INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

ISEL

Building Intelligence Open System (BIOS)

Fábio Ruivo Ferreira
(Licenciado)

Dissertação de natureza científica realizada para obtenção do grau de
Mestre em Engenharia Informática e de Computadores

Orientadores:
Professor Coordenador A. Luís Osório, ISEL - DEETC
Professor Coordenador João Calado, ISEL - DEM

Júri:
Presidente: Professor Coordenador Manuel Martins Barata, ISEL - DEETC
Vogais: Professor Coordenador José Manuel Igreja, ISEL - DEEA

Dezembro de 2010

Agradecimentos

Um agradecimento aos Professores A. Luís Osório e João Manuel Calado pela orientação, disponibilidade e apoio na realização desta dissertação.

Ao Grupo Sousa Pedro, pelas reuniões onde foram discutidas muitas ideias sobre o tema desta dissertação.

À OMRON e à Schneider, pelo *hardware* disponibilizado e também aos Engenheiros Nuno Veríssimo e João Neto, da OMRON e aos Engenheiros João Rodrigues e Fernando Ferreira, da Schneider pelo apoio na criação do demonstrador.

Ao ISEL – Instituto Superior de Engenharia de Lisboa. Em especial, ao DEETC – Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores e ao DEM – Departamento de Engenharia Mecânica pela disponibilização do espaço e dos meios electrónicos e informáticos necessários à realização desta dissertação.

Aos meus colegas do ISEL, pelo apoio e companheirismo demonstrado ao longo de todo o meu percurso académico e pelas longas horas passadas nos laboratórios a discutir soluções, foram sem dúvida de grande valor para o meu sucesso académico.

Aos meus amigos da Residência de Estudantes Maria Beatriz, pelos longos anos de amizade, companheirismo e convívio, pelos bons momentos vividos que para sempre marcarão a minha vida e a minha forma de ser.

À minha família, por todo o apoio dado ao longo destes anos. Em especial à minha mãe, um agradecimento do fundo do coração, porque sem o seu apoio incondicional não seria possível chegar onde cheguei.

A todos, que de uma forma directa ou indirecta, me ajudaram a concretizar esta dissertação e me ajudaram nos momentos mais difíceis do meu percurso académico.

Resumo

Os edifícios estão a ser construídos com um número crescente de sistemas de automação e controlo não integrados entre si. Esta falta de integração resulta num caos tecnológico, o que cria dificuldades nas três fases da vida de um edifício, a fase de estudo, a de implementação e a de exploração.

O desenvolvimento de *Building Automation System* (BAS) tem como objectivo assegurar condições de conforto, segurança e economia de energia. Em edifícios de grandes dimensões a energia pode representar uma percentagem significativa da factura energética anual. Um BAS integrado deverá contribuir para uma diminuição significativa dos custos de desenvolvimento, instalação e gestão do edifício, o que pode também contribuir para a redução de CO2.

O objectivo da arquitectura proposta é contribuir para uma estratégia de integração que permita a gestão integrada dos diversos subsistemas do edifício (e.g. aquecimento, ventilação e ar condicionado (AVAC), iluminação, segurança, etc.). Para realizar este controlo integrado é necessário estabelecer uma estratégia de cooperação entre os subsistemas envolvidos. Um dos desafios para desenvolver um BAS com estas características consistirá em estabelecer a interoperabilidade entre os subsistemas como um dos principais objectivos a alcançar, dado que o fornecimento dos referidos subsistemas assenta normalmente numa filosofia multi-fornecedor, sendo desenvolvidos usando tecnologias heterogéneas.

Desta forma, o presente trabalho consistiu no desenvolvimento de uma plataforma que se designou por *Building Intelligence Open System* (BIOS). Na implementação desta plataforma adoptou-se uma arquitectura orientada a serviços ou *Service Oriented Achitecture* (SOA) constituída por quatro elementos fundamentais: um bus cooperativo, denominado BIOSbus, implementado usando Jini e JavaSpaces, onde todos os serviços serão ligados, disponibilizando um mecanismo de descoberta e um mecanismo que notifica as entidades interessadas sobre alterações do estado de determinado componente; serviços de comunicação que asseguram a abstracção do *hardware* utilizado da automatização das diversas funcionalidades do edifício; serviços de abstracção de subsistemas no acesso ao bus; clientes, este podem ser nomeadamente uma interface gráfica onde é possível fazer a gestão integrada do edifício, cliente de

coordenação que oferece a interoperabilidade entre subsistemas e os serviços de gestão energética que possibilita a activação de algoritmos de gestão racional de energia eléctrica.

Palavras – chave: BAS, interoperabilidade, subsistemas de um edifício, SOA

Abstract

Buildings are being constructed with a growing number of automatisms that are not integrated between them. This miss of integration results in a technological chaos, which creates difficulties in the three phases of a building life cycle, which are study, assembling and management of a building.

The development of building automation systems (BAS) is strategic to answer comfort, security and energy saving requirements. In large buildings energy can represent a significant percentage of the year's energetic bill. An integrated BAS is expected to contribute for a significant decreased of costs, development, assembling and management cost, and it might contribute to a reduction of CO_2.

The goal of the proposed architecture is to create an integration strategy aiming to establish an integrated control strategy of the building sub-systems (e.g. heating, ventilation and air-conditioning (HVAC), lightning, security, etc.). To accomplish this integrated control is needed the establishment of a communication strategy among the involved sub-systems. One of the challenges to develop such a BAS is on how to establish interoperability between sub-systems as a main goal, since the suppliers of the referred subsystems are normally based on multi-vendors philosophy, being developed using heterogeneous technologies.

Thus, the present work consisted in the development of a platform that was designated as Building Intelligence Open System (BIOS). In the implementation of such platform is adopted a Service Oriented Architecture (SOA) that is composed by four main elements: a cooperative bus, named BIOSbus, implemented using Jini and JavaSpaces, where all the services will be connected, providing a mechanism of discovery and a mechanism that notifies the interested entities about changes in certain component; communication services ensure the abstraction of the *hardware* used in the automation of the several functionalities of the building; sub-system services that abstracts each sub-system and make it usable in the bus; clients, this can be coordination services that offers the interoperability between sub-systems, energy management services or user-interface.

Keywords: BAS, interoperability, building sub-systems, SOA

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

BAS	Building Automation System
BIOS	Building Intelligence Open System
oBIX	Open Building Information Exchange
PLC	Programmable Logical Controlers
DDC	Direct/Distributed Digital Controls
SDK	*Software* Development Kit
API	Application Programming Interface
SOA	Service Oriented Achitecture
OPC	OLE for Process Control
OPC UA	Open Process Control Unified Architecture
OPC DA	OLE for Process Control Data Access
AVAC	Aquecimento, Ventilação e Ar Condicionado

1. Introdução

Os edifícios de serviços de grandes dimensões, como por exemplo hospitais, centro comerciais ou aeroportos, requerem uma gestão contínua das suas instalações. A pressão sobre a eficiência energética e a evolução tecnológica levou a que a gestão deste tipo de edifícios evoluísse para sistemas de gestão centralizada, visando assegurar uma gestão integrada dos seus subsistemas (iluminação, climatização, etc.) que conduza a reduções nos custos de desenvolvimento, de operação e evolução (Figura 1). Estes custos podem ser, por exemplo, redundância de *hardware*, energético, manutenção ou com pessoal para assegurar a supervisão da infra-estrutura.

Neste capítulo serão identificados os problemas que a gestão integrada de edifícios oferece, bem como a estratégia adoptada para os resolver.

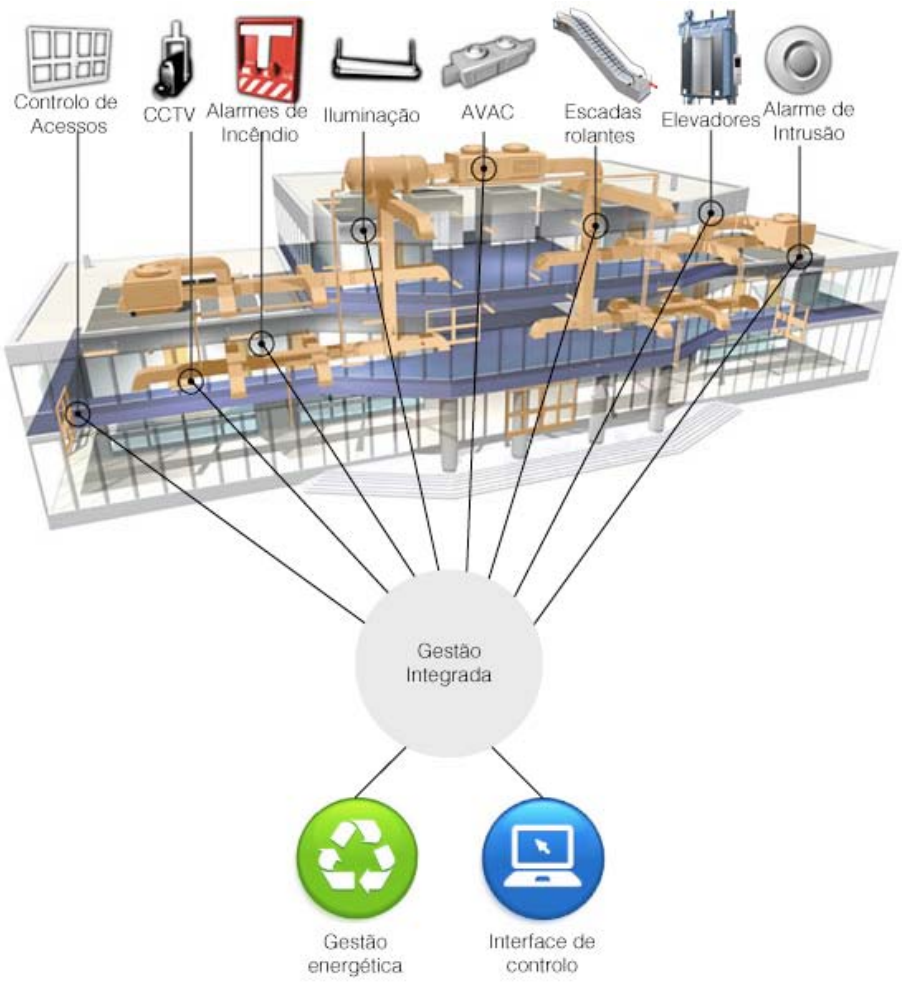


Figura 1 - Edifício e os seus diversos subsistemas

1.1 Problema

Grande parte dos produtos existentes no mercado, para a gestão e controlo dos diversos subsistemas, não disponibiliza um protocolo aberto que permita a cooperação, tornando-se assim difícil a concepção de sistemas que permitam uma operação/gestão integrada de subsistemas provenientes de diferentes fabricantes. Torna-se assim difícil a implementação de arquitecturas cuja concepção seja suportada por sistemas cooperativos, onde a interoperabilidade entre diferentes produtos, eventualmente oriundos de diferentes fabricantes, é um requisito que deverá ser satisfeito. É possível encontrar no mercado soluções completas que prometem a gestão integrada de todos os subsistemas do edifício, ou outras que são dedicadas apenas a alguma área específica, como por exemplo iluminação ou climatização. Todavia, quando se verifica a necessidade de dotar o edifício de uma nova funcionalidade, nomeadamente com recurso a equipamento de um fornecedor diferente daquele que se encontra instalado, deparamo-nos com uma tarefa de difícil execução ou mesmo impossível, dado que produtos de diferentes fornecedores usando tecnologia proprietária não conseguem inter-operar. É frequente encontrarmos associado a determinado subsistema informação que seria de fundamental importância para outro subsistemas, mas a dificuldade de integração resulta por exemplo na redundância de *hardware* ou no desperdício energético, o que se traduz em maiores custos de instalação, de gestão e de manutenção da globalidade da infra-estrutura.

O desenvolvimento de um BAS apresenta uma série de dificuldades, as quais podem ser subdivididas nas 3 fases da vida de um projecto: estudo, implementação e exploração. Na Figura 2 é apresentado um resumo dessas dificuldades.

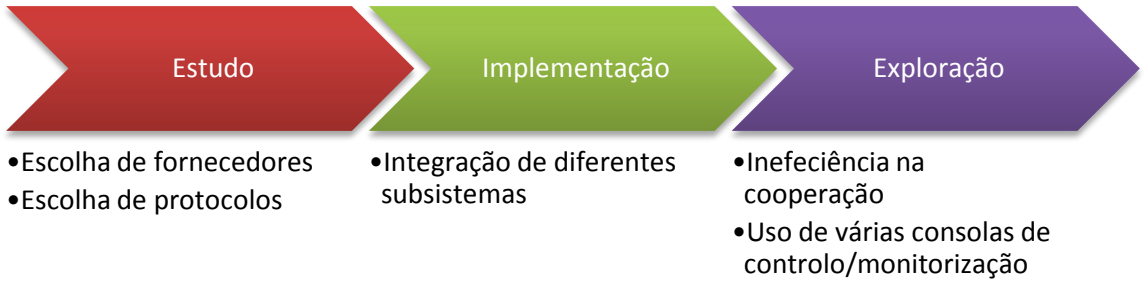


Figura 2 - Fases de um projecto e suas dificuldades

Na **fase de estudo** de um projecto é necessário efectuar a escolha dos fornecedores do equipamento que permite dotar o edifício de várias funcionalidades automatizadas. Todavia, esta escolha tem consequências que se reflectem durante toda vida útil do edifício, mesmo que os fabricantes afirmem que utilizam tecnologia aberta. Hoje em dia existem várias opções disponíveis no mercado e a sua escolha depende da aplicação e da topologia do edifício, o que dificulta a tarefa de selecção dos fabricantes.

Na **fase de implementação** de um projecto, há necessidade de interligar subsistemas de fabricantes diferentes. No entanto, se a escolha efectuada na fase anterior não foi bem ponderada, a opção tomada pode inviabilizar ou dificultar a cooperação entre dois subsistemas, nomeadamente se estiver em causa a utilização de protocolos distintos.

A **fase de exploração** de um projecto corresponde à sua implementação e funcionamento. É nesta fase que, como consequência de opções erradas em fases anteriores, se podem identificar eventuais ineficiências ao nível da cooperação entre subsistemas. Esta ineficiência poderá influenciar negativamente os custos de exploração da infra-estrutura, bem como a sua eficiência energética. Além que existem situações de falta de interoperabilidade que obriga ao uso de várias consolas de controlo. Não menos grave é a dificuldade de expansão de um sistema, pois fica dependente de muitos factores característicos do equipamento utilizado e não comuns a equipamento idêntico de outro fabricante, o que torna por vezes a expansão difícil ou eventualmente impossível.

Os protocolos abertos têm como objectivo dar resposta a este tipo de problema, mas estes ainda assentam em especificações de baixo nível. A definição destes protocolos tem origem em comunidades científicas e empresariais com culturas diversas, o que origina dificuldades acrescidas em todas as fases previamente mencionadas.

Este projecto visa a procura de uma estratégia tecnológica que permita assegurar a desejada interoperabilidade entre a diversidade de subsistemas utilizados na automatização de um edifício, tornando-o independente de fornecedores ou protocolos utilizados. Cada um destes subsistemas visa dotar o edifício de um conjunto de funcionalidades que pode ser potenciada através de uma gestão integrada.

Um exemplo do valor acrescentado associado a um edifício inteligente, é o facto de estarem reunidas as condições para alcançar uma eficiente gestão de energia. É comum

que edifícios de grandes dimensões tenham uma potência eléctrica contratada com o fornecedor de energia eléctrica, de tal forma que, se o valor contratado for ultrapassado (p. e. em apenas um único intervalo de 15m), o máximo valor atingido constituirá o novo valor de potência eléctrica contratada, provocando uma indexação do tarifário a um escalão superior. Um dos momentos mais críticos é o início da manhã, com o arranque simultâneo de diversos equipamentos (elevadores, climatização, equipamentos industriais, etc.), verificando-se valores elevados na potência tomada pelo edifício e consequentemente valores elevados no consumo de energia eléctrica. Recorrendo à gestão integrada do edifico inteligente será possível fazer uma gestão de energia por prioridades, através do cruzamento de dados dos diversos subsistemas. Desta forma é possível garantir que a potência eléctrica contratada não é ultrapassada. Uma possível estratégia para o conseguir seria, usar o sistema de CCTV, o sistema de controlo de acessos ou através de identificação de horas de ponta, como forma de identificar a chegada de um elevado número de pessoas ao edifício. Esta prioridade de escoamento rápido das pessoas, deverá resultar numa prioridade de fornecimento de energia ao sistema de elevadores e escadas rolantes enquanto é reduzida as necessidades energéticas do sistema de climatização. Esta técnica é denominada de deslastre de cargas.

Um sistema de gestão integrada em que os subsistemas do edifício inteligente colaborem e comuniquem de forma eficiente é vital para ter acesso a toda a informação do edifício, para por exemplo poder inclui-la no algoritmo de balanceamento de carga eléctrica. A interoperabilidade aqui é chave sendo que só desta forma se consegue uma gestão eficiente da energia eléctrica e de todos os demais recursos do edifício.

1.2 Motivação

Ao tema actual da preservação do meio ambiente, manifesta-se num conjunto de iniciativas que visam reduzir a emissão de gases com efeito de estufa, sendo as mais relevantes o protocolo de Quioto em vigor desde 2005 e a cimeira de Copenhaga em 2009. Por um lado, a eficiência energética é cada vez mais valorizada e, por outro, os edifícios (residenciais e serviços) são responsáveis por 38,7% de todo o consumo energético da Europa a 27 [1], tal como se pode verificar no gráfico presente na Figura

3. Através da optimização da gestão integrada do edifício será possível alcançar uma redução significativa dos correspondentes consumos de energia eléctrica.

O nível de automação nos edifícios tem aumentado constantemente ao longo dos anos [2]. Tal como abordado no capítulo anterior, tem que existir cooperação entre os diversos subsistemas de forma a potenciar a eficiência energética. A automação de determinado subsistema tem que estar preparada para a interagir de forma automática com os demais intervenientes de um edifício.

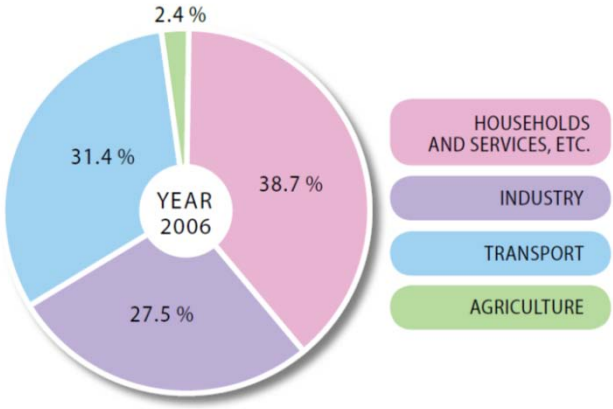


Figura 3 - Consumo energético da UE-27 por sector [1]

As actuais soluções que garantem a gestão integrada de um conjunto de subsistemas num edifício resultam de processos de integração especializados e geram soluções "*one of a kind*", de difícil reutilização em situações idênticas. Tais soluções fazem subir os custos de implementação e criam limitações ao nível do fabricante, fornecedor e/ou protocolo.

1.3 Estratégia

Para a resolução deste problema propõe-se uma solução de alto nível que se abstraia das camadas de automação. Adoptou-se uma filosofia SOA (*Service Oriented Architectures*), em que se prevê um bus lógico através do qual os diversos serviços comunicam. Esses serviços podem ser representantes de cada um dos subsistemas ou mesmo serviços proporcionando funcionalidades de gestão e controlo. Na Figura 4 encontra-se uma abordagem introdutória ao que será no futuro a lógica do sistema.

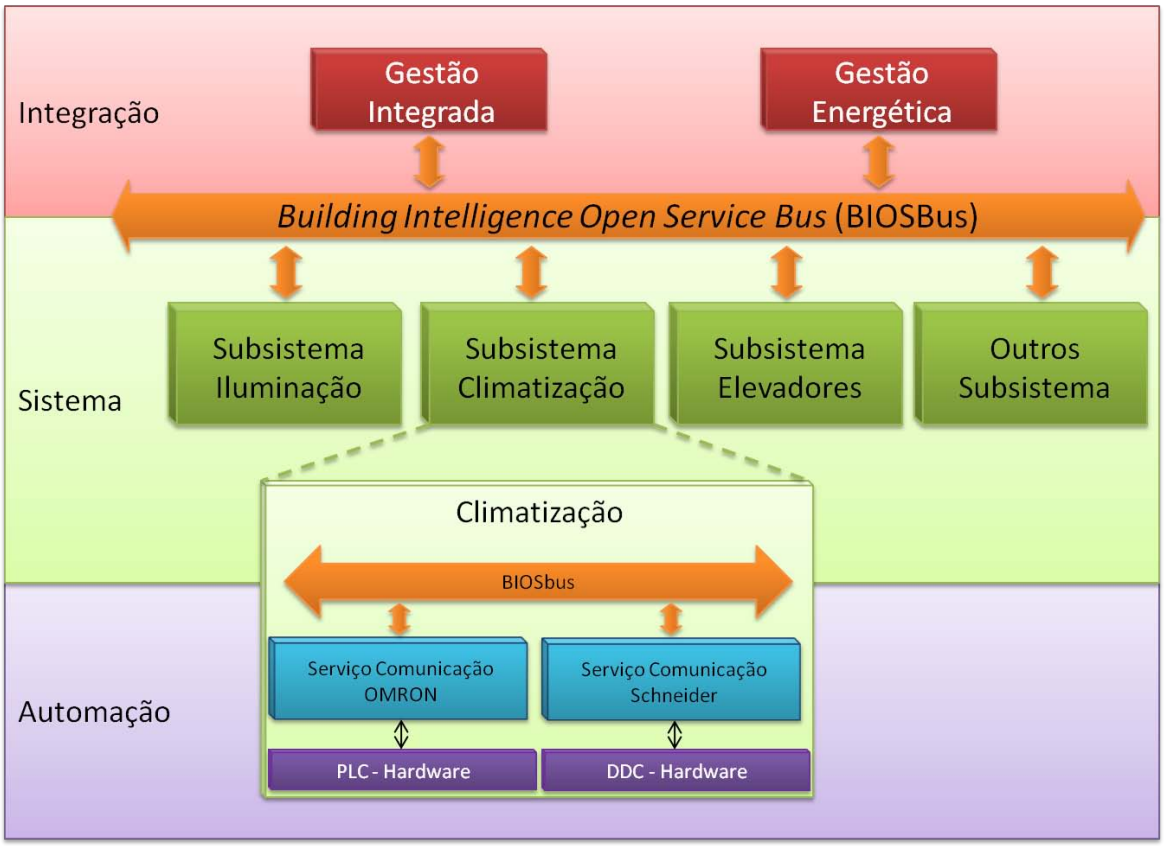


Figura 4 - Building Intelligence Open Service Bus (BIOSBus)

Todavia, o esquema na Figura 4 dá origem a uma questão fundamental, ou seja, como irá ser estabelecido um padrão para a infra-estrutura tecnológica de um edifício de modo a que processos de gestão, manutenção, administração e análise (*business intelligence* – BI), entre outros, possam ser desenvolvidos e associados de forma dinâmica de forma a que cada uma destas componentes de valor acrescentado consiga dialogar com qualquer outro sistema já instalado. A hipótese da necessidade de estabelecimento de uma linguagem de cooperação aberta, a ser implementada por uma diversidade de sistemas tecnológicos baseados em protocolos e linguagens especializadas, necessita de ser desenvolvida e validada através da experimentação sobre protótipos (demonstradores), envolvendo a diversidade de standards e tecnologias existentes. A proposta de um *Building Intelligence Open Service Bus* (BIOSBus) pretende incorporar uma associação ágil e flexível entre os processos associados a uma visão integrada de um edifício e a sua infra-estrutura tecnológica de suporte.

Este Bus deverá incorporar o padrão emergente SOA (*service oriented architecture*) como estratégia para o desenvolvimento e adopção de sistemas cooperativos em

edifícios, este tipo de sistema possibilita que, os recursos distribuídos pelos diversos subsistemas sejam livremente acedidos beneficiando a eficiência de todos os intervenientes do sistema. No essencial os sistemas cooperativos são desenvolvidos de "raiz" (ou adaptados) para a cooperação, sendo que o desenvolvimento de novos serviços (ou automatização de novos processos) não deverá requerer o envolvimento dos fornecedores dos sistemas existentes, com os quais o novo sistema/serviço terá que integrar. Esta perspectiva evolutiva para um edifício integrado, requer o desenvolvimento de uma infra-estrutura aberta, inteligente e capaz de associar, numa lógica "*Plug-and-play*", novos serviços (sistemas/aplicações) na automatização de novos processos.

1.4 Notação utilizada

Ao longo do texto desta dissertação surgem termos em inglês em situações em que sua tradução para português não reflecte o seu real significado, ou por serem termos que já são universalmente aceites. Tal situação acontece porque a documentação existente sobre este tema é, na sua maioria, publicada em língua inglesa e, sempre que possível, são utilizadas traduções que se consideram apropriadas ou que já se encontram enraizadas na língua portuguesa. Estes termos são apresentados em caracteres itálicos.

Para evitar a repetição de expressões técnicas longas, que poderiam tornar fastidiosa a leitura desta dissertação, são utilizados acrónimos ao longo do texto. A correspondência entre os termos técnicos e os seus respectivos acrónimos é feita no início deste documento, sendo explicado o seu significado na primeira ocorrência do respectivo acrónimo no texto.

Todas as referências bibliográficas utilizadas ao longo da dissertação são evocadas entre parêntesis recto e são apresentadas no final desta dissertação.

1.5 Estrutura da dissertação

Esta dissertação inicia-se com a análise do Estado da Arte. No capítulo 2 é efectuado o enquadramento do problema e são analisadas as tecnologias emergentes mais relevantes para a solução do problema.

Na segunda parte desta dissertação é proposta uma solução para o problema identificado. No capítulo 3 está descrita a arquitectura proposta, onde é efectuada uma primeira abordagem ao sistema BIOS. No capítulo 4 são apresentados os detalhes de

implementação, incluindo o desenvolvimento do demonstrador e concretização da arquitectura proposta.

2. Estado da Arte

Neste capítulo são reunidas as características e aspectos relevantes de inovação na área da gestão integrada de edifícios inteligente. Começa-se por enquadrar o problema descrevendo as características que definem um edifício e o que implica a sua gestão, identificando-se depois os desenvolvimentos tecnológicos para a área em questão.

2.1 Enquadramento do problema

De forma a ter consciência do que envolve a gestão integrada de um edifício de serviços, são levantadas neste capítulo as características mais relevantes, nomeadamente através da identificação e caracterização de cada um dos seus subsistemas e sempre que for relevante são apresentados exemplos onde existe mais valia na adopção de subsistemas cooperantes.

Para efectuar o controlo e monitorização de um edifício pode-se recorrer a dois tipos de sistemas de controlo, proprietário e aberto, estes serão analisados neste capítulo. No caso dos sistemas abertos são analisadas as mais valias, bem como a identificação de alguns protocolos abertos.

2.1.1 Subsistemas de um edifício inteligente

Neste subcapítulo são descritos os subsistemas mais relevantes e comuns num edifício. Os subsistemas abordados consideram a segurança (incluindo CCTV, alarmes de intrusão, alarmes internos e controlo de acessos), o controlo de iluminação, o controlo de climatização, os sistemas de controlo para interacção humana, elevadores e a gestão de energia eléctrica.

2.1.1.1 Segurança

2.1.1.1.1 CCTV

Um sistema de segurança de pessoas e bens é baseado num circuito fechado ou circuito interno de televisão (CCTV – do inglês *Closed-circuit Television*), um sistema de televisão que distribui sinais provenientes de câmaras localizadas em locais específicos,

para um ou mais pontos de visualização cujo objectivo mais comum é fornecer segurança através da vigilância.

Os sistemas CCTV encontram-se em evolução, quer em termos de tecnologia, quer em termos aplicacionais. Em termos tecnológicos, é hoje possível dispor o sistema em formato digital, usufruindo das mais-valias da era digital, como por exemplo a facilidade de armazenamento, bem como a qualidade de imagem. Em termos aplicacionais o circuito interno de televisão já não é apenas um sistema simples de monitorização visando apenas a segurança, tendo evoluído para áreas como o reconhecimento facial, reconhecimento de matrículas, entre outros. Na Figura 5 pode ser observado algum equipamento específico de sistemas CCTV.

Figura 5 - Sistema completo de CCTV[3]

2.1.1.1.2 Alarmes de intrusão

Existem inúmeros sensores cuja utilização visa a detecção de entrada de intrusos num determinado edifício, tais como sensores de abertura de janelas, vibração do chão, ou o mais comum, sensores de movimento, entre outros.

Existem diversas hipóteses de acções a desencadear em função dos sinais gerados pelos sensores acima mencionados, as quais dependem de configuração específica efectuada caso a caso. Se o intruso ainda se encontra no exterior do edifício, poderá acender-se toda a iluminação exterior, tanto para potenciar a gravação do sistema CCTV, como para tentar intimidar o intruso numa tentativa de o afugentar. Estando o intruso já dentro do edifício pode-se accionar um alarme silencioso, que faz imediatamente uma chamada para o proprietário/autoridades, alertando sobre o acontecimento. Durante essa chamada, também pode ser emitido o áudio da divisão em que o intruso se encontra [4].

No entanto, podem ser tomadas acções mais complexas, como por exemplo, tentativa de isolamento do intruso através do fecho automático de portas, corte de energia a elevadores, etc.

2.1.1.1.3 Alarmes internos (Incêndio, Inundação, Gás)

Cada edifício tem as suas necessidades, existindo inúmeros tipos de sensores possibilitando configurar um sistema para satisfazer determinado tipo de requisitos específicos, sendo os mais comuns dos sistemas de alarme, os de detecção de incêndios. Existem outros sistemas que têm por objectivo assegurar outras funcionalidades, como por exemplo, detecção de inundações e detecção de fugas de gás. [5]

Já existe tecnologia para detectar diferentes tipos de gás, podendo estes ser o comum gás da cozinha (Butano/Propano), CO_2, ou mesmo um outro tipo de gás pouco comum no dia-a-dia mas importante para uma empresa que lide com esse tipo de produtos, sendo assim possível detectar uma fuga atempadamente.

A Figura 6 ilustra um hipotético sistema de detecção de incêndios.

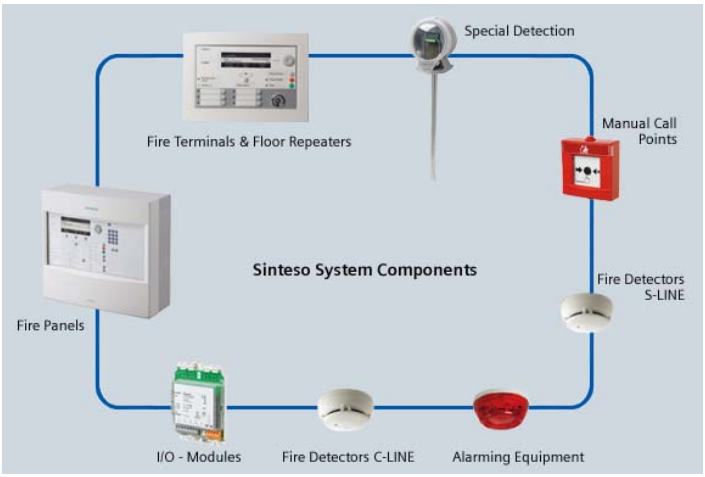


Figura 6 - Sistema de detecção de incêndio, Sinteso Siemens[6]

2.1.1.1.4 Controlo de acessos

Um sistema de controlo de acessos é um tipo de segurança física que permite o acesso a um edifício, divisão ou zona, apenas por pessoas autorizadas. O controlo é feito através de meios tecnológicos. Por exemplo, através de um cartão, código ou impressão digital pode ser identificado o ocupante que pretende aceder a determinada zona privada e conceder-lhe, ou não, a autorização.

É importante que este subsistema seja cooperante pois, por exemplo em caso de incêndio e dependendo de algumas excepções de segurança, o sistema deve permitir livre acesso para que o edifício ou zona possa ser evacuada com maior facilidade.

2.1.1.2 Controlo de Iluminação

Um sistema de controlo de iluminação é no geral caracterizado por um módulo central de processamento que comunica com os diversos interruptores, directamente com a própria iluminação ou com outra central, possibilitando uma gestão distribuída.

Este sistema permite controlar a iluminação de variadas formas, por exemplo através dos convencionais interruptores, ou a partir de um sistema central que possibilita o controlo por divisão ou mesmo de todo o edifício. É abstraída a forma convencional onde apenas os interruptores faziam o controlo de uma série de lâmpadas. Com um sistema de controlo de iluminação, esta pode ser controlada de qualquer interface. Essas interfaces são definidas no módulo central de processamento, podendo ser do mais variado tipo: interruptores, *dimmers*[1], controlo remoto, sensores, internet, etc. Com este subsistema automatizado proporciona ao ocupante economia e conforto, podendo ainda ser acedido pelo sistema de segurança, por exemplo, na resposta ao alarme de intrusão.

Através de sensores de presença evita-se consumos desnecessários de energia eléctrica, proporcionando ainda conforto. Através desses sensores pode-se activar a iluminação apenas quando existe pelo menos um ocupante presente. [7]

2.1.1.2.1 Controlo em tempo real

Ter a noção em tempo real dos consumos de energia eléctrica de um edifício é de fundamental importância para alcançar uma gestão racional de energia. Através do sistema de supervisão e controlo (onde podem estar incluídas acções como deslastre de cargas) podem ser verificadas informações desde o consumo de energia eléctrica de uma lâmpada até ao conjunto do edifício. Posto isto, fica facilitada a verificação se existem sectores do edifício com maiores consumos que outros, podendo ser detectadas possíveis anomalias.

Com o sistema de supervisão e controlo do consumo de energia eléctrica, será possível detectar e diagnosticar avarias de uma forma automática, facilitando e centralizando a gestão do edifício.

[1] ***Dimmer*** é um tipo de dispositivos utilizado para variar a intensidade de uma luz.

2.1.1.3 Controlo de Climatização

Um sistema AVAC (Aquecimento, Ventilação e Ar Condicionado) seja por necessidades funcionais do edifício, seja apenas para assegurar o conforto dos ocupantes, é hoje em dia valorizado e indispensável na concepção de um edifício.

A opção por um sistema com gestão centralizada, oferece um conjunto de potencialidades semelhantes ao controlo de iluminação. Por exemplo, através da identificação de rotinas efectuadas geralmente por determinado ocupante podem passar a serem executadas de forma automática. Deste subsistema resulta melhorias no conforto dos ocupantes e ainda a nível da gestão de energia (economia).

Em relação à gestão energética é importante fazer uma gestão ocupacional partindo da quantidade de pessoas presentes na divisão/edifício obtida através de sensores ou pelos identificadores de cada ocupante para uma melhor gestão dos recursos [8], ou considerando o cruzamento de informação da agenda de reuniões de determinada sala ou mesmo do sistema de controlo de acessos. Desta forma, o sistema de climatização será automaticamente ajustado ou antecipadamente accionado de modo a serem criadas as condições de conforto térmico desejadas pelos ocupantes.

2.1.1.4 Sistemas de Controlo

O controlo dos diversos subsistemas de um edifício pode ser efectuado de forma centralizada ou distribuída . A facilidade de interacção com o sistema de controlo poderá traduzir-se em conforto para os ocupantes. De seguida são enumeradas várias filosofias de controlo.

2.1.1.4.1 PC

Para edifícios com alguma dimensão é comum existir uma sala de controlo que tem geralmente mais que uma consola (PC) para monitorizar o edifício. Essas consolas podem utilizar *software* próprio ou basear-se em aplicações Web, por exemplo aplicações acessíveis através de um Web browser. Este controlo pode ser efectuado através da intranet ou mesmo internet.

Para um controlo mais focado, nomeadamente de determinado piso ou secção, existem sistemas mais compactos normalmente embutidos na parede onde, através de um ecrã táctil, é possível efectuar o controlo.

2.1.1.4.2 PDA

Pode-se ter o melhor de dois mundos com o PDA, pois este pode ter uma aplicação que se liga directamente ao sistema de supervisão e controlo do edifício simulando as funcionalidades do LCD, ou ter acesso a diversos sistemas de controlo através da internet.

2.1.1.5 Elevadores

Para um edifício que tenha uma rede de elevadores com alguma dimensão torna-se relevante controla-la a partir de um ponto central. Desta forma, será possível realizar remotamente um conjunto de operações como por exemplo bloquear um elevador, enviar um elevador para determinado piso, identificar em tempo real o estado de um elevador (e.g. em que piso se encontra, se existe avaria).

Este subsistema pode cooperar com outros subsistemas de diversas formas. Por exemplo, podemos ter interacção com o sistema de controlo de iluminação de tal forma que um pouco antes de serem abertas as portas do elevador podem ser ligadas as luzes do piso de chegada. Outro possível cooperação é com o sistema de controlo de acessos, podendo este dar ou não acesso a determinado piso.

2.1.1.6 Gestão Energética

Verificar as necessidades energéticas da divisão/edifício dependendo se estão presentes ocupantes ou até quantos estão presentes dá-se o nome de gestão ocupacional que é um tipo de gestão energética.

O sistema pode monitorizar toda a energia eléctrica que está a ser consumida em tempo real, ou ainda prever o que se vai consumir até ao final do mês/ano podendo ser detectadas atempadamente eventuais anomalias evitando assim consumos desnecessários de energia eléctrica.

A gestão de energia pode ser condicionada pela sazonalidade, pois as necessidades nos períodos de inverno não são as mesma que num período de verão. Por exemplo, em

certas empresas não existe a necessidade de ter águas quentes sanitárias durante a época de verão.

Na área dos edifícios a eficiência energética é, hoje em dia, uma das preocupações principais , daí que seja usual os seus componentes terem objectivos de supervisão dos consumos de energia conducente à sua redução.

2.1.2 Tipos de sistemas de controlo

Os sistemas de controlo podem ser divididos em dois tipos: os proprietários que são desenvolvidos pelos fornecedores/fabricantes e são geralmente fechados à cooperação; e os abertos que são na sua maioria baseados em normas promovidas por associações de empresas do ramo e são desenvolvidos com o intuito principal de possibilitar a cooperação, independência de fornecedor e a sustentabilidade à evolução do sistema.

2.1.2.1 Sistemas proprietários

Em geral, os sistemas proprietários exibem desempenhos aceitáveis para o fim que foram concebidos. Contudo, a empresa proprietária do sistema tem geralmente o monopólio dos algoritmos utilizados no controlo do sistema e do correspondente *software*, bem como, frequentemente é exigida a utilização de *hardware* de determinado fabricante específico.

Ao investir num sistema deste tipo não se está apenas a comprar um produto mas sim a criar uma parceria, que geralmente apenas favorece o fornecedor. Assim, torna-se impraticável, ou mesmo financeiramente difícil, depois de realizado um investimento inicial num sistema proprietário de determinado fornecedor, mudar para outro. Para além dos aspectos económico/financeiros, os sistemas proprietários revelam-se tendencialmente incompatíveis com *software*s e *hardware*s de outros fornecedores.

2.1.2.2 Sistemas Abertos (SA)

A utilização de tecnologias baseadas em SA vem alterar o paradigma de negócio da indústria de automação de edifícios, nomeadamente na relação entre o proprietário, o integrador de sistemas e o fabricante. Os proprietários têm a oportunidade de escolher produtos, tecnologias ou aplicações de fornecedores diferentes. Os integradores de sistemas e fabricantes podem utilizar produtos de terceiros ou tecnologias e aplicações desenvolvidas por si nos seus sistemas devendo seguir um conjunto de normas que caracterizam um sistema aberto. No âmbito do novo paradigma, o foco deixa de ser a tecnologia de suporte a determinado sistema e passa para as soluções, serviços e funcionalidades que os clientes necessitam. Ou seja, na implementação de determinado sistema/solução integrada, a preocupação deverá deixar de ser a tecnologia utilizada, passando esta para os processos a que esse sistema responde.

2.1.2.2.1 Porquê um SA

Quais as razões que levam a escolher um sistema aberto em vez de um proprietário? No que aos edifícios diz respeito, será o sistema aberto mais eficiente, proporcionando um melhor desempenho no controlo da instalação ou um melhor nível de conforto aos ocupantes? Não necessariamente, um sistema proprietário poderá apresentar níveis de desempenho tão bons ou melhores que uma solução aberta equivalente. Isso leva as pessoas a fazerem juízos errados tentando encontrar nas soluções abertas funcionalidades e desempenhos não disponíveis na solução proprietária equivalente. Todavia, existem inúmeras razões para que se escolha um sistema aberto em vez de um proprietário: oferta competitiva, instalação e manutenção consistente, integração de sistemas e interoperabilidade, aquisição de informação e permutabilidade (*interchangeability* / *Plug-and-play*). [9]

Oferta competitiva

Com sistemas proprietários só existe uma oferta competitiva no início de um projecto. Uma vez tomada a decisão inicial relativamente a determinada solução específica, passa a ser o fornecedor a ditar as “regras do jogo”. Por outro lado, quando se utilizam soluções abertas, que asseguram interoperabilidade entre *hardware* e *software* de diferentes fabricantes e uma integração do tipo *Plug-and-play*, verificamos a existência

de uma oferta competitiva no início do projecto, durante a sua exploração e até numa possível expansão.

Instalação consistente

Cada sistema proprietário tem as suas normas de instalação, bem como requisitos de expansão e fornecimento de produtos. Um SA tem normas de instalação consistentes, ou seja, utiliza normas bem conhecidos e documentados, sem que exista dependência do tipo de edifício ou dos fabricantes dos equipamentos que proporcionam uma gestão técnica centralizada do edifício dito inteligente.

Manutenção consistente

Uma instalação consistente resulta numa manutenção consistente, traduzindo-se numa redução de custos de exploração, dado que quem realiza a manutenção apenas necessita de conhecer a arquitectura e funcionalidades do SA, não estando as operações de manutenção na dependência de fabricantes ou produtos.

Integração de sistemas e interoperabilidade

As soluções abertas de gestão técnica de um edifício envolvem todos os subsistemas necessários para assegurar os requisitos de projecto no que se refere ao controlo da instalação, assegurando que os referidos subsistemas são capazes de comunicar entre si, sendo a sua exploração efectuada a partir de um ponto central, proporcionando-se assim as condições necessárias para assegurar a dita interoperabilidade.

Aquisição de informação

Através de um SA totalmente integrado no edifício consegue-se todo o tipo de informação em tempo real relativamente ao funcionamento da instalação. Essa informação pode ser utilizada por exemplo para o sistema de gestão de energia eléctrica, conseguindo-se assim um edifício energeticamente mais eficiente.

Permutabilidade entre produtos

Produzindo produtos de acordo com os standards/requisitos do SA, resulta em produtos estruturados para facilitar a permutabilidade entre produtos com funções semelhantes mas de fabricantes diferentes. Embora, mesmo nos sistemas ditos abertos esta seja uma matéria de elevada complexidade, constitui um paradigma difícil de abordar quando consideramos sistemas proprietários.

Uma outra abordagem relativamente ao porquê de escolher um sistema aberto será apresentada mais à frente no Capítulo 2.1.2.2.3, onde será efectuada uma análise de requisitos para um *Building Automation System* (BAS) sustentável.

As arquitecturas dos sistemas de gestão técnica centralizada de edifícios baseadas em soluções abertas parecem ser a solução para ultrapassar as dificuldades da automação de edifícios, conduzindo a soluções economicamente mais vantajosas quer do ponto de vista do investimento inicial, quer do ponto de vista dos custos de exploração (*total cost of ownership* – TCO). Todavia, embora existam tecnologias e soluções declaradamente abertas que no entanto são implementadas de forma proprietária. Por isso ao escolher uma tecnologia ou solução aberta, há que assegurar que ela suporta todos o requisitos referidos no Capítulo 2.1.2.2.2.

2.1.2.2.2 Requisitos

As soluções abertas são economicamente/financeiramente mais vantajosas que as soluções proprietárias. Mesmo nas situações em que o investimento inicial é semelhante, as soluções abertas apresentam custos de exploração e manutenção tendencialmente mais baixos. No domínio da gestão técnica centralizada de um edifício dito inteligente, a opção por uma solução baseada em tecnologias abertas, cria a expectativa de um sistema que faço o controlo total do edifício (*End-to-End*) e não uma solução que apenas assegura o controlo de alguns subsistemas. Para responder a estes requisitos, o sistema aberto tem que responder às necessidades funcionais dos edifícios, combinar sistemas de fabricantes diferentes, proporcionando opções de escolha sobre os produtos e induzindo requisitos para a competitividade ao nível do preço. São estes os objectivos a alcançar com a implementação de uma solução aberta, os quais se podem enumerar da seguinte forma: aberto, interoperável, multi-fornecedor e solução total [9].

Aberto

A tecnologia tem que estar disponível a qualquer fabricante para que este desenvolva produtos (*hardware* ou *software*) independentes ou para fazerem parte de uma solução completa, assegurando de uma forma relativamente fácil a interoperabilidade com produtos de outros fornecedores.

Multi-fornecedor

As soluções de gestão técnica centralizada de edifícios inteligentes, têm que ter a possibilidade de ser produzidas incorporando produtos de vários fornecedores, sem que seja necessário recorrer a *gateways*[2], ou a *software* ou ferramentas de rede, específicos do fabricante.

Interoperável

Capacidade de cooperar de forma transparente com outro sistema heterogéneo podendo dessa forma partilhar informação ou mesmo fazer pedidos sobre componentes dos vários subsistemas. Desta forma, uma rede de subsistemas heterogéneos é transformada num só sistema mais robusto e com maior capacidade de resposta aos diversos pedidos dos utilizadores.

Solução total

Uma abordagem que tenha aplicação em todos os níveis de um edifício, desde sensores a *software* e em todos os subsistemas. Pretende-se assim, uma solução global e integrada para o sistema da gestão técnica do edifício, centralizada do ponto de vista de operação, que para além de assegurar todas as funcionalidades inerentes às especificações da automação do edifício, possuindo ainda funcionalidades de gestão e manutenção do próprio sistema, como por exemplo metodologias de tolerância a falhas da infra-estrutura física e eventualmente instrumentação de *software*.

Uma tecnologia para ser considerar aberta tem que responder a **todos** os requisitos enumerados anteriormente. Existem algumas tecnologias ditas abertas mas que não conseguem responder a todos os objectivos, sendo exemplo:

- Empresas que indicam que a sua tecnologia proprietária é aberta. Estas empresas fornecem o protocolo e a definição da arquitectura a terceiros, e até podem oferecer certificação. E assim sendo, a sua tecnologia está disponível a qualquer fabricante para que se possa desenvolver produtos compatíveis com essa tecnologia específica. No entanto, esta tecnologia não pode ser

[2] *Gateway* – É um dispositivo dedicado ou um software que faz a tradução entre protocolos permitindo que protocolos totalmente independentes comuniquem entre si. Mas desta forma cria-se dependências, pois se algum dos protocolos for alterado o gateway também tem que ser adaptado.

considerada aberta, dado que apenas a própria empresa controla a definição do protocolo.

- Seleccionar uma solução proprietária que garanta interoperabilidade entre AVAC e o sistema de segurança ou AVAC e o controlo de iluminação, não é considerado um sistema aberto, pois a sua interoperabilidade está limitada.
- Por exemplo a utilização do protocolo Modbus. Os fabricantes podem aplicar Modbus para vários produtos, e é possível utilizar produtos Modbus de vários fabricantes, mas isso não é suficiente para ser considerado sistema aberto, pois não garante uma solução completa.
- Praticamente todas as soluções proprietárias são soluções completas, mas como são específicas de um fabricante não podem ser consideradas soluções abertas.

Resumindo os pontos-chave, um sistema aberto tem que ser uma solução completa oferecida por vários fornecedores, que usa um tipo de comunicação que possibilite a interacção dos diversos dispositivos envolvidos.

2.1.2.2.3 Análise de requisitos para um Building Automation System (BAS) sustentável

Os requisitos para um sistema de gestão técnica centralizada que assegure a automação de edifícios sustentável, são [10]:

Continua disponibilidade da maioria dos componentes do sistema, seja *hardware* ou *software*, compatível com o sistema original

Idealmente, este requisito passa pela possibilidade de utilização de produtos de diferentes fabricantes. Um exemplo simples, é a obsolescência de determinada interface gráfica que tem que ser substituída ou simplesmente adoptada uma versão mais actualizada. Essa substituição tem que ser garantida, possivelmente necessitando também de um PC novo, sem que haja uma necessidade de reestruturação do sistema de gestão técnica centralizada.

Disponibilidade da informação

Para garantir a sustentabilidade do BAS é importante que os técnicos tenham acesso facilitado à informação e formação, seja para fazer uma implementação de “raiz”, ou mesmo num futuro em que seja necessário expandir ou fazer manutenção do sistema.

Ou seja, no caso de um sistema proprietário o número de técnicos formados nessa tecnologia é reduzido e verifica-se a existência de retracção das empresas em disponibilizar informações sobre os seus sistemas proprietários. Num sistema aberto existem entidades independentes que dão formação e disponibilizam informação sobre a arquitectura de forma facilitada. Desta forma, o sistema aberto dá mais garantia de uma contínua disponibilidade de informação.

Expansão do BAS

Os componentes mais recentes do BAS têm que ser compatíveis com os antigos (sistemas legados), de modo a que o sistema seja facilmente mantido e/ou expandido. Apesar de se poder utilizar gateways, esta não é a melhor solução já que requer um conhecimento especializado o que pode limitar a assistência técnica. A solução ideal é utilizar produtos cujos fornecedores/fabricantes asseguram que na sua evolução se mantém compatíveis para que a expansão do BAS acompanhe facilmente a evolução do edifício.

BAS evolutivo

O BAS tem que estar preparado para acompanhar os desenvolvimentos tecnológicos. Um exemplo pode ser a substituição de uma interface gráfica em aplicação PC, para uma em aplicação Web.

A solução: Sistema aberto

Uma solução que se considera viável e economicamente mais vantajosa, é o BAS ser um sistema concebido no âmbito deste paradigma emergente, ou seja, ser concebido com base em tecnologia aberta. Tendo em conta os vários factores já enumerados faz-se agora uma abordagem focando os requisitos de sustentabilidade a que estes conseguem responder:

Continua disponibilidade da maioria dos componentes do sistema

A única forma de assegurar que o sistema cumpre este requisito, passa pela possibilidade de não depender de apenas um fabricante. Todavia, para poder recorrer a vários fabricantes existe a necessidade de o sistema ser definido sobre standards de comunicação, aplicação e gestão. Para garantir coerência na definição e utilização desses standards é necessário criar uma associação, de modo a, por exemplo, certificar os produtos, garantido que o fabricante produz produtos que vão ser interoperáveis com

os restantes. Dois exemplos que serão abordados mais à frente são o LonWorks que é certificado pela *LonMark International Association* e o BACnet que é certificado pelos *BACnet Testing Laboratories*.

Disponibilidade da informação

Toda a informação relacionada com os standards dos BAS, protocolos, rede, algoritmos de controlo, documentação, etc., necessita estar disponível para todos. Depende da abordagem de cada associação, mas existem certificações ou mesmo formação em escolas, relativamente a estes standards (Sistemas Abertos).

Expansão do BAS

Como já existe garantia de interoperabilidade assegurada pelo SA, torna-se simples e directa a expansão do BAS. Assim, a rede e a interface gráfica antiga são usadas para comunicar e controlar os componentes novos.

Pode-se concluir que se consegue atingir um BAS sustentável através de uma escolha inicial cuidada e estudada, que tenha em conta as necessidades e que inclua o uso de protocolos e redes estandardizadas e um acesso facilitado aos produtos e informação (Sistema Aberto).

2.1.2.3 Protocolos Abertos

Um BAS tem como objectivo melhorar a eficiência e eficácia da gestão técnica centralizada de todo o equipamento que proporciona um conjunto de funcionalidades executadas de forma automática, que permitem classificar determinada instalação de inteligente. Estas funcionalidades são divididas em 3 níveis conforme representado na Figura 7. Nível de campo, onde a informação é recolhida (e.g. sensores) e são executadas as acções de controlo (e.g. actuadores). Nível de automação é onde são implementados e executados, por exemplo, os algoritmos de controlo, tolerância a falhas, instrumentação de *software*, etc. Nível de gestão será responsável pela configuração e gestão do sistema, bem como, pela sua exploração (e.g. monitorização).

De acordo com o Estado da Arte neste domínio do conhecimento, os Protocolos Abertos mais relevantes são BACnet, LonWorks e o KNX. São estes os protocolos que permitem a integração das funcionalidades intrínsecas aos três níveis de funcionalidades acima referidas. Todavia, existem outros que se especificam apenas para responder a um conjunto restrito de requisitos. Na Tabela 1 encontra-se uma lista com alguns protocolos utilizados na automação de edifícios. Como se pode verificar na Figura 7, o KNX e LonWorks são mais predominantes nos níveis de campo e de automação, enquanto o BACnet prevalece nos níveis de automação e gestão [11].

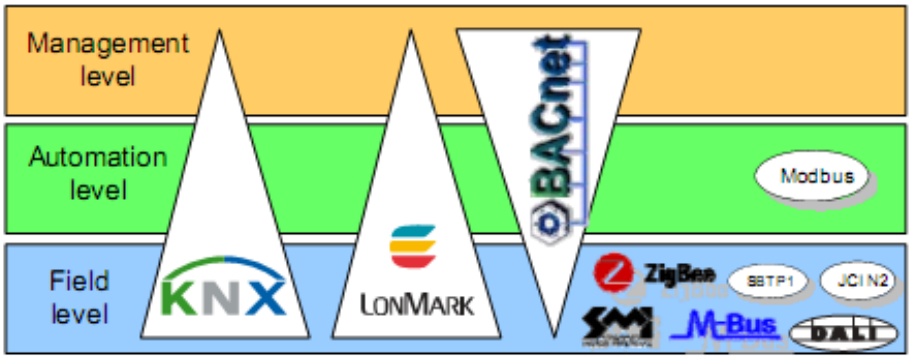


Figura 7 – Níveis de funcionalidades [11]

Neste relatório são analisados os três protocolos referidos anteriormente, KNX, LonWorks e BACnet, considerados os protocolos abertos com maior sucesso e visibilidade.

Protocolo	**Patrocinador**
BACnet	Building Automation Industry
LonTalk.	Echelon Corp.
KNX	Konnex
Modbus	Modicon
ZigBee	ZigBee Aliance
IBIbus	Intelligent Building Institute
CAB	Canada
X-10	X-10 Corp.
Smart House	Smart House L.P.
CEBus	EIA
Michigan Parallel Standard	University of Michigan
Integrated Smart-Sensor Bus Delft	University of Technology
Time-triggered protocol	University of Wien, Austria (Automotive)

Tabela 1 – Protocolos de automação [12]

2.2 Tecnologias em edifícios

Na primeira parte deste capítulo é apresentado o estudo sobre os três principais sistemas abertos usados pela indústria, BACnet, LonWorks e KNX, sendo enumeradas as características chave de cada um destes sistemas, permitindo efectuar a sua comparação. Deste estudo resultou um artigo científico que foi apresentado na edição de 2010 da *International Conference on Theoretical, Experimental and Applied Signal and Image Processing Techniques and Systems* (IWSSIP'10), realizada no Rio de Janeiro, Brasil.

Além deste estudo, são ainda analisados dois projectos com interesse para esta dissertação o *Open Building Information eXchange* (oBIX), e o *Open Process Control Unified Architecture* (OPC-UA).

2.2.1 BACnet

O BACnet (*Building Automation and Control Networking Protocol*) foi pensado para responder às necessidades de automação de um edifício, fosse ele de pequenas ou grandes dimensões.

O seu desenvolvimento iniciou-se em 1987 por parte da *American Society of Heating, Refrigerating and Air-Conditioning Engenieers* (ASHRAE), tendo este desenvolvimento ficado concluído apenas em 1995, quando a ASHRAE promoveu a publicação do BACnet como um standard ANSI (American National Standards Institute). Em 2003 foi adoptado pelo CEN (European Committee for Standardization) e pela ISO (International Organization for Standardization) o que implicou uma maior expansão na sua utilização, nomeadamente na Europa e Ásia. A versão mais recente do standard é o "*BACnet—A Data Communication Protocol for Building Automation and Control Networks*, ANSI/ASHRAE Std. 135, 2004" [13].

2.2.1.1 Entidades de desenvolvimento e suporte

2.2.1.1.1 Standing Standard Project Commitee (SSPC)

A evolução do BACnet é constante sendo mantida através do ASHRAE *Standing Standard Project Commitee* (SSPC), em cujo grupo estão representados sectores da indústria relacionada com a construção de edifícios. O processo de desenvolvimento é totalmente aberto, estando o grupo formado no âmbito do SSPC receptivo a comentários e sugestões.

2.2.1.1.2 BACnet Interest Groups (BIGs)

Os chamados BIGs são grupos interessados no BACnet, podendo ser empresas ou universidades. Já existe o BIG-EU (Europa), BIG-NA (América do norte), BIG-AA (Austrália e Ásia), entre outros. Estes grupos têm como objectivo divulgar e apoiar a utilização do BACnet, através de actividades promocionais, formação e trocas de experiências.

2.2.1.1.3 BACnet Testing Laboratories (BTL)

Um elemento muito importante para a manutenção da coerência dos produtos BACnet no mercado é o BTL, operado pela BMA (*BACnet Manufacturers Association*). É o BTL que testa e certifica os produtos, assegurando a coerência e consistência entre os produtos e as normas indicadas no standard.

2.2.1.2 Comunicação

Para que dois subsistemas de um edifício inteligente comuniquem entre si, existe a necessidade de uma troca de mensagens entre eles, em que para além do conteúdo da mensagem será necessário ter em conta a forma como é enviada. De modo a que dois tipos de subsistemas comuniquem entre si, foram definidos no standard seis protocolos que tratam essa comunicação, onde se inclui o Ethernet, o ARCNET, o Master-Slave/Token-Passing (MS/TP), o LonTalk, o Point-To-Point (PTP) e mais recentemente o Internet Protocol (IP). Cada um destes protocolos tem as suas características e utilidades específicas como podemos verificar a seguir.

Numa instalação que utiliza o standard BACnet, são utilizados vários protocolos de comunicação, estabelecendo-se diversas camadas no sistema. É assim possível efectuar a distinção, por exemplo, entre uma comunicação entre controladores de diferentes subsistemas e uma comunicação entre os componentes de campo, como os sensores e os actuadores. Consegue-se assim efectuar a distinção entre comunicações consideradas críticas e não críticas, ou seja, as primeiras necessitam de uma maior eficiência de comunicação enquanto as segundas apresentam requisitos de comunicação mais flexíveis. É assim possível reduzir custos sem comprometer desempenhos. Na Figura 8 encontra-se um exemplo de 3 tipos diferentes de tecnologias LAN interligadas através de routers.

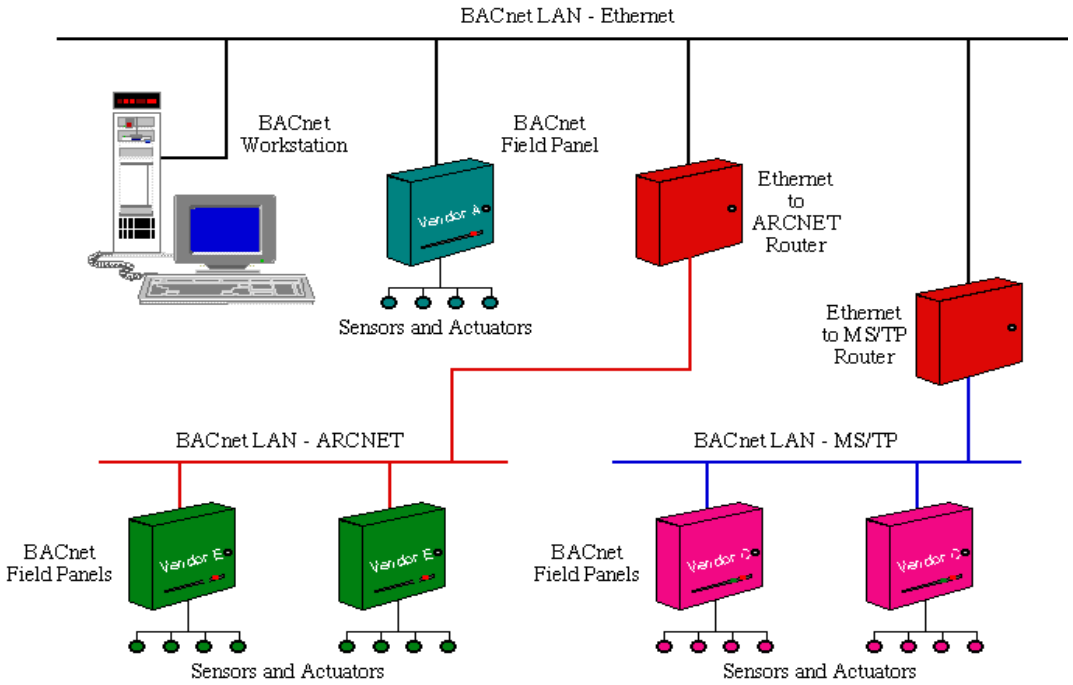


Figura 8 - Exemplo de três tipos diferentes de tecnologias LAN interligadas através de routers [14]

2.2.1.3 BACnet e Modelo OSI

O modelo OSI descreve por camadas a comunicação entre dois dispositivos de rede. Na sua forma original este modelo tem sete camadas, mas o BACnet apenas adaptou quatro delas, Física, Ligação, Rede e Aplicação, como se pode verificar na Tabela 2.

A camada mais baixa, a Física é a responsável por enviar a representação binária da informação, sendo esta normalmente implementada sobre *hardware* dedicado. A camada de Ligação define como os componentes são endereçados e de que forma a informação vai ser enviada entre eles. A camada de Rede além de fazer o *Routing* das mensagens permitindo que dois tipos diferentes da camada de Ligação (e.g. Ethernet e MS/TP) possam comunicar. A camada de Aplicação é onde se dá significado à informação, definindo os objectos, suas propriedades e serviços por eles disponibilizados (Capítulo 2.2.1.4).

<table>
<tr><th colspan="5">Camadas BACnet</th><th>Camadas Equivalentes OSI</th></tr>
<tr><td colspan="5">Camada de Aplicação BACnet</td><td>Aplicação</td></tr>
<tr><td colspan="5">Camada de Rede BACnet – BACnet/IP</td><td>Rede</td></tr>
<tr><td colspan="2">Ethernet</td><td>MS/TP</td><td>PTP</td><td rowspan="2">LonTalk</td><td rowspan="2">Ligação Física</td></tr>
<tr><td>Ethernet</td><td>ARCNET</td><td>EIA-485</td><td>EIA-232</td></tr>
</table>

Tabela 2 - Modelo de 4 camadas do BACnet [15]

2.2.1.3.1 Ethernet

A Ethernet é uma tecnologia de ligação para redes locais (LAN), estando definida como um standard internacional (ISO 8802-3), sendo das tecnologias LAN a mais utilizada hoje em dia. Esta tecnologia suporta vários tipos de meios físicos, nomeadamente cabos UTP, fibra óptica e tecnologia *wireless*. Consegue velocidades mais elevadas actualmente 10 Gbps mas podendo ir a velocidades superiores, o que a torna especialmente adaptada para ser utilizada na comunicação entre sistemas ou interfaces de controlo.

2.2.1.3.2 MS/TP

MS/TP é uma tecnologia de muito baixo custo, mas também de baixa velocidade, entre os 9.6Kbps e os 76Kbps. Esta tecnologia é um standard ANSI e está especificada na norma EIA-485, utilizando como meio físico o RS-485. Pode ser implementada num só chip, o que aliado ao seu baixo custo faz desta tecnologia um protocolo de eleição para efectuar a comunicação entre componentes de campo.

2.2.1.3.3 BACnet/IP

Devido ao uso em massa das redes IP foi necessário redefinir em 1999 o Standard de modo a que este suportasse o IP. Este está totalmente integrado na camada de Rede, o que permite que a comunicação entre componentes BACnet seja efectuada sobre IP. Posto isto, sabendo o endereço IP do componente com quem se quer comunicar, é possível utilizar a internet como meio de comunicação entre componentes, como se pode verificar na Figura 9.

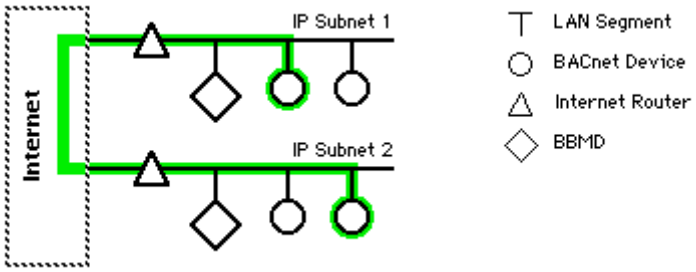


Figura 9 - Exemplo de uma ligação IP [16]

2.2.1.3.4 BACnet/WS (Web Services)

Através de uma série de serviços (Capítulo 2.2.1.4.3) disponibilizados sobre a rede e com o acesso por um browser de internet, é possível evitar assim a necessidade de uma aplicação específica para a gestão técnica centralizada de um edifício inteligente. Desta forma, as tarefas inerentes à gestão técnica centralizada de um edifício inteligente, podem ser efectuadas através dos *Web Services* a partir de qualquer posto de trabalho da rede (internet ou intranet).

2.2.1.4 Representação e gestão dos dispositivos

2.2.1.4.1 Objectos

O BACnet disponibiliza 18 tipos de objectos Standard para representação e gestão de dispositivos,

Tabela 3. Toda a informação no BACnet é representada através de objectos. Um objecto pode representar um input/output físico, ou um grupo lógico de pontos que representam algum tipo de funcionalidade. Todos os objectos são identificados e contêm determinadas propriedades associadas, sendo apenas através destas que o objecto é gerido e controlado [17].

Por exemplo um objecto correspondente a um sensor de temperatura de um sistema AVAC, além da temperatura actual, pode também conter outras informações úteis, por exemplo em que unidade está a ser medida a temperatura (e.g. Celsius), ou a descrição do local em que está a ser realizada a medida.

Accumulator
Analog Input
Analog Output
Analog Value
Averaging
Binary Input
Binary Output
Binary Value
Calendar
Command
Device
Event Enrollment
File
Group
Life Safety Point
Life Safety Zone
Loop
Multi-state Input
Multi-state Output
Multi-state Value
Notification Class
Program
Pulse Converter
Schedule
Trend Log

Tabela 3 - Objectos BACnet [18]

2.2.1.4.2 Propriedades

O standard BACnet identifica 123 propriedades para caracterização dos objectos, existindo 3 propriedades que têm de ser definidas em todos os objectos; *Object-identifier*, *Object-name*, e *Object-type*. Um subgrupo de propriedades é específico para cada tipo de objecto, sendo algumas de especificação obrigatória e outras opcionais [19]. Na Tabela 4, estão presentes as propriedades obrigatórias de um *Device Object*. Algumas propriedades são necessariamente implementadas seguindo as especificações, mas outras podem ser definidas pelo fabricante. Todas estas propriedades podem ser lidas na rede, podendo ou não ser escritas, dependendo da definição da propriedade.

BACnet permite que os fabricantes adicionem propriedades ou eventualmente objectos. Todavia, propriedades proprietárias podem resultar em falhas na interoperabilidade.

Property Identifier	Value
Object_Identifier	(Device, Instance 2749)
Object_Name	"RE1 Penthouse"
Object_Type	DEVICE
System_Status	(OPERATIONAL)
Vendor_Name	"Contemporary Controls"
Vendor_Identifier	245
Model_Name	"BASR-8M"
Firmware_Revision	"1.0"
Application_Software_Version	"1.0"
Protocol_Version	2
Protocol_Revision	
Protocol_Services_Supported	(List of services)
Protocol_Object_Types_Supported	(List of object types)
Object_List	(List of all the objects)
Max_APDU_Length_Accepted	1476
Segmentation_Supported	(NO SEGMENT)
APDU_Timeout	(3000 MSEC)
Number_Of_APDU_Retries	0
Device_Address_Binding	
Database_Revision	1

Tabela 4 - Propriedades obrigatórias de um Device Object [18]

2.2.1.4.3 Serviços

Os serviços são o mecanismo que o BAS usa para aceder às propriedades e efectuar acções sobre os objectos BACnet. É a forma como: um dispositivo BACnet pede informação a outro dispositivo; executa acções sobre outros dispositivos, através das propriedades; comunica eventos para outros objectos. O único serviço obrigatório para qualquer dispositivo é o *Read-property*. Existem no total 32 serviços Standard.

Para o utilizador ou instalador, a implementação do serviço é-lhe completamente transparente, dizendo apenas respeito aos fabricantes. Para implementar um BAS será necessário conhecer que objectos e serviços são suportados e quais os dispositivos. Essa informação poderá ser encontrada no protocolo de conformidade de cada dispositivo (*Protocol Implementation Conformance Statement* - PICS) [19].

2.2.1.5 PICS

O PICS é basicamente uma lista de funcionalidades que um dispositivo suporta. Esta lista de objectos está presente no dispositivo, bem como, os pedidos de serviços que este pode receber e enviar. É útil comparar os requisitos do BAS com o PICS de modo a averiguar se este responde às necessidades [19].

2.2.2 LonWorks

O sistema LonWorks foi desenvolvido pela Echelon Corp. em 1990. Mais tarde, em 1999, o seu protocolo de comunicação LonTalk torna-se um ANSI/EIA standard 709, que é posteriormente actualizado em 2002, sendo esta a versão mais actualizada [9].

O sistema LonWorks é um sistema de rede do tipo *event-triggered*.[3] Sendo composto pelo protocolo de comunicação, LonTalk, por um controlador dedicado, Neuron Chip, e por ferramentas de gestão da rede [13].

2.2.2.1 LonMark

Em 1994 foi criada a LonMark, entidade autorizada para certificar, formar e fazer a promoção da interoperabilidade entre os diversos fabricantes. Em 2005, LonMark, já tinha certificado 600 dispositivos [20].

2.2.2.2 LonTalk

O LonTalk é o único protocolo suportado pelo sistema LonWorks, que permite a comunicação, entre os vários dispositivos existentes na rede LonWorks. Trata-se de um protocolo completo que implementa as sete camadas do modelo OSI.

2.2.2.2.1 LonTalk e o Modelo OSI

O LonTalk baseia-se no Modelo OSI (tal como acontece com o BACnet). Na Tabela 5 é possível identificar a correspondência entre as várias camadas.

[3] *Event-triggered* – A execução de tarefas é disparada por intermédio de um sinal de controlo.

	Camada OSI	Objectivo	Serviços prestados	CPU*
7	Aplicação	Compatibilidade a nível das aplicações	Objectos LonMark, propriedades de configuração, SNVTs, transferência de ficheiros	APP
6	Apresentação	Interpretação da informação	Variáveis de rede, interface de rede	NET
5	Sessão	Acções remotas	Pedidos/Respostas, autenticação, serviços de rede	NET
4	Transporte	Fiabilidade entre extremidades	Sinais de mensagem recebida, detecção de mensagens duplicadas	NET
3	Rede	Endereçamento de destino	Informação do caminho da mensagem, endereçamento *unicast* e *multicast*	NET
2	Ligação	Meios de acesso	*Framing*, codificação de dados, verificação de erros CRC, CSMA, controlo e detecção de colisões, prioridades	MAC
1	Física	Meios físicos	Meios de comunicação e esquemas de modelação	MAC XCVR

*Ver Capítulo 2.2.2.3

Tabela 5 - LonTalk e Modelo OSI[21]

2.2.2.2.2 Meios de comunicação

O LonTalk suporta vários meios de comunicação: rede eléctrica, par entrelaçado, rádio frequência, infravermelhos, cabo coaxial, fibra óptica. O mais comum neste tipo de sistema é par entrelaçado com velocidade de 78,1kbps (FT-10), permitindo segmentos de 500 metros com um cabo de baixo custo. Existe a variante usando a rede eléctrica, a qual também é bastante utilizada (LP-10). É comum usar-se um bus em par entrelaçado com uma velocidade de 1,25Mbps (TP-1250) para fazer a interligação entre os subsistemas baseados em FT-10. A fibra óptica aparece também como uma variante para implementar este bus.

Mais recentemente os Bus TP-1250 começam a ser substituídos por mecanismos de *IP tunneling*, LonWorks/IP. Através deste mecanismo é possível integrar dispositivos LonWorks em redes IP e internet [13].

2.2.2.2.3 Endereçamento

O endereçamento é feito de forma hierárquica. No topo dessa hierarquia encontra-se o domínio (*Domain*). Por exemplo, se são implementadas duas aplicações diferentes sobre o mesmo meio comunicação, podem ser usados dois domínios para manter essas aplicações completamente separadas. O tamanho da identificação (ID) destes domínios pode ir desde 0 a 48 bits, sendo que o dado mais pequeno pode ser incluído em qualquer das partes (*frames*) das mensagens, mas tem que ser suficientemente grande para que não existam interferências. Existe a possibilidade de um nó (*Node*) (Capítulo 2.2.2.2.4) ser membro de dois domínios.

No segundo nível da hierarquia encontra-se a *subnet*. Podem existir até 255 *subnets* por domínio. Uma *subnet* é um grupo lógico de nós. Um router inteligente opera ao nível da *subnet*, determinando de forma mais precisa o encaminhamento das mensagens.

O terceiro nível de endereçamento é o nó. Podem existir 127 nós por *subnet*. O que perfaz 255 x 127 = 32.385 nós que podem estar num determinado domínio. Um nó pode servir mais que um domínio, permitindo, por exemplo, que um sensor transmita os seus outputs para vários domínios.

Os nós, podem também ser agrupados, podendo existir 256 grupos num determinado domínio. Um grupo pode ter nós pertencentes a múltiplas *subnets* desde que dentro do mesmo domínio. São permitidos no máximo 64 nós que usem serviços do tipo *acknowledged* (Capítulo 2.2.2.2.5) e um número ilimitado de nós que utilizem serviços apenas informativos. Um nó pode pertencer a 15 grupos. O endereçamento ao nível do grupo reduz o tamanho do endereço que tem que ser transmitido com cada mensagem e subdividindo a rede consegue-se aliviar o tráfego de rede [21]. Toda esta hierarquia é apresentada na Figura 10.

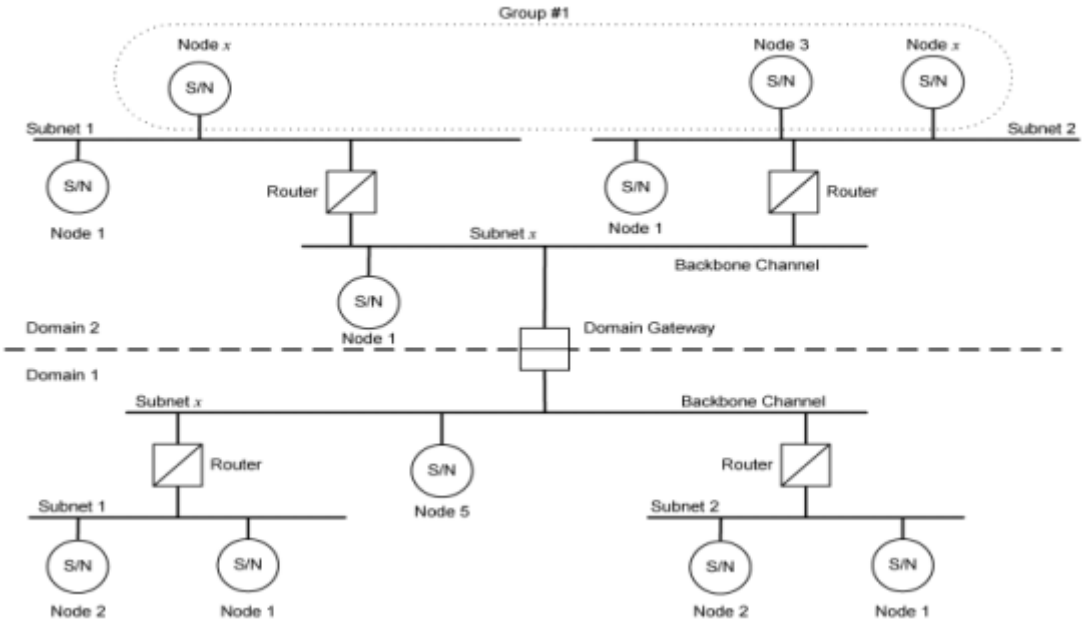


Figura 10 - Hierarquia/Segmentação da rede [13]

Cada nó possui um ID único de 48 bits (*Neuron ID*) que lhe é atribuído no momento da sua fabricação. Este endereço geralmente é apenas utilizado na instalação e configuração da rede, mas pode ser utilizado por aplicações que pretendam saber o número de série de determinado produto.

O meio de comunicação utilizado pelo nó não afecta a sua forma de endereçamento. Tal como um domínio pode ter na sua composição, vários meios de comunicação.

Na Tabela 6 estão presentes os 5 formatos de endereçamento de um nó.

Endereçamento	**Nós endereçados**
Domínio, *Subnet* = 0	Todos os nós do domínio
Domínio, *Subnet*	Todos os nós da *subnet*
Domínio, *Subnet*, Nó	Nó lógico específico
Domínio, Grupo	Todos os nós do grupo
Domínio, *Subnet*, *Neuron ID**	Nó físico específico

*O domínio e *subnet* são apenas necessários para facilitar o encaminhamento (routing)

Tabela 6 - Formatos de endereçar um nó [21]

2.2.2.2.4 Nó

As redes LonWorks são constituídas por dispositivos de controlo inteligente, os nós, que se servem do protocolo comum, LonTalk, para comunicar entre si. A "inteligência" dos referidos nós é providenciada pelos *Neuron Chips* (Capítulo 2.2.2.3), os quais permitem a implementação do protocolo LonTalk e a execução das funções de controlo.

Os nós (sensores inteligentes - de temperatura, de pressão, etc. -, actuadores, interfaces para o operador - PCs, *displays*, etc.) possuem uma interface física que permite a sua ligação ao meio de transmissão e incluem processamento local e recursos de rede, podendo ser ligados a dispositivos locais (I/O). Estes últimos permitem que cada nó processe os dados de entrada, recebidos dos sensores, e os dados de saída para os actuadores. Na Figura 11 encontra-se a arquitectura típica de um nó.

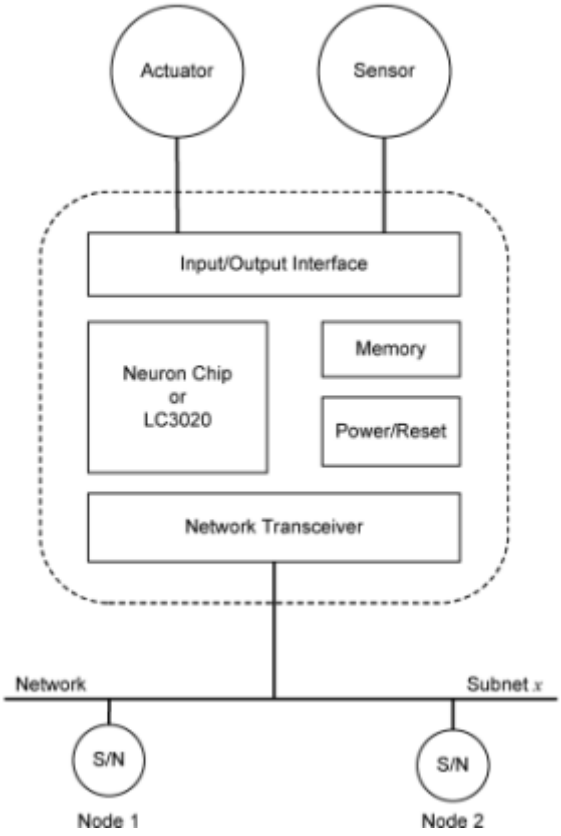


Figura 11 - Arquitectura típica de um nó [13]

Os recursos computacionais permitem que cada nó processe os dados, enquanto os recursos de rede possibilitam a sua interacção com outros nós existentes na rede. Os nós são programados para enviar mensagens a outros nós, de acordo com os eventos (mudanças de estado) detectados. Os endereços dos destinatários das mensagens de notificação são codificados numa tabela interna do *Neuron Chip*, através de uma ferramenta de configuração. Os dados enviados são formatados em variáveis de rede, denominadas por SNVTs (*Standard Network Variable Types*) (Capítulo 2.2.2.2.6) [22].

2.2.2.2.5 Serviço de mensagens

Existem 4 tipos básicos de serviços de mensagens. Os primeiros dois são serviços dos quais se espera uma resposta (*acknowledged*):

Acknowledged (ACKD), uma mensagem é enviada para um nó ou para um grupo de nós, e espera-se *acknowledge* de cada um desses nós. No caso de não se receber o *acknowledge* de todos os receptores, o emissor reenvia a mensagem. O número de tentativas e o tempo de espera de resposta pode ser configurado. O *acknowledge* é gerado automaticamente pelo controlador de rede não necessitando da intervenção de uma aplicação. O *acknowledge* além de servir para garantir que o receptor recebeu a mensagem serve também para evitar o envio de mensagens duplicadas.

Pedido/Resposta (REQUEST), uma mensagem é enviada para um nó ou para um grupo de nós, e espera-se uma resposta de cada um desses nós. A mensagem recebida é processada pela aplicação do receptor e só posteriormente será enviada a resposta. Existem as mesmas características de reenvio e tempo de espera como no ACKD.

Os outros dois tipos são serviços em que não é expectável a existência de resposta:

Repeated (UNACKD_RPT), uma mensagem é enviada para um nó ou para um grupo de nós repetidas vezes e não se espera uma resposta. Este serviço é tipicamente para envio de mensagens *multicast* (envio para vários nós) para grupos de nós que, no caso desses nós terem que confirmar a recepção, haveria uma elevada probabilidade de ocorrer uma sobrecarga na rede.

Unacknowledged (UNACKD), uma mensagem é enviada para um nó ou para um grupo de nós sendo que não é esperada uma resposta. Este tipo de serviço é utilizado quando

se necessita de uma maior velocidade de transmissão ou para o envio de uma grande quantidade de dados.

O LonTalk fornece um serviço de detecção de mensagens duplicadas, pelo que normalmente apenas é entregue uma mensagem, mesmo que sejam enviados duplicados. As duplicações ocorrem, por exemplo, quando um acknowlege ou uma resposta se perde. Apenas para o serviço REQUEST, dado que a resposta incluiu informação, é que a mensagem é entregue mais que uma vez. A capacidade de detecção de mensagens duplicadas é conseguida através da memorização das transacções recebidas em cada nó [21].

2.2.2.2.6 Variáveis de rede

A ligação lógica entre os sensores e os actuadores é realizada através de variáveis de rede. É através de uma ferramenta de configuração (LonMaker) que se ligam as variáveis compatíveis dos diferentes dispositivos LonWorks.

O grupo LonMark definiu uma lista normalizada de variáveis, denominada SNVT, uma das quais é a *SNVT_occupancy*, para definir variáveis do mesmo tipo. Este atributo desempenha um papel fundamental na interoperabilidade dos dispositivos. O SNVT permite, assim, efectuar a associação das variáveis de entrada e de saída.

Existe também uma lista normalizada de variáveis de configuração, denominada SCPTs (*Standard Configuration Parameter Types*) e uma lista de perfis para os dispositivos comuns, chamada FPs (*Functional Profiles*) [22].

2.2.2.3 Neuron Chip

O *Neuron Chip* é um dispositivo fabricado e distribuído pela Motorola e pela Toshiba, que implementa o protocolo LonTalk. Especialmente concebido para a tecnologia LonWorks, permite controlar todos os dispositivos utilizados naquela tecnologia. É o dispositivo preferencial do protocolo LonTalk porque foi criado especificamente para o efeito. No entanto, aquele protocolo também pode ser implementado numa arquitectura de processador genérica. O *Neuron Chip* possui (Figura 12):

- Três processadores de 8 bits (NET, MAC e APP) - dois deles especialmente vocacionados para a execução do protocolo LonTalk, cabendo ao terceiro as

aplicações do nó. Esta separação permite salvaguardar qualquer impacto negativo da aplicação na rede e vice-versa;

- Memórias EEPROM, RAM e ROM, variáveis de acordo com o modelo do Neuron Chip (todas as versões incluem, pelo menos, 0.5K de EEPROM e 1K de RAM);
- Um transdutor para conexão ao meio físico de comunicação;
- *Hardware* e *software* para construir dispositivos de controlo;
- 11 pinos de I/O;
- *Firmware* [4]LonWorks, incluindo o protocolo LonTalk, um sistema operativo de tempo real, *device drivers*[5], etc.

No caso de insuficiência do *Neuron Chip* (aplicações demasiado complexas), poder-se-á utilizar um processador externo, desde que o *firmware* do *Neuron Chip* seja alterado [22].

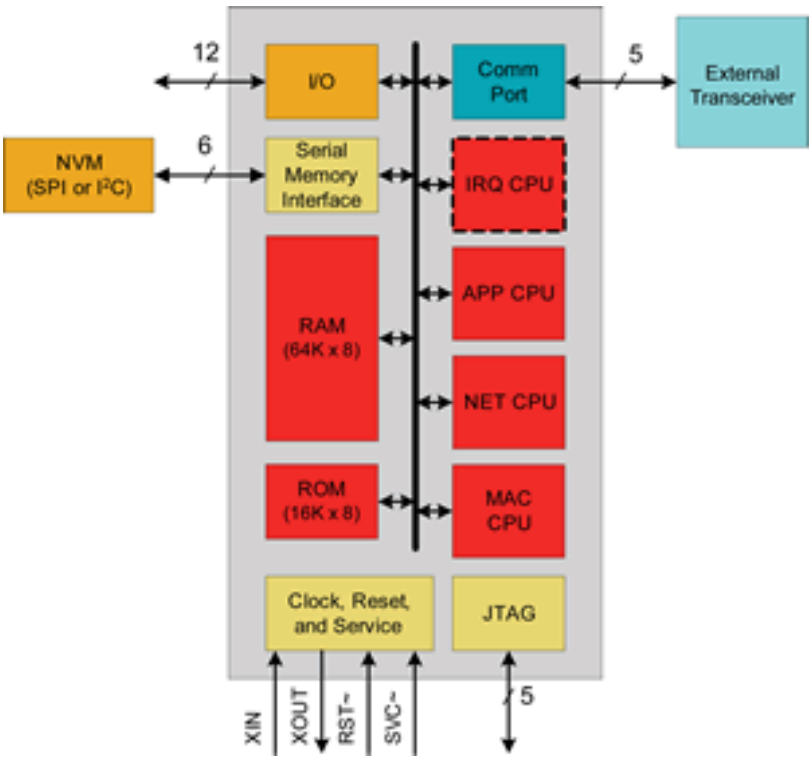


Figura 12 - Diagrama de blocos do processador Neuron 5000[23]

[4] *Firmware* é o conjunto de instruções operacionais programadas directamente no hardware de um equipamento electrónico.

[5] *Device driver* é o comando de um dispositivo ou programa. Faz a interface entre dispositivos diferentes (e.g. Impressora e PC)

2.2.3 KNX

O KNX (Konnex) é um sistema distribuído dedicado às necessidades de um edifício. Em 2002, o KNX foi definido através da combinação de 3 sistemas de gestão de edifícios já existentes EIB (*European Installation Bus*), Batibus e EHS (*European Home System*). Em 2004, foi publicado como standard europeu (EN 50090). Finalmente em 2006, tornou-se standard internacional (ISO/IEC 14543). O KNX é especificado e gerido pela *Konnex Association* (Capítulo 2.2.3.1) [24].

Este sistema é designado como o standard único europeu – "*one-single-standard*" e vem responder aos dois principais Standards Americanos (BACnet e LonWorks) [25]. As redes de dispositivos KNX são sempre desenhadas de forma a criar aplicações distribuídas, de forma a potenciar a interacção entre elas.

2.2.3.1 Konnex Association

Em 1999, a *Konnex Association* é criada através da cooperação entre o *BatiBUS Club International* (BCI), o *European Installation Bus Association* (EIBA) e o *European Home Systems Association* (EHSA).

A actualização, promoção, certificação e formação do standard é o objectivo principal desta associação [25].

2.2.3.2 Modos de configuração

2.2.3.2.1 S-mode (System mode)

Este modo de configuração é direccionado para instaladores especialistas, possibilitando a definição de funções de controlo do edifício mais sofisticadas. As componentes de um *S-mode* podem ser planeadas através do *software* ETS Professional (capítulo 2.2.3.4.2), através do qual é possível interligar os componentes e configura-los através de uma série de parâmetros disponibilizados pelo fabricante. Assim consegue-se uma maior flexibilidade nas funções de controlo do edifício.

2.2.3.2.2 E-mode (Easy mode)

Este modo de configuração é direccionado para instaladores com a formação básica em KNX. Os produtos *E-mode* compatíveis oferecem um número limitado de funções em comparação com *S-mode*.

Os componentes *E-mode* já vêm de fábrica programados e carregados com os parâmetros por omissão. Com uma versão mais básica do ETS (ETS Starter) é possível reconfigurar parcialmente o componente.

2.2.3.2.3 "A-mode" (Automatic mode)

Este modo de configuração é direccionado para o cliente final, tendo a filosofia Plug-and-Play [24], em que o instalador não necessita de conhecimentos de KNX para o configurar. Este modo será especialmente indicado para ser usado, por exemplo, em electrodomésticos.

2.2.3.3 Comunicação

Os componentes dividem-se basicamente em 3 classes; alimentadores, sensores (os que emitem ordens) e actuadores (os que executam ordens). Apenas os sensores e actuadores são elementos activos. A comunicação entre os actuadores e sensores é feita directamente entre os mesmos.

2.2.3.3.1 Meios de comunicação

O KNX fornece uma séria de meios de comunicação. Um meio de comunicação pode ser usado em combinação com um ou diversos modos de configuração. Permitindo assim ao fabricante uma melhor escolha das combinações para determinado produto ou mercado.

As redes KNX são tipicamente implementadas seguindo um modelo de duas camadas. No nível de campo pode-se encontrar os sensores, actuadores e controladores que interagem com o ambiente. Estes componentes do nível de campo estão interligados por uma espinha dorsal com capacidades de gestão (e.g. estação de trabalho) que têm uma visão global de toda a rede. Na Figura 13 pode-se verificar um exemplo típico desta topologia [26].

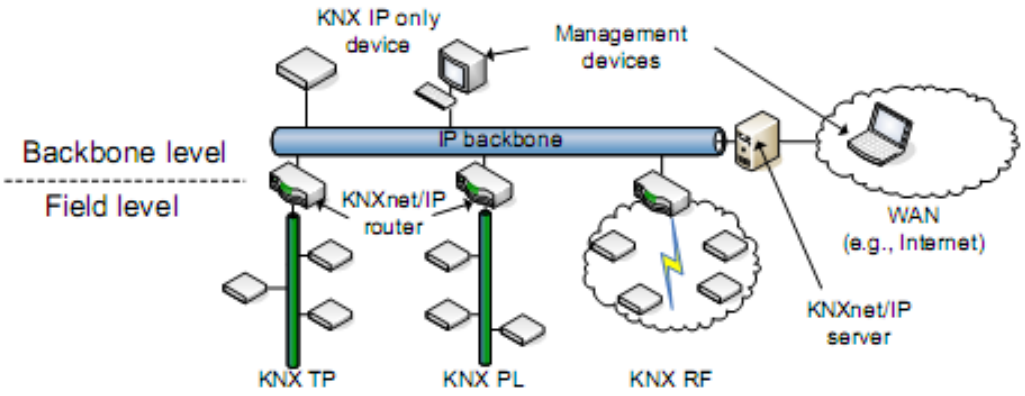


Figura 13 - Exemplo de uma rede KNX[26]

TP-0, (*Twisted pair, type 0*)

Este meio de comunicação é uma reutilização de um dos meios de comunicação definidos no protocolo BatiBus. Atinge uma velocidade de transferência de 4,8kbps. Apesar de ser possível operar na mesma rede do protocolo BatiBus não permite trocar informação com o KNX.

TP-1, (*Twisted pair, type 1*)

Este meio de comunicação é uma reutilização de um dos meios de comunicação definidos no protocolo EIB. Atinge uma velocidade de transferência de 9,6kbps. Os produtos que sejam certificados para operar numa rede TP-1, sejam do protocolo KNX ou EIB, conseguem cooperar sobre a mesma rede.

PL-110, (*Power-line, 110 kHz*)

Este meio de comunicação é igualmente uma reutilização de um dos meios de comunicação definidos no protocolo EIB. Atinge uma velocidade de transferência de 1,2kbps. Os produtos que sejam certificados para operar numa rede PL-110, sejam do protocolo KNX ou EIB, conseguem cooperar sobre a mesma rede.

PL-132, (*Power-line, 132 kHz*)

Este meio de comunicação é uma reutilização de um dos meios de comunicação definidos no protocolo EHS. Atinge uma velocidade de transferência de 2,4kbps. Os produtos dos dois tipos apenas conseguem cooperar através de um conversor de protocolo.

RF, (*Radio frequency, 868 MHz*)

Não existindo nenhuma implementação deste meio de comunicação em algum dos protocolos que deram origem ao KNX, esta implementação teve que ser desenvolvida directamente para o KNX. Atinge uma velocidade de transferência de 38,4kbps. É um meio sem fios utilizando a tecnologia de rádio frequência.

Ethernet

Este meio de comunicação existente em praticamente todos os edifícios pode ser usado em conjunto com o KNX/IP, que permite fazer *tunnelling* das mensagens KNX encapsulando-as em mensagens IP.

Endereçamento

O endereçamento físico dos componentes do sistema é feito recorrendo a conjuntos de 3 octetos. Num mesmo sistema podem ser suportados até 65536 componentes. Uma linha (*line*) permite 256 componentes, que podem ser agrupadas em linhas mestras (*main lines*), que por sua vez podem ser agrupadas em áreas. Um domínio é formado por 15 áreas interligadas pela espinha dorsal da rede (*backbone line*), Figura 14.

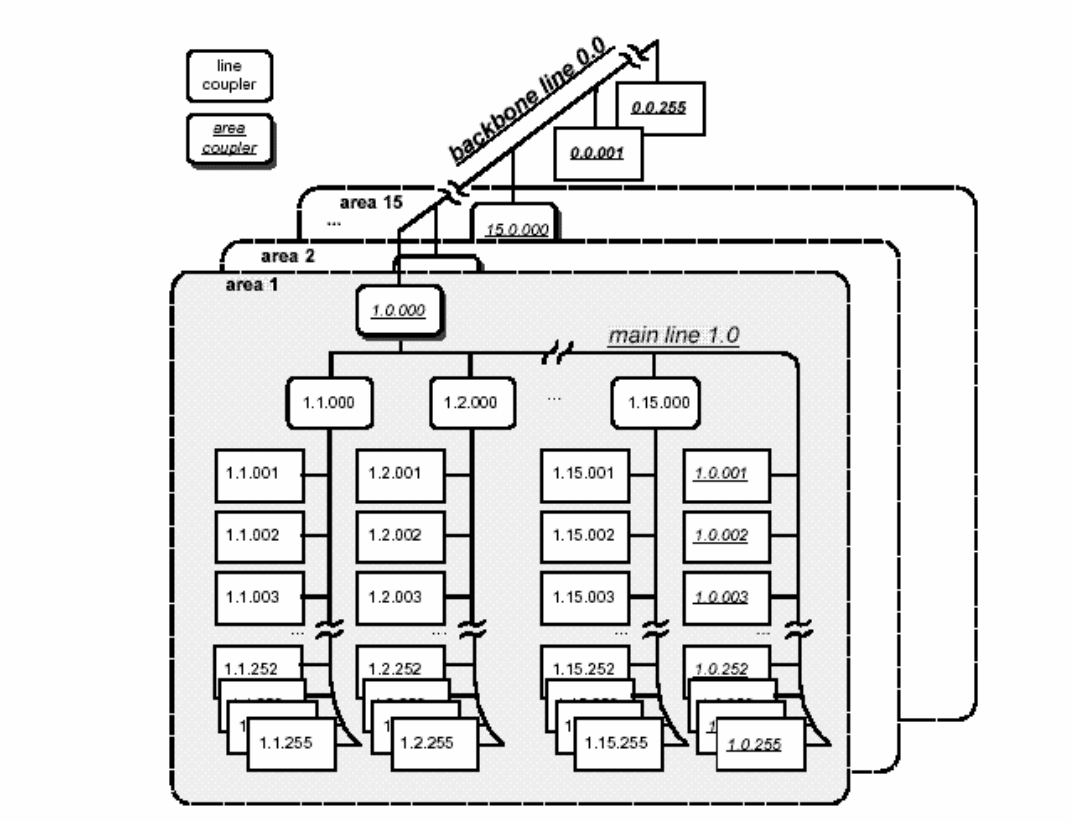


Figura 14 - Topologia lógica [24]

2.2.3.4 Representação e gestão dos dispositivos

2.2.3.4.1 Data-points

Os *data-points* são usados por um sistema KNX para dar e receber ordens, enquanto interligados formando uma rede lógica distribuída. Esta interligação lógica entre componentes tem como nome *binding*.

Existem 3 tipos de esquemas para interligar *data-points* [24]:

- Free *binding*: Apenas necessita de seguir algumas regras, nomeadamente os tipos de data-points (e.g. booleano);
- *Structured binding*: Estipula de forma precisa o padrão para a ligação com todos os data-points;
- *Tagged binding*: É semelhante ao anterior, mas tem sempre associado a informação de destino.

2.2.3.4.2 Engineering-Tool-Software (ETS)

Um *Engineering-Tool-Software* (ETS) é a ferramenta base de todo o sistema KNX, que permite planear, projectar e parametrizar os componentes e rede KNX. Encontra-se na versão 3 desde 2004 e, a versão 4 já foi apresentada mas ainda não foi libertada.

2.2.4 Comparação LonWorks, BACnet e KNX

A Tabela 7 resume a comparação entre as 3 tecnologias, LonWorks, BACnet e KNX. Posteriormente será efectuada uma comparação mais detalhada de algumas das características apresentadas na tabela.

LonWorks	**BACnet**	**KNX**
História		
1988 Desenvolvido pela Echelon 1994 Criada a LonMark 1995 1º Dispositivo certificado pela LonMark 1999 ANSI 709 2002 Actualização ANSI 709 2005 600º Dispositivo certificado pela LonMark	1987 Início do desenvolvimento pela ASHRAE 1995 ASHRAE-135 2000 Criada a BTL 2001 ANSI standard 2003 ISO 16484-5 2003 ASHRAE-135.1 2004 Actualização do ANSI 2005 100º Dispositivo certificado pela BTL	1988 Formação da EIB pela Siemens 1999 Formação do KNX 2004 EN 50090 2006 ISO/IEC 14543
Arquitectura de rede		
"Bottom Up" Protocolo de comunicação único Peer-to-Peer	"Top Down" Topologia de rede por camadas	"Bottom Up" Topologia livre de baixa velocidade
Gestão de rede		
Ferramentas de gestão de rede disponíveis através de aproximadamente 30 fontes Ferramenta única para aceder a todos meios de comunicação	Ferramentas de gestão de rede disponíveis através de apenas 5 fontes Não existe ferramenta única para aceder aos protocolos suportados Ferramentas de gestão geralmente fornecidas pelo fabricante	ETS 3 que é independente do fabricante

Arquitectura dos dispositivos		
Neuron Chip Neuron C (Linguagem)	Independente do processador Independente da linguagem	Inicialmente foi utilizado um processador 68HC05
Implementação do modelo OSI		
Todas as 7 camadas	Apenas 1,2,3 e 7	Apenas 1,2,3,4 e 7
Certificação		
LonMark	BTL	KNX
Dispositivos		
Routers, web servers, gateways, NICs, AVAC e não-AVAC	Routers, dispositivos de supervisão, gateways, AVAC e não-AVAC	Maioritariamente dispositivos de controlo de instalações mais reduzidas; Fraca oferta em produtos AVAC
Meios de comunicação		
Único protocolo: LonTalk Par entrançado - 78.1kbps ou 1.25Mbps *Powerline* - 4.8kbps *Wireless*, fibra óptica, infravermelhos estão publicados mas pouco utilizados	Vários protocolos suportados: Ethernet, ARCNET, MS/TP, LonTalk, PTP	Par entrançado - 9.6kbps *Powerline* - 1.2kbps até 2.4kbps *Wireless* - 38.4kbps
Internet		
LonWorks/ IP i.LON – Dispositivo que disponibiliza *Web Services* XML	BACnet/IP BACnet/WS	KNX/IP

Tabela 7 - Comparação entre LonWorks, BACnet e KNX [9] [27]

2.2.4.1 Rede

Uma diferença bastante significativa é o facto de o LonWorks utilizar apenas um protocolo de comunicação, LonTalk, enquanto o BACnet permite a utilização de 6 tipos

diferentes de protocolos. Existem vantagens e desvantagens, uma vez que desta forma o BACnet consegue responder melhor às necessidades específicas do edifício, mas o facto de existirem vários protocolos também pode ser gerador de confusão, dado que dispositivos que implementam protocolos distintos não conseguem comunicar directamente. Dai surge a vantagem do LonWorks pois, criando uma série de regras que se aplicam a todos os dispositivos, torna-se mais simples a interoperabilidade.

Como se pode verificar na Figura 7, tanto o LonWorks como o KNX direccionam-se mais para o nível de campo, respondendo a necessidades de controlo, “bottom up”, enquanto o BACnet tem como ênfase o sistema de gestão, “top down”. O que acontece em algumas situações é que o sistema de gestão entre os diversos subsistemas é implementado através de BACnet e os níveis de campo são tratados por protocolos como o LonWorks, KNX, ModBus, etc.

2.2.4.2 Arquitectura dos dispositivos

Uma desvantagem que o LonWorks apresenta é o facto da implementação dos seus dispositivos ter como base um processador específico, Neuron Chip. Este facto tem a vantagem de facilitar a criação de novos dispositivos, mas o facto de este ser produzido apenas pela Toshiba e pela Cypress Semiconductor constitui naturalmente uma desvantagem [28].

2.2.4.3 Mercado

Tanto o LonWorks como o BACnet apresentam uma desvantagem de mercado em relação ao KNX, pois ambos têm um número muito baixo de técnicos certificados. Para esta situação contribui o facto dos protocolos referidos apenas serem utilizados em edifícios de grandes dimensões com recurso a técnicos com elevada formação.

O KNX tem como vantagem, em relação aos outros dois protocolos, o preço, pois tanto BACnet como LonWorks têm custos de implementação mais elevados [27].

2.2.5 Open Building Information eXchange (oBIX)

O *Open Building Information eXchange* (oBIX) [29] foi desenvolvido para possibilitar a comunicação entre as várias componentes mecânicas e eléctricas de um edifício, bem como com as próprias aplicações empresarias.

2.2.5.1 Modelo

oBIX fornece um modelo flexível para descrever os dados e as operações disponíveis no servidor, sendo tudo representado através de objectos. Esses objectos também são usados para descrever os tipos de dados (classes) e operações (assinaturas dos métodos). A flexibilidade do oBIX baseia-se na possibilidade de definir qualquer tipo de objecto e suporte de subtipos (*is-a*) e composição (*has-a*).

oBIX segue uma abordagem RESTful (*Representational State Transfer*). Este tipo de abordagem baseia-se em recursos que compartilham uma interface uniforme e um conjunto muito restrito de operações sobre esses recursos. Esta abordagem considera o protocolo HTTP (*World Wide Web)*, onde apenas os comandos *GET, PUT* e *POST* são usados para aceder aos recursos. Isto também é verdade para os serviços oBIX, pois apenas três tipos de pedidos de rede são definidos no nível dos *WebService* (WS). Os dois primeiros tipos são de leitura (*read* - aplicável a qualquer objecto) e de escrita (*write* - para objectos *writable*). Para operações mais complexas, além das básicas "get" e "set", existe a possibilidade de definir operações próprias, em que estas podem ser definidas ao nível do oBIX. Para invocar essas operações um terceiro tipo de pedido é fornecido (*invoke*). A abordagem *RESTful* do oBIX permite suportar dois tipos de protocolo de *binding*, um simples *HTTP REST binding* e um *SOAP binding*.

O *Naming* em oBIX é realizado de duas formas complementares. Primeiro, cada objecto pode ter um nome (*name*). Segundo, pode ser atribuído um URI (*href*) a cada objecto, sendo isto necessário para que um objecto possa ser referenciado a partir do exterior, como forma de se poder candidatar a qualquer pedido de rede (*read*, *write* ou *invoke*).

2.2.5.2 Objectos

O objecto raiz especifica uma série de atributos obrigatórios (e.g. name), que são herdadas por todos os objectos filho. No modelo de dados estão definidos vários tipos primitivos, os quais são *boolean*, *integer*, *float*, *enumeration*, *string*, *points in time* e *time spans*. São também definidos objectos de forma a abranger uma série de aplicabilidades genéricas, sendo estes contentores (*lists* e *feeds*), erros, referências para outros objectos, operações.

O objecto *Lobby* tem uma localização bem conhecida no servidor e serve como ponto de entrada. Este tem três sub-objectos:

- *About*, fornece informações acerca do servidor;
- *Batch*, recebe uma sequência de operações a realizar;
- *Watch*, regista objectos num serviço de notificação.

Através destes dois últimos é possível reduzir o *overhead* do protocolo.

A interacção e representação dos dados primitivos provenientes do sistema de automação, é efectuada através da classe *Points*, que permite operações de leitura e escrita.

O histórico pode ser representado através da classe oBIX *History Record*, que reúne pares *Point* e *Time Stamp*. O objecto *History* define uma lista de registos históricos e uma série de métodos que permitem a pesquisa desse histórico.

O oBIX define um modelo normalizado de gestão e utilização de alarmes. Um servidor oBIX fornece formas de registo e alerta dos objectos de alarme (*feeds*). Sempre que um servidor detecta que o valor de um objecto corresponde a um condição de alarme definida, um objecto alarme é adicionado ao *feed*, de tal forma que esse objecto contém uma *timestamp* e a referência para a sua origem.

2.2.5.3 Representação

Os tipos de objecto base são directamente mapeados para elementos individuais XML. Por exemplo, se um objecto é um número inteiro ou é derivado de um número inteiro, ele é representado como <*int*/>. Quaisquer outros objectos que não são derivados de um

tipo de objecto base, são representados usando o elemento <*obj/*>, em que neste caso o atributo *is* especifica a classe (contracto) de um objecto.

Uma distinção semelhante existe a respeito de como os atributos de um objecto são representados. Os atributos base, por exemplo, o nome, são mapeados para atributos XML, estes são chamados de *facets*. Os atributos adicionados são representados como sub-objectos.

Os métodos correspondem, por um lado, aos pedidos de rede abordados acima (*read*, *write* e *invoke*) e, por outro, as operações acrescentadas ao nível do oBIX, representadas como sub-objectos.

Os sub-objectos podem ser incluídos no XML completo ou através de uma referência. A inclusão no XML ou o uso de referência deve ser definida no momento em que se define a hierarquia da classe. Por exemplo, um objecto de alarme oBIX é codificado como:

```
<obj name="somealarm" is="obix:Alarm">
   <ref name="source" href="/myhouse/somewhere"/>
   <abstime name="timestamp" val="2006-10-12_12:11:02"/>
</obj>
```

O objecto é referenciado pelo atributo is, em que se dá o nome de contrato. Actua como um template para o objecto que o referencia. Todos os sub-objectos deste objecto de referência são herdados, ainda assim, o objecto que o referencia pode fazer *override* desses sub-objectos, tal como na lógica orientada a objectos. Os objectos podem herdar múltiplos contratos. Os contratos também podem ser vazios, apenas descrevendo a semântica de um objecto - um exemplo é o *read-only oBIX Point*.

O exemplo de um contrato que descreve um modelo genérico de uma caldeira (classe ou template):

```
<obj href="def:furnace">
   <bool name="burnerOn"/>
   <real name="curTemp" is="obix:Point"/>
   <real name="setTemp" val="50.0" is="obix:WriteablePoint"/>
</obj>
```

Pode-se observar o uso de contractos derivados de *Point* (o contracto *WriteablePoint* adiciona uma operação de escrita) e que o valor por defeito do *setTemp* é 50. Um objecto que aplica o contracto anterior:

```
<obj name="furnace" href="myhouse/heating/furnace" is="def:furnace">
   <bool name="burnerOn" val="true"/>
   <real name="curTemp" val="45.3"/>
</obj>
```

Esta seria uma instância da classe caldeira. É herdado o valor por omissão de 50 para *setTemp*. Ao especificar o sub-objecto *curTemp*, qualquer valor por defeito possivelmente herdado é substituído, no entanto, o contrato *Point* mantém-se. Um *href* é especificado para permitir o acesso por parte dos clientes oBIX.

2.2.5.4 Conclusão

A abordagem do oBIX tem como vantagem o facto de ser implementado recorrendo a Web Services, uma das formas mais populares de implementar arquitecturas do tipo SOA, estes são independentes da plataforma, podendo ser implementados utilizando qualquer linguagem de programação e serem executados em qualquer plataforma de *hardware* ou sistema operativo. Este tipo de abordagem permite ainda o uso de técnicas como descoberta, segurança, transacções, eventos, etc.

Esta abordagem é muito semelhante à abordagem BIOS, pois ambas se baseiam em arquitecturas do tipo SOA, usufruindo das vantagens enumeradas acima, e ambas pretendem promover a cooperação entre os diversos subsistemas do edifício. Uma das principais diferenças é o facto da solução BIOS implementar um bus de serviços, este contém serviços próprios de descoberta e troca de informação simplificando a cooperação entre os demais serviços. Este tema será abordado nos capítulos seguintes.

2.2.6 OPC-UA

A versão mais recente do *OLE for Process Control* (OPC) sofreu uma evolução para *Open Process Control Unified Architecture* (OPC UA) que é um standard que tem como objectivo o transporte de dados desde da camada de automação (e.g. PLCs ou DDCs) até ao nível de gestão, que inclui por exemplo SCADA (S*upervisory Control And Data Acquisition*) ou mesmo um ERP (*Entrepise Resource Planning*) [30].

OPC Unified Architecture baseia-se em XML, WebServices e SOA e assenta sobre as normas já existentes do OPC *Classic*, reutilizando as componentes já definidas nessas normas, como por exemplo os três tipos de dados suportados - dados actuais (OPC DA), dados de histórico (OPC HDA) e alarmes e eventos (OPC A&E).

O standard OPC UA é independente de fabricante ou qualquer protocolo específico, pois este foi desenvolvido recorrendo a tecnologias web standard.

2.2.6.1 OPC *Classic*

O OPC *Classic* é a base/início do OPC UA. Baseia-se numa tecnologia cliente/servidor, tendo sido projectado com base no Microsoft Active X (OLE), COM e tecnologia DCOM. Resumidamente, o OPC Classic define um conjunto de interfaces, métodos e propriedades que permite a transferência de dados entre um servidor e um cliente. Um servidor OPC fornece as informações a pedido de um cliente OPC.

As características chave do OPC Classic e do OPC UA são colocadas em forma de comparação na Tabela 8.

	OPC Classic	**OPC UA**
Comunicação	DCOM	UA TCP SOAP/HTTP
Plataforma	Microsoft	Multi-plataforma
Desenvolvimento	C/C++	C/C++, .NET, C#, Java, outras
Servidores locais	COM	UA stack based on Binary TCP
Espaço de endereçamento	Estrutura hierárquica	Estrutura hierárquica por defeito. Fornece um modelo de informação muito rico que permite a modelação de qualquer sistema através de referências diferentes.
Funcionalidades	Real-time, Histórico ou Eventos	Real-time, Historical data, Historical Events, Events, Programas e modelos específicos à industria.

Tabela 8 - Características chave OPC Classic e OPC UA [31]

2.2.6.2 Especificação OPC UA

A especificação OPC UA está dividida em 12 partes, tal como se pode ver na Figura 15, existindo as *core specifications* que definem a base de todo o OPC UA e as *access type specifications* que servem para definir os *information models*. Apenas a parte 1 desta especificação é pública a pessoas que não sejam membros da *OPC Foundation* [32]. Abaixo são introduzidas cada uma destas partes, que para maior detalhe requer a consulta de cada uma das partes na especificação. De qualquer forma o livro [33] pode ser consultado em substituição da especificação propriamente dita.

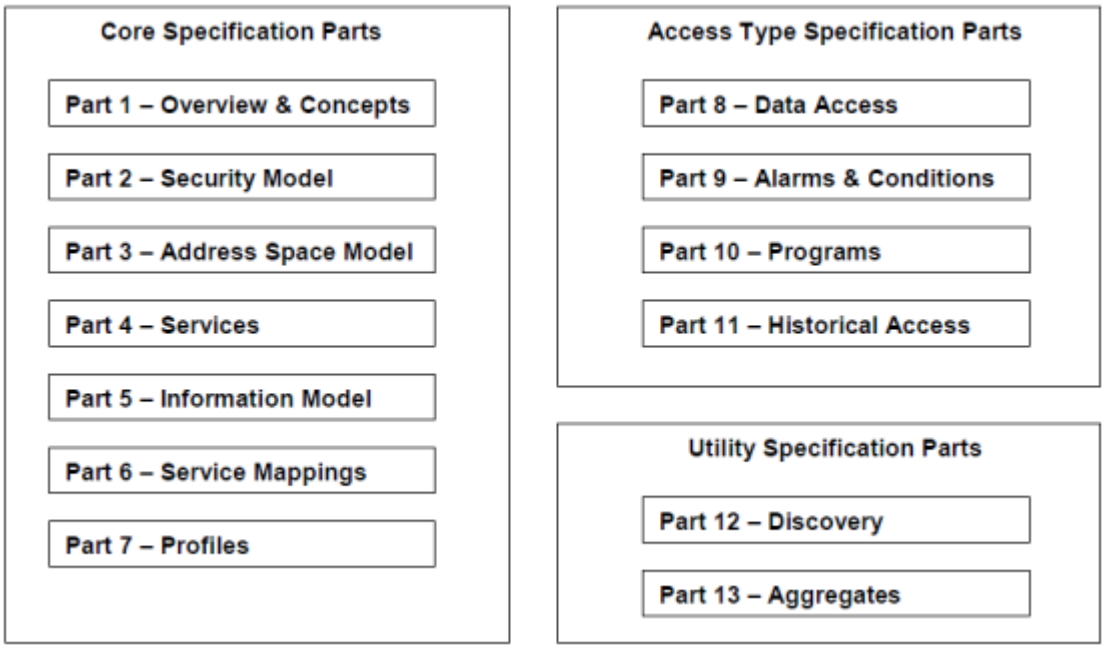


Figura 15 - Especificação OPC UA [32]

As duas primeiras partes da especificação não são normativas. A parte *Overview & Concepts* (UA Parte 1) dá uma visão geral sobre OPC UA e a UA Parte 2 descreve os requisitos e o modelo de segurança para as interacções entre *OPC UA Clients* e *OPC UA Servers*.

As partes 3 e 4 da especificação descrevem como modelar e aceder à informação, constituindo as especificações chave para desenvolver aplicações OPC UA.

A parte *Address Space Model* (UA Parte 3) descreve o conteúdo e estrutura da informação que o servidor disponibiliza para os clientes (*Address Space*).

A parte *Services* especifica as várias possibilidades de interacção entre clientes e servidores. Um cliente usa um serviço para aceder à informação disponibilizada pelo servidor. Os serviços são abstractos pois definem a informação que vai ser trocada mas não como vai ser trocada, tanto ao nível físico como da API utilizada pelas aplicações. A Figura 16 mostra as camadas de abstracção associada à arquitectura de comunicação.

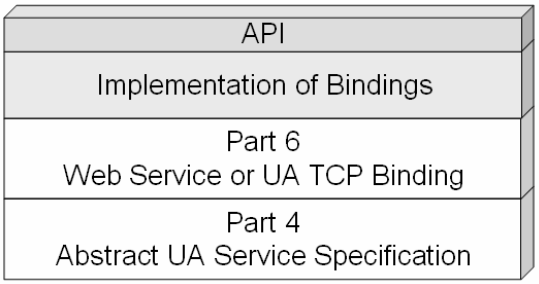


Figura 16 - Arquitectura de comunicação [33]

Na parte 6 são especificados os mapeamentos dos *UA Services* para mensagens, os mecanismos de segurança aplicados a essas mensagens, bem como as codificações suportadas.

O modelo de informação é especificado na parte 5, sendo definidos os tipos e as suas relações, suportados por um *OPC UA Server*.

A parte 7 (*Profiles*) especifica os perfis que estão disponíveis para clientes e servidores. Estes perfis definem grupos de serviços ou de funcionalidades, sendo desta forma simplificada a verificação de compatibilidade entre servidor e cliente.

Nas partes 8 à 11 são especificados os modelos de informação para acesso aos diversos tipos de dados, divididos tal como no *OPC Classic*. A *Data Access* (DA) (parte 8) lida com a representação e uso dos dados de automação de um servidor OPC UA. A *Alarm and Conditions* (AC) (parte 9) especifica o processo de alarme e monitorização de máquinas de estado e eventos. Em *Programs* (parte 10) é definida uma máquina de estados base cujo objectivo é executar, manipular e monitorizar os programas/métodos definidos no OPC UA que são como uma forma de suporte aos serviços disponibilizados, conseguindo assim uma mais fácil interoperabilidade e gestão da automação. A *Historical Access* (parte 11) especifica o uso do histórico de acesso aos

serviços e como apresentar a informação de configuração dos dados e eventos de histórico.

A parte 12 (*Discovery*) define a forma de encontrar servidores numa rede e como um cliente obtém a informação necessária para estabelecer uma ligação com determinado servidor.

Por fim, a parte 13 (*Aggregates*) especifica a forma como se processa e retorna agregados como mínimo, máximo, média, etc. Estes podem ser calculados tendo por base informação em tempo real ou mesmo proveniente do histórico.

2.2.6.3 Conclusão

O OPC-UA é uma iniciativa que promove a uniformização da comunicação e controlo, entre *hardwares*, geralmente autómatos, e entre *hardware* e cliente, tais como aplicações gráficas, sistema de gestão integrada, etc.

O OPC-UA resolve nesta dissertação, alguns problemas ao nível da comunicação e controlo entre a arquitectura BIOS, mais concretamente na implementação dos serviços de subsistema, e o *hardware*. Através desta iniciativa é possível virtualizar o *hardware* num servidor OPC-UA, desta forma os interessados, chamados de clientes OPC-UA, têm acesso à monitorização e/ou ao controlo dos demais componentes de *hardware* presentes na configuração do servidor OPC-UA. Este tema será abordado nos capítulos seguintes.

3. Arquitectura proposta

Na arquitectura proposta é necessária a interacção de diversos serviços que fazem a representação e abstracção de *hardware* e *software* (*Service-oriented architecture* - SOA). Esses serviços são descritos neste capítulo em maior detalhe. A arquitectura proposta implementa um bus lógico cooperativo (BIOSbus) de forma a que a interacção entre os diversos serviços possa ser efectuada de forma independente.

Propõe-se uma arquitectura que promova a cooperação entre os diversos serviços: subsistemas do edifício, gestão energética, gestão integrada, monitorização, etc. Cada um destes serviços oferece as funcionalidades que permitem manipular e monitorizar a informação da qual são responsáveis, de uma forma transparente em relação ao que é intrinsecamente necessário para efectuar o seu controlo.

A Figura 17, baseada na Figura 7 presente no capítulo 2, resume os diversos níveis de funcionalidades de um edifício, estes, tal como descrito anteriormente, são nível de campo, de automação e de gestão. No nível de gestão encontram-se os elementos responsáveis pela configuração e gestão do edifício, onde é efectuada a manipulação da informação gerada no nível inferior, o nível de automação, mas não é feita qualquer intervenção que afecte de alguma forma os algoritmos de automação. O mesmo se aplica à arquitectura BIOS, pois esta opera ao nível de gestão, garantindo que não há intervenção directa nos níveis inferiores, permitindo apenas, através da utilização de variáveis de entrada e saída, fazer uma interacção com o controlador/algoritmo de automação. Com esta separação, é possível garantir que a arquitectura proposta consiga enquadrar os compromissos de cooperação entre os diversos sistemas sem que isso comprometa a integridade de cada um, pois o algoritmo de controlo não é afectado.

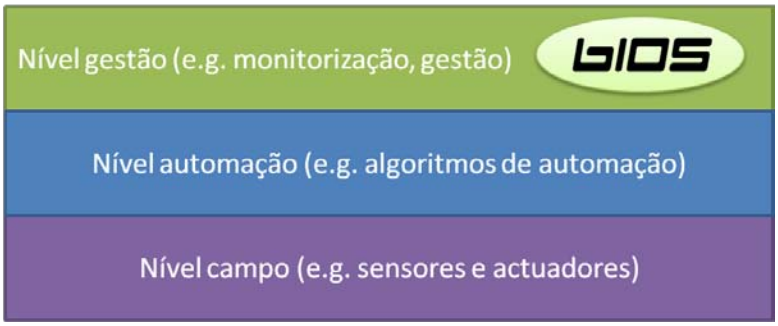


Figura 17 - Níveis de funcionalidades

Na Figura 18 encontra-se esquematizada a arquitectura proposta: os serviços de subsistema estão representados a verde; A laranja estão os serviços disponibilizados pelo bus cooperativo; a azul estão os serviços de comunicação, que estão encapsulados nos respectivos serviços subsistema; a vermelho estão os serviços consumidores, gestão integrada, energética e coordenação.

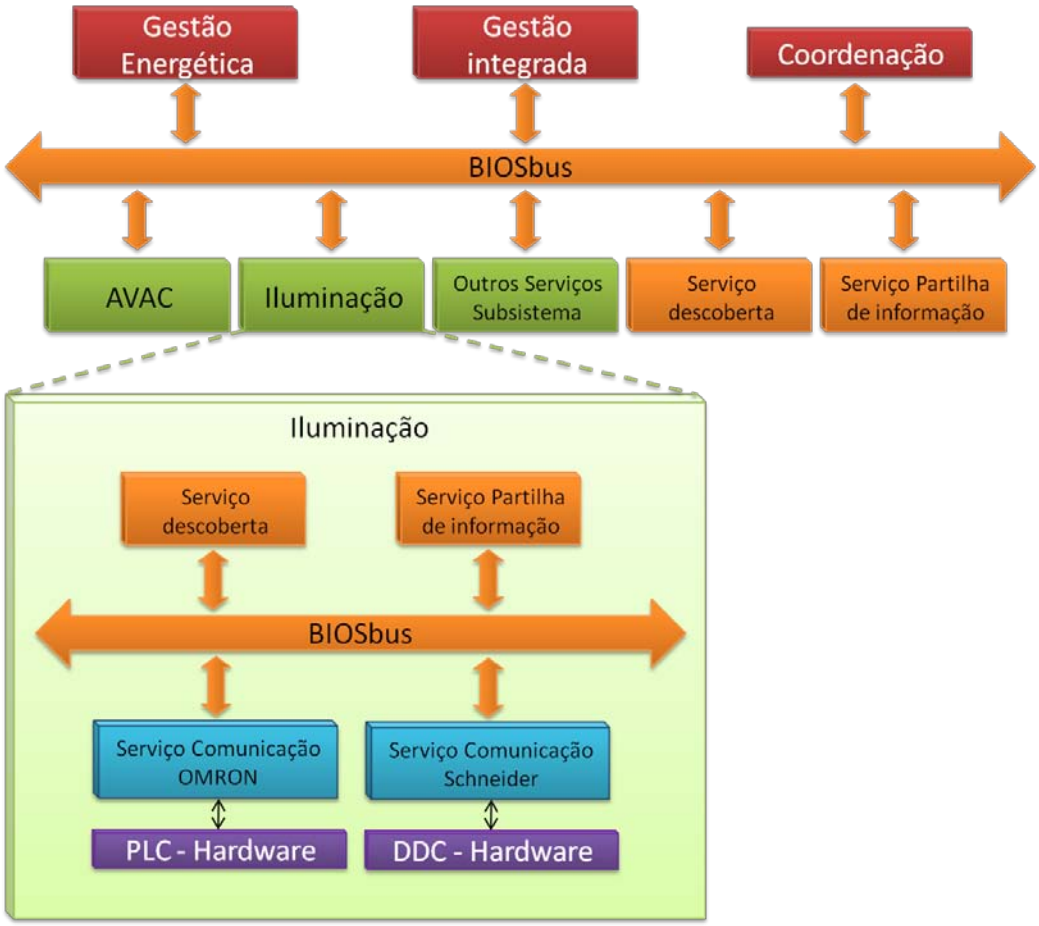


Figura 18 - Arquitectura proposta

Tal como mencionado acima, o objectivo é operar apenas no nível de gestão, mas visto que cada subsistema tem uma representação em *hardware*, é necessário que a implementação dos subsistemas recorra a serviços cuja responsabilidade seja a de efectuar a abstracção do *hardware*. Estes serviços denominam-se por serviços de comunicação. Um subsistema pode recorrer a mais que um *hardware*, podendo esses *hardwares* serem distintos ao nível do fornecedor ou protocolo. Isto de forma a facilitar a implementação do subsistema, tanto a nível de facilidade de adaptação à mudança ou mesmo da transparência em relação ao tipo de *hardware* que automatiza o subsistema,

os serviços de comunicação cooperam sobre um bus com funcionalidades idênticas ao BIOSbus mas internos a cada subsistema.

3.1 *Bus* cooperativo

Um *bus* cooperativo vem facilitar a divisão de responsabilidades entre os diversos sistemas, dado que o controlo e comunicações são separados por diversas entidades, simplificando-se assim a integração para o objectivo comum, a criação de condições que garantam a cooperação.

O uso da palavra *bus* advém da analogia ao barramento físico existente nos computadores para a comunicação entre os diversos dispositivos. Neste caso, o *bus* encontra-se num nível de abstracção mais elevado, tendo como vantagem principal a redução das comunicações ponto a ponto, pois cada entidade do *bus* não tem de conhecer todas as restantes. Desta forma fica simplificada a adaptação à mudança por parte do sistema, em que idealmente a adição ou remoção de entidades ao *bus* pode ser feita de forma automática.

O *bus* cooperativo é o concentrador de todas as entidades da arquitectura, este disponibiliza serviços que permitem simplificar a sua utilização, bem como potenciar a integração das entidades. Estas entidades são o serviço de descoberta e o serviço de partilha de informação, descritas de seguida.

3.1.1 Serviço de descoberta

Para adicionar um serviço ao bus, este tem que se registar na base de dados de serviços, ficando deste modo disponível para o serviço de descoberta o encontrar. O pedido de descoberta pode ser efectuado por qualquer outro cliente ou serviço do *bus*, sendo fornecido ao serviço cliente um *proxy* para o serviço procurado. Através do serviço de descoberta é possível obter um serviço sem que, à partida, se conheça a sua implementação ou mesmo quem o disponibiliza. Apenas através de um *proxy* para o serviço e da sua interface consegue-se executar as funcionalidades por ele disponibilizadas. O serviço de descoberta e de registo vem facilitar a integração e aumentar a tolerância à mudança.

3.1.2 Serviço de partilha de informação

Um serviço de partilha de informação é um repositório que permite armazenar informação e disponibilizá-la aos interessados. Este serviço oferece as seguintes funcionalidades: escrever, ler e registar notificação sobre alterações de estado.

Esta forma de partilhar informação oferece um desacoplamento temporal, pois a troca de informação é efectuada através de um repositório com acesso comum. Um cliente interessado em determinada informação, apenas tem que se registar como seu subscritor dela e desta forma o cliente é notificado sobre alterações no estado da informação.

Este tipo de abordagem reduz as necessidades de comunicação entre serviços, veja-se como exemplo o serviço de gestão energética, este não vai estar, constante e directamente, a monitorizar as alterações dos demais serviços por ele utilizados, apenas subscreve no serviço de partilha de informação os serviços sobre os quais quer ser informado e espera pelas notificações de alteração de estado.

3.2 Serviço de Comunicação

O serviço de comunicação, tal como o nome indica, é o serviço que abstrai as comunicações com os componentes físicos e/ou lógicos utilizados na automação das diversas funcionalidades do edifício.

As operações possíveis sobre as entidades envolvidas na automação são definidas numa interface, neste caso são apenas operações de leitura e de escrita. Assim sendo, os serviços que recorram ao serviço de comunicação apenas têm de conhecer esta interface, sendo desnecessário saber qual o tipo de automação, protocolo ou mesmo detalhes de implementação do serviço.

Sabendo que a interface do serviço de comunicação é independente do *hardware* que este virtualiza e sendo as implementações necessárias da responsabilidade de cada serviço de comunicação, consegue-se garantir que as implementações de outros serviços ou clientes não ficam dependentes do *hardware* mas sim do serviço de comunicação, mais concretamente da interface por este implementada.

Actualmente este serviço não funciona sem que haja um adaptador do sistema original para o BIOS. Para que essa conversão não fosse necessária, o *hardware* teria que disponibilizar o serviço com uma interface tal como especificado pelo sistema BIOS (no futuro pode vir a acontecer). Assim sendo, as diversas implementações do serviço de comunicação têm que recorrer a um adaptador para efectuar a comunicação entre o *hardware* e o serviço de comunicação em causa, pois cada *hardware* tem as suas necessidades e especificações para o seu controlo e monitorização por parte de uma entidade externa, especificações essas que são definidas através de protocolos.

3.3 Serviço de Subsistema

Este serviço disponibiliza as funcionalidades necessários para o controlo e monitorização de um determinado subsistema. Este tipo de serviço já usufrui da abstracção em relação ao *hardware* oferecido pelo serviço de comunicação e acresce a abstracção ao nível do subsistema, que inclui a abstracção de como se obtém um valor de determinado componente do subsistema ou que operações de controlo estão acessíveis num determinado subsistema.

As operações de monitorização e controlo possíveis de efectuar por parte de determinado subsistema são definidas numa interface. A cooperação entre os diversos subsistemas não implica que se tenha conhecimento sobre a implementação do serviço de determinado subsistema. Por exemplo, o subsistema "Elevadores", é virtualizado por um serviço com o mesmo nome e implementa uma interface que descrimina uma versão simplificada das operações de monitorização e controlo possíveis [34]:

- Monitorização:
    - Ocupado/Livre
    - Bloqueado/Desbloqueado
    - Piso actual
    - Pisos de destino
    - Sentido (sobe/desce)
- Controlo:
    - Marcar como Ocupado/Livre
    - Bloquear/ Desbloquear
    - Enviar para piso

Os clientes do serviço de "Elevadores" apenas têm o conhecimento da interface do serviço que descrimina as operações possíveis, podendo afirmar que para o cliente o subsistema de elevadores é uma "caixa preta" onde a interacção ocorre através das operações implementadas. Apesar de transparente ao cliente, estas operações têm que ser implementadas pelo serviço, e para o caso do subsistema "Elevadores" existe a necessidade de comunicar com o *hardware* responsável pela automação. Esta comunicação, tal como referido no capítulo anterior, depende do *hardware* que se encontra implementado, mas recorrendo aos serviços de comunicação esta implementação fica simplificada, pois, para além de não existir a necessidade de conhecer detalhes de *hardware*, a sua implementação do subsistema não fica dependente deste.

3.4 Sistema de gestão integrada

O sistema de gestão integrada usufrui da facilidade de cooperação oferecida pelos serviços acima descritos. Estes são todos concentrados como um único sistema auxiliando dessa forma o controlo e a monitorização do edifício.

Este sistema pode ser apresentado através de uma interface gráfica, sendo que dessa forma a gestão em tempo real do edifício fica facilitada, pois numa única interface estão centralizados os elementos existentes no edifício.

A gestão integrada além de agilizar a monitorização e controlo do edifício através da centralização dessas operações numa única interface, potencia também a identificação de alguma anomalia em determinado subsistema, fazendo chegar os devidos alarmes às entidades competentes, por exemplo, enviando um pedido de manutenção directamente para o piquete responsável. Utilizando o exemplo dos "Elevadores" abordado anteriormente, se o serviço de "Elevadores" informar que tem uma anomalia, ou mesmo que esta seja detectada através da análise dos dados deste subsistema, podem ser accionadas de imediato e de forma automática operações de segurança e/ou de manutenção.

3.5 Sistema de gestão energética

O sistema de gestão energética tem como objectivo efectuar uma gestão racional dos consumos de energia eléctrica do edifício, tendo disponível para isso a informação de todos os outros subsistemas.

A gestão dos consumos de energia pode ser conseguida de duas formas diferentes: através de algoritmos que previnem picos de consumo de energia, através de deslastre de carga, sendo esta técnica abordada de forma mais detalhada no capítulo 4; através da análise da informação dos diversos subsistemas de forma a serem detectadas antecipadamente anomalias que possam estar a resultar em consumos desnecessários de energia.

A possibilidade de efectuar uma gestão de energia por prioridades, é um exemplo de uma potencialidade do sistema de gestão energética. Tomando como exemplo o início da manhã de um edifício, em que a chegada de um elevado número de pessoas provoca um aumento do uso dos elevadores e da climatização e se for o caso o arranque de máquinas industriais. Nestes primeiros momentos da manhã o consumo energético pode exceder a energia contratada. As prioridades energéticas estão dependentes do negócio em que o edifício se insere. Para este exemplo as prioridades são as seguintes: o escoamento das pessoas; só depois a climatização. O domínio da hora de chegada de muitas pessoas ao edifício pode ser identificada ou, como no caso deste exemplo, configurada. Desta forma o sistema de gestão energética acciona o sistema de climatização antes da hora de ponta correspondente às entradas no edifício, de forma a que aquando da chegada das pessoas seja possível dar prioridade aos elevadores, sem que a climatização do edifício seja afectada, podendo, como último recurso, ser desligado o sistema de climatização durante a hora de ponta.

Este exemplo encontra-se simplificado, pois o objectivo é demonstrar a integração dos vários subsistemas para o objectivo comum de gestão energética. Os elementos intervenientes encontram-se na Figura 19. O subsistema "Elevadores" disponibiliza as funcionalidades "Activar estado hora de ponta" e "Activar estado normal", o subsistema AVAC disponibiliza as funcionalidades "Ligar" e "Desligar", existe uma simplificação das funcionalidades pois, seria também relevante por exemplo identificar a temperatura, as preferências do utilizador etc.

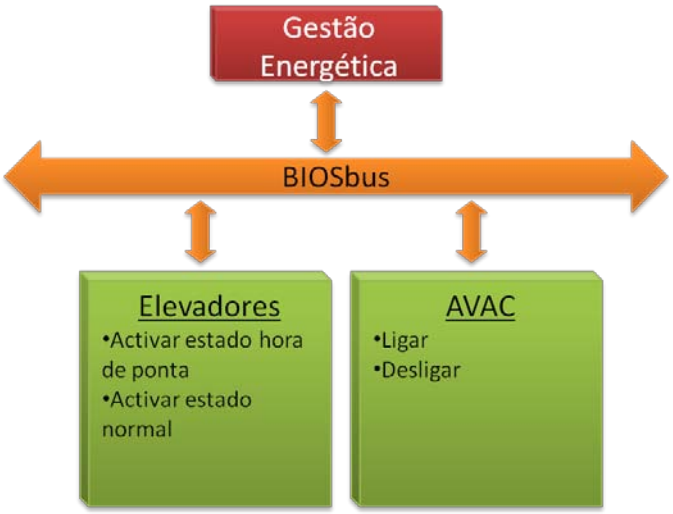


Figura 19 - Exemplo gestão energética

Concretizando o exemplo, o sistema de gestão energética é previamente configurado com a hora de ponta do edifício, e 60 minutos antes dessa hora é ligado o sistema de climatização (AVAC), desta forma no momento da hora de ponta já estará a climatização num estado onde o seu consumo energético é menor que na fase de arranque. Chegada a hora de ponta é alterado o estado dos elevadores para "hora de ponta", este estado coloca todos os elevadores com prioridade nas subidas. Todo este processo encontra-se demonstrado na Figura 20. Este exemplo poderia ser mais complexo, podia-se, por exemplo, integrar o subsistema "Energia" responsável pelos dados energéticos em tempo real de forma a verificar a necessidade de desligar o sistema de climatização.

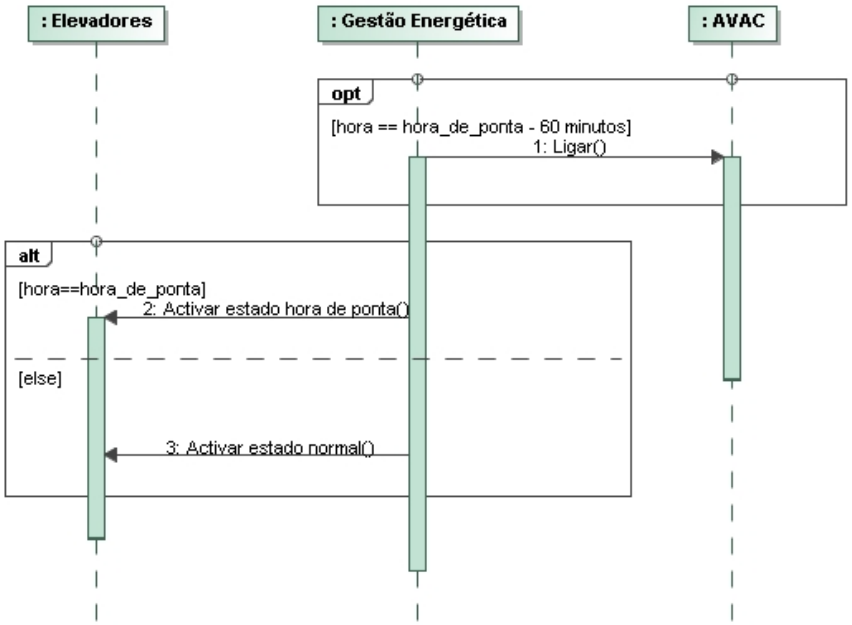


Figura 20 - Sequência exemplo gestão energética

3.6 Sistema de coordenação

O objectivo do sistema de coordenação é promover a interoperabilidade entre subsistemas, pois existem situações em que o estado de determinado componente de um subsistema pode ser importante para uma melhoria da gestão de um outro subsistema.

Torna-se como exemplo a coordenação conseguida entre os subsistemas "Controlo de Acessos", "Elevadores" e "Base de Dados Utilizadores", os intervenientes deste exemplo encontram-se representados na Figura 21.

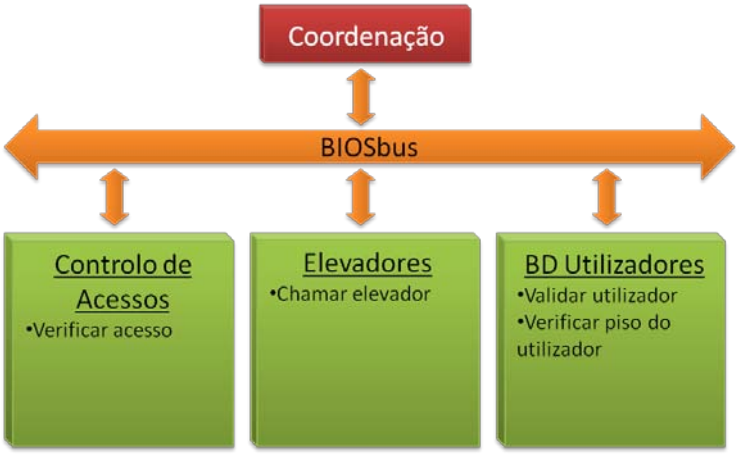


Figura 21 - Exemplo coordenação

Para este exemplo os três subsistemas são isolados sendo toda a cooperação conseguida através do serviço "Coordenação". Este caso inicia-se com a identificação de um utilizador no sistema de controlo de acessos, visto que para este exemplo o sistema não tem a informação dos utilizadores, esse pedido é reencaminhado para o serviço "Coordenação". Este, por sua vez, valida o utilizador junto do sistema "BD Utilizadores" cuja resposta é reencaminhada para o sistema "Controlo de Acessos". De forma a exemplificar melhor as potencialidades desta arquitectura pode-se tornar este exemplo mais complexo. Depois de validado o utilizador, verifica-se qual o piso que este tem acesso e com essa informação o serviço "Coordenação" pode automaticamente chamar o elevador, cujo destino é o piso em causa, através do sistema "Elevadores". Todo este processo encontra-se demonstrado na Figura 22.

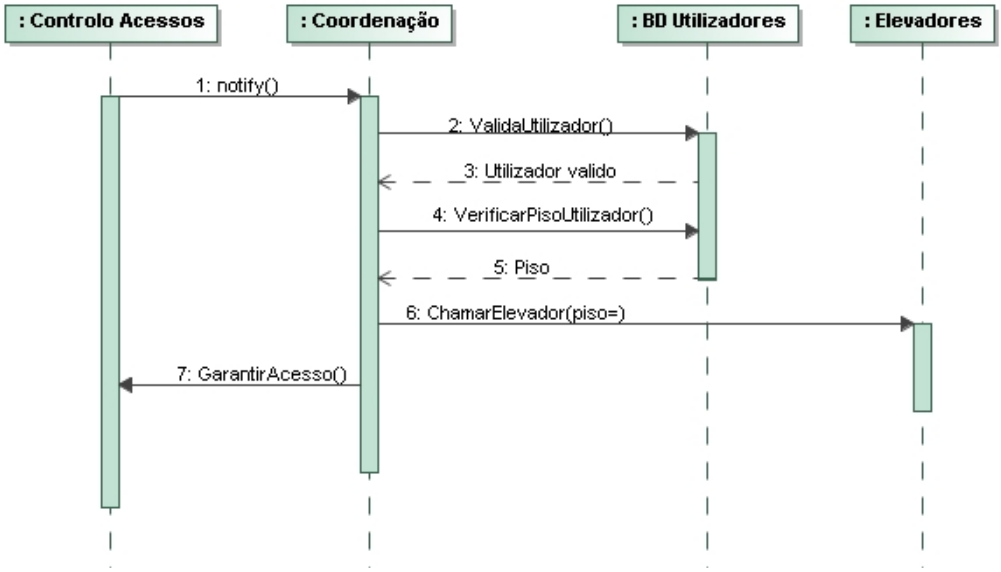


Figura 22 - Sequência exemplo coordenação

Através do sistema de coordenação é possível que a dependência entre os subsistemas seja garantida pelo BIOS. Não existindo a necessidade que estes tenham conhecimento da implementação dos demais subsistemas ou mesmo do *hardware* ou protocolo utilizado. Para este exemplo em concreto, o facto de ter o sistema "Base de Dados de Utilizadores", é possível centralizar toda a informação dos utilizadores do edifício, evitando redundância e possíveis inconsistências, podendo ser integrado um sistema já existente na empresa. Além de que com este tipo de abordagem potencia-se a criação de

valor acrescentado ao edifício, como o caso de chamar o elevador com o destino do utilizador identificado.

Apesar do exemplo aqui apresentado demonstrar a cooperação entre diversos subsistemas através do serviço "Coordenação", existe a possibilidade de que este seja feito dentro de um único subsistema. O caso implementado e descrito em pormenor no capítulo 4.2.3.5 é o subsistema "Alarme de CO_2", neste caso o serviço "Coordenação" é utilizado de forma a conseguir cooperação de hardware, pois o sensor e os actuadores são controlados por hardwares de fabricantes diferentes.

A arquitectura permite que a coordenação seja feita internamente ao subsistema, mas isso comprometeria o desacoplamento existente entre subsistemas, pois teriam de cooperar com outros directamente. Da forma proposta um subsistema não tem que ter conhecimento sobre quais são os subsistemas que se encontram ligados no BIOS.

3.7 Adaptação à mudança

A arquitectura proposta foi desenhada de forma a facilitar a adaptação à mudança . Por exemplo, na necessidade de adicionar um novo subsistema, sabendo que não existe uma dependência directa entre eles, pois todos os subsistemas cooperam através de um bus, este novo subsistema seria adicionado ao bus como um serviço subsistema, estando automaticamente disponível para que outros serviços cooperassem com ele. Esta automatização na adição de novos serviços advém das potencialidades do bus já descritas no capítulo 3.1. Idealmente, ao adicionar o novo subsistema, este ficaria dinamicamente disponível no sistema de gestão integrada, mais propriamente na interface gráfica.

No caso da mudança ser ao nível do *hardware*, a arquitectura proposta também vem facilitar a adaptação. Neste caso pretende-se que o subsistema se mantenha o mesmo, apenas existindo alteração do hardware responsável pelo seu controlo. Esta alteração é totalmente transparente para os restantes serviços, pois a esta mudança é interna ao serviço subsistema. Para os utilizadores dos serviço de comunicação, que é o caso do serviço subsistema, é desconhecida a implementação do serviço, tendo apenas a informação da interface. Neste caso apenas seria necessário alterar no serviço subsistema o novo serviço de comunicação responsável pelo hardware. Visto que a

interface do serviço de comunicação se mantém entre implementações, a implementação do serviço subsistema não sofre alterações, apenas é necessário que no seu relançamento seja indicado o novo serviço de comunicação responsável pela abstracção do novo *hardware*.

4. Detalhes de implementação

4.1 Demonstrador

A demonstração prática através da implementação de um demonstrador laboratorial é crucial para a avaliação do sucesso e estabilidade da solução proposta no capítulo anterior. Para um teste mais robusto à solução criou-se um cenário com diversos subsistemas envolvendo múltiplos protocolos e múltiplos fornecedores. Desta forma conseguiu-se criar um cenário laboratorial mais próximo da situação real.

Esta parte da dissertação teve o apoio da OMRON e da Schneider, empresas que cederam material para a montagem do demonstrador. O objectivo foi conseguir um cenário em que fosse demonstrado a cooperação entre os diversos componentes.

Antes de qualquer implementação da solução proposta, foi necessário estudar e programar todos os sistemas de forma isolada, em conformidade com as especificações do respectivo fabricante. Esse estudo e montagem está descrito nos capítulos seguintes.

4.1.1 Sistema de automação OMRON

A OMRON é uma empresa cujo seu *core bussiness* é a automação industrial (e.g. linha de montagem). A automação dos processos é fundamentalmente efectuada recorrendo a *Programmable Logical Controlers* (PLC). Estes são, na prática, microprocessadores que foram estruturados para serem programados e configurados sem que seja necessário programar o microprocessador directamente, não sendo necessário ter o conhecimento da linguagem de programação do microprocessador mas sim de uma linguagem simplificada que permite a programação do autómato. Isto trás vantagens pois o intuito é que pessoas sem formação em informática ou em electrónica consigam construir soluções de automação (e.g. electricistas).

A lista e respectiva descrição dos equipamentos disponibilizados pela OMRON para esta dissertação encontram-se no Anexo 1.

A programação destes PLC's é efectuada através de *software* proprietário, o *CX-Programmer*, e recorrendo a linguagem *ladder*. Esta linguagem é de representação

gráfica e análoga a um esquema eléctrico, encontra-se na Figura 23 um *print screen* do *software* de desenvolvimento.

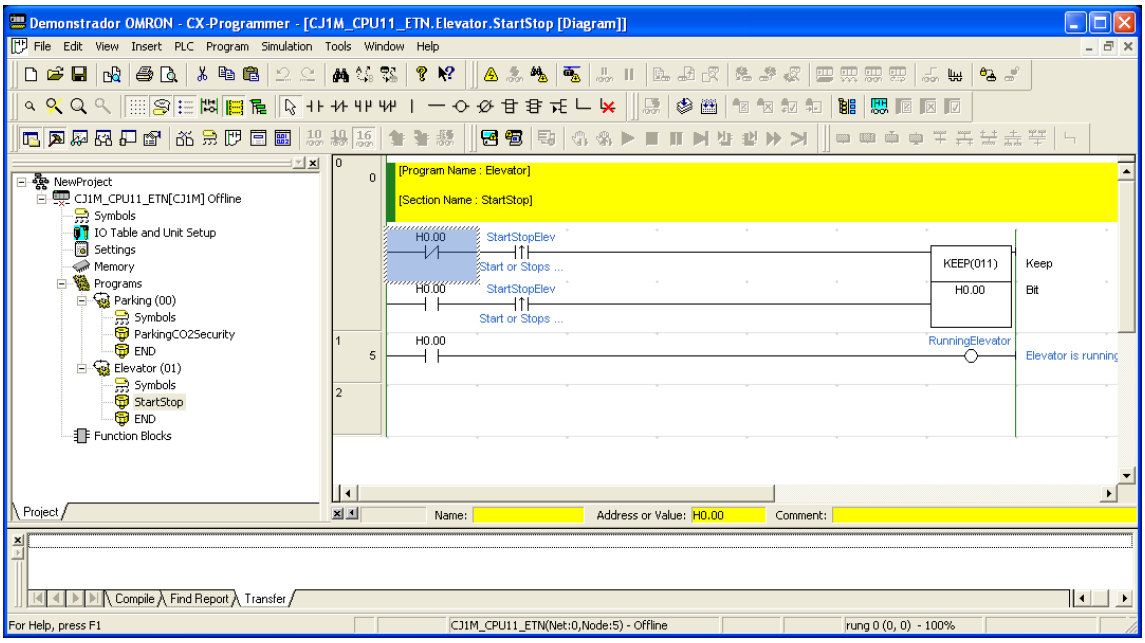
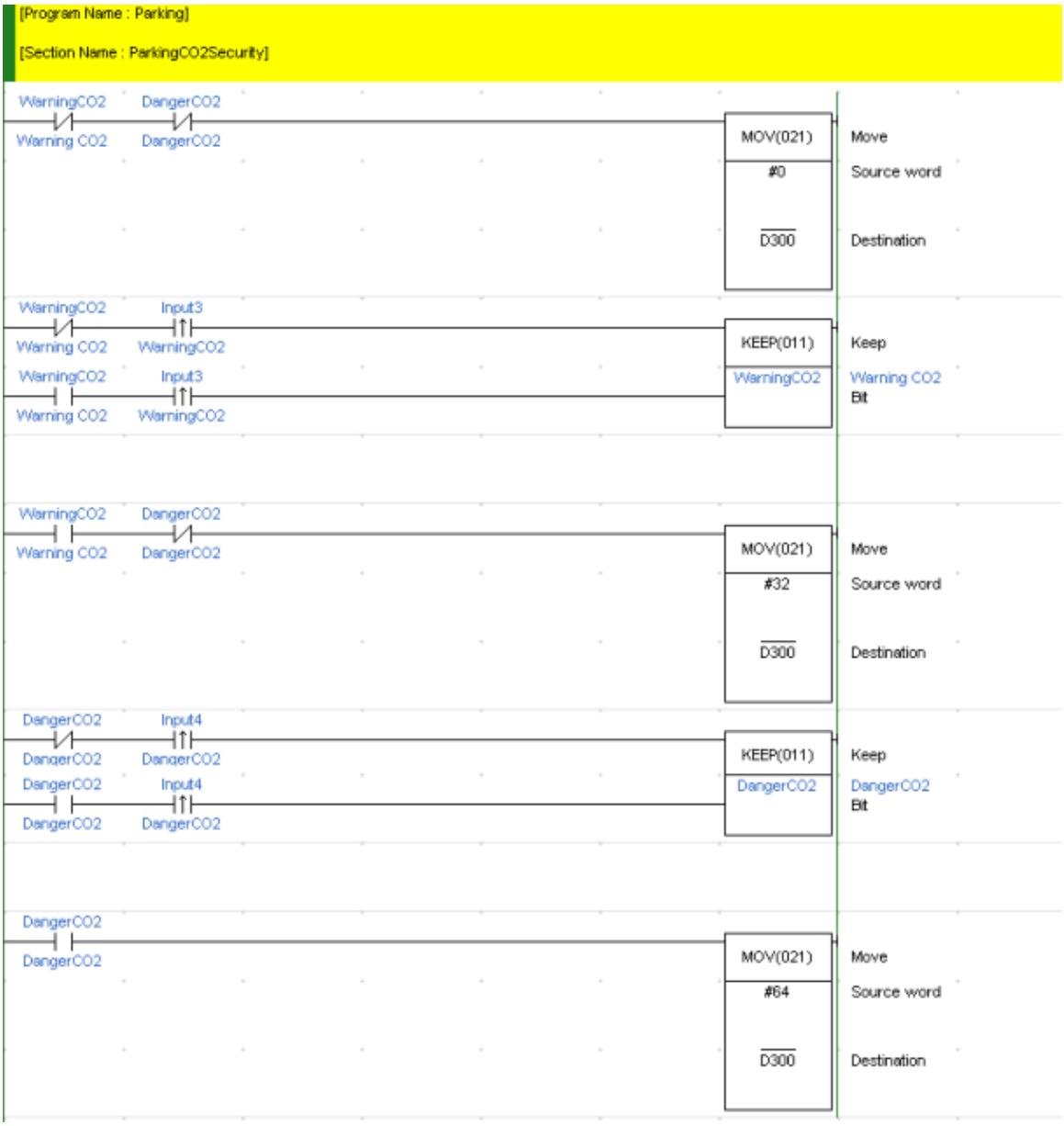


Figura 23 - CX-Programmer

Um exemplo da linguagem *ladder* que se mostra na Figura 23, simula o controlo do subsistema de elevadores, para o caso em que existe apenas um elevador, sendo possível através de uma variável de entrada fazer o seu bloqueio.

Outro subsistema que foi simulado utilizando o PLC da OMRON, foi um hipotético sistema de alarme de CO_2 num parque de estacionamento. A programação deste subsistema pode ser consultada na Figura 24. A programação foi realizada no pressuposto de que o sistema de controlo recebe, através de duas entradas, o valor do sensor de CO_2 e, dependendo desse valor, são executadas acções de controlo, que passam por aumentar a velocidade do motor do extractor de gases e/ou accionar o alarme de perigo.


[Program Name : Parking]
[Section Name : ParkingCO2Security]
WarningCO2
DangerCO2
Warning CO2
DangerCO2
MOV(021)
#0
D300
Move
Source word
Destination
WarningCO2
Input3
Warning CO2
WarningCO2
KEEP(011)
WarningCO2
Keep
Warning CO2
Bit
DangerCO2
Input4
DangerCO2
DangerCO2
#32
#64

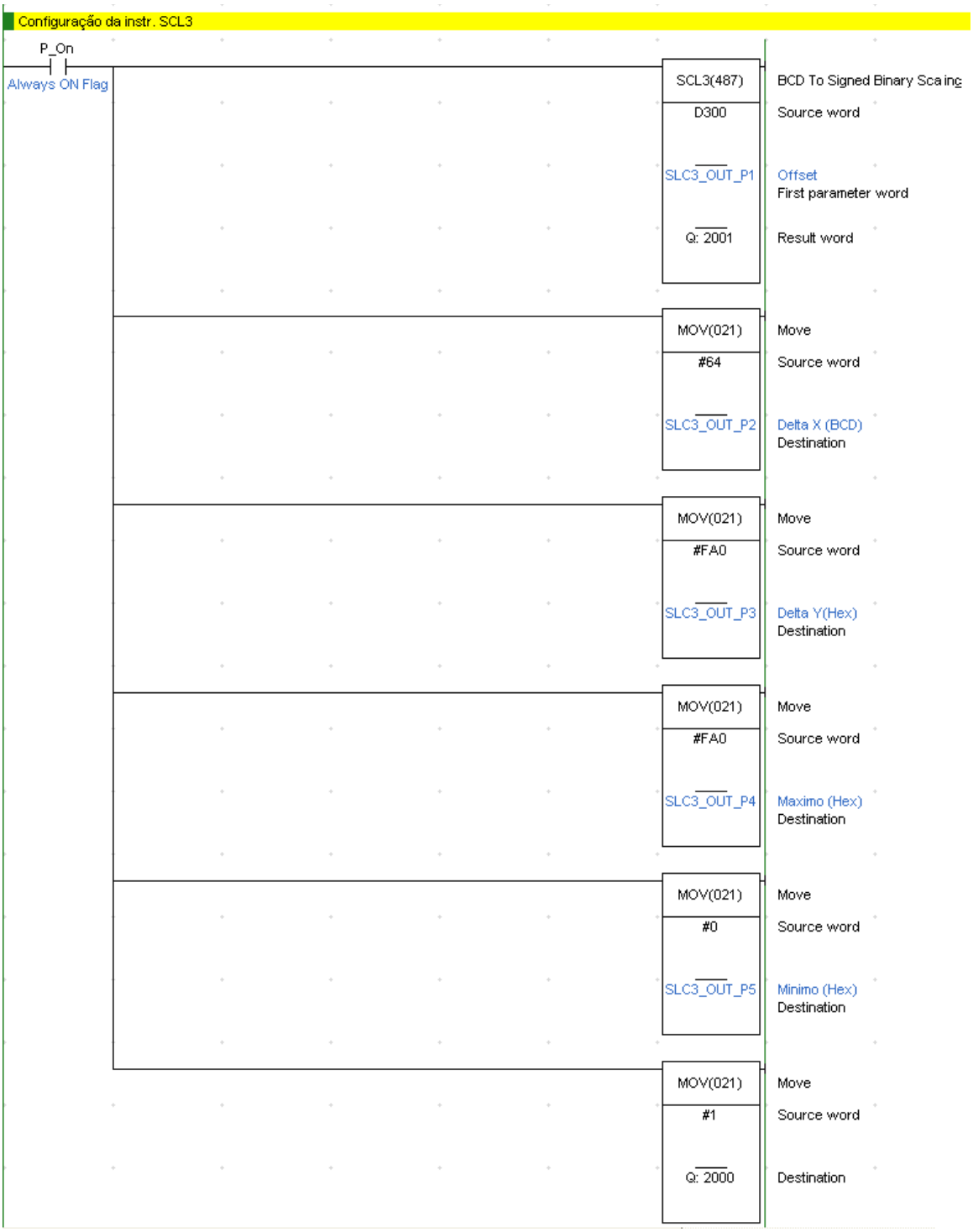


Figura 24 - Sistema de alarme de CO_2

4.1.2 Sistema de automação Schneider

A Schneider além de soluções de automação industrial, tem, através da sua subsidiária TAC, soluções específicas para a gestão técnica de edifícios. Estas soluções utilizam *Direct/Distributed Digital Controls* (DDC) que, ao contrário do PLC que centra o processamento num único ponto, esta abordagem subdivide o processamento pelos vários componentes utilizados na automação das diversas funcionalidades do edifício. Por exemplo, em vez de um único processador estar responsável pelo controlo de um edifício, este controlo pode ser efectuado recorrendo, por exemplo, a um processador por divisão e um processador "mestre" por piso para comunicação e controlo extra piso. Tem a vantagem de se minimizar a quantidade de cabos saídos de um ponto central e, no caso de avaria de um dos processadores, os outros não ficam comprometidos.

Os componentes utilizados no demonstrador laboratorial acima mencionado encontram-se descritos no Anexo 2.

A programação destes componentes foi efectuada através de um *software* proprietário da TAC Schneider. Todavia, como estes componentes estão desenvolvidos para a área de automação de edifícios, existe programação desenvolvida para os casos mais comuns. De qualquer forma, para uma solução mais personalizada é possível recorrer à programação *ladder*. Os diversos componentes, mesmo que já pré-programados, têm que ser associados, para que se reconheçam entre eles. Esta associação é efectuada recorrendo ao *software* proprietário TAC Vista Workstation (Figura 25).

Estes componentes implementam o protocolo LonWorks, apesar de neste protocolo não existir noção de mestre e escravo, neste caso o Xenta 731 é o componente que tem a interface IP e que faz a gestão das comunicações e controlo dos componentes com o exterior, ligados via MS/TP, tal como se pode verificar na arquitectura física presente na Figura 26.

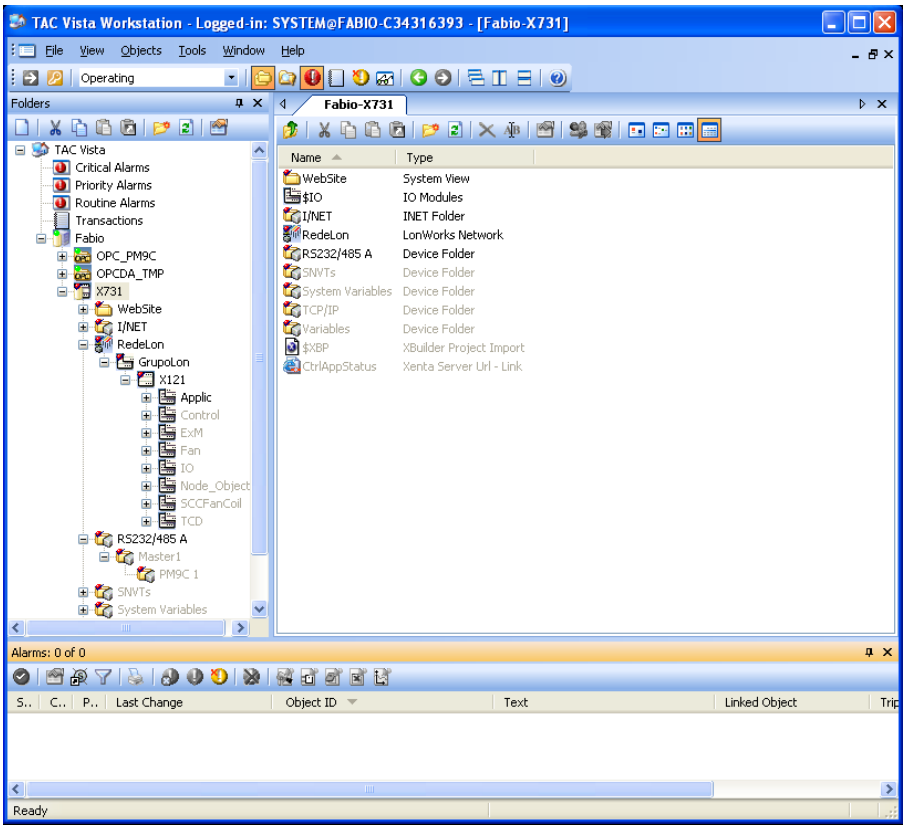


Figura 25 - TAC Vista Workstation

4.1.3 Desenvolvimento

O desenvolvimento do demonstrador laboratorial foi efectuado de forma independente entre subsistemas. Toda a comunicação de cooperação é efectuada através de Ethernet e controlada pelo BIOS. Na Figura 26 é apresentada a arquitectura física dos componentes do protótipo.

Na Figura 27 encontra-se uma fotografia da bancada de testes, em que além do material listado acima, foi também utilizado um hub para fazer a ligação entre os dois sistemas e o computador, uma *breadboard* para fazer alguns testes e duas fontes de alimentação, uma DC e outra AC.

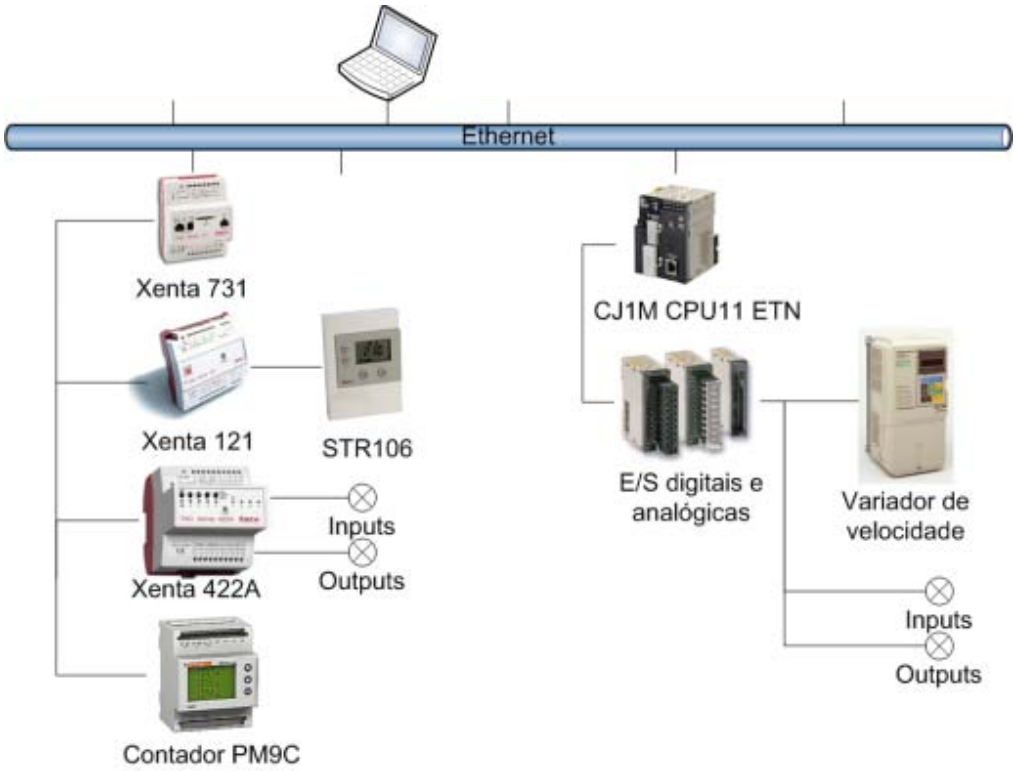


Figura 26 - Arquitectura Física

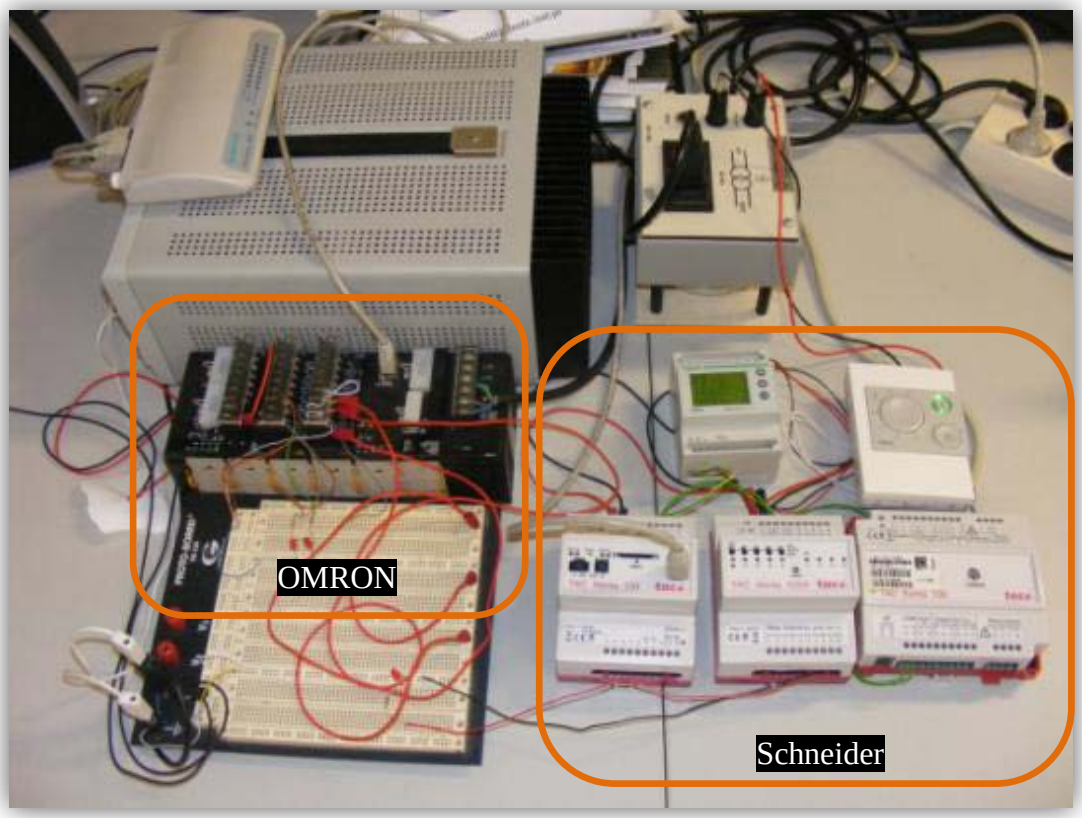


Figura 27 - Bancada de testes

4.2 Desenvolvimento da arquitectura

O desenvolvimento da arquitectura foi efectuado recorrendo à plataforma Java, podendo dessa forma tirar o partido de algumas iniciativas *open source*, nomeadamente o Jini e o JavaSpaces.

O desenvolvimento da arquitectura, tal como na arquitectura proposta, está dividido em quatro partes: bus cooperativo, serviços de comunicação, serviços de subsistema e clientes. Na Figura 28 apresenta-se a arquitectura implementada, a qual inclui todas estas partes: o bus cooperativo a laranja; os serviços de comunicação a azul; os serviços de subsistema a verde, incluindo todos os parâmetros que cada subsistema permite monitorizar e/ou controlar; e os clientes a vermelho.

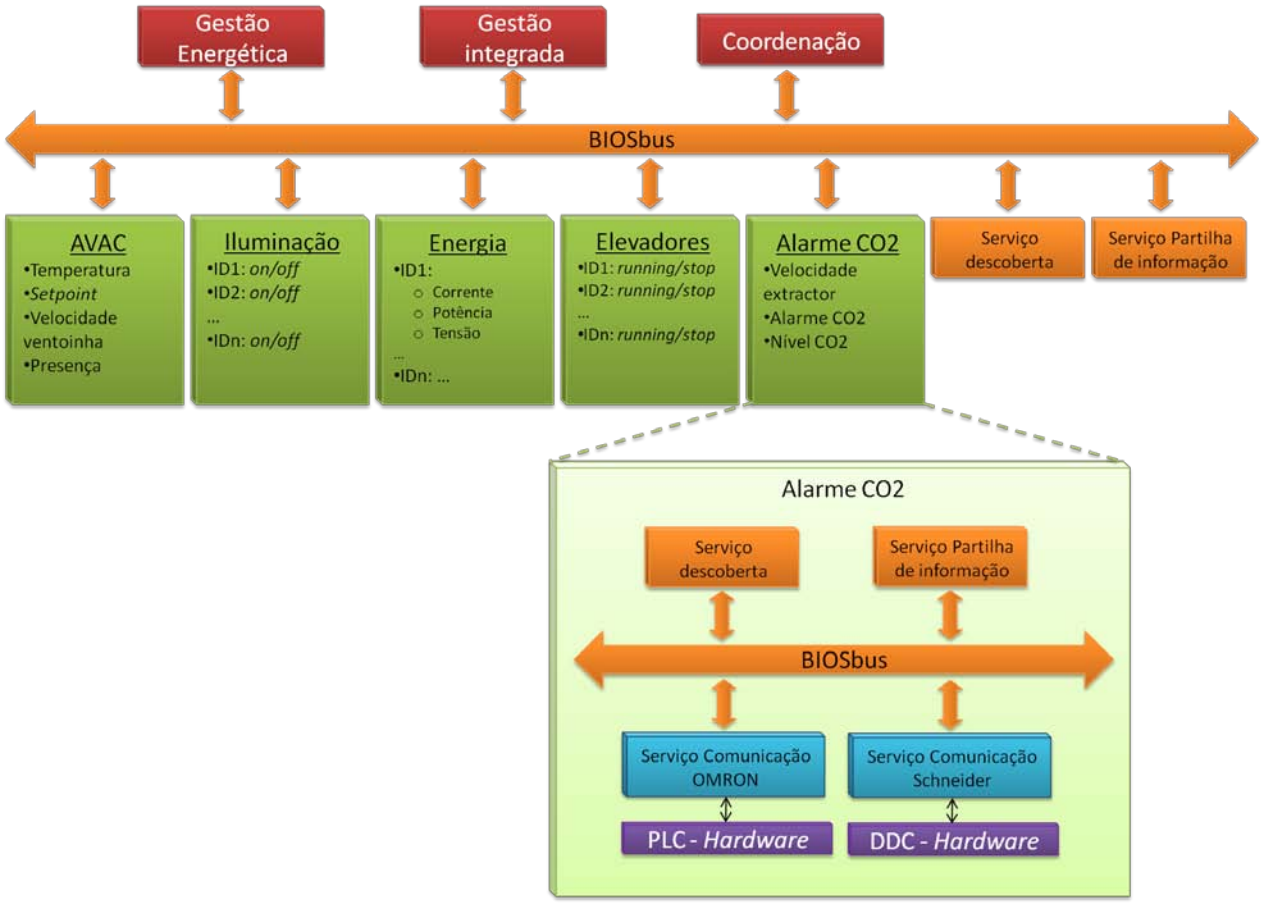


Figura 28 - Arquitectura implementada

4.2.1 Bus cooperativo

Para a implementação do bus cooperativo recorreu-se a duas tecnologias *open source*, Jini e JavaSpaces.

4.2.1.1 Jini

Jini é o nome de um ambiente de computação distribuída que potência a lógica *plug-and-play*, o que significa que um serviço pode-se conectar à rede e anunciar a sua presença, e os clientes que pretendem usar esse serviço podem localizá-lo e executar as funcionalidades por ele oferecidas [35]. Isto é possível pois o Jini suporta mecanismos de registo e descoberta automáticos.

Um sistema Jini é uma colecção de clientes e serviços que comunicam através de protocolos Jini. Isto é possível através da especificação de uma infra-estrutura (*middleware*) que incluí um *Application Programming Interface* (API) para que um programador possa desenvolver serviços e componentes que façam uso deste *middleware*.

O Jini está a ser desenvolvido no âmbito do programa Apache River[36], tendo sido lançada em Março de 2010 a versão 2.1.2.

4.2.1.2 JavaSpaces

A linguagem de coordenação Linda foi proposta por David Gelernter em 1985. Uma implementação Linda é basicamente uma colecção de tuplos, alguns desses incorporam código a executar e outros, que são os que têm interesse para esta dissertação, apenas têm informação passiva. Esta colecção de tuplos é conhecida como *tuple space* ou TS [37]. A linguagem Linda propõe um conjunto de operações simples, existindo dois tipos, um é utilizado para colocar tuplos no TS (*out*) e outro para obter os tuplos do TS (*in* - além de ler remove o tuplo do TS; *read* - apenas retorna uma copia do tuplo).

A linguagem de coordenação Linda tem mais valias para o objectivo desta dissertação quando comparada com outros tipos de sistemas distribuídos. Esta permite um desacoplamento temporal [37], podendo um recurso ser colocado no TS sem que o seu consumidor já esteja pronto para o consumir. O facto de toda a comunicação passar pelo TS permite que os intervenientes do sistema não sejam conhecidos entre eles.

O JavaSpaces é um sistema distribuído baseada na linguagem de coordenação Linda [38], e a sua implementação recorre a diversas tecnologias da Sun. Além de fazer parte do sistema Jini faz extensão à API Jini. O suporte da rede é fornecido pelo protocolo Java RMI (*Remote Method Invocation*). A distribuição das classes para os diversos clientes é feita através de *Hypertext Transfer Protocol* (HTTP) através de mecanismos de *Code Base* [38].

Os tuplos podem ser criados pelo programador desde que a classe implemente a interface Entry do Jini. A pesquisa de tuplos no TS, já prevista no projecto Linda, é feita tendo apenas em consideração os campos públicos dessa implementação. Para fazer a pesquisa criam-se anti-tuplos (em JavaSpaces são templates) para posteriormente ser feita a correspondência com os tuplos no TS [38].

O JavaSpaces oferece ainda um mecanismo de registo e subscrição de eventos, em que o objecto registado é notificado sempre que exista uma escrita que corresponda à Entry para a qual pretende ser notificado.

4.2.2 Serviços de comunicação

O serviço de comunicação disponibilizado implementa a interface `ICommunication` descrita na Figura 29. A comunicação tem uma interface simples em que as interacções com os componentes são efectuadas através de métodos `setTag` e `getTag`, em que apenas alteram ou obtêm o valor de determinado endereço. Para o caso do componente a interagir ser do tipo booleano existe a necessidade, em alguns sistemas, nomeadamente nos PLCs, de ter a informação de qual o endereço do bit a interagir, daí existir um dos métodos `getTag` e `setTag` específico para responder a esta necessidade.

É possível registar um *listner* para ser notificado cada vez que exista alterações em determinado componente. O método criado para esse efeito é o `monitorDataChange`. Com esta funcionalidade é possível evitar que as aplicações tenham que implementar um sistema de *polling* para verificação das alterações.

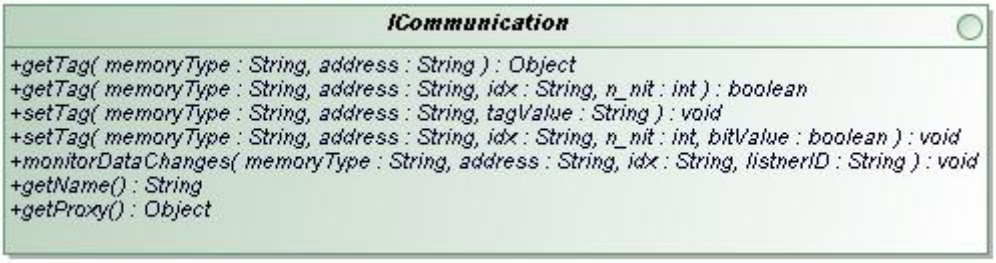


Figura 29 - ICommunication

Cada *hardware* tem as suas necessidades e especificação para o seu controlo e monitorização por parte de uma entidade externa, no caso dos implementados tem-se os protocolos OPC-UA e FINS para a comunicação entre a aplicação e o *hardware*.

O código responsável pelo lançamento dos serviços de comunicação pode ser visualizado na Listagem 1. O código utilizado para o lançamento de serviços é semelhante entre os restantes implementados, as diferenças residem no tipo de serviço a ser lançado, neste caso o tipo é `ICommunication` onde são lançados os serviços de comunicação da OMRON e da Schneider.

```
public static void main(String[] args) throws RemoteException {
   LinkedList<ICommunication> communications=new
   LinkedList<ICommunication>();

   communications.add(new CommunicationOMRON());
   communications.add(new CommunicationSchneider());

   new CommunicationService(communications);

   Object keepAlive = new Object();
   synchronized (keepAlive) {
      try {
         keepAlive.wait();
      }
      catch (InterruptedException e) {  e.printStackTrace();}
   }
}

public CommunicationService(LinkedList<ICommunication> communications) throws
RemoteException {
   System.out.println("Em Construtor de CommunicationService...");
   if ( System.getSecurityManager()==null ) {
      System.setSecurityManager(new RMISecurityManager());
   }

   LookupDiscovery lookupDiscovery=null;
   try {
      for(ICommunication communication : communications){
         serviceProxySimpleCommunication.add(communication);
```

```
        lookupDiscovery = new LookupDiscovery(
                        new String[]{"Communications"});
        joinManagers.add(new JoinManager(communication.getProxy(),
                        new Entry[]{new Name(communication.getName())},
                        this,lookupDiscovery,leaseRenewalManager));
      }
    } catch (RemoteException e1) {
      e1.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }
    System.out.println("Serviços lançados");
}
```

Listagem 1 - Lançamento serviços de comunicação

4.2.2.1 Serviço de comunicação Schneider

Os componentes instalados têm autonomia na sua gestão e controlo. Todavia, para a monitorização e controlo do sistema por parte de uma entidade externa, estas operações têm que ser realizadas através do componente Xenta 731, que além de ser um controlador, também é um *webserver* que de forma proprietária disponibiliza acções sobre os componentes.

Visto que o *webserver* apenas permite monitorização do valor das variáveis envolvidas no sistema de controlo, disponibilizando apenas páginas de internet previamente programadas pelo instalador, foi necessário encontrar outra solução, para interagir com o sistema. A solução encontrada foi o OPC UA, em que o servidor é o Xenta 731 e através de um SDK (*Software Development Kit*) de cliente OPC UA em Java facilita a implementação do serviço de comunicação.

A Schneider apenas disponibiliza um servidor OPC clássico (OPC DA), e isso implica recorrer a um *Wrapper* (Figura 30), já previsto na migração para o novo OPC UA [33], que encapsulasse o servidor DA num servidor UA. Recorreu-se à versão de avaliação de um *Wrapper* disponibilizado online pela empresa Unified Automation, de nome UaGateway [39]. Desta forma fica resolvida a parte do servidor OPC UA, pois a estrutura e informação de todos os componentes já existente no Xenta 731 mantêm-se.

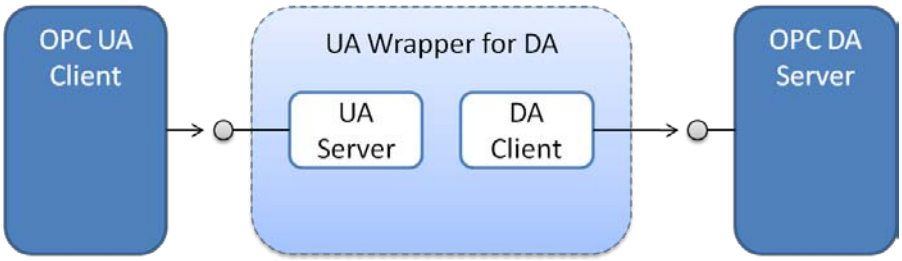


Figura 30 - UA wrapper que permite o acesso a servidores OPC DA [33]

O cliente OPC UA foi desenvolvido recorrendo ao SDK disponibilizado pela empresa Prosys, de nome Prosys OPC UA Java SDK [40] e é este cliente que vai assegurar a implementação do serviço de comunicação Schneider.

No lançamento do serviço de comunicação o cliente liga-se ao servidor OPC UA, e as implementações dos métodos `getTag` e `setTag` utilizam directamente os métodos disponibilizados pelo SDK `readValue` e `writeValue`. Estes métodos recebem um `NodeId`, que é o identificador de cada componente no servidor OPC UA. A implementação destes métodos encontram-se na Listagem 2.

```
@Override
public Object getTag(String urlStr, String description) throws
RemoteException{
   try {
      return
client.readValue(getNodeId(urlStr,description)).getValue().getValue();
   } catch (ServiceException e) {
      e.printStackTrace();
      throw new RemoteException("ServiceException",e.getCause());
   } catch (StatusException e) {
      e.printStackTrace();
      throw new RemoteException("StatusException",e.getCause());
   }
}

@Override
public void setTag(String urlStr, String description, String tagValue) throws
RemoteException{
   try {
      client.writeValue(getNodeId(urlStr,description), tagValue);
   } catch (ServiceException e) {
      e.printStackTrace();
      throw new RemoteException("ServiceException",e.getCause());
   } catch (StatusException e) {
      e.printStackTrace();
      throw new RemoteException("StatusException",e.getCause());
   }
}
```

Listagem 2 - Métodos `getTag` e `setTag` - Schneider

Para a notificação de alteração de estado de determinado elemento é necessário no registo, identificar o `Node` que se quer acompanhar e o `ListnerID` a ser notificado quando existirem alterações. Recorre-se a uma funcionalidade semelhante disponibilizada pelo servidor OPC UA que permite a subscrição de notificação de alterações em determinado `Node`. Essas notificações são feitas para um `MonitoredDataItemListener`, classe disponibilizada pelo SDK. Quando existem alterações dos valores do `Node` é chamado o evento `onDataChange`. Foi feita uma re-implementação deste evento para que seja identificado o `listenerID` associado ao `Node` que sofreu a alteração. A implementação do método `monitorDataChange` encontra-se na Listagem 3.

Identificado qual o `NodeId` e `ListnerID` associado é criado um `DataChangeEntry` com o `listnerID` e é colocado no JavaSpace, terminando aqui a intervenção do serviço de comunicação. Desta forma não é necessário ter conhecimento de que entidade fez o registo, apenas o seu `listnerID`, pois esta estará registada no JavaSpace para ser notificada quando for introduzida um `DataChangeEntry` com o seu `listnerID`. Todo este processo está esquematizado na Figura 31, incluindo a posterior notificação do cliente por parte do subsistema.

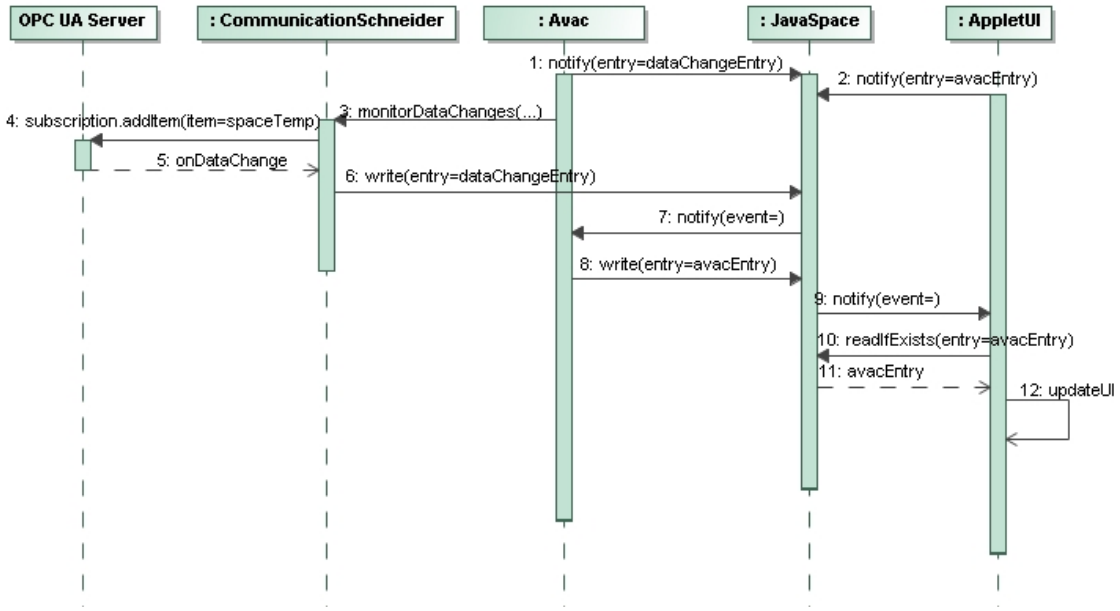


Figura 31 - Sequência de notificação de alterações - Schneider

```
public void monitorDataChanges(String urlStr, String description,
     String idx, String listnerID) throws RemoteException {
   MonitoredDataItem item;
   try {
      NodeId nodeId=getNodeId(urlStr, description);
      item = new MonitoredDataItem(nodeId, Attributes.Value,
           MonitoringMode.Reporting);
      subscription.addItem(item);
      item.addChangeListener(dataChangeListener);
      listnersList.put(nodeId,listnerID);

   } catch (ServiceException e) {
      e.printStackTrace();
      throw new RemoteException("ServiceException",e.getCause());
   } catch (StatusException e) {
      e.printStackTrace();
      throw new RemoteException("StatusException",e.getCause());
   }
}
private MonitoredDataItemListener dataChangeListener =
   new MonitoredDataItemListener() {
   @Override
   public void onDataChange(MonitoredDataItem sender,
             DataValue prevValue,
             DataValue value)
   {
      MonitoredItem i = sender;
      String listnerID=listnersList.get(i.getNodeId());

      try {
         javaSpace.write(new DataChangedEntry(
                   listnerID),
                   null,
                   1000*60);
      } catch (RemoteException e) {
         e.printStackTrace();
      } catch (TransactionException e) {
         e.printStackTrace();
      }
   }
};
```

Listagem 3 - Método `monitorDataChanges` - Schneider

4.2.2.2 Serviço de comunicação OMRON

A OMRON disponibiliza uma forma simplificada para a interacção e monitorização do PLC por parte de uma entidade externa, sendo as correspondentes operações asseguradas através de um protocolo proprietário, de nome FINS, através da porta Ethernet. Este protocolo tem comandos pré estabelecidos representados através de códigos em hexadecimal, estando todos esses códigos na documentação [41]. Os comandos utilizados são construídos dependendo das necessidades: leitura ou escrita,

espaço de memória pretendido, *bit* ou *word*, endereço e dados. Para o envio destes comandos é utilizando o protocolo UDP.

O envio de comandos resulta sempre numa resposta, essa também em código hexadecimal e com necessidade de ser tratada para interpretação de mais alto nível.

Os métodos `getTag` e `setTag` apenas necessitam de construir o comando pretendido e envia-lo via UDP e, para o caso do `getTag`, a resposta tem de ser interpretada para poder ser devolvida. A implementação destes métodos encontram-se na Listagem 4.

```
@Override
public String getTag(String memoryType, String address) throws
RemoteException{
   String finsCmd=
      FinsProtocolInfo.getCommandHeader()+
      FinsProtocolInfo.read_mem+
      ((memoryType.equals("CIO"))?FinsProtocolInfo.cio_bit:"82")+
      address+
      "00"+
      FinsProtocolInfo.NUM_ITEMS_1;
   String answer;
   try {
      answer = CommunicationManager.executeCmd(finsCmd);
   } catch (SocketException e) {
      throw new RemoteException("SocketException",e);
   } catch (IOException e) {
      throw new RemoteException("IOException",e);
   }
   String state = ResponseManager.decryptFinsResponse(answer);
   Integer stateValue=Integer.parseInt(state,16);
   return stateValue.toString();
}
@Override
public void setTag(String memoryType, String address, String tagValue)
      throws RemoteException {
   String finsCmd=
      FinsProtocolInfo.getCommandHeader()+
      FinsProtocolInfo.write_mem+
      FinsProtocolInfo.cio_word+
      address+
      "00"+
      FinsProtocolInfo.NUM_ITEMS_1+
      tagValue;
   try {
      CommunicationManager.executeCmd(finsCmd);
   } catch (SocketException e) {
      throw new RemoteException("SocketException",e);
   } catch (IOException e) { throw new RemoteException("IOException",e);}
}
```

Listagem 4 - Método `getTag` e `setTag` - OMRON

Como o PLC não fornece nenhum mecanismo de subscrição para notificação de alterações de estado de algum espaço de memória sem que seja necessária uma intervenção na programação do próprio autómato, o método `monitorDataChanges` não foi implementado da mesma forma que para o equipamento da Schneider. Neste caso é necessário que seja o próprio serviço de comunicação a fazer a verificação de alterações através de *polling*. Todo o restante processo é semelhante ao acima descrito, tal como se pode verificar na Figura 32. O código desenvolvido para efectuar o processo indicado encontra-se na Listagem 5.

```
@Override
public void monitorDataChanges(String memoryType, String address,
     String idx, String listnerID) throws RemoteException {
   State state=new State(memoryType,address,idx,listnerID,null);
   prevState.add(state);
}

public void startJavaSpaces(){
   new Thread(){
      public void run(){
         javaSpace = (JavaSpace) FindService.lookup(JavaSpace.class);
         while(true){
            try{
               verifyChanges();
            } catch (RemoteException e) {
               System.err.println("ERROR: RemoteException in "+name+"
                                    verifyChanges ("+e.getMessage()+")");
               return;
            }
            try {
               Thread.sleep(500);
            } catch (InterruptedException e) {e.printStackTrace();}
         }
      }
   }.start();
}
private void verifyChanges() throws RemoteException {
   for(State state:prevState)
   {
      String currState=(state.idx.equals(""))?
            getTag(state.memoryType, state.address):
               String.valueOf(getTag(state.memoryType, state.address,
state.idx, 1));

      if(state.state==null || !currState.equals(state.state))
      {
         try {
            javaSpace.write(new DataChangedEntry(state.listnerID), null,
1000*60);
         } catch (RemoteException e) {e.printStackTrace();
         <restante código não relevante>
}
```

Listagem 5 - Método `monitorDataChanges` - OMRON

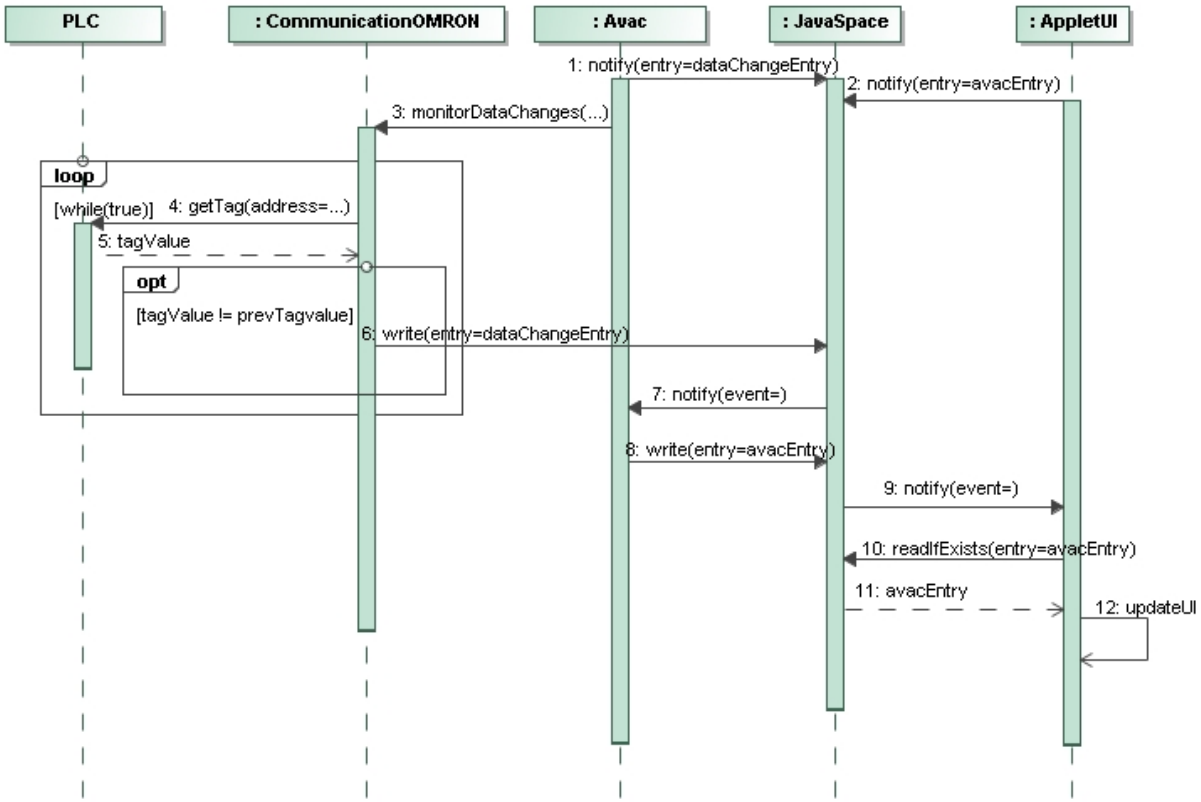


Figura 32 - Sequência de notificação de alterações - OMRON

4.2.3 Serviços de Subsistema

Os serviços de subsistema que estão implementados são o AVAC, iluminação, energia, elevadores e sistema de alarme de CO_2 do parque de estacionamento. Todos estes subsistemas são apenas de teste, em que todas as operações são efectuadas através do *hardware* embora tendo um formato simplificado do que seria a implementação real, não a nível das acções de controlo implementadas, mas sim na quantidade de variáveis controladas.

Os subsistemas são constituídos por diversos componentes, desde sensores, actuadores, etc. As funcionalidades básicas (`getState`, `setState`, `monitorComponentChanges`) dos componentes estão encapsuladas nas classes `Component`, simplificando assim a sua utilização nas implementações dos subsistemas. Existem implementados dois tipos de componentes, os que representam componentes do tipo *bit* (`boolean`) e os do tipo *word* (`Object`) Ainda nestes dois tipos faz-se a distinção entre componentes que são apenas de leitura ou se são também de escrita. Como por exemplo um sensor, que é apenas um

componente de leitura, só existindo a necessidade de ficar disponível a operação de leitura (`getState`). É nas classes `Component` que é efectuada a procura do serviço de comunicação, sendo passado no construtor todas as informações necessárias para a procura do serviço e respectivos endereços do componente, desta forma abstrai a implementação do serviço de subsistema dessa procura bem como da utilização directa dos endereços do componente.

Em qualquer um dos serviços de subsistema são monitorizadas as alterações de estado dos componentes, sendo o registo dessas notificações oferecido pelos serviços de comunicação, tal como descrito no capítulo anterior, sendo utilizado o nome do serviço como `listnerID`. Sempre que é detectada uma alteração é criado uma Entry específica para cada subsistema que é posteriormente colocada no JavaSpace, dando suporte a funcionalidade do bus que permitir que clientes se registem em determinado subsistema para ser notificados de alterações.

4.2.3.1 Serviço de subsistema AVAC

Este serviço disponibiliza o controlo e monitorização do subsistema de AVAC do edifício, permitindo monitorizar temperatura real (`spaceTemp`), velocidade da ventilação actual (em percentagem)(`fanSpeed`), a temperatura pretendida (`setpoint`). Este subsistema permite ainda a monitorização e controlo da ocupação do espaço (`occupancyRoomUnit`), tendo os seguintes estados possíveis: ocupado, desocupado e em espera. Este serviço implementa a interface `IAvac` (Figura 33) que oferece os métodos necessários para todas as interacções descritas acima.

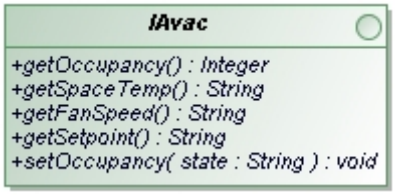


Figura 33 - IAvac

Na Listagem 6 encontra-se a implementação do serviço do subsistema AVAC, é possível verificar os diversos componentes que fazem abstracção das funcionalidades acima descritas, o registo no serviço de notificação de alterações (`monitorComponentChanges`) e as implementações do métodos presentes na interface

descrita anteriormente. As implementações dos métodos tornam-se programaticamente simples a este nível, sendo em alguns casos apenas necessário invocar o método `getState` do componente, pois tudo o restante já se encontra abstraído pelo componente mas principalmente pelo serviço de comunicação.

As implementações dos restantes serviços de subsistema são assemelham-se a esta estrutura aqui apresentada.

```
public class Avac extends AbstractSubsystem implements IAvac,
RemoteEventListener{
   ComponentWordReadOnly spaceTemp;
   ComponentWordReadOnly setpoint;
   ComponentWordReadOnly fanSpeed;
   ComponentWordReadOnly occupancyRoomUnit;
   ComponentWord occupancyManual;

   AvacEntry previousState;

   <...>

   public void startJavaSpaces(){
      new Thread(){
         public void run(){
            javaSpace = (JavaSpace) FindService.lookup(JavaSpace.class);
            try {

            spaceTemp.monitorComponentChanges(name);
            setpoint.monitorComponentChanges(name);
            fanSpeed.monitorComponentChanges(name);
            occupancyRoomUnit.monitorComponentChanges(name);
            occupancyManual.monitorComponentChanges(name);

            javaSpace.notify(new DataChangedEntry(name), null,
(RemoteEventListener) proxy, Lease.FOREVER, null);

            } catch (...) { <...>}
            try {
               creatAndAddEntryOnJS();
            } catch (...) { <...>}       }
      }.start();
   }

   @Override
   public String getFanSpeed() throws RemoteException {
      return fanSpeed.getState().toString();
   }

   @Override
   public String getSetpoint() throws RemoteException {
      return setpoint.getState().toString();
   }

   @Override
```

```
   public String getSpaceTemp() throws RemoteException {
      return spaceTemp.getState().toString();
   }
   @Override
   public Integer getOccupancy() throws RemoteException {
      int occMan=(Integer)occupancyManual.getState();
      if (occMan<=3 && occMan>=0)
         return occMan;
      return (Integer)occupancyRoomUnit.getState();
   }
   @Override
   public void setOccupancy(String state) throws RemoteException {
      occupancyManual.setState(state);
   }
   <...>
}
```

Listagem 6 - Implementação AVAC

4.2.3.2 Serviço de subsistema Iluminação

Este serviço disponibiliza o controlo e monitorização do subsistema de iluminação do edifício, permitindo que, a cada ponto de iluminação, esteja associado um identificador. Isto possibilita que a cada identificador seja associada uma divisão ou uma zona da divisão, ou mesmo um piso inteiro. Este serviço implementa a interface `ILighting` (Figura 34) a qual oferece os métodos necessários para todas as interacções descritas acima.

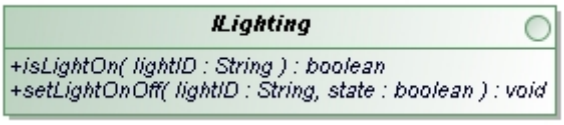


Figura 34 - ILighting

4.2.3.3 Serviço de subsistema Energia

Este serviço disponibiliza a monitorização do subsistema de energia do edifício, este subsistema é importante para posteriormente ser utilizado em algoritmos de gestão racional de energia eléctrica. À semelhança do subsistema de iluminação este oferece a possibilidade de que a monitorização seja efectuada a diversos pontos consumidores de energia eléctrica. As grandezas que são medidas neste subsistema são a corrente, tensão e potência activa. Este serviço implementa a interface `IEnergy` (Figura 35) que oferece os métodos necessários para todas as interacções descritas acima.

Figura 35 - IEnergy

4.2.3.4 Serviço de subsistema Elevadores

O serviço elevadores implementado, constitui uma versão simplificada uma vez que apenas permite o controlo se determinado elevador se encontra em funcionamento ou não. Isto é útil para efectuar o deslastre de cargas quando o algoritmo de gestão racional de energia eléctrica assim o exigir. A cada elevador está associado um identificador, podendo por isso efectuar-se operações sobre um elevador específico. Este serviço implementa a interface `IElevators` (Figura 36) que oferece os métodos necessários para todas as interacções descritas acima.

Figura 36 - IElevators

4.2.3.5 Serviço de subsistema Alarme de CO_2 de um parque de estacionamento

O sistema de alarme de CO_2 é um subsistema comum nos parques de estacionamento. Dependendo dos níveis de CO_2 obtidos através do correspondente sensor, poderá eventualmente ser accionado o motor do extractor de gases e dependendo do nível de perigo, accionar um hipotético alarme. O serviço que representa este subsistema faz monitorização destes três componentes: sensor CO_2, velocidade do extractor (em percentagem) e alarme, mas tal como mencionado no Capítulo 4.1.1 este subsistema prevê a possibilidade de o sensor de CO_2 seja externo, logo possibilita que o valor do sensor seja introduzido por parte de uma entidade externa através do método

`setExternalCO2Level`. Este serviço implementa a interface `IParkingCO2` (Figura 37) que oferece os métodos necessários para todas as interacções descritas acima.

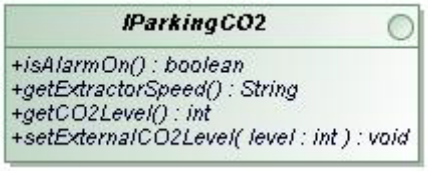


Figura 37 - IParkingCO2

O sensor é simulado no *hardware* através de dois interruptores, simulando três níveis de perigo (normal, aviso e perigo). Esta simulação do sensor é efectuada no *hardware* Schneider e os restantes componentes estão no *hardware* da OMRON, permitindo assim comprovar a interoperabilidade conseguida através desta arquitectura. Esta interoperabilidade é conseguida recorrendo ao serviço de coordenação descrito no capítulo seguinte (capítulo 4.2.4).

4.2.4 Serviço de coordenação

A entidade de coordenação garante a interoperabilidade do sistema, esta regista-se para ser notificada de alterações de determinado componente, o código necessário para este registo encontra-se na Listagem 7. O exemplo implementado é a coordenação do subsistema de alarme de CO_2 em que quando se verifica alterações no sensor de CO_2 controlado pelo *hardware* da Schneider, essas alterações são encaminhadas para o *hardware* da OMRON, para que através do algoritmo de automação já implementado no PLC, sejam executadas as correspondentes acções de controlo e alarme. Os intervenientes do exemplo implementado encontram-se representados na Figura 38.

```
public Coordinator(){

   if (System.getSecurityManager() == null) {
      System.setSecurityManager(new RMISecurityManager());
   }

   Exporter exporter = new BasicJeriExporter(
   TcpServerEndpoint.getInstance(0),new BasicILFactory());
   try {
      proxy = (RemoteEventListener) exporter.export(this);
   } catch (ExportException e) {
      e.printStackTrace();
   }
```

```
   javaSpace = (JavaSpace) FindService.lookup(JavaSpace.class);
   parkingCO2=(IParkingCO2)FindService.lookup(new Name("ParkingCO2"));
   prevLevel=0;

   try {
      parkingCO2EventID = javaSpace.notify(new ParkingCO2Entry(),
                     null, proxy,
                     Lease.FOREVER, null).getID();
   } catch (RemoteException e) {
         e.printStackTrace();
   } catch (TransactionException e) {
         e.printStackTrace();
   }
}
public void notify(RemoteEvent event) throws UnknownEventException,
      RemoteException {

   long eventID=event.getID();
   if(eventID==parkingCO2EventID){
      try {
         getAndRefreshFromJSParkingCO2();
      }

   <restante código não relevante>

}
```

Listagem 7 - Registo de notificação para coordenação

O subsistema ParkingCO2 é notificado, através dos respectivos serviços de comunicação, sempre que existe alguma alteração de estado dos seus componentes. Visto que o serviço Coordinator está registado no JavaSpaces para ser notificado sempre que existe alterações no subsistema ParkingCO2, este serviço ao receber uma notificação de alteração verifica se essa foi no sensor de CO_2 e em caso afirmativo utiliza o método do subsistema ParkingCO2 que permite colocar o valor do sensor por parte de uma entidade externa, neste caso essa alteração é propagada até ao *hardware* da OMRON e através das rotinas já implementadas no PLC vai colocar o extractor a velocidade correspondente e activar ou não o alarme. O diagrama de sequência presente na Figura 39 resume o processo descrito acima.

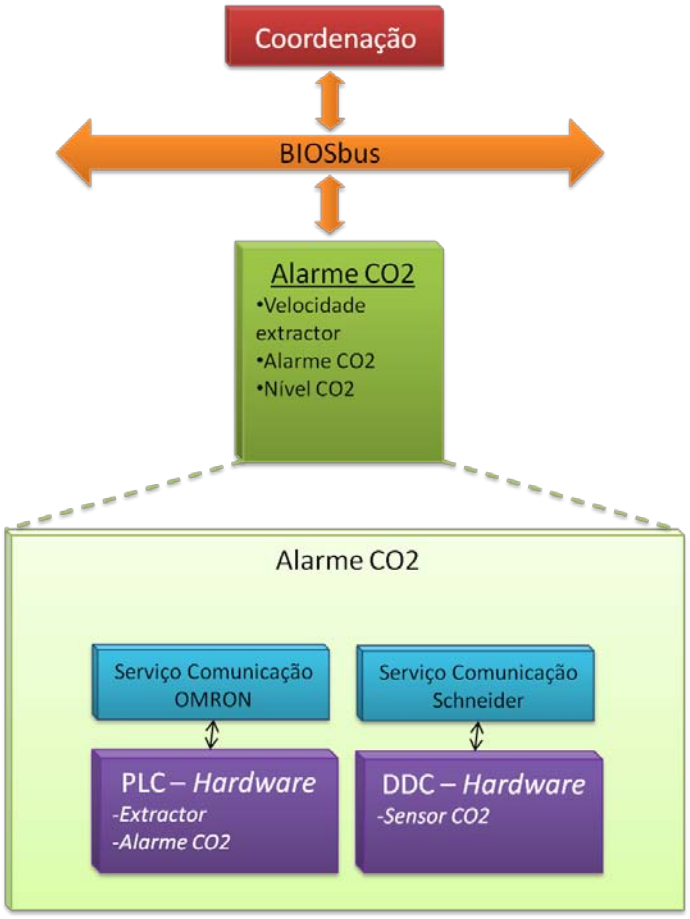


Figura 38 - Exemplo implementado

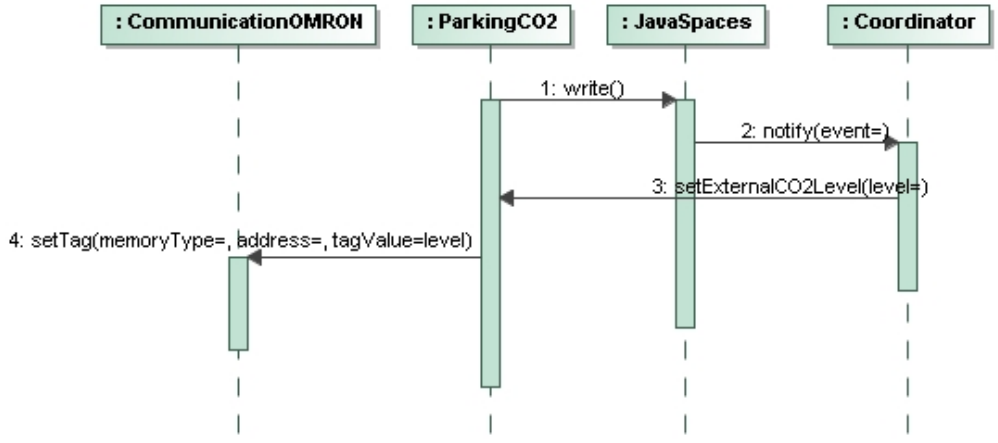


Figura 39 - Diagrama sequência exemplo implementado

Tal como descrito no capítulo anterior o sensor de CO_2 é simulado através de dois interruptores no hardware Schneider: quando não existe nenhum interruptor ligado é

enviado o valor 0 (normal) para o hardware OMRON em que do algoritmo de automação implementado resulta em parar o motor do extractor de fumos e desligar o alarme; com um interruptor ligado é enviado o valor 1 (aviso) que resulta em colocar o motor do extractor a 50% da sua velocidade máxima; com dois interruptores ligados é enviado o valor 2 (perigo) que resulta em colocar o motor do extractor na sua velocidade máxima e accionar o alarme de perigo (Listagem 8). Não confundir o que é responsabilidade do nível de automação, todas as decisões que dizem respeito à velocidade do extractor ou alarme de perigo são da responsabilidade do nível de automação. No nível de gestão, onde o BIOS se insere, é onde é conseguida esta cooperação, pois através do sistema de Coordenação consegue-se que um subsistema seja controlado por hardwares de fabricantes distintos ou mesmo integrar diferentes subsistemas tal como foi exemplificado no capítulo 3.

```
private void getAndRefreshFromJSParkingCO2()  throws RemoteException,
   UnusableEntryException, TransactionException, InterruptedException
{
   ParkingCO2Entry parkingCO2Entry = null;

   parkingCO2Entry = (ParkingCO2Entry)javaSpace.readIfExists(
      new ParkingCO2Entry(),null,1000*60*30);
   if(parkingCO2Entry!=null)
   {
      int level=parkingCO2Entry.co2Level;
      if(level!=prevLevel){
         switch(level){
         case 2:
            parkingCO2.setExternalCO2Level(2);
            break;
         case 1:
            if(prevLevel==0)
               parkingCO2.setExternalCO2Level(1);
            if(prevLevel==2)
               parkingCO2.setExternalCO2Level(2);
            break;
         case 0:
            parkingCO2.setExternalCO2Level(1);
            break;
         }
         prevLevel=level;
      }
   }
}
```

Listagem 8 - Algoritmo de coordenação

4.2.5 Serviço de gestão energética

A gestão energética é um serviço que possibilita, por exemplo, através da interface gráfica, a activação de algoritmos de gestão racional de energia eléctrica. O algoritmo implementado apenas verifica se o consumo de energia eléctrica (`power`) corresponde a 5% (`maxContractedEnergyInUse`) de se ultrapassar a potência contratada (`contractedEnergy`) e em caso afirmativo faz o deslastre de cargas. Neste caso coloca os elevadores (`elevators`) a garantir apenas os serviços mínimos, em que apenas um elevador fica em funcionamento. Na Listagem 9 encontra-se a implementação deste algoritmo.

Esta arquitectura potencia a criação de outros algoritmos, pois o acesso a determinado subsistema está facilitado, sendo apenas necessário recorrer ao serviço subsistema em causa para obter a sua informação e controlo.

```
float power = energyEntry.energyPoints.get(energyMainPointID).power;
if(power>=contractedEnergy*maxContractedEnergyInUse){
   String[] ids;
   try {
         ids = elevators.getAllIDs();
      for(String id:ids){
         if(!id.equals(mainsElevatorID)){
            if(!elevators.isLocked(id))
               elevators.Lock(id, true);
         }
      } catch (RemoteException e) {e.printStackTrace();}
}
```

Listagem 9 - Algoritmo de gestão energética

4.2.6 Interface Gráfica

A interface gráfica integra todos os subsistemas mencionados acima e ainda o serviço de gestão energética. Apesar do demonstrador ser composto por dois sistemas de controlo diferente, através do BIOS é possível integrar com facilidade todos os subsistemas numa única aplicação gráfica.

A aplicação implementada apenas tem conhecimento das interfaces dos serviços de subsistema, obtendo os serviços através do serviço de *Lookup* do Jini, e registando-se no serviço JavaSpaces para que seja notificada de alterações de estado para posterior actualização da interface gráfica. Estas notificações estão esquematizadas na Figura 31 e Figura 32.

Na Figura 40 encontra-se uma primeira versão da interface gráfica, não sendo o grafismo um elemento importante para esta dissertação, tentou-se que o aspecto fosse o melhor possível.

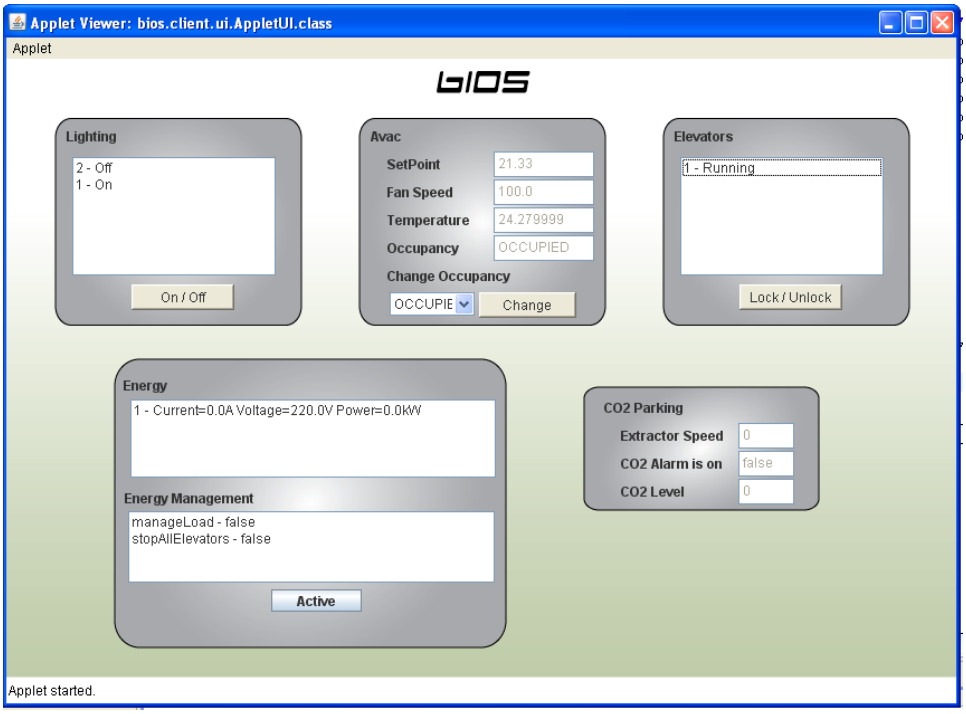


Figura 40 - Interface gráfica

5. Conclusões e Trabalho Futuro

O objectivo desta dissertação de Mestrado era o de estudar soluções que permitissem facilitar o desenvolvimento da integração de subsistemas utilizados na automação de um edifício inteligente. O ciclo de vida de um edifício está dividido em três partes, a elaboração do projecto, a sua implementação e ainda, a que fica até ao fim da vida do edifico, a sua exploração/gestão. Através da cooperação é possível facilitar as decisões e desenvolvimentos em todos os momentos do ciclo de vida do edifício, pois é possível reduzir redundâncias de *hardware*, optimizar os algoritmos de gestão racional de energia eléctrica e até maximizar o controlo sobre um edifício, conseguindo fazer a monitorização de todo o edifício através de uma única aplicação gráfica.

O estudo desenvolvido passou pela avaliação do Estado da Arte neste domínio do conhecimento. Numa primeira parte são descritas as mais valias de optar por um sistema aberto em vez de sistemas proprietários. Juntamente com esta comparação foram estudados e comparados três protocolos abertos BACnet, LonWorks e KNX. Na segunda parte são descritas iniciativas que tendem a resolver alguns dos factores críticos que dizem respeito a interoperabilidade. Nesta descrição foi dado ênfase ao oBIX, projecto cujos objectivos se aproximam do BIOS e ao OPC UA cujos conceitos vieram a ser utilizados para assegurar a comunicação com os componentes Schneider.

Posteriormente foi proposta uma arquitectura cujo objectivo era resolver o problema descrito. A solução adoptada foi uma arquitectura orientada a serviços. As várias responsabilidades, seja de comunicação entre *hardware* e arquitectura, ou abstracção de determinado subsistema físico, foram divididas pelos diversos serviços propostos. Todos estes serviços comunicam e interagem através de um bus, aproximando esta arquitectura da lógica *plug-and-play*, podendo facilmente ser adicionados novos serviços e os existentes os reconhecerem.

Foi construído um demonstrador recorrendo ao *hardware* cedido pelas empresas OMRON e Schneider. Esse demonstrador tenta que os testes da arquitectura proposta se aproximem ao máximo de um caso real. Analisado o material disponível foi desenvolvido um caso de estudo, em que foram criados subsistemas controlados por esse *hardware* e posteriormente integrados na arquitectura proposta. Esta arquitectura

foi concretizada para estes casos específicos, em que os subsistemas considerados foram o AVAC, a iluminação, a gestão racional de energia eléctrica, elevadores e sistema de monitorização de CO_2. Todos estes subsistemas têm representação no *hardware*. Podendo concluir que após a implementação da arquitectura proposta esta responde às necessidades e consegue atingir os objectivos inicialmente propostos, nomeadamente o factor mais relevante o da interoperabilidade.

No que diz respeito a propostas de trabalho futuro, esta dissertação poderia ser melhorada através da inclusão de:

- Carregamento dinâmico dos diversos subsistemas, tanto ao nível de serviços, como posteriormente o carregamento da interface gráfica;
- Dinamismo na interface gráfica, de forma a facilitar e melhorar a sua usabilidade para o utilizador;
- Autenticação do utilizador, recorrendo a um servidor *Lightweight Directory Access Protocol* (LDAP) de forma a restringir o acesso do utilizador a determinadas funcionalidades da interface gráfica;
- Teste da arquitectura num ambiente real, através da implementação da solução num edifício;
- Desenvolver algoritmos de gestão racional de energia eléctrica mais complexos, podendo por exemplo envolver agentes ou sistemas de apoio à decisão.

6. Bibliografia e referências

[1] **European Comission.** *Statistical Pocketbook 2009 - EU energy and transport in figures.* s.l. : European Communities, 2009.

[2] **Merz, Hermann, Hübner, Christof e Hansemann, Thomas.** *Building Automation - Communication Systems with EIB/KNX, LON and BACnet.* s.l. : Springer, 2009.

[3] **2M CCTV.** 2M CCTV. [Online] [Citação: 1 de Setembro de 2010.] http://www.2mcctv.com/.

[4] **Briere, Danny e Hurley, Pat.** *Smart Houses for dummies.* s.l. : Wiley Publishing, 2007.

[5] **Elsenpeter, Robert C. e Velte, Toby J.** *Build Your Own Smart Home.* s.l. : McGraw-Hill, 2003.

[6] **Siemens.** Siemens. [Online] [Citação: 1 de Setembro de 2010.] http://www.siemens.com.

[7] **Electronic House.** Systems - Lighting Control. [Online] [Citação: 1 de Novembro de 2009.] http://www1.electronichouse.com/info/systems/lighting.html.

[8] **Hordeski, Michael F.** *HVAC Control in the new millennium.* s.l. : The Fairmont Press, 2001.

[9] **Inc., Strata Resource.** *Investigating Open Systems - Comparing LonWorks and BACnet.* 2006.

[10] **Winkelman, Patrick.** Sustainable BAS. *Automated Buildings.* [Online] 2009. [Citação: 1 de Dezembro de 2009.] http://www.automatedbuildings.com/news/mar09/articles/distech/090219023638distech.htm.

[11] **Granzer, Wolfgang, Kastner, Wolfgang e Reinisch, Christian.** *Gateway-free Integration of BACnet and KNX using Multi-Protocol Devices.* s.l. : Vienna University of Technology, Automation Systems Group, 2008.

[12] **Frank, Randy.** *Understanding Smart Sensors.* s.l. : ARTECH HOUSE, INC, 2000.

[13] **Kastner, WOLFGANG, et al.** *Communication Systems for Building Automation and Control.* s.l. : IEEE, 2005.

[14] **Newman, H.M.** BACnet - The New Standard Protocol. *BACnet.* [Online] 1997. http://www.bacnet.org/Bibliography/EC-9-97/EC-9-97.html.

[15] **Thomas, George.** Connecting BACnet Devices to an IP Infrastructure. *Automated Buildings.* [Online] 2009. [Citação: 1 de Março de 2010.] http://www.automatedbuildings.com/news/mar09/articles/cctrls/090220020828cctrls.htm.

[16] **Bender, Joel e Newman, Mike.** BACnet/IP. *BACnet Tutorial.* [Online] [Citação: 1 de Março de 2010.] http://www.bacnet.org/Tutorial/BACnetIP/.

[17] **Swan, Bill.** The Language of BACnet - Objects, Properties and Services. *PolarSoft.* [Online] 2003. [Citação: 1 de Março de 2010.] http://www.polarsoft.biz/langbac.html.

[18] **Thomas, George.** Object Modeling a Physical BACnet Device. *Automated Buildings.* [Online] 2008. [Citação: 1 de Março de 2010.] http://www.automatedbuildings.com/news/aug08/articles/cctrls1/080724032404cctrls.htm.

[19] **Real Time Automation.** BACnet Overview - An Application Layer Protocol for Building Automation. *Real Time Automation.* [Online] [Citação: 1 de Abril de 2010.] http://www.rtaautomation.com/bacnet/.

[20] **LonMark International.** *Overview Handbook.* 2009.

[21] **Toshiba Corporation.** *Neuron Chip.* 2006.

[22] **Fernandes, Pedro Miguel de Miranda.** Aplicações Domóticas para Cidadãos com Paralisia Cerebral. *Biblioteca digital da Universidade de Aveiro.* [Online] [Citação: 1 de Dezembro de 2009.] http://portal.ua.pt/bibliotecad/default1.asp?OP2=0&Serie=0&Obra=28&H1=2&H2=4.

[23] **Echelon.** Neuron® 5000 Processor. *Echelon.* [Online] [Citação: 1 de Março de 2010.] http://www.echelon.com/products/neuron/default.htm.

[24] **Konnex.** *KNX System-architecture.* 2004.

[25] **Konnex Association.** *Introduction to KNX.* 2004.

[26] **Lechner, Daniel, Granzer, Wolfgang e Kastner, Wolfgang.** *Security for KNXnet/IP.* 2008.

[27] **Kell, Alan e Colerbrook, Peter.** *Open Systems for Homes and Buildings: Comparing LonWorks and KNX.* s.l. : 2003.

[28] **Echelon.** Neuron Chips. *Echelon - Developers.* [Online] [Citação: 1 de Março de 2010.] http://www.echelon.com/developers/lonworks/neuron.htm.

[29] **OASIS Open.** *oBIX 1.0 - Committee Specification 01.* 2006.

[30] **Tan, Vu Van, Yoo, Dae-Seung e Yi, Myeong-Jae.** *Device Integration Approach to OPC UA-Based Process Automation Systems with FDT_DTM and EDDL.* 2009.

[31] **Mandrusiak, Manny.** OPC UA Education hits the road. *Automated Buildings.* [Online] Janeiro de 2010. [Citação: 1 de Junho de 2010.] http://www.automatedbuildings.com/news/jan10/columns/091229120909mandrusiak.htm.

[32] **OPC Foundation.** *OPC Unified Architecture - Specification - Part 1: Overview and Concepts.* 2009.

[33] **Mahnke, Wolfgang, Leitner, Stefan-Helmut e Damm, Matthias.** *OPC Unified Architecture.* s.l. : Springer, 2009.

[34] **Karg, Steve.** Elevator Simulator Design. [Online] [Citação: 2010 de 10 de 30.] http://kargs.net/elevator.html.

[35] **Newmarch, Jan.** *Foundations of Jini™ 2 Programming.* s.l. : Apress, 2006.

[36] **Apache River.** Apache River. [Online] [Citação: 1 de Setembro de 2010.] http://incubator.apache.org/river/RIVER/index.html.

[37] **Gelernter, Davic.** *Generativa Communication in Linda.* s.l. : Yale University, 1985.

[38] **Wells, G. C., Chalmers, A. G. e Clayton, P. G.** *Linda implementations in Java for concurrent systems.* s.l. : John Wiley & Sons, Ltd, 2003.

[39] **Unified Automation.** Unified Automation. [Online] [Citação: 1 de Setembro de 2010.] http://www.unified-automation.com.

[40] **Prosys.** Prosys OPC UA Java SDK. [Online] [Citação: 1 de Setembro de 2010.] https://www.prosysopc.com/opc-ua-sdk.php.

[41] **OMRON.** *FINS Commands.* 2001.

Anexo 1: Sistema de automação OMRON - Descrição dos equipamentos

Os equipamentos disponibilizado pela OMRON são dotados de um bus que permite agregar outros módulos ao módulo central de processamento. A solução disponibilizada pela OMRON e aplicada no demonstrador laboratorial tem um único processador que sobre o mesmo bus controla os demais equipamentos, todos equipamentos da OMRON envolvidos nesta solução encontram-se descritos na tabela seguinte:

PA 202	
	Fonte de alimentação
CJ1M CPU11 ETN	
	Processador com comunicação Ethernet integrada
ID 201	
	Entradas digitais
OD 201	
	Saídas digitais

MAD 42	
	Entradas e saídas analógicas
SCU41-V1	
	Módulo de comunicação RS232/485

Anexo 2: Sistema de automação Schneider - Descrição dos equipamentos

Os equipamentos disponibilizado pela Schneider comunicam sobre o protocolo LonWorks e ao contrário da solução da Schneider estes são descentralizados, o que significa que o controlo se divide pelos diversos equipamentos. Todos os equipamentos disponibilizados pela Schneider e envolvidos no demonstrador laboratorial encontram-se descritos na tabela seguinte:

Xenta 731	
	Controlador central, ponto de acesso da rede exterior
Xenta 121	
	Controlador de AVAC pré-programado
STR 106	
	Módulo de parede para controlo de AVAC (Sensor de temperatura, velocidade da ventilação, estado da presença (ocupado, desocupado ou em pausa) , etc.);
Xenta 422A	
	Módulo de extensão de entradas e saídas digitais

PM9C	
	Contador de energia em tempo real