

Faculdade de Engenharia da Universidade do Porto



**Interface Homem-Máquina para
Domótica baseado em tecnologias Web**

João Alexandre Oliveira Ferreira

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Prof. Dr. Mário de Sousa

Julho de 2008

A Dissertação intitulada

“Interface Homem/Máquina para domótica com tecnologias Web”

foi aprovada em provas realizadas em 17/Julho/2008

o júri

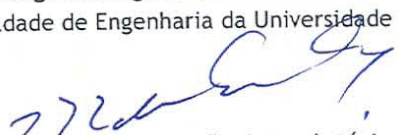
Presidente Professor Doutor Rui Manuel Esteves Araújo
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto



Professor Doutor José Manuel Tavares Vieira Cabral
Professor Auxiliar da Escola de Engenharia da Universidade do Minho

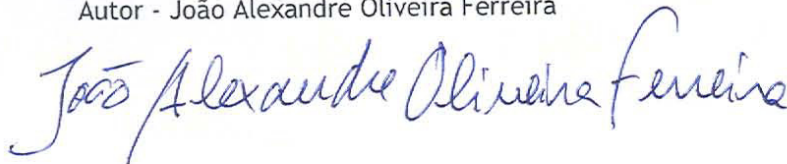


Professor Doutor Mário Jorge Rodrigues de Sousa
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto



O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - João Alexandre Oliveira Ferreira



Faculdade de Engenharia da Universidade do Porto

Resumo

A instalação de um sistema de automação numa casa ou num edifício vem adquirindo uma importância crescente numa sociedade moderna e evoluída.

A automatização de tarefas que até agora eram feitas de uma forma manual, proporciona uma melhor qualidade de vida aos utilizadores deste tipo de sistemas.

O conjunto de serviços prestados pelos equipamentos que automatizam essas tarefas é direccionado à gestão de quatro funções essenciais: o conforto, a eficiência energética, a segurança e as comunicações. É sobre este último aspecto que esta dissertação se vai focar.

A evolução das redes de transmissão de dados, da microelectrónica, e o aparecimento de novas tecnologias ou metodologias destinadas à integração *Web* de serviços, contribuiram para a manifestação de novas possibilidades abrangidas pelo campo das comunicações na domótica.

Num mundo onde as pessoas estão em constante movimento, é objectivo do sistema de domótica tornar a vida e o trabalho um pouco mais fáceis - fornecendo, por exemplo, um interface intuitivo que forneça ao utilizador a autonomia de poder controlar todas as funções de gestão da sua habitação, casa de férias ou escritório via PDA, telemóvel, *smartphone*, computador ou qualquer outro dispositivo com acesso à Internet, quer esteja em casa ou em viagem.

Nesta dissertação irão ser estudadas diversas tecnologias que servirão de base a uma plataforma de baixo custo, baixo consumo e sem código proprietário, cuja função essencial é monitorizar e controlar uma instalação de domótica do standard KNX via *Web* através de qualquer dispositivo com acesso à Internet.

Com base no estudo efectuado, serão propostas várias abordagens para a concepção de uma arquitectura funcional base para o sistema.

Depois de uma análise das vantagens e desvantagens de cada uma das abordagens propostas, será escolhida a que reúna as melhores características, de acordo com os objectivos propostos.

Na fase seguinte, dar-se-á o processo de implementação de um exemplo da arquitectura escolhida e, por fim, a validação do mesmo.

Abstract

The installation of an automation system in house or a building has been assuming a major role in a modern and developed society.

The automation of tasks, which used to be performed manually, offers a better quality of life to those who use this kind of systems.

The set of services performed by the equipments responsible for the automation of these tasks is directed towards four main functions: comfort, energetic efficiency, safety and communications. This dissertation will be focusing this last aspect.

The development of data transmitting networks, of microelectronics, and the coming out of new technologies or methodologies whose aim is the Web integration of services have contributed to the emergence of new possibilities in the field of communications in home automation.

In a world in which people are absorbed in a constant movement, it is the aim of the home automation system to make life and work easier – allowing the possibility of an intuitive interface, for instance – which provides the user with the autonomy of controlling every single management function of his house, holiday home or office by means of PDA, mobile phone, smartphone, personal computer or any other device connected to the internet, while remaining at home or travelling.

This dissertation will focus the study of a wide range of technologies which will work as a basis for a low-cost platform, low-consumption and no proprietary code, whose main function is to monitor and control a home automation installation of KNX standard via Web by means of any device connected to the internet.

Based on the study performed, some approaches for the conception of a basis functional architecture for the system will be outlined.

After detailed analysis of advantages and disadvantages of each approach presented, the one with the best characteristics, according to the aims previously set, will be chosen.

The next stage will focus on the implementation process of an example of the chosen architecture, as well as the validation of the referred example.

Ao meu Avô, Família e Amigos

Agradecimentos

Ao longo deste trabalho foram vários os que contribuíram para que fosse possível a realização desta dissertação.

Agradeço, em primeiro lugar, ao meu orientador, o Professor Doutor Mário de Sousa, por todo o apoio prestado ao longo das várias etapas desta dissertação.

Agradeço à Faculdade de Engenharia pela disponibilização de todos os meios necessários, e ao técnico Nuno Guerra pelo apoio prestado no laboratório.

Agradeço, em particular, aos meus primos, Rita e Filipe, por todo o apoio prestado, necessário para a conclusão desta dissertação.

Índice

1. Introdução	1
1.1 Enquadramento	1
1.2 Motivação	2
1.3 Objectivos	3
1.4 Estrutura do relatório	4
2. Domótica	5
2.1 Introdução	5
2.2 Arquitectura	5
2.3 Funcionalidades	8
2.3.1 O conforto	8
2.3.2 A eficiência energética	9
2.3.3 A segurança	9
2.3.4 As comunicações	10
2.4 Sistemas comerciais	11
3. Tecnologias	15
3.1 O standard KNX/EIB	15
3.1.1 Introdução	15
3.1.2 Principais características	16
3.1.3 Topologia da rede	17
3.1.4 Meios de transmissão	18
3.1.4.1 Par entrançado	19
3.1.4.2 Rede eléctrica	20
3.1.4.3 Wireless	20
3.1.4.4 IP	21
3.1.5 Modos de configuração	22
3.1.5.1 Modo S	22
3.1.5.2 Modo E	22
3.1.5.3 Modo A	23
3.1.6 Modo de endereçamento de dispositivos	23

3.1.7	Comunicação entre dispositivos	24
3.2	Tecnologias Web.....	26
3.2.1	JavaScript.....	26
3.2.2	AJAX	28
3.2.3	jQuery.....	29
3.2.4	CGI.....	30
3.2.5	JAVA	31
3.2.6	PHP.....	34
3.3	Sistemas Embebidos.....	35
3.3.1	Sistemas Operativos para sistemas embebidos.....	35
3.3.1.1	Linux embebido.....	36
3.3.2	Cross-Compiling.....	37
4.	Arquitectura Funcional	39
4.1	Primeira abordagem	39
4.1.1	Primeira alternativa.....	40
4.1.2	Segunda alternativa.....	41
4.1.3	Terceira alternativa	42
4.2	Segunda abordagem	43
4.3	Terceira abordagem.....	44
4.3.1	KNX @ HOME	44
4.3.1.1	KNXService	44
4.3.1.2	KNXWeb	45
4.3.1.3	KNXAdmin.....	46
4.4	Escolha da arquitectura funcional	46
4.4.1	Considerações sobre a primeira abordagem	47
4.4.2	Considerações sobre a segunda abordagem.....	47
4.4.3	Considerações sobre a terceira abordagem.....	47
4.4.4	Conclusão	48
5.	Validação experimental	49
5.1	Descrição do <i>hardware</i>	49
5.1.1	Sistema embebido para a plataforma do servidor	49
5.1.1.1	ICnova AP7000 Base.....	49
5.1.1.2	Picotux	51
5.1.1.3	Router	52

5.1.1.4	Avaliação da plataforma para o servidor.....	54
5.1.2	Sistema de testes	55
5.1.3	Gateway KNX/EIB	58
5.1.4	Topologia do sistema de monitorização e controlo Web.....	59
5.2	Software.....	60
5.2.1	Primeira fase.....	60
5.2.1.1	Aplicação CGI de interface com a rede KNX/EIB	60
5.2.1.2	Interface <i>Web</i> exemplo para a validação.....	62
5.2.2	Segunda fase.....	65
5.3	Testes e resultados.....	67
5.3.1	Primeira fase.....	67
5.3.2	Segunda fase.....	67
6.	Conclusão e Perspectivas de Desenvolvimento	69
ANEXO 1		75

Lista de Figuras

FIGURA 2.1 – ARQUITECTURA CENTRALIZADA	6
FIGURA 2.2 – ARQUITECTURA DESCENTRALIZADA	7
FIGURA 2.3 – ARQUITECTURA DISTRIBUÍDA	7
FIGURA 2.4 – ARQUITECTURA DO SISTEMA VIVIMAT [37]	12
FIGURA 2.5 – VISÃO GERAL DO CONTROLO WEB DO SISTEMA DOMUS-INT [38].....	13
FIGURA 2.6 – CONSOLA TÁCTIL DO SISTEMA CARDIO [40].....	14
FIGURA 3.1 – TOPOLOGIA DE UMA REDE KNX/EIB	18
FIGURA 3.2 – TOPOLOGIA DOS ENDEREÇOS FÍSICOS	23
FIGURA 3.3 – NÍVEIS DOS ENDEREÇOS DE GRUPO.....	24
FIGURA 3.4 – INTERACÇÃO ASSÍNCRONA DAS APLICAÇÕES AJAX [16]	29
FIGURA 3.5 – INTERACÇÃO SÍNCRONA DAS APLICAÇÕES WEB TRADICIONAIS [16]	29
FIGURA 3.6 – DIAGRAMA DE SEQUÊNCIA CGI SIMPLES	31
FIGURA 3.7 – SEQUÊNCIA DE UM PROGRAMA JAVA SIMPLES	33
FIGURA 3.8 – PICOTUX [26].....	37
FIGURA 4.1 – ARQUITECTURA PROPOSTA NA 1ª ALTERNATIVA DA 1ª ABORDAGEM.....	40
FIGURA 4.2 – ARQUITECTURA PROPOSTA NA 2ª ALTERNATIVA DA 1ª ABORDAGEM.....	41
FIGURA 4.3 – ARQUITECTURA PROPOSTA NA 3ª ALTERNATIVA DA 1ª ABORDAGEM.....	42
FIGURA 4.4 – ARQUITECTURA PROPOSTA NA 2ª ABORDAGEM.....	43
FIGURA 4.5 – KNXWEB	45
FIGURA 4.6 – KNXADMIN.....	46
FIGURA 5.1 – ICNOVA AP7000.....	50
FIGURA 5.2 – PICOTUX	51
FIGURA 5.3 – <i>ROUTER</i> OEM DISPONÍVEL PARA <i>DESIGNS</i> DE TERCEIROS.....	52
FIGURA 5.4 – <i>ROUTER</i> COMERCIAL DISPONÍVEL PARA O UTILIZADOR FINAL	54
FIGURA 5.5 – VISTA FRONTAL DO PAINEL EIB	56
FIGURA 5.6 – VISTA TRASEIRA DO PAINEL EIB	56
FIGURA 5.7 – LEGENDA DA INSTALAÇÃO EIB.....	57
FIGURA 5.8 – GATEWAY IP SIEMENS N148/21	58
FIGURA 5.9 – TOPOLOGIA DO SISTEMA DE MONITORIZAÇÃO E CONTROLO <i>WEB</i>	59
FIGURA 5.10 – ESTRUTURA DE APLICAÇÃO DO SERVIDOR	60
FIGURA 5.11 – DIAGRAMA DE EXECUÇÃO DO CGI.....	61

FIGURA 5.12 – INTERFACE WEB COM DISPOSITIVO DESACTIVADO	62
FIGURA 5.13 – INTERFACE WEB COM DISPOSITIVO ACTIVADO	63
FIGURA 5.14 – INTERFACE WEB COM ERRO NO DISPOSITIVO	63
FIGURA 5.15 – NOVA TOPOLOGIA DO SISTEMA DE MONITORIZAÇÃO E CONTROLO WEB	66

Lista de Tabelas

TABELA 3.1 – TIPOS DE DADOS EIS	25
TABELA 5.1 – <i>HARDWARE</i> ICNOVA AP7000	50
TABELA 5.2 – <i>SOFTWARE</i> ICNOVA AP7000	50
TABELA 5.3 – <i>HARDWARE</i> PICOTUX	51
TABELA 5.4 – <i>SOFTWARE</i> PICOTUX	51
TABELA 5.5 – <i>HARDWARE</i> ROUTER OEM	53
TABELA 5.6 – <i>SOFTWARE</i> ROUTER OEM	53
TABELA 5.7 – <i>HARDWARE</i> ROUTER LINKSYS WRT54G	53
TABELA 5.8 – DISPOSITIVOS PRESENTES POR DIVISÃO NA SIMULAÇÃO	55

Lista de acrónimos

- API** - *Application Programming Interface*
- AVAC** - *Aquecimento, Ventilação e Ar Condicionado*
- EIS** - *EIB Interworking Standards*
- EIBA** - *European Installation Bus Association*
- EIB** - *European Installation Bus*
- ETS** - *EIB Tool Software*
- EHS** - *European Home System*
- FEUP** - *Faculdade de Engenharia da Universidade do Porto*
- GSM** - *Global System for Mobile communications*
- GPRS** - *General Packet Radio Service*
- HSDPA** - *High-Speed Downlink Packet Access*
- HTML** - *HyperText Markup Language*
- IP** - *Internet Protocol*
- KNX/EIB** - *Protocolo aberto da associação Konnex*
- ODM** - *Original Design Manufacturer*
- OEM** - *Original Equipment Manufacturer*
- PC** - *Computador pessoal (personal computer)*
- PDA** - *Personal Digital Assistant*
- SMS** - *Short Message Service*
- TCP** - *Transmission Control Protocol*
- UMTS** - *Universal Mobile Telecommunication System*
- UPS** - *Uninterruptible Power Supply*
- WAP** - *Wireless Application Protocol*
- WWW** - *World Wide Web*
- XML** - *Extensible Markup Language*

Capítulo 1

1. Introdução

1.1 Enquadramento

Este trabalho surge no seguimento de outros projectos já desenvolvidos na FEUP no âmbito da domótica de baixo custo. Nestes projectos foram desenvolvidos módulos sensores e actuadores de baixo custo, um servidor compatível com a norma KNX/EIB e um painel de simulação didáctico com tecnologia EIB da *Siemens*. De modo a poder interligar todos estes projectos entre si e a oferecer a possibilidade de monitorização e controlo remoto sobreveio a ideia deste projecto.

Nesta dissertação irão ser estudadas diversas tecnologias que servirão de base a uma plataforma de baixo custo, baixo consumo e sem recorrer a código proprietário, cuja função essencial é supervisionar, monitorizar e controlar uma instalação de domótica do *standard* KNX/EIB via *Web* através de qualquer dispositivo com acesso à Internet.

Para tal, serão estudadas as tecnologias que permitem suportar esta implementação e, com base neste estudo, serão propostas várias abordagens para a arquitectura funcional do sistema. Depois de escolhida a melhor arquitectura, proceder-se-á à sua implementação e validação com um exemplo de aplicação.

1.2 Motivação

A motivação principal para a realização deste trabalho resulta da importância crescente que a automação de casas e edifícios vem adquirindo numa sociedade moderna e evoluída.

São coisas simples - como a redução do tempo gasto nas rotinas diárias, a redução dos consumos de energia (tornada possível por uma maior eficiência na regulação da iluminação e climatização) e a segurança oferecida pela detecção de possíveis inundações, fugas de gás, incêndios e intrusos - que possibilitam uma melhoria significativa na qualidade de vida e na autonomia da população, principalmente em pessoas que requerem cuidados especiais.

Contudo, os custos de uma instalação de domótica são elevados e, normalmente, são utilizados sistemas proprietários, muito pouco flexíveis, e que tornam o aumento das suas capacidades inicialmente projectadas muito difícil e dispendioso.

Desde há alguns anos para cá, com a constante evolução da electrónica no campo dos sensores/actuadores e das redes de comunicações de dados, é-nos possível, agora, dispor de uma grande variedade de equipamentos com preços competitivos e com a capacidade de se integrarem numa rede de comunicação.

Tudo isto, juntamente com a massificação dos serviços de *Internet* de banda larga e a evolução das redes móveis de alto débito 3G e 3.5G, UMTS e HSDPA respectivamente (que disponibilizam acesso à *Internet* em qualquer lugar para PDAs, *Smartphones*, telemóveis e computadores), abre caminho à possibilidade de controlar um edifício em qualquer lugar e a qualquer hora, dispondo de um acesso à *Internet* e de um *Web browser*.

Deste modo, com um sistema de controlo baseado na *Web*, é possibilitada aos utilizadores uma elevada autonomia no que respeita ao controlar um edifício à distância. Procedimentos como o acesso às imagens das várias câmaras de videovigilância, gestão de alarmes de intrusão, fogo, inundações, fugas de gás e controlo do sistema de rega de uma moradia, são agora possíveis, enquanto os seus habitantes se encontram calmamente de férias do outro lado do mundo.

Estão, então, reunidas as condições para que, face à tecnologia existente, haja possibilidade para que uma implementação de baixo custo (sem recorrer a sistemas proprietários que comportam custos elevados) e que forneça a possibilidade de controlar uma instalação de domótica à distância, seja mais atractiva e fomente a massificação da domótica.

1.3 Objectivos

O objectivo deste trabalho consiste no estudo das tecnologias *Web* actuais, das implementações já existentes ou em desenvolvimento para o *interface* de uma instalação de domótica com a WWW, das plataformas de *hardware* que possam suportar estas implementações e dos aspectos fundamentais do protocolo de domótica KNX/EIB. Depois, com base no estudo efectuado, projectar uma arquitectura funcional para o sistema, implementá-la e, por fim, validá-la.

Os objectivos específicos deste trabalho são os seguintes:

- Estudo das tecnologias *Web* actualmente existentes numa perspectiva de proporcionar uma comunicação do tipo cliente-servidor e que permita interagir com uma instalação de domótica KNX/EIB com a maior eficiência possível.
- Estudo das implementações existentes actualmente ou em desenvolvimento, na sua versão de código livre, que possam servir parcial ou totalmente para o suporte da interface de comunicação com uma rede física de domótica KNX/EIB e da interface *Web* para o seu controlo à distância.
- Estudo da arquitectura base do protocolo de comunicação KNX/EIB para redes de domótica.
- Com base no estudo efectuado, propor diversas abordagens para o problema da concepção de uma arquitectura funcional para o sistema a implementar, e, dependendo das vantagens e desvantagens de cada uma dessas abordagens, seleccionar a mais conveniente.
- Estudo das plataformas de hardware de baixo custo e baixo consumo energético existentes no mercado que permitam suportar a implementação de um sistema de controlo à distância para instalações KNX/EIB de acordo com a abordagem escolhida.
- Após a definição da arquitectura funcional, passar à fase da implementação da mesma.
- Por fim, validar a implementação com um exemplo de aplicação simples.

1.4 Estrutura do relatório

Esta dissertação encontra-se estruturada em 6 capítulos dos quais, o primeiro é composto por esta introdução ao trabalho.

No segundo capítulo é apresentado o conceito de domótica, as arquitecturas possíveis deste tipo de sistemas e uma abordagem sobre as suas funcionalidades.

No terceiro capítulo serão estudadas diversas tecnologias que poderão servir de base ao problema proposto, bem como o essencial do *standard* aberto KNX/EIB.

No quarto capítulo, com base no estudo do capítulo anterior, são propostas diversas arquitecturas funcionais para a implementação do sistema e escolhida a mais conveniente.

No quinto capítulo será feita a implementação de um exemplo prático simples, de modo a poder validar a abordagem seleccionada sobre a concepção da arquitectura funcional e os principais resultados dessa implementação.

O sexto e último capítulo contém as conclusões gerais sobre o que foi feito neste trabalho.

Capítulo 2

2. Domótica

Neste capítulo será apresentado o tema domótica juntamente com as suas principais características.

2.1 Introdução

A palavra domótica deriva das palavras *Domus* (casa) e Robótica (controlo automatizado de algo), ou seja, a domótica define-se como a possibilidade do controlo de forma automática de habitações, tornando-as no que vulgarmente se costuma designar por casas inteligentes.

Podemos observar este conceito a partir de duas posições distintas.

Do ponto de vista de um utilizador de uma habitação automatizada, a domótica pode ser vista como um sistema que lhe proporciona uma melhor qualidade de vida através do uso das novas tecnologias, oferece uma redução do trabalho doméstico, um aumento do bem-estar e da segurança dos habitantes e uma melhor racionalização no consumo energético da habitação.

Do ponto de vista tecnológico, a definição de domótica pode ser vista como um sistema que integra diversos equipamentos domésticos que têm a capacidade de comunicar entre si através de um canal de comunicação, para que possam desempenhar tarefas de forma autónoma, que até agora eram feitas de forma manual [1][2][3].

2.2 Arquitectura

Tanto os sistemas de automação de habitações (normalmente instalações de pequena escala no contexto residencial), como os sistemas de automação de edifícios (tipicamente instalações de elevada dimensão como hotéis e centros comerciais) visam uma melhoria da interacção e da comunicação entre os dispositivos encontrados normalmente nestes edifícios.

Os requisitos, a complexidade e o tipo de arquitecturas inerentes a estas duas situações são diferentes, mas em ambos os casos é óbvia a necessidade de comunicação entre os sensores e os respectivos actuadores. Para isso utilizaram-se barramentos de comunicação de dados, com várias semelhanças das já bastante disseminadas, redes de campo (utilizadas no domínio da automação industrial).

A classificação da arquitectura dos sistemas de automação é feita com base no local onde se encontra a “inteligência” do sistema domótico. Podemos dispor de uma arquitectura centralizada, uma arquitectura descentralizada, uma arquitectura distribuída e uma arquitectura que é um misto das anteriores.

Num sistema de domótica, uma arquitectura centralizada significa que existe um controlador central que, de acordo com o programa nele executado, os dados que recebe dos sensores e a informação introduzida pelos utilizadores, actua em conformidade nas saídas - os actuadores.

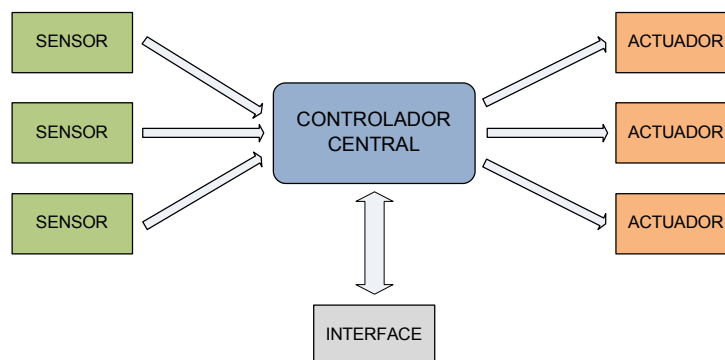


Figura 2.1 – Arquitectura centralizada

Numa arquitectura descentralizada, ao contrário da arquitectura anterior, existem vários controladores distribuídos (cada um com os seus sensores e actuadores locais), interligados entre si por um barramento de dados para trocarem informação.

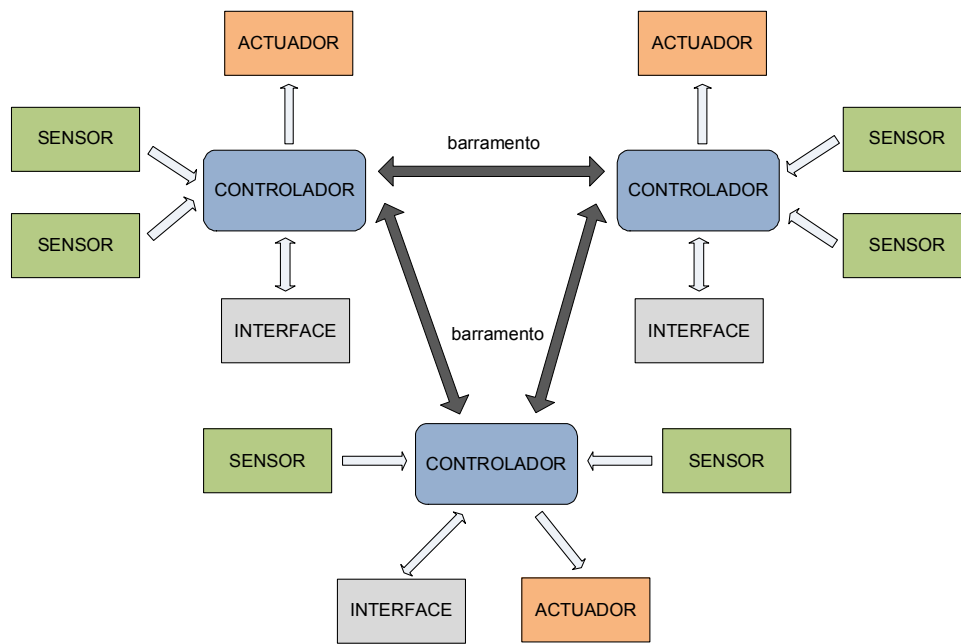


Figura 2.2 – Arquitetura descentralizada

Num sistema de domótica, uma arquitetura distribuída caracteriza-se pelo facto de cada elemento do sistema, seja ele um sensor, um actuador ou um simples interface, ser também um controlador capaz de actuar e enviar informação para um barramento de dados - de acordo com o algoritmo nele executado, de acordo com os dados adquiridos por ele próprio (sensor, por exemplo) e de acordo com os dados recebidos de outros dispositivos do barramento (actuador, por exemplo) [4].

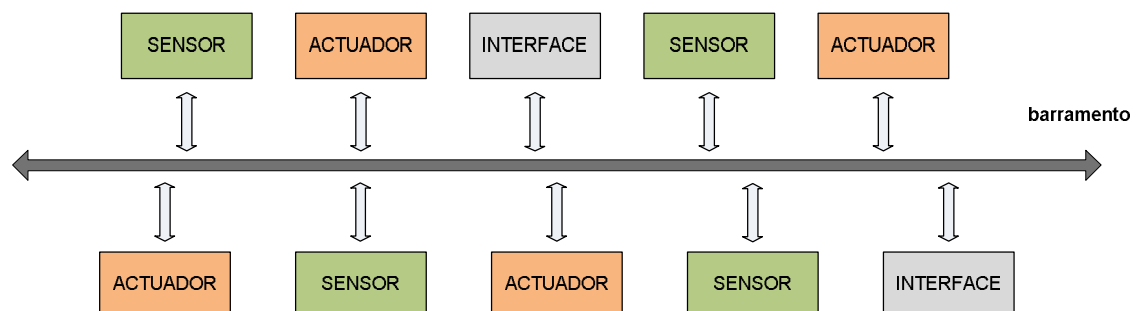


Figura 2.3 – Arquitetura distribuída

2.3 Funcionalidades

Como já anteriormente referenciado, a domótica pode ser vista como um conjunto de serviços prestados por um equipamento automático ou dispositivos com um certo nível de "inteligência" dentro de uma habitação, direccionados à gestão de quatro funções essenciais: o conforto, a eficiência energética, a segurança e as comunicações.

A proporção do investimento feito em cada uma destas funções, aquando da instalação de uma rede de domótica, dependerá de qual a finalidade do edifício. Diferentes necessidades, requerem diferentes meios.

Apesar de, em muitos aspectos, as quatro funções atrás referenciadas se sobreporem, vamos tentar diferenciar o domínio de cada uma delas [1][2][3].

2.3.1 O conforto

O conceito de conforto é essencialmente destinado às instalações AVAC (aquecimento, ventilação e ar condicionado), embora também se possam incluir nesta área todos os outros sistemas que possam contribuir para a comodidade e bem-estar dos utilizadores dos edifícios que integram redes de domótica.

Na década de 70, foi efectuado um grande investimento nos sistemas AVAC, uma vez que foram estes os primeiros sistemas de edifícios a serem electronicamente controlados.

Apesar da importância do controlo de sistemas AVAC também abranger, em grande parte, o consumo de energia, esta deve-se fundamentalmente à presença deste tipo de sistemas em quase todas as instalações efectuadas hoje em dia. Devido à sua topologia, torna-se necessário que o controlo deste tipo de sistemas seja o mais distribuído possível, ou seja, que em cada divisão ou local exista um sistema de controlo individual.

Para além dos sistemas AVAC, podemos exemplificar outras funções clássicas, aplicáveis no domínio do conforto:

- Controlo por infravermelhos dos vários automatismos presentes no edifício
- Automatização da irrigação de jardins
- Abertura e fecho automático de portas
- Controlo e supervisão de todos os sistemas do edifício
- Accionamento automático de vários sistemas com base em dados do meio ambiente, como por exemplo a recolha de toldos e o fecho de persianas e janelas em caso de tempestade ou vento forte.

2.3.2 A eficiência energética

A exigência da optimização dos consumos energéticos é uma realidade que implica não a ausência do consumo, mas sim a sua gestão e quando necessário e sua racionalização (falhas de energia recorrendo a uma UPS, por exemplo). O grande objectivo é o de satisfazer necessidades domésticas com o mais baixo custo.

O aproveitamento da energia e redução do seu consumo é um dos aspectos mais importantes da instalação de um sistema de domótica. As acções destinadas a reduzir o consumo estão intimamente ligadas à integração de todos os dispositivos da habitação no sistema de domótica e, destas, podemos destacar:

- O aproveitamento das tarifas bi-horárias de energia para agendamento do uso dos equipamentos domésticos com cargas mais elevadas como, por exemplo, máquinas de lavar roupa e lavar louça;
- Detecção de fontes de perdas nos sistemas de climatização como, por exemplo, a suspensão do funcionamento do sistema em zonas onde sejam detectadas portas ou janelas abertas;
- Redução do consumo do ar condicionado, na ausência de utilizadores nas várias divisões através da detecção automática de presença;
- Actuação sobre as persianas de modo a que seja aproveitada a luz solar, no caso do pré-aquecimento de uma divisão juntamente com o sistema de ar condicionado, por exemplo;
- Monitorização dos consumos de água e/ou de gás por zonas, podendo, desta forma, detectar possíveis fugas ou actos de vandalismo num edifício público.

2.3.3 A segurança

Actualmente, a segurança constitui uma preocupação crescente, sendo cada vez maior o número de interessados que colocam o assunto Segurança no topo das suas prioridades.

Esta função, a segurança, pode ser dividida em duas áreas: a segurança de pessoas e a segurança de bens.

Na categoria da segurança de pessoas podem ser incluídas tarefas como:

- Iluminação automática em zonas de risco. Detectando a presença de uma pessoa e avaliando o grau de luminosidade da área, é possível calcular o grau de risco e actuar em conformidade com a situação. A título de exemplo, se um utilizador se levantar durante a noite com o intuito de se dirigir à casa de banho, o sistema detecta automaticamente a pessoa no corredor e o grau de luminosidade no local, e actua sobre as luzes de presença no corredor.

- Detecção de fugas de gás em várias divisões críticas de uma habitação (por exemplo, os quartos), abrindo válvulas de emergência para extrair o gás para o exterior.
- Alarmes de emergências médicas. Em caso da existência de pessoas com necessidades especiais (como idosos e pessoas incapacitadas), existem accionamentos de emergência cuja activação gera um aviso pré-definido anteriormente para o telemóvel de um familiar ou para os serviços de emergência.

Na categoria da segurança de bens podem-se destacar funções como:

- Detecção de intrusos com diversos sensores de presença no interior da habitação e sensores magnéticos de detecção de abertura de portas e janelas;
- Detecção de possíveis focos de incêndio no interior de uma habitação, actuando sobre os aspersores de emergência na divisão onde foi detectada a anomalia;
- Alarmes de inundações e fugas de gás;
- Simulação de presença na habitação, aquando de uma ausência prolongada desta por parte dos proprietários, actuando sobre a iluminação e os estores de uma forma aleatória.

2.3.4 As comunicações

Neste sentido, existem várias possibilidades, dependendo do tipo de edifício.

A evolução e o surgimento de novas tecnologias no domínio das telecomunicações e das redes de transmissão de dados, bem como o facto de sistemas domóticos avançados poderem ser baseados no uso destes tipos de redes, faz com que este seja um terreno fértil para a investigação e o desenvolvimento de novas arquitecturas e sistemas de integração.

Como já foi referenciado, a evolução das redes de transmissão de dados, a evolução da microelectrónica, com um elevado nível de integração na área dos semicondutores e o aparecimento de novas tecnologias ou metodologias destinadas à integração *Web* de serviços, contribuíram para a manifestação de novas possibilidades abrangidas no campo das comunicações na domótica.

De entre todas as possibilidades abrangidas nesta área, são alvo de particular investimento e desenvolvimento as iniciativas de supervisão, controlo e monitorização das instalações de domótica à distância.

Dentro desta área podemos destacar dois métodos:

- Controlo de instalações de domótica através de mensagens SMS, que consiste na implementação da tecnologia GSM/GPRS para o controlo remoto da instalação de domótica. O sistema poderá responder para o telemóvel do utilizador com a informação de vários alarmes. Por exemplo, se for detectada uma intrusão no edifício, pode ser enviada uma imagem da câmara de segurança da divisão onde foi detectada a presença indesejada.
- Controlo de instalações de domótica através via TCP/IP usando um *browser* e tecnologias *Web* para a implementação de serviços. Este método será o foco desta dissertação.

2.4 Sistemas comerciais

Actualmente, existem numerosas soluções comerciais baseadas em vários protocolos criados para o efeito dos sistemas de automação de edifícios.

Podem ser utilizados sistemas inicialmente desenvolvidos nos Estados Unidos, como o X-10, o CEBus (*Consumer Electronics Bus*), o Smart House e o LonWorks, ou sistemas inicialmente desenvolvidos na Europa, como o BatiBUS, o EIB (*European Installation Bus*) e o EHS (*European Home Systems*).

Já no final de década de 90, surge a associação *Konnex*, que resulta da fusão entre a EIBA, BCI e EHSA, associações responsáveis pelo sistema EIB, BatiBUS e EHS, respectivamente. Esta nova associação tinha por objectivo, definir um único protocolo, aberto e padronizado internacionalmente, para os sistemas de automação de habitações e edifícios, o KNX. O estudo presente nesta dissertação vai-se focar sobre este protocolo.

Existem, também, vários sistemas comerciais que utilizam protocolos proprietários, podendo, ou não, oferecer compatibilidade com os protocolos *standards* já definidos.

Dos sistemas disponíveis em Portugal, capazes de oferecer a monitorização e o controlo da instalação de domótica via *Web*, podemos destacar:

A) *VIVIMAT*

O sistema domótico VIVIMAT é um sistema proprietário desenvolvido pela empresa espanhola Dinitel, e utiliza o protocolo RS-485 como o nível físico de transmissão.

Este é um sistema centralizado que pode ser ampliado com a introdução de módulos adicionais, interligados através de um barramento de comunicação.

Ajusta-se, portanto, às necessidades de todo o tipo de casas de nova construção, e permite o controlo e manutenção local e remota através de teclado, computador, painel de visualização, telefone, WAP e *Internet* [35][36].

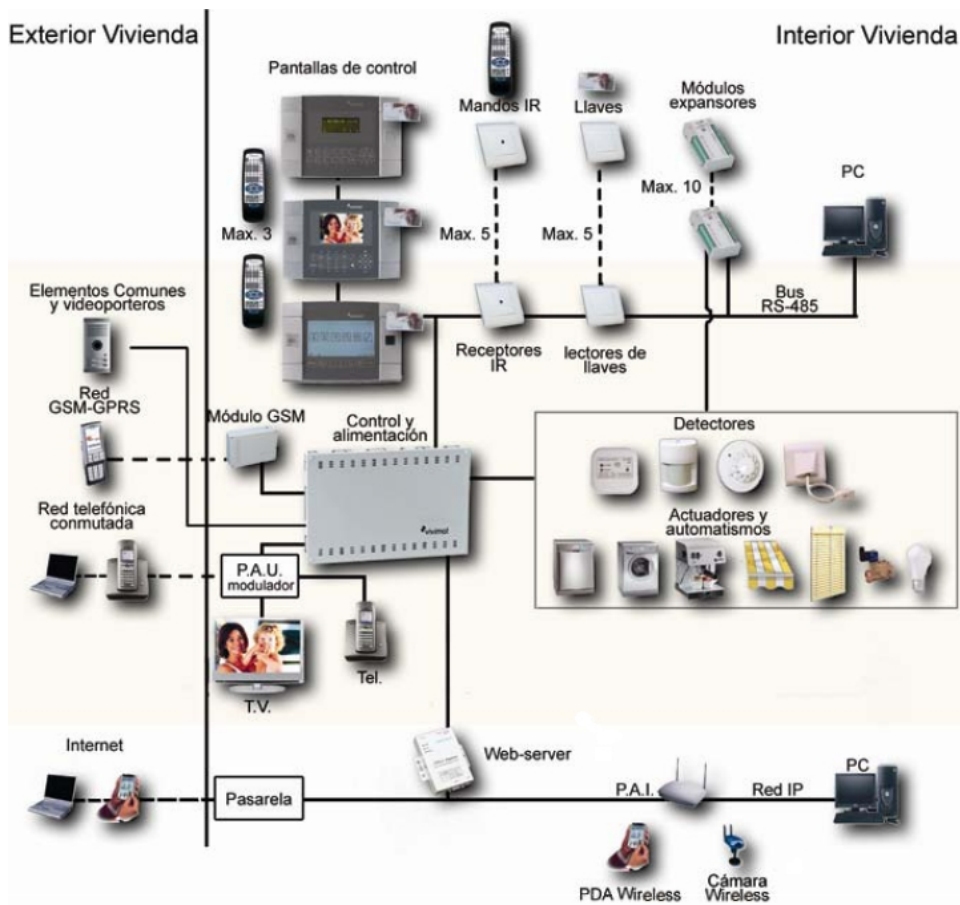


Figura 2.4 – Arquitectura do sistema VIVIMAT [37]

B) DOMUS-INT

Este sistema proprietário, desenvolvido em Portugal pela empresa JG Domótica, baseia-se num ecrã táctil que incorpora o processamento da informação. A este painel é ligado um cabo bifilar, ao qual estão ligados em anel os módulos sensores. Em cada divisão da casa é instalado um destes módulos - que incorpora como entradas um receptor de infravermelhos, um sensor de movimento, um sensor de luminosidade e um sensor de temperatura, e, como saídas, um emissor de infravermelhos, dois interruptores para controlo de iluminação e um outro para controlo de aquecimento. As persianas são controladas por módulos centralizados num quadro próprio (com uma ligação ao painel táctil). Este sistema pode ser controlado por painel táctil, SMS, *Internet* e a visualização do estado do sistema pode ser feita na televisão. As suas limitações assentam na impossibilidade de regulação da intensidade luminosa [35][38][39].

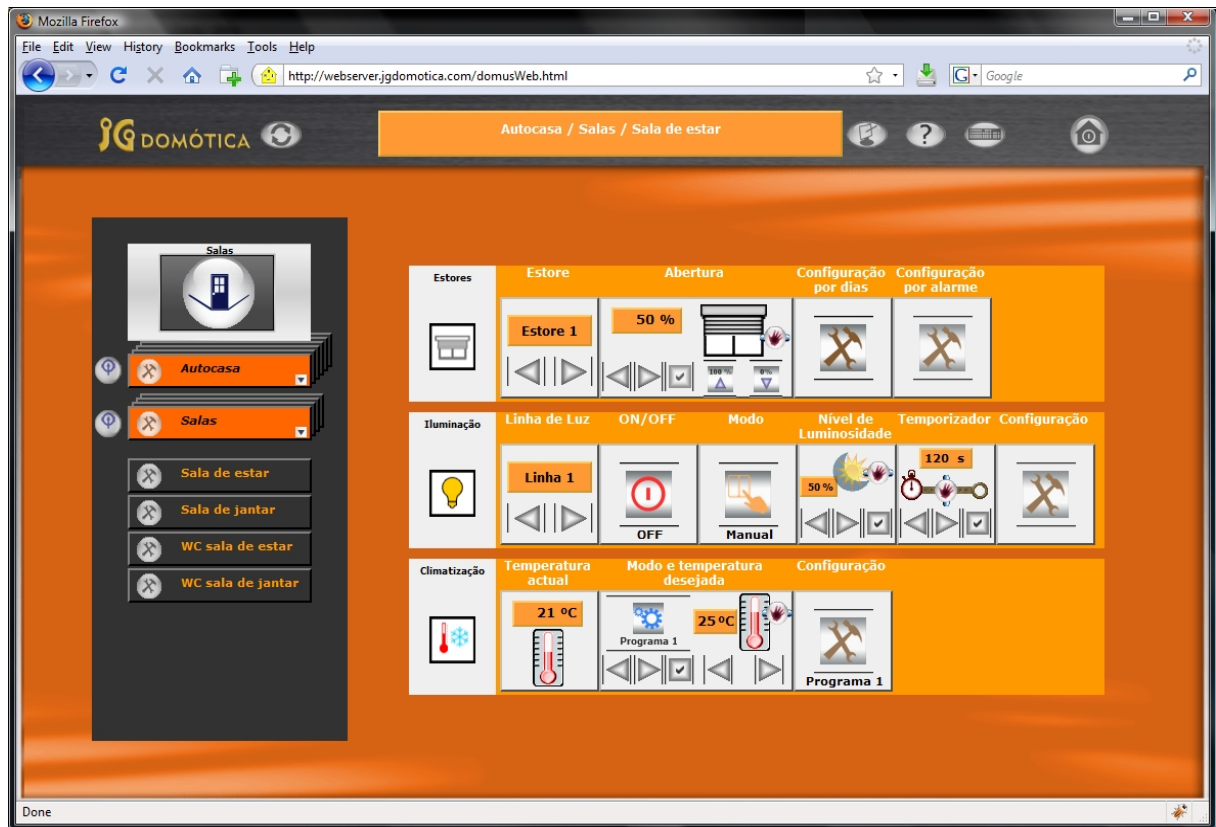


Figura 2.5 – Visão geral do controlo Web do sistema DOMUS-INT [38]

C) *CARDIO*

O *CARDIO* é um sistema proprietário desenvolvido pela empresa canadiana *SECANT Home Automation*. É um sistema centralizado, controlado localmente por uma consola táctil, e pode facilmente acoplar-se à instalação eléctrica de uma vivenda (tanto em construção, como já construída), com algumas modificações relativamente à instalação normal. Além disso, permite controlar qualquer dispositivo do protocolo X10, injectando sinais na rede eléctrica através do interface X10 fornecido [35][40][41].

O controlo de todas as funções deste sistema pode fazer-se de várias formas:

- A partir da consola táctil colocada em qualquer ponto da habitação. Uma série de ícones e submenus guiam o utilizador para efectuar o controlo da sua casa de forma directa e personalizada.
- A partir de uma consola táctil móvel, onde o utilizador pode movimentar-se pela casa e controlar assim as funções da mesma.
- De qualquer telefone da casa, sendo guiado por uma voz em português através de menus de opções.

- De qualquer telefone exterior (incluindo telemóveis).
- De um computador (interior ou exterior através da *Internet*).



Figura 2.6 – Consola táctil do sistema CARDIO [40]

Capítulo 3

3. Tecnologias

Para a elaboração deste projecto, foi feita, numa primeira fase, uma análise prévia do tipo de tecnologia disponível de modo a suportar os objectivos pretendidos. Neste capítulo, serão abordadas algumas das tecnologias mais importantes que poderão ser aplicadas no caso em estudo.

3.1 O standard KNX/EIB

3.1.1 Introdução

Em Maio de 1999, membros das associações *BatiBUS Club International* (BCI), da *European Installation Bus Association* (EIBA) e da *European Home Systems Association* (EHSA) fundaram a *Konnex Association*. O objectivo principal desta associação era a promoção de um novo e comum *standard* para a implementação de sistemas de automação de edificios e habitações. Este *standard*, denominado KNX, é baseado na tecnologia já bem estabelecida do EIB, mas agora alargado com os mecanismos de configuração e meios físicos do BatiBUS e EHS.

A *Konnex Association*, detentora do *standard* KNX, é uma organização internacional sem fins lucrativos sediada em Bruxelas, Bélgica, e representa mais de uma centena de empresas líderes mundiais na área dos sistemas de electrónica para a automação de edificios [5].

3.1.2 Principais características

- Interoperabilidade – Garante que os produtos de diferentes fabricantes utilizados em diversas aplicações irão operar e comunicar uns com os outros. Isto permite um alto grau de flexibilidade na expansão e na modificação de instalações.
- Qualidade do produto – A Associação *Konnex* requer um alto nível de controlo de qualidade durante todas as etapas da vida do produto. Assim, todos os produtos desenvolvidos pelos membros da *Konnex* que ostentam a marca KNX têm de demonstrar compatibilidade com a norma ISO 9001.
- O KNX pode ser utilizado para todas as aplicações no controlo de habitações e edifícios: desde o controlo da iluminação e persianas até vários sistemas de segurança, aquecimento, ventilação, ar condicionado, controlo de água, gestão de energia, monitorização e alarme.
- O KNX está apto para ser utilizado em diferentes tipos de edifícios. Este pode ser utilizado em construções novas e, mesmo, nas já existentes. As instalações KNX podem também ser utilizadas num edifício de qualquer dimensão, desde uma simples habitação até edifícios de vários andares, como hotéis, centros comerciais, blocos de apartamentos, hospitais e escolas.
- O KNX suporta vários tipos de modos de configuração: O S-Mode: O método de configuração mais poderoso, realizado através de um computador ligado ao barramento, o A-Mode: método de configuração automático mas bastante limitado e o E-Mode: método de configuração por um controlador de dispositivo, no barramento de dados.
- O KNX suporta vários meios físicos de comunicação: Par trançado (TP), Rede eléctrica (*Powerline*), Rádio Frequência (RF) e IP/Ethernet
- A expansão, mudanças e actualizações são possíveis, sem que se verifique a necessidade de redesenhar e reconstruir a instalação.
- Ferramenta de configuração independente do fabricante - A ferramenta de *software* ETS (*Engineering Tool Software*), ferramenta de configuração independente do fabricante, permite o planeamento, engenharia e a configuração de todos os produtos KNX certificados. Esta ferramenta única permite ao integrador do sistema combinar produtos de diferentes fabricantes na mesma instalação. As bases de dados de produtos com material certificado de todos os fabricantes KNX podem ser importadas para o ETS.

A norma KNX, devido ao facto de ser uma fusão de três sistemas já existentes anteriormente com as suas respectivas vantagens, não apresenta grandes desvantagens. Podemos, então, destacar como pontos mais desfavoráveis desta tecnologia a relativa complexidade das instalações e custo também relativamente elevado do equipamento certificado.

3.1.3 Topologia da rede

O KNX define-se como uma rede totalmente distribuída, uma vez que não requer um controlador central na instalação e todos os dispositivos que se ligam ao barramento de comunicação de dados têm o seu próprio microprocessador integrado e toda a electrónica de acesso ao meio. Esta topologia distribuída pode acomodar até 65.536 dispositivos, correspondendo a um espaço de endereço individual de 16bits. Numa rede KNX, cada segmento eléctrico pode conter até um máximo de 64 dispositivos ligados, sendo que dois ou mais segmentos podem ser interligados através de um repetidor, formando um segmento lógico designado por linha.

Uma linha pode incluir até 4 segmentos eléctricos interligados entre si por repetidores, resultando numa capacidade máxima de 256 dispositivos. O uso de mais do que um segmento eléctrico só deve ter lugar para aumentar a capacidade de instalações já existentes.

Como é ilustrado na figura 3.1, as várias linhas podem ser interligadas numa linha principal, fazendo uso de acopladores de linha, tendo este agrupamento a designação de área. Todo o domínio de uma rede KNX é formado por um máximo de 15 áreas, que são conectadas entre si através de uma linha de área, sendo a ligação da linha principal à linha de área feita através de um acoplador de área [6].

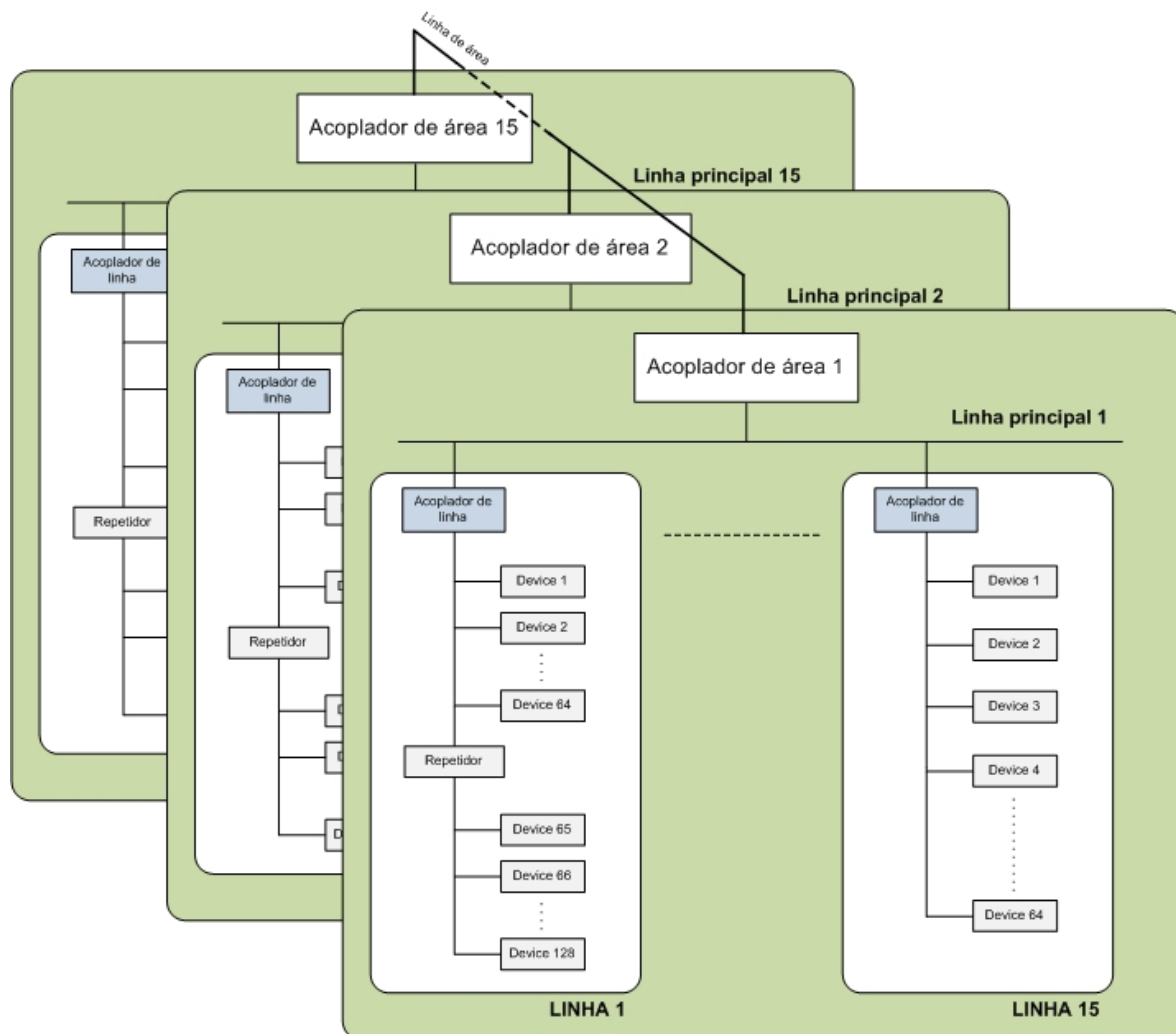


Figura 3.1 – Topologia de uma rede KNX/EIB

3.1.4 Meios de transmissão

A tecnologia KNX/EIB é suportada por diversos tipos de meios de transmissão, como, por exemplo, o par trançado, a rede eléctrica, IP, rádio frequência e infravermelhos. Juntamente com este meios, existe ainda a possibilidade de fazer uma interligação com outro tipo de dispositivos, como é o caso de um modem GSM, para a transmissão de alertas de situações de emergência para o telemóvel de responsável do edifício, seja ele uma habitação familiar ou um edifício comercial. Esta interligação de dois meios diferentes é possibilitada através de uma unidade conversora, o *gateway*.

Diferentes aplicações requerem diferentes soluções. Isto é especialmente verdade nos vários meios físicos de comunicação. Enquanto o par trançado é um sistema com a

máxima fiabilidade e o mais baixo custo (logo aplicado na grande maioria das instalações), a opção de comunicação pela rede da instalação eléctrica é aplicável se não houver outra opção viável. Esta última opção é normalmente utilizada em edifícios onde a instalação de rede KNX/EIB é posterior à construção do edifício, onde nunca foi pensada a possibilidade da automação do edifício em questão.

Se nenhuma rede eléctrica estiver disponível no local da instalação ou se esta for demasiado complexa, a rádio frequência é uma boa solução, embora seja menos acessível financeiramente. A opção IP é inestimável para aplicações onde seja necessária uma grande largura de banda e/ou haja já instalado no edifício um *backbone* de rede IP (como grandes edifícios de escritórios, hotéis, centros comerciais e hospitais). De seguida, apresenta-se um breve resumo das principais características dos vários meios de comunicação.

3.1.4.1 Par entrançado

Existem dois tipos de especificações na norma KNX/EIB para a implementação do meio físico recorrendo ao par entrançado, o TP0 e o TP1.

O modo TP0 foi herdado do sistema BatiBUS, muito popular em França. Os produtos certificados KNX TP0 podem operar no mesmo barramento dos dispositivos BatiBUS, mas não podem trocar informação entre si. Este modo, com uma taxa de transferência 4800 bit/s, está a desaparecer, uma vez que a maioria dos fabricantes de equipamento está a mudar para TP1.

O modo TP1 foi introduzido com o EIB e é usado actualmente por mais de 90% dos produtos KNX. O KNX TP1 combina a transmissão de dados com elevada qualidade com o baixo custo do hardware que o implementa. Devido a estes factos, a grande maioria das instalações utiliza o TP1 como meio físico de comunicação. A topologia do TP1 é bastante flexível: linear, em estrela, em árvore ou como uma combinação entre as três. A taxa de transferência é de 9600 bit/s e os dispositivos ligados a uma rede TP1 podem ser alimentados pelo barramento, se tal for necessário. Os equipamentos certificados como EIB e KNX TP1 podem operar e comunicar entre si no mesmo barramento. Em ambos os casos é implementado um mecanismo que controla o acesso ao canal de comunicação, de modo a detectar colisões, o CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) [5][7][8].

3.1.4.2 Rede eléctrica

Neste meio físico é utilizada a instalação rede eléctrica do edifício como canal de comunicação. Este meio é maioritariamente adoptado em situações onde a instalação da rede de domótica é posterior à da construção do edifício, uma vez que permite a utilização da instalação eléctrica já existente no mesmo. Contudo, este meio físico apresenta algumas limitações (ruído, interferências, atenuação) que impedem que a taxa de transmissão seja elevada.

Tal como acontece com o par entrançado, referido anteriormente, existem duas especificações para este meio físico: o PL110 e o PL132.

A especificação PL110 foi igualmente tomada do EIB. Embora neste momento, apenas alguns fabricantes apoiem a PL110, eles continuam a oferecer uma gama completa de dispositivos para controlo de iluminação, estores e aquecimento.

A informação digital é transmitida na rede através da modulação em frequência SFSK (*Spread Frequency Shift Keying*) com uma taxa de transmissão de 1200 bit/s. Os dispositivos EIB PL110 podem operar e comunicar com dispositivos KNX/EIB PL110.

A especificação PL132 foi herdada do EHS (*European Home System*), e está actualmente a ser utilizada apenas por alguns fabricantes. Esta especificação também utiliza a rede eléctrica para a transferência de dados, mas com uma modulação diferente, a MSK (*Minimum frequency-shift keying*) com uma taxa de transmissão de dados de 2400 bit/s. Uma vez que praticamente não existem produtos para este meio e os dispositivos KNX/EIB PL132 não podem comunicar com dispositivos EHS PL132, a especificação PL132 pode vir a desaparecer no futuro [5][7][8].

3.1.4.3 Wireless

O KNX/EIB permite a utilização do ar como meio de comunicação. Na norma estão definidos dois tipos de redes que utilizam o ar como meio de transmissão dos dados: por rádio frequência e por infravermelhos.

A especificação que define a rádio frequência (KNX RF) como método de comunicação ainda é relativamente recente na especificação KNX. Embora ainda seja apoiada apenas por alguns fabricantes, estão em processo de desenvolvimento e de certificação novos produtos de vários outros fabricantes, que em breve estarão no mercado.

O KNX RF envia as tramas KNX através de sinais rádio na banda de frequência dos 868 MHz e com uma potência na ordem do 25 mW (dispositivos de curto alcance). Com uma taxa de transmissão de 16384 bit/s, pode ser transmitido um número de tramas equivalente

ao KNX TP1. Para a utilização do KNX RF, podem ser utilizados componentes de baixo custo e de baixo consumo eléctrico, que permitam uma implementação de comunicação bidireccional ou unidireccional. Para instalações de tamanho pequeno e médio, geralmente não são necessários repetidores para garantir a qualidade do sinal.

O KNX RF é também muito importante, na medida em que permite a expansão e a renovação de redes de cabo já existentes, em que os custos de refazer ou renovar toda a instalação seriam inoportáveis.

A especificação da utilização dos infravermelhos como meio de comunicação está definida na norma EIB (EIB.IR), e manteve-se na especificação KNX, sem que qualquer alteração fosse efectuada. Neste método de transmissão, o sinal é transmitido por infravermelhos até uma distância de cerca de 12 metros, e a comunicação pode ser unidireccional ou *half-duplex* bidireccional com a transmissão assíncrona de pacotes de dados. O tipo de modulação do sinal transmitido é ASK (*Amplitude-shift keying*) com uma frequência de 447.5 kHz, com uma taxa de transmissão de 1300 bit/s para o modo unidireccional e com uma taxa de transmissão de 7000 bit/s para a transmissão bidireccional.

Neste tipo de sistema, os sinais infravermelhos (IR) são transmitidos por emissores IR ou por controlos remotos ao accionar um interruptor ou um botão, respectivamente. Um receptor IR, quando recebe um sinal, encaminha-o para um decodificador interno, onde os sinais são convertidos em tramas KNX/EIB e enviados para o barramento [5][7][8][9].

3.1.4.4 IP

A especificação do KNX/EIB herda do EIB a especificação EIBnet/IP, que define a integração do protocolo EIB sobre as redes IP (*Internet Protocol*). A norma agora denominada por KNXnet/IP especifica que as tramas KNX/EIB possam ser encapsuladas em pacotes IP. Desta maneira, as redes LAN (*local area network*), bem como a *Internet*, podem ser utilizadas como meio de transmissão para tramas KNX/EIB.

Considerando a massificação das redes IP e infra-estruturas de suporte das mesmas, estas representam um *backbone* interessante e bem adequado para as instalações EIB/KNX. As redes IP oferecem, comparativamente com as redes EIB/KNX, vantagens para aplicações onde seja necessária uma elevada largura de banda e velocidade e/ou haja já instalado no edifício um *backbone* de rede IP (como os grandes edificios de escritórios, hotéis, centros comerciais e hospitais), possibilitando ainda o acesso remoto, o controlo, a supervisão e mesmo a sua configuração remota.

A utilização deste meio físico é normalmente acompanhada com outro meio físico como por exemplo o KNX/EIB TP1. A norma KNXnet/IP define um servidor que funciona como um *gateway* e que interliga a rede KNX/EIB (TP1, normalmente) a uma rede IP [5][7][8].

3.1.5 Modos de configuração

Depois de um dispositivo KNX ser ligado ao meio, ele é configurado para se integrar na instalação KNX. Uma vantagem adicional do KNX é o número de modos de configuração que este protocolo suporta.

A norma KNX permite, também, que cada fabricante possa seleccionar o modo de configuração ideal, de acordo com o mercado alvo ao qual o produto se destina.

3.1.5.1 Modo S

O *System mode* ou S-Mode foi o primeiro modo de configuração a ser estabelecido, e já foi especificado pelo EIB. Este modo necessita do software de configuração oficial da *Konnex Association*, o ETS (*Engineering Tool Software*) para levar a cabo o processo de configuração. Este modo é o mais poderoso e oferece o mais alto grau de flexibilidade para a realização de funções de controlo de edifícios, devendo ser aplicado nas instalações de maior complexidade. Neste modo de configuração, o fabricante deve fornecer uma base de dados do produto com as funções e propriedades de todos os seus dispositivos KNX/EIB, que será carregada pelo ETS. Este mecanismo de configuração destina-se a instaladores especialmente treinados em redes KNX/EIB e com conhecimentos sobre a configuração de funções avançadas de controlo em edifícios [5][7][8].

3.1.5.2 Modo E

O *Easy mode* ou *E-mode* é um novo modo de configuração introduzido com o KNX, que não necessita de um computador com o ETS para a configuração da rede, embora ofereça funções mais limitadas do que o modo de configuração referido anteriormente. Neste modo, as propriedades dos dispositivos ligados na rede podem ser lidas através do barramento, dispensando também a necessidade da base de dados do fabricante com as propriedades do produto. Este método de configuração está dividido em dois diferentes modos, o *easy controller mode* e o *easy push button mode*.

No primeiro modo, é necessário um controlador de dispositivo na rede KNX, que realiza o processo de configuração de acordo com regras de conexão definidas.

No segundo modo, a configuração não requer dispositivos ou ferramentas auxiliares, e pode ser realizada por cada dispositivo, dispondo cada um deles de um botão integrado necessário à sua própria configuração. Este método de configuração é destinado aos instaladores com treino KNX/EIB básico, e os dispositivos estão já pré-programados e carregados por omissão com um conjunto de parâmetros de funcionamento [5][7][8].

3.1.5.3 Modo A

Existe, ainda, um terceiro modo, denominado *automatic mode*, que foi recentemente especificado na norma KNX. Este modo oferece um processo de configuração *plug-and-play* sem qualquer interacção por parte do utilizador, e destina-se normalmente aos utilizadores menos experientes. Como este método apresenta funcionalidades muito limitadas, a oferta no mercado dos dispositivos que suportam este modo é reduzida [5][7][8].

3.1.6 Modo de endereçamento de dispositivos

A gestão dos dispositivos conectados num barramento de rede KNX/EIB pode ser efectuada de 2 modos: o endereçamento físico e o endereço de grupo.

Cada dispositivo ligado ao barramento é identificado por um endereço físico único (dois ou mais dispositivos não podem partilhar o mesmo endereço).

O endereço físico de 16 bit é constituído por um número de área (4bits), de linha (4bits) e de dispositivo no barramento (8bits), que descreve a sua localização e define univocamente cada dispositivo na rede. Este tipo de endereçamento é apenas usado como endereço de destino para a inicialização, para a programação e para as operações de diagnóstico.

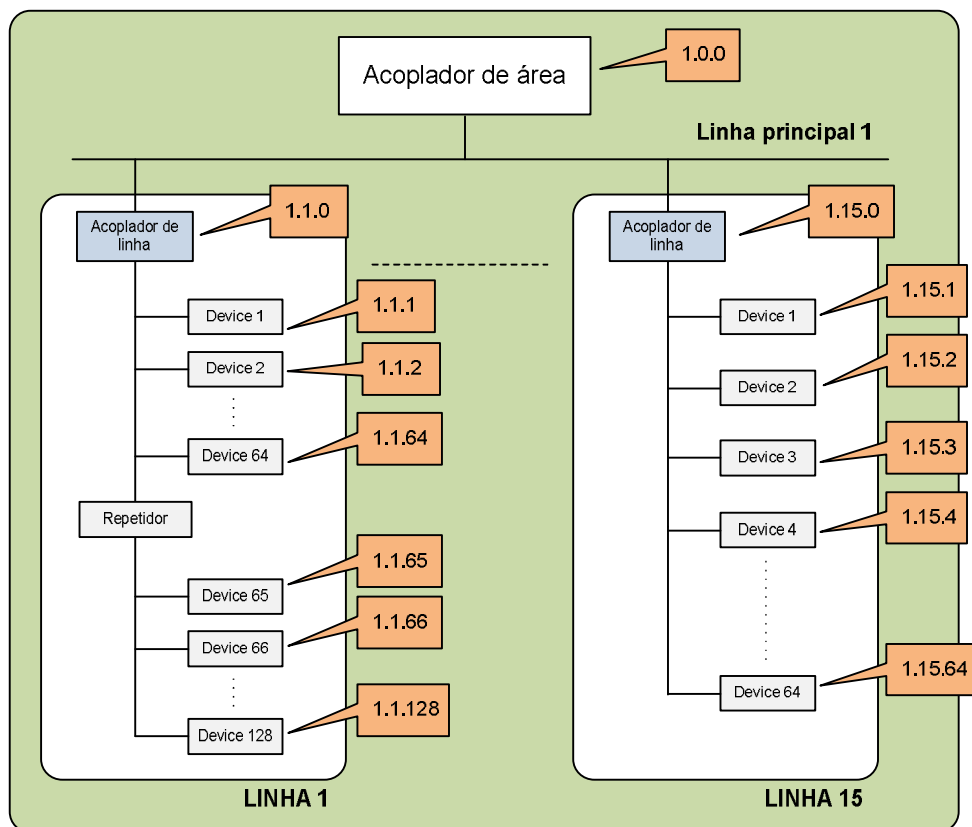


Figura 3.2 – Topologia dos endereços físicos

Para além do endereço físico, cada dispositivo pode ter um ou mais endereços lógicos, denominados de endereços de grupo.

Os endereços de grupo estabelecem quais os dispositivos ligados ao barramento que vão actuar em sintonia entre si, isto é, que sensor controla cada actuador ou conjunto de actuadores.

O modo de endereçamento de grupo corresponde ao modo de funcionamento normal da rede. Este modo associa funcionalmente os dispositivos no barramento, uma vez que todos os dispositivos que tenham o mesmo endereço de grupo recebem as mesmas mensagens.

Os elementos transmissores de uma instalação KNX/EIB, os sensores, apenas podem transmitir telegramas para um endereço de grupo distinto, ao contrário dos elementos actuadores, que podem ter vários endereços de grupo, o que lhes permite reagir a diversos sensores.

O endereçamento de grupo oferece bastante flexibilidade, uma vez que permite adicionar um dispositivo ao barramento de uma maneira muito simples, bastando ligá-lo ao endereço de grupo correcto.

Os endereços de grupo podem ter duas formas, de acordo com as necessidades na hierarquização das funções do sistema, endereço de nível 2 e endereço de nível 3. Os endereços de nível 2 dividem-se em dois campos: o grupo principal e subgrupo, enquanto os de nível 3 se dividem em três campos: grupo principal, grupo intermédio e subgrupo, como se pode observar na figura 3.3 [7][8][10][11].

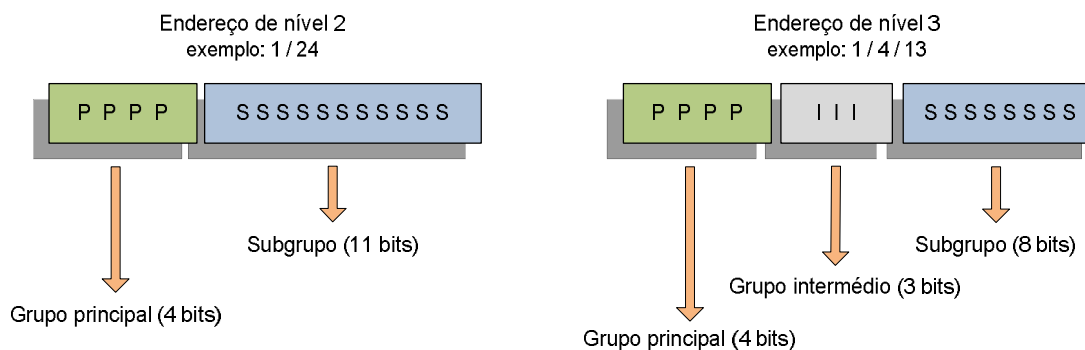


Figura 3.3 – Níveis dos endereços de grupo

3.1.7 Comunicação entre dispositivos

Para que dois dispositivos possam comunicar entre si, estes devem partilhar um dialecto comum - de modo a que os dados trocados entre ambos tenham o mesmo significado para os dois dispositivos - e garantir a máxima compatibilidade entre dispositivos de diferentes

fabricantes. Para isso, os dispositivos KNX/EIB estão ligados logicamente entre si por meio de objectos de comunicação. Estes objectos contêm informações importantes sobre o estado do dispositivo, por exemplo, se uma lâmpada está ligada ou desligada, a temperatura de um sensor ou se foi pressionado um interruptor. Cada dispositivo pode ter um ou mais objectos de comunicação.

Assim, de modo a normalizar a comunicação entre os diversos dispositivos, estabeleceu-se o EIS (EIB Interworking Standard), que contém 15 tipos de dados (formato, estrutura) úteis para cada função atribuída aos objectos de comunicação para controlar os diversos dispositivos.

A seguinte tabela resume os diferentes tipos de dados EIS que estão disponíveis.

Tabela 3.1 – Tipos de dados EIS

TIPO EIS	Função EIB	Dimensão	Descrição
EIS 1	SWITCHING	1bit	Acendido / Apagado, ON/OFF, TRUE/FALSE
EIS 2	DIMMING	4bit	Aumenta (+)/ Diminui(-), Valor
EIS 3	TIME	3bytes	Dia da semana, hora, minutos e segundos
EIS 4	DATE	3bytes	Dia / mês / ano
EIS 5	VALUE	2bytes	Para enviar valores físicos como Temperatura, Intensidade luminosa, Tensão, Corrente
EIS 6	SCALING	8bit	Utiliza-se para transmitir valores relativos com uma resolução de 8 bits (Humidade Relativa=100%)
EIS 7	CONTROL DRIVE	1bit	Mover baixo/cima, Abrir/Fechar
EIS 8	PRIORITY	1bit	Activado / Desactivado
EIS 9	FLOAT VALUE	4bytes	Codificação de um número em vírgula flutuante
EIS 10	16BIT COUNTER	2bytes	Representa os valores de um contador de 16 bits
EIS 11	32BIT COUNTER	4bytes	Representa os valores de um contador de 32 bits
EIS 12	ACCESS	4bytes	Código de acesso de segurança
EIS 13	CHARACTER ASCII	8bit	Codificação de caracteres ASCII
EIS 14	8BIT COUNTER	8bit	Representa os valores de um contador de 8 bits
EIS 15	CHARACTER STRING	14bytes	Transmite uma string até 14 bytes

Normalmente, os dispositivos podem dispor de vários objectos de comunicação, que podem ser de diversos tipos EIS. Por exemplo, um interruptor de duas teclas para controlo de iluminação pode dispor de objectos do tipo EIS 1, para as funções de comutação (acender/apagar) e de objectos de regulação, do tipo EIS 2, para o envio de ordens de incremento/decremento de intensidade luminosa.

Cada objecto de comunicação tem um endereço de grupo associado, que é único se se tratar de um objecto de comunicação emissor (elemento sensor), ou que podem ser vários, se se tratar de um objecto de comunicação receptor (elemento actuador).

Um objecto de comunicação emissor e um receptor ligam-se entre si, associando o mesmo endereço de grupo, desde que ambos os objectos sejam do mesmo tipo.

Quando o valor no elemento sensor se altera, este transmite o novo valor para o endereço de grupo associado. Todos os objectos de comunicação receptores que tenham o mesmo endereço de grupo reagem a esta alteração e actuam de acordo com a sua função. Por exemplo, supondo que o objecto de comutação de um interruptor está ligado ao endereço de grupo 1/2/3, o objecto de comutação de um actuador responsável por ligar uma lâmpada, também deve estar ligado ao endereço de grupo 1/2/3 [7][8][10][11][12].

3.2 Tecnologias Web

3.2.1 JavaScript

O enorme crescimento da *World Wide Web* levou a uma grande procura de páginas *Web* dinâmicas e interactivas, capazes de oferecer ao utilizador uma boa experiência de navegação. Para atrair a atenção de possíveis clientes, as empresas apostam cada vez mais na *Internet*, como forma de mostrarem ao mundo os seus produtos e/ou os seus serviços. Como forma de tornar isto possível, é importante que este meio, a página *Web*, seja o mais agradável e intuitiva possível, possibilitando um nível de interactividade elevado e capaz de cativar todo o possível cliente.

O HTML, em si, é estático, e apenas permite uma mínima interacção com os utilizadores. Como consequência disto, as páginas *Web* que se desejem verdadeiramente interactivas e funcionais não podem utilizar somente o HTML. Face à necessidade de uma nova abordagem para introduzir alguma dinâmica num conteúdo que era estático até à data, surgiu o JavaScript.

O JavaScript é uma linguagem de programação interpretada, desenvolvida, inicialmente, pela Netscape em 1995, e é, na maioria das vezes, usada para o desenvolvimento de aplicações *Web* que se executam do lado do cliente.

Isto significa que qualquer código escrito em JavaScript é entregue juntamente com a página HTML, e é executado dentro do browser do cliente, em vez de correr directamente no servidor que está a disponibilizar essa mesma página.

O JavaScript, apesar do nome, não está relacionado com o Java tradicional, apenas partilha alguma semelhança na sua sintaxe, sendo totalmente diferente no conceito e no uso.

Uma das mais frequentes utilizações do JavaScript é a validação dos dados introduzidos por utilizadores num formulário. Isto significa que os dados que foram inseridos são verificados, com o intuito de concluir se são os mais adequados para o campo de preenchimento em questão. No caso de os dados não estarem correctamente inseridos, o preenchimento do formulário não deverá prosseguir até que sejam correctamente inseridos. Esta acção é de grande importância, principalmente em servidores que sirvam centenas ou mesmo milhares de páginas por minuto, uma vez que não é necessário um poder de processamento adicional por parte destes para a validação dos dados do formulário e reenvio da página com uma notificação de erro ao utilizador.

A verificação dos dados por parte do servidor, apesar de esporadicamente necessária, (para uma consulta à base de dados, por exemplo) usa largura de banda útil que, numa altura de grande carga, pode mesmo diminuir o número de utilizadores ligados a esse mesmo servidor.

Como os formulários incorrectamente preenchidos não são submetidos para o servidor, uma vez que são validados na máquina cliente, o utilizador não tem de estar à espera que os dados sejam enviados, validados e reenviados de volta com possíveis mensagens de erro, contribuindo assim para uma experiência de navegação mais agradável e eficaz para o utilizador final.

Porém existem alguns inconvenientes no seu uso. O código JavaScript pode facilmente adicionar dezenas ou mesmo centenas de linhas de código a uma página *Web*, tornando a sua manutenção extremamente complicada e de difícil compreensão.

Esta desvantagem pode ser facilmente ultrapassada, guardando as várias secções de código JavaScript num ficheiro com a extensão `.js` e incluindo-o no cabeçalho de todas as páginas HTML que necessitem deste código. Fica assim, desta maneira, separado o código JavaScript do conteúdo HTML da página propriamente dita. [13][14]

3.2.2 AJAX

À medida que a *World Wide Web* se vai expandindo e a largura de banda das ligações vai aumentando, as pessoas procuram cada vez mais e melhor conteúdo, aplicações mais inteligentes e com respostas mais rápidas.

Até muito recentemente, à medida que a complexidade das aplicações ia crescendo, os utilizadores começaram a deparar-se com sistemas lentos e com dificuldade de resposta, onde páginas inteiras tinham de ser recarregadas apenas para se actualizar um único elemento. A plataforma *Web* mostrava-se, então, frágil e pouco eficaz para as aplicações mais exigentes.

De modo a ultrapassar estas dificuldades e possibilitar o desenvolvimento de páginas com melhor tempo de resposta, interactivas e dinâmicas, com uma maior eficácia, adoptou-se uma nova metodologia, o Ajax.

Tornando-se popular em 2005 com o Google Suggest, o Ajax ou “*Asynchronous JavaScript and XML*” não é mais do que uma combinação de várias tecnologias *standard* já existentes como o JavaScript, o DHTML/HTML e o XML.

O objecto *XMLHttpRequest* é o componente fundamental do Ajax, e o que o torna tão útil. Ele proporciona um mecanismo que permite ao cliente, através do JavaScript, trocar informação directamente com o servidor *Web*. A informação recebida pode ser processada em *background* e usada para, dinamicamente, actualizar elementos numa página *Web*, sem a necessidade de recarregar toda a página. Desta forma, a informação circula em *background* e as páginas são actualizadas como se se tratassem de aplicações *standalone* típicas, ao invés das aplicações *Web* tradicionais, disponíveis até à altura.

A parte assíncrona do termo Ajax (figura 3.4) significa que o browser não vai esperar pela informação que vai ser devolvida pelo servidor, mas vai processá-la apenas quando esta for enviada pelo servidor. Esta é a parte crucial do Ajax, na qual a aplicação *Web* não fica num estado de espera, aguardando o retorno da informação do servidor. Se a aplicação parar à espera da informação (aplicações tradicionais), essa aplicação será síncrona (figura 3.5), e com ligações à *Internet* lentas, isso pode representar um problema [15][16].

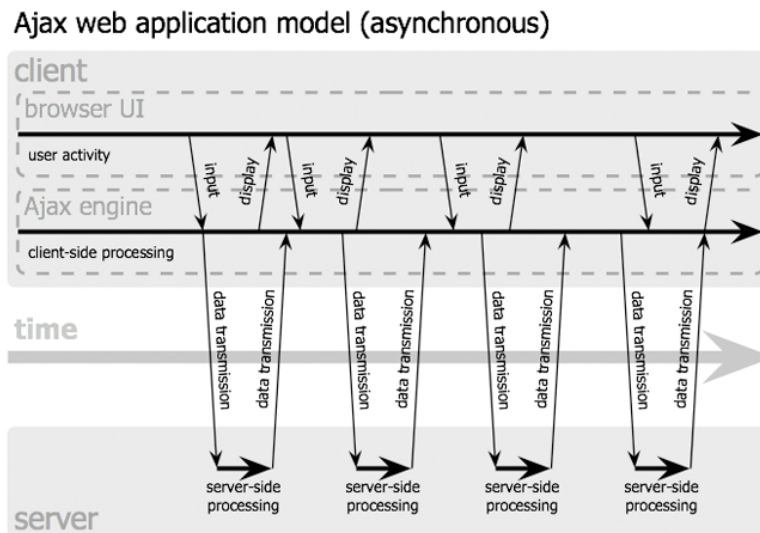


Figura 3.4 – Interação assíncrona das aplicações Ajax [16]

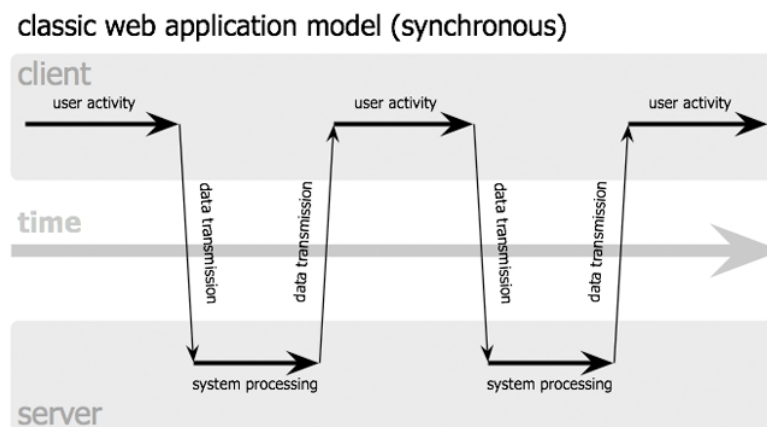


Figura 3.5 – Interação síncrona das aplicações *Web* tradicionais [16]

3.2.3 jQuery

jQuery é uma biblioteca JavaScript poderosa e extremamente leve, que permite aos programadores e aos *Web designers* o adicionar de elementos dinâmicos e interactivos às suas páginas, atenuar as inconsistências dos vários *browsers* e reduzindo fortemente o tempo de desenvolvimento das aplicações, promovendo a produtividade.

Criado por *John Resig*, o jQuery é um projecto *open-source* que promove uma maneira diferente de escrever código em Javascript e que proporciona um instrumento essencial para fornecer uma compatibilidade robusta multi-plataforma e simplificar a maneira de como se percorrem os documentos HTML, a manipulação de eventos, o executar de animações e o adicionar de interações Ajax numa página *Web* [17].

3.2.4 CGI

Common Gateway Interface, ou CGI, é um conjunto de regras (*standard*) que descreve a forma como um servidor *Web* comunica com outras aplicações que correm na mesma máquina, e como estas comunicam com o servidor *Web*. Sem este *standard* bem definido e suportado por parte dos servidores, seria impossível aumentar as funcionalidades destes e gerar conteúdo dinâmico sem recorrer a metodologias proprietárias do servidor ou modificar o código do servidor. Esta última tarefa requer que o código fonte do servidor esteja disponível (o que nem sempre é possível); requer, ainda, um extenso conhecimento técnico em redes/programação para a implementação das novas funcionalidades e, por fim, se houver a necessidade de mudar estas implementações para uma nova plataforma, será necessário refazer grande parte do código ou despende bastante tempo a fazer o *port* do código para a nova plataforma.

O CGI proporciona uma solução simples e robusta para estes problemas. Com a especificação deste protocolo, e através de servidores *Web* que o implementem, qualquer aplicação desenvolvida numa qualquer linguagem de programação pode comunicar directamente com qualquer servidor *Web*, uma vez que esta comunicação é tratada através do *standard input* e do *standard output*. Esta simplificação, significa que qualquer programador que saiba ler e imprimir dados para o *standard input/output*, respectivamente, usando uma linguagem de programação, é capaz de implementar uma aplicação CGI no servidor *Web* sem despende muito esforço extra.

De todo um conjunto de linguagens, há duas que são maioritariamente utilizadas nas aplicações CGI, o C e Perl. Estas duas linguagens inserem-se em duas categorias diferentes: as linguagens compiladas e as linguagens interpretadas, respectivamente. Cada um destes dois tipos tem as suas vantagens e as suas desvantagens. Nas linguagens compiladas, como o C/C++, o código fonte do programa é traduzido pelo compilador para um código máquina capaz de ser interpretado pelo CPU. Deste processo resulta um código muito eficiente, que pode ser executado todas as vezes que forem necessárias sem haver necessidade de repetir, novamente, o processo de compilação. Depois da execução deste processo, que apenas ocorre uma vez, o código gerado apenas precisa de ser carregado e executado, sem nenhum *overhead* adicional.

Contudo, existem algumas desvantagens, como uma maior dificuldade na programação, na depuração e na actualização das aplicações, traduzindo-se num maior tempo despendido no desenvolvimento destas.

No caso das linguagens interpretadas como o Perl ou o PHP, o código fonte tem de ser analisado, interpretado e executado de cada vez que os programas correm, ao invés de uma única compilação inicial, acrescentando um maior *overhead* à execução destes. Por este

motivo, os programas interpretados são geralmente menos eficientes do que os programas compilados. A grande vantagem das linguagens interpretadas é que estas permitem geralmente um desenvolvimento mais rápido, uma maior facilidade na actualização e na depuração das aplicações. Normalmente, a decisão de se usar uma linguagem deste tipo é baseada na facilidade de futuras alterações, ou quando há restrições ao nível do tempo disponível para o desenvolvimento da aplicação.

Normalmente, uma aplicação CGI é um pequeno programa que é executado em tempo real no servidor, podendo este produzir conteúdo dinâmico com o intuito de o incorporar numa página *Web* estática.

Um exemplo muito frequente da utilização desta tecnologia é na gestão de base de dados, em que a aplicação CGI pode fazer consultas, inserir, remover ou alterar a informação com base nos dados introduzidos num *Web browser* cliente. O *browser* envia os dados introduzidos no formulário HTML para o servidor *Web*, submetendo esta informação para o programa CGI. O programa CGI é executado nesse instante, processa os dados recebidos e retorna o resultado deste processamento de volta para o servidor *Web*, que, por sua vez reencaminha esta informação para o *browser*. Esta sequência pode ser observada na seguinte figura [18][19].

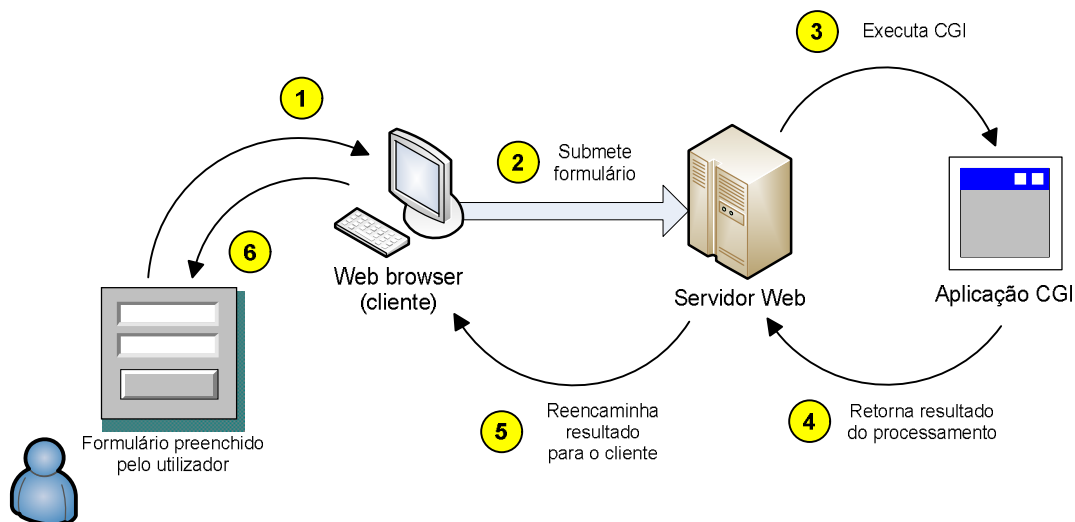


Figura 3.6 – Diagrama de sequência CGI simples

3.2.5 JAVA

Foi a meio da década de 90 que a *Internet* viu o seu crescimento aumentar exponencialmente em número de servidores disponíveis e páginas alojadas, bem como o aparecimento das primeiras aplicações comerciais na *Web*.

Estas primeiras aplicações disponíveis recorriam a simples formulários e a programas CGI (como foi elucidado na secção CGI) como primeiro passo para trazer a interactividade e o dinamismo necessário à execução deste tipo de aplicações. No entanto, a utilização de formulários não trouxe, por si só, uma verdadeira interactividade às páginas *Web*: a comunicação (pedido/resposta HTTP) entre o cliente e o servidor, onde é efectuada a execução e o processamento, faz com que não existia uma resposta directa e verdadeiramente em tempo real às acções do utilizador.

O aparecimento da tecnologia JAVA veio imprimir profundas mudanças à *World Wide Web* tradicional e mudar de forma decisiva este cenário. Para esta mudança contribuiu a introdução de pequenos programas que corriam do lado cliente (*browser*): os *applets* (abordados, posteriormente, nesta secção), que podiam ser embutidos directamente no código HTML das páginas *Web* para realizar alguns tipos de tarefas como animações ou algum tipo de processamento local de informação.

O JAVA é uma linguagem de programação orientada a objectos com uma sintaxe semelhante ao C/C++, desenvolvida pela *Sun Microsystems* e que surgiu na altura da massificação da *World Wide Web*, em meados da década de 90.

Esta linguagem difere das linguagens mais tradicionais, como o C ou C++, em que o código fonte é compilado directamente para código máquina.

Em JAVA é seguida uma filosofia diferente. Como existe código transferido pela *Internet* que será executado localmente numa grande variedade de plataformas clientes, o JAVA foi projectado para ser multi-plataforma, tanto no código fonte do programa como no ficheiro binário, depois de compilado.

Uma vez que não é possível correr o mesmo ficheiro binário (por exemplo, em plataformas Windows, Linux ou MAC OS), o JAVA compila o seu código fonte para um ficheiro intermédio independente da plataforma, o *bytecode*. Este ficheiro será interpretado aquando da sua execução por uma máquina virtual. Esta máquina virtual (*JAVA Virtual Machine* ou JVM) é o núcleo onde se encontra o motor responsável por executar o *bytecode* do programa JAVA e é específica para cada plataforma. Desta maneira e, apenas trocando a JVM para o sistema operativo e arquitectura de hardware em causa, será possível correr a mesma aplicação sem necessidade de a recompilar numa grande variedade de plataformas, “*write once, run anywhere*”. Na figura 3.7 podemos observar uma sequência simples da criação de um programa JAVA.

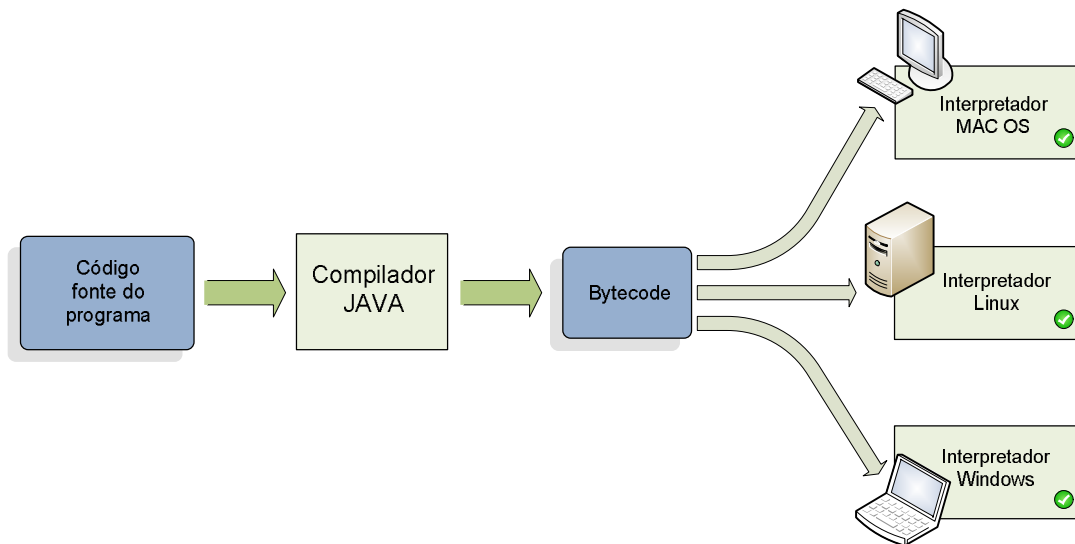


Figura 3.7 – Sequência de um programa JAVA simples

Em JAVA há uma distinção entre alguns tipos de aplicações, sendo os de maior importância os programas *standalone*, os *applets* e os *servlets*.

Os programas *standalone* são os programas típicos de uso geral, que desempenham as mesmas funções que os programas escritos numa qualquer outra linguagem de programação.

Um *applet* é um programa JAVA compilado, que se integra numa página *Web*. Ao ser realizado o acesso com um *browser* àquela página, o *bytecode* é carregado através da *Web*, sendo, de seguida, executado na máquina local onde reside o *browser* cliente.

Os *applets* são usados, normalmente, para fornecer interactividade a aplicações *Web* onde o simples HTML não é capaz de o fazer por si só. Como já foi referido anteriormente, quem vai interpretar e executar a aplicação é a JVM presente na máquina cliente que carregou a aplicação. Deste modo, a independência da plataforma está garantida, podendo funcionar em ambientes Windows, Linux, Mac OS entre outros.

Um *servlet* JAVA pode ser encarado como um *applet*, mas a correr do lado do servidor *Web*. À semelhança de outras tecnologias que executam do lado do servidor, como o CGI ou o PHP, este programa JAVA manuseia dinamicamente pedidos e respostas tipicamente em HTML com um cliente, permitindo, assim, adicionar conteúdos dinâmicos a uma aplicação *Web*.

Um *servlet* JAVA requer um servidor de aplicação especial que proporciona o ambiente para este executar em cooperação com o servidor *Web*. Um exemplo deste servidor de aplicação, e que implementa as especificações definidas pela *Sun Microsystems*, é o *Apache Tomcat* desenvolvido pela *Apache Software Foundation* [20][21].

3.2.6 PHP

Como já foi referenciado anteriormente, neste capítulo, foi a meio da década de 90 que se deu a grande expansão da *World Wide Web*.

Até ao aparecimento do PHP, a solução para criar páginas *Web* dinâmicas passava maioritariamente pelo recurso às aplicações CGI, também já atrás descritas neste capítulo. Isto significava escrever bastante código em C e compilar cada vez que se fazia uma alteração à aplicação, o que, dependendo da dimensão e complexidade da aplicação, poderia tornar-se uma tarefa bastante penosa em termos de tempo de desenvolvimento despendido.

É segundo este contexto que surge o PHP, um acrónimo recursivo para “*PHP: Hypertext Preprocessor*”. Esta nova tecnologia é uma linguagem de programação interpretada, orientada a objectos, livre e utilizada para gerar conteúdo dinâmico em páginas *Web* de uma maneira relativamente rápida. O código PHP tem uma sintaxe semelhante à do C/C++ e do JAVA, e pode fazer tudo o que uma aplicação CGI era capaz de fazer até agora, como recolher dados de formulários e interagir com bases de dados. Os blocos de código PHP são embutidos directamente no código HTML e são executados no servidor quando este recebe o pedido de um cliente para carregar a página.

Quando alguém visita uma página concebida com esta tecnologia, o servidor *Web* processa, inicialmente, o código PHP; depois, analisa quais as partes que deve mostrar aos visitantes (conteúdos, imagens), ocultando os blocos do código PHP depois de interpretados e executados. São nestes blocos executadas as operações como, por exemplo, o acesso a bases de dados, cálculos numéricos e operações sobre ficheiros. No fim desta sequência, no servidor, o código resultante é traduzido para um ficheiro HTML que é a página que vai ser enviada para o *browser* cliente.

O PHP pode ser utilizado numa grande variedade de sistemas operativos e é compatível com a maior parte dos servidores *Web* existentes nas suas variantes livres ou comerciais.

O PHP é considerado uma das linguagens mais fáceis de aprender e de utilizar para o desenvolvimento de aplicações *Web* dinâmicas. A simplicidade e facilidade de desenvolvimento do PHP, juntamente com uma grande comunidade e repositórios de bibliotecas PHP de código aberto, fazem desta linguagem a favorita de programadores e de *Web designers* a nível mundial [22][23].

3.3 Sistemas Embebidos

Um sistema embebido é um sistema computacional que é parte integrante de um dispositivo destinado à execução de aplicações específicas. Devido às suas funções limitadas, estes dispositivos podem ser otimizados para as suas necessidades particulares. O termo embebido significa que o sistema se encontra dentro de um outro sistema e faz parte deste, como por exemplo uma televisão ou um automóvel.

Tradicionalmente, a maior parte deste tipo de dispositivos era utilizado para controlo de processos, medidas e aquisição de dados. Contudo, devido ao enorme desenvolvimento da indústria da microelectrónica, é hoje possível atingir um nível de integração muito elevado ao nível da construção de circuitos integrados. Além da integração dos mais diversos tipos de periféricos (como, por exemplo, interfaces para diversos barramentos de comunicações, processadores dedicados ao processamento de *streams* de vídeo e processadores áudio multicanal), há ainda um aumento capacidade de poder de processamento com um reduzido consumo eléctrico e um baixo custo.

O primeiro sistema embebido moderno reconhecido como tal foi o sistema de navegação e orientação da Missão *Apollo*, sendo a aplicação da primeira produção em massa o sistema de orientação dos mísseis balísticos intercontinentais *Minuteman I* e *II* da *Boeing*, em 1960. Devido à produção em larga escala dos circuitos integrados usados neste sistema, os preços caíram de 100\$ até aos 3\$ cada, levando, na década de 80, a uma massificação do uso deste tipo de sistemas nas mais diversas aplicações [24].

3.3.1 Sistemas Operativos para sistemas embebidos

Os sistemas operativos destinados a este tipo de dispositivos podem ser classificados em duas categorias distintas: os sistemas operativos de tempo-real e os sistemas operativos tradicionais sem requisitos de tempo real.

Os sistemas operativos de tempo real (RTOS) são sistemas operativos que garantem respostas a cada evento, dentro de um intervalo de tempo definido. Este tipo de sistemas é usado principalmente em sistemas críticos que necessitam de ter um comportamento determinístico, isto é, o prazo de execução de uma tarefa não pode ser violado. Dentro deste tipo de sistemas operativos, os que mais se destacam são o *VxWorks* e o *RTlinux*, e têm aplicações em diversas áreas como, por exemplo, os robots industriais, a indústria automóvel, espacial, aeronáutica e o controlo industrial.

Os sistemas operativos tradicionais sem requisitos de tempo real são sistemas operativos geralmente de uma utilização mais generalizada em que estes, ao contrário dos RTOS, não garantem tempos de resposta bem definidos aos eventos recebidos. São exemplos destes sistemas operativos diversas distribuições de Linux destinadas ao mercado dos computadores embebidos.

3.3.1.1 Linux embebido

O poder, a fiabilidade, a flexibilidade e escalabilidade do Linux, combinados com o seu suporte a uma imensidão de arquitecturas de microprocessadores (e respectivo *hardware* circundante) e protocolos de comunicação, estabeleceram o Linux como uma plataforma de software cada vez mais popular para um vasto leque de projectos e de produtos.

Porque o código fonte do Linux está aberto e livremente disponível, muitas variações e configurações do sistema operativo e seu respectivo software de suporte têm evoluído no sentido de dar resposta às diversas necessidades dos mercados e aplicações para os quais o Linux está sendo adaptado. Existem versões adaptadas para dispositivos de muito baixa capacidade, bem como versões especiais para aplicações de tempo real. Apesar das origens do Linux como sistema operativo para a arquitectura dos computadores pessoais (*IBM PC compatible*), existem diversos *ports* para um sem número de arquitecturas que não x86, incluindo ARM, MIPS, PowerPC, SPARC e mesmo alguns microcontroladores.

O uso do Linux abrange todo o espectro de aplicações computacionais, desde o computador mais pequeno do mundo, o *Picotux* (figura 3.8) - que é apenas ligeiramente maior do que um conector de rede RJ45 -, passando pelos dispositivos de comunicações portáteis como os telemóveis e *smartphones*, pelos sistemas de entretenimento (como leitores/gravadores DVD de sala e a *Set-top Box* da televisão por cabo), passando também por praticamente todos os equipamentos de suporte de redes de dados (como os *routers*, *firewalls*, pontos de acesso *wireless*), marcando presença, também, em sistemas de controlo de automóveis, na indústria aeroespacial, na área da automação industrial, na instrumentação médica e, por fim, terminando nos supercomputadores mais avançados do mundo [24][25].



Figura 3.8 – Picotux [26]

3.3.2 Cross-Compiling

Ao contrário dos computadores pessoais, os sistemas embebidos não são, habitualmente, programados na plataforma onde a aplicação a desenvolver será executada. De modo a efectuar a compilação de um programa tendo como destino uma outra arquitectura computacional, são necessários *toolchains* (conjunto de ferramentas de *software* que são usadas para construir aplicações) especiais com o *Cross-compiler* específico para essa mesma arquitectura. O termo *Cross-compiler* representa um compilador capaz de gerar um código executável para uma plataforma diferente daquela em qual o compilador corre.

A sua razão de existência deve-se ao facto de os sistemas embebidos - plataforma alvo destas compilações - disporem de recursos de *hardware* bastante limitados, como a memória não volátil para o armazenamento das aplicações (e sistema operativo caso esteja disponível), a memória RAM e, também, o processador (que não seria capaz de compilar as aplicações mais complexas com a celeridade desejada).

Dependendo das necessidades das aplicações, existem abordagens diferentes para implementar o *software*. Uma possibilidade é a de controlar directamente o *hardware* nas aplicações. Esta é a abordagem com o mais alto desempenho, mas exige mais conhecimento sobre a arquitectura e periféricos utilizados, relativamente à utilização de um sistema operativo. Por outro lado, a existência de um sistema operativo proporciona funcionalidades para multitarefas, alocação de memória e permite que o programador desenvolva as suas aplicações com independência da arquitectura de *hardware* subjacente [27].

Capítulo 4

4. Arquitectura Funcional

Como resultado do estudo anterior, foram seleccionadas um conjunto de tecnologias disponíveis, que permitiram obter diferentes abordagens para a concepção de uma arquitectura funcional capaz de cumprir os objectivos inicialmente propostos.

Neste capítulo serão abordadas as várias soluções encontradas com base no estudo efectuado.

4.1 Primeira abordagem

Nesta primeira abordagem, a implementação do sistema seria efectuada com base num conjunto de APIs JAVA, denominado de Calimero. Estas APIs, desenvolvidas *na Vienna University of Technology*, permitem o estabelecimento de um túnel de comunicação sobre uma rede IP com um *gateway* KNX, processo denominado *KNXnet/IP Tunnelling*. O *gateway* é um dispositivo que permite fazer o interface entre uma rede IP e uma sub-rede KNX/EIB - TP1. Assim sendo, não é necessário um conhecimento detalhado sobre o protocolo KNXnet/IP, permitindo uma abstracção para as camadas de aplicação superiores a esta API.

Como esta API já se encontra desenvolvida em JAVA, a aplicação da camada superior que a vai utilizar terá de ser, também, desenvolvida nesta linguagem de programação.

4.1.1 Primeira alternativa

Nesta primeira alternativa, dispomos de um servidor *Apache Tomcat*, que executa duas funções distintas.

A primeira função executada é a de servidor *Web*, responsável por carregar para os clientes a página HTML com o *applet* JAVA, com todas as informações e controlos necessários para fazer a interface com a rede de domótica KNX/EIB.

A segunda função do servidor é correr um *servlet* JAVA que utiliza a API Calimero, já atrás mencionada. Este irá ser o responsável por toda a gestão da rede de domótica e da comunicação com o *applet* que é executado no browser cliente.

Do lado do cliente o *applet* JAVA recebido é executado. Este interage com o *servlet* que está a correr no servidor via pedidos e respostas HTTP, enviando-lhe as acções feitas pelo utilizador do cliente.

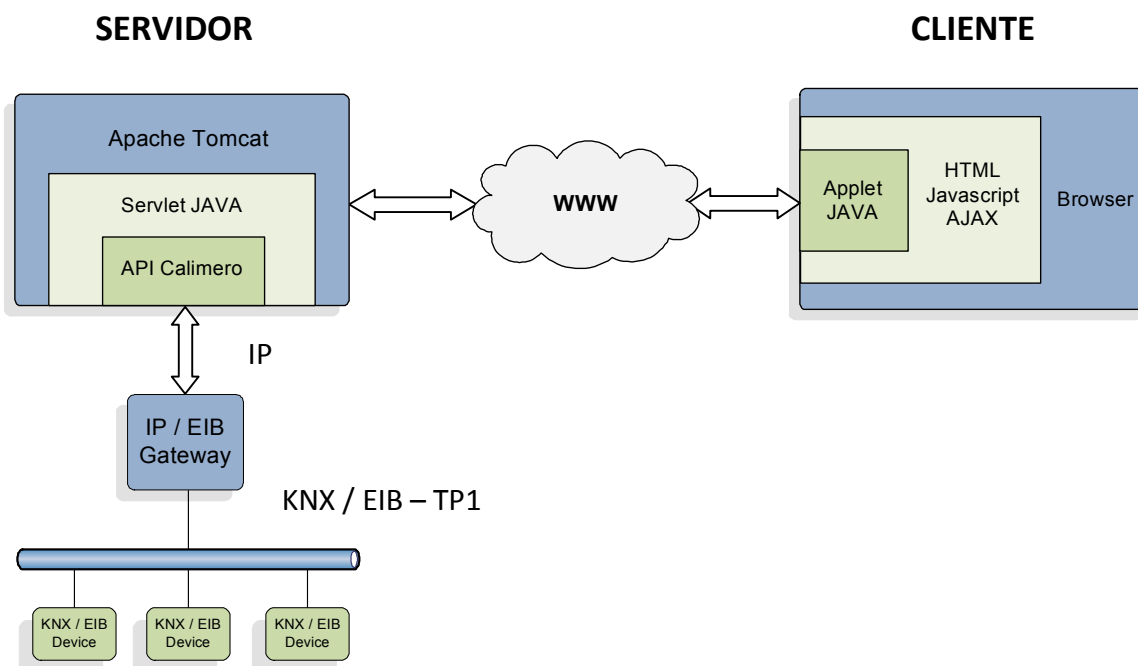


Figura 4.1 – Arquitectura proposta na 1ª alternativa da 1ª abordagem

4.1.2 Segunda alternativa

A segunda alternativa é uma pequena variação da referida anteriormente.

Nesta alternativa, continua a ser carregado para o *browser* do cliente um *applet* JAVA que contém a API responsável pela comunicação com a instalação de domótica através de um túnel de comunicação sobre a rede IP, o Calimero.

Agora, em vez de ser um *servlet* JAVA a executar do lado do servidor para controlar e gerir a instalação de domótica, será o *applet* JAVA a comunicar directamente com o *gateway* IP/KNX da instalação.

Nesta alternativa, terão de ser analisadas as limitações impostas por detrás do conceito de *applet* JAVA, e se estão, ou não, em condições de impossibilitar a satisfação dos objectivos impostos.

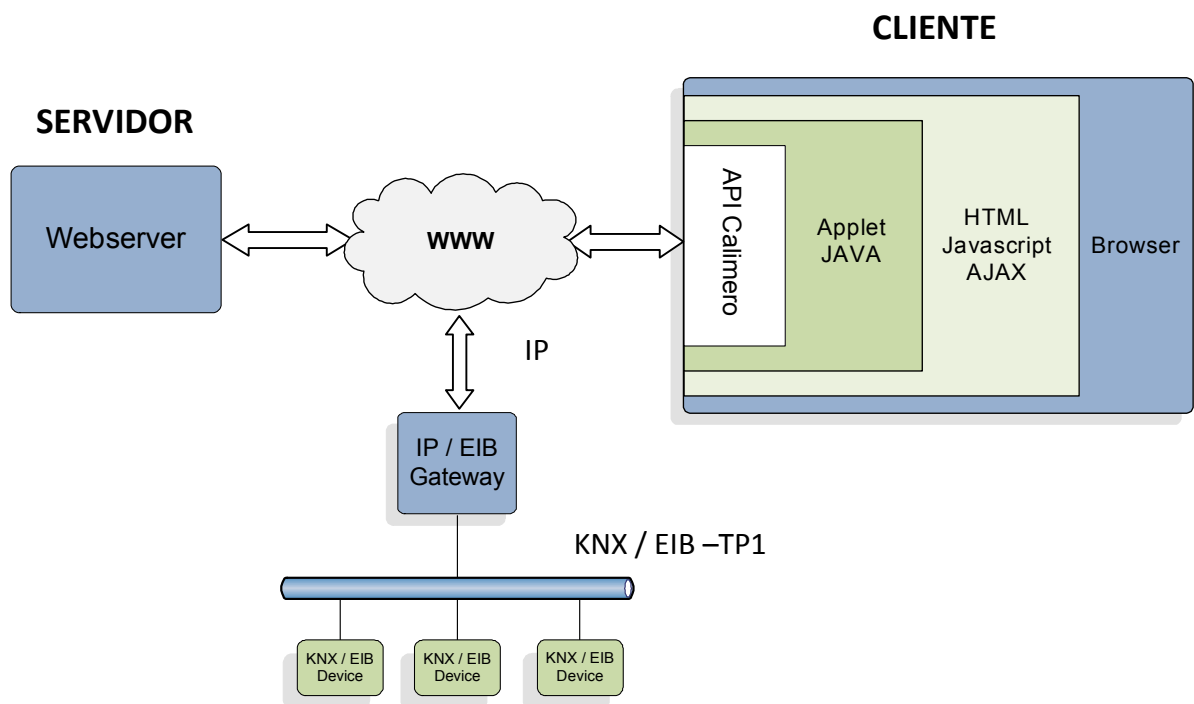


Figura 4.2 – Arquitectura proposta na 2ª alternativa da 1ª abordagem

4.2 Segunda abordagem

A segunda abordagem para a concepção da arquitectura funcional recaiu sobre um tipo de tecnologia diferente, o CGI.

Nesta arquitectura, o *browser* cliente interage com um programa CGI que reside no servidor *Web*. Este programa CGI, por sua vez utiliza um cliente KNX/EIB de interface para estabelecer um canal de comunicação sobre uma rede IP com um *gateway* que efectua o interface entre a rede de dados IP e o barramento KNX/EIB - TP1 da instalação domótica. Este *gateway* também poderá ser um, capaz de fazer o interface com outros meios de comunicação utilizados nas instalações KNX/EIB, como por exemplo a rede eléctrica.

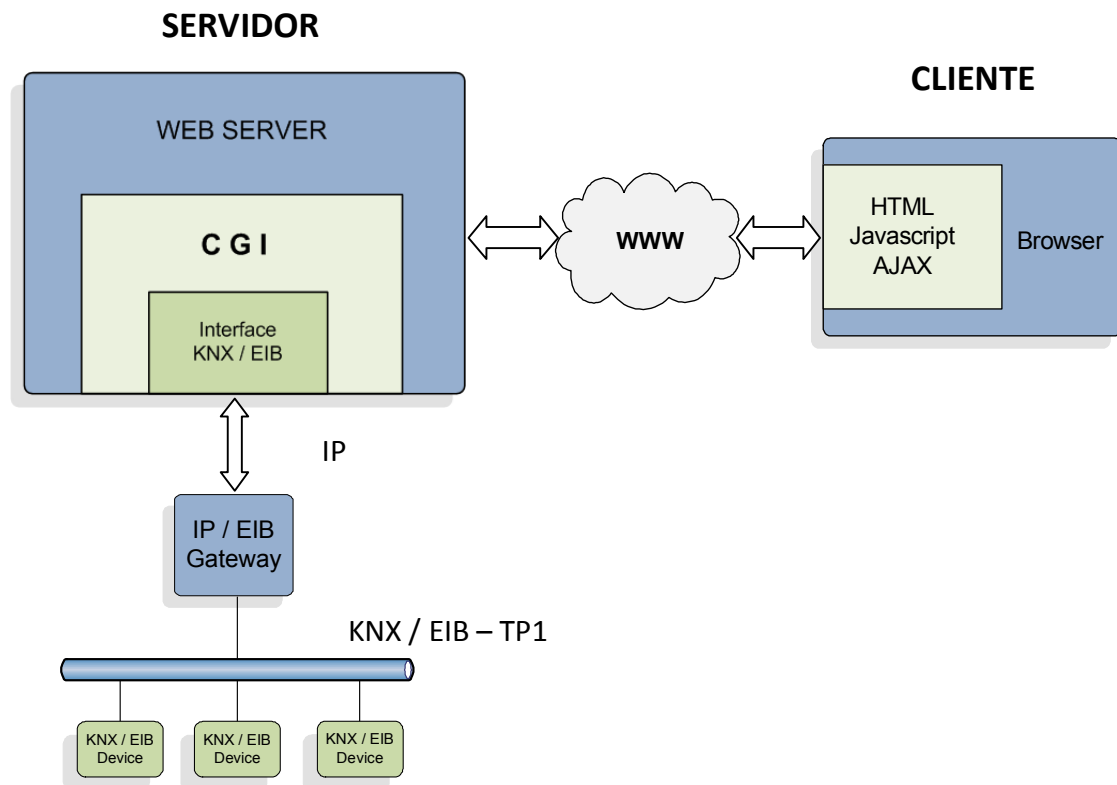


Figura 4.4 – Arquitectura proposta na 2ª abordagem

4.3 Terceira abordagem

A terceira abordagem sobre o problema da concepção da arquitectura funcional recaiu numa linha de pensamento bastante diferente das anteriores abordagens.

Esta abordagem visa utilizar uma plataforma já desenvolvida no âmbito de uma tese de *Stephan Wuchterl* na *University of Applied Sciences Deggendorf, Center for Innovative Communication Systems* orientada pelo *Professor Dr-Ing. Andreas Grzempa*.

Esta plataforma, o *KNX@HOME*, conta actualmente com cerca de sete colaboradores que continuaram a desenvolver o trabalho iniciado por *Stephan Wuchterl*.

4.3.1 KNX @ HOME

O *KNX@HOME* é um projecto de código aberto e desenvolvido com o intuito de controlar instalações de domótica KNX/EIB via HTTP.

Este projecto é baseado em JAVA, uma vez que utiliza a API Calimero, já referida anteriormente; futuramente, os autores esperam introduzir AJAX, de modo a trazer um maior dinamismo e eficiência.

O *KNX@HOME* é constituído por três programas separados: o *KNXService*, *KNXWeb* e o *KNXAdmin*.

Esta terceira abordagem visa a adaptação desta aplicação, tendo em vista os objectivos inicialmente propostos para este projecto.

4.3.1.1 KNXService

O *KNXService* é o núcleo desta aplicação. Este módulo é o responsável por toda a gestão da rede KNX/EIB, da gestão de uma base de dados relacional HSQL e do estabelecimento de um túnel de comunicação sobre a rede IP com um *gateway* IP / KNX. Foi neste módulo que o autor utilizou a API Calimero para a comunicação sobre IP com o *gateway* IP/KNX, embora esta tenha sofrido algumas modificações de modo a melhor satisfazer os objectivos desta aplicação.

4.3.1.2 KNXWeb

Este componente é a aplicação *Web* na qual os utilizadores do KNX@Home podem configurar e interagir com o sistema de uma forma visual. O KNXWeb permite, ainda, a diferenciação de um utilizador normal e de um utilizador com privilégios de administrador - que é o responsável por criar e configurar a informação que será disponibilizada via *Web*, bem como gerir as contas dos diversos utilizadores e respectivos privilégios sobre a actuação na instalação de domótica.

As acções de controlo neste interface com o utilizador são enviadas para o KNXService que as vai interpretar e enviar os comandos respectivos para o barramento da rede KNX/EIB.

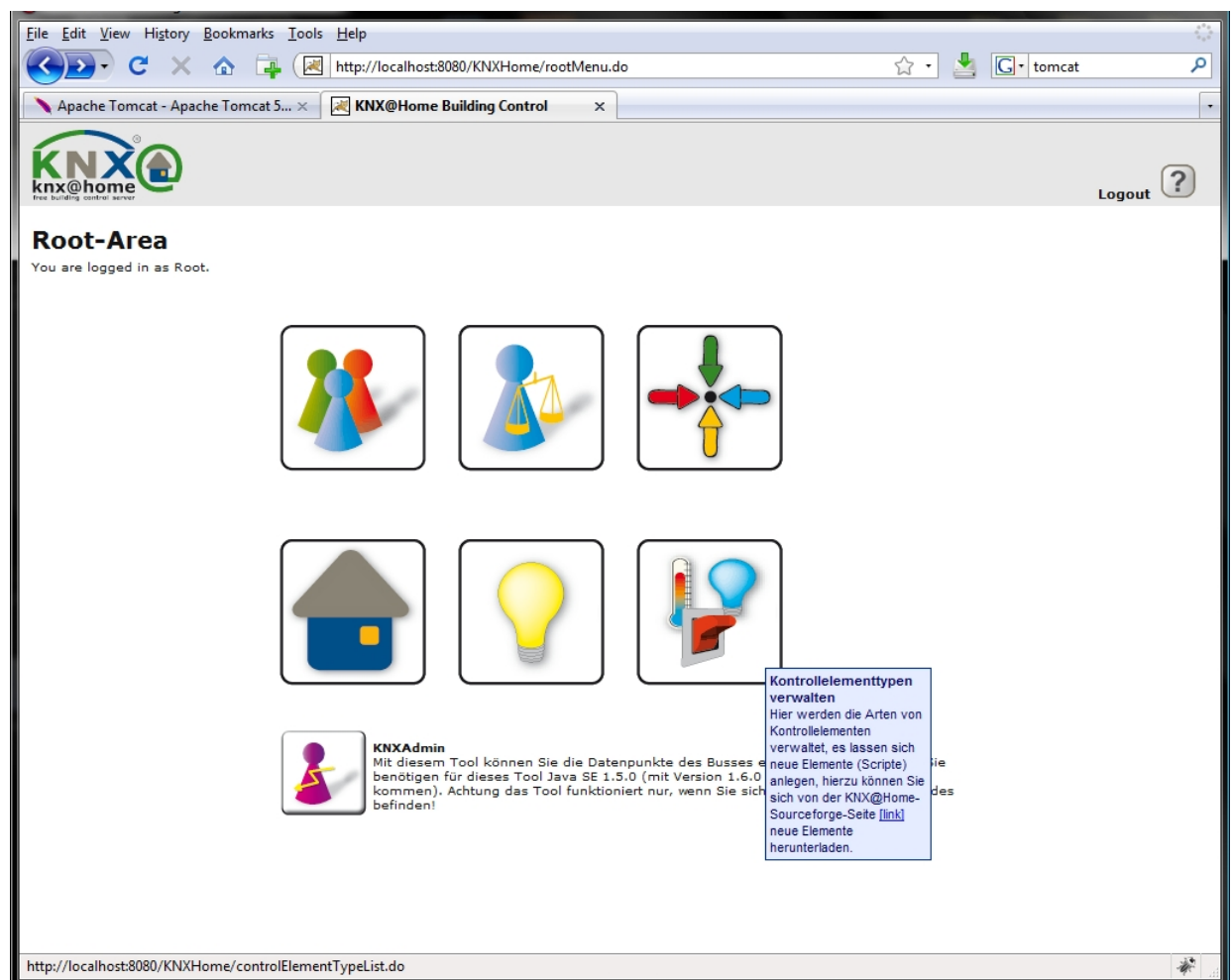


Figura 4.5 – KNXWeb

4.3.1.3 KNXAdmin

Este terceiro componente do KNX@Home permite gerir tudo o que se passa no barramento EIB/KNX.

Esta ferramenta permite-nos monitorizar todas as mensagens / eventos trocados entre o KNXService e a rede EIB/KNX, podendo ficar, assim, com um histórico sobre o estado do sistema, principalmente por razões de segurança.

É também com esta aplicação que os utilizadores adicionam, removem e configuram os endereços de grupo e os respectivos tipos de dados EIS associados, já referenciados na secção 3.1.7.

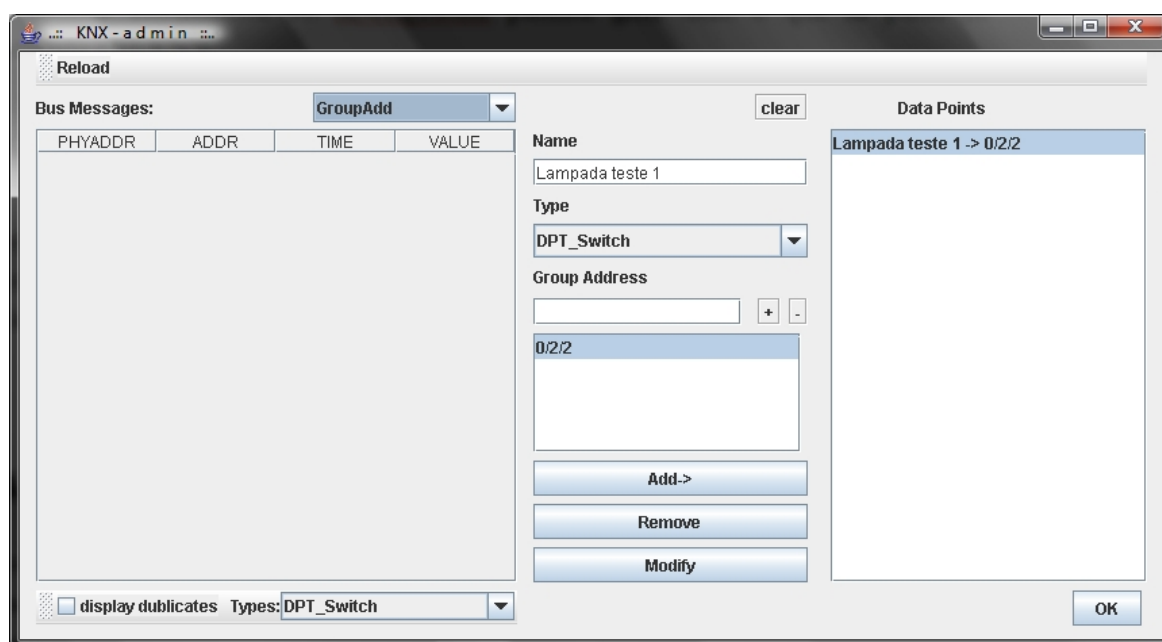


Figura 4.6 – KNXAdmin

4.4 Escolha da arquitectura funcional

Depois da apresentação das abordagens propostas para a arquitectura funcional, é necessário compreender as vantagens e as desvantagens de cada uma destas arquitecturas propostas.

4.4.1 Considerações sobre a primeira abordagem

Analisando as três alternativas propostas na primeira abordagem, podemos optar por seleccionar a mais eficiente, num processo de eliminação sequencial.

Primeiramente, podemos excluir a segunda alternativa, devido a facto de, no conceito de *applet* JAVA, por questões de segurança, não ser possível este carregar ou importar bibliotecas externas de outros *packages* excepto `java.*` e, também, devido ao facto de o *applet* não poder comunicar ou transferir informação sem ser com a máquina de onde este foi transferido. Deste modo, o *applet* não pode integrar a API Calimero dentro deste e nem pode comunicar directamente com o *gateway* IP da rede KNX/EIB.

De seguida, a terceira alternativa pode também ser excluída. Das três alternativas, esta era a que envolvia uma maior complexidade e exigia uma maior quantidade recursos por parte do servidor, não sendo a melhor opção para ser implementada num sistema embebido com capacidade limitadas à partida.

Resta-nos a primeira alternativa, que, de todas as propostas, é a mais equilibrada para o uso da tecnologia JAVA.

4.4.2 Considerações sobre a segunda abordagem

A segunda abordagem apresenta-se como uma proposta extremamente eficiente no que respeita às exigências de recursos computacionais. Esta característica é possível graças à adopção de aplicações CGI desenvolvidas em C, trazendo uma maior eficiência na execução no código.

4.4.3 Considerações sobre a terceira abordagem

Na terceira abordagem, a adaptação do projecto KNX@Home apresenta alguns factores limitativos. Toda a documentação disponível sobre o projecto e parte do interface *Web* encontra-se escrita em alemão, não existindo qualquer versão na língua inglesa, e que, aliada à complexidade do próprio projecto e aos requisitos computacionais mínimos necessários para o servidor do projecto (>1GHz, >256MB RAM e 200MB livres), torna esta abordagem praticamente inviável.

4.4.4 Conclusão

Depois de terem sido apresentadas as várias abordagens e respectivas considerações, avançou-se para uma solução final.

A alternativa JAVA limita, em parte, o domínio dos dispositivos capazes de receber a aplicação *Web*, uma vez que coloca todo o peso do interface do lado do cliente. Esta solução restringe à partida grande parte dos PDA's, *smartphones*, telemóveis e outros dispositivos portáteis onde não há uma *Java Virtual Machine* bem definida, *standard* e sem custos para o utilizador como existe para as plataformas *desktop* comuns por parte da Sun Microsystems. A mesma limitação referente à máquina virtual pode também ser observada no servidor, se se recorrer a uma plataforma embebida com *hardware* mais “exótico”, como é o caso da grande parte dos sistemas embebidos.

A terceira abordagem, apesar estar funcionalmente completa apresenta as limitações já referidas nas considerações sobre a mesma. De todas estas limitações, são fundamentalmente os requisitos mínimos de *hardware* necessários para o servidor, que inviabilizam a escolha desta proposta. Seria necessário ter um PC disponível e sempre ligado (com o respectivo consumo eléctrico) para além do valor envolvido na sua compra. Esta solução não seria compatível com o baixo consumo e o baixo custo para a plataforma de hardware do servidor, tal como o definido nos objectivos propostos inicialmente.

A segunda abordagem proposta para a implementação da arquitectura funcional demonstra um bom compromisso entre a eficiência de execução das aplicações CGI, como foi evidenciado na secção 3.2.4, e uma complexidade de implementação compatível com o tempo disponível para a realização deste projecto. Devido à sua melhor eficiência, esta abordagem apresenta os menores requisitos em termos de hardware. Esta abordagem demonstra também a maior abrangência no suporte a dispositivos clientes, uma vez que a base do interface de controlo apenas depende de elementos simples HTML e do JavaScript (disponível em praticamente todos os *browsers*).

Com base nesta análise, a proposta escolhida para a implementação da arquitectura funcional do sistema foi a segunda abordagem apresentada.

Capítulo 5

5. Validação experimental

Este capítulo é dedicado à implementação prática de um exemplo simples e à apresentação dos resultados experimentais que permitem validar a arquitectura funcional proposta no capítulo anterior. O exemplo prático ilustrativo do funcionamento desta arquitectura consiste no comando de um dispositivo KNX/EIB do tipo EIS1 via *Web*.

5.1 Descrição do *hardware*

5.1.1 Sistema embebido para a plataforma do servidor

De modo a suportar a aplicação de servidor relativa a este projecto, é necessária a escolha de uma plataforma de *hardware* que cumpra os objectivos propostos no que respeita ao baixo custo e ao baixo consumo eléctrico desta.

Foram seleccionadas três opções que cumprem estes objectivos e são apresentados de seguida:

A) ICnova AP7000 Base

B) Picotux

C) Router

5.1.1.1 ICnova AP7000 Base

Esta secção é dedicada à apresentação do sistema embebido ICnova AP7000 (figura 5.1) da empresa *In-Circuit* [28], cujas principais características se encontram descritas nas tabelas 5.1 e 5.2. Esta plataforma tem um custo unitário de 95€.

Tabela 5.1 – *Hardware* ICnova AP7000

CPU	Atmel AVR32 @ 140MHz
RAM	64 MB
Flash	8 MB
Ethernet	10/100Mbit
Interfaces	USB-UART via CP2102
Alimentação	Conversor DC/DC integrado

Tabela 5.2 – *Software* ICnova AP7000

Sistema Operativo	uClinux 2.6.24
Shell	Busybox 1.0
Aplicações	Servidor <i>Web</i> , Telnet, SSH <i>daemon</i>



Figura 5.1 – ICnova AP7000

5.1.1.2 Picotux

Outro sistema embebido que cumpre os objectivos estabelecidos para o presente trabalho é o Picotux (figura 5.2) da empresa Kleinhenz Elektronik [26], tendo um custo unitário de 142€. As principais características desta plataforma encontram-se nas tabelas 5.3 e 5.4.

Tabela 5.3 – *Hardware* Picotux

CPU	32-bit ARM 7 Netsilicon NS7520 @ 55mhz
RAM	8 MB
Memória Flash	2 MB
Ethernet	10/100Mbit
Interfaces	Série TTL, até 230.400 bps
Alimentação	250mA @ 3,3V

Tabela 5.4 – *Software* Picotux

Sistema Operativo	uClinux 2.4.27
Shell	Busybox 1.0
Sistema de ficheiros	CRAMFS, JFFS2, NFS
Aplicações	Servidor <i>Web</i> , Telnet



Figura 5.2 – Picotux

5.1.1.3 Router

O principal requisito para uma plataforma de monitorização e controlo *Web* é um acesso sempre disponível de banda larga à Internet. Como a generalidade dos ISPs (*Internet Service Providers*) fornece um *router* para os serviços de banda larga, e, como já foi referido na secção Sistemas embebidos do capítulo 3, a maioria dos equipamentos de rede (onde se encaixam os *routers*) é baseada em Linux, surgiu a ideia de poder utilizar o *router* como plataforma para a aplicação do servidor. Esta opção não requer mais nenhum equipamento extra, como é o caso das duas opções anteriormente mencionadas, beneficiando da ausência do custo de aquisição da plataforma, da ausência do consumo extra de energia por parte de uma segunda plataforma e da liberdade oferecida para o controlo local através de plataformas móveis (PDAs, *Smartphones*, etc), proporcionada pelas capacidades *wireless* da generalidade dos *routers* disponíveis no mercado.

A) Router OEM

Existem empresas que se dedicam exclusivamente ao desenvolvimento e à concepção de equipamentos de rede, que fornecem em regime de OEM/ODM *outsourcing* a vários clientes, nomeadamente a grandes empresas do sector de redes de computadores, como a Linksys, Cisco, Dlink (entre outras), sendo estas que fazem o desenvolvimento de todo o *software*, a comercialização e fornecem o suporte deste tipo de equipamentos ao utilizador final.

A opção seleccionada para esta hipótese é o *router wireless* ilustrado na figura 5.3. Este equipamento é baseado num projecto da Accton [43], empresa situada em Taiwan e dedicada principalmente à concepção e fabrico de equipamentos de rede para terceiros (OEM/ODM), como já foi referido.

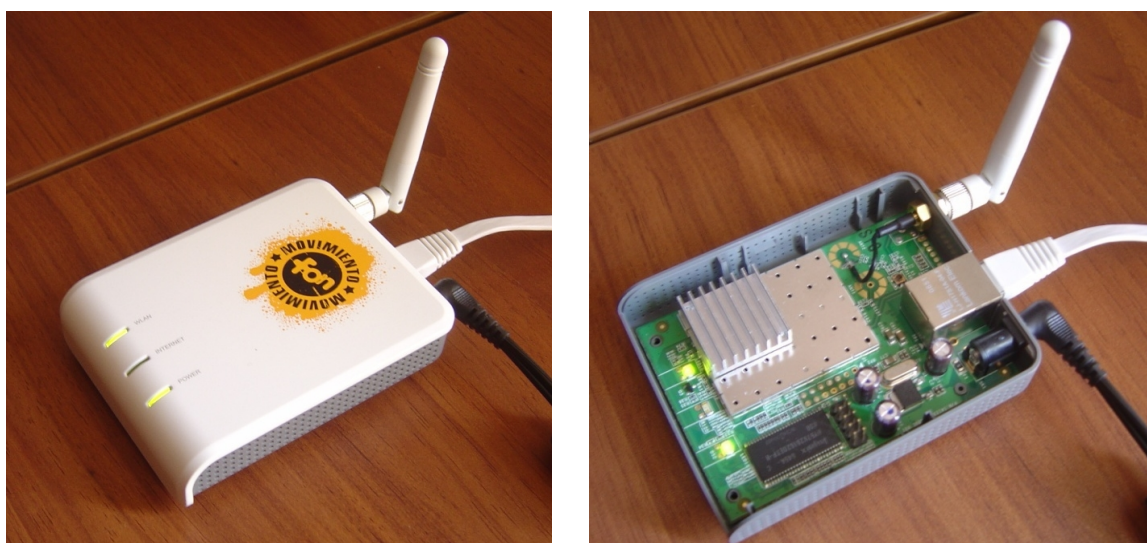


Figura 5.3 – Router OEM disponível para *designs* de terceiros

Tabela 5.5 – *Hardware router OEM*

CPU	Atheros AR2315 @ 180MHz
RAM	16 MB
Memória Flash	8 MB
Ethernet	10/100Mbit
Wireless	IEEE 802.11b / 802.11g (até 54 Mbps)

Tabela 5.6 – *Software router OEM*

Sistema Operativo	OpenWRT (linux)
Shell	Busybox 1.0
Sistema de ficheiros	JFFS2
Aplicações	Servidor <i>Web</i> , Telnet, SSH <i>daemon</i>

B) Router comercial

Existe também a possibilidade de se usar um *router* comercial, já disponível no mercado para o utilizador final. Como opção para esta hipótese, seleccionou-se o *router wireless* Linksys WRT54G. Este modelo é um dos equipamentos de maior sucesso no mercado, devido à sua boa relação funcionalidade / qualidade / preço. Este modelo existe desde Dezembro de 2002, tendo, até à actualidade, sofrido várias revisões de hardware ao nível da memória e do processador. Apesar de o fabricante não divulgar as especificações técnicas no que respeita ao *hardware* e ao *software*, é possível, através da cooperação de uma comunidade *online* com vários utilizadores deste modelo recolher informação sobre o *hardware* envolvido nas várias actualizações.

Tabela 5.7 – *Hardware router Linksys WRT54G*

CPU	Broadcom 5354 @ 240Mhz
RAM	8 MB
Memória Flash	2 MB
Ethernet	10/100Mbit
Wireless	IEEE 802.11b / 802.11g (54Mbps)



Figura 5.4 – *Router* comercial disponível para o utilizador final

5.1.1.4 Avaliação da plataforma para o servidor

A plataforma *picotux* (figura 5.2) apresenta-se como boa solução, uma vez que dispõe de um reduzido consumo eléctrico ($< 1W$) e os interfaces necessários. Contudo, esta plataforma dispõe de um sistema de armazenamento demasiado pequeno (2MB *Flash*) para o armazenamento da aplicação *Web*, tendo ainda de partilhar 720KB com o S.O. Esta situação acaba por inviabilizar a escolha desta solução.

A plataforma da empresa alemã *In-Circuit* [28], *ICnova AP7000 Base* (figura 5.1), apesar de ser bastante recente, apresenta argumentos fortes, como: CPU bastante rápido (pode funcionar até 200MHz), memória RAM oito vezes superior ao *picotux*, memória *Flash* para armazenamento quatro vezes superior. Apesar de o seu consumo eléctrico ser no máximo três vezes superior ($< 2,5 W$), este mantém-se numa gama de valores aceitável para estar sempre ligado, aproximadamente 0.09€ por mês (*picotux*) e 0,21€ por mês (*ICnova AP7000*).

A escolha de um *router* na sua vertente comercial como plataforma para o servidor não é trivial. Apesar de ser, sem dúvida, uma plataforma eficaz (pelas razões já atrás mencionadas), esta opção pode apresentar problemas de legalidade dúbia. Para os fabricantes dos *routers* disponíveis no mercado, não é geralmente vantajoso que aplicações externas sejam executadas nos seus equipamentos, pelo que optam pela protecção dos mesmos contra essa eventualidade. Isto implica que qualquer acção que tenha como objectivo o ultrapassar destas restrições possa ser encarada pelo fabricante como uma violação da sua propriedade intelectual.

Para ultrapassar estas possíveis dificuldades e tornar a escolha de um *router* como uma opção viável (com vantagens também ao nível da personalização do *software* e ao nível da contenção dos custos), seria necessário adquirir o equipamento directamente na origem (onde é tipicamente concebido o *hardware*), da mesma forma que as grandes empresas (já

atrás enunciadas) fazem, antes de estas lhe colocarem o seu *software* proprietário, e disponibilizar o produto final nos seus revendedores oficiais.

Existe, no entanto, um outro problema decorrente desta solução: a elevada diversidade do *hardware* envolvido neste tipo de equipamentos, nomeadamente na arquitectura dos microprocessadores que os equipam. Devido a esta característica, os programas desenvolvidos para uma plataforma poderão não ser compatíveis com outros equipamentos, mesmo sendo do mesmo fabricante, e inclusive, do mesmo modelo. No modelo pré-seleccionado como opção - o *router* comercial Linksys WRT54G - a situação descrita pode ser verificada. Neste modelo, desde que foi lançado (Dezembro 2002) até à actualidade, o *hardware* tem vindo a sofrer várias alterações ao nível da memória, periféricos e, principalmente, ao nível dos processadores. Estes têm sido alterados de acordo com o evoluir dos novos processos de fabrico (com a respectiva melhoria da eficiência energética e da capacidade de processamento) e da integração de novas tecnologias (USB por exemplo) por parte dos diversos fabricantes deste tipo de microprocessadores. Ao longo desta evolução, o fabricante do *router* em questão (Linksys) optou por usar processadores de diversos fabricantes diferentes (Broadcom, Atheros, Texas Instruments), conforme lhe fosse mais vantajoso em termos de custos e de funcionalidades oferecidas. Esta diversidade levou a diversas incompatibilidades no *software* entre as várias revisões do mesmo modelo.

Podemos, então, concluir, que a plataforma mais adequada (e primeira opção dentro das disponíveis) ao objectivo do caso em estudo é a ICnova AP7000 Base.

5.1.2 Sistema de testes

De modo a poder validar fisicamente o comportamento da arquitectura proposta, foi utilizado um painel didáctico com uma instalação de domótica EIB da Siemens, construído na FEUP no âmbito de um projecto anterior [10].

Este projecto pretendia simular um auditório duplo e de uma divisão de controlo denominada por régie. A lista de dispositivos por divisão é apresentada tabela 5.1.

Tabela 5.8 – Dispositivos presentes por divisão na simulação

Auditórios		Régie	
2	Lâmpadas incandescentes reguláveis (Halogéneo)	1	Lâmpada incandescente fixa
2	Lâmpadas fluorescentes compactas reguláveis	1	Touch Panel Vision
2	Lâmpadas fluorescentes compactas fixas	1	Dispositivo de comando quádruplo
1	Simulador de estores eléctricos		
1	Receptor de infravermelhos		
1	Dispositivo de comando quádruplo		

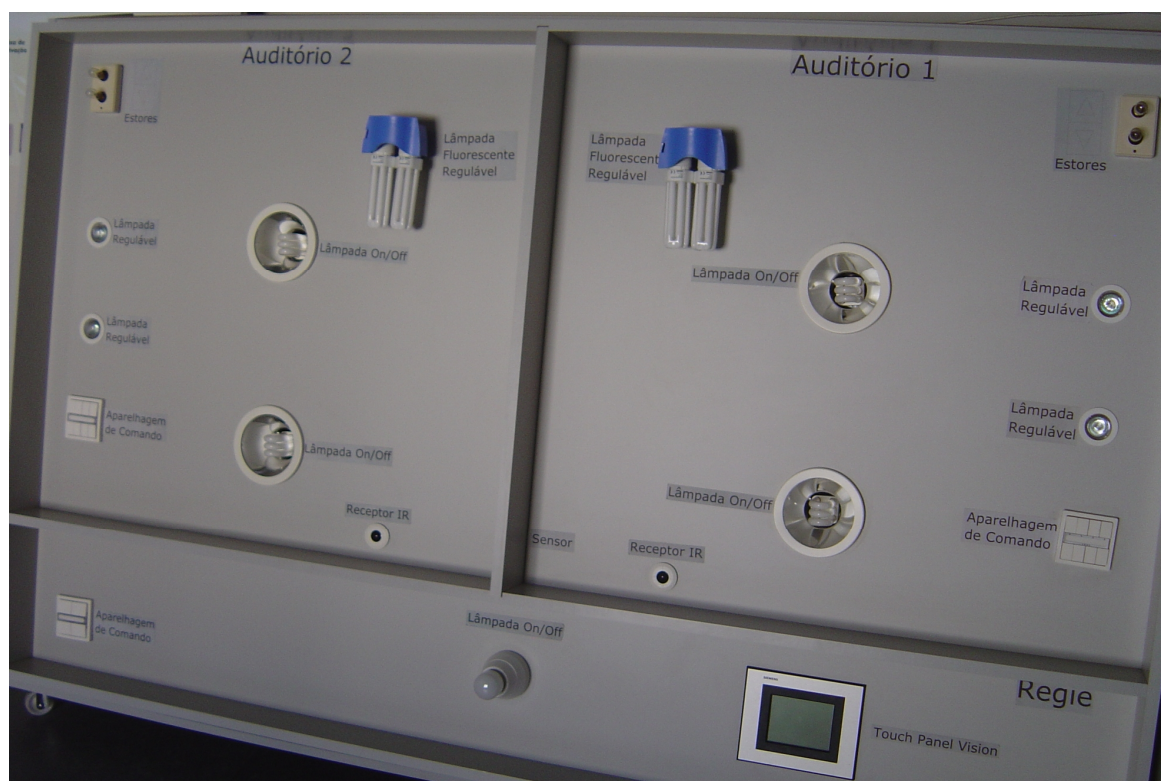


Figura 5.5 – Vista frontal do painel EIB

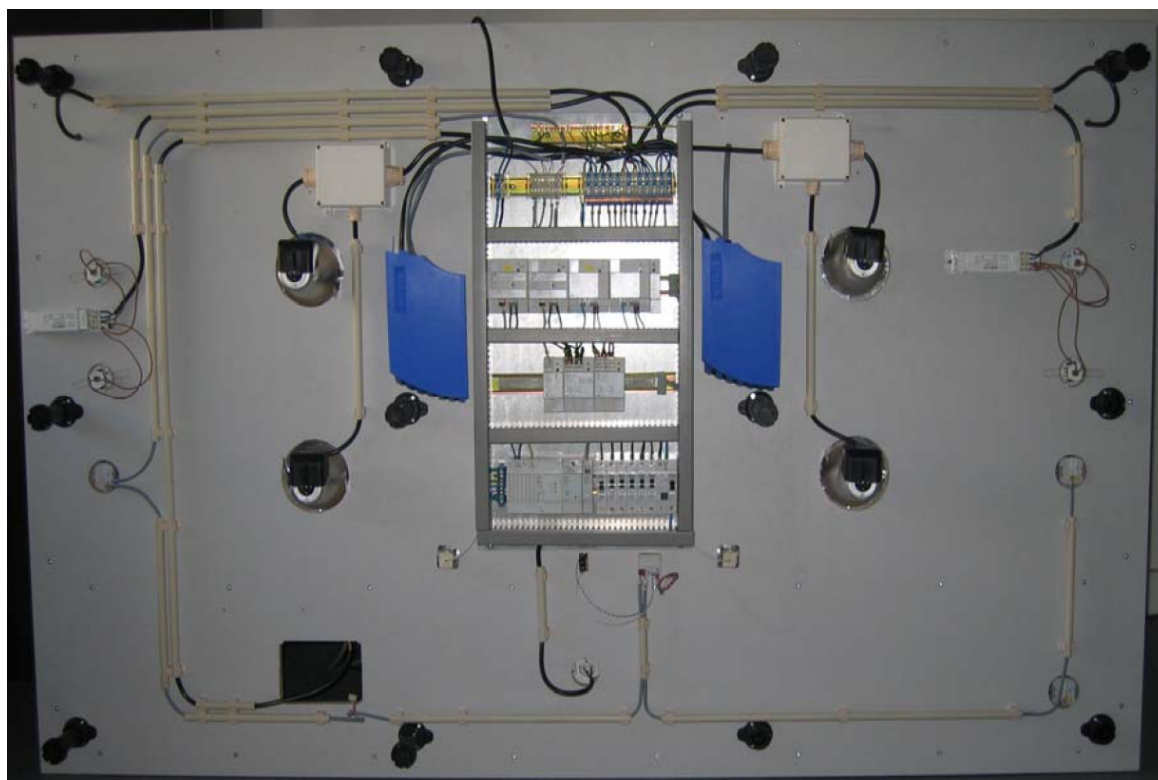


Figura 5.6 – Vista traseira do painel EIB

- Legenda:**
- 1 Fonte de alimentação
 - 2 Filtro Indutivo
 - 3 Ligador de duas ligações
 - 4 Descodificador de infravermelhos
 - 5 Saída binária
 - 6 Controlo de estores
 - 7 Regulador de Fluxo de Lâmpadas Fluorescentes
 - 8 Regulador de Fluxo de Lâmpadas Fluorescentes
 - 9 Gateway RS-232

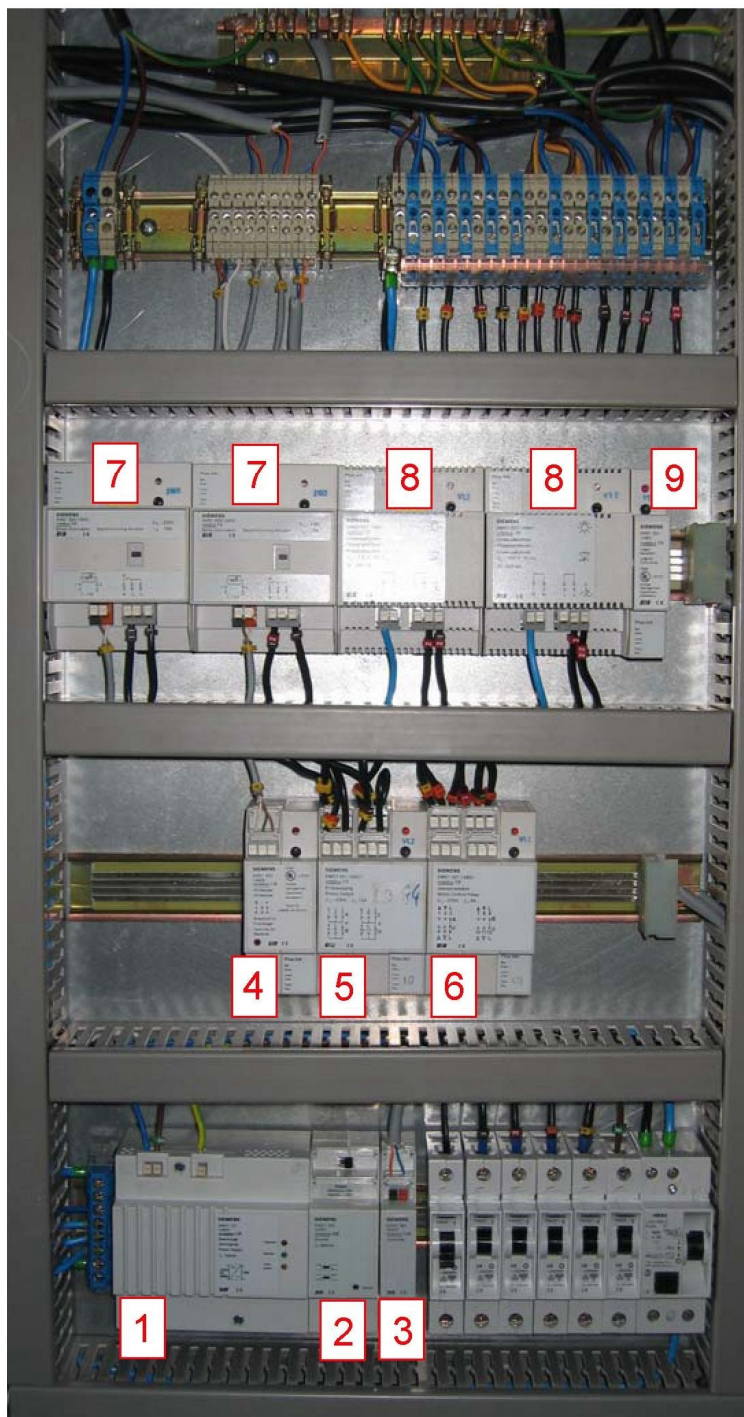


Figura 5.7 – Legenda da instalação EIB

5.1.3 Gateway KNX/EIB

Como já foi referido inúmeras vezes ao longo deste relatório, um *gateway* é um dispositivo que permite fazer o interface entre duas redes que utilizam diferentes protocolos de comunicação, de modo a permitir a interoperabilidade entre dois sistemas distintos.

O *gateway* necessário para a aplicação ao caso em estudo teria de permitir a interligação entre o barramento KNX/EIB TP1 (par entrançado) e as redes de dados que utilizam o protocolo IP, de modo a que computadores ou outros equipamentos possam trocar dados com dispositivos KNX/EIB.

O gateway IP escolhido foi o Siemens N148/21 (figura 5.8)



Figura 5.8 – Gateway IP Siemens N148/21

Este *gateway* IP utiliza a norma KNXnet/IP *Tunneling*, que possibilita o envio de tramas KNX/EIB através de uma rede IP, permitindo configurar, monitorizar e controlar a rede KNX/EIB remotamente.

A conexão física ao barramento KNX/EIB - TP1 do painel de simulação, apresentado na secção anterior, é estabelecida através de um terminal de duas ligações (conector branco e vermelho na figura 5.8). Para a ligação à rede de dados (IP através de 10BaseT) o dispositivo contém um conector RJ45. O *gateway* pode ser alimentado por 24V AC/DC (conector vermelho e preto da figura 5.8) [29][30].

5.1.4 Topologia do sistema de monitorização e controlo Web

Na figura 5.9 podemos observar a topologia dos diversos equipamentos que integram o sistema de configuração, monitorização e controlo *Web*.

No domínio da habitação (ou edifício a controlar), a instalação KNX/EIB – TP1 está ligada à rede IP do edifício via *gateway* IP (Siemens N148/21). É nesta rede de dados que estão ligados os diversos dispositivos que compõem o sistema, entre eles o sistema embebido com a aplicação de servidor (que irá ser apresentada na próxima secção), os clientes locais (PC, PDA ou *smartphone*) e o *modem/router*, que fornece ao exterior do domínio do edifício, o acesso à rede local e à instalação KNX/EIB.

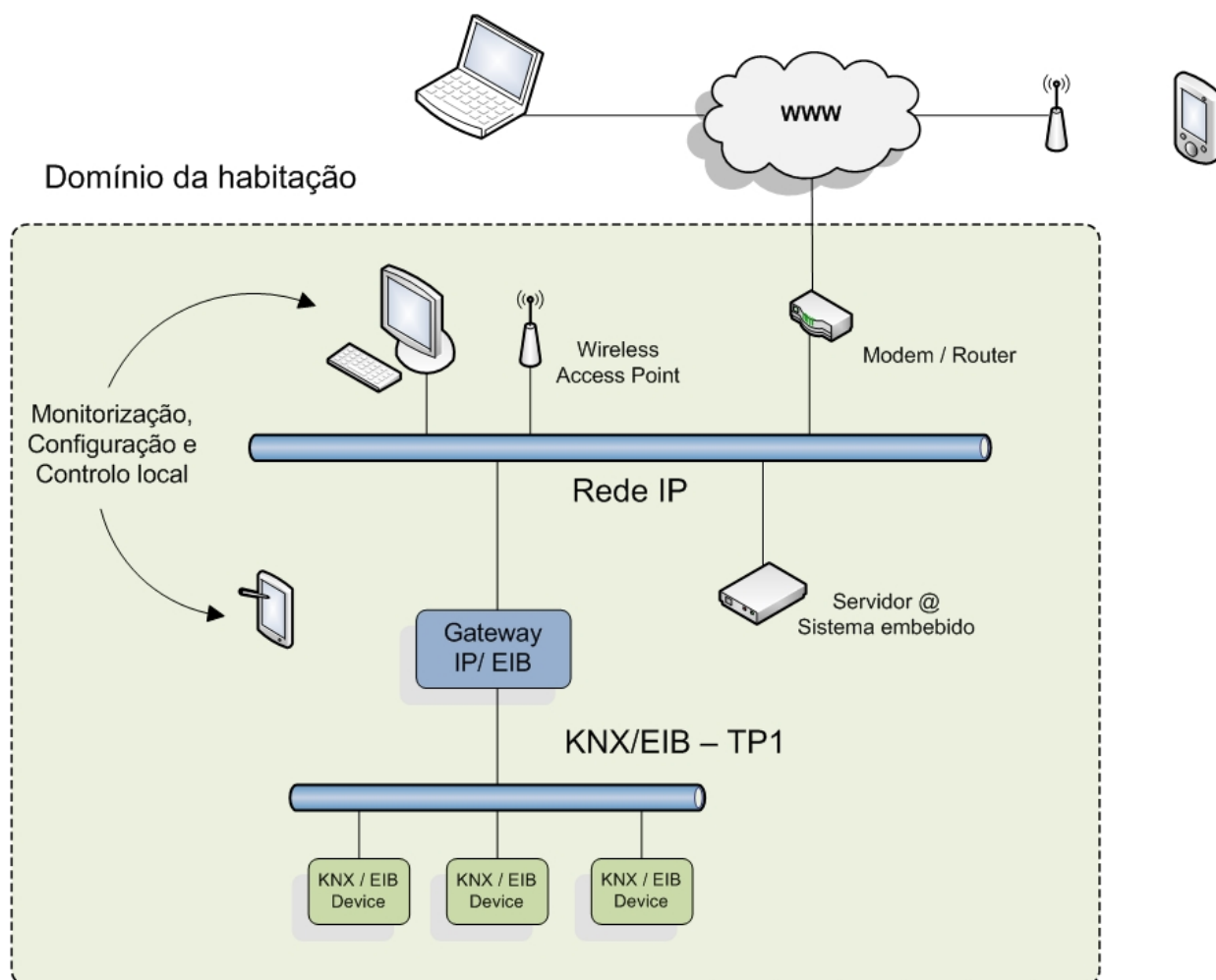


Figura 5.9 – Topologia do sistema de monitorização e controlo *Web*

5.2 Software

5.2.1 Primeira fase

A primeira fase tem como objectivo a implementação da estrutura de aplicação do servidor. Nesta fase, a implementação terá como destino a execução num PC, facilitando assim o desenvolvimento da aplicação e o *debug* de possíveis erros.

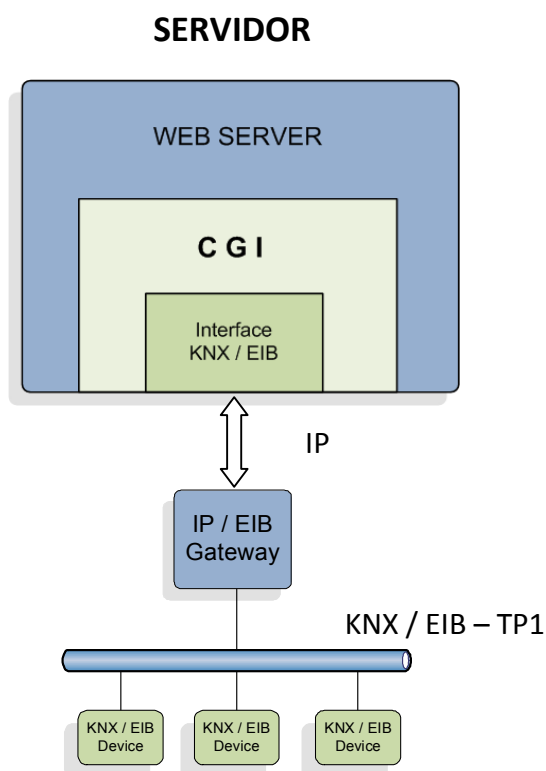


Figura 5.10 – Estrutura de aplicação do servidor

5.2.1.1 Aplicação CGI de interface com a rede KNX/EIB

O objectivo principal da aplicação CGI a implementar é o de receber os dados introduzidos pelo utilizador no browser e actuar em conformidade na rede KNX/EIB.

No diagrama da figura 5.11, podemos visualizar de forma gráfica o comportamento desta aplicação.

Primeiramente, um evento no browser efectua um pedido assíncrono ao servidor, via Ajax, com os dados relevantes para a acção a efectuar (endereço de grupo e valor). Na chamada da aplicação CGI são recolhidos os dados por um dos dois métodos disponíveis: GET ou

POST. De seguida, existe uma rotina que analisa o conteúdo dos dados e verifica se estes são válidos, isto é, se são coerentes com os dispositivos KNX/EIB. Em caso afirmativo, é efectuado um túnel de comunicação na rede IP tendo como destino o *gateway* IP/KNX. Depois de ser efectuada uma conexão válida, a trama KNX/EIB com os dados recebidos do cliente, é encapsulada numa trama IP e enviada para o *gateway*. Este fica responsável por fazer o processo inverso, ou seja, retirar da trama IP a trama KNX/EIB, e então, injectá-la no barramento KNX/EIB – TP1 para que esta actue sobre o dispositivo endereçado. Depois de a comunicação ter sido efectuada com sucesso, a ligação com o *gateway* IP é terminada e é actualizado o novo estado do dispositivo na página *Web*.

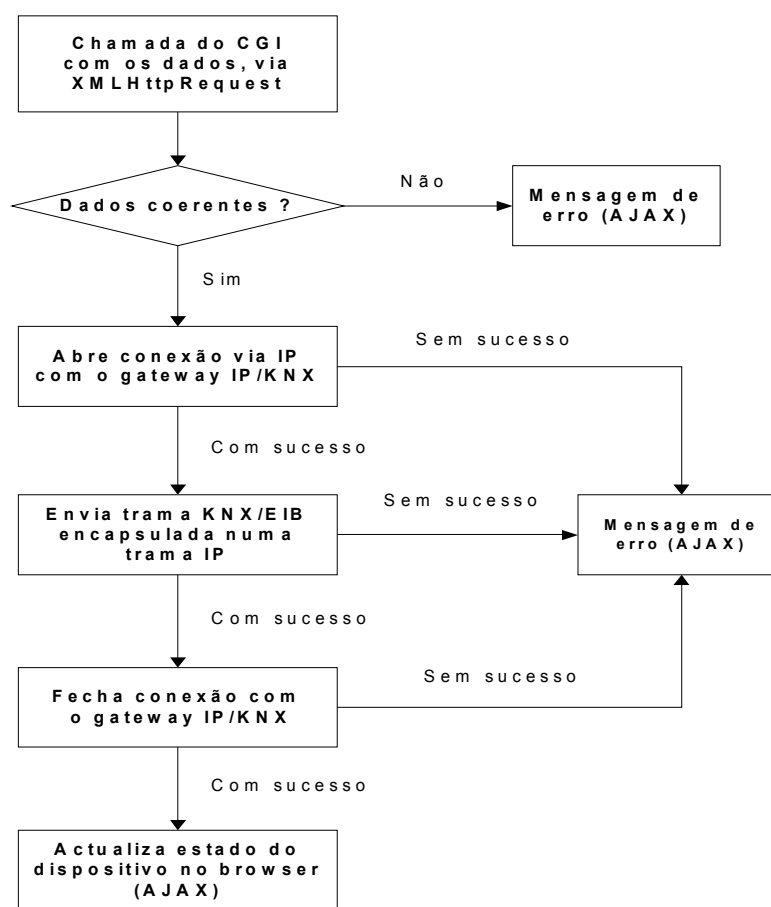


Figura 5.11 – Diagrama de execução do CGI

De modo a que a aplicação CGI possa fazer a conexão ao gateway IP/KNX, é necessário um cliente KNX/EIB que estabeleça um túnel de comunicação IP até este.

Chegados a esse ponto, temos duas hipóteses: implementar um cliente KNX/EIB ou usar um cliente já desenvolvido que forneça os métodos necessários.

Uma vez que a hipótese da implementação seria incompatível com o tempo disponível para a realização deste projecto, devido fundamentalmente à necessidade de um estudo

mais profundo e exaustivo do *standard* KNX/EIB, optou-se pelo uso de um cliente já desenvolvido.

Nesta área existem duas hipóteses disponíveis, o *eibnetmux* [31] e o *eibd* [32]. Ambos são clientes KNX/EIB de código aberto, que fornecem métodos em C e PHP que permitem aceder aos dispositivos na rede, abstraindo o utilizador da complexidade do protocolo KNXnet/IP *Tunnelling*.

Para este projecto foi utilizado o *eibd* porque é o cliente que conta com uma maior maturidade, ao contrário do mais recente *eibnetmux* que ainda se encontra numa versão experimental, demonstrando alguma instabilidade no seu funcionamento.

5.2.1.2 Interface *Web* exemplo para a validação

De modo a facultar ao utilizador a actuação de um dispositivo na rede KNX/EIB-TP1, foi elaborado um interface *Web* simples, que implementa uma função do tipo EIS1 “*Switching*“ para ligar/desligar uma lâmpada, com toda a tecnologia escolhida para esta aplicação.

Este interface, ilustrado na figura 5.12, permite inserir um endereço de grupo e mudar o seu valor. Quando é accionado o botão, é submetido para o servidor um pedido assíncrono (Ajax) com o endereço e o valor requeridos. A aplicação CGI, no servidor, interpreta os dados recebidos e actua sobre a rede, tal como foi explicado na secção anterior. No final da execução do CGI, se todos os passos tiverem sido correctamente efectuados - isto é, sem erros - o novo estado é dinamicamente alterado na página *Web*, sem ser efectuado um novo carregamento desta. A figura 5.13 ilustra a mudança no estado do dispositivo no caso de uma actuação efectuada com sucesso, e a figura 5.14 ilustra um caso de erro no processo de mudança de estado ou uma falha no dispositivo.

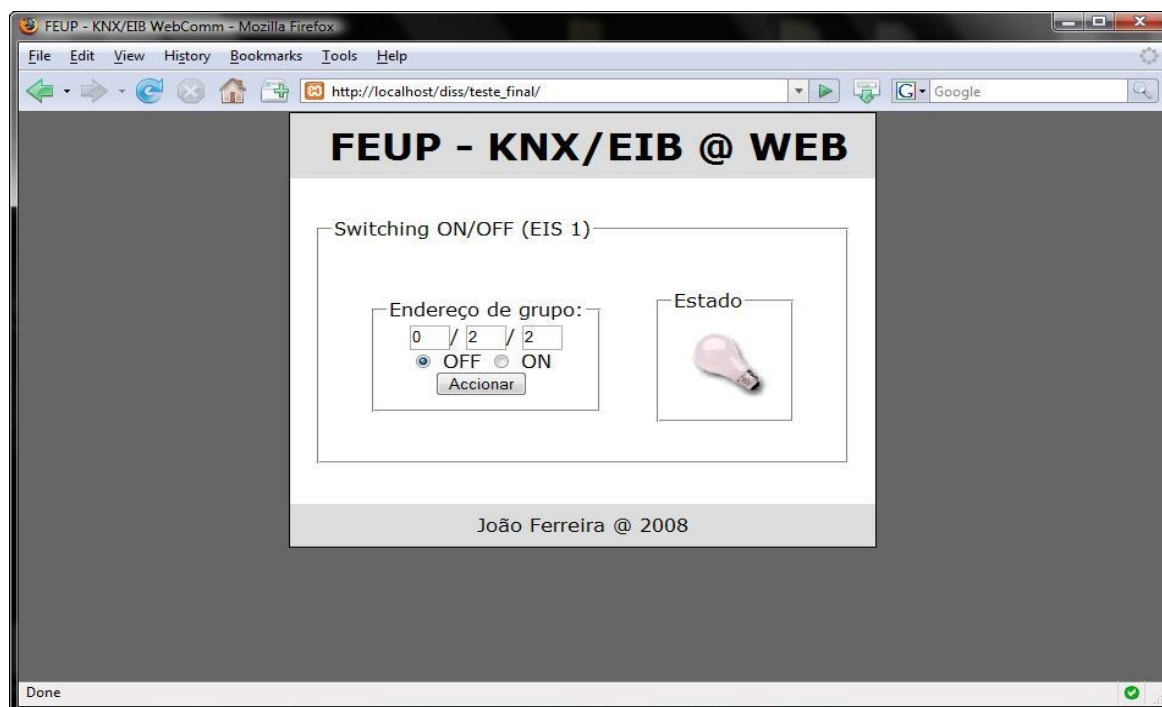
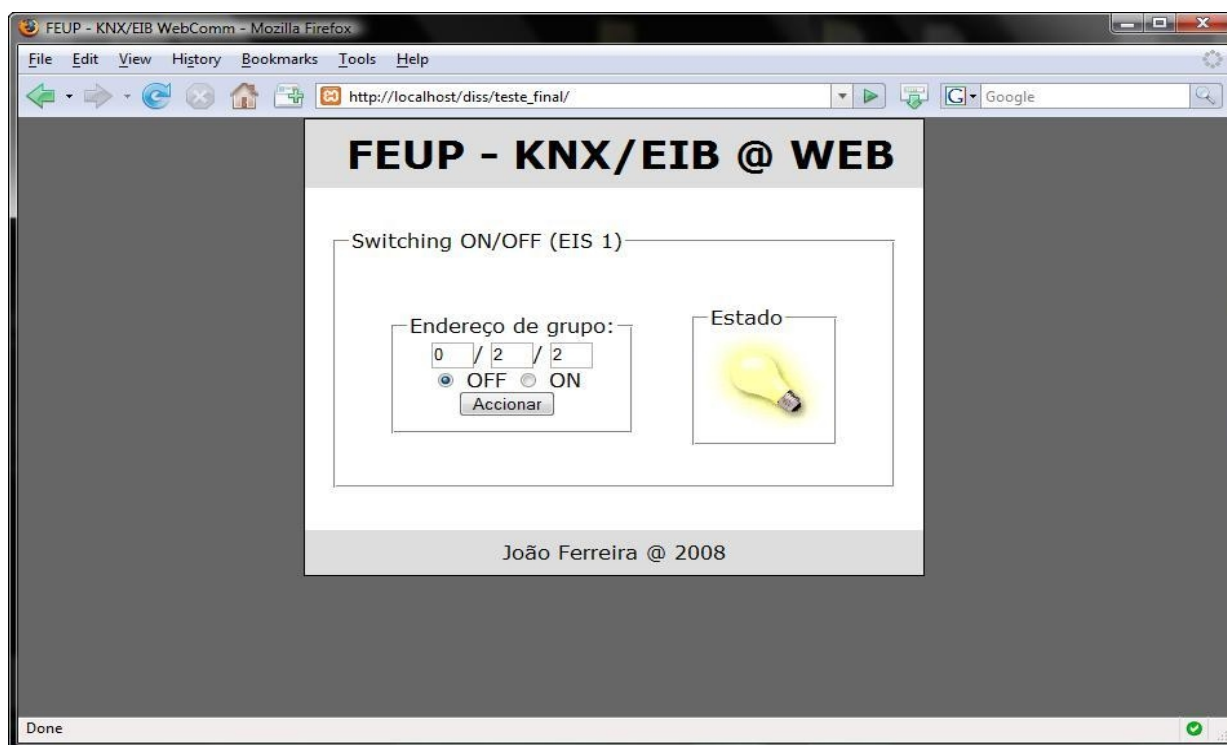
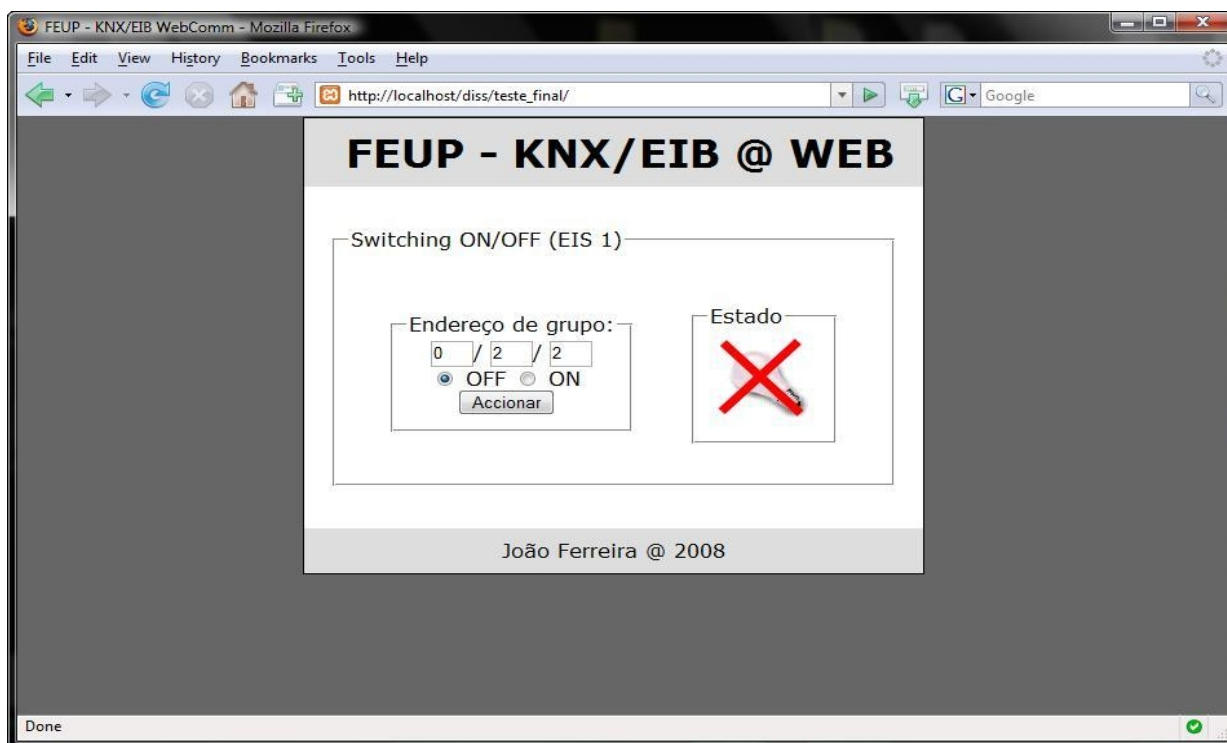


Figura 5.12 – Interface *Web* com dispositivo desactivado

Figura 5.13 – Interface *Web* com dispositivo activadoFigura 5.14 – Interface *Web* com erro no dispositivo

De modo a ultrapassar as pequenas discrepâncias existentes entre os diversos *browsers*, facilitar a selecção de elementos de uma página e a reduzir ao máximo a complexidade das comunicações assíncronas, a interacção em Ajax com o servidor foi implementada recorrendo a biblioteca jQuery, já brevemente resumida no capítulo 3.

Foi utilizado o método `jQuery.get()` presente nesta biblioteca para implementar o pedido assíncrono ao servidor.

Este método encontra-se definido na documentação oficial da biblioteca [17]:

```
jQuery.get( url, [data], [callback] )
```

O parâmetro url representa o caminho relativo para uma página a carregar, sendo, neste caso em estudo, o caminho para a aplicação CGI no servidor.

O parâmetro data representa as variáveis que serão enviadas para o servidor, que, neste caso, serão os três níveis do endereço de grupo do dispositivo e o valor a actuar neste.

O parâmetro callback representa a função que será invocada quando os dados forem carregados com sucesso.

Os parâmetros [] são opcionais.

```
$("#accionar").click(  
function(){  
    var val ;  
    if( $("#val_off").attr('checked')==true){val=0;}  
    if( $("#val_on").attr('checked')==true){ val=1;}  
    $.get("/cgi-bin/cgi_eis1.cgi",  
    {g_addr_1 : $("#g_addr_1").val() ,  
    g_addr_2 : $("#g_addr_2").val(),  
    g_addr_3:$("#g_addr_3").val(), val : val},  
    function(data){ $("#imagem").html(data); })  
})
```

O código acima mencionado representa a implementação efectuada deste método no caso em estudo. O selector `$()` do jQuery permite-nos seleccionar de forma fácil todos os

elementos constituintes de uma página Web. A acção de um *click* no botão com a identificação “accionar” irá seleccionar os *radio buttons* e verificar o seu estado. De seguida, será chamada a aplicação CGI do servidor com os dados relevantes. Quando receber a resposta do servidor, o elemento HTML com a identificação de “imagem” será actualizado com os novos dados. Este elemento será a imagem que representa o estado dispositivo.

5.2.2 Segunda fase

Após a implementação efectuada na primeira etapa, a segunda fase tinha como por objectivo o *port* da implementação efectuada na etapa atrás descrita, para o sistema embebido escolhido na secção 5.1.1, a placa ICnova AP7000.

Após o estudo efectuado sobre o método e as ferramentas de *cross-compiling*, foi elaborado um guia de *cross-compiling* da aplicação para a plataforma escolhida (AVR32), que se encontra como anexo deste relatório. Este guia tem todos os passos necessários para a compilação de toda a aplicação do servidor para um sistema embebido.

Espera-se, também, que este guia seja útil para a grande maioria dos sistemas embebidos, uma vez que apenas será necessário mudar a variável *host* que é passada como argumento aos vários *scripts* de configuração das bibliotecas utilizadas.

A aplicação CGI de teste, o cliente KNX/EIB de interface com o *gateway* IP/EIB e todas as bibliotecas necessárias por estas, foram compiladas com sucesso de acordo com o guia desenvolvido e apresentado como anexo deste relatório.

Após a fase de compilação ter sido efectuada sem grandes problemas, é na fase de execução da aplicação onde foram encontrados os problemas mais relevantes, tendo como consequência desta situação, o dispêndio de uma porção de tempo importante.

A execução do cliente KNX/EIB no sistema embebido gera um erro (*segmentation fault*), que faz com que toda a aplicação termine, sem efectuar qualquer acção no sistema.

Após um longo período testes, e com a cooperação do programador que desenvolveu o cliente, foi possível isolar o problema. O problema do *segmentation fault* era causado por uma biblioteca externa utilizada pelo cliente KNX/EIB, denominada de “*GNU Pth - The GNU Portable Threads* “. A compilação das rotinas de teste presentes nesta biblioteca e a sua execução na plataforma embebida puderam comprovar a fonte do problema: sempre que estas eram executadas, causavam o erro já descrito.

Após uma breve pesquisa pela *Internet*, pôde-se observar que este problema já não é novo, havendo algumas situações de *segmentation fault* com a biblioteca *GNU Pth* em sistemas equipados com o microprocessador AVR32.

Como ambos os clientes (*eibnetmux* e *eibd*) disponíveis para a integração com a aplicação CGI, capazes de estabelecer o túnel de comunicação sobre IP com o *gateway*, utilizam na

sua concepção a mesma biblioteca, *GNU Pth*, não foi possível a substituição de um cliente pelo outro, uma vez que o problema se iria manter.

Numa tentativa de perceber a origem do *segmentation fault* na execução do cliente KNX/EIB, e, juntamente com a colaboração do responsável deste e do responsável da *Atmel* pelo *port* do Linux para a plataforma AVR32, chegou-se à conclusão que o problema estaria no *software* do sistema embebido ICnova AP7000 (ao nível do *kernel* ou das bibliotecas presentes no S.O.). Até ao presente momento, não foi possível descobrir a exacta localização do problema, bem como a sua resolução.

De modo a poder continuar com o principal objectivo da segunda fase da implementação, foi utilizada a opção do *router* OEM, já atrás mencionada como uma alternativa possível. Disto resultou uma pequena variação da topologia do sistema de monitorização e controlo *Web* apresentado na secção 5.1.4. A nova topologia é apresentada na figura 5.15.

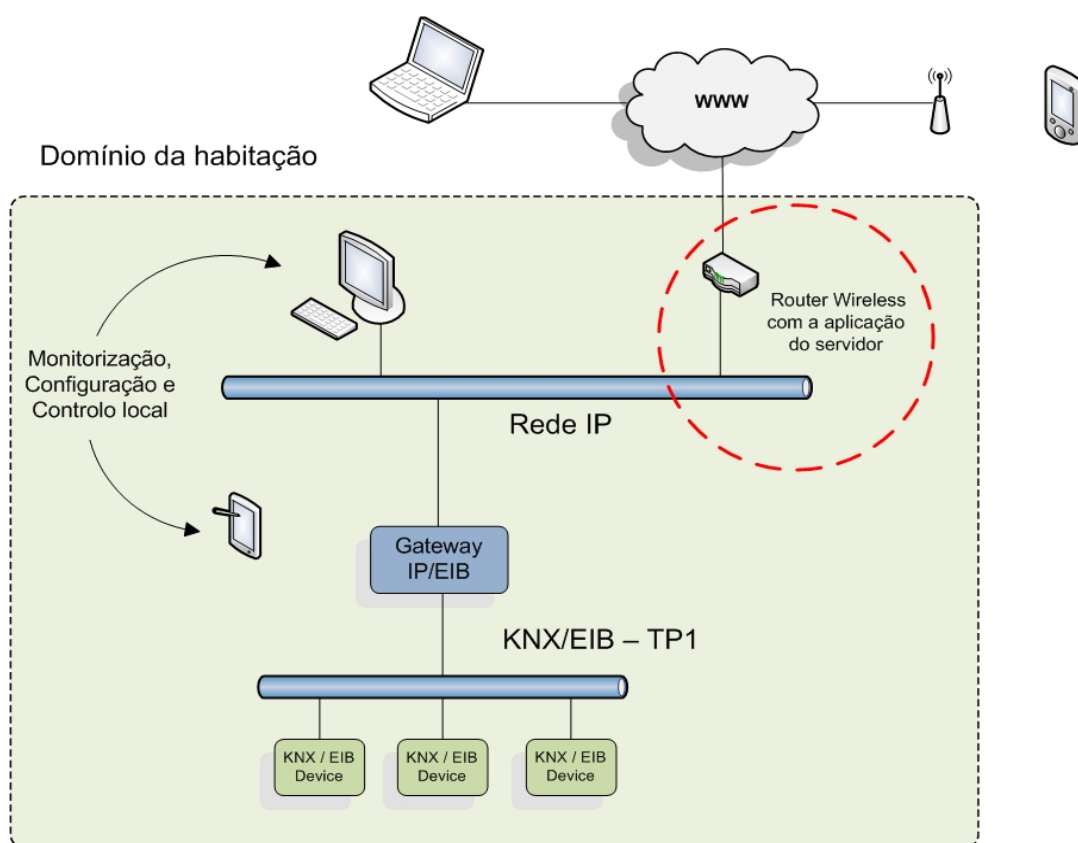


Figura 5.15 – Nova topologia do sistema de monitorização e controlo Web

A grande diferença nesta nova alternativa encontra-se ao nível da plataforma onde é executada a aplicação do servidor.

Após o estudo do *hardware* (nomeadamente o processador e a sua arquitectura interna) presente no *router* OEM já atrás mencionado (figura 5.3), um CPU Atheros AR231X

(MIPS), foi utilizado o *toolchain* disponível para esta plataforma, para recompilar a aplicação do servidor, tal como havia sido feito para a plataforma ICnova AP7000.

A compilação para esta plataforma da aplicação CGI, do cliente KNX/EIB e das bibliotecas requeridas por estes foi executada com sucesso, utilizando o guia já desenvolvido anteriormente.

Nesta plataforma, ao contrário da ICnova AP7000, a aplicação funcionou sem qualquer tipo de problema. A única limitação encontra-se no suporte à passagem de parâmetros pelos métodos GET ou POST das aplicações CGI, imposta por parte do servidor *Web*.

Para isso foi utilizado um outro servidor *Web*, o Boa *Webserver* [42]. Esta opção é utilizada numa grande variedade de sistemas embebidos devido à sua simplicidade, à sua eficácia e ao suporte total para aplicações CGI.

5.3 Testes e resultados

5.3.1 Primeira fase

Os objectivos da primeira fase de desenvolvimento - a implementação da estrutura de aplicação do servidor, tendo como plataforma de execução um PC, e do interface de controlo apresentado na secção 5.2.1.2 - foram completados com sucesso. A partir do interface *Web* desenvolvido, foi possível actuar em vários dispositivos do tipo EIS1, os actuadores binários ON/OFF (Siemens N561) das lâmpadas do painel didáctico EIB, de acordo com o exemplo definido para a validação da arquitectura implementada.

5.3.2 Segunda fase

A segunda fase tinha como objectivo o *port* da aplicação desenvolvida na etapa anterior, bem como todos os componentes necessários à execução da mesma, tendo como destino o sistema embebido seleccionado para o efeito.

Apesar da escolha inicial ter recaído sobre o sistema embebido ICnova AP7000, os problemas encontrados ao nível do software desta plataforma não permitiram avançar com esta para a solução final. Para continuar com o processo de validação da arquitectura proposta, foi escolhida a opção de utilizar um *router* OEM como plataforma de suporte para aplicação desenvolvida.

Os novos problemas encontrados com passagem para o novo sistema foram resolvidos e toda a aplicação ficou, agora, num estado funcional. Deste modo, a solução final que permitiria validar a arquitectura funcional proposta estava concluída.

Para a validação final do sistema foi testado, mais uma vez, o controlo dos vários dispositivos EIS 1 do painel didáctico EIB, via interface *Web*. As acções sobre este sistema de domótica foram efectuadas, com sucesso, através de um *browser* de um PC e de um *browser* de um PDA. As interacções assíncronas implementadas em Ajax entre cliente – servidor funcionaram com o comportamento desejado, também, nestas duas plataformas clientes distintas.

Capítulo 6

6. Conclusão e Perspectivas de Desenvolvimento

Nesta dissertação apresentou-se o projecto e a implementação de um sistema de supervisão, monitorização e controlo remoto via *Web* de instalações de domótica respeitantes com *standard* aberto KNX/EIB.

Este sistema tinha como objectivos essenciais: o uso de uma plataforma de hardware com um baixo custo de aquisição, um baixo consumo eléctrico (de modo a estar sempre ligada) e desprovida de *software* proprietário (execução e desenvolvimento), capaz de servir como base para a estrutura de *software* a desenvolver. Esta estrutura de *software* teria de permitir o acesso aos dispositivos da instalação de domótica, através de qualquer dispositivo com um *browser* e acesso à *Internet*.

Neste trabalho, foi feito um estudo sobre as diversas tecnologias *Web* disponíveis, capazes de suportar a estrutura de software e os objectivos já atrás mencionados. Como resultado do estudo efectuado, foi possível conceber três possíveis abordagens para a concepção da arquitectura funcional, recorrendo a diferentes tecnologias. Após serem discutidas as considerações sobre vantagens e desvantagens de cada uma das três propostas, foi eleita a solução mais conveniente para a arquitectura funcional do sistema.

A fase seguinte foi a implementação prática de um exemplo simples, de modo a poder validar a arquitectura escolhida. Nesta fase, numa etapa inicial, foi concebido um *interface Web* (baseado em HTML, JavaScript e Ajax de modo a poder actuar num dispositivo do tipo EIS1) e uma aplicação CGI compatível com esse *interface*. Esta aplicação teve, nesta etapa, como plataforma de execução, um PC (de modo a facilitar o desenvolvimento e o *debug* de possíveis erros). Esta etapa foi concluída com sucesso, funcionando como o previsto, isto é, acendendo uma lâmpada no painel didáctico EIB através de um *interface Web*.

A segunda etapa desta fase, tinha por objectivo o *port* da aplicação *Web* de controlo, desenvolvida com sucesso na primeira etapa, para a execução no sistema embebido

seleccionado (ICnova AP7000). Nesta etapa, foi efectuado um estudo sobre o processo de *cross-compiling* que permitiu recompilar a aplicação CGI, o cliente KNX/EIB e respectivas bibliotecas necessárias. Foi concebido uma guia de compilação de toda a aplicação do servidor, para o sistema embebido, estando como anexo deste documento.

Nesta etapa, foram verificados os primeiros problemas.

Apesar das compilações de todos os componentes da aplicação do servidor terem sido efectuadas com sucesso, a execução de uma biblioteca externa (*GNU Pth*) da qual toda a aplicação depende, gera um erro (*segfault*), que não foi possível solucionar até ao presente momento. Com a cooperação do responsável pela API cliente KNX/EIB e do responsável pelo *port* do Linux para a plataforma AVR32 (CPU da ICnova AP7000), pôde-se concluir que existe um *bug* (sem solução no tempo útil para este projecto) ao nível do *software* do sistema embebido escolhido para esta aplicação (ICnova AP7000).

De modo a prosseguir com o processo de validação, foi utilizada outra opção no que diz respeito à plataforma de *hardware* do servidor.

Como a grande maioria do equipamento de gestão de redes é baseado em Linux, e, numa instalação onde exista um acesso contínuo à Internet, a probabilidade de existir um *router* é elevada, surge a hipótese da utilização deste como plataforma de *hardware* de suporte para a estrutura do *software* do servidor.

Para a validação da arquitectura escolhida com esta nova plataforma, procedeu-se a uma análise do *hardware* envolvido, no interior do *router* OEM disponível para o efeito. Após o conhecimento do fabricante, do modelo e da arquitectura interna do CPU, procedeu-se à recompilação de todo o *software* (aplicação CGI, cliente KNX/EIB e respectivas bibliotecas) recorrendo a um *toolchain* para esta plataforma específica.

Ao contrário do sistema embebido seleccionado para o efeito, no *router* não houve nenhum problema na execução das aplicações e respectivas bibliotecas. A única limitação foi o não suporte à passagem de parâmetros para o CGI, através dos métodos GET ou POST. De modo a colmatar esta falha foi utilizado um outro servidor *Web*, o *Boa Webserver*, que oferece uma grande eficácia, boa simplicidade de utilização e configuração, e ainda um bom suporte para aplicações CGI.

Resolvida esta falha, toda a aplicação do servidor desenvolvida ficou operacional, tendo ficado completa a implementação para validação da arquitectura funcional proposta.

Esta solução final funcionou de acordo com o que era previsto para o exemplo de validação do sistema, com a interacção entre cliente e servidor a ser efectuada via Ajax e agindo correctamente sobre os actuadores do tipo EIS 1 no painel didáctico EIB disponível para este projecto.

Pôde-se concluir que a nova abordagem e solução final da arquitectura proposta nesta dissertação apresentou grandes benefícios quando comparada com a abordagem prevista

inicialmente. Destaca-se a melhor eficiência energética (uma vez que não é necessário um sistema embebido dedicado sempre ligado), a inexistência do custo da aquisição deste (uma vez que toda a aplicação é executada no *router* como um serviço extra) e a liberdade oferecida para o controlo local (edifícios ou habitações) através de plataformas móveis (PDAs, *Smartphones*, etc), proporcionada pela capacidade de rede sem fios da plataforma escolhida, o *router wireless*.

Com a implementação da plataforma base do servidor, tanto ao nível do *hardware* como ao nível da arquitectura do *software*, espera-se como perspectiva de desenvolvimento futuro, que estejam criadas as condições necessárias ao desenvolvimento de uma aplicação completa e funcional de monitorização e controlo via *Web*.

Referências Bibliográficas

- [1] Alves, José, “*Casas Inteligentes*”, Centro Atlântico, 2003
- [2] <http://www.electronica-pt.com/index.php/content/view/67/43/> - Acedido em 11/Junho/2008
- [3] <http://acasainteligente.blog.com/> - Acedido em 11/Junho/2008
- [4] <http://www.casadomo.com/> - Acedido em 3/Junho/2008
- [5] “*KNX: The World’s Only Open Standard for Home and Building Control*” - www.knx.org
- [6] KNX Handbook Volume 3: System Specifications Chapter 1: Architecture
- [7] KNX HANDBOOK, KNX Specification, Konnex Association, 2004
- [8] Palma, Diana, “*FEUP KNX-Domótica KNX/EIB de Baixo Custo*”, Março 2008
- [9] Communication Medium - IR – EIBA Handbook Series, Release 3.0
- [10] Monteiro, José, “*Montagem de um sistema didáctico utilizando a tecnologia Instabus/EIB da Siemens*”, Dezembro 2004
- [11] Santos, Domingos, “*Sistemas e Planeamento Industrial*”, Setembro 2007
- [12] EIBA Handbook Series, Release 3.0, Volume 1, Part 2, Abril 1999
- [13] <http://ezinearticles.com/?JavaScript-for-Web-Design---Advantages-and-nDisadvantages&id=645013> – Acedido em 15/Junho/2008
- [14] <http://www.expertrating.com/courseware/JavaScriptCourse/JavaScript-Introduction-1.asp>
Acedido em 15/Junho/2008
- [15] Holzner, Steven, “*Ajax Bible*”, John Wiley & Sons, 2007
- [16] <http://www.adaptivepath.com/ideas/essays/archives/000385.php> - Acedido em 16/Junho/2008
- [17] www.jquery.com - Acedido em 16/Junho/2008
- [18] <http://hoohoo.ncsa.uiuc.edu/cgi/intro.html> - Acedido em 17/Junho/2008
- [19] Gundavaram, Shishir, “*CGI Programming on the World Wide Web*”, Março 1996.
- [20] Coelho, Pedro, “*Programação em Java 2*” FCA, 2003
- [21] <http://doc.ece.uci.edu/~klefstad/s/180a/app.html> - Acedido em 20/Junho/2008
- [22] David Powers, “*PHP Solutions: Dynamic Web Design Made Easy*”, Apress, 2006

- [23] www.php.net - Acedido em 21/Junho/2008
- [24] http://www.upto11.net/generic_wiki.php?q=embedded_system - Acedido em 21/Junho/2008
- [25] “*Using Linux in Embedded Systems and Smart Devices*” - www.linuxdevices.com - Acedido em 23/Junho/2008
- [26] www.picotux.com - Acedido em 23/Junho/2008
- [27] <http://www.scratchbox.org/documentation/general/tutorials/introduction.html>
- [28] http://www.ic-board.de/product_info.php?info=p75_ICnova-AP7000-Base.html
- [29] *Federal Standard 1037C* - <http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm> - Acedido em 26/Junho/2008
- [30] “*Siemens N148/21 Operating and mounting instructions*”, Janeiro 2007
- [31] <http://sourceforge.net/projects/eibnetmux/> - Acedido em 2/Maio/2008
- [32] <http://sourceforge.net/projects/bcusdk/> - Acedido em 2/Maio/2008
- [33] WEINZIERL ENGINEERING GmbH, “*Implementation of KNX Standard*”, Outubro 2006
- [34] Carter, Alex “*EIB and Konnex succeed in bringing true building interoperability in accordance with EN50090*”, Agosto 2004
- [35] Flores, António, “*A criação de valor no binómio: casa inteligente / consumidor*”
- [36] <http://www.acasainteligente.com/sistemas.asp?idSist=5> - Acedido em 26/Junho/2008
- [37] <http://www.domotica-vivimat.com/index.php> - Acedido em 28/Junho/2008
- [38] <http://webserver.jgdomotica.com/> - Acedido em 28/Junho/2008
- [39] <http://www.jgdomotica.com> - Acedido em 29/Junho/2008
- [40] <http://cardio.pt/> - Acedido em 29/Junho/2008
- [41] <http://www.secant.ca/En/index.htm> - Acedido em 29/Junho/2008
- [42] Boa *Websserver* - <http://www.boa.org/>
- [43] Accton : Networking Equipments - <http://www.accton.com/index.htm>

ANEXO 1

Guia de *Cross-Compiling* do *eibd* para a arquitectura AVR32

O AVR32 GNU Toolchain é um conjunto de programas *standalone* usados para criar aplicações para os microprocessadores da família AVR32 da Atmel. Estas aplicações podem correr como aplicações embebidas ou em cima de um sistema operativo como o Linux.

Este *toolchain* é distribuído gratuitamente pela Atmel e está disponível no *site* da mesma, tanto para plataformas Linux como para plataformas Windows.

Instalação do AVR32 Toolchain no Ubuntu:

1. Descarregar o *toolchain* mais recente do site da [Atmel](#) para a distribuição de Linux que vai ser usada (neste caso Ubuntu, mas também existem para Fedora ou SUSE).
2. Confirmar se as versões mais recentes das seguintes bibliotecas, usadas pelo *toolchain* estão presentes no sistema. Se não estiverem instaladas, deverão ser instaladas (pode-se utilizar o gestor de pacotes da distribuição, neste caso o Synaptic).

Nome	Versão
libdwarf	20080208
xerces-c	2.8.0
libelf	0.8.9
libboost	1.33.1

3. Descomprimir o *toolchain* para um directório e instalar todos os pacotes (apesar de não serem todos necessários). Alguns pacotes dependem de outros que devem ser previamente instalados. Os pacotes podem ser instalados por várias ordens diferentes, como por exemplo a seguinte:

- libavrtools
- libavr32sim
- libavr32ocd
- libelfdwarf
- avrfwupgrade
- avr32-uclibc-kernelheaders
- avr32-uclibc-_0.9
- avr32trace
- avr32program
- avr32-binutils_2.17
- avr32-gcc
- avr32-gbd
- avr32gbdproxy
- avr32headers
- avr32parts
- avr32-linux-binutils
- avr32-linux-gcc
- avr32-linux-gdb

4. Descarregar os seguintes arquivos do site do [projecto](#) :

- pthsem-2.0.7.tar.gz
- argp-standalone-1.3.tar.gz
- bcusdk_0.0.3.tar.gz

5. Descomprimir o arquivo pthsem-2.0.7.tar.gz e aceder ao novo directório:

- tar -zxf pthsem-2.0.7.tar.gz
- cd pthsem-2.0.7

6. Copiar os ficheiros `config.guess` e `config.sub` do directório `ICnova_Base/package/gnuconfig` do CD para a pasta `pthsem-2.0.7` e substituir os existentes se tal for pedido.

7. Compilar e instalar a biblioteca `pthsem-2.0.7` para a arquitectura AVR32:
 - `./configure --host=avr32-linux --prefix=/usr/avr32-linux`
 - `make`
 - `make install`

8. Descomprimir o arquivo `argp-standalone-1.3.tar.gz` e aceder ao novo directório:
 - `tar -zxf argp-standalone-1.3.tar.gz`
 - `cd argp-standalone-1.3`

9. Copiar os ficheiros `config.guess` e `config.sub` do directório `ICnova_Base/package/gnuconfig` do CD para a pasta `argp-standalone-1.3` e substituir os existentes se tal for pedido.

10. Compilar e instalar a biblioteca `argp-standalone-1.3` para a arquitectura AVR32:
 - `./configure --host=avr32-linux --prefix=/usr/avr32-linux`
 - `make`
 - `make install`

11. Descomprimir o arquivo `bcusdk_0.0.3.tar.gz` e aceder ao novo directório:
 - `tar -zxf bcusdk_0.0.3.tar.gz`
 - `cd bcusdk_0.0.3`

12. Copiar os ficheiros `config.guess` e `config.sub` do directório `ICnova_Base/package/gnuconfig` do CD para a pasta `bcusdk_0.0.3` e substituir os existentes se tal for pedido.

13. Compilar e instalar o EIBD para a arquitectura AVR32:

- `./configure --host=avr32-linux --prefix=/usr/avr32-linux --with-pth=/usr/avr32-linux --without-pth-test --enable-onlyeibd --enable-eibnetiptunnel`
- `make`
- `make install`

14. Para criar o ficheiro executável do EIBD para colocar no sistema embebido, com todas as bibliotecas contidas nesse mesmo executável, aceder a `/bcusdk_0.0.3/eibd/server`:

- `rm eibd`
- `make`

15. Copiar o último comando executado no *make* anterior e adicionar `-static` e executar outra vez. O ficheiro executável `eibd` resultante, presente nesse directório é o que deverá ser copiado para plataforma AVR32.

16. Para compilar o CGI com o acesso ao EIBD:

- `avr32-linux-gcc -o cliente_eibd.cgi cgi.c common.c -leibclient`