o facto de o evento `Form_KeyDown` só ser evocado se o foco da janela estiver no próprio objecto `form`. Acontece que uma `form` só pode ter o foco se, e só se, ela não contiver objectos de controlo (p. ex. do tipo `CommandButton`) visíveis (propriedade `Visible = TRUE`) ou activos (propriedade `Enabled = TRUE`). Como é evidente, isto não acontece na maioria das `forms`.

Assim, o evento `Text1_KeyDown` é contemplado nas diversas `forms` da aplicação Cliente *Intranet*, excepto na `Config_Devices_Form` (ver Secção A.13), na `Gen_Events_Form` (ver Secção A.14) e na `Show_Events_Form` (ver Secção A.15). Como já foi referido atrás, o código do evento `Text1_KeyDown` corresponde simplesmente a uma chamada a `Form_KeyDown`, passando como parâmetro o código ANSI da tecla premida. Resolve-se assim o problema de o evento `Form_KeyDown` não ser evocado se a `form` tiver objectos activos ou visíveis.

Como também já foi referido, as `forms` `Config_Devices_Form`, `Gen_Events_Form` e `Show_Events_Form` não utilizam esta abordagem de navegação via teclado. Nas duas primeiras, e para ilustrar a alternativa de concepção, são utilizadas directamente as teclas <TAB>, <LEFT>, <RIGHT>, <UP>, <DOWN>, <SPACE> e <ENTER>. Em complemento, e para manter homogénea a filosofia de mudança de cor (em função da acção a propriedade `Backgroud` dos botões é `vbMagenta`, `Gray`, `vbGreen` ou `vbYellow`), são contemplados os eventos

`_GotFocus` e `_LostFocus` para ir manipulando a cor dos objectos em concordância (entre outros aspectos).

De notar que nas janelas que utilizam o evento `Text1_KeyDown`, não convém utilizar a tecla <TAB> (versão PC), uma vez que esta não provoca a evocação do evento `Text1_KeyDown`. O “efeito secundário” é a “perda de sincronismo” entre a manipulação das cores dos botões e do foco. Esse problema pode resolver-se com relativa facilidade acrescentando o evento `_GotFocus` (só este) dos botões e manipulando correctamente a variável `SelIndX` do form e as cores dos botões (propriedade `background`). Para exemplificar esta solução, ela é contemplada na form `Config_Form` (ver Secção A.11.3). Note-se no entanto que o <TAB> “vai” também ao objecto `Text1` (não implica grande problema – apenas num dos <TAB> nenhum botão adquire a cor `vbMagenta`).

Para o suporte da navegação “alternativa” por teclado, associados aos objectos do tipo `CommandButton` de cada form, existe um array de `CommandButton` (variável global à aplicação) definido no módulo `Globals.bas` (Secção A.2.2), variáveis essas listadas na Tabela 5.1.

Tabela 5.1: Arrays de `CommandButton` definidos em `Globals.bas` e sua associação lógica aos diversos forms da aplicação Cliente *Intranet*

Variável (Nome)	Dimensão	Form associado
<code>ButtMain(i)</code>	5	<code>Main_Form</code>
<code>ButtControl(i)</code>	9	<code>Control_Form</code>
<code>ButtMonitor(i)</code>	3	<code>Monitor_Form</code>
<code>ButtMonitorBR(i)</code>	7	<code>Monitor_BR_Form</code>
<code>ButtBRInfo(i)</code>	7	<code>BR_Info_Form</code>
<code>ButtSelectCAMInfo(i)</code>	6	<code>Select_CAM_Info_Form</code>
<code>ButtCAMView(i)</code>	8	<code>CAM_View_Form</code>
<code>ButtConfig(i)</code>	3	<code>Config_Form</code>
<code>ButtConfBuffers</code>	7	<code>Config_Buffers_Form</code>
<code>ButtConfDevices</code>	6	<code>Config_Devices_Form</code>

Nas forms que utilizam as funcionalidades de navegação “alternativa”, a definição do conteúdo (nome dos objectos `CommandButton` contemplados no respectivo form) dos arrays é feita no evento `Form_Load` da respectivo form.

Assim, e apenas como exemplo, no caso da `Entry_Form`, no evento `Timer1_Timer` (Secção A3.3), quando é feito o `Main_Form.Show`, é também feito o `Main_Form.Text1.SetFocus`, para atribuir o foco à `TextBox Text1` da `Main_Form`. Desta forma o objecto `Text1` da `Main_Form` passa a receber todos os *keystrokes* do teclado. O procedimento utilizado para mudar de janela visível (o `SwitchToForm` definido no módulo `Globals.bas`, e evocado na maioria das restantes forms) também executa o `Text1.SetFocus` da janela que vai mostrar.

O código da `Form_KeyDown` da `Main_Form` (a título de exemplo) contempla uma estrutura do tipo `Select-Case`, para as teclas `<DIR>` (código 39), `<ESQ>` (código 37) e `<ENTER>` (código 13). É feita a manipulação (casos teclas `<RIGHT>` e `<LEFT>`) da variável `SelInd` (índice do array `ButtMain(SelInd)`) e a manipulação da cor (propriedade `backgroud`) do botão respectivo. No caso `<ENTER>`, é feita a chamada ao procedimento `ButtClick`, passando como parâmetro o índice (`SelInd`) que corresponde ao contexto (botão) da navegação.

O procedimento `ButtClick` tem também uma estrutura do tipo `Case-Select`, que em função do índice passado irá activar a nova janela (por chamada ao já referido procedimento `SwitchToForm` definido em `Globals.bas`).

De notar ainda que, por norma, o *click* esquerdo do rato num botão de uma janela (eventos `<CommandButton.Name>_Click` faz uma chamada a `ButtClick(i)` da `form`, e não, por exemplo, uma mudança directa de janela.

Estas funcionalidades de navegação “alternativa” têm uma implementação um pouco mais complexa noutras `forms` da aplicação Cliente *Intranet*. A título de exemplo, na `Config_Buffers_Form` (ver código na Secção A.12.3), em `Form_KeyDown` são ainda considerados os códigos correspondentes às teclas `<+>` (códigos 107 e 187) e `<->` (códigos 109 e 189) para fazer a manipulação dos `labels L_PARM(i)` correspondentes ao botão com foco (índice `SelInd6`). (Note-se que na versão eMVB3.0, as teclas `<+>` e `<->` são substituídas pelas teclas `<UP>` (código 38) e `<DOWN>` (código 40).

Ainda na `Config_Buffers_Form`, a navegação alternativa inclui só os botões de envio da configuração dos *buffers* do SFF (os que têm ‘>’ na propriedade `Caption` – botões `B_SEND(i)`), o botão `B_RELOAD` e o botão `B_BACK` da `form`. Os botões `B_INCR(i)` e `B_DECR(i)`, não participam da funcionalidade de navegação “alternativa”. Acresce ainda que os botões `B_SEND(i)` podem estar com a propriedade `Enabled = FALSE` (em função do estado do sistema). Quando é executado o evento `B_SEND(i)_Click`, a propriedade `BackColor` do botão passa a `vbYellow` e é despoletado o `Timer1` (`Timer1.Enabled = TRUE`) da `form`. Quando passam os 500ms configurados no `Timer1.Interval`, o botão volta a ter a cor `vbMagenta` e `Timer1.Enabled = FALSE`.

A utilização deste `Timer` para manipulação de cor (amarelo durante 500ms) existe também nas `forms` `Config_Devices_Form`, `Control_Form`, e `Gen_Events_Form` (com variantes no código em função da `form`) para os botões correspondentes ao envio de dados via `Wsk1`. Não está implementada nesta versão, mas uma filosofia semelhante poderia ser utilizada em `View_CAM_Info_Form` para os botões de ON/OFF.

De notar ainda que não sendo fácil para os cliente *Pocket PC* a manipulação de algumas das `CheckBox` (por exemplo para seleccionar as classes de alarmes/eventos a receber – o objecto `chkAlarm(i)` da `Main_Form`), essa configuração pode ser feita ou através de outro cliente (`Config_Devices_Form` – Secção A.13), ou através do Servidor *Intranet* (como se verá no Capítulo 6 desta dissertação).

5.3. Súmula das Comunicações com Outras Aplicações

Um Cliente *Intranet* comunica com outros clientes, com o Servidor *Intranet*, com a *Gateway SMS* (Pacheco, 2002c) e com o *Image Center* via *Sockets UDP*. Para o efeito é contemplado um objecto do tipo *Winsock* (o *Wsk1*) na *Main_Form* (Secção A.4.2). O evento *Main_Form.Wsk1_DataArrival* é evocado sempre que o cliente recebe um pacote UDP (a propriedade *Main_Form.Wsk1.Protocol = sckUDPProtocol*). Na Secção A.4.3, é descrito o código para o evento *Wsk1_DataArrival*.

Com os outros clientes, com o *Image Center* e com a *Gateway SMS*, a informação (pacotes UDP) é enviada em modo *broadcast*. Antes de o cliente efectuar o envio (método *Wsk1.SendData*), a propriedade *RemoteHost* do objecto *Wsk1* é igualada a *BROADCAST_ADDR*. Esta constante está definida em *Globals.bas* e tem o valor “255.255.255.255”. Nesses casos a propriedade *RemotePort* do objecto *Wsk1* é igualada a *CLIENT_PORT*). Esta constante está definida em *Globals.bas* e tem o valor “1319”.

No caso da comunicação com o Servidor *Intranet*, a informação é enviada em *unicast*. Antes de o cliente efectuar o envio (método *Wsk1.SendData*), a propriedade *RemoteHost* do objecto *Wsk1* é igualada a *SERVER_ADDR*. Esta variável está definida em *Globals.bas* e tem o valor que for obtido no evento *Main_Form.Form_Load* na leitura ao ficheiro *config.txt* (conforme já referido no início da Secção 5.2). Nesses casos a propriedade *RemotePort* do objecto *Wsk1* é igualada a *SERVER_PORT*). Esta constante está definida em *Globals.bas* e tem o valor “1309”.

O cliente pode enviar diversas mensagens UDP. Na Tabela 5.2 são descritas essas mensagens e a *form.event* relacionada. As mensagens têm um significado especial em função do primeiro carácter. De todo o modo, são mensagens enviadas por acção do utilizador, e serão contextualizadas nas secções subsequentes deste capítulo a propósito das funcionalidades e interações (com o SFF) suportadas pela aplicação Cliente *Intranet*.

Note-se que a aplicação *Gateway SMS* é uma aplicação relativamente simples (não descrita em detalhe nesta dissertação). Essencialmente contempla um objecto de comunicações tipo *Winsock* com a propriedade *Protocol = sckUDPProtocol*.

A aplicação *Gateway SMS* recebe as mensagens do tipo “%SMS:...” (descrita sucintamente na Tabela 5.2 e explicada posteriormente a propósito do *Config_Devices_Form*). Esta mensagem é utilizada para saber o número de telemóvel de destino para o envio do SMS e quais as classes de alarme que devem ser enviadas por SMS para esse número. Em função dessas classes de alarme, sempre que a aplicação *Gateway SMS* recebe uma mensagem do tipo “!EVT:...”, se a classe está contemplada na configuração (contida numa mensagem “%SMS:...” recebida) é enviado o SMS. A aplicação *Gateway SMS* utiliza o mesmo valor para a propriedade *LocalPort* do seu objecto *Winsock*: “1319”. O envio de SMS é feito por um conjunto de rotinas que comunicam com um telemóvel via porta RS232.

Tabela 5.2: Tipos de pacotes UDP enviados pelo Cliente *Intranet*

Mensagens	Destino	Descrição	Janela
#OP #CB_SPEED	Servidor	Pedir informação sobre o estado do SFF	Main_Form.Form_Load
?QUEUE ?BUFF ?AGV1 ?AGV2 ?RA1 ?RA2	Servidor	Pedir informação relativa a dispositivos do SFF	Monitor_BR_Form.Form_Load
>CAM<i>:ON >CAM<i>:OFF	Servidor	Informar servidor sobre subscrição ou não de imagens da câmara <i>i</i>	CAM_View_Form.B_CM_OFF_Click CAM_View_Form.B_CM_ON_Click
#SA1:ACTIVATE #SA2:ACTIVATE #CYL1:ACTIVATE #CYL2:ACTIVATE #CYL3:ACTIVATE #CB_SPEED:SLOWER #CB_SPEED:FASTER #OP:EM_STOP	Servidor	Controlo do Sistema	Control_Form.Send_Command
&B<i>MAX	Servidor	Pedido de informação sobre o tamanho máximo do <i>buffer i</i>	Config_Buffers_Form.Form_Load
&B<i>MAX:<val>	Servidor	Pedido de alteração do tamanho máximos do do <i>buffer i</i> com o valor <i>val</i>	Config_Buffers_Form.B_SEND_Click
%HMD:<a1>:<cam> %PK1:<a1>:<cam> %PK2:<a1>:<cam>	Clientes	Configuração do cliente HMD/PK1/PK2 com as classes de alarme <i>a1</i> e câmara <i>cam</i>	Config_Devices_Form.B_Info_Cfg_Click
%SMS:<a1>:<tel>	Clientes e Gateway SMS	Configuração da Gateway SMS para enviar os eventos/alarmes das classes <i>a1</i> para o número <i>tel</i>	Config_Devices_Form.B_SMS_Cfg_Click
!EVT:<a1>:<t>:<x>	Clientes	Envio de um evento/alarme da classe <i>a1</i> , tipo <i>t</i> e com o texto <i>x</i>	Gen_Events_Form.B_Broad_Event_Click

Se as mensagens listadas na Tabela 5.2 correspondem a mensagens enviadas pela aplicação Cliente *Intranet* como resultado da acção do utilizador da aplicação, o mesmo não se passa com as mensagens recebidas na aplicação Cliente *Intranet*.

Como já foi referido no início desta secção, o evento `Main_Form.Wsk1_DataArrival` é evocado sempre que o cliente recebe um pacote UDP. Na Secção A.4.3, é descrito o código para o evento `Wsk1_DataArrival`.

O cliente pode receber pacotes UDP cujo primeiro carácter é o indicado na Tabela 5.3. Para cada caso (ver código do evento `Wsk1_DataArrival`) é evocado um procedimento específico para tratar cada tipo de mensagem.

Tabela 5.3: Tipos de mensagens UDP recebidos no Cliente *Intranet*

Tipo	Enviado por	Descrição	Procedimento evocado
?	Servidor	Informação RA/AGV/Q/B	<code>Arrived_Monitor_Update</code>
>	Servidor ou <i>Image Center</i>	Câmaras e Imagens	<code>Arrived_Video_Info</code>
&	Servidor	Tamanho Máx. <i>Buffers</i>	<code>Arrived_Buffer_Max</code>
#	Servidor	Estado do Sistema	<code>Arrived_Control_Update</code>
%	Servidor ou outros Clientes	Configuração HMD/PK/SMS	<code>Arrived_HMD_PK_SMS</code>
!	Servidor ou outros Clientes	Alarmes/Eventos	<code>Arrived_Alarm_Event</code>

Conforme já foi dito atrás a propósito da Tabela 5.2, a contextualização e acções despoletadas pela chegada das mensagens brevemente referidas na Tabela 5.3 serão explicadas em mais detalhes nas secções seguintes deste capítulo.

5.4. Funcionalidades Suportadas: Monitorização do SFF

Quando se executa a aplicação cliente, é activada a janela de entrada, a `Entry_Form` (Secção A.3), e conforme já foi descrito na Secção 5.2 deste capítulo, é activada a janela `Main_Form` (Figura 5.2a). Esta janela reflecte quatro funcionalidades distintas do cliente.

5.4.1. Monitorização dos Tapetes, *Buffers* e Robôs

Uma das funcionalidades mais relevantes disponibilizadas aos Clientes *Intranet* é a possibilidade de monitorizarem, em tempo real, o estado do SFF.

Quando a opção correspondente ao botão `B_Monitor` da `Main_Form` é seleccionada, é visualizada a janela `Monitor_Form` (Figura 5.4b), que por sua vez contém duas opções, via botões `B_Belts_Rob` e `B_Cam` (ver Anexo A.5). Ao seleccionar a opção correspondente ao botão `B_Belts_Rob`, é visualizada a janela `Monitor_BR_Form` (Figura 5.4c). Esta janela permite seleccionar um de entre cinco itens de monitorização do SFF sobre: os braços robóticos (RA1, RA2); os AGVs (AGV1, AGV2); a quantidade de peças nos *buffers* e a quantidade de peças nas filas.

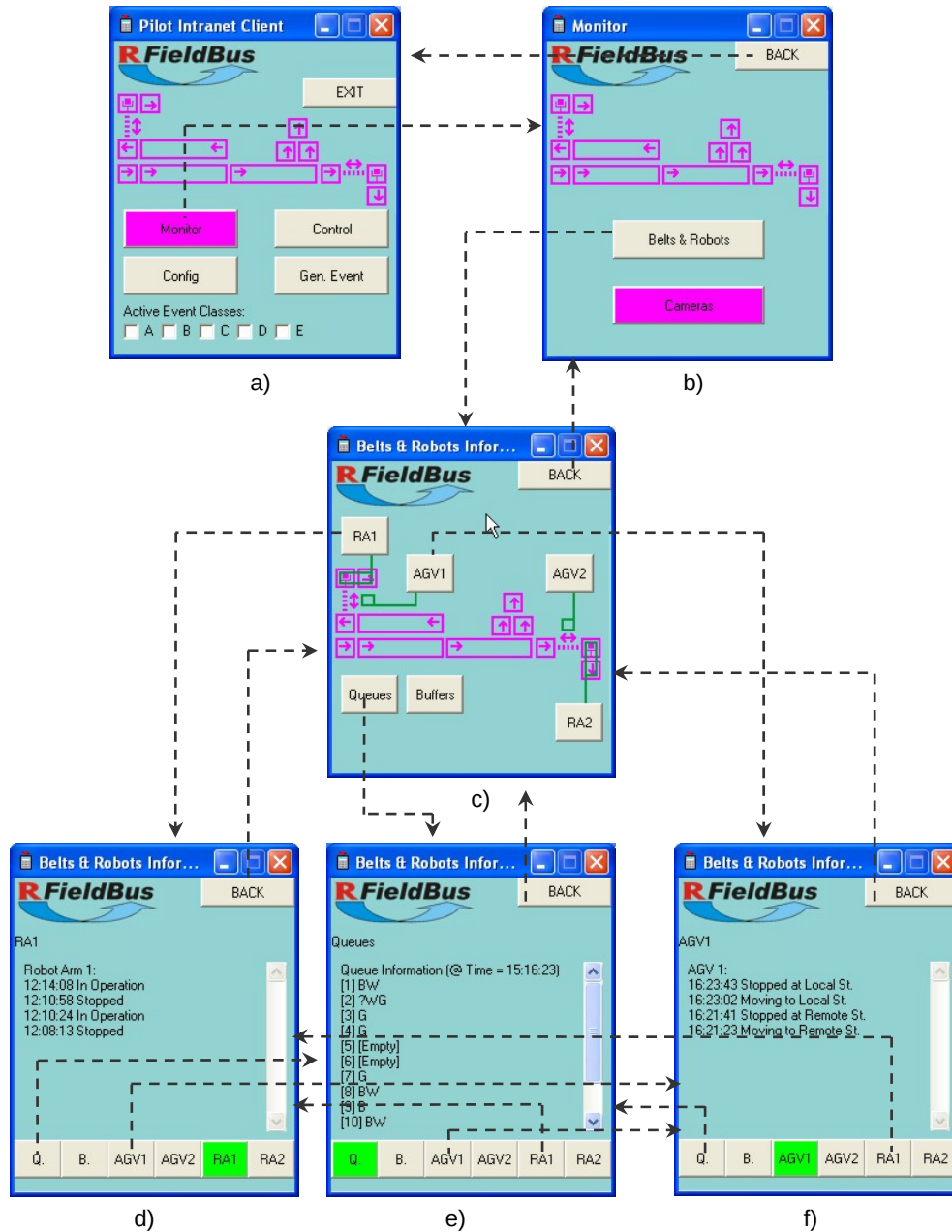


Figura 5.4: A janela acessível a partir da Main_Form quando se selecciona a opção Monitor: Monitor_Form (b). A partir desta a opção Belts & Robots permite a visualização da janela Monitor_BR_Form (c). Por sua vez, ao seleccionar-se um dos equipamentos, é feita a visualização da janela BR_Info_Form (d – contexto RA1; e – contexto QUEUES; f – contexto AGV1)

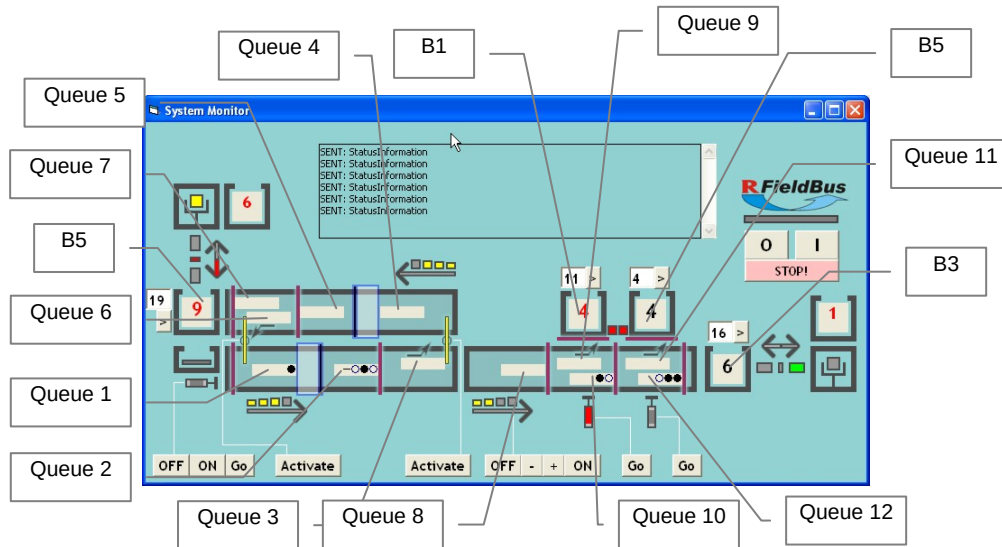


Figura 5.5: A aplicação RF-HMI e as definições de *Queues* e *Buffers*

De notar que as filas são zonas específicas dos tapetes rolantes (zonas delimitadas, por exemplo, pelos sensores IR associados aos tapetes). A Figura 5.5 ilustra essas zonas, utilizando para o efeito uma captura da aplicação RF-HMI (ver Capítulo 6).

Na janela `Monitor_BR_Form` (ver Anexo A.6), quando se selecciona um dos botões (`B_AGV1`, `B_AGV2`, `B_RA1`, `B_RA2`, `B_BUFFERS` ou `B_QUEUES`) é visualizada a janela `BR_Info_Form`, sendo o contexto de visualização função do botão seleccionado em `Monitor_BR_Form`. Note-se que em `Monitor_BR_Form`, a selecção de um desses botões irá (em `Monitor_BR_Form.ButtonClick`) evocar o procedimento `SwitchToForm` para mudar para `BR_Info_Form` (Figura 5.4d/e/f) e executar nesta última form o botão relativo ao contexto seleccionado em `Monitor_BR_Form`.

Em `BR_Info_Form`, o procedimento `ButtonClick` vai actualizar em concordância com o contexto de visualização, a `Label1.Caption` com as strings “Queues”, “Buffers”, “AGV1”, etc., e colocar na `TextBox Text_BR_Info` o conteúdo do array `Info_Monitor(i)`, no qual `i` corresponde ao contexto de visualização do dispositivo.

O array de strings `Info_Monitor(i)` é uma variável global definida em `Globals.bas`. Esta variável é actualizada em `Main_Form`, no evento `Wsk1_DataArrival`, conforme se explica a seguir.

No evento `Main_Form.Wsk1_DataArrival` caso o pacote UDP recebido tenha como primeiro carácter o carácter “?” (ver Tabela 5.3), será evocado o procedimento `Arrived_Monitor_Update` (ver código na Secção A.4.3).

Este procedimento vai identificar (variável local `id_dev`) sobre o que é que é a informação recebida. Para cada um dos casos vai actualizar a variável global `Info_Monitor (id_dev)`, como indicado na Tabela 5.4 para alguns exemplos (alguns deles visíveis na Figura 5.4).

Tabela 5.4: Exemplo de `Info_Monitor(id_dev)` no procedimento `Arrived_Monitor_Update`

Mensagem	id_dev	Exemplo para <code>Info_Monitor(id_dev)</code>
?QUEUE:<info>	1	Queue Information (@Time = 15:16:23) [1] BW [2] ?WG [3] G [4] G [5] [Empty] [6] [Empty] [7] G [8] BW [9] B [10] BW [11] B [12] [Empty]
?BUFF:<info>	2	Buffer Information (@Time = 12:29:46) [1] 6 [2] 14 [full] [3] 18 [4] 14 [5] 10 [full]
?AGV1:<info>	3	AGV 1: 16:21:23 Moving to Remote St. 16:21:41 Stopped at Remote St. 16:23:02 Moving to Local St. 16:23:43 Stopped at Local St.
?AGV2:<info>	4	AGV 2: 17:31:39 Moving to Local St. 17:32:01 Stopped at Local St. 17:38:02 Moving to Remote St. 17:38:41 Stopped at Remote St.
?RA1:<info>	5	Robot Arm 1: 12:08:13 Stopped 12:10:24 In Operation 12:10:58 Stopped 12:14:08 In Operation
?RA1:<info>	6	Robot Arm 2: 13:12:13 In Operation 13:13:09 Stopped 13:16:27 In Operation 13:17:21 Stopped

Para além disso, o procedimento `Arrived_Monitor_Update` vai actualizar a `TextBox Text_BR_Info` para a janela de contexto “activa” (determinada através da variável pública `SelInd4` manipulada da `BR_Info_Form`). Desta forma, se `BR_Info_Form` estiver activa, e se a mensagem recebida disser respeito ao contexto de visualização, a caixa de texto é actualizada com a informação recebida na mensagem UDP vinda do servidor.

Note-se também, que quando o evento `Form_Load` da `Monitor_BR_Form` executa, são enviados pedidos ao servidor (`Main_Form.Wsk1.SendData`) relativamente a estes parâmetros de monitorização (ver Tabela 5.2, mensagens de pedido de informação com o cabeçalho “?”). Por defeito, e antes da primeira actualização da informação relativamente aos dispositivos (recepção de uma mensagem UDP vinda do servidor em `Main_Form.Wsk1_DataArrival` com cabeçalho “?”), a variável `Monitor_Info(i)` contem a string “Updating...”.

5.4.2. Monitorização do SFF Através das Câmaras

A outra possibilidade de monitorização do SFF corresponde à visualização em tempo real das imagens recolhidas pelas cinco câmaras (ver Capítulo 3) associadas ao SFF. Esta opção corresponde a seleccionar o botão `B_Cam` do `Monitor_Form` (Figura 5.6b), que permite a visualização do `Select_CAM_Info_Form` (Figura 5.6c).

Na `Select_CAM_Info_Form` (ver Secção A.8) é possível seleccionar uma das cinco câmaras para visualização (botões `B_CMx`). Ao seleccionar a câmara (ver por exemplo o evento `B_CM1_Click` na Secção A.8.3), é visualizada a `CAM_View_Form` e executado (para o exemplo da câmara 1) `CAM_View_Form.ButtonClick 1`.

O procedimento `ButtonClick` (ver Secção A.9.3) na `CAM_View_Form` vai, entre outros aspectos, actualizar a `CAM_View_Form.CamPic.Picture` (`Set CamPic.Picture = cam (index)`) com a imagem guardada na variável global `cam(i)`, e actualizar as duas `Labels` referidas na Secção A.9.2.

O procedimento `ButtonClick` na `CAM_View_Form` é relativamente elaborado devido à funcionalidade suportada de navegação entre botões na janela. O botão da câmara seleccionada aparece à cor `vbGreen`, e o mesmo acontecerá no botão relacionado da `Select_CAM_Info_Form` (ver linha de código no procedimento `ButtonClick` da `CAM_View_Form: Select_CAM_Info_Form.ButtonClick Index`).

Existem três variáveis globais definidas em `Globas.bas` que são utilizadas para guardar informação relevante relacionada com a informação relacionada com as câmaras:

```
Public cam(5) as IPPictureDisp
Public g_txt_cam(5) as String
Public g_cam_enable(5) as Boolean
```

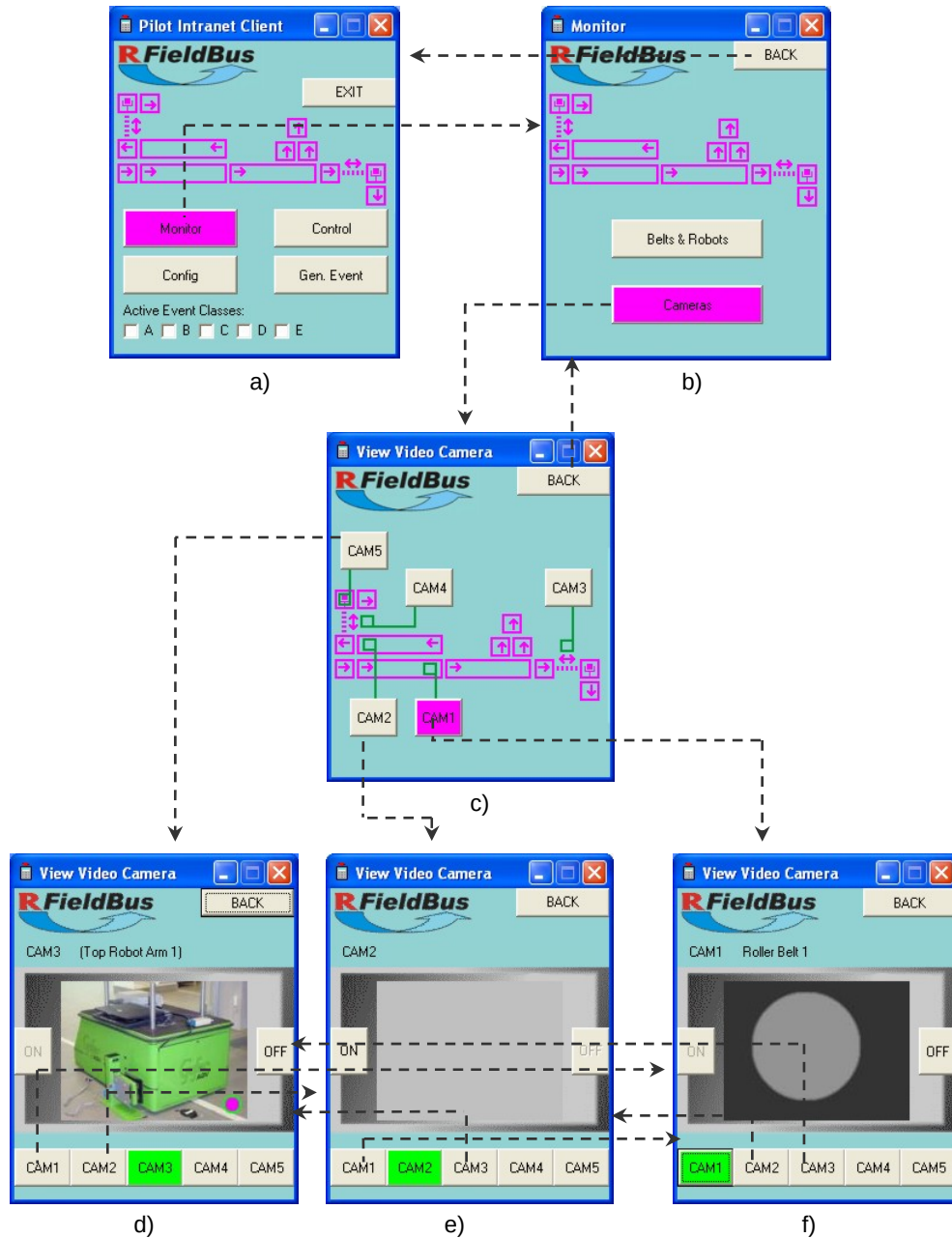


Figura 5.6: A partir de Monitor_Form (b) pode seleccionar-se o Select_CAM_Info_Form (c). Deste, pode seleccionar-se, View_CAM_Form (d, e, f) uma câmara para visualizar (ou activar / desactivar a recepção de imagens dela provenientes). No exemplo, a recepção das CAM 1 e 3 está activada, enquanto a CAM 2 está desactivada. Na visualização de CAM 1 vê-se uma peça preta (B).

Estas variáveis são utilizadas pela `CAM_View_Form` para a correcta visualização da informação (texto e imagens). Na `CAM_View_Form` é alterada a variável `g_cam_enable(i)` (quando executados os eventos `B_CM_ON_Click` ou `B_CM_OFF_Click` (ver Secção A.9.3). De notar, nesses eventos, que é enviado ao Servidor *Intranet* (endereço IP `SERVER_ADDR`) (ver Tabela 5.2 as mensagens com o cabeçalho “>”) a informação de acção `ON` ou `OFF` de uma câmara (corresponde a informar o servidor se o cliente subscreve ou não as imagens da câmara específica). De notar que o servidor utiliza esta informação para a sua interações com a aplicação *Image Center*. Se não houver “subscritores” de uma determinada câmara digital, o *Image Center* não difunde (*broadcast* UDP) as imagens dessa câmara, evitando assim uma carga de comunicações na *Intranet* desnecessária. (Ver Capítulo 6 para mais detalhes sobre isto).

As três variáveis atrás referidas são actualizadas no evento `Wsk1_DataArrival` da `Main_Form`. Quando é recebida pelo Cliente Intranet uma mensagem UDP com o cabeçalho “>” (ver Tabela 5.3) no evento `Wsk1_DataArrival` é evocado procedimento `Arrived_Video_Info` (ver código na Secção A.4.3).

Existem dois subtipos de mensagens recebidas com cabeçalho “>” (Tabela 5.6).

Tabela 5.5: Mensagens UDP recebidas com o primeiro caracter a “<”, e processadas em `Main_Form.Arrived_Video_Info`

Subtipo	Mensagem	Possibilidades para os campos das mensagens
1	>CAM:<CamID><Info>	CamID = {1, 2, 3, 4, 5} Info = {ON, OFF}
2	>IMG:<ImgID><FragID><CamID><TotFrag><Info>	Caso FragID = 0 ImgID = (byte) FragID = 0 CamID = {1, 2, 3, 4, 5} TotFrag = (byte) Info = (String)
	>IMG:<ImgID><FragID>	Caso FragID {1..TotFrag} ImgID = (byte) FragID = (byte ≠ 0) IMG = bloco gráfico

No caso de se tratar de uma mensagem com cabeçalho “>CAM:” (tipo 1 na Tabela 5.6), esta diz respeito à activação/desactivação da subscrição das imagens de uma câmara. Esta informação é guardada no `array` global já referido `g_cam_enable(CamID)`, e é feita uma manipulação adequada da propriedade `Enabled` dos botões `B_CM_OFF` e `B_CM_ON` se a janela estiver activa (isto é, se `Cam_View_Form.SelInd15 = <CamID>`).

No caso de se tratar de uma mensagem com cabeçalho “>IMG:” (tipo 2 na Tabela 5.6), o processamento é relativamente mais complicado.

O processo de recepção de imagens é baseado em dois formatos da mensagem UDP (do tipo “>IMG:...”) recebida.

A primeira mensagem de uma imagem é identificada pelo valor 0 no campo **FragID** (ver Tabela 5.6). Neste caso, os campos seguintes identificam a câmara utilizada (campo **CamID** com valores entre 1 e 5), o número de pacotes UDP de dados da imagem (campo **TotFrag**) e o restante da mensagem UDP é informação de texto que é apresentada ao utilizador juntamente com a imagem (**CAM_View_Form.L_CAM_Info.Caption** – Secção A.9.1) e guardada no array global já referido **g_txt_cam(CamID)**. A título de exemplo, na Figura 5.5f, esta **Label** contém “Roller Belt 1”.

Todos os campos utilizados são 8 *bits* sem sinal; isto é, com valores entre 0 e 255. Assim, e na prática o número de fragmentos em que um ficheiro (imagem) pode ser dividido pelo *Image Center* fica limitado a 255 (o fragmento 0 é usado para informação adicional, e não para os dados da imagem).

Como as imagens utilizadas são da ordem dos poucos *kbytes* (ver Secção 3.4.3 do Capítulo 3), esta limitação não impõe restrições à aplicação. Mesmo com fragmentos UDP relativamente pequenos, por exemplo de 256 *bytes*, as imagens utilizadas são divididas num número de fragmentos que varia entre 5 e 20.

Os pacotes UDP seguintes (**FragID** > 0) contêm apenas o identificador de imagem (**ImgID**) e a sequência de fragmento (**FragID**) (ver Tabela 5.6). O identificador de imagem **ImgID** é utilizado para agrupar as diversas mensagens UDP (fragmentos) do tipo “>IMG:...” relativas a uma mesma imagem. A sequência de fragmento (**FragID**) é um número de 1 até ao valor recebido **TotFrag** recebido na mensagem inicial (em que **FragID** era igual a 0).

Em termos de processamento, no procedimento **Arrived_Video_Info** a recepção do fragmento 0 (**FragID** = 0) origina as seguintes acções principais (ver código em A.4.3).

1. Fechar o ficheiro com identificador **#(10+<CamID>)** (que pode estar aberto no caso da imagem anterior não ter sido completamente recebida).
2. Abrir o ficheiro **cam\rcvcam<CamID>.jpg** com identificador **#(10+<CamID>)**;
3. Actualizar a variável **g_txt_cam_temp(CamID)** local à **Main_Form** com o campo **Info** da mensagem recebida.
4. A relação entre os campos **CamID** e **ImgID** é armazenada na **Collection ListCamIDs**, usando **ImgID** como chave. Se a chave é repetida (o que pode acontecer com imagens não completamente recebidas) então a entrada antiga é removida e depois inserida a nova entrada.

Em termos de processamento, no procedimento **Arrived_Video_Info** a recepção de um fragmento de imagem com **FragID** ≠ 0 origina as seguintes acções principais (ver código em A.4.3).

1. Procurar **CamID** na **Collection ListCamIDs**, utilizando **ImgID** como chave. Se não for encontrado a mensagem é ignorada (imagem inválida).
2. Obter o campo de identificação (**FragID**) do fragmento.
3. Obter os dados binários da mensagem correspondentes ao fragmento da imagem (campo **IMG** da Tabela 5.6).

4. Acrescentar ao ficheiro com o identificador `#(10+<CamID>)` os dados binários. Nota: quando utilizado num ficheiro binário (`Open For Binary`) o comando `Put` do VB6.0 começa sempre a escrever os dados do início do ficheiro de forma sequencial, mesmo que o ficheiro já exista e tenha dados.
5. Incrementar o contador de fragmentos recebidos. Quando o contador excede o valor de `TotFrag` para a câmara, significa que foi recebido o último fragmento dessa imagem.

Caso seja o último fragmento de uma imagem, vão ser executadas as seguintes acções principais no procedimento `Arrived_Video_Info`.

1. Fecha o ficheiro com identificador `#(10+<CamID>)`.
2. Remove a entrada da `Collection ListCamIDs`.
3. Actualiza o variável global `g_txt_cam(CamID)` com o valor da variável temporária `g_txt_cam_temp(CamID)` (esta última é utilizada para evitar que o utilizador do cliente visualize uma imagem antiga com um `L_CAM_Info.Caption` entretanto actualizado pela chegada de um primeiro fragmento de uma imagem).
4. Lê o ficheiro `cam\rcvcam<CamID>.jpg` para um objecto do tipo `IPictureDisp` (o `datapcx` local ao procedimento `Arrived_Video_Info`).
5. Copia para a variável global já referida `cam(CamID)` o objecto anterior.
6. Se esta imagem é da câmara seleccionada pelo utilizador para visualização, então os dados respectivos no `CAM_View_Form` são actualizados (a imagem: `CAM_View_Form.CamPic.Picture` e a informação em `CAM_View_Form.L_CAM_Info.Caption`).

De notar, que nesta versão *Cliente Intranet* para PC (em VB6.0), é possível receber múltiplas imagens (referentes a outras tantas câmaras) em simultâneo. São utilizados identificadores de ficheiros separados (5 identificadores diferentes – #11 ... #15) e os arrays `TotFrag(CamID)` e `CurFrag(CamID)` globais a `Main_Form`. Estas duas variáveis são inicializadas para `CamID` aquando da recepção da mensagem com `FragID = 0`, respectivamente com os valores de `TotFrag` (recebidos na mensagem) e com o valor 1.

Cada vez que é recebido um fragmento válido com `FragID ≠ 0` para `CamID`, `CurFrag(CamID)` é incrementado em uma unidade. Quando `CurFrag(CamID) > TotFrag(CamID)`, então foram recebidos todos os fragmentos da imagem.

No caso da versão para plataforma *Microsoft Windows Pocket PC 2002* em `eMVB3.0`, não é possível a um cliente ter activa mais do que uma câmara.

De facto, o componente `WinSockCtl` do `eMVB3.0` só suporta dois protocolos: o `sckTCPProtocol` (TCP) e o `sckIRDAProtocol` (IrDA). Foi desenvolvido um componente para suporte básico de UDP (Pacheco, 2002d).

Por outro lado, em `eMVB3.0`, para ler imagens apenas se podem utilizar formatos BMP, não sendo possível ler directamente formatos JPG como em VB6.0. Por isso na versão *Pocket PC* utiliza-se uma DLL para fazer conversão de ficheiros JPG para BMP.

O componente UDP desenvolvido no âmbito do projecto RFieldbus tem uma interface para escrever para ficheiros em modo binário. No entanto, por razões de simplicidade de implementação, o componente apenas suporta um ficheiro de cada vez (as operações são `OpenFile`, `CloseFile`, `WriteToFile`). A consequência é que apenas pode ser recebida uma imagem de cada vez, pois não é possível ter múltiplos ficheiros abertos ao mesmo tempo.

5.4.3. Outros Aspectos de Monitorização do SFF

Existem ainda outros aspectos de monitorização do sistema que não acessíveis a partir da opção do botão `B_Monitor` do `Main_Form`. Esses aspectos estão relacionados com o estado (`STOP|RUN`) do sistema e com a velocidade dos tapetes rolantes.

Essa informação chega ao cliente obviamente também via mensagem UDP vinda do servidor, e tendo um cabeçalho “#” (ver Tabela 5.3) é processada no procedimento `Arrived_Control_Update` evocado no `Main_Form.Wsk1_DataArrival` (ver código Secção A.4.3). A Tabela 5.7 ilustra os formatos possíveis das mensagens UDP recebidas neste caso.

Tabela 5.6: Mensagens recebidas sendo o primeiro carácter um “#” e processadas em `Arrived_Control_Update` da `Main_Form`

Mensagem	Possibilidades para os campos das mensagens
#OP:<ParmVal>	ParmValue = {STOP, STOPPING, RUN, ERROR}
#CB_SPEED:<ParmVal>	ParmValue = {0, 1, 2, 3, 4}

A informação recebida é utilizada para actualizar a `L_STATUS.Caption` da `Config_Buffers_Form` (Secção A.12) e a `L_STATUS.Caption` da `Control_Form` (Secção A.10), e caso “#OP:STOP” alterar a propriedade `Enabled` de alguns dos botões das referidas forms para `TRUE`.

Só se o sistema estiver no estado `STOP` é que é possível alterar o tamanho máximo dos *buffers* (no `Config_Buffers_Form`) e actuar os *Swivel Arms* e cilindros pneumáticos e alterar a velocidade (`Control_Form`). Caso a mensagem seja “#OP:RUN” ou “#OP:STOPPING” ou “#OP:ERROR” a propriedade `Enabled` dos referidos botões e das referidas forms passa a `FALSE`. Caso seja recebido “#CB_SPEED:<ParmVal>”, `L_SPEED.Caption` da `Control_Form` é actualizada com o novo valor da velocidade.

5.5. Funcionalidades Suportadas: Configuração do SFF e de Clientes

Outra das funcionalidades disponibilizadas aos Clientes *Intranet* é a possibilidade de alterarem alguns dos parâmetros do SFF (o tamanho máximo dos *buffers*) e poderem configurar clientes de capacidade reduzida (HMD, *Pocket PCs*) ou a *Gateway* de SMS.

5.5.1. Configuração de Parâmetros do SFF

Quando a opção correspondente ao botão **B_Config** da **Main_Form** é seleccionada, é visualizada a janela **Config_Form** (Figura 5.7b), que por sua vez contém duas opções, via botões **B_Buf** e **B_Dev** (ver Secção A.11).

Ao seleccionar a opção correspondente ao botão **B_Buf**, é visualizada a janela **Config_Buffers_Form** (Figura 5.7c).

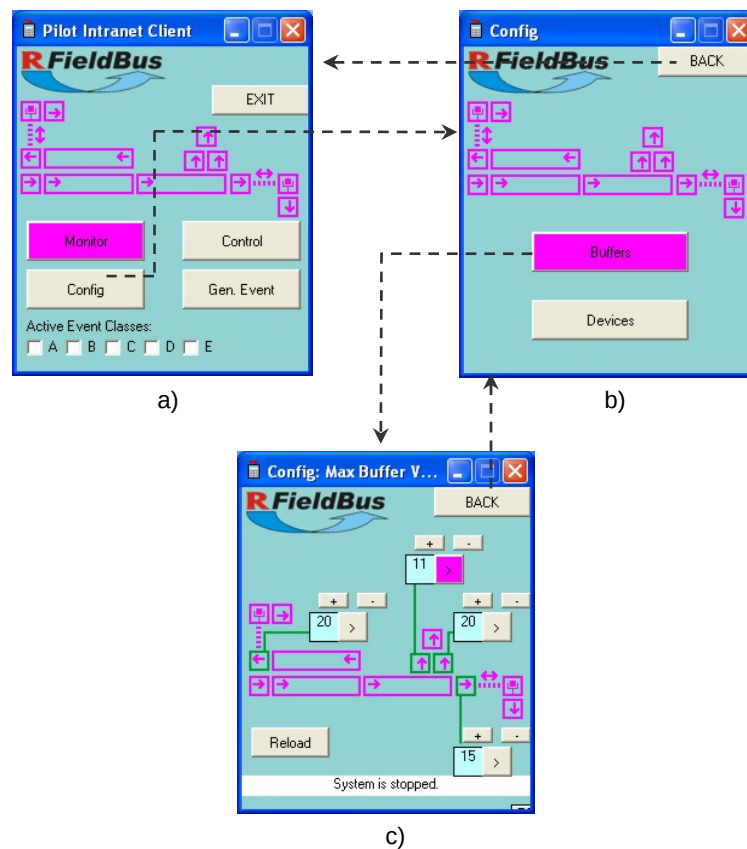


Figura 5.7: A partir de **Config_Form** (b) pode seleccionar-se o **Config_Buffers_Form** (c)

O código relativo a esta janela (Secção A.12.3) só é relativamente menos trivial no que diz respeito à navegação entre botões da janela (já referida genericamente na Secção 5.2 deste capítulo) e à gestão das cores dos botões (incluindo a acção de envio de mensagens UDP suportadas em `Config_Buffers_Form.B_SEND(i)`).

Quando a propriedade `Enabled` dos botões `B_SEND(i)` está a `TRUE`, é enviado para o Servidor *Intranet*, via `Main_Form.Wsk1.SendData` a mensagem `"&B<i>MAX:<L_PARM(i)>"` (ver Tabela 5.2, mensagens com cabeçalho "&". `L_PARM(i)` é uma `Label` que contém o valor pretendido para o novo valor do

tamanho máximo do *buffer i*. Como já foi referido na Secção 5.4.3 deste capítulo, só é possível enviar este comando se o sistema estiver em estado *STOP*. Os valores das Labels *L_PARM(i)* são actualizados pelo Servidor *Intranet* sempre que há uma alteração no *RF_CON*.

Reportando outra vez à Tabela 5.3, quando em *Main_Form.Wsk1_DataArrival* é recebida uma mensagem UDP com “&” como primeiro carácter, é evocado o procedimento *Arrived_Buffer_Max* da *Main_Form* (ver código na Secção A.4.3).

A informação é utilizada para actualizar a *L_PARM(i)* da *Config_Buffers_Form* e a variável global *info_bufmax(i)* definida em *Global.bas* (Secção A.2.2).

De notar que *info_bufmax(i)* reflecte o estado real do SFF. Se o utilizador alterar o valor das *L_PARM(i)* e depois quiser visualizar o valor real pode utilizar o botão *B_RELOAD* disponível para o efeito em *Config_Buffers_Form*.

Por fim, de referir que o evento *Form_Load* da *Config_Buffers_Form* envia (*Form_Main_Wsk1.SendData “&B” & i & “MAX”*) pedidos de actualização dos valores reais de *BMAX(i)* (já referidos na Tabela 5.2), pelo que estes serão actualizados quase de imediato (com o *Arrived_Buffer_Max*).

5.5.2. Configuração de Clientes de Output e da Gateway SMS

Outra funcionalidade de configuração presente nos Clientes *Intranet* está relacionada com a possibilidade de configuração de dispositivos de capacidades limitadas e da *Gateway SMS*. Em relação aos primeiros, o interesse dessa funcionalidade já foi abordado na Secção 5.2 deste capítulo.

Assim, se se seleccionar o botão *B_Dev* da *Config_Form*, abrir-se-á a janela correspondente à *Config_Devices_Form* (Figura 5.8c).

Como já foi referido na Secção 5.2 deste capítulo, a navegação na janela *Config_Devices_Form* (Secção A.13) recorre ao esquema mais clássico do VB6.0 (utilização das teclas <TAB> e cursores, complementado com os eventos *_GotFocus* e *_LostFocus* dos botões da form para a manipulação das cores dos botões (ver código na Secção A.13.3). Existem dois botões que dizem respeito às configurações já mencionadas: *B_Info_Cfg* e *B_SMS_Cfg*.

O primeiro permite o *broadcast* (envio para outros Clientes *Intranet*) da configuração de um dos três dispositivos (*HMD|PK1|PK2*), consoante a selecção activa do *B_Info_Select(i)*. A mensagem enviada é obtida a partir da composição de uma string da propriedade *Caption* da *CheckBox Ck1(i)* (os valores do *Caption* são A, B, C, D ou E, respectivamente para *Ck1(1)* até *Ck1(5)*) e do valor da opção de câmara (obtido a partir do *OptionButton Opt1(i)*).

Assim, por exemplo, para a selecção ilustrada na Figura 5.8c, a mensagem enviada (ver Tabela 5.2, as mensagens com cabeçalho “%”) no evento B_Info_Cfg_Click do Config_Devices_Form será:

“%PK1:AC:2”

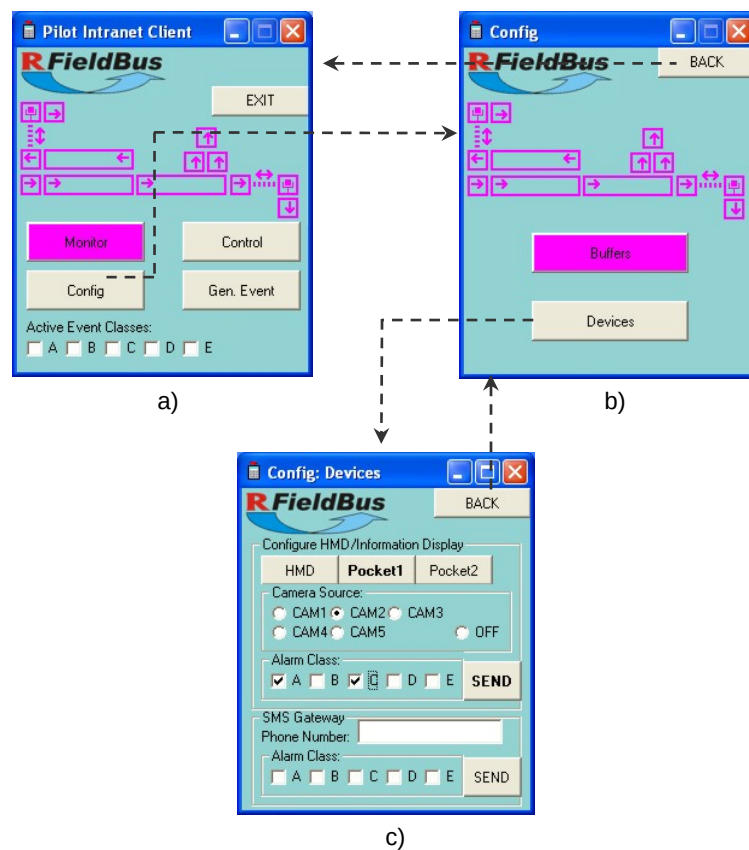


Figura 5.8: A partir de Config_Form (b) pode seleccionar-se o Config_Devices_Form (c)

Note-se que, como é utilizada a mesma **form** para os 3 dispositivos, a configuração para cada um deles é guardada em variáveis globais definidas em **Globals.bas**:

```
Public info_cam(3) As Integer
Public info_alarm(3) As String
```

Assim, quando é executado o evento `B_Info_Select(i)` da `Config_Devices_Form` (ver código na Secção A.13.3), os objectos do tipo `CheckBox Ck1(i)` e do tipo `OptionButton Opt(i)` vão ser actualizados a partir das variáveis `info_alarm(i)` e `info_cam(i)`, respectivamente.

Estas variáveis também são alteradas se algum outro cliente (ou se o Servidor *Intranet*) faz um *broadcast* para o mesmo efeito.

Assim, o procedimento `Form_Main.Arrived_HMD_PK_SMS` (ver Tabela 5.3) será chamado quando o primeiro carácter da mensagem UDP recebida em `Form_Main.Wsk1_DataArrival` for “%”.

Dependendo de se tratar de uma mensagem “%HMD:...”, “%PK1:...”, ou “%PK2:...”, as variáveis globais `info_cam(i)` e `info_alarm(i)` vão ser actualizadas sendo $i = 1, 2$ ou 3 , respectivamente. Em simultâneo, são actualizadas diversas propriedades dos objectos da `Config_Devices_Form` (por exemplo, a propriedade `Value` do `Ck1(i)`).

No caso de a mensagem recebida ser “%SMS:...”, é actualizada a `TextBox Config_Devices_Form.T_SMS_Number` e a propriedade `Value` do `Ck2(i)`.

5.6. Funcionalidades Suportadas: Comando de Dispositivos do SFF

A partir da `Main_Form`, quando se selecciona o botão `B_Control`, é visualizada a `Control_Form` (Figura 5.2c). O código e os objectos considerados nessa form estão discriminados na Secção A.10.

Sucintamente, existem botões que permitem o envio, via `Wsk1`, dos comandos discriminados na Tabela 5.8 (e já referidos na Tabela 5.2).

Tabela 5.7: Mensagens enviadas com o primeiro carácter “#” na `Control_Form`

Evento (Botão)	Mensagem enviada
<code>B_SA1_Click</code>	“#SA1:ACTIVATE”
<code>B_SA2_Click</code>	“#SA2:ACTIVATE”
<code>B_C1_Click</code>	“#CYL1:ACTIVATE”
<code>B_C2_Click</code>	“#CYL2:ACTIVATE”
<code>B_C3_Click</code>	“#CYL3:ACTIVATE”
<code>B_SP1_Click</code>	“#CB_SPEED:SLOWER”
<code>B_SP2_Click</code>	“#CB_SPEED:FASTER”
<code>B_STOP_Click</code>	“#OP:EM_STOP”

Existem mensagens recebidas do Servidor *Intranet* relacionadas com a operação do sistema. Elas já foram referidas na Secção 5.4.3 e na Tabela 5.7. Elas só têm relação com as três últimas mensagens da Tabela 5.8. Obviamente, e também já foi referido atrás, à

excepção do botão `B_STOP_Click` da `Control_Form`, todos os outros estão desactivados caso o sistema não esteja no estado `STOP`.

5.7. Funcionalidades Suportadas: Geração e Visualização de Alarmes/Eventos

A partir da `Main_Form`, quando se selecciona o botão `B_Event`, é visualizada a `Gen_Events_Form` (Figura 5.2e). O código e os objectos considerados nessa form estão descritos na Secção A.14.

Sucintamente, existe um botão (o `Gen_Events_Form.B_Broad_Event`) que permite o envio, via `WSK1`, de um evento/alarme em *broadcast* para todos os Clientes *Intranet* e para a *Gateway SMS*. Esses eventos serão processados pelos clientes que tiverem seleccionado a recepção dessa classe de alarme. Essa selecção pode ser feita quer no `Main_Form` (objecto do tipo `CheckBox chkAlarm(i)`) ou em `Show_Events_Form` (Figura 5.9) (objecto do tipo `CheckBox chkAlarm(i)`).

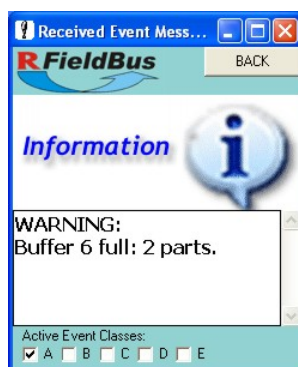


Figura 5.9: `Show_Events_Form` da aplicação cliente

Os eventos/alarmes recebidos no Cliente *Intranet* serão processados pelo procedimento `Main_Form.Arrived_Alarm_Event` (Secção A.4.3), que é evocado no evento `Main_Form.Wsk1_DataArrival` quando o primeiro carácter da mensagem UDP recebida é “!” (ver Tabela 5.3).

Se o alarme/evento recebido for de uma classe “subscrita” no cliente, é evocado o procedimento `ShowAlarm` definido no módulo `Globas.bas` (Secção A.2.2). Dependendo do tipo, é actualizada a propriedade do objecto `Show_Events_Form.Picture1.Picture` com o ficheiro formato `BMP` correspondente ao tipo de evento/alarme (ver Secção A.16.3 e Figura A.16). O objecto do tipo `TextBox Sow_Events_Form.T_Event` é actualizado com a informação recebida na mensagem UDP.

Se a form activa não é a `Show_Events_Form`, esta é visualizada. É utilizado um ficheiro áudio associado ao tipo de evento/alarme recebido (ver Secção A.17).

De notar que, como se verá em detalhe no Capítulo 6 desta dissertação, a única mensagem do tipo “!EVT...” recebida no Cliente proveniente do Servidor *Intranet* é da classe A e corresponde ao evento *buffer full*. Outros eventos poderiam ser programados no Servidor *Intranet*.

Capítulo 6

Suporte da Interacção entre Clientes *Intranet* e o SFF

Neste capítulo são descritas as funcionalidades, os aspectos de usabilidade e as opções de implementação das aplicações Servidor *Intranet* e RF-HMI, numa perspectiva de suporte às interacções dos Clientes *Intranet* com o SFF. Essas aplicações são desenvolvidas em *Microsoft Visual Basic 6.0* (VB6.0).

6.1. Introdução

Como já foi referido do Capítulo 3, existem diversas aplicações de *software* envolvidas no suporte às funcionalidades descritas, no Capítulo 5, para os Clientes *Intranet*. A Figura 6.1 sintetiza essas aplicações bem como os mecanismos de comunicação entre elas.

As aplicações AGV1 e AGV2 correspondem a *Viewers* das imagens recebidas das câmaras associadas a *Laptops* transportados pelos dois AGVs. Como já foi referido também, a aplicação *Image Centre* congrega os cinco fluxos de vídeo, que serão difundidos via UDP para a *Intranet* se subscritos por algum cliente.

A aplicação RF-HMI é a componente de interface gráfica (sinóptica) do SFF com o operador. O estado do sistema é reflectido, em tempo real, nessa interface, uma vez que qualquer alteração de estado do SFF é enviada pela aplicação RF-CON, via *Windows Messages*, para a aplicação RF-HMI. O operador do PC1 pode também controlar o SFF através dos botões disponibilizados na interface (Figura C.1, Anexo C). Estes botões, quando premidos, correspondem ao envio de mensagens *Windows* para o RF-CON.

Como já transpareceu do Capítulo 5, são considerados vários tipos de Clientes *Intranet*. A sua tipologia é ilustrada na Figura 6.2.

Os clientes HMD/PK1/PK2 são, como já foi referido, clientes especiais, de funcionalidades reduzidas. Tipicamente só suportam a visualização de uma câmara e a visualização de alarmes/eventos gerados quer na *Intranet* (por outros clientes), quer pelo SFF (via Servidor *Intranet*), quer pelo próprio servidor *Intranet*. Nestes, o fluxo de comunicações UDP não é bidireccional: é só do PC1 para eles. O mesmo se passa com o cliente que tem o papel de *Gateway SMS*. Como todos os clientes utilizam a mesma porta para o *socket IP*, só é possível executar uma aplicação cliente (incluindo HMD/PK1/PK2 e *Gateway SMS*) por máquina.

Ilustra-se também na Figura 6.2 uma plataforma PDA, versão aplicação cliente desenvolvida em eMVB3.0.

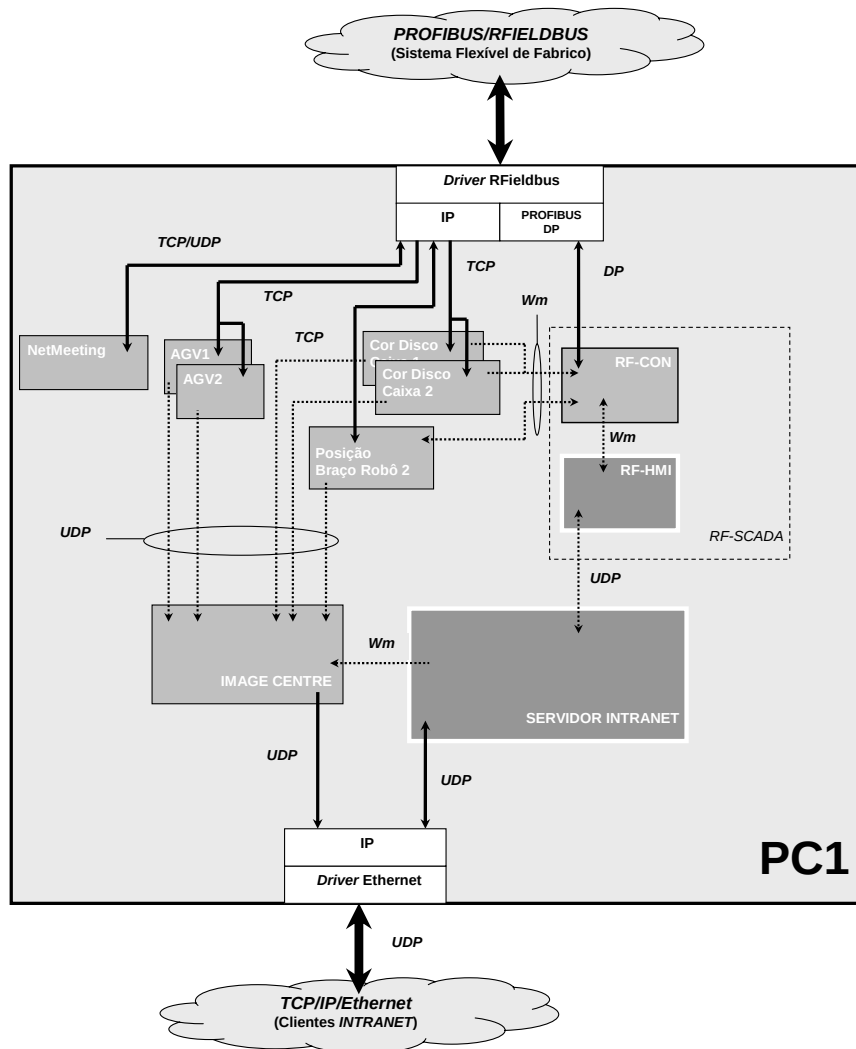


Figura 6.1: Aplicações de software que executam no PC1

De forma a melhor compreender as interações entre as diferentes aplicações envolvidas (quer nos clientes – e em função dos seu tipo), quer no PC1, as secções seguintes irão ser ilustradas com exemplos que incluem os três tipos de clientes: o “normal”, um de funcionalidades limitadas (HMD) e a *Gateway SMS* (quando envolvidos nas interações).

Estas respostas por parte do Servidor *Intranet* são enviadas para a Intranet em *broadcast* (propriedade `RemoteHost` do objecto `WskCLIENTS` = 255.255.255.255), e são processadas pelos Clientes *Intranet* activos que não a *Gateway* SMS ou os HMD/PK1/PK2. Estes ignoram esses pacotes UDP, enquanto que os Clientes *Intranet* “normais” processam essas respostas conforme já descrito na Secção 5.4.3 do Capítulo 5.

O resumo destas interações dos clientes com o PC1, e neste caso particular só com a aplicação Servidor *Intranet*, está sintetizado na Figura 6.4. De notar nesta figura (bem como nas restantes deste capítulo) da utilização do símbolo (Figura 6.3a) para denotar o evento que despoleta a interação ilustrada e do símbolo (Figura 6.3b) para denotar uma acção intermédia executada por um dos componentes de *software* envolvidos na interação descrita.



Figura 6.3: Símbolos utilizados para denotar o início (a) e uma acção intermédia (b) da interação descrita

Como é ilustrado na Figura 6.4, os Clientes *Gateway* SMS e HMD não processam as respostas enviadas pelo servidor. Os componentes RF - HMI, RF - CON e *Image Centre* do PC1 também não estão envolvidos na interação descrita. Nas figuras seguintes deste capítulo, a ausência de componentes específicos do PC1 ou de Clientes *Intranet* específicos significa que eles não estão envolvidos nessa interação em particular.

É isso que se passa com a interação despoletada pelo evento `From_Load` da `BR_Monitor_Form` de um cliente (ver Secção 5.4.1). Quando é seleccionada a esta form da aplicação Cliente *Intranet* (por ex., botão `B_Belts_Rob` da `Monitor_Form`), são enviados seis pedidos UDP relativos aos dispositivos do SFF (ver código na Secção A.6.3).

As respostas a cada um desses seis pedidos são processadas no procedimento `WskCLIENTE_Arrived_DEC_INFO_Request` (ver código na Secção B.2.5) que é evocado pelo evento `WskCLIENTS_DataArrival` do servidor. Essas mensagens são construídas a partir dos valores da variável global `Status_Val(i)` do Servidor *Intranet*, e são enviadas em *broadcast* para a *Intranet* e processadas apenas nos Clientes *Intranet* “normais” (C1 e C2), conforme descrito na Secção 5.4.1 e cujo código está descrito na Secção A.4.3. A Figura 6.5 sintetiza esta interação.

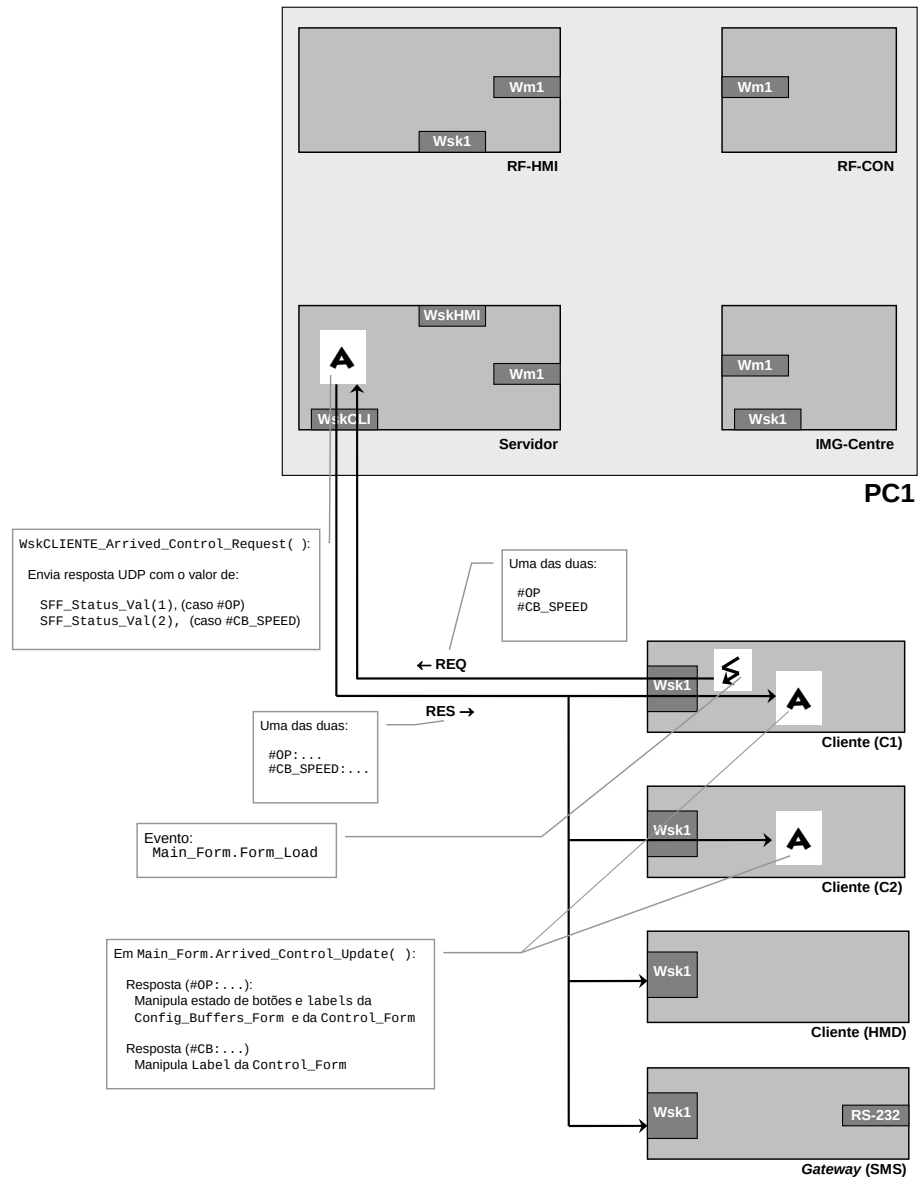


Figura 6.4: Interação despoletada com o evento `Main_Form.Form_Load` de uma aplicação Cliente Intranet

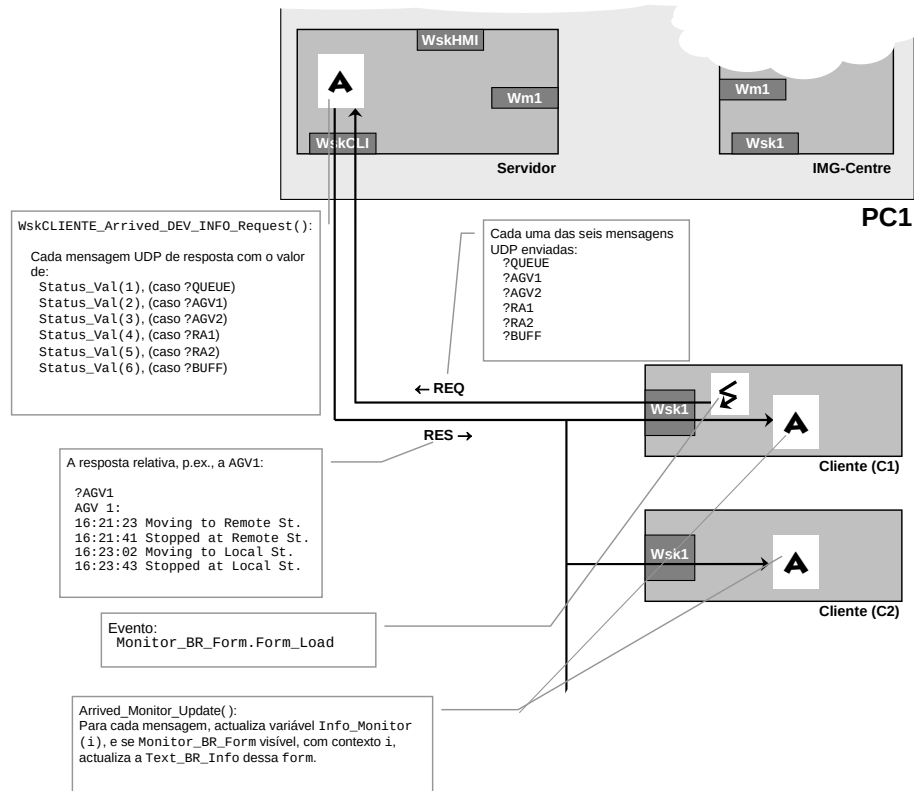


Figura 6.5: Interação despoletada com o evento BR_Monitor_Form.Form_Load de uma aplicação Cliente *Intranet*

Obviamente que as variáveis `Status_Val(i)` (ver a sua definição na Secção B.2.1) do servidor têm de ser periodicamente actualizadas com os valores provenientes do SFF.

A aplicação RF-HMI (descrita no Anexo C desta dissertação) é, como já foi referido no Capítulo 3, essencialmente uma aplicação de interface com o operador. Ela recebe da aplicação RF-COM, via *Windows Messages* (Wm), todas as alterações de estado do SFF. Essas mensagens são descritas na Tabela 6.1. As Wm recebidas na aplicação RF-HMI são processadas no evento `wm1_ReceivedLong` (ver código na Secção C.2.3).

Tabela 6.1: Mensagens enviadas de RF - CON para RF - HMI

Mensagem	Descrição
wmRF_Msg_QUEUE p1, p2	Informação de uma peça que chegou (e de que tipo) ou saiu a/de uma zona delimitada dos tapetes rolantes (<i>queue</i>). O parâmetro p1 identifica a <i>queue</i> . O parâmetro p2 indica o tipo de peça chegada ou a saída de peça da <i>queue</i> .
wmRF_Msg_IR p1, p2	Alteração do valor de um sensor de presença (baseado em infra-vermelhos) associado ao sistema. O parâmetro p1 da mensagem identifica o sensor IR em particular.
wmRF_Msg_CAM p1, p2	Identificação de peça (cor) proveniente das aplicações de detecção de cor associadas às câmaras CAM1 e CAM2. O parâmetro p1 identifica a câmara e o parâmetro p2 corresponde ao resultado da identificação.
wmRF_Msg_BUFF p1, p2	Alteração da quantidade de peças num <i>buffer</i> . O parâmetro p1 identifica o <i>buffer</i> e o parâmetro p2 a quantidade de peças nesse <i>buffer</i> .
wmRF_Msg_BUFF_CNF p1, p2	Alteração do parâmetro correspondente ao tamanho máximo admissível para um <i>buffer</i> . O parâmetro p1 identifica o <i>buffer</i> e p2 o valor novo de configuração.
wmRF_Msg_MM p1, p2	Alteração da velocidade dos tapetes rolantes. O parâmetro p1 identifica o tapete (a aplicação de controlo actual apenas usa uma mesma velocidade para todos os tapetes) e p2 o valor novo de velocidade.
wmRF_Msg_AGV p1, p2	Ação relacionada com os AGVs. O parâmetro p1 identifica o AGV e p2 codifica o movimento.
wmRF_Msg_ROBOT p1, p2	Ação relacionada com os RAs. O parâmetro p1 identifica o RA e p2 codifica o movimento.
wmRF_Msg_OUTP p1, p2	Alteração do estado do SFF. O parâmetro p1 = 0 e p2 codifica o estado.
wmRF_Msg_SA p1, p2	Alteração do estado dos <i>Swivel Arms</i> . O parâmetro p1 identifica o SA e p2 codifica o movimento.
wmRF_Msg_CYL p1, p2	Alteração do estado dos cilindros pneumáticos. O parâmetro p1 identifica o CYL e p2 codifica o movimento.

Das onze mensagens constantes da Tabela 6.1 todas, à excepção de quatro (wmRF_Msg_IR, wmRF_Msg_CAM, wmRF_Msg_SA, wmRF_Msg_CYL), são relevantes para a implementação das funcionalidades descritas para os Clientes *Intranet* no Capítulo 5. Na realidade as duas últimas também o são parcialmente, apesar de não serem utilizadas na presente versão da aplicação Cliente *Intranet* (conforme explicado na Secção 6.5).

Essa informação que é recebida no RF-HMI vinda do RF-CON pode ser facilmente enviada para o Servidor *Intranet* via objecto *Winsock*. Assim, para o efeito, na

aplicação RF-HMI foi acrescentado um objecto deste tipo, o `wsk1` com as propriedades `LocalPort = 1409`, `RemotePort = 1419`, `RemoteHost = 127.0.0.1` e `Protocol = sckUDPProtocol` (ver Secção C.2.2). Este objecto implementa um canal de comunicação UDP com a aplicação Servidor *Intranet*, na qual foi também incluído um objecto do tipo Winsock, o `WskHMI` (Secções B.1.2 e B2.2).

O código (aparece a sombreado no Anexo C) possibilita a interação entre o SFF e os Clientes *Intranet* via Servidor *Intranet*. No essencial, a alguns dos procedimentos que são evocados pelo evento `wm1_ReceivedLong` para processar as sete mensagens descritas na Tabela 6.1, foram acrescentados envios (`wsk1.SendData`) da informação recebida (feita a conversão adequada dos parâmetros das mensagens *Windows* recebidas. Estão nesse caso os procedimentos de processamento das mensagens `wmRF_Msg_BUFF` (só para o caso de *buffer full* – nesse caso é enviado um alarme/evento), `wmRF_Msg_BUFF_CNF`, `wmRF_Msg_MM` e `wmRF_Msg_OUTP`.

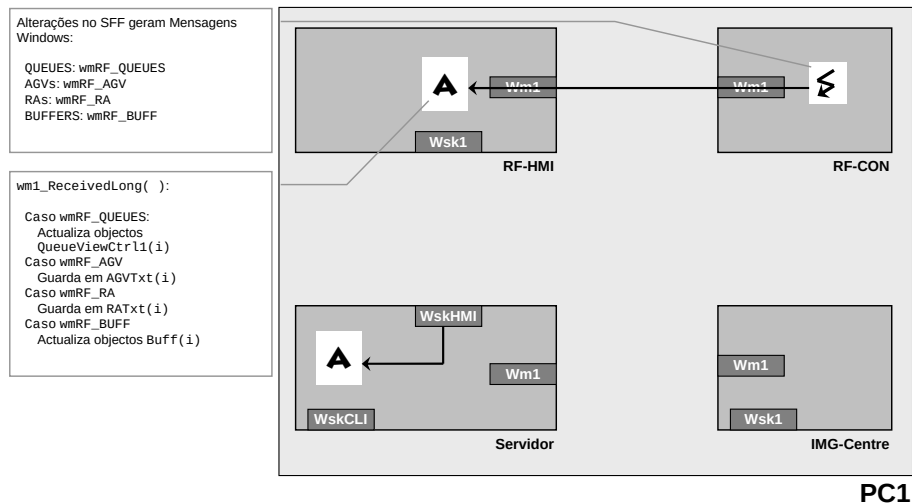


Figura 6.6: Armazenamento na aplicação RF-HMI da informação contida em algumas mensagens *Windows* recebidas de RF-CON

Já a informação recebida do RF-CON nas mensagens `wmRF_Msg_QUEUE`, `wmRF_Msg_BUFF`, `wmRF_Msg_AGV` e `wmRF_Msg_ROBOT` é guardada nos objectos `QueueViewCtrl1(i)`, `Buff(i)` e nas variáveis globais `AGVTxt(i)` e `RATxt(i)`. Estas duas últimas, relativas aos AGVs e RAs, vão conter os últimos quatro movimentos (ver o código dos procedimentos `SaveAGV` e `SaveRA` na Secção C.2.3). A Figura 6.6 ilustra este cenário.

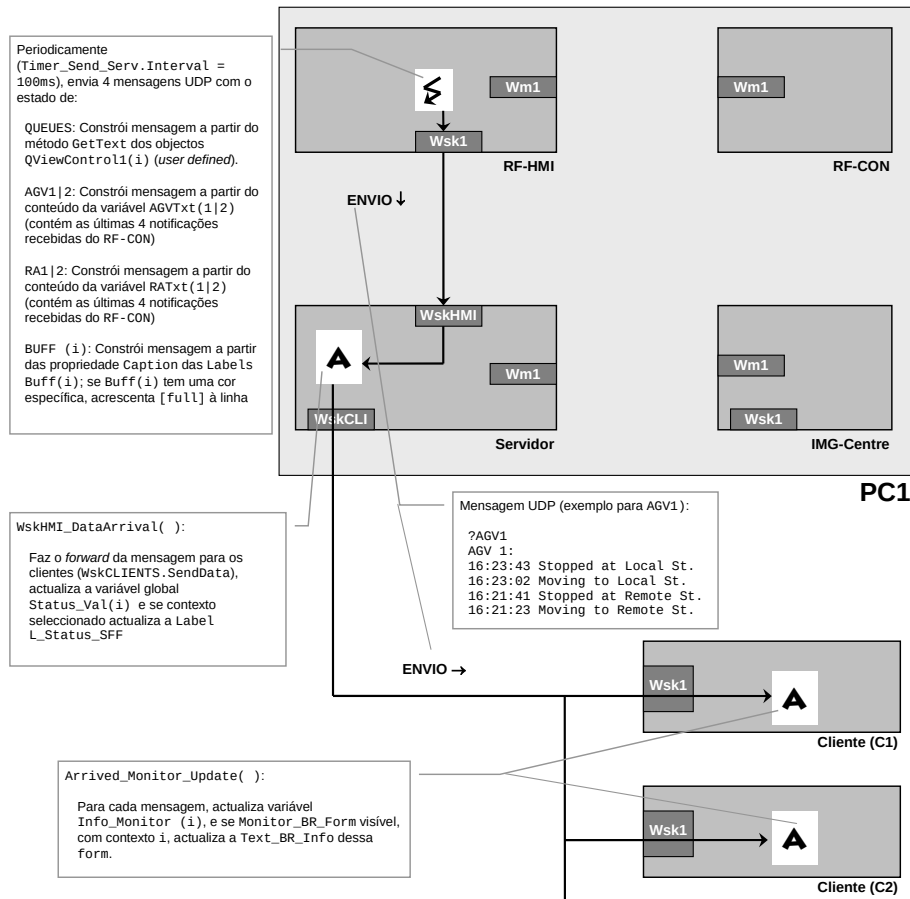


Figura 6.7: Envio periódico (Timer_Send_Serv_Timer) aos Clientes Intranet de informação relativa aos Buffers, às Queues, aos AGVs e aos RAs

Adicionalmente, foi considerado no RF-HMI o objecto Timer Timer_Send_Serv, em cujo evento Timer_Send_Serv_Timer (programado com um intervalo de 100ms), e cujo código é indicado na Secção C.2.5, são enviadas mensagens UDP com a informação das *queues* (mensagem construída a partir dos objectos QueueViewCtrl1(i)), dos buffers (mensagem construída a partir dos objectos Buff(i)) e variáveis AGVTxt(i) e RATxt(i). Todas estas mensagens incluem informação temporal (é utilizada a função TimeStamp() – ver Secção C.2.7). A Figura 6.7 ilustra estas interações. Estas mensagens são processadas no Servidor Intranet pelo procedimento WskHMI_Update_Server_dev_status, que é evocado pelo evento WskHMI_DataArrival (ver código na Secção B.2.4). A variável global Status_Val(i) da aplicação Servidor é actualizada em função da mensagem específica recebida do RF-HMI, e é feito o *broadcast* da mensagem para os

Clientes *Intranet* (`WskCLIENTS.SendData`). Os Clientes “normais” processam essas mensagens conforme já descrito na Secção 5.4.1

Relativamente às mensagens *Windows* `wmRF_Msg_OUTP` e `wmRF_Msg_MM` (ver Tabela 6.1), como já referido atrás, elas, quando recebidas pelo RF-HMI, são logo enviadas para o Servidor *Intranet* (Figura 6.8). Optou-se por não fazer o mesmo com as mensagens *Windows* descritas na Figura 6.6 (a alternativa, eventualmente desvantajosa – devido à sobrecarga de mensagens *Windows*, seria fazer o armazenamento destes dados no servidor em vez de no RF - HMI).

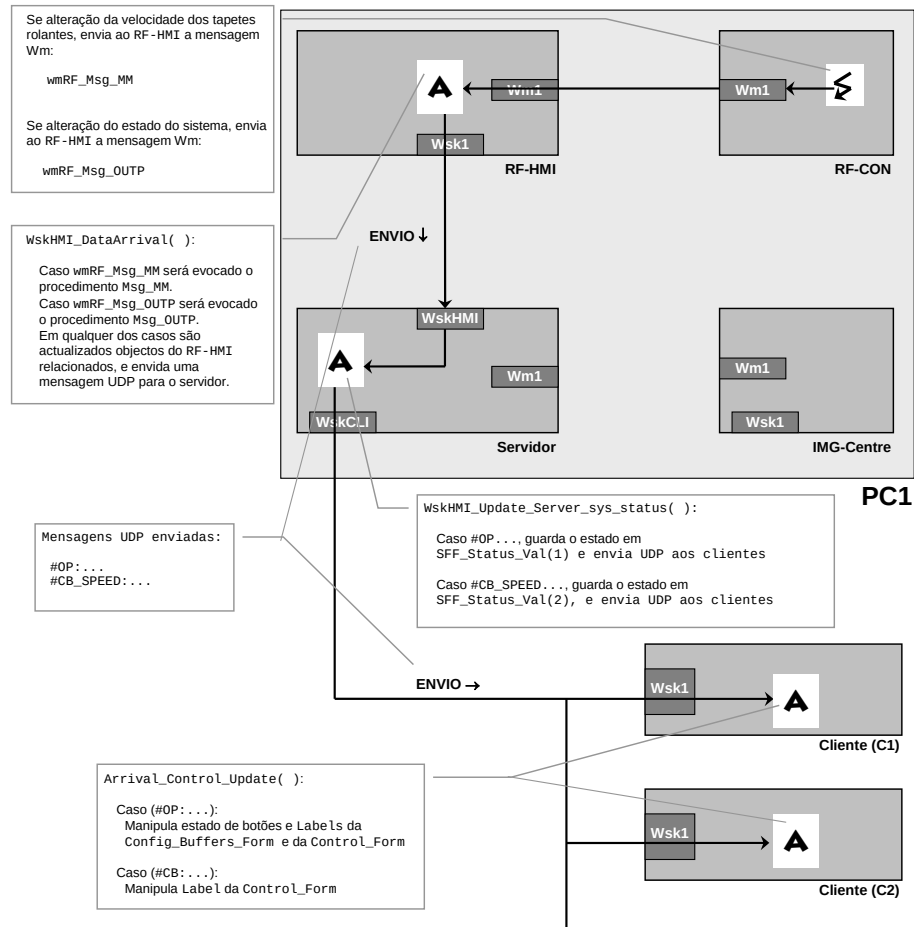


Figura 6.8: Interações resultantes do envio de mensagens `wmRF_OUTP` e `wmRF_MM` do RF - CON para o RF - HMI

6.3.2. Monitorização do SFF Através das Câmaras

Como foi descrito na Secção 5.4.2, os Clientes *Intranet* podem monitorizar os cinco *video streams* gerados pelas câmaras associadas ao SFF. O Servidor *Intranet* tem, relativamente a esta funcionalidade, a função de activação/desactivação da distribuição desses fluxos de imagens por parte da aplicação *Image Centre*.

Quando um Cliente *Intranet* “normal” subscreve um determinado fluxo de vídeo (evento *B_CM_ON_Click* na Secção A.9.3), é simplesmente enviada uma mensagem UDP para o Servidor *Intranet* correspondente a um pedido de subscrição de uma câmara específica (Figura 6.9).

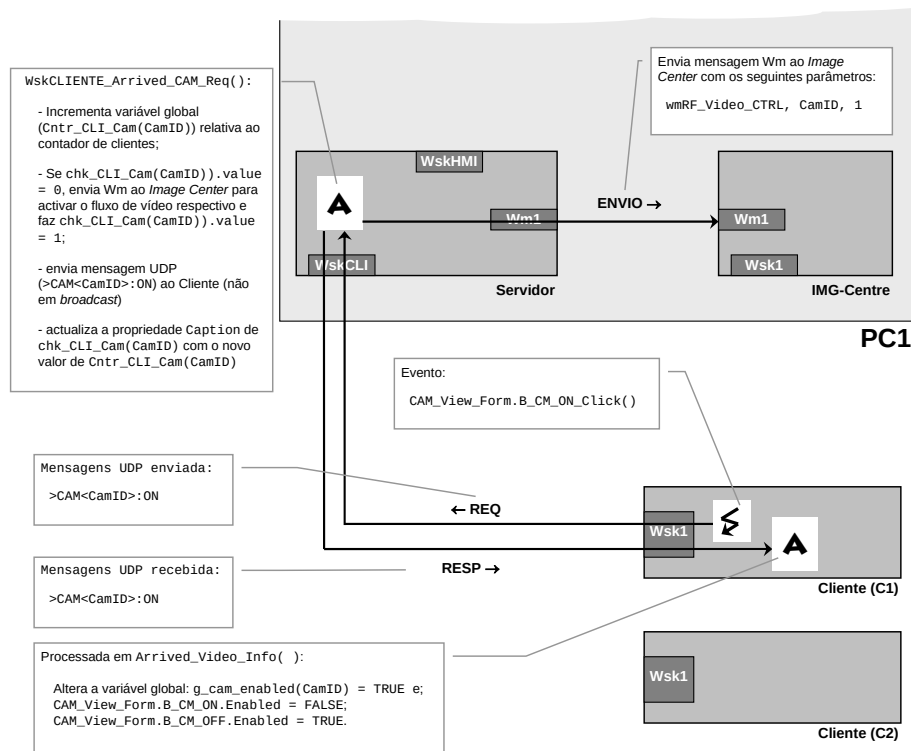


Figura 6.9: Interação relativa ao pedido de um Cliente *Intranet* para subscrever um determinado fluxo de vídeo de uma câmara específica do SFF (caso em que esse fluxo de vídeo não tem ainda nenhuma subscrição)

Esse pedido é processado pelo Servidor *Intranet* (procedimento *WskCLIENTE_Arrived_CAM_Request* que é evocado no evento *WskCLIENTS_DataArrival* – Secção B.2.5). É incrementada a variável global *Cntr_CLI_Cam(CamID)* e se a propriedade *Value* do objecto *chk_CLI_Cam(CamID)* tem o valor 0, é enviada ao *Image Centre* a mensagem *Windows wmRF_Video_CTRL* com os parâmetros *p1 = CamID* e *p2 = 1*, para que seja activada a difusão UDP do fluxo vídeo relativo a *CamID*. É alterado para 1 o valor

e alterada `chk_CLI_Cam(CamID).Caption` da `Server_Form` para reflectir o novo número de subscrições do fluxo de vídeo) “0 Clients”). Não é enviada nenhuma mensagem UDP aos Clientes. Esta interacção é descrita na Figura 6.11.

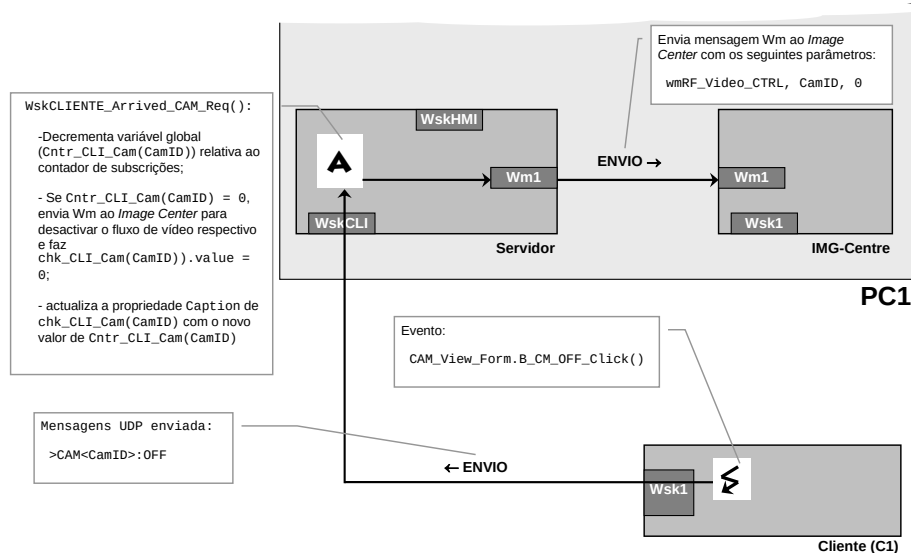


Figura 6.11: Interação relativa ao pedido de um Cliente *Intranet* para indicar o fim de subscrição (caso em que esse fluxo de vídeo passa a não ter subscrições)

No caso de o cancelamento de subscrição por parte de um Cliente *Intranet* não conduzir a um valor de 0 subscrições, tem lugar simplesmente a actualização adequada dos objectos e variáveis da `Server_Form` (Figura 6.12).

De notar que há algumas funcionalidades que não são implementadas na versão descrita nesta dissertação. Por exemplo, não existe uma confirmação de activação do fluxo por parte do IMG-CENTRE a condicionar o envio da confirmação ao Cliente *Intranet* (Figura 6.9). Por outro lado, a gestão dos clientes activos não inclui as subscrições (feitas por interpostos clientes) por parte dos clientes de funcionalidades reduzidas HMD/PK1/PK2 (ver Capítulo 5). Permissões de acesso a câmaras por parte de Clientes *Intranet* também não estão implementadas.

De notar que no Servidor *Intranet* é possível desactivar a difusão de um fluxo de vídeo (evento `chk_CLI_Cam_Click()`, cujo código está descrito na Secção B.2.7). Nesse caso, é enviada a mensagem UDP `>CAM<CamID>:OFF` em *broadcast* a todos os Clientes *Intranet* e é enviada a mensagem Windows `wmRF_Video_CTRL (CamID, 0)` ao IMG-CENTRE. O processamento desta mensagem nos Clientes *Intranet* já foi descrito na Secção 5.4.2 (ver código específico no procedimento `Arrived_Video_Info()` na Secção A.4.3). A Figura 6.13 sintetiza as interacções associadas.

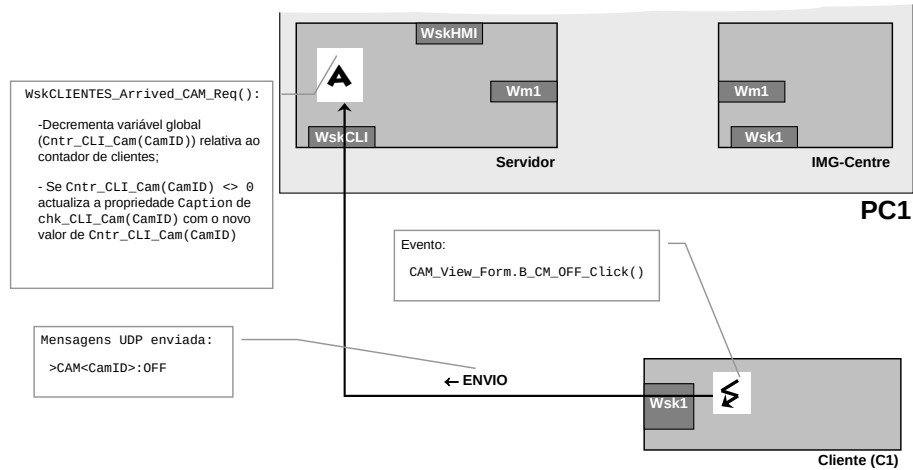


Figura 6.12: Interação relativa ao pedido de um Cliente *Intranet* para indicar o fim de subscrição (caso em que esse fluxo de vídeo continua a ter outras subscrições)

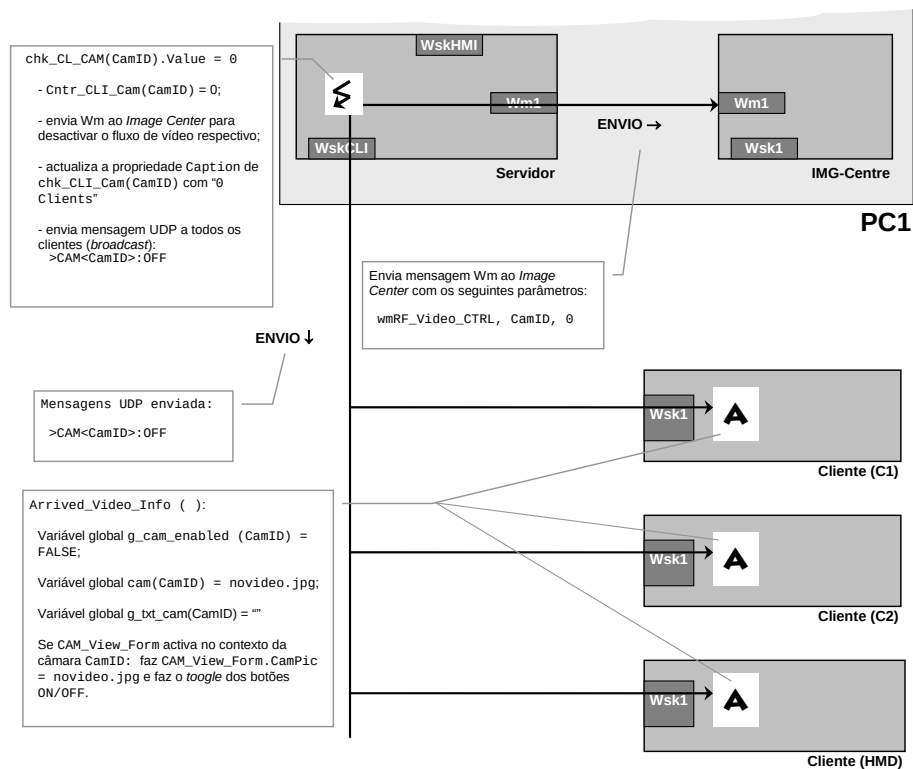


Figura 6.13: Acção de desactivação de um fluxo de vídeo (chk_CLI_Cam_Click()) utilizando o interface do Servidor *Intranet*

Finalmente, e ainda relativamente às câmaras associadas ao SFF, sempre que o IMG-CENTRE recebe uma nova imagem relativa a uma câmara, e esse fluxo de vídeo está subscrito por pelo menos um cliente, o IMG-CENTRE envia *broadcasts* UDP com fragmentos de imagem (seguindo o formato descrito na Secção 5.4.2). Os clientes que têm essa câmara subscrita processam esses fragmentos (Figura 6.14).

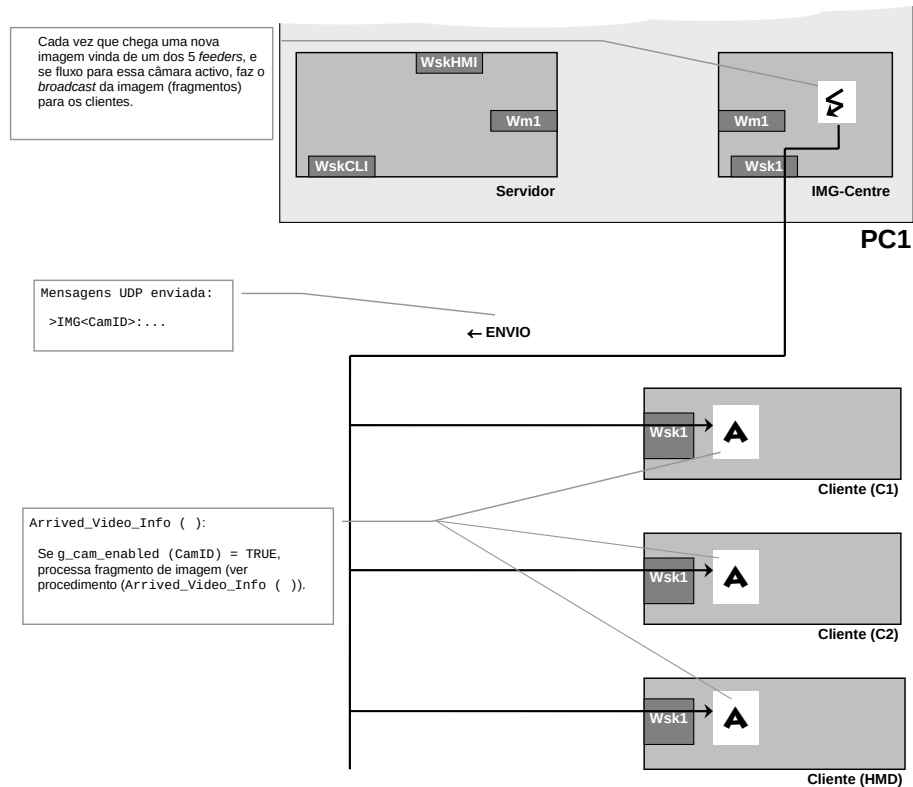


Figura 6.14: Envio de fragmentos (mensagens UDP) de imagens por parte de IMG - CENTRE

6.4. Suporte à Configuração do SFF e de Clientes de Output

6.4.1. Configuração de Parâmetros do SFF

Como foi referido (Secção 5.5.1), um Cliente *Intranet* “normal” pode solicitar a alteração do tamanho máximo de quatro dos *buffers* associados ao SFF. Pode fazê-lo se o SFF estiver no estado “STOP” (Figura 6.8).

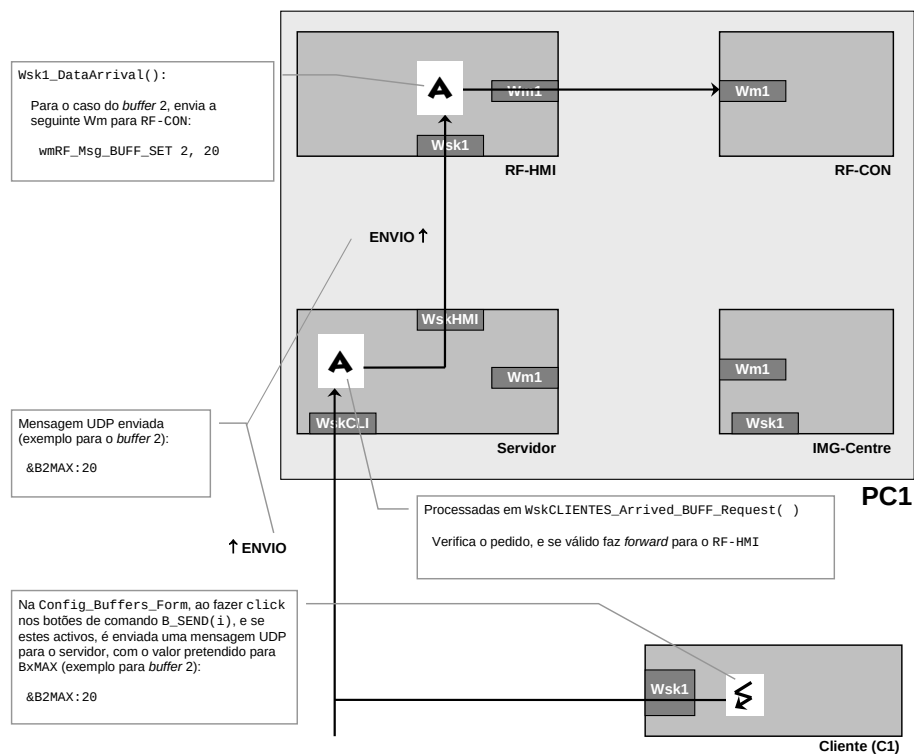


Figura 6.15: Pedido de alteração do tamanho máximo feito pelo Cliente C1

Quando o Servidor *Intranet* recebe um pedido de alteração do tamanho máximo de um *buffer* (Figura 6.15 – para o exemplo relativo a uma alteração de B2), este pedido vai ser processado pelo procedimento **WskCLIENTES_Arrived_BUFF_Request()** e faz **envio** para a aplicação **RF-HMI** (**WskHMI.SendData**) (ver código na Secção B.2.5). Por sua vez, na aplicação **RF-HMI**, este pacote UDP (evento **wsk1_DataArrival**) será processada: é enviada uma mensagem *Windows* (via evento **cmdBuffSet_Click()**) para o **RF-CON** com a solicitação de alteração (ver

código em Secção C.2.4). A mensagem *Windows* é para o exemplo retratado na Figura 6.15 a seguinte `wmRFMsg_BUFF_SET`, `BufID`, `BufVal(BufID)`.

O RF-CON tem a responsabilidade de notificar o RF-HMI da alteração da configuração de um *buffer*. Quando isso acontece, é enviada uma mensagem *Windows* (a `wmRF_Msg_BUFF_CNF` (para o exemplo da Figura 6.16, com os parâmetros 2, 20).

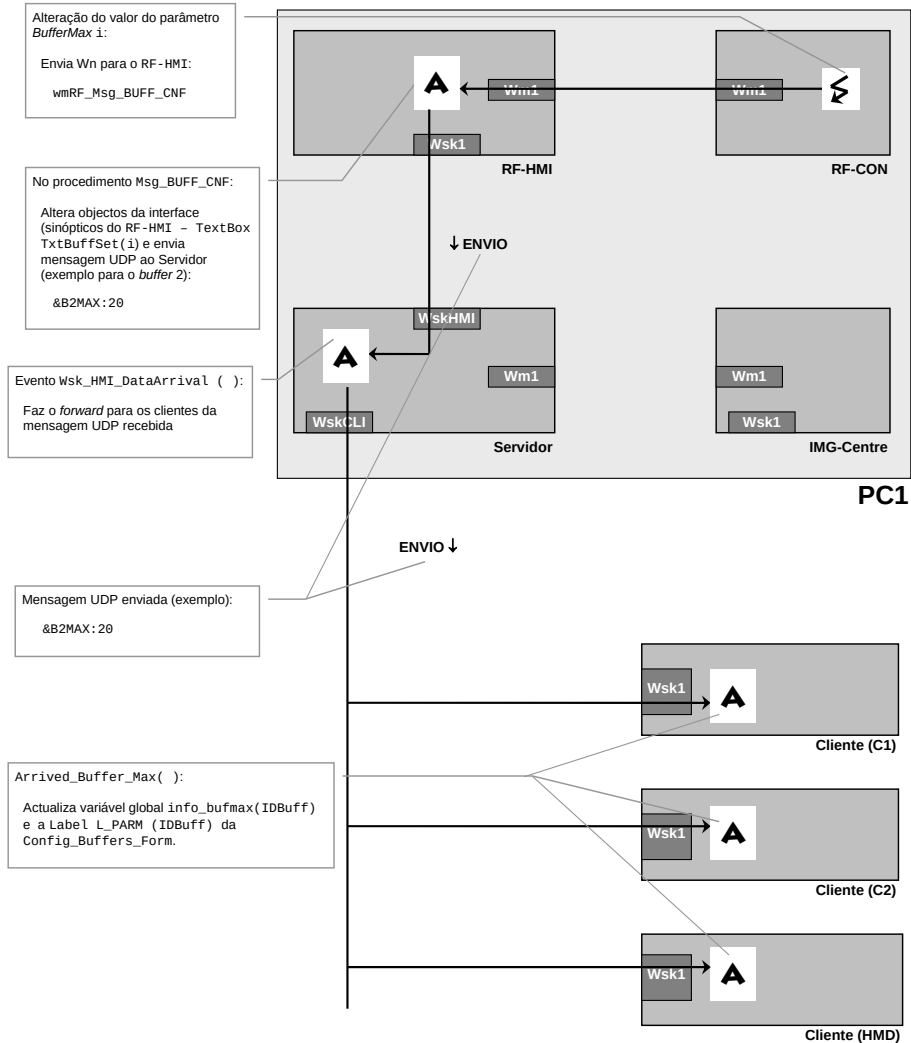


Figura 6.16: Interações resultantes da mensagem `wmRF_BUFF_CNF` enviada pelo RF-CON ao RF-HMI

Na aplicação RF-HMI, esta mensagem é processada (ver código na Secção C.2.3) pelo procedimento `Msg_BUFF_CNF` (evocado pelo evento `wm1_ReceivedLong`). É

feito o envio para a aplicação Servidor Intranet (para o exemplo `wsk1.SendData "&B2MAX:20"` e actualizada a `TextBox txtBuffSet(2)` da RF-HMI. No servidor, esta mensagem UDP (processada no evento `WskHMI_DataArrival` – ver código na Secção B.2.4) é reenviada (*broadcast*) aos Clientes *Intranet*, que por sua vez a vão processar conforme descrito na Secção 5.5.1.

6.4.2. Configuração de Clientes de Output e da Gateway SMS

Os Clientes *Intranet* “normais” e o Servidor *Intranet* podem configurar os Clientes *Intranet* de funcionalidades reduzidas (HMD/PK1/PK2) e a *Gateway* SMS. O dessas funcionalidades já foi descrito, para a aplicação Cliente, na Secção 5.5.2. Um Cliente *Intranet* envia a configuração de um desses dispositivos via duas mensagens UDP para todos os Clientes e para o Servidor *Intranet*. As Figuras 6.17 e 6.18 ilustram as interações envolvidas para o caso da configuração do Cliente HMD e da *Gateway* SMS, respectivamente, em ambos os casos enviadas pelo Cliente C1.

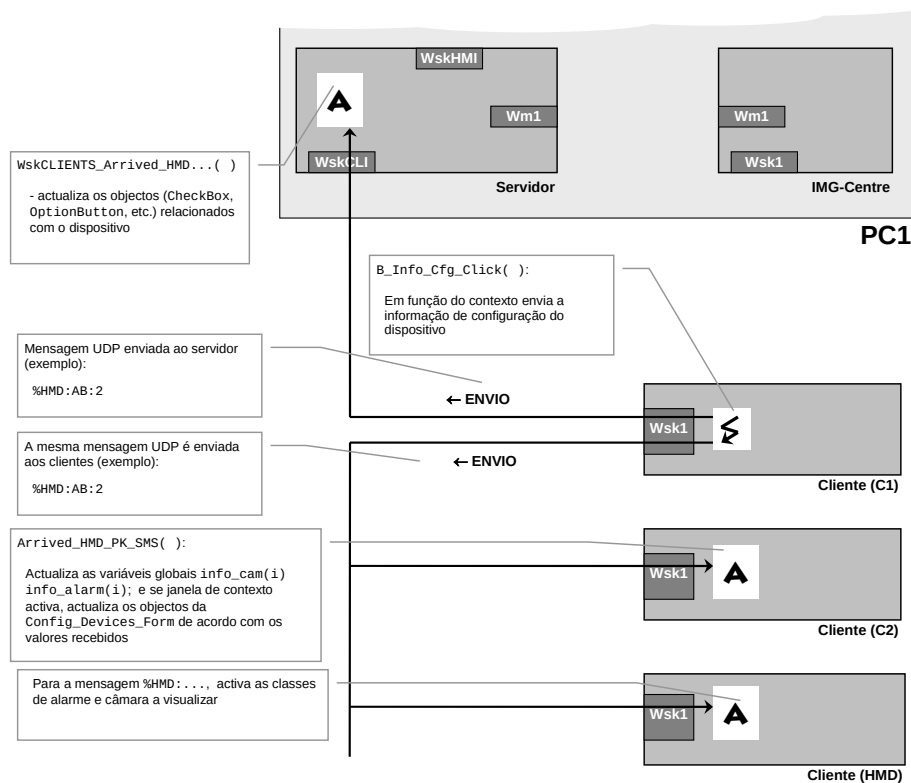


Figura 6.17: Exemplo de configuração do Cliente HMD (enviada pelo Cliente C1)

No caso da configuração do Cliente HMD (Figura 6.17), quando o HMD recebe o pacote UDP de configuração, utiliza-o para obter as classes de alarme que deverá

processar e a câmara para a qual deve processar os pacotes UDP to tipo “>IMG:...” receber. Como já foi dito, esta interação não está nesta versão integrada com a gestão de subscritores de fluxos vídeo feita pelo Servidor *Intranet* (ver comentário feito na Secção 6.4.2).

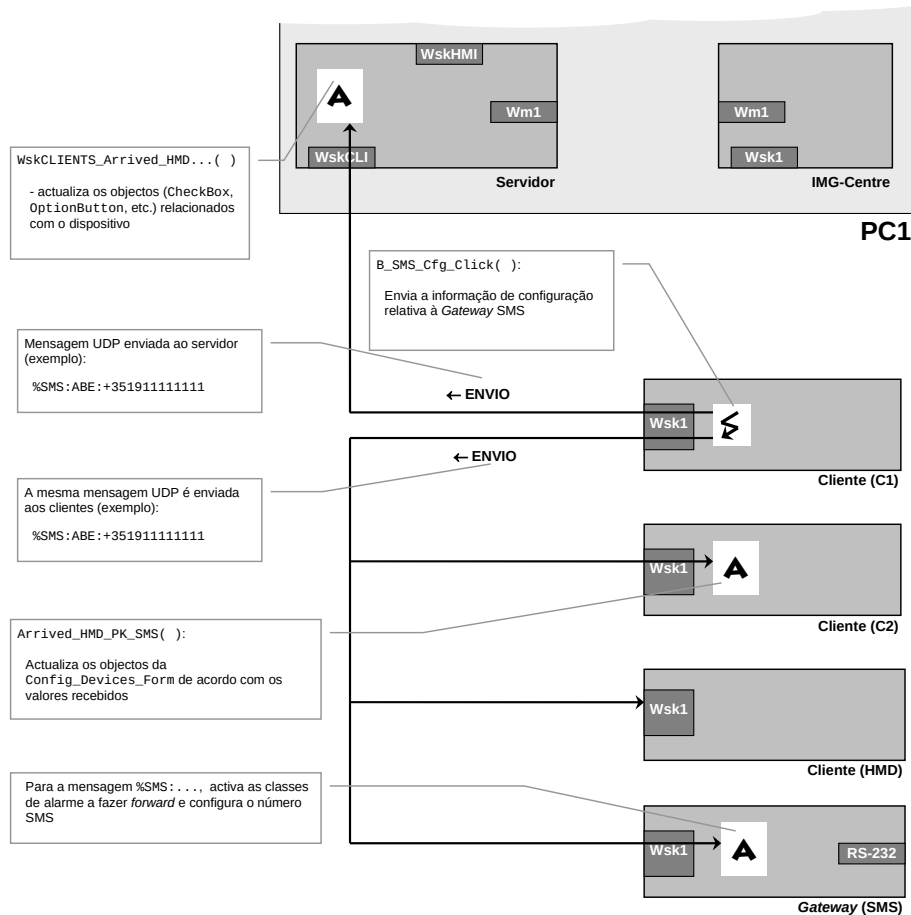


Figura 6.18: Exemplo de configuração da Gateway SMS (enviada pelo Cliente C1)

No caso da configuração da *Gateway SMS* (Figura 6.18), quando esta recebe o pacote UDP de configuração, utiliza-o para configurar o processamento de classes de alarmes/eventos difundidos na *Intranet*. De notar que a aplicação Servidor *Intranet* contém os objectos (ver Secção B.1) e o código (ver Secção B.2.5, procedimento `WskCLIENTE_Arrived_HMD_PK_SMS_Config`) necessário a manter uma réplica consistente com todos os outros Clientes da *Intranet*.

No Servidor *Intranet* também é possível configurar estes clientes especiais. As Figura 6.19 e 6.20 ilustram os casos da configuração do HMD e da *Gateway SMS*, respectivamente (ver código do Servidor *Intranet* nas Secções B.2.9 e B.2.10).

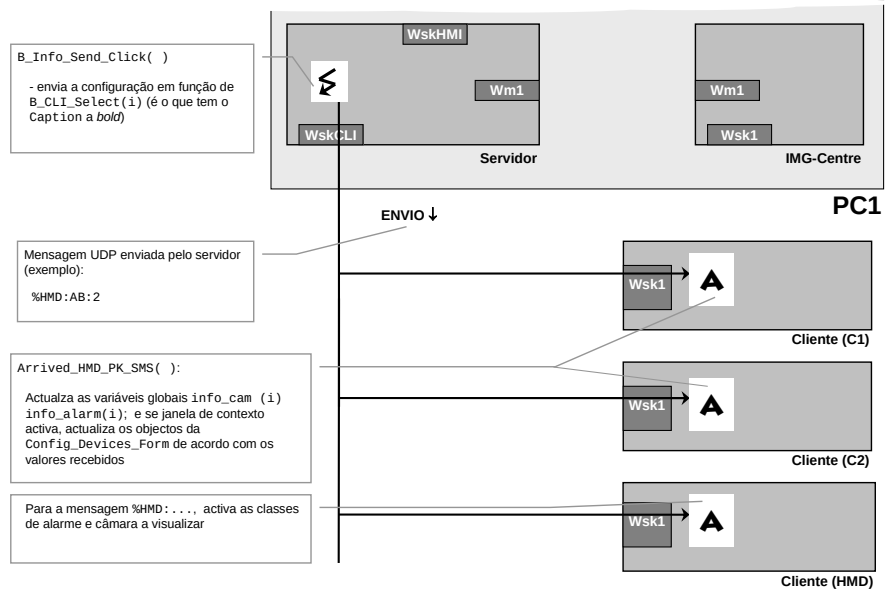


Figura 6.19: Exemplo de configuração do Cliente HMD (enviada pelo Servidor *Intranet*)

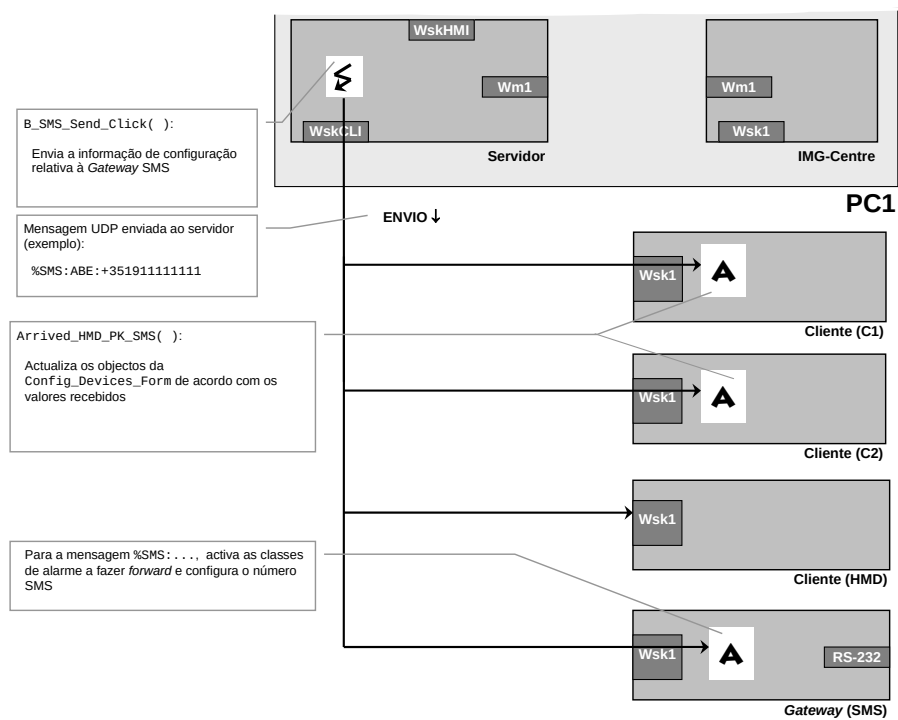


Figura 6.20: Exemplo de configuração da *Gateway SMS* (enviada pelo Servidor *Intranet*)

6.5. Suporte ao Comando de Dispositivos do SFF

Conforme foi descrito na Secção 5.6, os Clientes *Intranet* “normais” podem fazer operações remotas no SFF: pará-lo (comando `#EM_STOP`) alterar a velocidade dos tapetes rolantes (comandos `#CB_SPEED:SLOWER` e `#CB_SPEED:FASTER`) e accionar os *Swivel Arms* (comandos `#Sax:ACTIVATE`) ou os Cilindros Pneumáticos (comandos `CYLx:ACTIVATE`). Estes dois últimos só se o SFF estiver no estado STOP.

No Servidor *Intranet*, estes comandos são processados no procedimento `WskCLIENTE_Arrived_Control_Request` que é evocado no evento `WskCLIENTS_DataArrival` (ver código na Secção B.2.5). É simplesmente verificada a validade dos comandos (como já foi referido, um possível melhoramento da aplicação seria a verificação de permissão dos clientes), e são enviados via UDP para o RF-HMI (Figura 6.21).

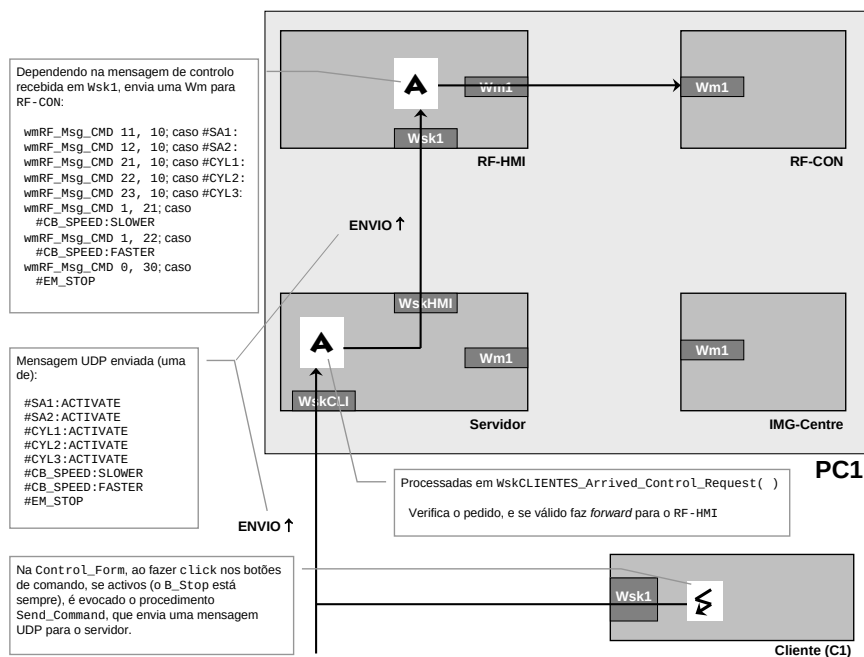


Figura 6.21: Interações resultantes do envio de comandos ao SFF

No RF-HMI, as mensagens UDP são processadas no evento `wsk1_DataArrival` (ver código na Secção C.2.4), e consoante a mensagem recebida, é enviada uma mensagem Windows à aplicação RF-CON (conforme Figura 6.21).

Na versão descrita nesta dissertação, não é dado *feedback* aos clientes sobre esta actuação. Esta situação é ilustrada na Figura 6.22.

Tipicamente esses eventos serão de uma classe que não a classe A (reservada para eventos/alarmes gerados pelo SFF). Se a classe de alarme estiver configurada nos Clientes de funcionalidades reduzidas (por exemplo o Cliente HMD), esse alarme é visualizado no visor durante alguns segundos. Se a classe de alarme está configurada na *Gateway SMS*, é feito o reenvio dessa mensagem via SMS para o número de telemóvel também previamente configurado na *Gateway SMS*. Estas interações não envolvem o PC1 (Figura 6.23). Também é possível gerar o mesmo tipo de alarmes/eventos no Servidor *Intranet* (Secção B.2.8), caso se opte por não ter uma aplicação Cliente *Intranet* a executar no PC1. Nesse caso, as interações resultantes são as que se indicam na Figura 6.24.

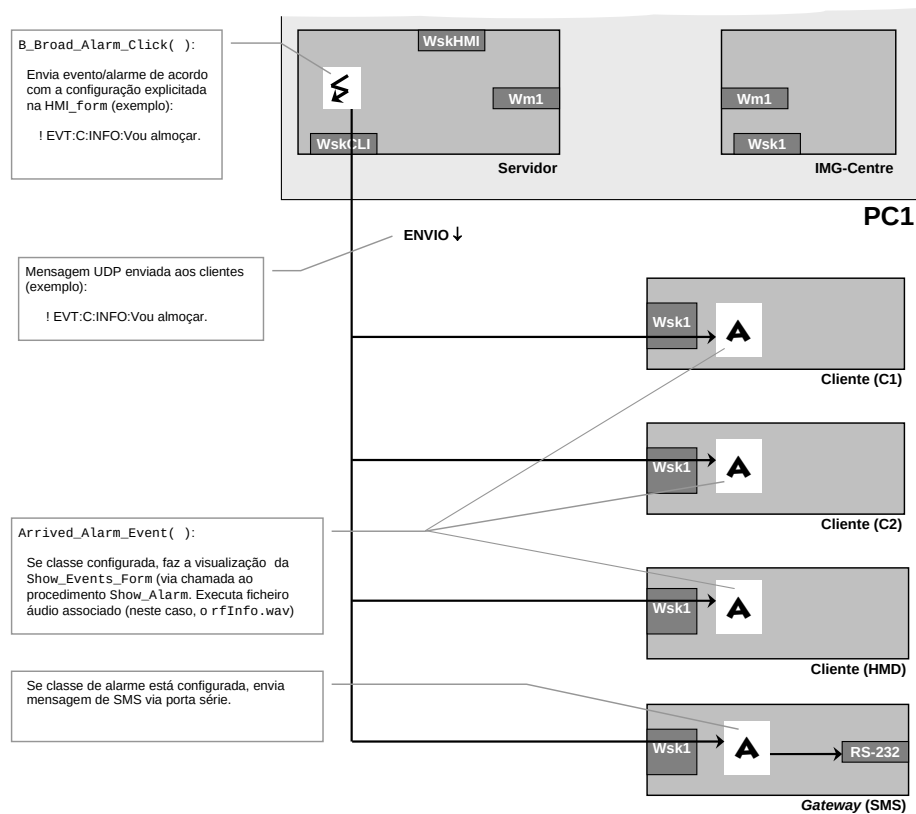


Figura 6.24: Interações no caso da geração de um alarme/evento no Servidor *Intranet*

Finalmente, de referir o único evento do SFF mapeado para os Clientes *Intranet* (Figura 6.25). Quando o RF-HMI recebe do RF-CON uma mensagem `Windows wmRF_Msg_BUFF` (ver código do procedimento `Msg_BUFF` na Secção C.2.3), se o *buffer* associado à mensagem passa a estar no estado *full*, é enviada uma mensagem UDP de evento/alarme (conforme se exemplifica na Figura 6.25) para o Servidor *Intranet*.

Capítulo 7

Conclusões

O principal objectivo desta dissertação foi o de demonstrar a exequibilidade do desenvolvimento e da utilização de aplicações multimédia de uso genérico em sistemas de informação industriais associados à automatização de processos de fabrico. Este capítulo sumaria as contribuições desta dissertação e perspectiva trabalhos na sua sequência.

7.1. Revisão dos Objectivos e Contribuições da Dissertação

Os sistemas de automação fabril utilizam, tipicamente, arquitecturas distribuídas de computação. São, por isso, sistemas distribuídos de controlo computadorizado. Neste tipo de sistemas, diversos módulos de *software* e *hardware* cooperam com vista a um objectivo comum. A abordagem usual é suportar essa cooperação por recurso a uma infra-estrutura de comunicações especialmente vocacionada para o cumprimento dos requisitos específicos dos sistemas de fabrico automatizado. Este tipo de redes de comunicação industrial é usualmente conhecido rede de campo (*fieldbus*).

As redes do tipo *fieldbus* são redes de comunicação industriais capazes de satisfazer, entre outros, os requisitos de correcto funcionamento temporal exigido pelas aplicações distribuídas. Permitem, por isso, um suporte adequado a sistemas distribuídos de tempo-real. Note-se que num sistema computacional de tempo-real, o correcto funcionamento não depende só da correcção lógica do resultado da computação, mas também do instante de tempo no qual o resultado é produzido.

As redes do tipo *fieldbus* têm outras especificidades, que vão desde as características físicas da cablagem e da transmissão de dados, até aos serviços específicos da camada de aplicação, passando pela forma como são concretizadas em termos de plataformas computacionais (normalmente os nós são sistemas computacionais embebidos, relativamente complexos, e com interfaces de sensorização e actuação).

Existem inúmeras soluções comerciais de redes do tipo *fieldbus*, quase todas baseadas em normas internacionais de especificação independentes dos fabricantes (normas abertas). Uma das soluções comerciais mais importantes, quer pela sua penetração no mercado, quer pelo número de empresas que a suportam, baseia-se na norma internacional PROFIBUS (acrónimo de PROcess FIeld BUS).

Nas décadas de 80 e 90, o advento das redes do tipo *fieldbus* quebrou com a tradicional filosofia de controlo centralizado e comunicações ponto-a-ponto vigente até então nos sistemas de automação de fabrico. Nos anos mais recentes, as pressões do mercado e a crescente disponibilidade de técnicas e tecnologias relacionadas com as

modernas tecnologias de informação e comunicação, têm levado os fabricantes a fazer evoluir as soluções existentes de forma a torná-las compatíveis com uma realidade mais ubíqua e interactiva do ambiente fabril. Para isso, duas funcionalidades cruciais (adicionais) são essenciais às redes do tipo *fieldbus*.

1. Suporte de mobilidade de dispositivos, e por isso a necessidade de integrar capacidades de comunicações sem fios (*wireless*) adequadas ao ambiente industrial.
2. Suporte de tráfego multimédia, e por isso a eventual necessidade de integrar protocolos e funcionalidades normalmente encontrados noutro tipo de tecnologias de comunicação, como é o caso do conjunto de protocolos associados à *Internet*.

O requisito *wireless* é intuitivamente óbvio. O requisito de compatibilidade com os protocolos associados à *Internet* resulta do facto de aplicações como monitorização associada a câmaras e microfones digitais, de acesso remoto a bases de dados de manutenção e de guias de operações de fabrico (incluindo gráficos, vídeos e som), entre outras, serem aplicações largamente difundidas e suportadas pela pilha de protocolos de comunicações TCP/IP (*Transmission Control Protocol / Internet Protocol*), respectivamente protocolos de camada de transporte e de camada de rede. Tipicamente, as redes de *fieldbus* não especificam nem suportam protocolos de camada de rede (*Network Layer* – NL) nem de camada de transporte (*Transport Layer* – TL). Daí os esforços no sentido de incorporar mecanismos de suporte a comunicações sem fios, e no sentido de incorporar o suporte a tráfego e aplicações TCP/IP.

O Projecto Europeu IST-1999-11316 RFieldbus (*High Performance Wireless Fieldbus in Industrial-Related Multimedia Environment*) teve como objectivo essencial a de extensão da norma PROFIBUS de forma a ser possível suportar nós móveis (via comunicações rádio) e tráfego multimédia (via encapsulamento de fragmentos IP em tramas PROFIBUS).

Nesta dissertação, os aspectos essenciais associados à tecnologia RFieldbus foram descritos no Capítulo 2, permitindo assim dar o enquadramento tecnológico e computacional à aplicação industrial descrita no Capítulo 3.

O objectivo desta dissertação foi o de ilustrar e concretizar o desenvolvimento de aplicações de *software* (Capítulos 5 e 6), tentando demonstrar, por essa via, a hipótese de que, com estas novas tecnologias de informação e comunicação industrial seria viável uma perspectiva diferente da forma como os operários fabris podem interagir com as máquinas e com os sistemas de informação ao nível do controlo de operações de fabrico.

Essas aplicações, descritas em detalhe no Capítulo 5, permitem uma série de funcionalidades e de interacções multimédia, que são obviamente facilitadas (por exemplo no que concerne a imagens vídeo) pelo facto de a tecnologias RFieldbus permitir fluxos horizontais de comunicação TCP/IP. A interoperação entre as aplicações Cliente *Intranet* e o sistema industrial é suportada em diversas aplicações de “interface”: o Servidor *Intranet*, o *Image Centre* e código específico adicionado na aplicação RF-HMI para sustentar o fluxo de comunicações verticais (entre o sistema industrial e os Clientes *Intranet*). Estas aplicações de interface foram descritas em detalhe no Capítulo 6 desta dissertação.

7.2. Trabalho Futuro

Em termos genéricos, e dada a complexidade e heterogeneidade das funcionalidades, aplicações e tecnologias descritas no âmbito desta dissertação, quase que se podia dizer que todas as aplicações poderiam der repensadas e reestruturadas.

Por exemplo, seria obvio agora o reequacionar do desenvolvimento das aplicações descritas nos Capítulos 5 e 6 utilizando a *framework .Net*. Preferimos aqui limitar esta secção a alguns dos melhoramentos possíveis e já referidos no Capítulo 6:

- a confirmação por parte da aplicação IMG-CENTRE da activação de fluxos de vídeo (ver Secção 6.3.2);
- a gestão de fluxos de vídeo activos considerando as subscrições efectuadas por clientes relativas aos clientes de *output* (ver Secção 6.3.2);
- a gestão de permissões e níveis de restrição de execução de funcionalidades para clientes (limitações referidas nas Secções 6.3.2 e 6.5);
- a implementação de *feedback* por parte do RF-SCADA (HMI/CON) relativa a algumas funcionalidades de comando por parte do clientes (limitações referidas na Secção 6.5);
- a implementação no fluxo vertical de comunicações de outros eventos/alarmes despoletados ao nível do SFF, e não só o de *buffer* cheio (limitação referida na Secção 6.6).

Referências

- Alves, M., Tovar, E. e Oliveira, J. (1998). "Towards a More Flexible Transportation System". Actas da 8th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM'98), Portland, EUA, pp. 783-792.
- Alves, M. (2003). "Real-Time Communications over Hybrid Wired/Wireless PROFIBUS-Based Networks". Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, Fevereiro. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=152.
- Alves, M., Tovar, E. e Vasques, F. (2001a). "On the Adaptation of Broadcast Transactions in Token-Passing Fieldbus Networks with Heterogeneous Transmission Media". Actas da 4th IFAC International Conference on Fieldbus Systems and Their Applications (FET'2001), pp. 278-284. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=125.
- Alves, M., Brandão, V., Tovar, E., Pacheco, F. and L. M. Pinho (2001b). "Specification of the Manufacturing Automation Field Trial". Relatório Técnico IPP-HURRAY (HURRAY-BTR-0131), Dezembro. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=154.
- Alves, M., Tovar, E., Vasques, F., Hammer, G. e K. Roether (2002). "Real-Time Communications over Hybrid Wired/Wireless PROFIBUS-based Networks". Actas da 14th Euromicro Conference on Real-Time Systems - ECRTS'02, pp. 142-150. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=134.
- Balena, F., (1999). "Programming Microsoft Visual Basic 6.0". Microsoft Press.
- Cardoso, A. e Tovar, E. (1994). "Redes de Comunicação Industriais para os Níveis Inferiores da Hierarquia CIM". 1.º Encontro Nacional do Colégio de Engenharia Electrotécnica, Lisboa, Maio, 107-112.
- Compaq (2001). "Compaq 11Mbps Wireless LAN User's Guide". Hewlett Packard. Informação disponível on-line em: <http://www.compaq.com.au/products/wireless/wl410/>.
- Day, J. e Zimmermann, H (1983). "The OSI Reference Model". Proceedings of the IEEE, Vol. 71, No. 12, pp. 1334-1340.
- EN 50170. (1996). "General Purpose Field Communication System". EN 50170-1 (P-NET), EN 50170-2 (PROFIBUS), EN 50170-3 (WorldFIP). CENELEC.
- Eshed Robotec (1991). "SCORBOT ER-VII User's Manual". Catalog No. 100017, ISBN 965-291-003-3.
- Eshed Robotec (1994). "ACL – Advanced Control Language Versions 1.43, F1.44 and ATS – Advanced Terminal Software Version 1.44"; Reference guide for controller A, 3rd Edition, Revised.. Catalog No. 100020, ISBN 965-291-033-3.
- Ferreira, L., Machado, S. e Tovar, E. (2001). "Scheduling IP Traffic in Multimedia Enabled PROFIBUS Networks". Actas da 8th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2001), pp. 169-176. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=132.
- Grow, R. (1982). "A Timed Token Protocol for Local Area Networks". Actas da Electro'82, Token Access Protocols, Paper 17/3.
- IEEE 802.4. (1985). "IEEE Standard 802.4: Token Passing Bus Access Method and Physical Layer Specification", IEEE.
- IEEE 802.11b (1999). "Wireless LAN Medium Access Control and Physical Layer Specification". IEEE standard board, USA.
- ISO 9314-2 (1989). "Information Processing Systems - Fibre Distributed Data Interface (FDDI)". Part 2: Token Ring Media Access Control (MAC). ISO.
- Jonker, P. e Caarls, J. (2003). "Towards Tele-Presence: Combining Ambient Awareness, Augmented Reality and Autonomous Robots". ERCIM News, No. 53, Abril. Disponível on-line em: http://www.ercim.org/publication/Ercim_News/enw53/jonker.html.

- Lenhart, G. (1993). "A Fieldbus Approach to Local Control Networks". *Advances in Instrumentation and Control*, Vol. 48, No. 1, pp. 357-365.
- Longman Dictionary (2005). "Longman Dictionary of Contemporary English". 4.^a Edição. British National Corpus.
- Marques, L. (2002a). "RFieldbus ISEP Pilot: Robot Arms Applications". Relatório Técnico IPP-HURRAY.
- Marques, L. (2002b). "Colour detection and part classification in a Flexible Manufacturing System". Relatório Técnico IPP-HURRAY.
- MOST (2004). "Most Specification Version 2.3-00", MOST (Media Oriented Systems Transport) Cooperation. Disponível on-line em: <http://www.mostnet.org>.
- NetMeeting (2001). "Microsoft NetMeeting Users' Guide". Microsoft Corporation. Informação disponível on-line em: <http://www.microsoft.com/windows/netmeeting/>.
- Pacheco, F. (2002a). "Developing S7200 Ladder Programs for a Flexible Manufacturing System". Relatório Técnico IPP-HURRAY.
- Pacheco, F. (2002b). "RFieldbus ISEP Pilot: ActiveX Component". Relatório Técnico IPP-HURRAY.
- Pacheco, F. (2002c). "RFieldbus ISEP Pilot: SMS Gateway". Relatório Técnico IPP-HURRAY.
- Pacheco, F. (2002d). "RFieldbus ISEP Pilot: UDP Component for eMVB3.0 Platforms". Relatório Técnico IPP-HURRAY.
- Pacheco, F. (2002e). "RFieldbus ISEP Pilot: Windows Messages Class". Relatório Técnico IPP-HURRAY.
- Pacheco, F. e Tovar, E. (2002). "User-interface Technologies for the Industrial Environment: Towards the Cyber-factory". Actas da 6th CaberNet Radicals Workshop, Funchal, Madeira, Fevereiro. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=131.
- Pacheco, F., Tovar, E., Kalogeras, A. e Pereira, N. (2001). "Supporting Internet Protocols in Master-Slave Fieldbus Networks". Actas da 4th IFAC International Conference on Fieldbus Systems and Their Applications (FET'2001), Novembro, pp. 260-266. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=127.
- Pereira, N. (2002). "The Robuter II Application for the RFieldbus Manufacturing Application Pilot". Relatório Técnico IPP-HURRAY.
- Pereira, N. e Pacheco, F. (2002). "The RF-SCADA Application – Description and Implementation Details". Relatório Técnico IPP-HURRAY.
- Pereira, N., Pacheco, F., Pinho, L., Prayati, A., Nikoloutsos, E., Kalogeras, A., Hintze, E., Adamczyk, H., e Rauchhaupt, L. (2002). "Integration of TCP/IP and PROFIBUS Protocols". Actas da WIP Session da 4th IEEE International Workshop on Factory Communication Systems (WFCS2002), Vasteras, Suécia, Agosto. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=144.
- Piloto LASCRI (2003). Informação disponível on-line em: <http://www.hurray.isep.ipp.pt/rfpilot/>.
- Rauchhaupt, L. (2002). "System and Device Architecture of a Radio Based Fieldbus – The RFieldbus System". Actas da 4th IEEE International Workshop on Factory Communication Systems (WFCS2002), Vasteras, Suécia, Agosto, pp. 185-192.
- Robosoft (1997). "Robuter Rectangular Base Documentation". Robosoft, Fevereiro.
- Rodriguez, A., Gatrell, J., Karas, J. e Peschke, R. (2001). "TCP/IP Tutorial and Technical Overview", IBM, ibm.com/RedBooks.
- Disponível on-line em: <http://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf>.
- Schenker, J. L. (2000). "Not Very PC". *Time Europe*, Vol. 155, No. 8, 28 de Fevereiro. Disponível on-line em: <http://www.time.com/time/europe/magazine/2000/228/devices.html>.
- Siemens (1998). "SIMATIC S7-200 Programmable Controller System Manual". Publicação Siemens No. C7900-67076-C230-02. Informação disponível on-line em: http://www2.automation.siemens.com/s7-200/index_76.htm.

- Siemens (2000). "SIMATIC ET 200M Distributed I/O device Manual". Publicação Siemens No. EWA 4NEB 780 6006-0204, Edition 05. Informação disponível on-line em: http://www2.automation.siemens.com/simatic/dp/html_76/produkte/produkte.htm.
- Siemens (2001). "MicroMaster 420 User Documentation". Issues 12/2001. Publicação Siemens No. 6SE6400-5BA00-0BP0. Informação disponível on-line em: http://www.industry.siemens.com/airports/EN/products_systems/drives_micro_master.htm.
- Stankovic, J. (1988). "Real-Time Computing Systems: the Next Generation". Em Tutorial: *Hard Real-Time Systems*, Stankovic, J. and K. Ramamritham (Editores), IEEE Computer Society Press, Los Alamitos, EUA, pp. 14-38.
- Tacke, C. e Basset, T. (2002). "eMbedded Visual Basic: Windows CE and Pocket PC Mobile Applications". SAMS Publishing.
- Thomasse, J. (1997). "The Fieldbuses". Actas do International Symposium on Intelligent Components and Instruments for Control Applications (SICICA97), pp. 13-23.
- Tovar, E. (1999). "Supporting Real-Time Communications with Standard Factory-Floor Networks". Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, Julho. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=60.
- Tovar, E. e Vasques, F. (1999). "Cycle Time Properties of the PROFIBUS Timed Token Protocol". Computer Communications, Elsevier Science, Vol. 22, No. 13, 1206-1216. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=48.
- Tovar, E. e Vasques, F. (1999b). "Real-Time Fieldbus Communications Using Profibus Networks". IEEE Transactions on Industrial Electronics, Vol. 46, No. 6, pp. 1241-1251. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=44.
- Tovar, E. e Vasques, F. (2000). "Distributed Computing for the Factory-floor: a Real-Time Approach Using WorldFIP Networks". Computers in Industry, Vol. 44, No. 1, pp. 11-30, Elsevier Science. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=53.
- Tovar, E., Pinho, L., Pacheco, F. e Alves, M. (2003). "Bringing Industrial Multimedia to the Factory-Floor: What is at stake with RFieldbus?". Actas da 5th IFAC International Conference on Fieldbus Systems and their Applications (FET03), Aveiro, Portugal, pp. 131-138. Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=162.
- Tovar, E., Vasques, F., Pacheco, F. e Ferreira, L. (2001). "Industrial Multimedia over Factory-Floor Networks". Actas da 10th IFAC Symposium on Information Control Problems in Manufacturing (INCOM '01). Disponível on-line em: http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=123.
- Winnov (2001). "Winnov VideumCam Traveller User's Guide". Informação disponível on-line em: <http://www.winnov.de/products/collaboration/vidcamtraveler.htm>.

Anexo A

Detalhes VB6.0 da Aplicação Cliente *Intranet*

Neste anexo são descritos os detalhes essenciais da aplicação desenvolvida para os clientes *Intranet* (versão PC). A aplicação é constituída por dois módulos (*Modules*), treze formulários (*Forms*) *VisualBasic* 6.0 (descritos neste anexo) e por uma série de ficheiros multimédia gráficos (.ico, .bmp, .jpg) ou áudio (.wav) utilizados pelos módulos e formulários.

A.1. Módulo *Sounds* (*Sounds .bas*)

A.1.1. Breve Descrição

Neste módulo são definidas constantes de parametrização associadas à função `sndPlaySound` incluída na biblioteca de funcionalidades multimédia do *Windows winmm.dll*.

A função `sndPlaySound` é invocada em vários *forms*, pelo que este módulo a declara global ao projecto.

A.1.2. Código do *Sounds.bas*

```
Public Const SND_ASYNC = &H1      ' play asynchronously
Public Const SND_NODEFAULT = &H2  ' silence not default, if sound not found
Public Const SND_NOSTOP = &H10    ' don't stop any currently playing sound
Public Const SND_NOWAIT = &H2000  ' don't wait if the driver is busy
Public Const SND_SYNC = &H0       ' play synchronously (default)

Public Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal
lpszSoundName As String, ByVal uFlags As Long) As Long
```

A.2. Módulo *Globals* (Globals.bas)

A.2.1. Breve Descrição

Neste módulo são declaradas variáveis globais ao projecto, e utilizadas no contexto de diversos *forms*. São ainda definidos dois procedimentos de gestão de visibilidade.

Relativamente a estes, o *SwitchToForm* é utilizado para navegação entre as janelas (formulários) e gestão da visibilidade das janelas activas da aplicação. O outro (*ShowAlarm*) está relacionado com a visualização da janela de alarmes (*Show_Events_Form*).

A.2.2. Código do *Globals.bas*

```
Public FormList(20) As Form
Public Const Gray = &H8000000F
Public i As Integer

' utilizada para guardar dados rec. sobre BUF, QUEUES, AGVs, RAS
Public Info_Monitor(7) As String
' utilizada para guardar dados rec. sobre MAX_BUF
Public info_bufmax(5) As Integer

' utilizadas para guardar a informação difundida sobre configurações HMD/PK1/PK2
Public info_cam(3) As Integer
Public info_alarm(3) As String

' utilizadas para guardar as imagens das 5 camaras
Public cam(5) As IPictureDisp

' Botoes do Main_Form
Public ButtMain(5) As CommandButton
' Botoes do Control_Form
Public ButtControl(9) As CommandButton
' Botoes do Monitor_Form
Public ButtMonitor(3) As CommandButton
' Botoes do Monitor_BR_Form
Public ButtMonitorBR(7) As CommandButton
' Botoes do BR_Info_Form
Public ButtBRInfo(7) As CommandButton
' Botoes do Select_CAM_Info_Form
Public ButtSelectCAMInfo(6) As CommandButton
' Botoes do CAM_View_Form
Public ButtCAMView(8) As CommandButton
' Botoes do Config_Form
Public ButtConfig(3) As CommandButton
' Botoes do Config_Buffers_Form
Public ButtConfBuffers(7) As CommandButton
' Botoes do Config_Devices_Form
Public ButtConfDevices(6) As CommandButton

Public Const CLIENT_PORT = 1319
Public Const SERVER_PORT = 1309
Public Const BROADCAST_ADDR = "255.255.255.255"
Public SERVER_ADDR As String

Public CamID As Byte
```

```

'Utilizadas para guardar informação de texto das imagens
Public g_txt_cam(5) As String
'E quais são as camaras enabled
Public g_cam_enable(5) As Boolean
'Pedido pendente para activar a camara
Public cam_req(5) As Boolean

Public Sub SwitchToForm(SrcForm As Form, DestForm As Form)
    DestForm.Left = SrcForm.Left
    DestForm.Top = SrcForm.Top
    For i = 0 To 20
        If Not FormList(i) Is Nothing Then
            If FormList(i).Name <> DestForm.Name Then
                FormList(i).Hide
            End If
        End If
    Next
    DestForm.Show
    On Error Resume Next
    DestForm.Text1.SetFocus
End Sub

Public Sub ShowAlarm(id, msg)
    Dim lastForm As Form
    For i = 0 To 10
        If Not FormList(i) Is Nothing Then
            If FormList(i).Visible Then
                Set lastForm = FormList(i)
            End If
        End If
    Next

    On Error Resume Next
    Show_Events_Form.Picture1.Picture = LoadPicture(App.Path & "\" & id & ".bmp")
    Show_Events_Form.Timer1.Enabled = True

    If Err Then
        Show_Events_Form.Picture1.Picture = LoadPicture(App.Path & "\\info.bmp")
        msg = id & ":" & vbCrLf & msg
    End If
    Show_Events_Form.T_Event.Tag = "OK"
    Show_Events_Form.T_Event.Text = msg
    Show_Events_Form.T_Event.Tag = ""
    If lastForm.Caption <> Show_Events_Form.Caption Then
        Set Show_Events_Form.g_lastForm = lastForm
        SwitchToForm lastForm, Show_Events_Form
    End If
    sndPlaySound "rf" & id, SND_FILENAME Or SND_ASYNC
End Sub

```

A.3. Entry_Form

A.3.1. Layout da Entry_Form

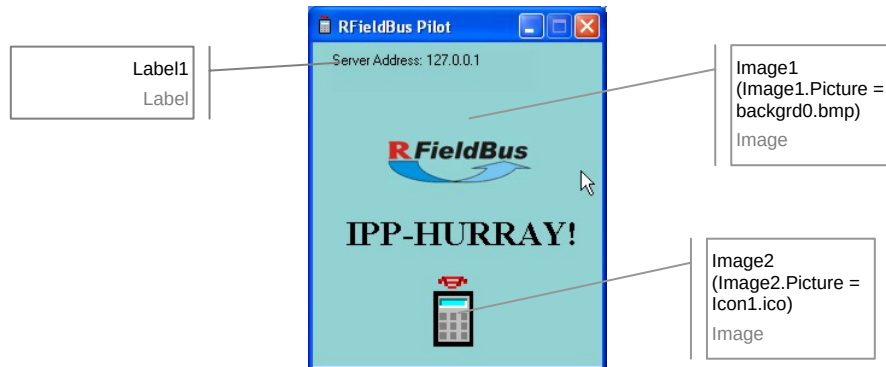


Figura A.1: Layout da Entry_Form

A.3.2. Objectos relevantes da Entry_Form

Tabela A.1: Relação de objectos relevantes da Entry_Form

Objecto (Nome)	Tipo	Observações
Image1	Image	Associado a ficheiro gráfico (backgrd0 . bmp), fundo
Image2	Image	Associado a ficheiro gráfico (Icon1 . ico)
Label1	Label	Visualiza o endereço IP do servidor <i>Intranet</i>
Timer1	Timer	Para fazer um <i>blink</i> temporizado desta janela de entrada

A.3.3. Código da Entry_Form

```

Private Sub Form_Load()
    sndPlaySound "about.wav", SND_ASYNC Or SND_NODEFAULT Or SND_NOWAIT
    On Error Resume Next
    SERVER_ADDR = "127.0.0.1" ' por defeito o endereço IP do Servidor é local
    Open "config.txt" For Input As #1
    If Err Then
        Open "config.txt" For Output As #1
        Print #1, "127.0.0.1"
        Close #1
        Open "config.txt" For Input As #1
    End If
    ' caso contrário, terá o endereço contido no ficheiro de texto "config.txt"
    Line Input #1, SERVER_ADDR
    Close #1
    Entry_Form.Label1.Caption = "Server Address: " & SERVER_ADDR
End Sub

```

```

Private Sub Timer1_Timer()
' utilizado para fazer blink do form de entrada, e ao fim de
' algum tempo passar para o Main_Form

Static i As Integer
Static tm As Double
If i = 0 Then tm = Timer1.Interval
tm = tm * 0.75
Timer1.Interval = tm + 1
i = i + 1
Image2.Visible = i Mod 2 = 0
If i >= 15 Then
    Main_Form.Show
    Main_Form.Left = Entry_Form.Left
    Main_Form.Top = Entry_Form.Top
    Entry_Form.Hide
    Main_Form.Text1.SetFocus
    Timer1.Enabled = False
End If
End Sub

```

A.4. Main_Form

A.4.1. Layout da Main_Form

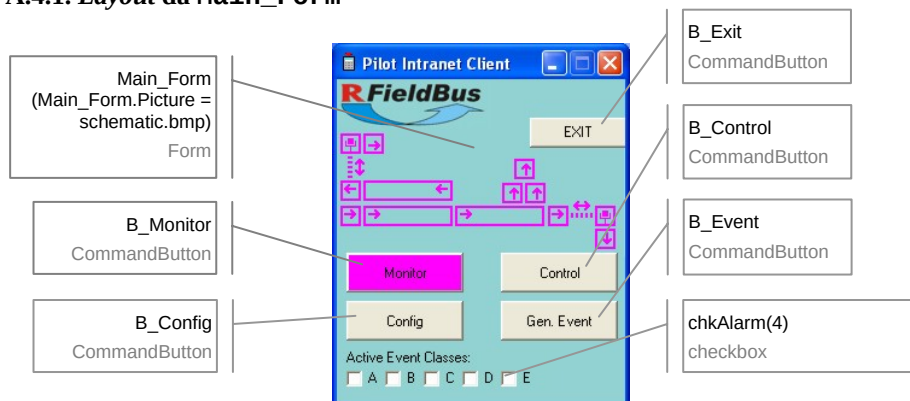


Figura A.2: Layout da Main_Form

A.4.2. Objectos relevantes da Main_Form

Tabela A.2: Relação de objectos relevantes da Main_Form

Objecto (Nome)	Tipo	Observações
B_Config	CommandButton	Para ir para a janela Config_Form
B_Control	CommandButton	Para ir para a janela Control_Form
B_Event	CommandButton	Para ir para a janela Gen_Events_Form
B_Exit	CommandButton	Para sair da aplicação
B_Monitor	CommandButton	Para ir para a janela Monitor_Form
chkAlarm(i)	CheckBox	Seleccionar as “classes” de alarme a receber no cliente
Main_Form	Form	
Label1	Label	Caption = Active Event Classes:
Text1	TextBox	Utilizado para navegação alternativa com o teclado (ver Cap. 5)
Wsk1	Winsock	Para comunicação UDP/IP com o servidor <i>Intranet</i>

A.4.3. Código da Main_Form

```

Option Explicit
Public SelInd As Integer
Dim ListCamIDs As New Collection

Private curImgId As Integer
Private g_txt_cam_temp(5) As String
Private TotFrag(5) As Byte
Private CurFrag(5) As Byte

Private Sub B_Exit_Click()
    ButClick (5)
End Sub

Private Sub B_CONFIG_Click()
    ' ir para a form de configuração
    ButClick (3)
End Sub

Private Sub B_Control_Click()
    ' ir para a form de controlo do sistema
    ButClick (2)
End Sub

Private Sub B_Event_Click()
    ButClick (4)
End Sub

Private Sub B_Monitor_Click()
    ' ir para a form de monitorização do sistema
    ButClick (1)
End Sub

Private Sub chkAlarm_Click(Index As Integer)
    Show_Events_Form.chkAlarm(Index).Value = chkAlarm(Index).Value
End Sub

```

```

Private Sub Form_Unload(Cancel As Integer)
Dim CamID As Integer
Main_Form.Wsk1.RemotePort = SERVER_PORT: Main_Form.Wsk1.RemoteHost = SERVER_ADDR
On Error Resume Next
For CamID = 1 To 5
    Close #(10 + CamID)
    Kill App.Path & "\cams\rcvcam" & CamID & ".jpg"
    ' se camara on, envia OFF ao servidor
    If g_cam_enable(CamID) Then Wsk1.SendData ">CAM" & CamID & ":OFF"
Next CamID
End
End Sub

Private Sub Form_Load()
Dim CamID As Integer
Set FormList(0) = Me

Entry_Form.Timer1.Enabled = False

Set ButtMain(1) = B_Monitor
Set ButtMain(2) = B_Control
Set ButtMain(3) = B_Config
Set ButtMain(4) = B_Event
Set ButtMain(5) = B_Exit

Wsk1.LocalPort = CLIENT_PORT
Wsk1.RemotePort = SERVER_PORT
Wsk1.RemoteHost = SERVER_ADDR
Wsk1.Bind

' faz pedidos de status do sistema ao servidor servidor
' servem depois para actualizar (via Wsk1_DataArrival) os
' dois labels do form Control_Form, e o label do Config_Buffers_Form
Wsk1.SendData "#OP"
Wsk1.SendData "#CB_SPEED"

i = 1
SelInd = i
ButtMain(SelInd).BackColor = vbMagenta

For CamID = 1 To 5
    Open App.Path & "\cams\rcvcam" & CamID & ".jpg" For Output As #1
    Close #1
Next CamID
End Sub

Private Sub Form_KeyDown(Keycode As Integer, Shift As Integer)
Select Case Keycode
    Case 39 ' Direita
        ButtMain(SelInd).BackColor = Gray
        If (SelInd < 5) Then
            SelInd = SelInd + 1
        Else
            SelInd = 1
        End If
        ButtMain(SelInd).BackColor = vbMagenta
    Case 37 ' Esquerda
        ButtMain(SelInd).BackColor = Gray
        If (SelInd > 1) Then
            SelInd = SelInd - 1
        Else
            SelInd = 5
        End If
        ButtMain(SelInd).BackColor = vbMagenta
    Case 13 ' Enter
        ButClick SelInd
End Select
End Sub

```

```

Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
    Form_KeyDown KeyCode, Shift
End Sub

Private Sub ButtClick(Index As Integer)
    For i = 1 To 5
        ButtMain(i).BackColor = Gray
    Next i
    ButtMain(Index).BackColor = vbMagenta
    SelInd = Index
    Select Case Index
        Case 1
            SwitchToForm Me, Monitor_Form
        Case 2
            SwitchToForm Me, Control_Form
        Case 3
            SwitchToForm Me, Config_Form
        Case 4
            SwitchToForm Me, Gen_Events_Form
        Case 5
            Unload Me
    End Select
End Sub

Private Sub Form_OKClick()
    End
End Sub

Private Sub Wsk1_DataArrival(ByVal bytesTotal As Long)
    'Mensagens UDP chegadas do Servidor INTRANET
    'ou do IMAGE CENTER (">IMG...")
    Dim msg_rec As String

    On Error Resume Next

    Wsk1.GetData msg_rec, vbString
    If msg_rec = "" Then Exit Sub
    Select Case Mid(msg_rec, 1, 1)
        Case "?"
            ' **** UPDATE SOBRE OS DISPOSITIVOS DO SFF ****
            ' **** QUEUE / BUFF / AGV / RA
            Arrived_Monitor_Update (msg_rec)
        Case ">"
            ' **** INFORMAÇÃO DAS CÂMARAS ****
            ' **** PODE SER ">CAM" (VINDA DO SERVIDOR INTRANET)
            ' **** PODE SER ">IMG" (VINDA DO IMAGE CENTER)
            Arrived_Video_Info (msg_rec)
        Case "&"
            ' **** TAMANHOS MÁXIMOS DOS BUFFERS ****
            Arrived_Buffer_Max (msg_rec)
        Case "#"
            ' **** CONTROL UPDATE ****
            Arrived_Control_Update (msg_rec)
        Case "%"
            ' **** difundida a configuração/informação de HMD/PK1/PK2 e SMS
            Arrived_HMD_PK_SMS (msg_rec)
        Case "!"
            ' **** MENSAGENS DE ALARME/EVENTOS *****
            Arrived_Alarm_Event (msg_rec)
        Case Else
            ' ignora
    End Select
End Sub

```

```
Private Sub Arrived_Monitor_Update(msg_r As String)
' **** INFORMAÇÃO DO SISTEMA :QUEUES, BUFFERS, RAS, AGVs ****
Dim p As Long
Dim dev_txt, info As String
Dim id_dev As Integer

    p = InStr(msg_r, ":")
    If p > 0 Then
        dev_txt = Mid(msg_r, 2, p - 2)
        info = Mid(msg_r, p + 1)
    Else
        dev_txt = Mid(msg_r, 2)
        info = "(no info)"
    End If

    Select Case UCase(dev_txt)
        Case "QUEUE"
            id_dev = 1
        Case "BUFF"
            id_dev = 2
        Case "AGV1"
            id_dev = 3
        Case "AGV2"
            id_dev = 4
        Case "RA1"
            id_dev = 5
        Case "RA2"
            id_dev = 6
        Case Else
            id_dev = 0
    End Select

    If id_dev <> 0 Then
        ' Actualiza a variável global do cliente que conterà a
        ' informação do dispositivo
        Info_Monitor(id_dev) = info
        ' Se a janela de de monitorização activa, também é feita uma
        ' actualização da caixa de texto respectiva
        If BR_Info_Form.SelInd14 = id_dev Then
            BR_Info_Form.Text_BR_Info = info
        End If
    End If
End Sub
```

```

Private Sub Arrived_Video_Info(msg_r As String)
    ' **** INFORMAÇÃO CAMERAS ****
    Dim parm_id, parm_value, strdata As String
    Dim p As Long
    Dim datapcx As IPictureDisp
    Dim Head As String * 3
    Dim ImgID, FragID, CamID As Byte

    p = InStr(msg_r, "CAM")

    If (p = 2 And Wsk1.RemotePort = SERVER_PORT) Then
        ' Informação vinda do SERVIDOR INTRANET
        ' Informa sobre ON/OFF DA CAMARA

        p = InStr(msg_r, ":")
        If p > 5 Then
            parm_id = Mid(msg_r, 5, p - 5)
            Select Case parm_id
                Case "1", "2", "3", "4", "5", "6"
                    ' na realidade, como só há 5 cameras no sistema
                    ' o valor "6" nunca vai ser recebido
                    parm_id = Val(parm_id)
                    parm_value = UCase(Mid(msg_r, p + 1))
                    If parm_value = "OFF" Then
                        'apaga a imagem real e coloca NO FEED
                        g_cam_enable(parm_id) = False
                        Set datapcx = LoadPicture(App.Path & "\cams\novideo.jpg")
                        Set cam(parm_id) = datapcx
                        g_txt_cam(parm_id) = ""
                        'Atualiza de acordo se janela activa
                        If CAM_View_Form.SelInd15 = parm_id Then
                            CAM_View_Form.B_CM_ON.Enabled = True
                            CAM_View_Form.B_CM_OFF.Enabled = False
                            Set CAM_View_Form.CamPic.Picture = datapcx
                            CAM_View_Form.L_CAM_Info.Caption = g_txt_cam(parm_id)
                        End If
                    End If
                    'se recebeu um ON e há um pedido de subscrição pendente
                    If parm_value = "ON" And cam_req(parm_id) Then
                        g_cam_enable(parm_id) = True
                        cam_req(parm_id) = False
                        If CAM_View_Form.SelInd15 = parm_id Then
                            CAM_View_Form.B_CM_ON.Enabled = False
                            CAM_View_Form.B_CM_OFF.Enabled = True
                        End If
                    End If
                Case Else
                    'câmara desconhecida
            End Select
        Else
            'pedido inválido
        End If
    End If

```

```

p = InStr(msg_r, "IMG")
' Informação vinda do IMAGE CENTER
' RECEPÇÃO DE IMAGENS (JPG)
' Formato dos Fragmentos IMG:
' se Frag 0: ">IMG" & <ImgID> & <FragID> & <CamID> & <TotFrag> & <> & <g_cam_txt>
' se Frag 1: ">IMG" & <ImgID> & <FragID> & <strdata>

If p = 2 Then
    'Identificação da imagem e do fragmento
    ImgID = Asc(Mid(msg_r, 5, 1))
    FragID = Asc(Mid(msg_r, 6, 1))

    ' Se primeiro fragmento (FragID = 0)
    If FragID = 0 Then
        'obtem o identificador de imagem e da camara e o total de fragmentos
        curImgID = ImgID
        CamID = Asc(Mid(msg_r, 7, 1))
        TotFrag(CamID) = Asc(Mid(msg_r, 8, 1))
        CurFrag(CamID) = 1

        If g_cam_enable(CamID) = False Then Exit Sub

        g_txt_cam_temp(CamID) = Mid(msg_r, 9)

        ' fecha o ficheiro associado à camare (pode não ter sido fechado se
        ' por alguma razão a imagem anterior não foi totalmente recebida
        On Error Resume Next
        Close #(10 + CamID)
        On Error GoTo 0

        ' abre um ficheiro rcvcamX.jpg, com um handler 1X
        Open App.Path & "\cams\rcvcam" & CamID & ".jpg" For Binary As #(10 +
            CamID)

        ' se a cam já estava na Collection ListCamIDs, quer dizer que
        ' por alguma razão a imagem anterior não foi totalmente recebida
        i = 1
        Do While i <= ListCamIDs.Count
            If ListCamIDs.Item(i) = CamID Then
                ListCamIDs.Remove i
            Else
                i = i + 1
            End If
        Loop

        On Error Resume Next
        'adiciona à Collection ListCamIDs a
        ListCamIDs.Add CamID, "" & ImgID
        If Err = 457 Then
            ListCamIDs.Remove "" & ImgID
            ListCamIDs.Add CamID, "" & ImgID
        End If
        Exit Sub

    ' se fragmentos restantes (FragID > 0)
    Else
        CamID = 0
        CamID = ListCamIDs("" & ImgID)
        If CamID = 0 Then 'Sair se id imagem inválido
            Exit Sub
        End If

        If FragID <> CurFrag(CamID) Then
            ListCamIDs.Remove "" & ImgID
            Exit Sub
        End If
    End If

```

```

'Dados do Fragmento
strdata = Mid(msg_r, 7)

' Guarda no ficheiro e incrementa o CurFrag para a CAM
Put #(10 + CamID), , strdata
CurFrag(CamID) = CurFrag(CamID) + 1

' se foi o último fragmento da imagem
If CurFrag(CamID) > TotFrag(CamID) Then
    Close #(10 + CamID)
    ListCamIDs.Remove "" & ImgID

    ' caso a camara ainda esteja subscrita no cliente
    If g_cam_enable(CamID) Then
        'copia para g_txt_cam(CamID) -> é o utilizado no CAM_View_Form
        g_txt_cam(CamID) = g_txt_cam_temp(CamID)

        ' Guarda em cam(i) - tipo IPPictureDisp - a imagem
        Set datapcx = LoadPicture(App.Path & "\cams\rcvcam" & CamID &
            ".jpg")

        Set cam(CamID) = datapcx
        ' e actualiza os objectos CamPic e L_CAM_Info
        ' se o contexto da CAM_View_Form é o da CamID
        If CAM_View_Form.SelInd15 = CamID Then
            Set CAM_View_Form.CamPic.Picture = datapcx
            CAM_View_Form.L_CAM_Info.Caption = g_txt_cam(CamID)
        End If
    End If
End If
End If
End If
End Sub

```

```
Private Sub Arrived_Buffer_Max(msg_r As String)
Dim p As Long
Dim parm_id, parm_val As String
Dim id_campo As Integer

p = InStr(msg_r, ":")
If p > 0 Then
    parm_id = UCase(Mid(msg_r, 2, p - 2))
    parm_val = Mid(msg_r, p + 1)

    Select Case parm_id
    Case "B1MAX":
        id_campo = 1
    Case "B2MAX":
        id_campo = 2
    Case "B3MAX":
        id_campo = 3
    Case "B4MAX":
        id_campo = 4
    Case "B5MAX":
        id_campo = 5
    Case Else:
        id_campo = 0
    End Select
    If id_campo > 0 Then
        ' ALTERA AS DIRECTAMENTE O CAPTION DAS LABELS (SE JANELA ACTIVA)
        Config_Buffers_Form.L_PARM(id_campo) = parm_val
        ' E ACTUALIZA DO CLIENTE LOCAL QE CONTERÁ ESSA INFORMAÇÃO
        ' PARA QUANDO A JANELA É SELLECCIONADA
        info_bufmax(id_campo) = parm_val
    Else
        'ignorar: parâmetro desconhecido
    End If
End If
End Sub
```

```

Private Sub Arrived_Control_Update(msg_r As String)
Dim p As Long
Dim parm_id, parm_value As String
p = InStr(msg_r, ":")
If p > 0 Then
    parm_id = UCase(Mid(msg_r, 2, p - 2))
    parm_value = UCase(Mid(msg_r, p + 1))
    Select Case parm_id
    Case "OP"
        p = 2

        'Atualiza os labels dos forms
        Select Case parm_value
        Case "STOP"
            Control_Form.L_STATUS.Caption = "System is stopped."
            p = 1
        Case "STOPPING"
            Control_Form.L_STATUS.Caption = "System is stopping..."
        Case "RUN"
            Control_Form.L_STATUS.Caption = "System is running."
        Case "ERROR"
            Control_Form.L_STATUS.Caption = "System is in ERROR state."
        Case Else
            p = 0
        End Select
        'ignorar: pedido desconhecido

        ' faz o enable ou disable do botoes dos forms
        Select Case p
        Case 1:
            Load Control_Form
            Load Config_Buffers_Form

            For i = 2 To 8
                ButtControl(i).Enabled = True
            Next
            For i = 1 To 6
                ButtConfBuffers(i).Enabled = True
            Next
            Config_Buffers_Form.L_STATUS.Caption = Control_Form.L_STATUS.Caption
            Config_Buffers_Form.L_STATUS.BackColor = &HFFFFFF
            Config_Buffers_Form.L_STATUS.ForeColor = 0
        Case 2:
            Load Control_Form
            Load Config_Buffers_Form

            For i = 2 To 8
                'botoes do form Control_Form
                ButtControl(i).Enabled = False
            Next
            For i = 1 To 6
                'botoes do form Config_Buffers_Form
                ButtConfBuffers(i).Enabled = False
            Next
            Config_Buffers_Form.L_STATUS.Caption = Control_Form.L_STATUS.Caption
            Config_Buffers_Form.L_STATUS.BackColor = 0
            Config_Buffers_Form.L_STATUS.ForeColor = &HFFFFFF
        End Select
        Case "CB_SPEED"
            Control_Form.L_SPEED.Caption = "Speed: " & parm_value
        Case Else
            'ignorar: pedido desconhecido
        End Select
    End If
End Sub

```

```

Private Sub Arrived_HMD_PK_SMS(msg_r As String)
' Configuração/Informação HMD-Pocket1-Pocket2 e SMS
Dim p As Long
Dim t2, txt_info, al_cl_lst, id_device As String
Dim i, dev As Integer

    p = InStr(msg_r, ":")
    If p > 0 Then
        id_device = Mid(msg_r, 2, p - 2)
        t2 = Mid(msg_r, p + 1)
        p = InStr(t2, ":")

        If p > 0 Then
            txt_info = Mid(t2, p + 1)
            al_cl_lst = Left(t2, p - 1)

            ' Actualizações de propriedades do form Config_Devices_Form
            ' e das variáveis do cliente que contem informação associada
            ' utilizada para quando se seleccion o referido form:
            ' info_cam(i); info_alarm(i)
            dev = -1
            Select Case id_device
                Case "HMD": dev = 0
                Case "PK1": dev = 1
                Case "PK2": dev = 2
                Case "SMS"
                    For i = 0 To 4
                        If InStr(al_cl_lst, Config_Devices_Form.Ck2(i).Caption)
                            > 0 Then
                            Config_Devices_Form.Ck2(i).Value = 1
                        Else
                            Config_Devices_Form.Ck2(i).Value = 0
                        End If
                    Next i
                    Config_Devices_Form.T_SMS_Number.Text = txt_info
            End Select

            'se HMD/PK1/PK2
            If dev >= 0 Then
                info_cam(dev) = txt_info
                info_alarm(dev) = al_cl_lst
                If Config_Devices_Form.B_Info_Select(dev).Font.Bold = True Then
                    For i = 0 To 4
                        If InStr(info_alarm(dev), Config_Devices_Form.Ck1(i).Caption)
                            > 0 Then
                            Config_Devices_Form.Ck1(i).Value = 1
                        Else
                            Config_Devices_Form.Ck1(i).Value = 0
                        End If
                    Next i
                    Config_Devices_Form.Opt1(Val(txt_info)) = True
                End If
            End If
        End If
    End If
End Sub

```

```

Private Sub Arrived_Alarm_Event(msg_r As String)
' **** MENSAGENS DE ALARME/EVENTOS ****
Dim p As Long
Dim t2, alarm_class, alarm_type, txt_info As String
Dim alarm_accepted As Boolean

    p = InStr(msg_r, ":")
    If p > 0 Then
        t2 = Mid(msg_r, p + 1)
        p = InStr(t2, ":")

        If p > 0 Then
            alarm_class = Left(t2, p - 1)
            alarm_accepted = False
            Select Case alarm_class
                Case "A"
                    alarm_accepted = Main_Form.chkAlarm(0) = 1
                Case "B"
                    alarm_accepted = Main_Form.chkAlarm(1) = 1
                Case "C"
                    alarm_accepted = Main_Form.chkAlarm(2) = 1
                Case "D"
                    alarm_accepted = Main_Form.chkAlarm(3) = 1
                Case "E"
                    alarm_accepted = Main_Form.chkAlarm(4) = 1
            End Select

            If alarm_accepted Then
                t2 = Mid(t2, p + 1)
                p = InStr(t2, ":")

                If p > 0 Then
                    ' vai abrir a janela Show_Events_Form (via)
                    ' chamada a ShowAlarm
                    txt_info = Mid(t2, p + 1)
                    alarm_type = Left(t2, p - 1)
                    ShowAlarm alarm_type, txt_info
                End If
            End If
        End If
    End If
End Sub

```

A.5. Monitor_Form

A.5.1. Layout da Monitor_Form

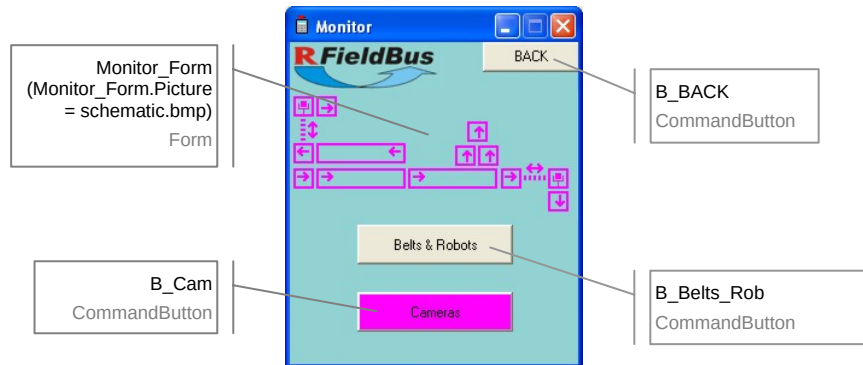


Figura A.3: Layout da Monitor_Form

A.5.2. Objectos relevantes da Monitor_Form

Tabela A.3: Relação de objectos relevantes da Monitor_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
B_Belts_Rob	CommandButton	Para ir para a janela Monitor_BR_Form
B_Cam	CommandButton	Para ir para a janela Select_CAM_Info_Form
Control_Form	Form	
Text1	TextBox	Utilizado para navegação alternativa com o teclado (ver Cap. 5)

A.5.3. Código da Monitor_Form

```
Option Explicit
Public SelInd As Integer

Private Sub Form_Load()
    Set FormList(11) = Me
    Set ButtMonitor(1) = B_Belts_Rob
    Set ButtMonitor(2) = B_Cam
    Set ButtMonitor(3) = B_Back

    SelInd = 1
    ButtMonitor(SelInd).BackColor = vbMagenta
End Sub
```

```

Private Sub B_BACK_Click()
    ButtClick(3)
End Sub

Private Sub B_Belts_Rob_Click()
    ButtClick(1)
End Sub

Private Sub B_Cam_Click()
    ButtClick(2)
End Sub

Private Sub Form_KeyDown(Keycode As Integer, Shift As Integer)
    Select Case Keycode
    Case 39 ' Direita
        ButtMonitor(SelInd).BackColor = Gray
        If (SelInd < 3) Then
            SelInd = SelInd + 1
        Else
            SelInd = 1
        End If
        ButtMonitor(SelInd).BackColor = vbMagenta
    Case 37 ' Esquerda
        ButtMonitor(SelInd).BackColor = Gray
        If (SelInd > 1) Then
            SelInd = SelInd - 1
        Else
            SelInd = 3
        End If
        ButtMonitor(SelInd).BackColor = vbMagenta
    Case 13 ' Enter
        ButtClick SelInd
    End Select
End Sub

Private Sub ButtClick(Index As Integer)
    For i = 1 To 3
        ButtMonitor(i).BackColor = Gray
    Next i
    ButtMonitor(Index).BackColor = vbGreen
    SelInd = Index
    Select Case Index
    Case 1
        SwitchToForm Me, Monitor_BR_Form
    Case 2
        SwitchToForm Me, Select_CAM_Info_Form
    Case 3
        Unload Me
    End Select
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, Main_Form
    Cancel = True
    Me.Hide
End Sub

Private Sub Text1_KeyDown(Keycode As Integer, Shift As Integer)
    Form_KeyDown Keycode, Shift
End Sub

```

A.6. Monitor_BR_Form

A.6.1. Layout da Monitor_BR_Form

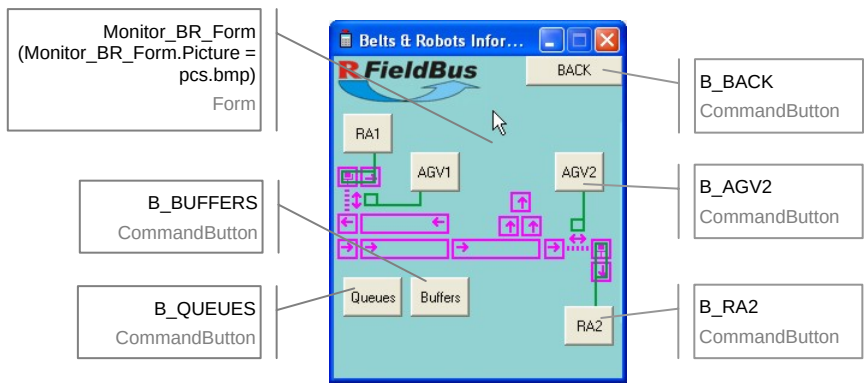


Figura A.4: Layout da Monitor_BR_Form

A.6.2. Objectos relevantes da Monitor_BR_Form

Tabela A.4: Relação de objectos relevantes da Monitor_BR_Form

Objecto (Nome)	Tipo	Observações
B_AGV1; B_AGV2	CommandButton	Ir para a janela BR_Info_Form, contexto AGV1 (AGV2)
B_Back	CommandButton	Para ir para a janela anterior
B_BUFFERS	CommandButton	Ir para a janela BR_Info_Form, contexto BUFFERS
B_QUEUES	CommandButton	Ir para a janela BR_Info_Form, contexto QUEUES
B_RA1; B_RA2	CommandButton	Ir para a janela BR_Info_Form, contexto RA1 (RA2)
Monitor_BR_Form	Form	
Text1	TextBox	Utilizado para navegação alternativa com teclado (ver Cap. 5)

A.6.3. Código da Monitor_BR_Form

```
Option Explicit
Public SelInd12 As Integer
```

```

Private Sub Form_Load()
    Set FormList(12) = Me

    Set ButtMonitorBR(1) = B_QUEUES
    Set ButtMonitorBR(2) = B_BUFFERS
    Set ButtMonitorBR(3) = B_AGV1
    Set ButtMonitorBR(4) = B_AGV2
    Set ButtMonitorBR(5) = B_RA1
    Set ButtMonitorBR(6) = B_RA2
    Set ButtMonitorBR(7) = B_Back

    SelInd12 = 1
    ButtMonitorBR(SelInd12).BackColor = vbMagenta

    Main_Form.Wsk1.RemotePort = SERVER_PORT
    Main_Form.Wsk1.RemoteHost = SERVER_ADDR

    'request information update
    Main_Form.Wsk1.SendData "?QUEUE"
    Main_Form.Wsk1.SendData "?BUFF"
    Main_Form.Wsk1.SendData "?AGV1"
    Main_Form.Wsk1.SendData "?AGV2"
    Main_Form.Wsk1.SendData "?RA1"
    Main_Form.Wsk1.SendData "?RA2"

    For i = 1 To 6
        Info_Monitor(i) = "Updating..."
    Next i
End Sub

Private Sub ButtClick(Index As Integer)
    For i = 1 To 7
        ButtMonitorBR(i).Enabled = True
        ButtMonitorBR(i).BackColor = Gray
    Next i

    SelInd12 = Index
    SwitchToForm Me, BR_Info_Form

    Select Case Index
    Case 1
        BR_Info_Form.B_QUEUES_Click
    Case 2
        BR_Info_Form.B_BUFFERS_Click
    Case 3
        BR_Info_Form.B_AGV1_Click
    Case 4
        BR_Info_Form.B_AGV2_Click
    Case 5
        BR_Info_Form.B_RA1_Click
    Case 6
        BR_Info_Form.B_RA2_Click
    Case 7
        B_BACK_Click
    End Select
End Sub

Public Function SetButton2(i)
    'utilizada pelo form BR_Info_Form
    ButtMonitorBR(i).BackColor = vbGreen
    SelInd12 = i
    For i = 1 To 6
        If i <> SelInd12 Then
            ButtMonitorBR(i).BackColor = Gray
        End If
    Next i
End Function

```

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    Select Case KeyCode
        Case 39 ' Direita
            ButtMonitorBR(SelInd12).BackColor = Gray
            If (SelInd12 < 7) Then
                SelInd12 = SelInd12 + 1
            Else
                SelInd12 = 1
            End If
            ButtMonitorBR(SelInd12).BackColor = vbMagenta
        Case 37 ' Esquerda
            ButtMonitorBR(SelInd12).BackColor = Gray
            If (SelInd12 > 1) Then
                SelInd12 = SelInd12 - 1
            Else
                SelInd12 = 7
            End If
            ButtMonitorBR(SelInd12).BackColor = vbMagenta
        Case 13 ' Enter
            ButtClick SelInd12
    End Select
End Sub

Private Sub B_AGV1_Click()
    ButtClick 3
End Sub

Private Sub B_AGV2_Click()
    ButtClick 4
End Sub

Private Sub B_RA1_Click()
    ButtClick 5
End Sub

Private Sub B_RA2_Click()
    ButtClick 6
End Sub

Private Sub B_QUEUES_Click()
    ButtClick 1
End Sub

Private Sub B_BUFFERS_Click()
    ButtClick 2
End Sub

Private Sub B_BACK_Click()
    Unload Me
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, Monitor_Form
    For i = 1 To 7
        ButtMonitorBR(i).Enabled = True
        ButtMonitorBR(i).BackColor = Gray
    Next i
    Cancel = True
    Me.Hide
End Sub

Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
    Form_KeyDown KeyCode, Shift
End Sub
```

A.7. BR_Info_Form

A.7.1. Layout da BR_Info_Form

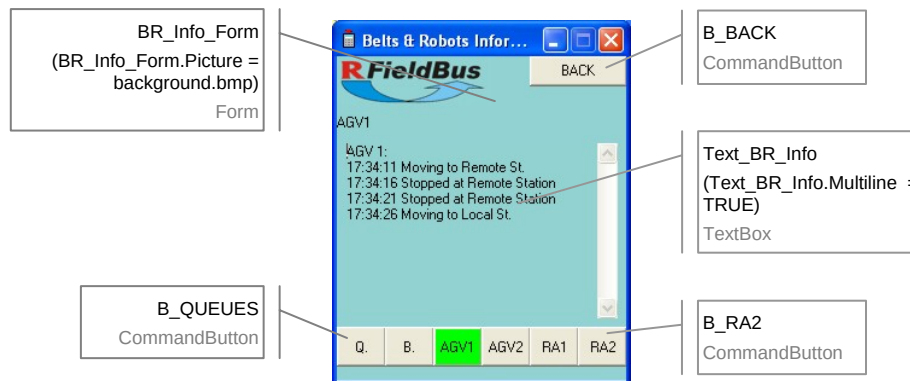


Figura A.5: Layout da BR_Info_Form

A.7.2. Objectos relevantes da BR_Info_Form

Tabela A.5: Relação de objectos relevantes da BR_Info_Form

Objecto (Nome)	Tipo	Observações
B_AGV1; B_AGV2	CommandButton	Mesma janela, mas contexto AGV1 (AGV2)
B_Back	CommandButton	Para ir para a janela anterior
B_BUFFERS	CommandButton	Mesma janela, mas contexto BUFFERS
B_QUEUES	CommandButton	Mesma janela, mas contexto QUEUES
B_RA1; B_RA2	CommandButton	Mesma janela, mas contexto RA1 (RA2)
BR_Info_Form	Form	
Label1	Label	Indicará o dispositivo relativo ao contexto
Text_BR_Info	TextBox	Atualizada por Wsk1 (se activa) ou Info_Monitor(i) quando selecciona
Text1	TextBox	Utilizado para navegação alternativa com teclado (ver Cap. 5)

A.7.3. Código da BR_Info_Form

```
Option Explicit
Public SelInd14 As Integer
```

```

Private Sub Form_Load()
    Set FormList(14) = Me

    Set ButtBRInfo(1) = B_QUEUES
    Set ButtBRInfo(2) = B_BUFFERS
    Set ButtBRInfo(3) = B_AGV1
    Set ButtBRInfo(4) = B_AGV2
    Set ButtBRInfo(5) = B_RA1
    Set ButtBRInfo(6) = B_RA2
    Set ButtBRInfo(7) = B_BACK

    For i = 1 To 6
        If ButtBRInfo(i).Enabled = True Then
            SelInd14 = i
            Exit For
        End If
    Next i
End Sub

Private Sub ButClick(Index As Integer)
    Dim msg_str As String

    Text_BR_Info.Text = Info_Monitor(Index)

    Select Case Index
        Case 1: Label1.Caption = "Queues"
        Case 2: Label1.Caption = "Buffers"
        Case 3: Label1.Caption = "AGV1"
        Case 4: Label1.Caption = "AGV2"
        Case 5: Label1.Caption = "RA1"
        Case 6: Label1.Caption = "RA2"
        Case 7: GoTo back_form
    End Select

    SelInd14 = Index
    For i = 1 To 7
        ButtBRInfo(i).Enabled = True
        ButtBRInfo(i).BackColor = Gray
    Next i

    ButtBRInfo(Index).BackColor = vbGreen
    Monitor_BR_Form.SetButton2 (Index)
    Text1.SetFocus

    Exit Sub
back_form:
    B_BACK_Click
End Sub

Sub B_AGV1_Click()
    ButClick 3
End Sub

Sub B_AGV2_Click()
    ButClick 4
End Sub

Sub B_RA1_Click()
    ButClick 5
End Sub

Sub B_RA2_Click()
    ButClick 6
End Sub

Sub B_QUEUES_Click()
    ButClick 1
End Sub

```

```
Public Sub B_BUFFERS_Click()  
    ButtClick 2  
End Sub  
  
Private Sub B_BACK_Click()  
    Unload Me  
End Sub  
  
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)  
    Select Case KeyCode  
        Case 39 ' Direita  
            ButtBRInfo(SelInd14).BackColor = Gray  
            If (SelInd14 < 7) Then  
                SelInd14 = SelInd14 + 1  
            Else  
                SelInd14 = 1  
            End If  
            ButtBRInfo(SelInd14).BackColor = vbMagenta  
        Case 37 ' Esquerda  
            ButtBRInfo(SelInd14).BackColor = Gray  
            If (SelInd14 > 1) Then  
                SelInd14 = SelInd14 - 1  
            Else  
                SelInd14 = 7  
            End If  
            ButtBRInfo(SelInd14).BackColor = vbMagenta  
        Case 13 ' Enter  
            ButtClick SelInd14  
    End Select  
End Sub  
  
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)  
    SwitchToForm Me, Monitor_BR_Form  
    For i = 1 To 5  
        ButtBRInfo(i).BackColor = Gray  
        ButtBRInfo(i).Enabled = True  
    Exit For  
    Next i  
    Cancel = True  
    Me.Hide  
End Sub  
  
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)  
    Form_KeyDown KeyCode, Shift  
End Sub
```

A.8. Select_CAM_Info_Form

A.8.1. Layout da Select_CAM_Info_Form

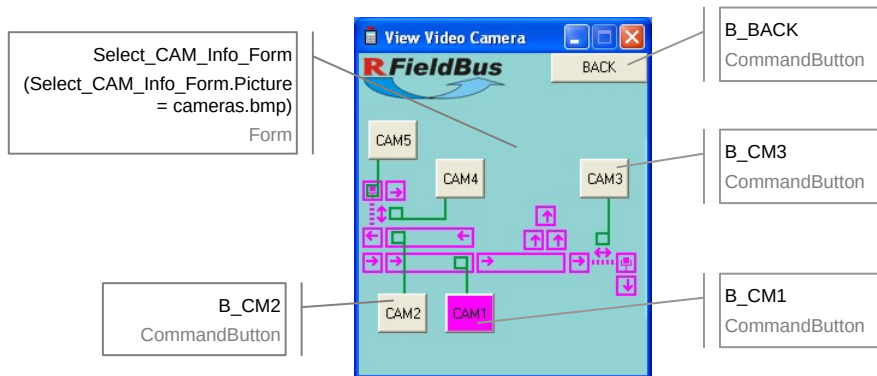


Figura A.6: Layout da Select_CAM_Info_Form

A.8.2. Objectos relevantes da Select_CAM_Info_Form

Tabela A.6: Relação de objectos relevantes da Select_CAM_Info_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
B_CM1.. B_CM5	CommandButton	Ir para CAM_View_Form, contexto CAM1 (...CAM5)
Select_CAM_Info_Form	Form	
Text1	TextBox	Utilizado para navegação alternativa com teclado (ver Cap. 5)

A.8.3. Código da Select_CAM_Info_Form

```
Option Explicit
Public SelInd13 As Integer

Private Sub Form_Load()
    Set FormList(1) = Me
    Set ButtSelectCAMInfo(1) = B_CM1
    Set ButtSelectCAMInfo(2) = B_CM2
    Set ButtSelectCAMInfo(3) = B_CM3
    Set ButtSelectCAMInfo(4) = B_CM4
    Set ButtSelectCAMInfo(5) = B_CM5
    Set ButtSelectCAMInfo(6) = B_BACK

    SelInd13 = 1
    ButtSelectCAMInfo(SelInd13).BackColor = vbMagenta
End Sub
```

```

Private Sub ButtClick(Index As Integer)
    For i = 1 To 6
        ButtSelectCAMInfo(i).Enabled = True
        ButtSelectCAMInfo(i).BackColor = Gray
    Next i
    SelInd13 = Index
    SwitchToForm Me, BR_Info_Form
    Select Case Index
    Case 1
        B_CM1_Click
    Case 2
        B_CM2_Click
    Case 3
        B_CM3_Click
    Case 4
        B_CM4_Click
    Case 5
        B_CM5_Click
    Case 6
        B_BACK_Click
    End Select
End Sub

Function SetButton1(i)
    ButtSelectCAMInfo(i).BackColor = vbGreen
    SelInd13 = i
    For i = 1 To 6
        If i <> SelInd13 Then
            ButtSelectCAMInfo(i).BackColor = Gray
        End If
    Next i
End Function

Private Sub Form_KeyDown(Keycode As Integer, Shift As Integer)
    Select Case Keycode
    Case 39 ' Direita
        ButtSelectCAMInfo(SelInd13).BackColor = Gray
        If SelInd13 < 6 Then
            SelInd13 = SelInd13 + 1
        Else
            SelInd13 = 1
        End If
        ButtSelectCAMInfo(SelInd13).BackColor = vbMagenta
    Case 37 ' Esquerda
        ButtSelectCAMInfo(SelInd13).BackColor = Gray
        If SelInd13 > 1 Then
            SelInd13 = SelInd13 - 1
        Else
            SelInd13 = 6
        End If
        ButtSelectCAMInfo(SelInd13).BackColor = vbMagenta
    Case 13 ' Enter
        ButtClick SelInd13
    End Select
End Sub

Sub B_CM1_Click()
    SwitchToForm Me, CAM_View_Form
    CAM_View_Form.ButtClick 1
End Sub

Sub B_CM2_Click()
    SwitchToForm Me, CAM_View_Form
    CAM_View_Form.ButtClick 2
End Sub

```

```

Sub B_CM3_Click()
    SwitchToForm Me, CAM_View_Form
    CAM_View_Form.ButtClick 3
End Sub

Sub B_CM4_Click()
    SwitchToForm Me, CAM_View_Form
    CAM_View_Form.ButtClick 4
End Sub

Sub B_CM5_Click()
    SwitchToForm Me, CAM_View_Form
    CAM_View_Form.ButtClick 5
End Sub

Private Sub B_BACK_Click()
    Unload Me
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, Monitor_Form
    For i = 1 To 6
        ButtSelectCAMInfo(i).Enabled = True
        ButtSelectCAMInfo(i).BackColor = Gray
    Next i
    Cancel = True
    Me.Hide
End Sub

Private Sub Text1_KeyDown(Keycode As Integer, Shift As Integer)
    Form_KeyDown Keycode, Shift
End Sub

```

A.9. CAM_View_Form

A.9.1. Layout da CAM_View_Form

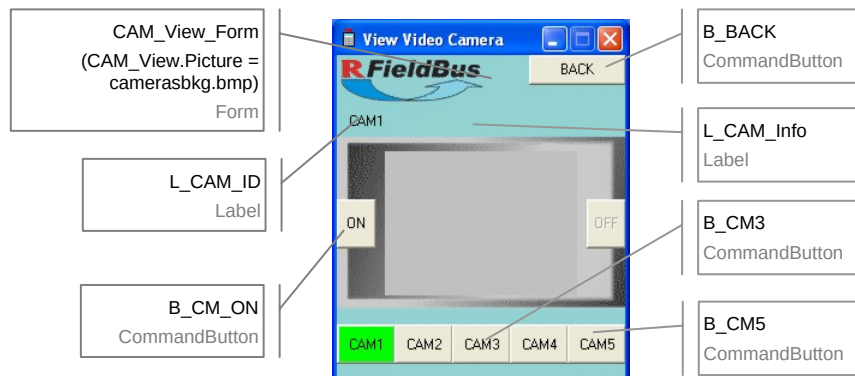


Figura A.7: Layout da CAM_View_Form

A.9.2. Objectos relevantes da CAM_View_Form

Tabela A.7: Relação de objectos relevantes da CAM_View_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
B_CM_OFF	CommandButton	Para desactivar a CAM de contexto, envia Wsk1
B_CM_ON	CommandButton	Para activar a CAM de contexto, envia Wsk1
B_CM1.. B_CM5	CommandButton	Fica em CAM_View_Form, contexto CAM1 (...CAM5)
CamPic	PictureBox	É actualizada (se activa) por Wsk1, ou variável cam(i)
L_CAM_ID	Label	É actualizada com B_CMx_Click()
L_CAM_Info	Label	É actualizada (se activa) por Wsk1, ou vari. g_cam_txt(i)
CAM_View_Form	Form	
Text1	TextBox	Utilizado para navegação alternativa com teclado (ver Cap. 5)

A.9.3. Código da CAM_View_Form

```

Option Explicit
Public SelInd15 As Integer
Public SelInd15_b As Integer
Dim selected_cam As Integer

Private Sub Form_Load()
    Set FormList(3) = Me

    Set ButtCAMView(1) = B_CM1
    Set ButtCAMView(2) = B_CM2
    Set ButtCAMView(3) = B_CM3
    Set ButtCAMView(4) = B_CM4
    Set ButtCAMView(5) = B_CM5
    Set ButtCAMView(6) = B_CM_ON
    Set ButtCAMView(7) = B_CM_OFF
    Set ButtCAMView(8) = B_BACK
End Sub

Private Sub Form_Unload(Cancel As Integer)
    End
End Sub

Private Sub B_CM_OFF_Click()
    Dim datapcx As IPictureDisp
    'limpa a image (coloca imagem NO FEED
    Set datapcx = LoadPicture(App.Path & "\cams\novideo.jpg")
    Set cam(selected_cam) = datapcx
    g_txt_cam(selected_cam) = ""
    Set CamPic.Picture = datapcx
    L_CAM_Info.Caption = g_txt_cam(selected_cam)
    'Envia o pedido de OFF ao servidor
    g_cam_enable(selected_cam) = False
    Main_Form.Wsk1.RemotePort = SERVER_PORT
    Main_Form.Wsk1.RemoteHost = SERVER_ADDR
    Main_Form.Wsk1.SendData ">" & "CAM" & selected_cam & ":OFF"
    'faz o toggle do enable dos botões
    B_CM_ON.Enabled = True
    B_CM_OFF.Enabled = False
End Sub

```

```

Private Sub B_CM_ON_Click()
    'faz SÓ um pedido de ON ao servidor
    Main_Form.Wsk1.RemotePort = SERVER_PORT
    Main_Form.Wsk1.RemoteHost = SERVER_ADDR
    Main_Form.Wsk1.SendData ">" & "CAM" & selected_cam & ":ON"
    cam_req(selected_cam) = True
End Sub

Private Sub B_CM1_Click()
    ButtClick 1
End Sub

Private Sub B_CM2_Click()
    ButtClick 2
End Sub

Private Sub B_CM3_Click()
    ButtClick 3
End Sub

Private Sub B_CM4_Click()
    ButtClick 4
End Sub

Private Sub B_CM5_Click()
    ButtClick 5
End Sub

Private Sub B_BACK_Click()
    Unload Me
End Sub

Public Sub ButtClick(Index As Integer)
    If Index < 6 Then
        SelInd15 = Index
        For i = 1 To 5
            ButtCAMView(i).Enabled = True: ButtCAMView(i).BackColor = Gray
        Next i
        ButtCAMView(Index).BackColor = vbGreen
        'cam(Index) contem as imagens
        Set CamPic.Picture = cam(Index)
        L_CAM_ID.Caption = "CAM" & Index
        L_CAM_Info.Caption = g_txt_cam(Index)
        If g_cam_enable(Index) Then
            B_CM_ON.Enabled = False: B_CM_OFF.Enabled = True
        Else
            B_CM_ON.Enabled = True: B_CM_OFF.Enabled = False
        End If
        Select_CAM_Info_Form.SetButton1 Index
        SelInd15_b = SelInd15
        selected_cam = SelInd15
    Else
        Select Case Index
            Case 6
                If B_CM_ON.Enabled = True Then
                    B_CM_ON_Click
                End If
            Case 7
                If B_CM_OFF.Enabled = True Then
                    B_CM_OFF_Click
                End If
            Case 8
                B_BACK_Click
        End Select
    End If
End Sub

```

```

Private Sub Form_KeyDown(Keycode As Integer, Shift As Integer)
    Select Case Keycode
    Case 39 ' Direita
        If ButtCAMView(SelInd15_b).BackColor <> vbGreen Then
            ButtCAMView(SelInd15_b).BackColor = Gray
        End If

        If (SelInd15_b < 8) Then
            SelInd15_b = SelInd15_b + 1
        Else
            SelInd15_b = 1
        End If
        If ButtCAMView(SelInd15_b).BackColor <> vbGreen Then
            ButtCAMView(SelInd15_b).BackColor = vbMagenta
        End If
    Case 37 ' Esquerda
        If ButtCAMView(SelInd15_b).BackColor <> vbGreen Then
            ButtCAMView(SelInd15_b).BackColor = Gray
        End If
        If (SelInd15_b > 1) Then
            SelInd15_b = SelInd15_b - 1
        Else
            SelInd15_b = 8
        End If
        If ButtCAMView(SelInd15_b).BackColor <> vbGreen Then
            ButtCAMView(SelInd15_b).BackColor = vbMagenta
        End If
    Case 13 ' Enter
        ButtClick SelInd15_b
    End Select
End Sub

Private Sub Text1_KeyDown(Keycode As Integer, Shift As Integer)
    Form_KeyDown Keycode, Shift
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, Select_CAM_Info_Form
    For i = 1 To 8
        ButtCAMView(i).BackColor = Gray
        ButtCAMView(i).Enabled = True
    Next i
    Cancel = True
    Me.Hide
End Sub

```

A.10. Control_Form

A.10.1. Layout da Control_Form

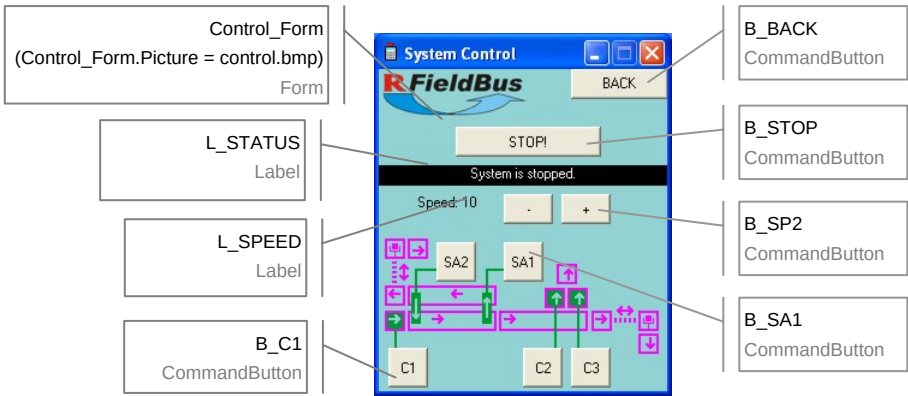


Figura A.8: Layout da Control_Form

A.10.2. Objectos relevantes da Control_Form

Tabela A.8: Relação de objectos relevantes da Control_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
B_C1;B_C2;B_C3	CommandButton	Envia (Wsk1) comando de activação cilindro C1 (C2, C3)
B_SA1;B_SA2	CommandButton	Envia (Wsk1) comando de activação Swivel Arm SA1 (SA2)
B_SP1; B_SP2	CommandButton	Envia (Wsk1) aumento (B_SP1) diminuição (B_SP2) de veloc.
B_STOP	CommandButton	Envia (Wsk1) pedido paragem do SFF
Control_Form	Form	
L_SPEED	Label	Visualização da velocidade dos tapetes (recebido via Wsk1)
L_STATUS	Label	Visualização do estado do sistema (recebido via Wsk1)
Text1	TextBox	Utilizado para navegação alternativa com o teclado (ver Cap. 5)
Timer1	Timer	Utilizado para manipular a cor dos botões de envio via Wsk1

A.10.3. Código da Control_Form

```
Option Explicit
Public SelInd As Integer
Public but As Integer
```

```

Private Sub Form_Load()
    Set FormList(5) = Me

    ' Precisa destes para os habilitar/desabilitar de outros forms (por exemplo
    ' em funcao do que recebe em wsk1
    Set ButtControl(1) = B_STOP
    Set ButtControl(2) = B_SP1
    Set ButtControl(3) = B_SP2
    Set ButtControl(4) = B_SA1
    Set ButtControl(5) = B_SA2
    Set ButtControl(6) = B_C1
    Set ButtControl(7) = B_C2
    Set ButtControl(8) = B_C3
    Set ButtControl(9) = B_Back

    SelInd = 1
    ButtControl(SelInd).BackColor = vbMagenta
End Sub

Private Sub B_BACK_Click()
    Unload Me
End Sub

Private Sub B_SA1_Click()
    Send_Command "#SA1:ACTIVATE", 4
End Sub

Private Sub B_SA2_Click()
    Send_Command "#SA2:ACTIVATE", 5
End Sub

Private Sub B_C1_Click()
    Send_Command "#CYL1:ACTIVATE", 6
End Sub

Private Sub B_C2_Click()
    Send_Command "#CYL2:ACTIVATE", 7
End Sub

Private Sub B_C3_Click()
    Send_Command "#CYL3:ACTIVATE", 8
End Sub

Private Sub B_SP1_Click()
    Send_Command "#CB_SPEED:SLOWER", 2
End Sub

Private Sub B_SP2_Click()
    Send_Command "#CB_SPEED:FASTER", 3
End Sub

Private Sub B_STOP_Click()
    Send_Command "#OP:EM_STOP", 1
End Sub

Private Sub Send_Command(mess As String, i As Integer)
    Timer1.Enabled = True
    but = i
    ButtControl(but).BackColor = vbYellow
    Main_Form.Wsk1.RemotePort = SERVER_PORT
    Main_Form.Wsk1.RemoteHost = SERVER_ADDR
    Main_Form.Wsk1.SendData mess
End Sub

```

```

Private Sub Form_KeyDown(Keycode As Integer, Shift As Integer)
    Select Case Keycode
        Case 39 ' Direita
            ButtControl(SelInd).BackColor = Gray
            If ButtControl(2).Enabled = True Then
                If (SelInd < 9) Then
                    SelInd = SelInd + 1
                Else
                    SelInd = 1
                End If
            Else
                If SelInd = 1 Then
                    SelInd = 9
                Else
                    SelInd = 1
                End If
            End If
            ButtControl(SelInd).BackColor = vbMagenta
        Case 37 ' Esquerda
            ButtControl(SelInd).BackColor = Gray
            If ButtControl(2).Enabled = True Then
                If (SelInd > 1) Then
                    SelInd = SelInd - 1
                Else
                    SelInd = 9
                End If
            Else
                If SelInd = 1 Then
                    SelInd = 9
                Else
                    SelInd = 1
                End If
            End If
            ButtControl(SelInd).BackColor = vbMagenta
        Case 13 ' Enter
            ButtClick SelInd
    End Select
End Sub

Private Sub ButtClick(Index As Integer)
    For i = 1 To 8
        'ButtControl(i).Enabled = True:
        ButtControl(i).BackColor = Gray
    Next i
    ButtControl(Index).BackColor = vbGreen
    SelInd = Index
    Select Case Index
        Case 1
            B_STOP_Click
        Case 2
            B_SP1_Click
        Case 3
            B_SP2_Click
        Case 4
            B_SA1_Click
        Case 5
            B_SA2_Click
        Case 6
            B_C1_Click
        Case 7
            B_C2_Click
        Case 8
            B_C3_Click
        Case 9
            B_BACK_Click
    End Select
End Sub

```

```

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, Main_Form
    Cancel = True
    Me.Hide
End Sub

Private Sub Timer1_Timer()
    ButtControl(but).BackColor = vbMagenta
    Timer1.Enabled = False
End Sub

Private Sub Text1_KeyDown(Keycode As Integer, Shift As Integer)
    Form_KeyDown Keycode, Shift
End Sub

```

A.11. Config_Form

A.11.1. Layout da Config_Form

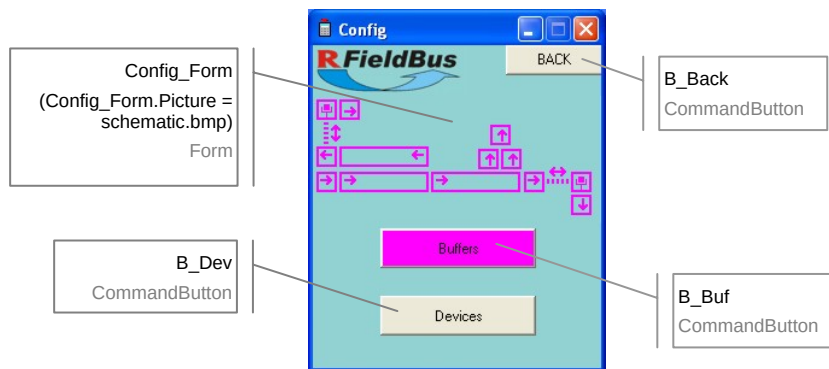


Figura A.9: Layout da Config_Form das aplicações cliente *Intranet*

A.11.2. Objectos relevantes da Config_Form

Tabela A.9: Relação de objectos relevantes da Config_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
B_Buf	CommandButton	Para ir para a janela Config_Buffers_Form
B_Dev	CommandButton	Para ir para a janela Config_Devices_Form
Config_Form	Form	
Text1	TextBox	Utilizado para navegação alternativa com o teclado (ver Cap. 5)

A.11.3. Código da Config_Form

```

Option Explicit
Public SelInd As Integer

Private Sub Form_Load()
    Set FormList(10) = Me
    Set ButtConfig(1) = B_Buf
    Set ButtConfig(2) = B_Dev
    Set ButtConfig(3) = B_Back
    SelInd = 1
    ButtConfig(SelInd).BackColor = vbMagenta
End Sub

Private Sub ButtClick(Index As Integer)
    ButtConfig(Index).BackColor = vbMagenta
    SelInd = Index
    Select Case Index
        Case 1
            SwitchToForm Me, Config_Buffers_Form
        Case 2
            SwitchToForm Me, Config_Devices_Form
        Case 3
            Unload Me
    End Select
End Sub

Private Sub B_BACK_Click()
    ButtClick (3)
End Sub

Private Sub B_Buf_Click()
    ButtClick (1)
End Sub

Private Sub B_Dev_Click()
    ButtClick (2)
End Sub

Private Sub Form_KeyDown(Keycode As Integer, Shift As Integer)
    Select Case Keycode
        Case 39 ' Direita
            ButtConfig(SelInd).BackColor = Gray
            If SelInd < 3 Then
                SelInd = SelInd + 1
            Else
                SelInd = 1
            End If
            ButtConfig(SelInd).BackColor = vbMagenta
        Case 37 ' Esquerda
            ButtConfig(SelInd).BackColor = Gray
            If SelInd > 1 Then
                SelInd = SelInd - 1
            Else
                SelInd = 3
            End If
            ButtConfig(SelInd).BackColor = vbMagenta
        Case 13 ' Enter
            ButtClick SelInd
    End Select
End Sub

```

```

Private Sub Form_OKClick()
    End
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, Main_Form
    Cancel = True
    Me.Hide
End Sub

Private Sub Text1_KeyDown(Keycode As Integer, Shift As Integer)
    Form_KeyDown Keycode, Shift
End Sub

Private Sub B_BACK_GotFocus()
    For i = 1 To 3
        ButtConfig(i).BackColor = Gray
    Next i
    SelInd = 3
    ButtConfig(3).BackColor = vbMagenta
End Sub

Private Sub B_Buf_GotFocus()
    For i = 1 To 3
        ButtConfig(i).BackColor = Gray
    Next i
    SelInd = 1
    ButtConfig(1).BackColor = vbMagenta
End Sub

Private Sub B_Dev_GotFocus()
    For i = 1 To 3
        ButtConfig(i).BackColor = Gray
    Next i
    SelInd = 2
    ButtConfig(2).BackColor = vbMagenta
End Sub

```

A.12. Config_Buffers_Form

A.12.1. Layout da Config_Buffers_Form

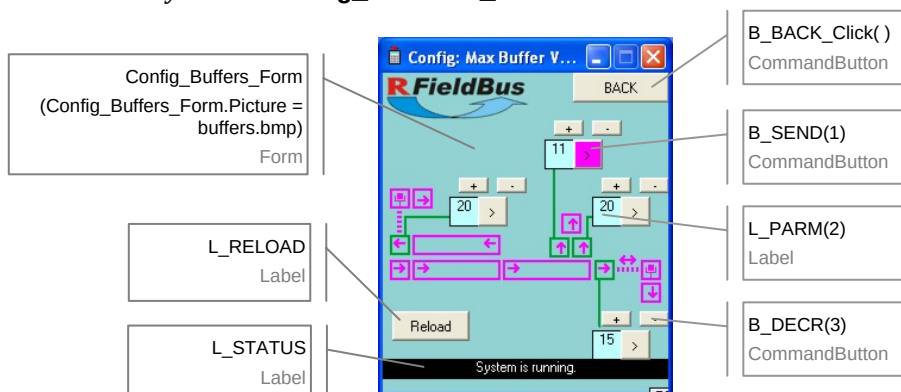


Figura A.10: Layout da Config_Buffers_Form

A.12.2. Objectos relevantes da Config_Buffers_Form

Tabela A.10: Relação de objectos relevantes da Config_Buffers_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
B_DECR(i)	CommandButton	Decrementa em uma unidade o valor de L_PARM(i).Caption
B_INCR(i)	CommandButton	Incrementa em uma unidade o valor de L_PARM(i).Caption
B_RELOAD	CommandButton	Actualiza as L_PARM(i) com os valores da variável info_bufmax(i), actualizada por Wsk1
B_SEND(i)	CommandButton	Envia (Wsk1) novo valor de configuração dos BUFFERS
Config_Buffers_Form	Form	
L_PARM(i)	Label	Label actualizado ou por Wsk1 (se janela Config_Buffers_Form) ou actualizado por leitura da variável info_bufmax(i) (no B_RELOAD_Click () ou Load da janela
L_STATUS	Label	Visualização do estado do sistema (recebido via Wsk1)
Text1	TextBox	Utilizado para navegação alternativa com o teclado (ver Cap. 5)
Timer1	Timer	Utilizado para manipular a cor dos botões de envio via Wsk1

A.12.3. Código da Config_Buffers_Form

```

Option Explicit
Public SelInd6 As Integer
Dim sel_but As Integer

Private Sub B_INCR_Click(Index As Integer)
Dim i As Integer
SelInd6 = Index
For i = 1 To 7
    ButtConfBuffers(i).BackColor = Gray
Next i
ButtConfBuffers(SelInd6).BackColor = vbMagenta
L_PARM(SelInd6) = L_PARM(SelInd6) + 1
Text1.SetFocus
End Sub

Private Sub B_DECR_Click(Index As Integer)
Dim i As Integer
SelInd6 = Index
For i = 1 To 7
    ButtConfBuffers(i).BackColor = Gray
Next i
ButtConfBuffers(SelInd6).BackColor = vbMagenta
L_PARM(SelInd6) = L_PARM(SelInd6) - 1
Text1.SetFocus
End Sub

```

```

Private Sub Form_Load()
    Dim i As Integer
    Set FormList(6) = Me

    'Pede a leitura dos valores máximos dos buffers
    Main_Form.Wsk1.RemotePort = SERVER_PORT
    Main_Form.Wsk1.RemoteHost = SERVER_ADDR
    For i = 1 To 5
        Main_Form.Wsk1.SendData "&B" & i & "MAX"
    Next i

    For i = 1 To 5
        Set ButtConfBuffers(i) = B_SEND(i)
    Next i

    Set ButtConfBuffers(6) = B_RELOAD
    Set ButtConfBuffers(7) = B_BACK

    SelInd6 = 1
    ButtConfBuffers(SelInd6).BackColor = vbMagenta
End Sub

Private Sub ButClick(Index As Integer)
    SelInd6 = Index
    For i = 1 To 7
        ButtConfBuffers(i).BackColor = Gray
    Next
    ButtConfBuffers(SelInd6).BackColor = vbMagenta
    Select Case Index
        Case 1, 2, 3, 5
            B_SEND_Click (Index)
        Case 6
            B_RELOAD_Click
        Case 7
            B_BACK_Click
    End Select
End Sub

Private Sub B_RELOAD_Click()
    Dim i As Integer
    For i = 1 To 5
        Config_Buffers_Form.L_PARM(i) = info_bufmax(i)
    Next
End Sub

Private Sub B_SEND_Click(Index As Integer)
    ButtConfBuffers(Index).BackColor = vbYellow
    sel_but = Index
    Timer1.Enabled = True
    Main_Form.Wsk1.RemotePort = SERVER_PORT
    Main_Form.Wsk1.RemoteHost = SERVER_ADDR
    Main_Form.Wsk1.SendData "&B" & Index & "MAX:" & L_PARM(Index)
End Sub

Private Sub B_BACK_Click()
    Unload Me
End Sub

```

```

Private Sub Form_KeyDown(Keycode As Integer, Shift As Integer)
    Select Case Keycode
    Case 39 ' Direita
        ButtConfBuffers(SelInd6).BackColor = Gray
        If (SelInd6 < 7) Then
            SelInd6 = SelInd6 + 1
            If SelInd6 = 4 Then SelInd6 = 5
        Else
            SelInd6 = 1
        End If
        ButtConfBuffers(SelInd6).BackColor = vbMagenta
    Case 37 ' Esquerda
        ButtConfBuffers(SelInd6).BackColor = Gray
        If (SelInd6 > 1) Then
            SelInd6 = SelInd6 - 1
            If SelInd6 = 4 Then SelInd6 = 3
        Else
            SelInd6 = 7
        End If
        ButtConfBuffers(SelInd6).BackColor = vbMagenta
    Case 107, 187 '+'
        If SelInd6 < 6 Then
            Me.L_PARM(SelInd6) = Me.L_PARM(SelInd6) + 1
        End If
    Case 109, 189 '-'
        If SelInd6 < 6 Then
            Me.L_PARM(SelInd6) = Me.L_PARM(SelInd6) - 1
        End If

    Case 13 ' Enter
        ButtClick SelInd6
    End Select
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, Config_Form
    Cancel = True
    Me.Hide
End Sub

Private Sub Text1_KeyDown(Keycode As Integer, Shift As Integer)
    Form_KeyDown Keycode, Shift
End Sub

Private Sub Timer1_Timer()
    Dim i As Integer
    Timer1.Enabled = False
    For i = 1 To 6
        If i = sel_but Then
            ButtConfBuffers(i).BackColor = vbMagenta
        Else
            ButtConfBuffers(i).BackColor = Gray
        End If
    Next i
End Sub

```

A.13. Config_Devices_Form

A.13.1. Layout da Config_Devices_Form

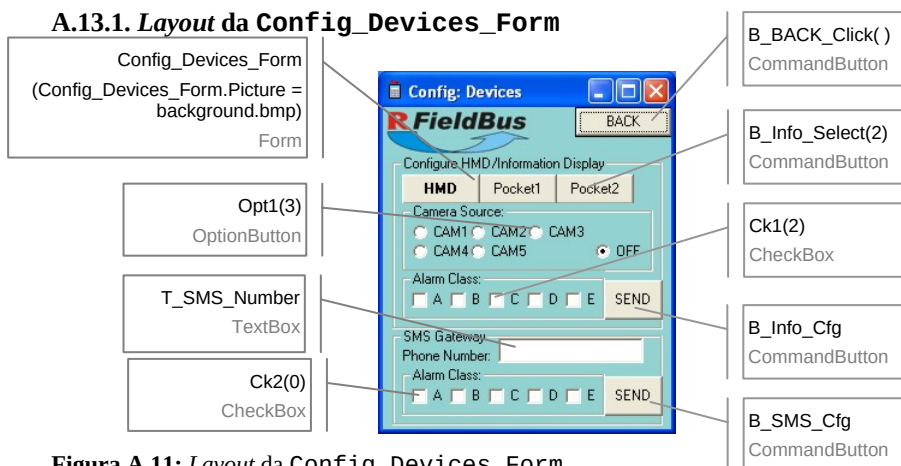


Figura A.11: Layout da Config_Devices_Form

A.13.2. Objectos relevantes da Config_Devices_Form

Tabela A.11: Relação de objectos relevantes da Config_Devices_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
B_Info_Cfg	CommandButton	Envia configuração (via Wsk1) de HMD, PK1, PK2, consoante o contexto activo
B_Info_Select(i)	CommandButton	Selecciona contexto (HMD, PK1, PK2)
B_SMS_Cfg	CommandButton	Envia configuração (via Wsk1) à Gateway SMS
Ck1(i)	CheckBox	Classes de alarmes/eventos a ser processados por HMD, PK1, PK2
Ck2(i)	CheckBox	Classes de alarmes/eventos a ser enviados automaticamente via SMS
Config_Devices_Form	Form	
Device, Frame1..3	Frame	
Label1		= "Phone Number:"
Opt1(i)	OptionButton	Seleccção de CAM a visualizar por HMD/PK1/PK2
T_SMS_Number	TextBox	
Timer1	Timer	Para manipular a cor dos botões de envio via Wsk1

A.13.3. Código da Config_Devices_Form

```

Option Explicit
Public SelInd7 As Integer

Dim g_opt1 As Integer
Dim g_ck1 As String
Dim g_ck2 As String

Dim g_button As Integer
Dim which_send As Integer

Private Sub Form_Load()
    Set FormList(7) = Me

    For i = 1 To 3
        If B_Info_Select(i).Enabled = True Then
            SelInd7 = i
            Exit For
        End If
    Next i

    Set ButtConfDevices(1) = B_Info_Cfg
    Set ButtConfDevices(2) = B_SMS_Cfg
    Set ButtConfDevices(3) = B_Info_Select(0)
    Set ButtConfDevices(4) = B_Info_Select(1)
    Set ButtConfDevices(5) = B_Info_Select(2)
    Set ButtConfDevices(6) = B_Back
End Sub

Private Sub B_Back_GotFocus()
    B_Back.BackColor = vbMagenta
End Sub

Private Sub B_Back_LostFocus()
    B_Back.BackColor = Gray
End Sub

Private Sub B_Info_Cfg_GotFocus()
    B_Info_Cfg.BackColor = vbMagenta
End Sub

Private Sub B_Info_Cfg_LostFocus()
    B_Info_Cfg.BackColor = Gray
End Sub

Private Sub B_Info_Select_GotFocus(Index As Integer)
    ButtConfDevices(Index + 3).BackColor = vbMagenta
End Sub

Private Sub B_Info_Select_LostFocus(Index As Integer)
    ButtConfDevices(Index + 3).BackColor = Gray
End Sub

Private Sub B_SMS_Cfg_GotFocus()
    B_SMS_Cfg.BackColor = vbMagenta
End Sub

Private Sub B_BACK_Click()
    Unload Me
End Sub

Private Sub B_SMS_Cfg_LostFocus()
    B_SMS_Cfg.BackColor = Gray
End Sub

```

```

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, Config_Form
    Cancel = True
    Me.Hide
End Sub

Private Sub Ck2_Click(Index As Integer)
    g_ck2 = Index
End Sub

Private Sub B_Info_Select_Click(Index As Integer)
    Dim i As Integer
    For i = 0 To 2
        B_Info_Select(i).FontBold = Index = i
    Next i
    g_button = Index

    For i = 0 To 4
        If InStr(info_alarm(g_button), Ck1(i).Caption) > 0 Then
            Ck1(i).Value = 1
        Else
            Ck1(i).Value = 0
        End If
    Next i
    Opt1(info_cam(g_button)).Value = True

    B_Info_Cfg.Enabled = True
    B_Info_Cfg.FontBold = False
End Sub

Private Sub Ck1_Click(Index As Integer)
    Dim fChanged As Boolean

    For i = 0 To 4
        If InStr(info_alarm(g_button), Ck1(i).Caption) > 0 Then
            If Ck1(i).Value <> 1 Then fChanged = True
        Else
            If Ck1(i).Value <> 0 Then fChanged = True
        End If
    Next i
    If Index <> info_cam(g_button) Then
        fChanged = True
    End If

    B_Info_Cfg.Font.Bold = fChanged
End Sub

Private Sub Opt1_Click(Index As Integer)
    Dim fChanged As Boolean

    g_opt1 = Index

    For i = 0 To 4
        If InStr(info_alarm(g_button), Ck1(i).Caption) > 0 Then
            If Ck1(i).Value <> 1 Then fChanged = True
        Else
            If Ck1(i).Value <> 0 Then fChanged = True
        End If
    Next i
    If Index <> info_cam(g_button) Then
        fChanged = True
    End If

    B_Info_Cfg.Font.Bold = fChanged
End Sub

```

```

Private Sub B_Info_Cfg_Click()
Dim i As Integer
Dim head_msg As String

    g_ck1 = ""
    For i = 0 To 4
        If Ck1(i).Value = 1 Then
            g_ck1 = g_ck1 & Ck1(i).Caption
        End If
    Next i
    which_send = 1
    Timer1.Enabled = True
    B_Info_Cfg.BackColor = vbYellow

    Select Case g_button
        Case 0 'HMD
            head_msg = "%HMD:"
        Case 1 'Pocket1
            head_msg = "%PK1:"
            Main_Form.Wsk1.SendData "%PK1:" & g_ck1 & ":" & g_opt1
        Case 2 'Pocket2
            head_msg = "%PK2:"
    End Select
    'envia para os clientes e servidor
    Main_Form.Wsk1.RemotePort = SERVER_PORT
    Main_Form.Wsk1.RemoteHost = SERVER_ADDR
    Main_Form.Wsk1.SendData head_msg & g_ck1 & ":" & g_opt1
    Main_Form.Wsk1.RemotePort = CLIENT_PORT
    Main_Form.Wsk1.RemoteHost = BROADCAST_ADDR
    Main_Form.Wsk1.SendData head_msg & g_ck1 & ":" & g_opt1

    info_alarm(g_button) = g_ck1
    info_cam(g_button) = g_opt1
    B_Info_Cfg.Font.Bold = False
End Sub

Private Sub B_SMS_Cfg_Click()
Dim i As Integer
g_ck2 = ""
For i = 0 To 4
    If Ck2(i).Value = 1 Then
        g_ck2 = g_ck2 & Ck2(i).Caption
    End If
Next i
which_send = 2
Timer1.Enabled = True
B_SMS_Cfg.BackColor = vbYellow
'envia para servidor
Main_Form.Wsk1.RemotePort = SERVER_PORT
Main_Form.Wsk1.RemoteHost = SERVER_ADDR
Main_Form.Wsk1.SendData "%SMS" & ":" & g_ck2 & ":" & T_SMS_Number.Text
'envia para clientes
Main_Form.Wsk1.RemotePort = CLIENT_PORT
Main_Form.Wsk1.RemoteHost = BROADCAST_ADDR
Main_Form.Wsk1.SendData "%SMS" & ":" & g_ck2 & ":" & T_SMS_Number.Text
End Sub

Private Sub Timer1_Timer()
Timer1.Enabled = False
Select Case which_send
    Case 1
        B_Info_Cfg.BackColor = vbMagenta
    Case 2
        B_SMS_Cfg.BackColor = vbMagenta
End Select
End Sub

```

A.14. Gen_Events_Form

A.14.1. Layout da Gen_Events_Form

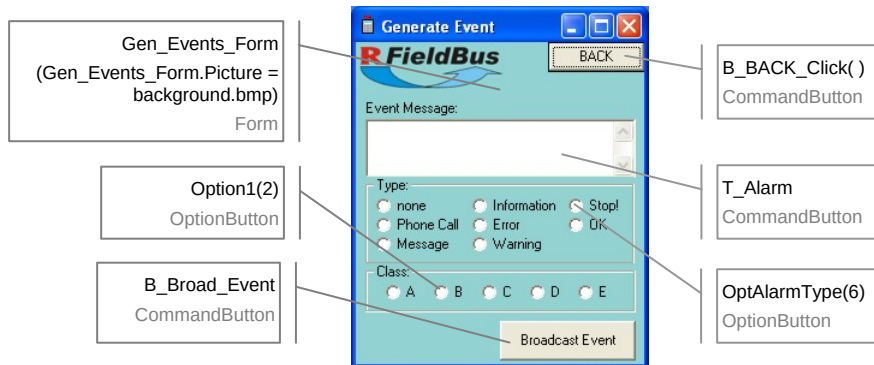


Figura A.12: Layout da Gen_Events_Form

A.14.2. Objectos relevantes da Gen_Events_Form

Tabela A.12: Relação de objectos relevantes da Gen_Events_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
B_Broad_Event	CommandButton	Envia (via Wsk1) evento/alarme ("!EVT ...") de acordo com as configurações da janela
Frame1, Frame3	Frame	
Gen_Events_Form	Form	
Label1		= "Event Message:"
OptAlarmType(i)	OptionButton	Selecciona tipo de evento
Option1(i)	OptionButton	Selecciona classe de evento
T_Alarm	TextBox	Texto associado ao evento/alarme a enviar
Timer1	Timer	Para manipular a cor dos botões de envio via Wsk1

A.14.3. Código da Gen_Events_Form

```
Option Explicit
Private g_option1 As String
Private g_alarm_type As String
```

```
Private Sub Form_Load()  
    Set FormList(8) = Me  
End Sub  
  
Private Sub B_BACK_Click()  
    Unload Me  
End Sub  
  
Private Sub B_Back_GotFocus()  
    B_Back.BackColor = vbMagenta  
End Sub  
  
Private Sub B_Back_LostFocus()  
    B_Back.BackColor = Gray  
End Sub  
  
Private Sub B_Broad_Event_Click()  
    Timer1.Enabled = True  
    B_Broad_Event.BackColor = vbYellow  
    Main_Form.Wsk1.RemotePort = CLIENT_PORT  
    Main_Form.Wsk1.RemoteHost = BROADCAST_ADDR  
    Main_Form.Wsk1.SendData "!EVT" & ":" & g_option1 & ":" & g_alarm_type & ":" &  
        T_Alarm  
End Sub  
  
Private Sub B_Broad_Event_GotFocus()  
    B_Broad_Event.BackColor = vbMagenta  
End Sub  
  
Private Sub B_Broad_Event_LostFocus()  
    B_Broad_Event.BackColor = Gray  
End Sub  
  
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)  
    SwitchToForm Me, Main_Form  
    Cancel = True  
    Me.Hide  
End Sub  
  
Private Sub optAlarmType_Click(Index As Integer)  
    g_alarm_type = optAlarmType(Index).Tag  
End Sub  
  
Private Sub Option1_Click(Index As Integer)  
    g_option1 = Option1(Index).Caption  
End Sub  
  
Private Sub Timer1_Timer()  
    Timer1.Enabled = False  
    B_Broad_Event.BackColor = vbMagenta  
End Sub
```

A.15. Show_Events_Form

A.15.1. Layout da Show_Events_Form

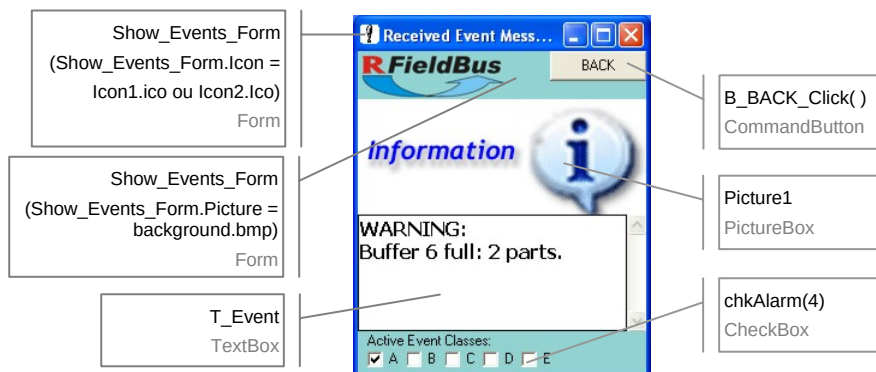


Figura A.13: Layout da Show_Events_Form

A.15.2. Objectos relevantes da Show_Events_Form

Tabela A.13: Relação de objectos relevantes da Show_Events_Form

Objecto (Nome)	Tipo	Observações
B_Back	CommandButton	Para ir para a janela anterior
chkAlarm (i)	OptionButton	Selecciona as classes de eventos/alarmes que se pretendem visualizar no cliente
Label1		= "Active Event Classes;"
Picture1	PictureBox	Em função do evento recebido (ver ShowAlarm(id, msg) no Globals.bas) é carregado o ficheiro error.bmp, info.bmp, ok.bmp, phone.bmp, stop.bmp ou wait.bmp (default).
Picture2	PictureBox	= "Icon1.ico", está invisível
Picture3	PictureBox	= "Icon2.ico", está invisível
Show_Events_Form	Form	
T_Event	TextBox	Texto associado ao evento/alarme recebido (via Wsk1)
Timer1	Timer	Para fazer o toggle do icon da janela (entre Icon1.ico e Icon2.ico)

A.15.3. Código da Show_Events_Form

```
Option Explicit
Public g_lastForm As Form

Private Sub B_BACK_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    Set FormList(9) = Me
End Sub

Private Sub chkAlarm_Click(Index As Integer)
    Main_Form.chkAlarm(Index).Value = chkAlarm(Index).Value
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    SwitchToForm Me, g_lastForm
    Cancel = True
    Timer1.Enabled = False
    Me.Hide
End Sub

Private Sub Timer1_Timer()
    Static b As Boolean
    b = Not b
    If b Then
        Me.Icon = Picture2.Picture
    Else
        Me.Icon = Picture3.Picture
    End If
End Sub
```

A.16. Ficheiros Gráficos

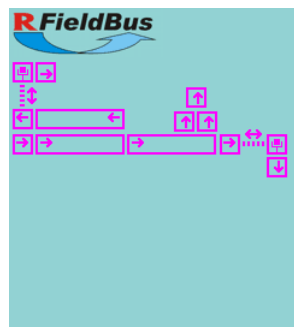
A.16.1. Utilizados na propriedade Picture das Forms

O ficheiro gráfico `schematic.bmp` (Figura A.14a) é utilizado para a propriedade `picture` das forms `Main_Form`, `Monitor_Form` e `Config_Form`.

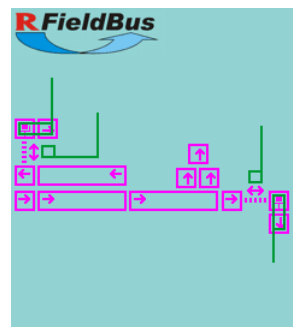
O ficheiro gráfico `pcs.bmp` (Figura A.14b) é utilizado para a propriedade `picture` da form `Monitor_BR_Form`.

O ficheiro gráfico `background.bmp` (Figura A.14c) é utilizado para a propriedade `picture` das forms `BR_Info_Form`, `Config_Devices_Form`, `Gen_Events_Form` and `Show_Events_Form`.

O ficheiro gráfico `cameras.bmp` (Figura A.14d) é utilizado para a propriedade `picture` da form `Select_CAM_Info_Form`.



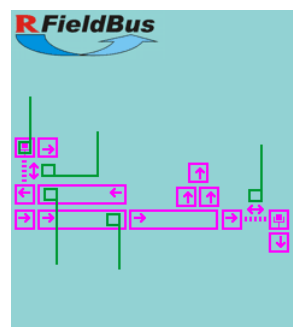
a)



b)



c)



d)

Figura A.14: Imagens utilizadas para a propriedade `picture` das forms

O ficheiro gráfico `camerasbkg.bmp` (Figura A.15a) é utilizado para a propriedade `picture` da form `CAM_View_Form`.

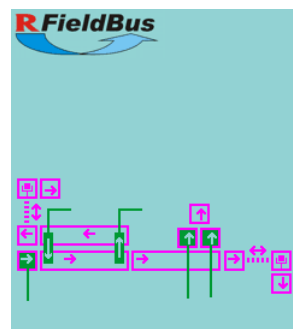
O ficheiro gráfico `control.bmp` (Figura A.15b) é utilizado para a propriedade `picture` da form `Control_Form`.

O ficheiro gráfico `buffers.bmp` (Figura A.15c) é utilizado para a propriedade `picture` da form `Config_Buffers_Form`.

Finalmente, o ficheiro gráfico `novideo.jpg` (Figura A.15d) é utilizado para a propriedade `picture` do objecto `CAM_View_Form.CamPic`, quando é feito um OFF da câmara (ou pelo cliente, ou pelo servidor).



a)



b)

c)		d)

Figura A.15: Outras imagens utilizadas para a propriedade `picture` das forms

A.16.2. Utilizados na propriedade `Icon` das *Forms*

Foram desenhados dois `icons` (utilizando o *Microsoft Development Environment*) para serem utilizados na definição da propriedade `icon` das *forms*.

O ficheiro gráfico `icon1.ico` é utilizado em todas as `forms`. É também utilizado para a propriedade `picture` do objecto `Image2` da `form Entry_Form`.

O ficheiro gráfico `icon2.ico` é utilizado para alternar com o `icon1.ico` como propriedade `icon` da `form Show_Events_Form`, como explicado na Secção A.15. Nessa `form`, são utilizados dois objectos do tipo `PictureBox` (`Picture2` e `Picture3`) não visíveis para fazer essa manipulação da propriedade da `form`.

A.16.3. Utilizados em outros objectos tipo `PictureBox` das `Forms`

Na `form Show_Events_Form` é utilizada uma `PictureBox` (`Picture1`) cuja propriedade `Picture` é actualizada (no procedimento `Show_Alarm` do módulo `Globals.bas`) com os ficheiros gráficos ilustrados na Figura A.16.

Esses ficheiros gráficos são: `error.bmp` (a); `info.bmp` (b); `ok.bmp` (c); `phone.bmp` (d); `stop.bmp` (e); `wait.bmp` (f).



Figura A.16: Imagens utilizadas para a propriedade `picture` do objecto `Picture1` da `form Show_Events_Form`

A.17. Ficheiros Áudio

Associado ao procedimento Show_Alarm do módulo Globals.bas, e por isso à form Show_Events_Form, para além dos ficheiros gráficos associados a cada tipo básico de evento, é disponibilizada informação sonora através do processamento dos seguintes ficheiros áudio: rfError.wav; rfInfo.wav; rfOK.wav; rfPhone.wav; rfStop.wav; rfWait.wav.

Anexo B

Detalhes VB6.0 da Aplicação Servidor *Intranet*

Neste anexo são descritos os detalhes essenciais da aplicação Servidor Intranet desenvolvida em *Visual Basic 6.0*. A aplicação é constituída por uma só form (*Server_Form*), um module (*windowsMessages.bas*) e um class module (*windowsMessages.cls*). Os detalhes destes últimos são descritos em (Pacheco, 2002e). Relativamente à *Server_Form*, o código relativo a esta será apresentando em secções separadas deste anexo. Primeiro os objectos relevantes, e depois os eventos/procedimentos por grupos de funcionalidade.

B.1. Layout e Objectos Relevantes da *Server_Form*

B.1.1. Layout da *Server_Form*

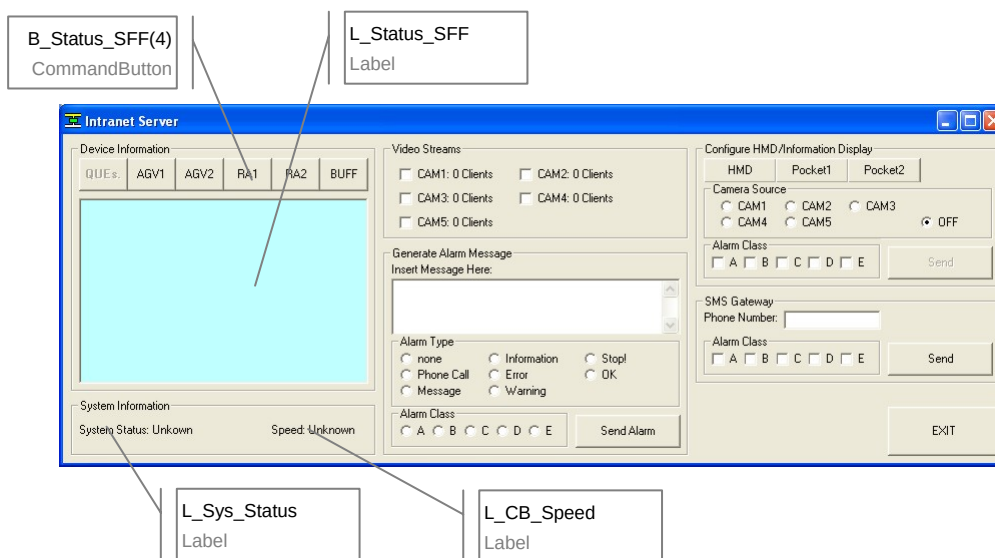


Figura B.1: Layout da *Server_Form* e objectos relacionados com a monitorização do estado do SFF

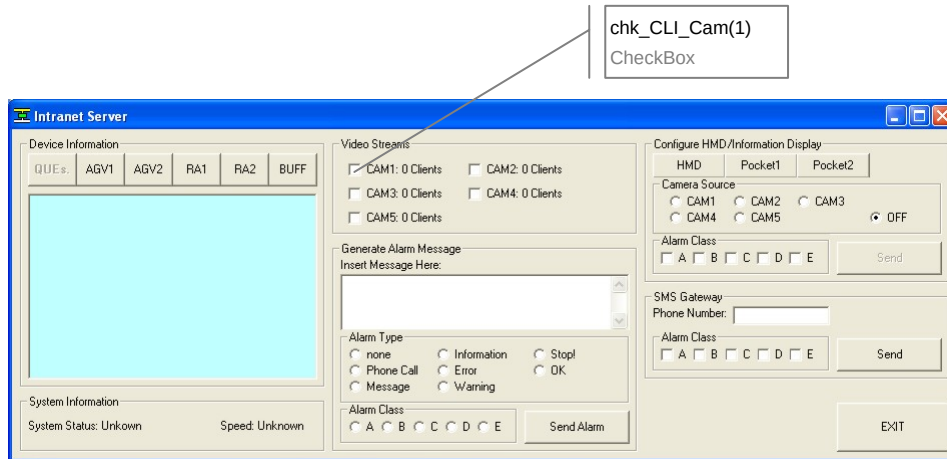


Figura B.2: Layout da Server_Form e objectos relacionados com a monitorização e gestão da subscrição (por parte de Clientes Intranet) e comandos de activação de vídeo streams UDP por parte do Image Center

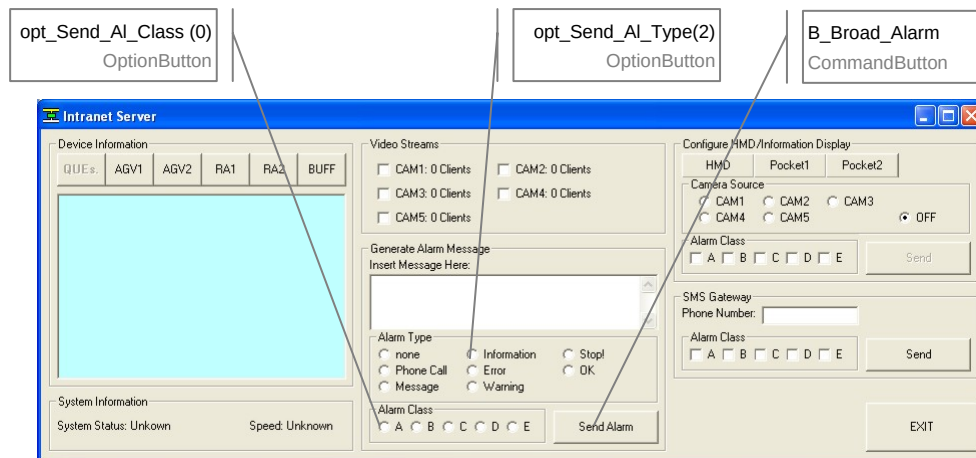


Figura B.3: Layout da Server_Form e objectos relacionados com a geração de alarmes/eventos para os Clientes Intranet

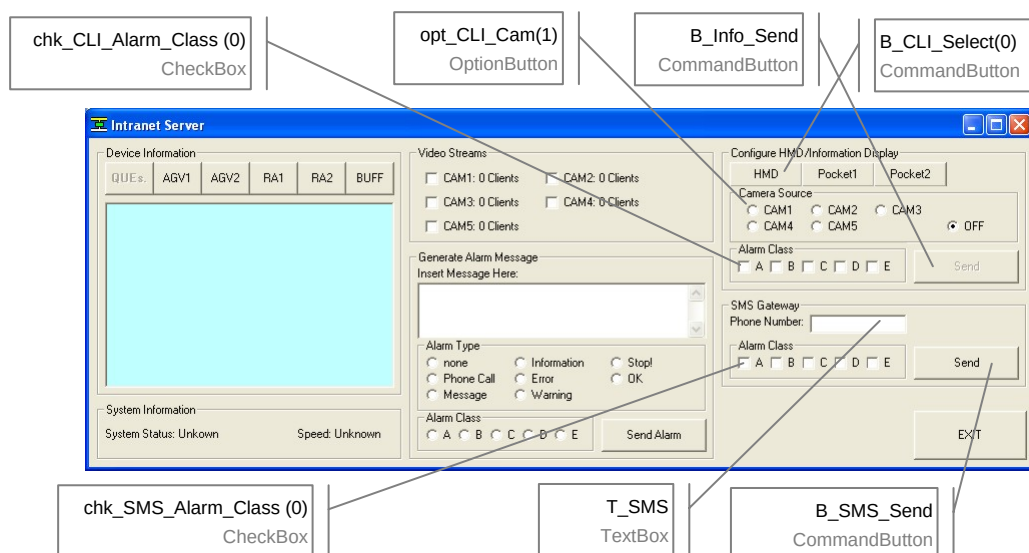


Figura B.4: Layout da Server_Form e objectos relacionados com a geração a configuração de Clientes Intranet de capacidade limitada (HMD/PK1/PK2) e com a configuração da Gateway SMS

B.1.2. Objectos relevantes da Server_Form

Tabela B.1: Relação de objectos relevantes da Server_Form

Objecto (Nome)	Tipo	Observações
B_Broad_Alarm	CommandButton	Faz broadcast de alarme/evento configurado para os Clientes <i>Intranet</i>
B_CLI_Select	CommandButton	Selecciona HMD/PK1/PK2; consulta em <code>chk_CLI_alarm(i)</code> e em <code>Opt_CLI_Cam(i)</code> em função da selecção
B_Exit	CommandButton	Para sair da aplicação
B_Info_Send	CommandButton	Faz <i>broadcast</i> para os Clientes <i>Intranet</i> da configuração do cliente HMD/PK1/PK2 (conforme <code>B_CLI_Select(i)</code>)
B_SMS_Send	CommandButton	Faz <i>broadcast</i> para os Clientes <i>Intranet</i> da configuração da Gateway SMS
B_Status_SFF(i)	CommandButton	Permite seleccionar a info a visualizar em <code>L_Status_SFF</code> dos dispositivos do SFF (RA1, RA2, AGV1, AGV2, QUEUES, BUFFERS)
chk_CLI_Alarm(i)	CheckBox	Configuração de classes se alarmes para clientes HMD/PK1/PK2, (a enviar via <code>B_Send_Info</code>)
chk_CLI_Cam(i)	CheckBox	ON/OFF de <i>Video Streams</i> ; a <i>caption</i> reflecte contador de subscrições
Chk_SMS_Alarm_Class(i)	CheckBox	Seleccção das classes se alarme a enviar em <i>broadcast</i> com <code>B_SMS_Send</code> ; para configuração da Gateway SMS
Frame1...Frame10	Frame	
L_CB_Speed	Label	Visualiza a velocidade dos CB
L_Mess_Alarm	Label	= "Insert Message Here"
L_SMS_Ph	Label	= "Phone Number":
L_Status_SFF	Label	Visualiza info dos dispositivos, em função de <code>B_Status_SFF(i)</code>
L_Sys_Status	Label	Visualiza o estado (STOP/RUN/etc. do SFF)
Opt_CLI_Cam(i)	OptionButton	Seleccção da cam para configuração dos clientes HMD/PK1/PK2 (a enviar via <code>B_Send_Info</code>)
Opt_Send_Al_Class(i)	OptionButton	Seleccção da classe de alarme a enviar em <i>broadcast</i> com <code>B_Broad_Alarm</code> ; a propriedade <i>caption</i> de cada (i) = A, B, C, D, E
Opt_Send_Al_Type(i)	OptionButton	Seleccção da classe de alarme a enviar em <i>broadcast</i> com <code>B_Broad_Alarm</code> ;
Server_Form	Form	
T_Alarm	TextBox	Texto associado ao evento/alarme a enviar em <i>broadcast</i> com <code>B_Broad_Alarm</code> ;
T_SMS_Number	TextBox	Para preencher com o número de telemóvel a enviar em <i>broadcast</i> com <code>B_SMS_Send</code> ; para configuração da Gateway SMS
Text1	TextBox	Fora da visibilidade da form, e utilizado para fazer <code>SetFocus</code> depois de <code>B_Status_SFF(i)_Click</code>
WskCLIENTS	Winsock	Para comunicação UDP/IP com Clientes
WskHMI	Winsock	Para comunicação UDP/IP com RF - HMI

B.2. Código da Server_Form

B.2.1. Declarações Globais

```
Option Explicit
Public WithEvents wm1 As WindowsMessages
Public wmRF_Video_CTRL As Long 'BROADCAST

Const CLIENT_PORT = 1319
Const SERVER_PORT = 1309
Const BROADCAST_ADDR = "255.255.255.255"

' Contador de clientes subscritores das camaras
Dim Cntr_CLI_Cam(6) As Integer

' Indice utilizado no array de botões B_Status_SFF(i)
Dim B_Status_Index As Integer

' Armazenamento da informação de status dos dispositivos do SFF
Dim Status_Val(7) As String

' Armazenamento da informação de MAX_BUFF do SFF
Dim BUF_Max_Val(5) As String

' Armazenamento da classes de alarme e cameras dos Clientes especiais
Dim CLI_Info_Alarm_Classes(3) As String
Dim CLI_Info_Cam(3) As Byte

' Indice utilizado no array de botões B_CLI_Select(i)
Dim B_CLI_Select_Index As Integer

' Variáveis relacionadas com o envio de alarmes
Dim selected_alarm_type As String 'tipo de alarme seleccionado
Dim selected_alarm_class As String 'classe de alarme seleccionada

' Indice utilizado no array de opções opt_CLI_Cam_Click(i)
Dim opt_CLI_Cam_Index As Integer

' Armazenamento da informação de estado do sistema (i=1) e velocidade (i=2)
Dim SFF_State_Val(3) As String
```

B.2.2. Evento Form_Load

```
Private Sub Form_Load()  
    Dim i As Integer  
  
    Set wm1 = New WindowsMessages  
    wmRF_Video_CTRL = wm1.GetMessageID("RF_VIDEO_CTRL")  
  
    Me.Show  
  
    WskCLIENTS.RemotePort = CLIENT_PORT  
    WskCLIENTS.Bind SERVER_PORT  
  
    ' Porta e Endereço IP do HMI  
    WskHMI.RemotePort = 1409  
    WskHMI.RemoteHost = "127.0.0.1"  
    WskHMI.Bind 1429  
  
    'por default o sistema está "STOP" e a velocidade é 0  
    SFF_State_Val(1) = "STOP"  
    SFF_State_Val(2) = "0"  
    'por default o tamanho máximo dos buffers é 0  
    For i = 1 To 5  
        BUF_Max_Val(i) = 0  
    Next i  
    B_Status_Index = 1: B_Status_SFF_Click 1 'vizualização das QUEUES  
End Sub
```

B.2.3 Evento B_Exit_Click

```
Private Sub B_Exit_Click()  
    End  
End Sub
```

B.2.4. Evento WskHMI_DataArrival (Pacotes UDP Vindos do RF-HMI)

```

Private Sub WskHMI_DataArrival(ByVal bytesTotal As Long)
' *** Mensagens UDP Recebidas do HMI
  Dim msg_rec As String

  On Error Resume Next
  WskHMI.GetData msg_rec, vbString
  If msg_rec = "" Then Exit Sub

  Select Case Left(msg_rec, 1)
    Case "!", "&"
      '!" Alarme/evento (do HMI é sobre os Buffers MAX)
      '&" Alteração dos Tamanhos Máximos dos Buffers
      WskCLIENTS.RemoteHost = BROADCAST_ADDR
      WskCLIENTS.SendData msg_rec ' faz broadcast para os clientes
    Case "?"
      'Informação sobre os dispositivos do SFF
      WskCLIENTS.RemoteHost = BROADCAST_ADDR
      WskCLIENTS.SendData msg_rec ' faz broadcast para os clientes

      'Atualiza informação no servidor
      WskHMI_Update_Server_dev_status (msg_rec)
    Case "#"
      'Informação de estado do sistema ou velocidades dos CB
      'OU accionamentos de SAX ou CYLx
      '(SAX e CYLx é informação não processada no clientes)
      WskCLIENTS.RemoteHost = BROADCAST_ADDR
      WskCLIENTS.SendData msg_rec ' faz broadcast para os clientes

      'Atualiza informação no servidor
      WskHMI_Update_Server_sys_status (msg_rec)
  End Select
End Sub

Private Sub WskHMI_Update_Server_sys_status(msg_r As String)
'*** Chamada de WskHMI quando mensagem "#" (Info de controlo dos sistema)
'Informação de estado do sistema ou velocidades dos CB OU accionamentos de SAX/CYLx
'(SAX e CYLx é informação não processados AQUI)

  Dim p As Long
  Dim parm_id, info As String

  p = InStr(msg_r, ":")
  If p > 0 Then
    parm_id = UCase(Mid(msg_r, 2, p - 2))
    info = UCase(Mid(msg_r, p + 1))

    Select Case parm_id
      Case "OP":
        L_Sys_Status.Caption = "System Status = " & info
        SFF_State_Val(1) = info
      Case "CB_SPEED":
        L_CB_Speed.Caption = "Speed = " & info
        SFF_State_Val(2) = info
    End Select
  End If
End Sub

```

```
Private Sub WskHMI_Update_Server_dev_status(msg_r As String)
  *** Chamada de WskHMI quando mensagem "?" (Info dos Dispositivos SFF)
  Dim p As Long
  Dim dev_txt, info As String
  Dim id_dev As Integer

  p = InStr(msg_r, ".")
  If p > 0 Then
    dev_txt = Mid(msg_r, 2, p - 2)
    info = Mid(msg_r, p + 1)
  End If

  Select Case UCase(dev_txt)
    Case "QUEUE": id_dev = 1
    Case "AGV1": id_dev = 2
    Case "AGV2": id_dev = 3
    Case "RA1": id_dev = 4
    Case "RA2": id_dev = 5
    Case "BUFF": id_dev = 6
    Case Else: id_dev = 0
  End Select

  If id_dev > 0 Then
    Status_Val(id_dev) = info
    If B_Status_Index = id_dev Then ' se dispositivo selecionado
      L_Status_SFF.Caption = info
    End If
  End If
End Sub
```

B.2.5. Evento WskCLIENTS_DataArrival (Pacotes UDP Vindos dos Clientes)

```

Private Sub WskCLIENTS_DataArrival(ByVal bytesTotal As Long)
  *** Pacotes UDP Recebidos dos Clientes

  Dim msg_rec As String

  On Error Resume Next
  WskCLIENTS.GetData msg_rec, vbString

  Select Case Left(msg_rec, 1)
    Case "?"
      'Pedido de informação sobre os dispositivos
      WskCLIENTE_Arrived_DEV_INFO_Request (msg_rec)
    Case "&"
      'Pedido de alteração do valor máximo de um BUFFER
      ' OU pedido de consulta do valor
      WskCLIENTE_Arrived_BUFF_Request (msg_rec)
    Case "#"
      'Pedido de consulta de estado do systema ("#OP"; "#CB_SPEED")
      'Ou Pedido de accção de controlo sobre o SFF:
      ' ("OP:STOP"; "OP:CB_SPEED:FASTER", "OP:SAX:ACTIVATE", ETC..)
      WskCLIENTE_Arrived_Control_Request (msg_rec)
    Case ">"
      'Pedido relacionado com as câmaras
      WskCLIENTE_Arrived_CAM_Request (msg_rec)
    Case "%"
      'Alteração de configuração de clientes de ou gateway SMS
      WskCLIENTE_Arrived_HMD_PK_SMS_Config (msg_rec)
  End Select
End Sub

Private Sub WskCLIENTE_Arrived_DEV_INFO_Request(msg_r As String)
  *** Pedido de informação sobre o estado dos dispositivos
  Dim id_dev As String
  Dim server_var_ind As Integer

  id_dev = Mid(msg_r, 2)
  Select Case UCase(id_dev)
    Case "QUEUE": server_var_ind = 1
    Case "AGV1": server_var_ind = 2
    Case "AGV2": server_var_ind = 3
    Case "RA1": server_var_ind = 4
    Case "RA2": server_var_ind = 5
    Case "BUFF": server_var_ind = 6
    Case Else: server_var_ind = 0
  End Select

  WskCLIENTS.RemoteHost = BROADCAST_ADDR
  'Envia a informação actualmente disponível para o dispositivo
  If server_var_ind > 0 Then
    WskCLIENTS.SendData "?" & id_dev & ":" & Status_Val(server_var_ind)
  Else 'Pedido desconhecido
    WskCLIENTS.SendData "?" & id_dev & ":"
  End If
End Sub

```

```

Private Sub WskCLIENTE_Arrived_BUFF_Request(msg_r As String)
'Pedido de alteração do valor máximo de um BUFFER
'OU pedido de consulta do seu valor

    Dim p As Long
    Dim parm_id, info, snd_msg As String
    Dim i As Integer

    p = InStr(msg_r, ":")

    ' alteração do valor
    If p > 0 Then
        parm_id = UCase(Mid(msg_r, 2, p - 2))
        info = Mid(msg_r, p + 1)
        Select Case parm_id
            Case "B1MAX": i = 1
            Case "B2MAX": i = 2
            Case "B3MAX": i = 3
            Case "B4MAX": i = 4
            Case "B5MAX": i = 5
            Case Else:    i = 0
        End Select

        If i > 0 Then
            'Envia o pedido para o HMI
            snd_msg = "&" & parm_id & ":" & info
            WskHMI.SendData snd_msg
        End If
    'Consulta de valor
    Else
        parm_id = UCase(Mid(msg_r, 2))
        Select Case parm_id
            Case "B1MAX": i = 1
            Case "B2MAX": i = 2
            Case "B3MAX": i = 3
            Case "B4MAX": i = 4
            Case "B5MAX": i = 5
            Case Else:    i = 0
        End Select

        If i > 0 Then
            'Envia a resposta para os clientes
            WskCLIENTS.RemoteHost = BROADCAST_ADDR
            snd_msg = "&" & parm_id & ":" & BUF_Max_Val(i)
            WskCLIENTS.SendData snd_msg
        End If
    End If
End Sub

```

```

Private Sub WskCLIENTE_Arrived_Control_Request(msg_r As String)
'Pedido de informação de estado do sistema ("OP"; "#CB_SPEED")
'Ou Pedido de accção de controlo sobre o SFF:
'("OP:STOP"; "OP:CB_SPEED:FASTER", "OP:SAX:ACTIVATE", ETC.)

    Dim p As Long
    Dim i As Integer
    Dim parm_id, info, snd_msg As String

    p = InStr(msg_r, ".")

    'Acção de Controlo
    If p > 0 Then
        parm_id = UCase(Mid(msg_r, 2, p - 2))
        info = UCase(Mid(msg_r, p + 1))
        i = 0
        Select Case parm_id
            Case "OP":
                If info = "STOP" Then i = 1
            Case "CB_SPEED":
                If info = "SLOWER" Or info = "FASTER" Then i = 1
            Case "SA1", "SA2", "CYL1", "CYL2", "CYL3"
                If info = "ACTIVATE" Then i = 1
            Case Else:
            End Select

        'Caso mensagem válida envia o comando para o HMI
        If i > 0 Then
            snd_msg = "#" & parm_id & "." & info
            WskHMI.SendData snd_msg
        End If

        'Pedido de informação de estado do sistema ou velocidade
    Else
        parm_id = UCase(Mid(msg_r, 2))
        Select Case parm_id
            Case "OP": i = 1
            Case "CB_SPEED": i = 2
            Case Else: i = 0
        End Select

        If i > 0 Then
            'se pedido válido, envia o estado do sistema
            WskCLIENTS.RemoteHost = BROADCAST_ADDR
            snd_msg = "#" & parm_id & "." & SFF_State_Val(i)
            WskCLIENTS.SendData snd_msg
        End If
    End If
End Sub

```

```

Private Sub WskCLIENTE_Arrived_CAM_Request(msg_r As String)
'Pedido relacionado com as câmaras
'request de ON/OFF

    Dim p As Long
    Dim CamID, info As String

    p = InStr(msg_r, ":")
    If p > 0 Then
        CamID = Mid(msg_r, 5, 1)
        info = Mid(msg_r, p + 1)
        If CamID <> 0 Then
            Select Case LCase(info)
                Case "on"
                    'recebe um pedido de on
                    Cntr_CLI_Cam(CamID) = Cntr_CLI_Cam(CamID) + 1
                    'se estiver desactivado no image center
                    If chk_CLI_Cam(CamID).Value = 0 Then
                        'pede ao image center para activar o envio
                        wm1.PostMessageLong HWND_BROADCAST, wmRF_Video_CTRL, CamID, 1
                    End If
                    chk_CLI_Cam(CamID).Value = 1

                    'confirma ao cliente específico (não Broadcast)
                    WskCLIENTS.SendData ">CAM" & CamID & ":ON"

                Case "off"
                    'recebe um pedido de off
                    Cntr_CLI_Cam(CamID) = Cntr_CLI_Cam(CamID) - 1
                    If Cntr_CLI_Cam(CamID) < 0 Then Cntr_CLI_Cam(CamID) = 0

                    If Cntr_CLI_Cam(CamID) = 0 Then
                        If chk_CLI_Cam(CamID).Value = 1 Then
                            wm1.PostMessageLong HWND_BROADCAST, wmRF_Video_CTRL, CamID, 0
                        End If
                        chk_CLI_Cam(CamID).Value = 0
                    End If
                End Select

                'actualiza as labels que indicam número de clientes
                If Cntr_CLI_Cam(CamID) = 1 Then
                    chk_CLI_Cam(CamID).Caption = "CAM" & CamID & ": 1 Client"
                Else
                    chk_CLI_Cam(CamID).Caption = "CAM" & CamID & ": " &
                        Cntr_CLI_Cam(CamID) & " Clients"
                End If
            End If
        End If
    End Sub

```

```

Private Sub WskCLIENTE_Arrived_HMD_PK_SMS_Config(msg_r As String)
'Alteração de configuração de clientes de ou gateway SMS

    Dim p As Long
    Dim id_device, info1, txt_info, al_cl_lst As String
    Dim cli_ind, i As Integer

    p = InStr(msg_r, ":")
    If p > 0 Then
        id_device = Mid(msg_r, 2, p - 2)
        info1 = Mid(msg_r, p + 1)
        p = InStr(info1, ":")
        If p > 0 Then
            txt_info = Mid(info1, p + 1)
            al_cl_lst = Left(info1, p - 1)
            cli_ind = -1
            Select Case id_device
                Case "HMD"
                    cli_ind = 0
                Case "PK1"
                    cli_ind = 1
                Case "PK2"
                    cli_ind = 2
                Case "SMS"
                    For i = 0 To 4
                        If InStr(al_cl_lst, chk_SMS_Alarm_Class(i).Caption) > 0 Then
                            chk_SMS_Alarm_Class(i).Value = 1
                        Else
                            chk_SMS_Alarm_Class(i).Value = 0
                        End If
                    Next i
                    T_SMS.Text = txt_info
            End Select

            'se "PK1", "PK2" ou "HMD"
            If cli_ind >= 0 Then
                'actualiza as variáveis so servidor
                CLI_Info_Cam(cli_ind) = txt_info
                CLI_Info_Alarm_Classes(cli_ind) = al_cl_lst
                'se o dispositivo estiver seleccionado
                If B_CLI_Select(cli_ind).FontBold = True Then
                    For i = 0 To 4
                        If InStr(CLI_Info_Alarm_Classes(cli_ind),
                            chk_CLI_Alarm_Class(i).Caption) > 0 Then
                            chk_CLI_Alarm_Class(i).Value = 1
                        Else
                            chk_CLI_Alarm_Class(i).Value = 0
                        End If
                    Next i
                    opt_CLI_Cam(txt_info).Value = True
                End If
                B_Info_Send.FontBold = False
            End If
        End If
    End If
End Sub

```

B.2.6. Outros Eventos Relacionados com a Visualização do Status do SFF

```

Private Sub B_Status_SFF_Click(Index As Integer)
    *** VISUALIZAÇÃO DO STATUS DO SFF ***
    'Selecciona o dispositivo

    Dim i As Integer
    B_Status_Index = Index
    L_Status_SFF.Caption = Status_Val(B_Status_Index)
    For i = 1 To 6
        B_Status_SFF(i).Enabled = B_Status_Index <> i
        B_Status_SFF(i).FontBold = B_Status_Index = i
    Next i
    Text1.SetFocus
End Sub

```

B.2.7. Outros Eventos Relacionados com a Gestão dos Fluxos de Vídeo

```

Private Sub chk_CLI_Cam_Click(Index As Integer)
    *** DESLIGAR / LIGAR EMISSÃO DE IMAGENS NO IMAGE CENTRE
    *** Enviado para o Image Center
    *** Enviado o ">CAMx:OFF para os clientes

    If chk_CLI_Cam(Index).Value = 1 Then
        wm1.PostMessageLong HWND_BROADCAST, wmRF_Video_CTRL, Index, 1
    Else
        Cntr_CLI_Cam(Index) = 0
        chk_CLI_Cam(Index).Caption = "CAM" & Index & ": 0 Clients"
        WskCLIENTS.RemoteHost = BROADCAST_ADDR
        WskCLIENTS.SendData ">CAM" & Index & ":OFF"
        wm1.PostMessageLong HWND_BROADCAST, wmRF_Video_CTRL, Index, 0
    End If
End Sub

```

B.2.8. Eventos Relacionados com o Broadcast de Alarmes/Eventos para os Clientes

```

Private Sub opt_Send_Al_Type_Click(Index As Integer)
    *** GERAÇÃO DE ALARME ***
    'Selecciona tipo de alarme a enviar em broadcast

    selected_alarm_type = opt_Send_Al_Type(Index).Tag
End Sub

Private Sub opt_Send_Al_Class_Click(Index As Integer)
    *** GERAÇÃO DE ALARME ***
    'Selecciona classe de alarme a enviar em broadcast

    selected_alarm_class = opt_Send_Al_Class(Index).Caption
End Sub

```

```

Private Sub B_Broad_Alarm_Click()
    '*** GERAÇÃO DE ALARME ***
    'Envia alarm

    WskCLIENTS.RemoteHost = BROADCAST_ADDR
    WskCLIENTS.SendData "!EVT" & ":" & selected_alarm_class & ":" &
        selected_alarm_type & ":" & T_Alarm
End Sub

```

B.2.9. Outros Eventos Relacionados com a Configuração da Gateway SMS

```

Private Sub B_SMS_Send_Click()
    '*** GATEWAY SMS ***
    'Envio de configuração
    Dim g_SMS_Al_Class As String
    Dim i As Integer

    g_SMS_Al_Class = ""
    For i = 0 To 4
        If chk_SMS_Alarm_Class(i).Value = 1 Then
            g_SMS_Al_Class = g_SMS_Al_Class & chk_SMS_Alarm_Class(i).Caption
        End If
    Next i
    WskCLIENTS.SendData "%SMS" & ":" & g_SMS_Al_Class & ":" & T_SMS
End Sub

```

B.2.10. Outros Eventos Relacionados com a Configuração dos HMD/PK1/PK2

```

Private Sub B_CLI_Select_Click(Index As Integer)
    '*** CLIENTES ESPECIAIS ***
    'Seleção do cliente para visualizar configuração

    Dim i As Integer
    Dim j As Integer
    For i = 0 To 2
        B_CLI_Select(i).FontBold = (Index = i)
    Next i
    B_CLI_Select_Index = Index
    For j = 0 To 4
        If InStr(CLI_Info_Alarm_Classes(B_CLI_Select_Index),
            chk_CLI_Alarm_Class(j).Caption) > 0 Then
            chk_CLI_Alarm_Class(j).Value = 1
        Else
            chk_CLI_Alarm_Class(j).Value = 0
        End If
    Next j
    opt_CLI_Cam(CLI_Info_Cam(B_CLI_Select_Index)).Value = True
    B_Info_Send.FontBold = False
    B_Info_Send.Enabled = True
End Sub

```

```

Private Sub chk_CLI_Alarm_Class_Click(Index As Integer)
    '*** CLIENTES ESPECIAIS ***
    'Seleção de classes de alarme

    Dim fChanged As Boolean
    Dim i As Integer
    For i = 0 To 4
        If InStr(CLI_Info_Alarm_Classes(B_CLI_Select_Index),
            chk_SMS_Alarm_Class(i).Caption) > 0 Then
            If chk_CLI_Alarm_Class(i).Value <> 1 Then fChanged = True
        Else
            If chk_CLI_Alarm_Class(i).Value <> 0 Then fChanged = True
        End If
    Next i
    If Index <> CLI_Info_Cam(B_CLI_Select_Index) Then
        fChanged = True
    End If
    B_Info_Send.FontBold = fChanged
End Sub

Private Sub opt_CLI_Cam_Click(Index As Integer)
    '*** CLIENTES ESPECIAIS ***
    'Seleção de câmara

    Dim fChanged As Boolean
    Dim i As Integer
    opt_CLI_Cam_Index = Index
    For i = 0 To 4
        If InStr(CLI_Info_Alarm_Classes(B_CLI_Select_Index),
            chk_SMS_Alarm_Class(i).Caption) > 0 Then
            If chk_CLI_Alarm_Class(i).Value <> 1 Then fChanged = True
        Else
            If chk_CLI_Alarm_Class(i).Value <> 0 Then fChanged = True
        End If
    Next i
    If Index <> CLI_Info_Cam(B_CLI_Select_Index) Then
        fChanged = True
    End If
    B_Info_Send.FontBold = fChanged
End Sub

Private Sub B_Info_Send_Click()
    '*** CLIENTES ESPECIAIS ***
    'Envio de configuração

    Dim g_CLI_Al_Class As String
    Dim i As Integer
    g_CLI_Al_Class = ""
    For i = 0 To 4
        If chk_CLI_Alarm_Class(i).Value = 1 Then
            g_CLI_Al_Class = g_CLI_Al_Class & chk_CLI_Alarm_Class(i).Caption
        End If
    Next i
    WskCLIENTS.RemoteHost = BROADCAST_ADDR
    WskCLIENTS.RemotePort = CLIENT_PORT
    Select Case B_CLI_Select_Index
        Case 0 'HMD
            WskCLIENTS.SendData "%HMD:" & g_CLI_Al_Class & ":" & opt_CLI_Cam_Index
        Case 1 'Pocket1
            WskCLIENTS.SendData "%PK1:" & g_CLI_Al_Class & ":" & opt_CLI_Cam_Index
        Case 2 'Pocket2
            WskCLIENTS.SendData "%PK2:" & g_CLI_Al_Class & ":" & opt_CLI_Cam_Index
    End Select
    CLI_Info_Alarm_Classes(B_CLI_Select_Index) = g_CLI_Al_Class
    CLI_Info_Cam(B_CLI_Select_Index) = opt_CLI_Cam_Index
    B_Info_Send.FontBold = False
End Sub

```

Anexo C

Detalhes VB6.0 da Aplicação RF - HMI

Neste anexo são descritos os detalhes essenciais da aplicação RF-HMI desenvolvida em *Visual Basic 6.0*. A aplicação é constituída por uma só form (Server_Form), um module (WindowsMessages.bas) e um class module (WindowsMessages.cls). Os detalhes destes últimos são descritos em (Pacheco 2002e). Relativamente ao código da HMI_Form, este será apresentando em secções separadas deste anexo. O código de acrescentado (*stub*) para fazer a interface entre o SFF e os Clientes *Intranet* é indicado a sombreado.

C.1. Layout da HMI_Form

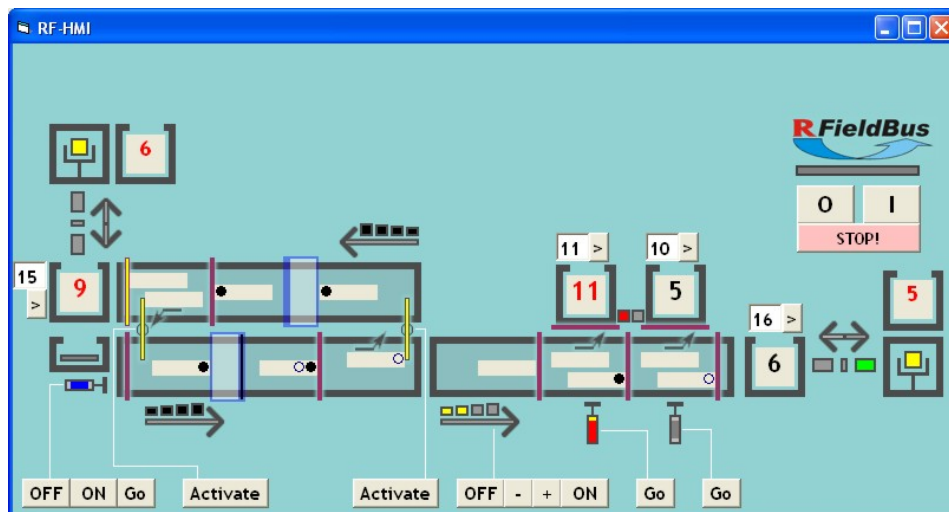


Figura C.1: Layout da HMI_Form

C.2. Código da HMI_Form

C.2.1. Declarações Globais

```
Option Explicit
Public WithEvents wm1 As WindowsMessages
Public WithEvents wm2 As WindowsMessages

Private Declare Function GetWindow Lib "user32" (ByVal hwnd As Long, ByVal wCmd As Long) As Long
Const GW_OWNER = 4

Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
Const HWND_TOP = 0
Const HWND_TOPMOST = -1
Const SWP_NOACTIVATE = &H10
Const SWP_NOMOVE = &H2
Const SWP_NOSIZE = &H1

Const INTRANET_SERVER_ADDR = "127.0.0.1"

'Constantes utilizadas na chegada de informação de classificação das peças
Const PART_UNKNOWN = &H1
Const PART_BLACK = &H2
Const PART_GRAY = &H4
Const PART_WHITE = &H8
Const PART_PICTURE = &H10
Const PART_ERROR = &H20

Const G_USE_WINDOWS_MSG = True

'Mensagens para comunicação com o RF-CON
Public wmRF_Msg_REG As Long
Public wmRF_Msg_IR As Long
Public wmRF_Msg_CAM As Long
Public wmRF_Msg_MM As Long
Public wmRF_Msg_BUFF As Long
Public wmRF_Msg_BUFF_SET As Long
Public wmRF_Msg_BUFF_CNF As Long
Public wmRF_Msg_AGV As Long
Public wmRF_Msg_SA As Long
Public wmRF_Msg_CYL As Long
Public wmRF_Msg_OUTP As Long
Public wmRF_Msg_QUEUE As Long
Public wmRF_Msg_ROBOT As Long
Public wmRF_Msg_CMD As Long
Public hWnd_CMD As Long
Public wmRF_Msg2_REG As Long
Public wmRF_Msg2_REG2 As Long
Public hWnd_SERVER As Long

'Variáveis utilizadas para guardar informação que vai chegando sobre o SFF
Private CamTxt(2) As String
Private CamCount(2) As Integer
Private AGVtxt(2) As String
Private AGVCount(2) As Integer
Private RATxt(2) As String
Private RACount(2) As Integer
```

C.2.2. Eventos Form_Load e Form_Unload

```

Private Sub Form_Load()
    Dim hwnd2 As Long

    hwnd_CMD = 0
    Set wm1 = New WindowsMessages
    wm1.ClearRegisteredMessages

    If G_USE_WINDOWS_MSG Then
        wmRF_Msg_REG = wm1.RegisterMessage("RF_MSG_REG", wmtLong)
        wmRF_Msg_IR = wm1.RegisterMessage("RF_MSG_IR", wmtLong)
        wmRF_Msg_CAM = wm1.RegisterMessage("RF_MSG_CAM", wmtLong)
        wmRF_Msg_MM = wm1.RegisterMessage("RF_MSG_MM", wmtLong)
        wmRF_Msg_BUFF = wm1.RegisterMessage("RF_MSG_BUFF", wmtLong)
        wmRF_Msg_BUFF_CNF = wm1.RegisterMessage("RF_MSG_BUFF_CNF", wmtLong)
        wmRF_Msg_AGV = wm1.RegisterMessage("RF_MSG_AGV", wmtLong)
        wmRF_Msg_SA = wm1.RegisterMessage("RF_MSG_SA", wmtLong)
        wmRF_Msg_CYL = wm1.RegisterMessage("RF_MSG_CYL", wmtLong)
        wmRF_Msg_OUTP = wm1.RegisterMessage("RF_MSG_OUTP", wmtLong)
        wmRF_Msg_QUEUE = wm1.RegisterMessage("RF_MSG_QUEUE", wmtLong)
        wmRF_Msg_ROBOT = wm1.RegisterMessage("RF_MSG_ROBOT", wmtLong)
    End If

    wmRF_Msg_CMD = wm1.GetMessageID("RF_MSG_CMD")
    wmRF_Msg_BUFF_SET = wm1.GetMessageID("RF_MSG_BUFF_SET")

    ' Esta mensagem é utilizada para indicar ao Servidor Internet
    ' qual a nossa janela (para evitar broadcasts)
    wmRF_Msg2_REG2 = wm1.GetMessageID("RF_MSG2_REG2")

    Me.Show
    If G_USE_WINDOWS_MSG Then
        wm1.Hook Me.hwnd
    End If

    ' Para receber broadcasts
    Set wm2 = New WindowsMessages
    wm2.ClearRegisteredMessages

    ' Mensagem do Servidor Internet a indicar qual a janela que processa as mensagens
    If G_USE_WINDOWS_MSG Then
        wmRF_Msg2_REG = wm2.RegisterMessage("RF_MSG2_REG", wmtLong)
    End If

    hwnd2 = GetWindow(Me.hwnd, GW_OWNER)
    If G_USE_WINDOWS_MSG Then
        wm2.Hook hwnd2
    End If

    ' Configuração do objecto Winsock
    wsk1.LocalPort = 1409
    wsk1.RemoteHost = "127.0.0.1"
    wsk1.RemotePort = 1429
    wsk1.Bind 1409

End Sub

Private Sub Form_Unload(Cancel As Integer)
    wm1.UnHook
End Sub

```

C.2.3 Mensagens Windows Recebidas de RF-CON

```

Private Sub wm1_ReceivedLong(ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long)
'*** MENSAGENS WINDOWS RECEBIDAS DO RF-CON
On Error Resume Next
Select Case Msg
Case wmRF_Msg_QUEUE
'Mensagem relativa a uma queue
'ENVIADA PARA O SERVIDOR INTRANET
Msg_QUEUE wParam, lParam
Case wmRF_Msg_IR
'Mensagem relativa aos detectores de peças (Infra-Red)
Msg_IR wParam, lParam
Case wmRF_Msg_CAM
'Mensagem relativa à cor da peça detectada
Msg_CAM wParam, lParam
Case wmRF_Msg_BUFF
'Mensagem relativa ao número de peças num determinado buffer
'ENVIADA PARA O SERVIDOR INTRANET
Msg_BUFF wParam, lParam
Case wmRF_Msg_BUFF_CNF
'Mensagem relativa à configuração do BxMAX
'ENVIADA PARA O SERVIDOR INTRANET
Msg_BUFF_CNF wParam, lParam
Case wmRF_Msg_MM
'Mensagem relativa à velocidade dos CBs (MM = MicroMaster)
'ENVIADA PARA O SERVIDOR INTRANET
Msg_MM wParam, lParam
Case wmRF_Msg_AGV
'Mensagem relativa a uma acção do AGVx
'ENVIADA PARA O SERVIDOR INTRANET
Msg_AGV wParam, lParam
Case wmRF_Msg_ROBOT
'Mensagem relativa a uma acção do RAX
'ENVIADA PARA O SERVIDOR INTRANET
Msg_ROBOT wParam, lParam
Case wmRF_Msg_OUTP
'Mensagem relativa a uma modificação do estado do sistema
Msg_OUTP wParam, lParam
Case wmRF_Msg_SA
'Mensagem relativa a uma acção do Sx
Msg_SA wParam, lParam
Case wmRF_Msg_CYL
'Mensagem relativa a uma acção do CYLx
Msg_CYL wParam, lParam
Case wmRF_Msg_REG
Msg_REG wParam, lParam
End Select
End Sub

```

```

Public Sub Msg_QUEUE(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
'*** Informação enviada posteriormente em Timer_Send_Serv para o servidor
    On Error Resume Next
    Select Case parm
    Case 1, 2, 4, 8, 16
        QueueViewCtrl1(id).AddItem parm
    Case 100
        QueueViewCtrl1(id).RemoveItem
    Case 101
        QueueViewCtrl1(id).Clear
    End Select
End Sub

Public Sub Msg_IR(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
    On Error Resume Next
    Select Case parm
    Case 0
        IR(id).Visible = False
    Case 1
        IR(id).BorderColor = &HFFFF
        IR(id).Visible = True
    Case 2
        IR(id).BorderColor = &HFF&
        IR(id).Visible = True
    End Select
End Sub

Public Sub Msg_CAM(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
    On Error Resume Next
    SaveCam id, parm
    Select Case parm
    Case PART_UNKNOWN
        CAM(id).Visible = False
        CAM_Pic(id).Visible = False
    Case PART_BLACK
        CAM(id).BorderColor = RGB(0, 0, 0)
        CAM(id).Visible = True
        CAM_Pic(id).Visible = False
    Case PART_GRAY
        CAM(id).BorderColor = RGB(128, 128, 128)
        CAM(id).Visible = True
        CAM_Pic(id).Visible = False
    Case PART_WHITE
        CAM(id).BorderColor = RGB(255, 255, 255)
        CAM(id).Visible = True
        CAM_Pic(id).Visible = False
    Case PART_ERROR
        CAM(id).BorderColor = RGB(255, 0, 0)
        CAM(id).Visible = True
        CAM_Pic(id).Visible = False
    Case PART_PICTURE
        CAM(id).BorderColor = RGB(0, 0, 255)
        CAM(id).Visible = False
        CAM_Pic(id).Visible = True
    End Select
End Sub

```

```
Public Sub Msg_BUFF(ByVal id As Integer, ByVal parm As Integer)
```

```
'*** Wm recebida do RF-CON
```

```
' Guardada em Buff(id).Caption para ser enviada posteriormente
```

```
' (via Timer_Send_Serv)
```

```
' Se full, envia para servidor EVENTO
```

```
Dim txt As String
```

```
Dim txt2 As String
```

```
If parm < 100 Then
```

```
txt = parm
```

```
Buff(id).ForeColor = 0
```

```
ElseIf parm < 200 Then
```

```
txt = parm - 100
```

```
Buff(id).ForeColor = &HFF
```

```
wsk1.RemoteHost = INTRANET_SERVER_ADDR
```

```
txt2 = "!EVT:A:WARNING:Buffer " & id & " full: " & txt & " parts."
```

```
wsk1.SendData txt2
```

```
ElseIf parm = 200 Then
```

```
txt = "x"
```

```
Buff(id).ForeColor = 0
```

```
ElseIf parm = 201 Then
```

```
txt = "OK"
```

```
Buff(id).ForeColor = 0
```

```
ElseIf parm = 202 Then
```

```
txt = "F"
```

```
Buff(id).ForeColor = 0
```

```
End If
```

```
Buff(id).Caption = txt
```

```
End Sub
```

```
Public Sub Msg_BUFF_CNF(ByVal id As Integer, ByVal parm As Integer)
```

```
'*** Wm recebida do RF-CON
```

```
' Informação sobre alteração de BxMAX
```

```
' Guardada em txtBuffSet(id).Text
```

```
' Enviada para servidor
```

```
Dim txt As String
```

```
txtBuffSet(id).Text = parm
```

```
txt = "&B" & id & "MAX:" & parm
```

```
wsk1.RemoteHost = INTRANET_SERVER_ADDR
```

```
wsk1.SendData txt
```

```
End Sub
```

```

Public Sub Msg_MM(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
' Informação sobre alteração de velocidade dos tapetes
' Enviada para servidor

    Dim C As Long
    Dim i, parm2 As Integer
    On Error Resume Next
    Select Case parm
        Case 1, 2, 3, 4: C = RGB(0, 255, 0)
        Case 5: C = RGB(255, 0, 0)
        Case 11, 12, 13, 14: C = RGB(255, 255, 0)
    End Select
    If parm > 10 Then
        parm2 = parm - 10
    Else
        parm2 = parm
    End If
    Select Case id
        Case 1
            For i = 0 To 5
                MM1A(i).BackColor = C
                MM1B(i).BackColor = C
            Next
            MM1A(0).Visible = parm <> 0 And parm < 10
            MM1B(0).Visible = parm <> 0 And parm < 10
            For i = 1 To 4
                MM1A(i).Visible = parm2 >= i
                MM1B(i).Visible = parm2 >= i
            Next i
        Case 2
            For i = 0 To 5
                MM2(i).BackColor = C
            Next
            MM2(0).Visible = parm <> 0 And parm < 10
            For i = 1 To 4
                MM2(i).Visible = parm2 >= i
            Next i
    End Select
End Sub

```

```

Public Sub Msg_AGV(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
' Informação relativa a uma acção do AGVx
' Guarda (procedimento SaveAGV) para enviar posteriormente (via Timer_Send_Serv)
' para o servidor
    Dim C As Long
    Dim i As Integer
    On Error Resume Next

'guarda para enviar posteriormente (via Timer_Send_Serv) para o servidor
SaveAGV id, parm

    If parm < 10 Then
        C = RGB(0, 255, 0)
    Else
        C = RGB(255, 0, 0)
        parm = parm - 10
    End If
    Select Case id
        Case 1
            For i = 1 To 5
                agv1(i).FillColor = C
            Next
            agv1(1).Visible = parm = 1
            agv1(3).Visible = parm = 4
            agv1(2).Visible = parm = 2 Or parm = 3
            agv1(4).Visible = parm = 2
            agv1(5).Visible = parm = 3
        Case 2
            For i = 1 To 5
                agv2(i).FillColor = C
            Next
            agv2(1).Visible = parm = 1
            agv2(3).Visible = parm = 4
            agv2(2).Visible = parm = 2 Or parm = 3
            agv2(4).Visible = parm = 2
            agv2(5).Visible = parm = 3
    End Select
End Sub

Public Sub Msg_ROBOT(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
' Informação relativa a uma acção do RAX
' Guarda (procedimento SaveRA) para enviar posteriormente (via Timer_Send_Serv)
' para o servidor
    Dim C As Long
    Dim i As Integer
    On Error Resume Next

'guarda para enviar posteriormente (via Timer_Send_Serv) para o servidor
SaveRA id, parm

    If parm <> 2 Then C = RGB(255, 255, 0) Else C = RGB(255, 0, 0)
    robot(id).FillColor = C
    robot(id).Visible = parm <> 0
End Sub

```

```

Public Sub Msg_OUTP(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
' Informação relativa ao estado (STO/RUN/etc.) do SFF
' Envia para o servidor
    Dim C As Long
    Dim i As Integer
    Dim t, tx As String
    Dim o As Variant

    On Error Resume Next
    tx = ""
    Select Case parm
        Case 1: C = RGB(0, 255, 0): t = "": tx = "Run"
        Case 2: C = RGB(255, 0, 0): t = "": tx = "Error"
        Case 3: C = RGB(0, 255, 0): t = "B": tx = "Stopping"
        Case 4: C = RGB(255, 0, 0): t = "B": tx = "Error"
        Case Else: C = RGB(128, 128, 128): t = "": tx = "Stop"
    End Select
    Outp(id).BackColor = C
    Outp(id).Visible = True
    Outp(id).Tag = t

    If id = 1 Then
        'envia para o servidor
        If tx <> "" Then
            wsk1.RemoteHost = INTRANET_SERVER_ADDR
            wsk1.SendData "#OP:" & UCase(tx)
        End If

        For Each o In HMI_Form.Controls
            t = o.Name
            If Left(t, 3) = "cmd" And o.Tag = "" Then
                o.Enabled = parm = 0
            End If
        Next o
    End If
End Sub

Public Sub Msg_SA(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
' Relativa a uma acção dos Swivel Arms
    Dim C As Long
    Dim i As Integer
    On Error Resume Next
    If parm <> 2 Then C = RGB(255, 255, 0) Else C = RGB(255, 0, 0)
    sa(id).BorderColor = C
    sa(id).Visible = parm <> 0
End Sub

```

```

Public Sub Msg_CYL(ByVal id As Integer, ByVal parm As Integer)
'*** Wm recebida do RF-CON
' Relativa a uma acção dos Cilindros
    Dim C As Long
    Dim i As Integer
    On Error Resume Next
    If id >= 100 Then 'Actuator
        If Int(id / 10) = 10 Then 'Push
            C = &HFF0000
        Else 'Pull
            C = &HFFF00
        End If
        If parm = 0 Then C = RGB(128, 128, 128)
        If parm = 2 Then C = RGB(255, 0, 0)
        CylAct(id Mod 10).FillColor = C
    Else 'Sensor
        Select Case parm
            Case 1: C = RGB(255, 255, 0)
            Case 2: C = RGB(255, 0, 0)
            Case Else: C = RGB(192, 192, 192)
        End Select
        Cyl(id).FillColor = C
    End If
End Sub

Public Sub Msg_REG(ByVal id As Long, ByVal parm As Long)
'Mensagem recebida para sincronização das janelas
    Dim j As Integer
    Dim i As Variant
    Dim t As String
    On Error Resume Next
    hWnd_CMD = id
    Select Case parm
        Case 1
            SetWindowPos Me.hwnd, HWND_TOP, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOACTIVATE Or
                SWP_NOSIZE
        End Select
    For Each i In HMI_Form.Controls
        t$ = i.Name
        If Left(t$, 3) <> "cmd" And Left(t$, 3) <> "txt" Then i.Visible = False
        If Left(t$, 5) = "Queue" Then
            i.Clear
            i.Visible = True
        End If
        If Left(t$, 4) = "Buff" Then
            i.Caption = "0"
            i.Visible = True
        End If
        If t$ = "Cyl" Then
            i.FillColor = RGB(192, 192, 192)
            i.Visible = True
        End If
        If t$ = "CylAct" Then
            i.FillColor = RGB(128, 128, 128)
            i.Visible = True
        End If
    Next
    Outp(99).Visible = True
    For j = 0 To 2
        AGVCount(j) = 0
        AGVTxt(j) = ""
        RACount(j) = 0
        RATxt(j) = ""
        CamCount(j) = 0
        CamTxt(j) = ""
    Next j
End Sub

```

```

Public Sub Msg2_REG(ByVal id As Long, ByVal parm As Long)
    On Error Resume Next
    hWnd_SERVER = id
    wm1.PostMessageLong hWnd_SERVER, wmRF_Msg2_REG2, HMI_Form.hwnd, 0
End Sub

```

```

Public Sub SaveCam(ByVal id As Integer, ByVal parm As Integer)
'Chamado pela Msg_CAM
    Dim txt As String
    Dim p As Integer
    On Error Resume Next
    Select Case parm
        Case PART_UNKNOWN
            txt = "!Unknown"
        Case PART_BLACK
            txt = "Black"
        Case PART_GRAY
            txt = "Gray"
        Case PART_WHITE
            txt = "White"
        Case PART_ERROR
            txt = "!Error"
        Case PART_PICTURE
            txt = ">>Picture"
    End Select
    CamTxt(id) = TimeStamp() & " " & txt & vbCrLf & CamTxt(id)
    If CamCount(id) > 2 Then
        p = InStrRev(CamTxt(id), vbCrLf)
        If p > 0 Then
            CamTxt(id) = Left(CamTxt(id), p - 1)
        End If
    Else
        CamCount(id) = CamCount(id) + 1
    End If
End Sub

```

```

Public Sub SaveAGV(ByVal id As Integer, ByVal parm As Integer)
'Chamado pela Msg_AGV
    Dim txt As String
    Dim p As Integer
    On Error Resume Next
    txt = ""
    If parm >= 10 Then
        txt = " ERROR"
        parm = parm - 10
    End If

    Select Case parm
        Case 1
            txt = "Stoped at Local St." & txt
        Case 2
            txt = "Moving to Local St." & txt
        Case 3
            txt = "Moving to Remote St." & txt
        Case 4
            txt = "Stoped at Remote St." & txt
    End Select
    AGVTxt(id) = TimeStamp() & " " & txt & vbCrLf & AGVTxt(id)
    If AGVCount(id) > 2 Then
        p = InStrRev(AGVTxt(id), vbCrLf)
        If p > 0 Then
            AGVTxt(id) = Left(AGVTxt(id), p - 1)
        End If
    Else
        AGVCount(id) = AGVCount(id) + 1
    End If
End Sub

```

```
Public Sub SaveRA(ByVal id As Integer, ByVal parm As Integer)
'Chamado pela Msg_RA
    Dim txt As String
    Dim p As Integer
    On Error Resume Next
    txt = ""
    If parm >= 10 Then
        txt = " ERROR"
        parm = parm - 10
    End If

    Select Case parm
        Case 1
            txt = "Stoped" & txt
        Case 2
            txt = "In Operation" & txt
        Case 3
            txt = "Error" & txt
    End Select

    RATxt(id) = TimeStamp() & " " & txt & vbCrLf & RATxt(id)

    If RACount(id) > 2 Then
        p = InStrRev(RATxt(id), vbCrLf)
        If p > 0 Then
            RATxt(id) = Left(RATxt(id), p - 1)
        End If
    Else
        RACount(id) = RACount(id) + 1
    End If
End Sub
```

C.2.4 Evento `wsk1_DataArrival` (Mensagens UDP vindas do Servidor *Intranet*)

```

Private Sub wsk1_DataArrival(ByVal bytesTotal As Long)
'*** MENSAGENS WINDOWS RECEBIDAS DO SERVIDOR INTRANET
'*** São enviadas ao RF-CON via Wm
Dim t, pc_id, new_value As String
Dim p As Integer
On Error Resume Next
wsk1.GetData t, vbString

'atualização de parâmetros do Buffers
If Left(t, 1) = "&" Then
    p = InStr(t, ":")
    If p > 0 Then
        pc_id = UCase(Mid(t, 2, p - 2))
        new_value = Mid(t, p + 1)
        Select Case pc_id
            Case "B1MAX"
                cmdBuffSet_Click 1 'Envia ao RF-HMI
            Case "B2MAX"
                cmdBuffSet_Click 2 'Envia ao RF-HMI
            Case "B3MAX"
                cmdBuffSet_Click 3 'Envia ao RF-HMI
            Case "B5MAX"
                cmdBuffSet_Click 5 'Envia ao RF-HMI
        End Select
    End If
'Comando do SFF
ElseIf Left(t, 1) = "#" Then
    p = InStr(t, ":")
    If p > 0 Then
        pc_id = UCase(Mid(t, 2, p - 2))
        new_value = Mid(t, p + 1)
        Select Case pc_id
            Case "OP":
                'Comando do sistema
                If new_value = "EM_STOP" Then wm1.PostMessageLong hWnd_CMD,
                    wmRF_Msg_CMD, 0, 30
            Case "CB_SPEED":
                'Alteração da velocidade dos tapetes
                If new_value = "SLOWER" Then wm1.PostMessageLong hWnd_CMD,
                    wmRF_Msg_CMD, 1, 21
                If new_value = "FASTER" Then wm1.PostMessageLong hWnd_CMD,
                    wmRF_Msg_CMD, 1, 22
            Case 'Actuação dos dispositivos pneumáticos
            Case "SA1":
                If new_value = "ACTIVATE" Then wm1.PostMessageLong hWnd_CMD,
                    wmRF_Msg_CMD, 11, 10
            Case "SA2":
                If new_value = "ACTIVATE" Then wm1.PostMessageLong hWnd_CMD,
                    wmRF_Msg_CMD, 12, 10
            Case "CYL1":
                If new_value = "ACTIVATE" Then wm1.PostMessageLong hWnd_CMD,
                    wmRF_Msg_CMD, 21, 10
            Case "CYL2":
                If new_value = "ACTIVATE" Then wm1.PostMessageLong hWnd_CMD,
                    wmRF_Msg_CMD, 22, 10
            Case "CYL3":
                If new_value = "ACTIVATE" Then wm1.PostMessageLong hWnd_CMD,
                    wmRF_Msg_CMD, 23, 10
        End Select
    End If
End If
End Sub

```

C.2.5. Evento Timer_Send_Serv_Timer (Envio periódico info. ao servidor)

```
Private Sub Timer_Send_Serv_Timer()  
    Dim txt2 As String  
    Dim t As String  
    Dim i As Integer  
    wsk1.RemoteHost = INTRANET_SERVER_ADDR  
  
    'Informação das Filas  
    For i = 1 To 12  
        t = t & vbCrLf & "[" & i & "]" & QueueViewCtrl1(i).GetText  
    Next i  
    t = "Queue Information (@ Time = " & TimeStamp() & ") " & t  
    txt2 = "?QUEUE:" & t  
    wsk1.SendData txt2  
  
    'Informação dos buffers  
    t = ""  
    For i = 1 To 6  
        t = t & vbCrLf & "[" & i & "]" & Buff(i).Caption  
        If Buff(i).ForeColor = &HFF& Then  
            t = t & " [full]"  
        End If  
    Next i  
    t = "Buffer Information (@ Time = " & TimeStamp() & ") " & t  
    txt2 = "?BUFF:" & t  
    wsk1.SendData txt2  
  
    'Informação sobre os AGVs  
    t = "AGV 1:" & vbCrLf & AGVTxt(1)  
    txt2 = "?AGV1:" & t  
    wsk1.SendData txt2  
    t = "AGV 2:" & vbCrLf & AGVTxt(2)  
    txt2 = "?AGV2:" & t  
    wsk1.SendData txt2  
  
    'Informação sobre os braços robóticos  
    t = "Robot Arm 1:" & vbCrLf & RATxt(1)  
    txt2 = "?RA1:" & t  
    wsk1.SendData txt2  
    t = "Robot Arm 2:" & vbCrLf & RATxt(2)  
    txt2 = "?RA2:" & t  
    wsk1.SendData txt2  
End Sub
```

C.2.6. Eventos da Interface com o Utilizador da RF-HMI

```

Private Sub cmdBuffSet_Click(Index As Integer)
    wm1.PostMessageLong hWnd_CMD, wmRF_Msg_BUFF_SET, Index, txtBuffSet(Index)
End Sub

Private Sub cmdMM_MINUS_Click(Index As Integer)
    wm1.PostMessageLong hWnd_CMD, wmRF_Msg_CMD, Index, 21
End Sub

Private Sub cmdSA_Click(Index As Integer)
    wm1.PostMessageLong hWnd_CMD, wmRF_Msg_CMD, 10 + Index, 10
End Sub

Private Sub cmdMM_PLUS_Click(Index As Integer)
    wm1.PostMessageLong hWnd_CMD, wmRF_Msg_CMD, Index, 22
End Sub

Private Sub cmdCyl_Click(Index As Integer)
    wm1.PostMessageLong hWnd_CMD, wmRF_Msg_CMD, 20 + Index, 10
End Sub

Private Sub cmdMM_ON_Click(Index As Integer)
    wm1.PostMessageLong hWnd_CMD, wmRF_Msg_CMD, Index, 1
End Sub

Private Sub cmdMM_OFF_Click(Index As Integer)
    wm1.PostMessageLong hWnd_CMD, wmRF_Msg_CMD, 1, 0
End Sub

Private Sub cmdMain_Click(Index As Integer)
    wm1.PostMessageLong hWnd_CMD, wmRF_Msg_CMD, 0, Index
End Sub

Private Sub tBlink_Timer()
    Dim i As Integer

    On Error Resume Next
    For i = 1 To 4
        If InStr(Outp(i).Tag, "B") Then
            Outp(i).Visible = Not Outp(i).Visible
        End If
    Next
End Sub

```

C.2.7. A Função TimeStamp()

```

Private Function TimeStamp() As String
    Dim n As Variant
    n = Now()
    TimeStamp = p(Hour(n)) & ":" & p(Minute(n)) & ":" & p(Second(n))
End Function

Private Function p(ByVal n As String) As String
    If Len(n) = 1 Then
        p = "0" & n
    Else
        p = n
    End If
End Function

```

