FERRAMENTA DE APOIO AO PROJETO, CONFIGURAÇÃO E GESTÃO DE INSTALAÇÕES DOMÓTICAS

José Carlos Baptista Costa

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
2012

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação do 2º ano do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato José Carlos Baptista Costa, N.º 1740113, 1740113@isep.ipp.pt
Orientação científica: Maria Benedita Campos Neves Malheiro, mbm@isep.ipp.pt

Empresa: Live Simply, Lda.

Supervisão: Miguel Silva, ms.simply@gmail.com

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
4 de Novembro de 2012

Aos meus Pais.

"… o primeiro passo para a sabedoria é saber que nascemos mortalmente ignorantes …"

(Filósofo desconhecido)

Agradecimentos

Gostaria de agradecer, em primeiro lugar, à minha esposa Vina e à minha filha Carla pelo incentivo e aceitação incondicional da minha ausência durante muitos fins de semana, ao meu filho Miguel e à minha nora Marguerite pela ajuda nas traduções do resumo, à minha orientadora Professora Doutora Benedita Malheiro pela disponibilidade para ajudar e transmitir os seus conhecimentos, aos Engenheiros Miguel Silva e Gustavo Pinto pelas valiosas sugestões e aos sócios da Live Simply pela ajuda e participação construtiva.

Resumo

Esta dissertação apresenta o trabalho realizado no âmbito da unidade curricular de Tese / Dissertação (TEDI) do Mestrado em Engenharia Eletrotécnica e de Computadores – Especialização em Automação e Sistemas em parceria com a empresa Live Simply, uma empresa de domótica que decidiu apostar na inovação e no desenvolvimento de serviços e produtos de valor acrescentado para consolidar a sua posição no mercado.

Neste contexto, foram identificadas como mais-valias para a Live Simply a conceção, por um lado, de uma ferramenta de apoio técnico de integração e simplificação das fases de projeto, configuração e gestão de instalações domóticas e, por outro lado, de uma interface com a instalação para o cliente consultar e alterar, em tempo real, o estado dos atuadores.

Depois de analisadas as tecnologias disponíveis, selecionaram-se as soluções a adotar (linguagens de programação, servidores de base de dados e ambientes de desenvolvimento), definiu-se a arquitetura do sistema, detalhando-se os módulos de projeto, configuração e gestão de instalações, a estrutura da base de dados assim como o *hardware* de controlo da instalação. De seguida, procedeu-se ao desenvolvimento dos módulos de *software* e à configuração e programação do módulo de *hardware*. Por último, procedeu-se a um conjunto exaustivo de testes aos diferentes módulos que demonstraram o correto funcionamento da ferramenta e a adequação das tecnologias empregues.

A ferramenta de apoio técnico realizada integra as fases do projeto, configuração e gestão de instalações domóticas, permitindo melhorar o desempenho dos técnicos e a resposta aos clientes. A interface oferecida ao dono da instalação é uma interface *Web* de aspeto amigável e fácil utilização que permite consultar e modificar em tempo real o estado da instalação.

Palavras-Chave

Automação; Domótica; Ferramenta de Apoio; Projeto; Gestão; Configuração; Sistema Embarcado.

Abstract

This dissertation presents the thesis work carried out during the Master Degree in Electrical and Computer Engineering - Specialisation in Automation and Systems in partnership with Live Simply, a home automation company that focuses on developing innovative value-added products and services to consolidate its leading market position.

In this context, we identified two key improvements to the Live Simply business. On the one hand, the design and development of a technical support tool to simplify the integration of the project, configuration and management phases of home facilities and, on the other hand, the development of an interface for the customer to check or modify in real time the status of his home automation devices.

After screening and comparing the suitable technologies (from programming languages to database servers, development environments and hardware platforms), we selected the development set, defined the architecture of the system, detailing the project, configuration and management modules, the structure of the database as well as the facility control hardware. Then, we developed and configured all software and hardware modules. Finally, we conducted an exhaustive set of tests on the different modules to confirm that the tool worked correctly and that the selected technologies were appropriate.

The resulting technical support tool integrates the different project phases and enables the configuration and management of home automation installations, thereby improving the overall technical performance and customer responsiveness. The included customer interface is Web-based and user-friendly, allowing customers to easily check and modify the status of their home automation devices in real time.

Keywords

Home Automation / Domotics; Support Tool; Project; Configuration; Management; Embedded System.

Résumé

Cette thèse présente le travail réalisé dans le cadre du Mémoire de la Maîtrise en Génie Électrique et Informatique – Spécialisation en Systèmes d'Automatisation en partenariat avec Live Simply, une entreprise de domotique qui met l'accent sur le développement de produits et services à valeur ajoutée innovateurs, en vue de consolider sa position de leader du marché.

Dans ce contexte, nous avons identifié deux moyens d'améliorer l'offre de Live Simply. Tout d'abord, un outil d'aide technique qui simplifie l'intégration des phases du projet, configuration et gestion des installations domotiques et, d'autre part, une interface qui permet au client de vérifier ou de modifier en temps voulu l'état d'activité de leur dispositif domotique.

Après avoir passé en revue les technologies disponibles (langages de programmation, les serveurs de bases de données et des environnements de développement), nous avons sélectionné les solutions à utiliser, défini l'architecture du système et de tous ses modules, ainsi que la structure de la base de données et des outils de commande. Ensuite, nous avons développé des logiciels et configuré et programmé le module matériel de base. Enfin, nous avons mené une série exhaustive de tests sur les différents modules pour confirmer que l'outil fonctionnait correctement et que les technologies choisies étaient appropriées.

L'outil d'aide technique résultant intègre les différentes phases du projet, et permet une configuration et gestion faciles des installations domotiques, améliorant ainsi la performance technique et la réactivité du client. L'interface achevée est une interface *Web* avec un aspect convivial et facile d'utilisation qui vise à permettre aux clients de vérifier et modifier l'état de leurs dispositifs domotiques en temps réel.

Mots-clés

Domotique/Automation; Système d'Aide; Project; Configuration; Gestion; Système Embarqué.

Índice

Índice de Excertos de Código

Índice de Figuras

Índice de Tabelas

Acrónimos

6LoWPAN – *IPV6 over Low power Wireless Personal Area Networks*

AA – Acoplador de área

AC – *Alternating Current*

AJAX – *Asynchronous JavaScript and XML*

A-Mode – *Automatic Mode*

API – *Application Programming Interface*

ARM – *Advanced RISC Machine*

BCI – *Batibus Club International*

BL – *Bus Line*

CAN – *Controller Area Network*

CEBus – *Consumer Electronics Bus*

CLI – *Common Language Infrastructure*

CLR – *Common Language Runtime*

CSMA/CA – C*arrier Sense Multiple Access with Collision Avoidance*

CSS – *Cascade Style Sheets*

DA – *Date*

DHCP – *Dynamic Host Configuration Protocol*

DTR – *Data Terminal Ready*

EEPROM – *Electrically-Erasable Programmable Read-Only Memory*

EHS – *European Home Systems*

EHSA – *European Home Systems Association*

EIB – *European Installation Bus*

EIBA – *European Installation Bus Association*

E-Mode – *Easy Mode*

ETS – *Engineering Tool Software*

ETX – *End of TeXt*

EUA – Estados Unidos da América

FA – Fonte de Alimentação

FFD – Full Function Device

FIAS – *Fidelio Interface Application Specification*

HAL – *Hardware Access Layer*

HBS – *Home Bus System*

HTML – *HyperText Markup Language*

HTTP – *HyperText Transfer Protocol*

IDE – *Integrated Development Environment*

IEEE – *Institute of Electrical and Electronics Engineers*

IP – *Internet Protocol*

IR – *InfraRed*

JDBC – *Java DataBase Connection*

KNX – *KonNeX*

LA – *Link Alive*

LD – *Link Descriptor*

LS – *Link Start*

LED – *Light Emitting Diode*

LWT – *Listen While Talk*

MAC – *Media Access Control*

MISO – *Master Input/Slave Output*

MOSI – *Master Output/Slave Input*

MSIL – *Microsoft Intermediate Language*

OS – *Operating System*

OSI – *Open Systems Interconnection*

PME – Pequena e Média Empresa

RF – *Radio Frequency*

RISC – *Reduced Instruction Set Computer*

RTC – *Real Time Clock*

RTS – *Request To Send*

RX – *Received*

SD Card – *Secure Digital Card*

S-Mode – *System Mode*

Socket – Abstração computacional que define a interface de comunicação de uma

aplicação associando um endereço IP, um protocolo de transporte e um porto.

SPI – *Serial Peripheral Interface Bus*

SQL – *Structured Query Language*

STX – *Start of TeXt*

TCP – *Transmission Control Protocol*

TI – *TIme*

TRON – *Real-time Operating system Nucleus*

TX – *Transmitted*

USB – *Universal Serial Bus*

W3C – *World Wide Web Consortium*

WPAN – *Wireless Personal Area Network*

XHTML – *eXtensible HyperText Markup Language*

XML – *eXtensible Markup Language*

1. INTRODUÇÃO

Neste capítulo fornece-se o enquadramento do projeto realizado assim como a estrutura desta dissertação.

1.1. TEMA

O termo *domotics* resulta da composição da palavra latina *domus* com a palavra inglesa *informatics* (a ciência da informação) e terá sido introduzido pelo jornalista Bruno de Latour em 1984 [1]. Em inglês, os termos *domotics* e *home automation* são sinónimos. Em português, a palavra domótica resulta da composição de *domus* (casa) com robótica (controlo automático).

A domótica é uma tecnologia relativamente recente que permite aplicar as tecnologias de computação e automação à gestão dos recursos habitacionais. Tem como principal objetivo simplificar o dia-a-dia das pessoas, automatizando tarefas rotineiras, contribuindo para tornar a vida mais confortável e segura, intervindo em áreas como a automação, climatização, som, vídeo, segurança, entretenimento, comunicação, eficiência energética e gestão técnica de edifícios. É ainda de realçar que a domótica permite proporcionar às pessoas com necessidades especiais uma melhor qualidade de vida.

A domótica não se restringe a habitações ou edifícios particulares, mas aplica-se também ao sector industrial e dos serviços onde, cada vez mais, a gestão energética é um fator de elevada importância quer por razões ecológicas, quer por razões económicas.

O conceito de domótica não é novo, levando mesmo alguns autores a referir que terá começado no século XIX com a distribuição de água e energia porta a porta. No entanto, só na década de 70 com o advento do microprocessador é que se pode afirmar que esta tecnologia iniciou a evolução que a conduziu ao estado atual [3].

A constante miniaturização dos componentes eletrónicos, integração de funcionalidades e redução de custos tem permitido a crescente penetração da tecnologia no mercado. Atualmente, os sistemas de domótica, tirando partido do estado avançado das tecnologias de comunicação, permitem aceder remotamente a uma habitação ou a um edifício e interagir com os seus sistemas, desde o simples micro-ondas até ao vídeo porteiro, usando um equipamento tão simples como, por exemplo, um telemóvel.

1.2. PROBLEMA E MOTIVAÇÃO

A empresa Live Simply Lda., com sede em Gondomar, é uma Pequena e Média Empresa (PME) que se dedica ao projeto, comercialização e instalação de sistemas de domótica maioritariamente da marca Only.

Esta empresa, enquanto PME, não dispõe de recursos técnicos para estudar o funcionamento e a programação do vasto conjunto de módulos Only e, consequentemente, identificou como um fator de aumento de competitividade e de redução de custos a criação de uma aplicação de configuração de instalações domóticas que permitisse, de um modo simples e rápido, criar e armazenar um projeto de configuração de uma instalação e programar os respetivos módulos. Adicionalmente, definiu, como estratégia de crescimento, que a criação de interfaces com os sistemas de domótica instalados que permitissem ao utilizador final interagir com o sistema seriam fatores de valorização dos produtos e serviços comercializados.

A motivação para realizar este projeto adveio do desafio pessoal que proporcionou. O domínio de aplicação desta tese é: (*i*) interessante, complexo e pouco explorado no Mestrado de Engenharia Eletrotécnica e de Computadores; (*ii*) encontra-se em expansão, sendo possível contribuir para o desenvolvimento de soluções racionais e, ao mesmo

tempo, inovadoras; e (*iii*) é multidisciplinar, *i.e.*, envolve as áreas de automação, telecomunicações e computação, proporcionando a oportunidade de estudo, aprendizagem, aplicação e integração de novas tecnologias.

De realçar que, até ao momento da redação desta tese, não foi encontrado no mercado nenhum *software* de configuração para sistemas baseados neste fabricante.

1.3. OBJETIVOS

O conjunto de aplicações a desenvolver deverá servir dois objetivos distintos. Por um lado, pretende-se criar um conjunto de ferramentas que permita aos técnicos da Live Simply projetar, configurar e gerir uma instalação domótica e, por outro lado, pretende-se desenvolver uma ferramenta que, utilizando tecnologias *Web*, permita ao utilizador da instalação domótica interagir com a mesma.

Os requisitos necessários para a implementação da primeira solução não têm custos associados já que utilizam a infraestrutura da rede de domótica, sendo apenas necessário um computador com interface série.

Já para a segunda solução vai ser necessário projetar um sistema de dimensões reduzidas de modo a ser alojado no quadro elétrico da instalação. Assim, será necessário utilizar um sistema do tipo embarcado com duas interfaces: uma para o barramento de dados de domótica – que utiliza o protocolo de comunicação proprietário da Only – e outra para o barramento de comunicação com o *router* da instalação – que utiliza o *Transmission Control Protocol* (TCP).

O configurador deverá guardar a configuração da instalação num cartão de memória do tipo *Secure Digital* (SD) *card*. Este cartão destina-se a ser inserido no sistema embarcado, que passaremos a denominar por `SimplyWay`, tornando-o autónomo, ou seja, capaz de criar documentos *HyperText Markup Language* (HTML), enviá-los para a aplicação cliente, informar o estado da instalação e agir sobre esta de acordo com os pedidos da aplicação cliente.

Posteriormente, a empresa Live Simply solicitou que o sistema `SimplyWay` suportasse adicionalmente o protocolo Micros Fidelio para permitir a interface com clientes que

utilizem este protocolo. Esta solicitação foi implementada, tendo esta nova funcionalidade sido incorporada no sistema `SimplyWay`.

1.4. PLANEAMENTO DO PROJETO

Na Figura 1 mostra-se a calendarização do projeto, especificando-se as tarefas e a sua duração.

ID	Task Name	Start	Finish
1	**Projeto**	**29-03-2011**	**09-08-2012**
2	Estudar estado da arte	29-03-2011	31-05-2011
3	Analisar problema proposto e equacionar soluções	31-05-2011	12-07-2011
4	Estudar documentação	14-07-2011	01-08-2011
5	Realizar e testar protocolo Only	01-09-2011	02-11-2011
6	**Gestão do projeto**	**03-11-2011**	**16-12-2011**
7	Realizar aplicação de gestão	03-11-2011	30-11-2011
8	Criar base de dados e suas funcionalidades	01-12-2011	09-12-2011
9	Testar	12-12-2011	16-12-2011
10	**Configuração da instalação**	**19-12-2011**	**06-03-2012**
11	Realizar aplicação de configuração	19-12-2011	10-02-2012
12	Adicionar funcionalidades à base de dados	13-02-2012	24-02-2012
13	Testar	27-02-2012	06-03-2012
14	**Gestão da instalação**	**07-03-2012**	**31-05-2012**
15	Estudar funcionalidades do hardware de control	07-03-2012	20-03-2012
16	Desenvolver aplicação de controlo	21-03-2012	08-05-2012
17	Introduzir protocolo Micros Fidelio	09-05-2012	17-05-2012
18	Testar	18-05-2012	31-05-2012
19	**Conclusão**	**01-06-2012**	**09-08-2012**
20	Testar em condições reais	01-06-2012	28-06-2012
21	Elaborar relatório final	29-06-2012	09-08-2012

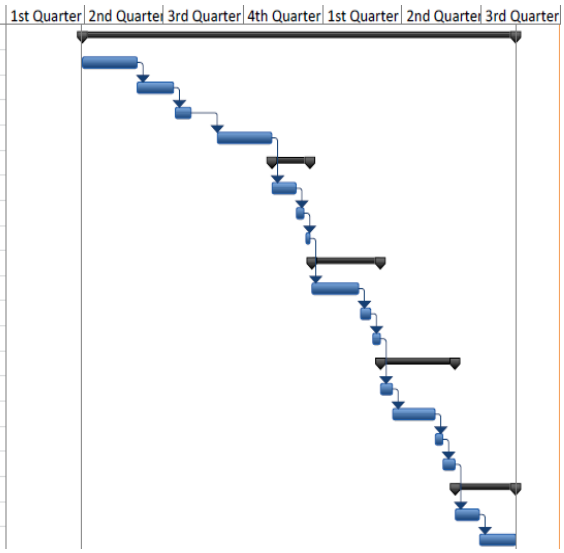

Figura 1 **Calendarização do projeto**

O projeto incluiu uma etapa inicial preparatória que envolveu o levantamento do estado da arte no domínio da domótica, a identificação do problema, a análise de requisitos e o estudo detalhado dos equipamentos utilizados pela empresa e respetivo protocolo de comunicação. As três etapas seguintes corresponderam ao desenvolvimento das três componentes do protótipo realizado: (*i*) criação e definição de projetos de instalações domóticas; (*ii*) configuração de instalações; e (*iii*) gestão de instalações. A etapa final focou-se na realização de testes exaustivos e na elaboração desta dissertação.

1.5. ORGANIZAÇÃO DA DISSERTAÇÃO

Este documento encontra-se organizado em cinco capítulos e um anexo. No Capítulo 1 apresenta-se o problema que se pretende solucionar com este trabalho assim como os objetivos propostos. O Capítulo 2 apresenta as soluções tecnológicas mais comuns em domótica, incluindo um estudo comparativo, e aborda os protocolos de comunicação utilizados em domótica, detalhando o protocolo Only (objeto deste trabalho). No Capítulo

3 descrevem-se as tecnologias adotadas para a realização do projeto, designadamente, linguagens de programação, servidores de base de dados, ambientes de desenvolvimento e os componentes de *hardware* utilizados no projeto. No Capítulo 4 apresenta-se a arquitetura do sistema, descrevendo-se os módulos de *software* desenvolvidos (Gestão do projeto, Configuração da instalação, Gestão da instalação), apresenta-se o armazenamento persistente de dados, detalhando-se a estrutura da base de dados e procedimentos desenvolvidos, a interligação entre os vários componentes de *hardware* e, por último, reportam-se os problemas e soluções encontrados assim como os testes realizados e os resultados obtidos. No Capítulo 5 efetua-se um balanço do trabalho realizado e sugerem-se melhoramentos. No Anexo A apresenta-se uma implementação com base no protocolo Micros Fidelio.

Neste documento aplicam-se as recomendações do Sistema Internacional de Unidades [2].

2. SISTEMAS DOMÓTICOS

Neste capítulo apresentam-se as principais tecnologias utilizadas em domótica, designadamente, as tecnologias abertas KNX e X10, a tecnologia proprietária Only utilizada pela Live Simply e, por último, referem-se tecnologias alternativas.

2.1. ENQUADRAMENTO

De um modo simplista, podemos considerar que um sistema de domótica é constituído por um conjunto de sensores, atuadores e interfaces que poderão ou não estar interligados em rede e onde a inteligência do sistema domótico reside em cada um dos módulos, num ou mais controladores ou ainda numa conjugação de ambos.

Em função da distribuição dos diferentes elementos de controlo define-se a forma de controlo como sendo centralizada ou descentralizada.

Nas soluções centralizadas a inteligência do sistema reside num controlador que em função da informação proveniente dos sensores envia informação para os atuadores de acordo com um algoritmo definido. Embora conduzam a arquiteturas mais baratas do ponto de vista dos módulos (módulos mais simples), trazem como inconvenientes uma grande quantidade de cablagem, interfaces homem-máquina mais complicados e uma total dependência do sistema de controlo.

Nas soluções descentralizadas, os vários elementos (módulos, interfaces e controladores) comunicam-se por um *bus*, sendo possível encontrar soluções em que não existe qualquer sistema central para o controlo. Neste caso, cada módulo pode ser visto como um sistema autónomo com capacidade de enviar e receber mensagens de e para outros módulos e reagir (no caso dos atuadores) em função das mensagens recebidas.

Encontram-se também dentro desta categoria, sistemas domóticos que, para além da inteligência de cada módulo, incluem sistemas de controlo / gestão que permitem executar funções mais complexas de acordo com o estado dos vários dispositivos como, por exemplo, a programação de tarefas em função da data e hora, a execução de cenários e a reação a situações de intrusão.

As soluções descentralizadas são as mais comuns e que trazem mais vantagens, embora tenham como inconveniente um maior custo por ponto controlado, bem como um acréscimo de custo pelas tecnologias empregues comparativamente às soluções centralizadas. No entanto, as vantagens das soluções descentralizadas como, por exemplo, uma maior robustez a falhas e maior facilidade no desenho de instalações e de utilização, sobrepõem-se às suas desvantagens, sendo prova disso a crescente adoção deste tipo de solução [4].

De notar, que em teoria será possível converter uma solução descentralizada numa solução centralizada, bastando para isso passar todo o controlo para o sistema de controlo centralizado. O modo como são transmitidos os dados assim como o seu significado definem o protocolo.

Atualmente, existem inúmeras soluções comerciais, desde sistemas oriundos dos Estados Unidos da América (EUA), como o X10 [5], o Consumer Electronics Bus (CEBus) [7], o SMARTHOUSE e o LonWorks [8], passando pelos desenvolvidos inicialmente na Europa como o BatiBus, o European Installation Bus (EIB) e o European Home Systems (EHS) [9] e finalizando com as tecnologias desenvolvidas no Japão como o Home Bus System (HBS) [10] [11] e o The Real-time Operating system Nucleus (TRON) [3] [12].

A Figura 2 apresenta a origem e as regiões de implantação dos principais protocolos de domótica.

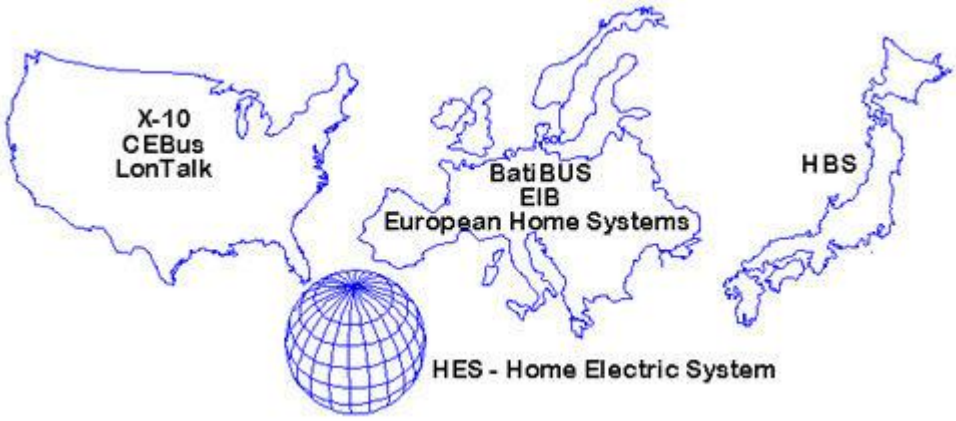

Figura 2 **Protocolos mais difundidos [23]**

Seguidamente iremos fazer uma abordagem aos dois protocolos abertos mais utilizados em domótica.

2.2. KNX

As especificações originárias do padrão Konnex (KNX) remontam aos anos 90. O KNX resulta da convergência de três protocolos existentes que eram promovidos pelas respetivas associações:

- European Home Systems Association (EHSA) com a tecnologia EHS;
- European Installation Bus Association (EIBA), com a tecnologia EIB;
- Batibus Club International (BCI), com a tecnologia Batibus.

Em 1999 estas três associações uniram-se para promover a Konnex Association, proprietária da tecnologia KNX. A EIB é presentemente um subgrupo da especificação KNX, sendo normal a designação EIB/KNX. O KNX está fortemente implantado na Europa. É um protocolo aberto e que permite uma arquitetura de controlo descentralizado.

2.2.1. MODOS DE CONFIGURAÇÃO

O KNX define três modos de configuração:

- *System Mode* (S-Mode) – Este modo de configuração, que se estabelece com o auxílio de um computador e da ferramenta Engineering Tool Software (ETS), é normalmente utilizado por integradores ou técnicos especializados;

– *Easy Mode* (E-Mode) – Este modo de configuração não requer computador e pode ser utilizado por pessoas com conhecimentos básicos de KNX, destinando-se à configuração de produtos com funções simples que se encontram pré-programados de fábrica;

– *Automatic Mode* (A-Mode) – Este modo de configuração é do tipo *Plug & Play*, não necessitando de qualquer configuração adicional.

Alguns produtos suportam mais do que um modo de configuração [13].

A Figura 3 ilustra a relação entre a funcionalidade e a complexidade do projeto.

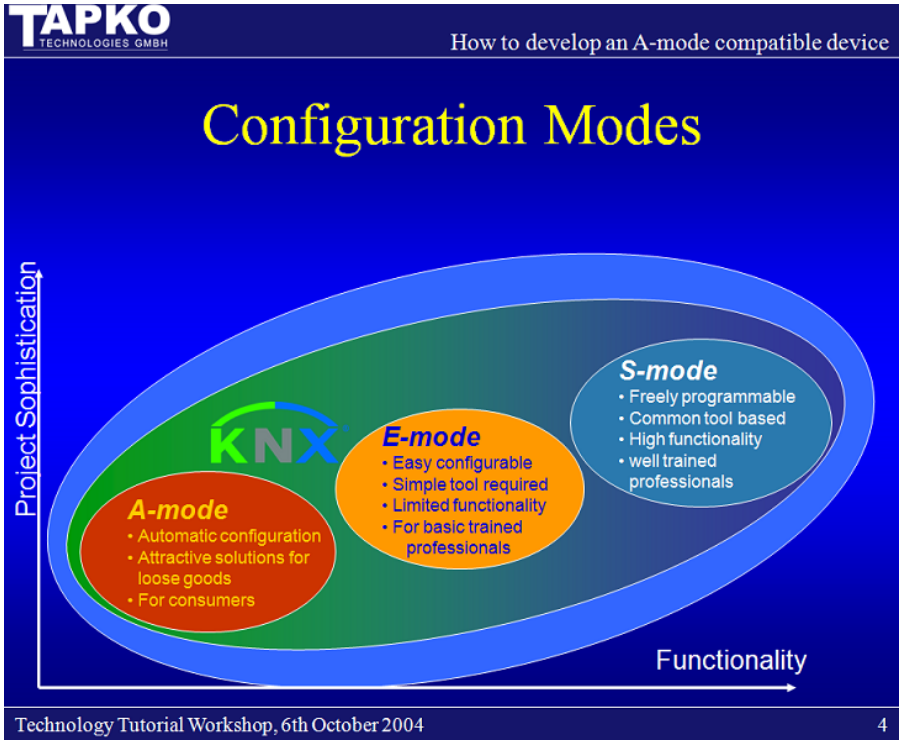

Figura 3 **KNX – Modos de configuração [14]**

2.2.2. ENGINEERING TOOL SOFTWARE

Trata-se de uma ferramenta de *software* de desenho e configuração de casas e edifícios inteligentes para plataformas Windows que armazena os dados em ficheiros do tipo eXtensible Markup Language (XML).

Existe em três versões (Demo, Lite e Professional) e permite ao utilizador preparar e configurar os projetos, usando especificações técnicas de produtos certificados. Com esta ferramenta é possível descrever a estrutura da casa, identificar os dispositivos a utilizar,

configurá-los, descrever o meio físico e proceder ao diagnóstico da instalação. Esta ferramenta proprietária é paga, sendo normalmente usada por integradores da tecnologia KNX. A versão atual é a 4 [13].

2.2.3. MEIO FÍSICO

O padrão KNX define vários meios físicos para a comunicação:

- Par entrançado (herdado do BatiBUS e do EIB);
- Rede elétrica (herdado do EIB e do EHS);
- Radiofrequência (RF) (também conhecido por KNX-RF);
- Infravermelhos (IR) (herdado do EIB.IR) limitação a 12 m;
- Ethernet (também conhecido por KNXnet/IP) [21].

Cada meio físico pode ser usado em combinação com um ou mais modos de configuração, cabendo a cada fabricante escolher a correta combinação de forma a atingir os objetivos de mercado e da aplicação. A informação é trocada através de datagramas ou pacotes de dados. Através do recurso a *Gateways* específicas é também possível a transmissão de datagramas KNX sobre outros meios como, por exemplo, a fibra ótica.

Dado que o par entrançado é o tipo de ligação mais utilizado, é de seguida detalhado.

2.2.3.1. PAR ENTRANÇADO

O KNX permite a utilização de um único par entrançado tanto para a comunicação como para a alimentação dum dispositivo, podendo, no entanto, ser utilizados pares distintos.

Os segmentos elétricos podem ter uma topologia arbitrária (linear, estrela, árvore ou combinação entre elas), desde que as limitações de impedância sejam respeitadas.

Numa linha – *Bus Line* (BL) na terminologia KNX – podem ser ligados até 64 dispositivos (*subscribers* ou participantes). Dois segmentos podem ser interligados através de um repetidor, passando a formar um novo segmento lógico designado linha. Teoricamente, uma linha pode incluir até quatro segmentos interligados por repetidores, aumentando, assim, a capacidade da linha até 256 dispositivos.

Várias linhas podem ser interligadas a uma linha principal através de acopladores de linha (AL) até a um máximo de 15 linhas, formando uma área.

Podem ser interligadas até 15 áreas, sendo a ligação da linha principal à linha de área efetuada através de um acoplador de área (AA). Cada linha tem de possuir a sua própria fonte de alimentação (FA) [16].

Na Figura 4 apresenta-se a topologia de uma rede KNX.

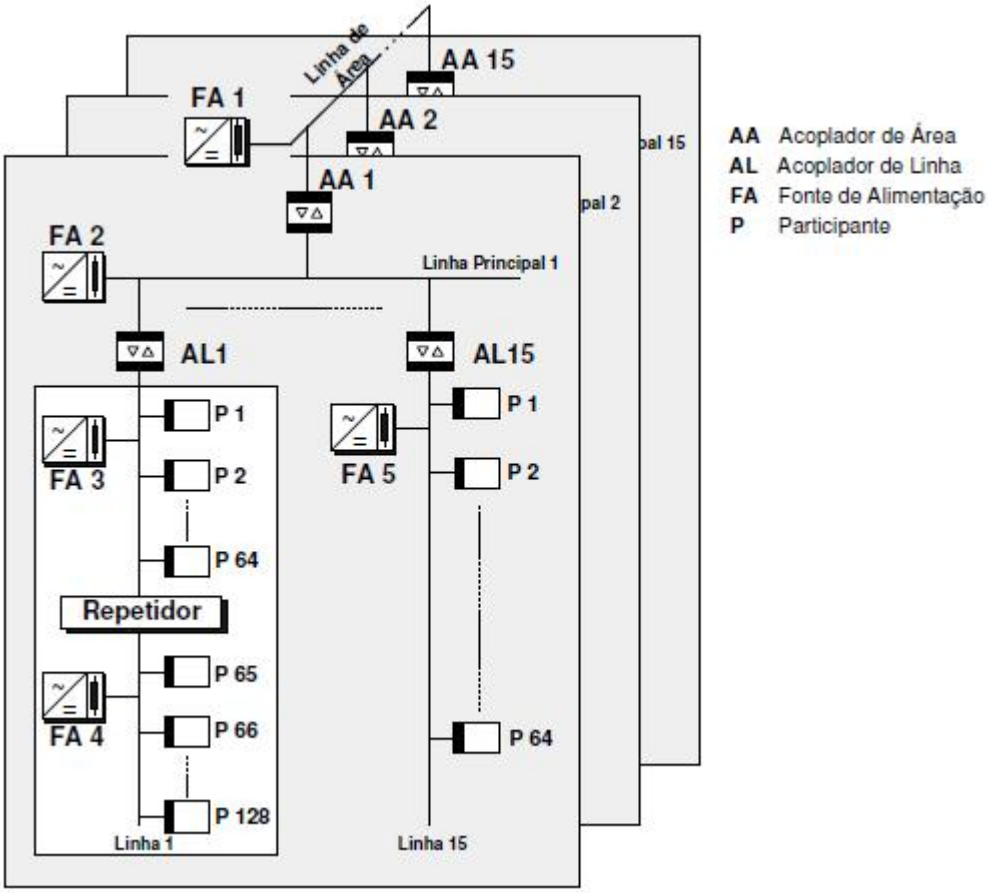

Figura 4 **KNX – Topologia de uma rede [16]**

2.2.4. ACESSO AO MEIO

O acesso ao meio baseia-se no protocolo C*arrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) que funciona do seguinte modo:

- O dispositivo que pretende transmitir identifica se o meio está livre para a transmissão. Caso não esteja, espera.

– Os dispositivos, enquanto transmitem, escutam o barramento. Por esta razão, o CSMA/CA também é referido como *Listen While Talk* (LWT). O objetivo deste comportamento é evitar colisões e garantir que as mensagens prioritárias tenham acesso ao meio, garantindo um processo de arbitragem. Assim, quando um dispositivo tenta impor o estado lógico '1' e deteta um estado lógico '0', pára de transmitir.

O 'Dispositivo 1' da Figura 5 perdeu o acesso ao meio porque a mensagem transmitida pelo 'Dispositivo 2' tem maior prioridade (bit '0' – dominante, no instante t5).

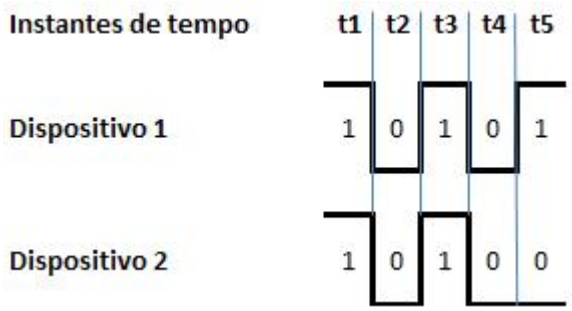

Figura 5 **KNX – Mecanismo de acesso ao meio**

A informação é trocada entre os diversos dispositivos ligados ao barramento sob a forma de datagramas e é transmitida simetricamente, ou seja, como uma diferença de potencial entre os dois fios e não entre qualquer fio e a terra. Deste modo, garante-se que as interferências que afetem ambos os condutores não influenciam a transmissão.

Os datagramas são enviados a um débito de 9600 b/s; cada caráter transmitido ocupa 1 B e é antecedido de um bit de início, seguido de um bit de paridade e finalizado por um bit de fim.

Um datagrama tem um comprimento que varia tipicamente entre 9 B e 23 B, demorando entre 20 ms a 40 ms a ser transmitido [16].

O formato dos datagramas KNX é representado na Figura 6.

Campo de controlo (8 b)	Endereço de origem (16 b)	Endereço de destino (16 + 1 b)	Contador (3 b)	Comprimento (4 b)	Dados (até 128 b)	Verifcação (8 b)

Figura 6 **KNX – Formato do datagrama**

O campo ‘Endereço de destino’ pode conter dois tipos distintos de endereço: o endereço físico de um único dispositivo ou de um grupo de dispositivos. Este campo tem 1 b adicional que permite distinguir entre endereços físicos e endereços de grupo.

O KNX obriga cada dispositivo a ter um endereço próprio unívoco (endereço físico) de modo a que possa ser inequivocamente identificado na rede. O formato do endereço de um dispositivo apresenta-se na Figura 7.

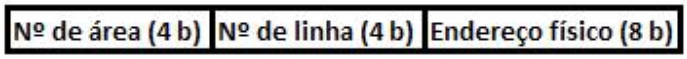

Figura 7 **KNX – Formato do endereço de um dispositivo**

2.3. X10

O X10 é um padrão industrial aberto, utilizado com mais expressão nos EUA, que foi desenvolvido entre 1976 e 1978 pela Pico Electronics. O seu nome deve-se ao facto de este ser o décimo projeto dos seus autores. A patente foi posteriormente adquirida pela X10 Ltd. e mantida até 1997. Define um protocolo aberto caracterizado por uma arquitetura de controlo descentralizado, permitindo que qualquer fabricante desenvolva a comunicação dos seus módulos em X10. Atualmente, o sistema X10 suporta radiofrequência.

2.3.1. MEIO FÍSICO

O X10 define os seguintes meios físicos para a comunicação: rede elétrica e radiofrequência.

2.3.2. ACESSO AO MEIO

No protocolo X10 a comunicação entre emissores e recetores faz-se enviando e recebendo sinais sobre a linha de corrente alternada – *Alternating Current* (AC). A informação digital é enviada sobre a forma de sequências curtas (*bursts*) de 120 kHz que são sobrepostas sobre o sinal AC nas passagens por zero. Cada sequência tem uma duração de 1 ms. Todos os módulos recebem os sinais em trânsito na rede.

O bit ’1’ é representado pelo *burst* e o bit ‘0’ é representado pela sua ausência. De notar, ainda, que cada bit é sempre representado pelo seu valor lógico seguido do seu complemento (ver Figura 8).

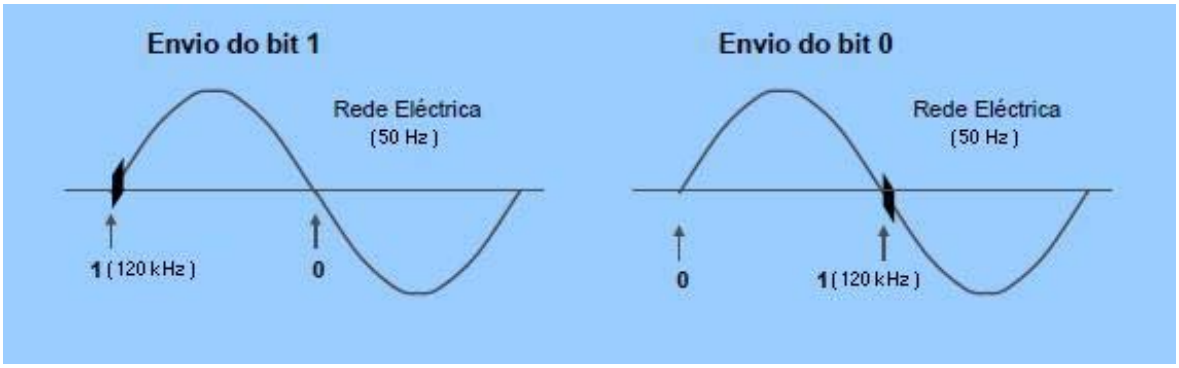

Figura 8 **X10 – Envio de sinais sobre a rede elétrica [22]**

As transmissões são sincronizadas pela passagem do sinal AC por zero e com um atraso máximo relativamente a esta passagem de cerca de 200 µs. Cada sequência de código é enviada duas vezes.

O X10, uma vez que usa as linhas de AC para comunicar, evidencia uma preocupação com a imunidade ao ruído, nomeadamente, adoptando o sincronismo na passagem por zero, o complemento do bit e o envio repetido da sequência.

O envio de um comando para um dispositivo pressupõe o envio de duas mensagens com a estrutura indicada na Figura 9. A primeira mensagem seleciona o dispositivo e a segunda mensagem contém o comando.

Cabeçalho	Código da habitação	Unidade / função

Figura 9 **X10 – Estrutura de uma mensagem**

A estrutura da mensagem do X10 é composta por três campos: ‘Cabeçalho’, ‘Código da habitação’ e ‘Unidade / função’.

O campo ‘Cabeçalho’ é sempre igual a ‘1110’ não sendo, neste caso, respeitada a regra do complemento.

O campo ‘Código da habitação’ permite especificar um valor entre zero e 16, sendo identificado por uma letra de ‘A’ a ‘P’.

O último campo ‘Unidade / função’ permite especificar o número do dispositivo ou a função. Para distinguir entre as duas situações é usado o último bit como se mostra na Tabela 1.

Tabela 1 X10 – Código de habitação e código de unidade / função [6]

Código de habitação				
	H1	H2	H4	H8
A	0	1	1	0
B	1	1	1	0
C	0	0	1	0
D	1	0	1	0
E	0	0	0	1
F	1	0	0	1
G	0	1	0	1
H	1	1	0	1
I	0	1	1	1
J	1	1	1	1
K	0	0	1	1
L	1	0	1	1
M	0	0	0	0
N	1	0	0	0
O	0	1	0	0
P	1	1	0	0

Código de unidade / função					
	D1	D2	D4	D8	D16
1	0	1	1	0	0
2	1	1	1	0	0
3	0	0	1	0	0
4	1	0	1	0	0
5	0	0	0	1	0
6	1	0	0	1	0
7	0	1	0	1	0
8	1	1	0	1	0
9	0	1	1	1	0
10	1	1	1	1	0
11	0	0	1	1	0
12	1	0	1	1	0
13	0	0	0	0	0
14	1	0	0	0	0
15	0	1	0	0	0
16	1	1	0	0	0
All Units Off	0	0	0	0	1
All Units On	0	0	0	1	1
On	0	0	1	0	1
Off	0	0	1	1	1
Dim	0	1	0	0	1
Bright	0	1	0	1	1
All Lights Off	0	1	1	0	1
Extended Code	0	1	1	1	1
Hail Request	1	0	0	0	1
Hail Acknowledge	1	0	0	1	1
Pre-Set Dim	1	0	1	X	1
Extended Data (analog)	1	1	0	0	1
Status=on	1	1	0	1	1
Status=off	1	1	1	0	1
Status Request	1	1	1	1	1

2.4. ONLY

Devido ao acordo de confidencialidade com o fabricante e com a Live Simply, efetua-se apenas uma descrição sumária do protocolo da Only e da sua implementação no meio físico.

Trata-se de um protocolo proprietário criado em 2005 pela empresa Enancer Electrónica S. A. que permite a coexistência, numa única rede, de módulos de automação, segurança, áudio e climatização organizados segundo uma arquitetura distribuída. O protocolo é implementado pelos diferentes módulos da marca Only fabricados pela empresa. A codificação dos bits tem como base um protocolo utilizado para a comunicação por

infravermelhos em equipamentos de alta-fidelidade que foi alterado de modo a minimizar a utilização de recursos dos microcontroladores, permitindo assim a utilização de processadores de baixo custo, mas suficientemente eficazes para o fim em vista.

2.4.1. MEIO FÍSICO

O meio físico especificado é constituído pelo neutro da instalação elétrica e por um condutor adicional. Estes dois condutores servem, não só, para a alimentação do *bus* (24 V), mas, também, para suporte à comunicação entre os diferentes módulos.

2.4.2. ACESSO AO MEIO

O acesso ao meio baseia-se no protocolo CSMA/CA já abordado na subsecção 2.2.4.

Os telegramas, cuja estrutura está exemplificada na Figura 10, são enviados a um débito de 38 400 b/s e são de comprimento variável.

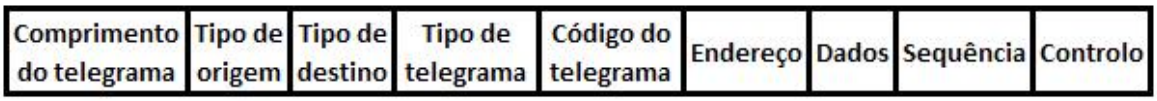

Figura 10 **Only – Estrutura de um telegrama**

2.4.3. PROGRAMAÇÃO

Todos os módulos podem ser programados no local não sendo necessário grandes conhecimentos de domótica; cada módulo dispõe de microinterruptores utilizados para o efeito; a programação por computador é também possível mediante aplicação informática adequada como se irá demonstrar ao longo deste trabalho.

2.5. ANÁLISE COMPARATIVA

A tecnologia KNX é bastante mais complexa e poderosa do que a tecnologia X10, não só no que respeita à capacidade de endereçamento, mas, também, ao conjunto de funcionalidades que disponibiliza. Face ao KNX, o X10 apresenta um número reduzido de comandos para controlar os atuadores.

No entanto, no que diz respeito à facilidade de instalação e custos associados, incluindo o custo dos módulos, o X10 ultrapassa o KNX. Ao invés do KNX, a instalação e manutenção

de uma instalação domótica baseada em X10 não requer técnicos especializados dada a sua simplicidade.

Em relação a ferramentas disponíveis, é possível encontrar na Internet uma grande variedade de aplicações para X10, incluindo algumas gratuitas. No caso do KNX, o técnico necessita de um *software* de configuração com um custo elevado.

Relativamente à tecnologia Only, verificamos que a funcionalidade e a versatilidade se encontram ao nível do KNX, embora com um custo menos elevado. Relativamente à instalação elétrica, necessita de um condutor adicional. No entanto, torna-se menos onerosa que o KNX em virtude de não haver necessidade de alojar módulos no quadro elétrico, com a exceção da fonte de alimentação. Para instalações elétricas de raiz, o custo do condutor adicional torna-se desprezível em comparação com o custo total da instalação.
A Tabela 2 sintetiza algumas das características apontadas [15].

Tabela 2 KNX, X10, ONLY – Análise comparativa

Característica	KNX	X10	ONLY
Duração de uma trama (ms)	[20, 40]	940	[3, 8]
Número máximo de dispositivos	≈60 000	256	4 096[1]
Custo dos dispositivos	Elevado	Baixo	Médio
Dificuldade de instalação	Alta	Baixa	Baixa
Funcionalidade	Alta	Baixa	Alta

2.6. TECNOLOGIAS ALTERNATIVAS

Várias soluções para domótica têm sido implementadas baseadas em redes de campo do tipo RS485 e Controller Area Network (CAN) *bus*. Foram desenvolvidas por empresas que mantiveram o seu protocolo fora do domínio público e, por esta razão, nunca se tornaram tão interessantes do ponto de vista comercial, quanto o KNX ou o X10. São, muitas vezes, simples de configurar, fáceis de instalar e até de baixo custo. O seu principal problema reside no facto de apenas o detentor do protocolo poder desenvolver módulos para a sua rede, impedindo outros fabricantes de o fazerem, reduzindo assim a sua difusão no mercado.

1 São possíveis 4096 módulos de cada tipo; até à data da redação desta tese existiam 32 módulos diferentes.

Paralelamente, existe ainda o grupo das soluções para domótica baseadas na tecnologia de redes sem fios. Apesar dos protocolos mais comuns das redes de área pessoal sem fios – Wireless Personal Area Network (WPAN) – não responderem convenientemente às necessidades de sensores e dispositivos de controlo de domótica, como é o caso do Bluetooth[2] ou do Wi-Fi[3], o mesmo não acontece com a especificação ZigBee[4]. Para além das soluções Wi-Fi, Bluetooth e ZigBee, existem outras soluções que utilizam a norma IEEE 802.15.4 que começam a ser comercializadas no domínio da domótica. É o caso da NXP que apresenta uma solução baseada em 6LoWPAN e IEEE 802.15.4 – NXP and Belkin Bring 6LoWPAN Connectivity to the Home[5].

O ZigBee, ao permitir a adoção da tecnologia da comunicação sem fios na domótica, apresenta como grande vantagem a possibilidade de realização de instalações domóticas sem cablagem e de baixo consumo energético, permitindo uma grande redução do custo da instalação. Em Portugal existem algumas aplicações de ZigBee para controlo de iluminação em naves industriais[6].

2.6.1. ZIGBEE

A especificação ZigBee surgiu em 2002 e define as camadas de rede e aplicação do modelo OSI[7]. As camadas físicas e de acesso ao meio são definidas pela norma IEEE 802.15.4.

O ZigBee destina-se à criação de redes de área pessoal sem fios (WPAN) de baixo consumo e baixo custo, tornando-se ideal para aplicações de domótica e para dispositivos alimentados a bateria [18].

Opera em três frequências distintas livres de licenciamento:

---

2 Protocolo de nível físico definido na norma IEEE 802.15.1 utilizado em redes de baixo alcance, permitindo trocar informação entre dispositivos como telefones móveis, computadores, impressoras, teclados, ratos e outros dispositivos.

3 IEEE 802.11 define um conjunto de técnicas de modulação que usam o mesmo protocolo base; utilizado para comunicações Internet, redes e vídeo.

4 Camada física do modelo OSI definida na norma IEEE 802.15.4.

5 Disponível em http://www.jennic.com/files/press_releases/120110_Belkin_NXP_WeMo_VIDEO_NEWS_RELEASE_FINAL_updated_logo.pdf. [Acedido em Maio de 2012].

6 Informação disponível em http://www.controlar.pt. [Acedido em Maio de 2012].

7 *Open Systems Interconnection* (OSI) é um modelo que caracteriza as funções de um sistema de comunicação fazendo uma divisão em sete camadas de abstração.

– 2,4 GHz (taxas de transmissão de 250 Kib/s a nível mundial);

– 915 MHz (taxas de transmissão de 40 Kib/s nos EUA e Austrália);

– 868 MHz (taxas de transmissão de 20 Kib/s na Europa).

O ZigBee prevê três tipos de dispositivos lógicos ou nós: *coordinator*[8], *router*[9] e *endpoint*[10]. O *coordinator* é o nó responsável pela criação e manutenção da rede. Pode também funcionar como *bridge* para outras redes ZigBee. O *router*, que pode funcionar como um nó normal, é um nó que opera habitualmente como *router* intermédio, permitindo que os restantes nós comuniquem sem a intervenção do coordenador. O *endpoint* é um nó que apenas comunica com a rede e não tem funções de gestão. Com o objetivo de poupar energia, estes nós que podem ser alimentados por baterias, ficam em *stand by* quando não estão a receber ou a enviar mensagens.

O ZigBee tem um alcance máximo de 150 m, permitindo um máximo de 65 535 nós por coordenador. As redes ZigBee podem apresentar três diferentes topologias distintas (Figura 11): estrela, árvore ou em malha.

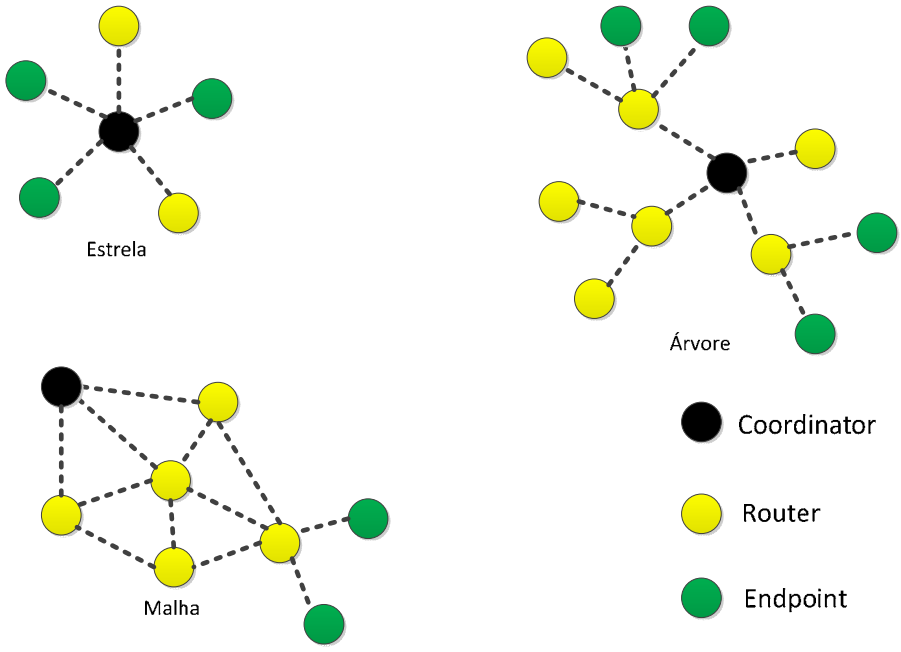

Figura 11 **ZigBee – Topologias de rede**

8 Implementados com base em dispositivos físicos da classe *Full Function Device* (FFD).
9 Implementados com base em dispositivos da classe FFD.
10 Implementados com base em dispositivos da classe FFD ou *Reduced Function Device* (RFD).

A topologia em estrela é a mais simples e limitada das três topologias. Consiste num *coordinator* e num conjunto de *endpoints* ou *routers*. Uma desvantagem desta topologia é a inexistência de um encaminhamento alternativo entre o *coordinator* e um *endpoint* no caso de uma falha de sinal de radiofrequência (RF) entre estes dois elementos.

A topologia em árvore é constituída por um *coordinator* ligado a um conjunto de *routers* e *endpoints*. Nesta topologia, cada *router* pode, por sua vez, estar ligado a um conjunto de *routers* e *endpoints*. Esta replicação pode, à semelhança dos ramos e folhas de uma árvore real, ocorrer em cada ramo da árvore e em vários níveis.

Nas topologias em estrela e em árvore, um nó *router* pode funcionar como *endpoint*. Nesse caso, a sua função retransmissão de mensagens (*relay*) não é utilizada.

A topologia em malha é semelhante à topologia em árvore, mas com o *coordinator* no topo da estrutura. As regras de comunicação nesta topologia são mais flexíveis do que na topologia em árvore uma vez que os *routers* podem comunicar entre si. A topologia em malha permite uma melhor propagação das mensagens porque apresenta caminhos alternativos no caso de uma ligação falhar.

A propagação de mensagens está a cargo da pilha protocolar e portanto torna-se transparente à camada de aplicação.

O ZigBee apresenta dois tipos de endereços: o endereço IEEE e o endereço de rede. O endereço IEEE, designado *Media Access Control* (MAC) *address*, é constituído por 64 b e assegura a cada módulo um identificador único. O endereço de rede é constituído por 16 b e identifica o nó na rede, ou seja, é local. Isto significa que dois nós em duas redes diferentes podem ter o mesmo endereço de rede, mas terão seguramente diferentes endereços IEEE – MAC *addresses*. Os endereços de rede são mapeados pelo nó pai que poderá ser um *router* ou um *coordinator*. O *coordinator* terá sempre o endereço de rede 0x0000 [19].

Face ao Bluetooth, o ZigBee tem uma taxa de transmissão mais baixa e os seus nós estão a maior parte do tempo dormentes. Isto significa que o ZigBee pode operar dispositivos alimentados por baterias durante períodos de vários meses. Adicionalmente, enquanto os nós ZigBee ‘acordam’ e ficam operacionais em apenas 15 ms, os nós Bluetooth podem demorar cerca de 3 s a responder. Em termos de consumo, um dispositivo ZigBee

dormente consome cerca de 3 µA e um dispositivo Bluetooth consome cerca de 0,2 mA [20]. O tamanho da pilha protocolar é de cerca de 32 KiB no ZigBee e 250 KiB no Bluetooth. Por último, enquanto que com ZigBee é possível atingir 65 535 células, com Bluetooth o número máximo é de 8 [50].

Pelas razões enunciadas, o ZigBee é o principal protocolo de comunicação sem fios adotado pelos fabricantes e integradores de sistemas na área da domótica.

2.6.2. NOVAS TENDÊNCIAS

Ultimamente, tem-se assistido à utilização de câmaras para deteção de movimento e a sistemas de reconhecimento de fala para controlo de instalações domóticas. Embora sejam tecnologias ainda pouco empregues nesta área, nota-se que há, por parte de alguns fabricantes, o interesse de incorporar estes novos meios no leque das soluções disponibilizadas, nomeadamente nas situações em que existe alguma incapacidade por parte dos utilizadores.

2.7. CONCLUSÃO

Neste capítulo apresentou-se o estado da arte no domínio dos sistemas domóticos, incluindo a tecnologia utilizada neste trabalho. Como a Live Simply utiliza os equipamentos da Only, as ferramentas a desenvolver têm que implementar as especificações deste protocolo.

No capítulo seguinte apresentam-se as tecnologias adotadas na construção da ferramenta de projeto, configuração e gestão de instalações domóticas.

3. AMBIENTE DE DESENVOLVIMENTO

Neste capítulo descrevem-se as linguagens, tecnologias, servidor de bases de dados e ambientes de desenvolvimento utilizados no desenvolvimento da ferramenta.

3.1. LINGUAGENS, TECNOLOGIAS E BIBLIOTECAS

3.1.1. .NET FRAMEWORK E C#

O .NET Framework é uma plataforma de desenvolvimento de aplicações que permite ao utilizador criar aplicações usando diferentes linguagens de programação sem ter que se preocupar com detalhes como, por exemplo, a gestão de memória.

A componente central do .NET Framework é a Common Language Runtime (CLR) que é uma linguagem de execução baseada num padrão desenvolvido pela Microsoft chamado Common Language Infrastructure (CLI). Como se trata de um padrão aberto, existem implementações alternativas como, por exemplo, os projetos Rotor (projeto Microsoft para Mac OS) e Mono, liderado inicialmente pela Novell e, posteriormente, pela Xamarin, disponível para diferentes plataformas [24].

Uma das principais vantagens do .NET Framework é a independência da linguagem de programação. Várias linguagens podem ser utilizadas, nomeadamente C# e VB.NET da Microsoft ou Python, Perl, Java de fora da esfera da Microsoft [25] [44].

A Microsoft disponibiliza uma versão mais leve do .NET Framework denominada .NET Micro Framework. O .NET Micro Framework permite o desenvolvimento de aplicações para dispositivos mais pequenos e com menos recursos como, por exemplo, os sistemas embarcados, utilizando as interfaces gráficas disponíveis para a versão mais completa [28]. Neste contexto, o C# é a linguagem de programação mais frequente, sendo possível fazer a depuração das aplicações tanto num emulador como no dispositivo final.

Inclui um conjunto de ferramentas designado .NET Micro Framework 4.1 Porting Kit que permite a adaptação do Framework ao *hardware* do fabricante do dispositivo [29] e suporta um alargado número de sistemas de desenvolvimento nomeadamente dispositivos baseados em processadores Advanced RISC Machines (ARM) [30].

C Sharp (C#) é uma linguagem desenvolvida pela Microsoft para a plataforma .NET. É uma linguagem orientada a objetos, inspirada nas linguagens C, C++ e Java que, quando compilada, gera uma linguagem intermédia designada por Microsoft Intermediate Language (MSIL) [26] [27].

3.1.2. BIBLIOTECA GHI

A GHI Library é uma biblioteca desenvolvida pela GHI Electronics, LCC para produtos embarcados compatíveis com o .NET Micro Framework. Esta biblioteca, que está disponível para descarga na Internet, permite aceder programaticamente às diversas características dos diferentes produtos disponibilizados por este fabricante[11]. Desta forma, qualquer um destes produtos pode ser programado no ambiente Microsoft Visual Studio, utilizando a linguagem de programação C#.

3.1.3. EXTENSIBLE MARKUP LANGUAGE E XML SCHEMA DEFINITION

A eXtensible Markup Language (XML) é uma linguagem de anotação destinada ao armazenamento e transporte de dados que implementa um mecanismo de identificação da

11 Informação relevante quanto à gama de produtos encontra-se disponível em http://www.ghielectronics.com. [Acedido em Janeiro de 2012]. O *software* encontra-se disponível em http://www.tinyclr.com. [Acedido em Janeiro de 2012].

estrutura de dados [31] [32]. A definição formal do vocabulário (anotações) de um documento XML encontra-se num ficheiro XML Schema Definition (XSD) dedicado [33] [34]. O esquema é uma representação abstrata das características dos objetos e da sua relação. Num ficheiro XSD é possível identificar a estrutura organizacional dos dados e os tipos de dados utilizados.

Existem ferramentas de edição de documentos XML que podem ser usadas de um modo gratuito[12].

3.1.4. EXTENSIBLE HYPERTEXT MARKUP LANGUAGE

Segundo o World Wide Web Consortium (W3C)[13], a eXtensible HyperText Markup Language (XHTML) consiste numa reformulação da HyperText Markup Language (HTML)[14] versão 4.0 baseada em XML. Enquanto o HTML é a linguagem utilizada para formatação e apresentação de dados (páginas *Web*), o XML é uma linguagem de representação de dados. Assim, o XHTML é uma linguagem de apresentação de dados com uma sintaxe e estrutura rigorosa, promovendo a criação de documentos válidos e bem formados [35] [36].

3.1.5. CASCADE STYLE SHEETS

As folhas de estilo em cascata – Cascade Style Sheets (CSS) – permitem definir e aplicar diferentes estilos a documentos *Web*. Os estilos especificam os tipos de letra, cores, espaçamentos, *etc*.

Uma das grandes vantagens da utilização de CSS é a separação do conteúdo da página *Web* do seu estilo de apresentação. Embora seja possível definir os estilos dentro de um documento *Web*, é aconselhável criar os estilos num documento separado da página *Web* de forma a poder facilmente aplicar, mudar e reutilizar estilos. Neste caso, a aplicação de uma folha de estilos consiste na especificação da referência correta na página *Web*. O ficheiro de estilos é um documento que pode ser desenvolvido com um normal editor de texto [37] [38].

---

12 A aplicação XML Notepad 2007 pode se descarregada e utilizada gratuitamente. Informação disponível em http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=7973. [Acedido em Maio de 2012].

13 Disponível em http://www.w3.org. [Acedido em Maio de 2012].

14 HTML é a linguagem de anotação de hipertexto adotada como padrão de facto para documentos *Web*. Informação disponível em http://www.w3.org/community/webed/wiki/HTML/Specifications. [Acedido em Maio de 2012].

3.1.6. JAVASCRIPT

A linguagem baseada em guiões JavaScript, inicialmente denominada LiveScript, resultou do esforço conjunto da Netscape Communications Corporation e da Sun Microsystems, Inc. Destina-se a desenvolver páginas *Web* interativas e dinâmicas, o que é impossível recorrendo exclusivamente a HTML ou XHTML. Trata-se de uma linguagem baseada em objetos que executa em modo interpretado do lado do navegador (cliente).

À semelhança das folhas de estilo CSS, é também possível implementar o código JavaScript na página *Web*. No entanto, é aconselhável fazê-lo num documento separado, permitindo assim uma melhor manutenção e reutilização do código. Nestes casos continua a ser necessário referir na página *Web* o ficheiro que contém o código JavaScript [39] [40].

3.1.7. ASYNCHRONOUS JAVASCRIPT AND XML

O Asynchronous JavaScript and XML (AJAX) é um conjunto de tecnologias que permite construir aplicações interativas para a *Web*. O AJAX combina JavaScript, HTML ou XHTML, XML, CSS e o Document Object Model (DOM) e permite a troca assíncrona de dados utilizando o objeto XMLHttpRequest [41].

O AJAX possibilita a atualização parcial de páginas *Web* do lado do cliente sem necessidade de recarregar a página na totalidade. O navegador interage com o servidor submetendo e efetuando pedidos assíncronos de dados em formato XML de forma completamente transparente para o utilizador. Os dados devolvidos pelo servidor (em XML) são processados pelo JavaScript (no navegador) de modo a atualizar as partes relevantes da página *Web*. As páginas que apresentam mapas do Google Maps constituem um bom exemplo de utilização desta tecnologia – o mapa é atualizado em função da interação do utilizador sem que a página seja recarregada na sua totalidade.

3.2. SERVIDORES DE BASES DE DADOS

3.2.1. MICROSOFT SQL SERVER EXPRESS

Trata-se de uma versão gratuita e mais leve da versão do servidor de bases de dados Microsoft SQL Server. Implementa a linguagem Structured Query Language (SQL) e, embora esteja limitado a bases de dados até 10 GiB, é uma boa solução para pequenos sistemas desenvolvidos em linguagens da Microsoft como, por exemplo, o Visual

Basic.NET e o C#. Existem interfaces para outras linguagens como, por exemplo, Java utilizando a Application Programming Interface (API) Java DataBase Connection (JDBC). Neste trabalho foi utilizada a versão 2008.

3.3. AMBIENTES DE DESENVOLVIMENTO INTEGRADO

3.3.1. MICROSOFT VISUAL STUDIO

O Microsoft Visual Studio, presentemente na versão 2012, é um poderoso ambiente de desenvolvimento integrado – Integrated Development Environment (IDE) – dedicado ao .NET Framework que inclui várias ferramentas de desenvolvimento da Microsoft e suporta várias linguagens de desenvolvimento como o Visual Basic .NET, C# .NET, C++ .NET e ASP.NET. Existem várias versões disponíveis desde a versão 2012 Express (gratuita) até à versão 2012 Ultimate. Neste trabalho foi utilizada a versão profissional 2010.

3.3.2. MICROSOFT SQL SERVER MANAGEMENT STUDIO

Presentemente na versão 2012 e contando também com uma versão gratuita (versão Express), o Microsoft SQL Server Management Studio é um ambiente de desenvolvimento que disponibiliza um conjunto de ferramentas gráficas e editores de *script* que permitem configurar, gerir, administrar e desenvolver todos os componentes do Microsft SQL Server [42]. A versão 2008 deste IDE foi usada para a criação, configuração e interação com a base de dados do projeto.

3.4. PLATAFORMAS DE *HARDWARE*

O *hardware* utilizado foi escolhido tendo em conta a facilidade de programação, o custo e o desempenho. A escolha da solução adotada de entre o conjunto de soluções possíveis baseou-se em razões comerciais que se prendem com a facilidade de aquisição e custo dos equipamentos.

3.4.1. FEZ DOMINO

A placa da GHI Fez Domino foi a solução de *hardware* escolhida para este projeto. Trata-se de um pequeno módulo de desenvolvimento que contém instalado o .Net Micro Framework da Microsoft. É facilmente expansível recorrendo a módulos adicionais que se podem comprar ou mesmo realizar no caso de aplicações mais específicas. Pode ser

alimentada por uma fonte de 7 V a 12 V com capacidade para 200 mA ou, ainda, a 5 V utilizando o cabo de programação *Universal Serial Bus* (USB).

É baseada no *chipset* USBizi que é um circuito integrado de 144 pinos fabricado também pela GHI. Este circuito tem por base o microcontrolador ARM7 da NXP LPC2388. O *firmware* inclui o .Net Micro Framework bem como diversos *Hardware Access Layer* (HAL). Na Figura 12 mostra-se um diagrama de blocos da arquitetura do USBizi.

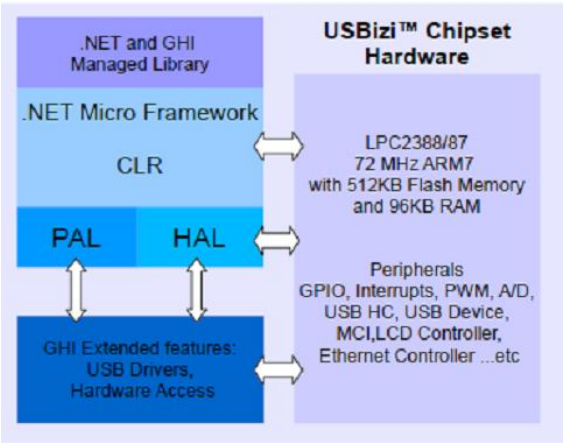

Figura 12 **Arquitetura USBizi [43]**

Como complemento a esta informação recomenda-se a consulta do manual do fabricante [43]. Na Tabela 3 mostram-se algumas das características mais relevantes do USBizi.

Tabela 3 Características do USBizi

Microsoft .NET Micro Framework V4	4 UART
72 MHz 32 b Processor	2 x CAN Channels
96 KiB Ram	8 x 10 b Analog Input
0,5 MiB Flash	10 b Analog Output
Embedded USB Host	4 b SD/MMC interface
Embedded USB Client	6 PWM
71 GPIO	100 mA everthing enabled
35 Interrupt lines	200 µA Hibernate Modes
2 x SPI (8 b/16 b)	-40°C to +85°C
I2C	RoHS Lead Free

De realçar que, embora o *hardware* disponibilize 96 KiB de memória volátil e 512 KiB de memória *Flash*, apenas estão disponíveis para a aplicação do utilizador respetivamente

64 KiB e 148 KiB. Na Figura 13 apresenta-se a imagem da placa com a indicação dos conetores de expansão assim como do *hardware* mais relevante.

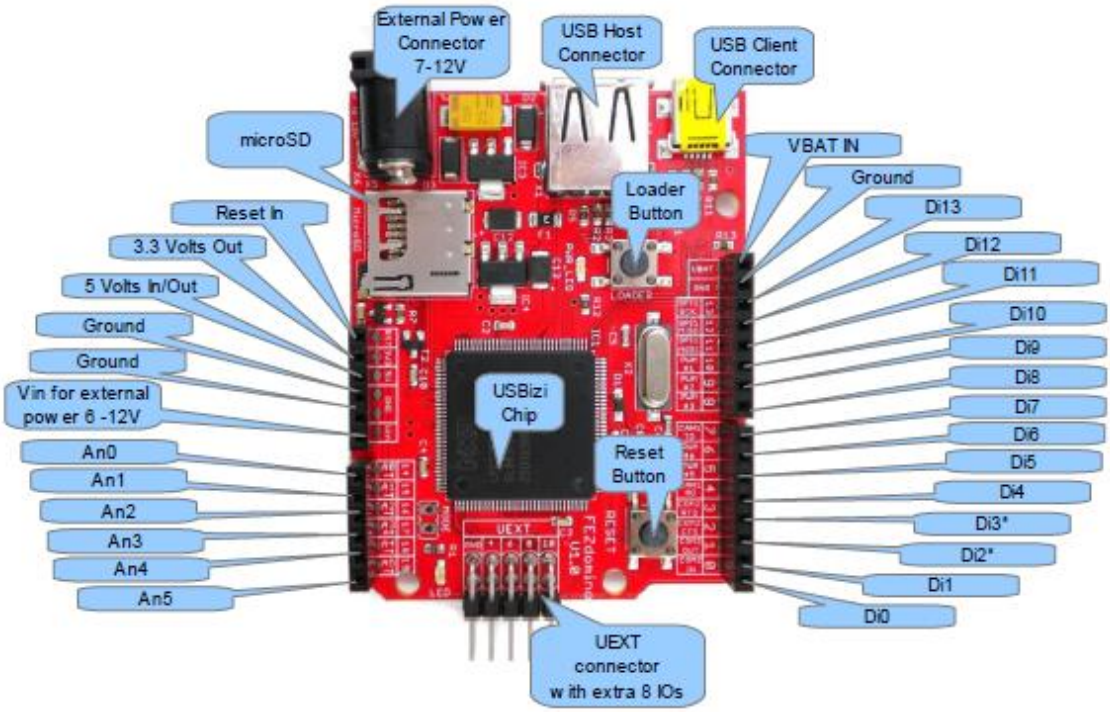

Figura 13 **Placa Fez Domino [45]**

Embora alguns dos pinos estejam reservados para funções específicas, muitos outros são multiplexados de modo a servirem diferentes funcionalidades. Por exemplo, os pinos Di0 e Di1 podem também servir para receção e transmissão de dados através da porta série COM1. Cabe ao utilizador escolher convenientemente as funcionalidades destes pinos de uso geral, tendo em atenção as suas características elétricas e funcionais.

Neste trabalho são necessários os seguintes periféricos:

- Porta USB para programação e depuração (USB *Client Connector*);
- Suporte para cartões de memória do tipo microSD que irá conter os ficheiros das configurações do `SimplyWay` e da instalação domótica.
- Porta série assíncrona para implementação da comunicação RS232 necessária para interligação com o barramento de domótica;

– Porta série síncrona para implementação das comunicações baseadas no protocolo *Transmission Control Protocol* (TCP), necessária para a comunicação com periféricos utilizando o protocolo Micros Fidelio ou HTTP;

A porta USB e o suporte para cartões microSD encontram-se disponíveis no *hardware* da placa não sendo necessária qualquer ação. No entanto, constata-se que no caso da porta assíncrona, os pinos Di0 e Di1, que irão ser programados para desempenharem as funções de transmissão e receção, apenas operam a níveis TTL (0 V a 5 V) que é incompatível com os -15 V e os +15 V especificados pelo protocolo RS232. Assim, será necessário adquirir uma placa de interface que efetue a conversão entre os dois níveis, permitindo a interligação com o sistema de domótica. A escolha caiu sobre a placa da CuteDigi que se apresenta na Figura 14 e que é baseada no circuito integrado MAX 232 da Maxim. A CuteDigi disponibiliza os sinais RX e TX devidamente condicionados para a interligação direta com o módulo de interface da Only.

Esta placa, à semelhança de muitas congéneres disponíveis no mercado, encaixa nos conectores de expansão da placa Fez Domino, permitindo que os sinais passem para outras placas que venham a ser supra-encaixadas. Cabe, uma vez mais, ao utilizador o cuidado de utilizar em cada placa apenas os sinais ainda não utilizados pelas outras de modo a evitar eventuais danos no *hardware*.

Figura 14 **Placa RS232 da CuteDigi [46]**

O módulo de interface da Only necessita de ser alimentado a 5 V e utiliza os pinos *Data Terminal Ready* (DTR) e *Request To Send* (RTS) para essa finalidade. Assim, a placa da CuteDigi será alterada de modo a colocar nos referidos pinos 5 V.

Para a ligação à porta série síncrona vai ser utilizada *a Serial Peripheral Interface Bus* (SPI) do Fez Domino. Dado que o *chipset* W5100 já inclui todas as funcionalidades do protocolo TCP, escolheu-se a placa Fez Connect que se apresenta na Figura 15.

Figura 15 **Placa Fez Connect [47]**

Esta placa, que é também compatível com os sistemas baseados no Arduino, possui, para além da interface de rede, outras funcionalidades como se mostra na Figura 16. Destas, apenas o relógio em tempo real (RTC) irá ser utilizado.

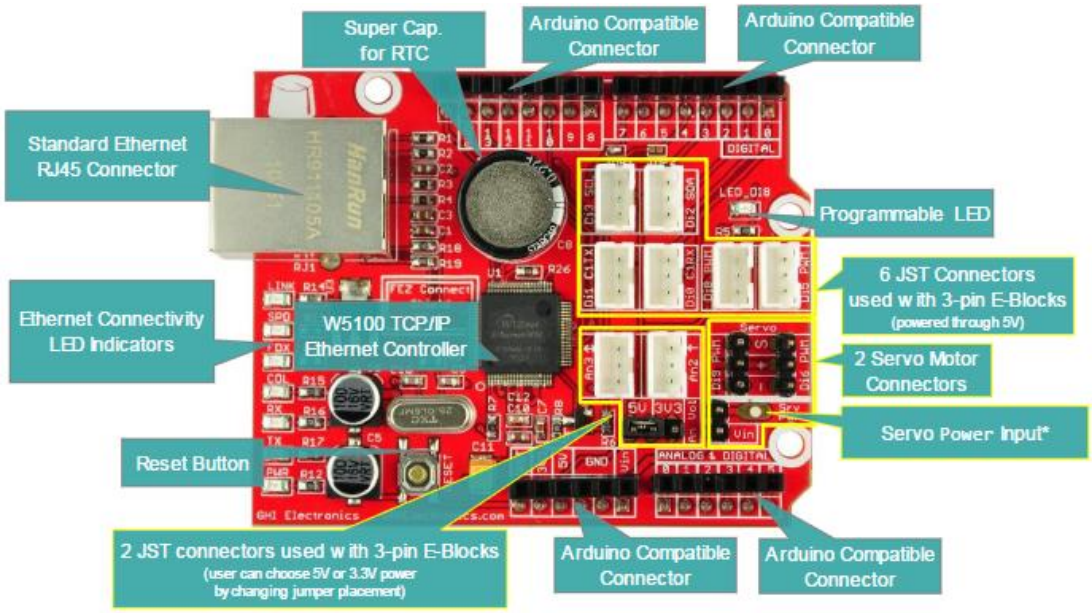

Figura 16 **Funcionalidades da placa Fez Connect [48]**

A Figura 17 apresenta configuração final do *hardware* de controlo após a interligação destes três módulos.

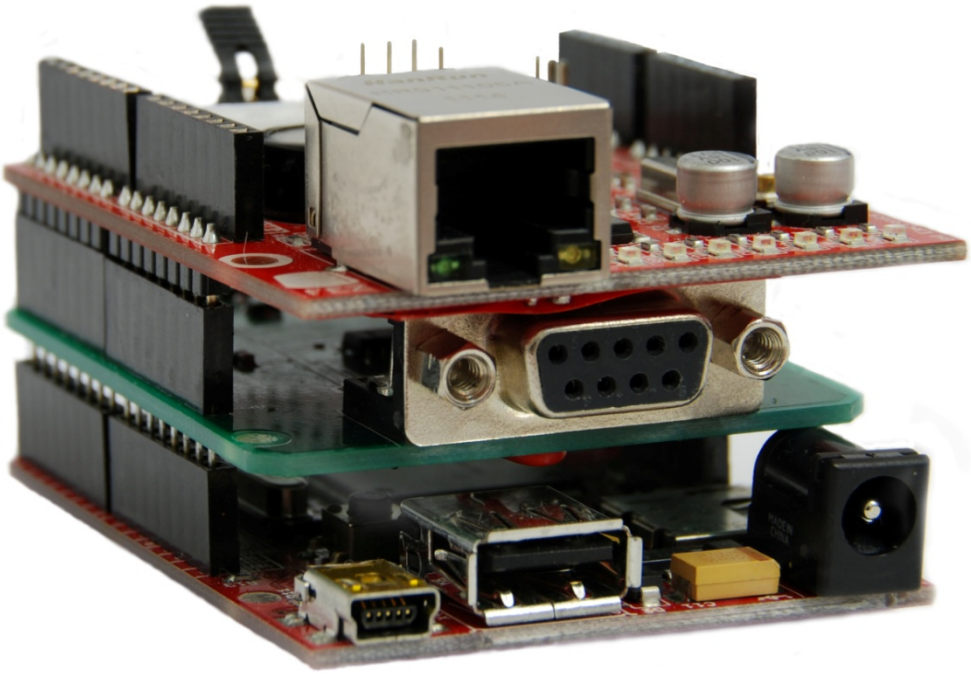

Figura 17 ***Hardware* de controlo**

3.5. CONCLUSÃO

Neste projeto adotaram-se os sistemas de desenvolvimento integrados Microsoft Visual Studio 2010 para o desenvolvimento em linguagem C# e Microsoft SQL Server Management Studio 2008 como ferramenta de configuração da base de dados Microsoft SQL Server Express. Para desenvolver as páginas *Web* foram integradas as várias tecnologias descritas neste capítulo. A interface entre o barramento de domótica e o sistema de controlo da instalação é efetuada através do *hardware* apresentado.

No próximo capítulo descreve-se o desenvolvimento da ferramenta, referindo-se os seus módulos constituintes e detalhando-se a sua arquitetura e funcionalidades.

4. DESENVOLVIMENTO

Neste capítulo descreve-se a arquitetura do sistema desenvolvido, detalhando-se as funcionalidades dos diferentes módulos e a sua interligação. Apresentam-se também os principais problemas ocorridos, as soluções encontradas, os testes realizados e os resultados obtidos.

4.1. ARQUITETURA

A ferramenta de apoio ao projeto, configuração e gestão de instalações domóticas baseadas no modelo Only é constituída por cinco módulos:

- Gestão do projeto;
- Configuração da instalação;
- Servidor de base de dados;
- Gestão da instalação;
- *Hardware* de interface.

Os dois primeiros módulos são destinados à utilização exclusiva dos técnicos da Live Simply. O servidor de bases de dados armazena os dados que são gerados pelos dois primeiros módulos. O último módulo é dedicado ao controlo dos módulos da instalação.

A Figura 18 apresenta uma vista global dos módulos constituintes do sistema.

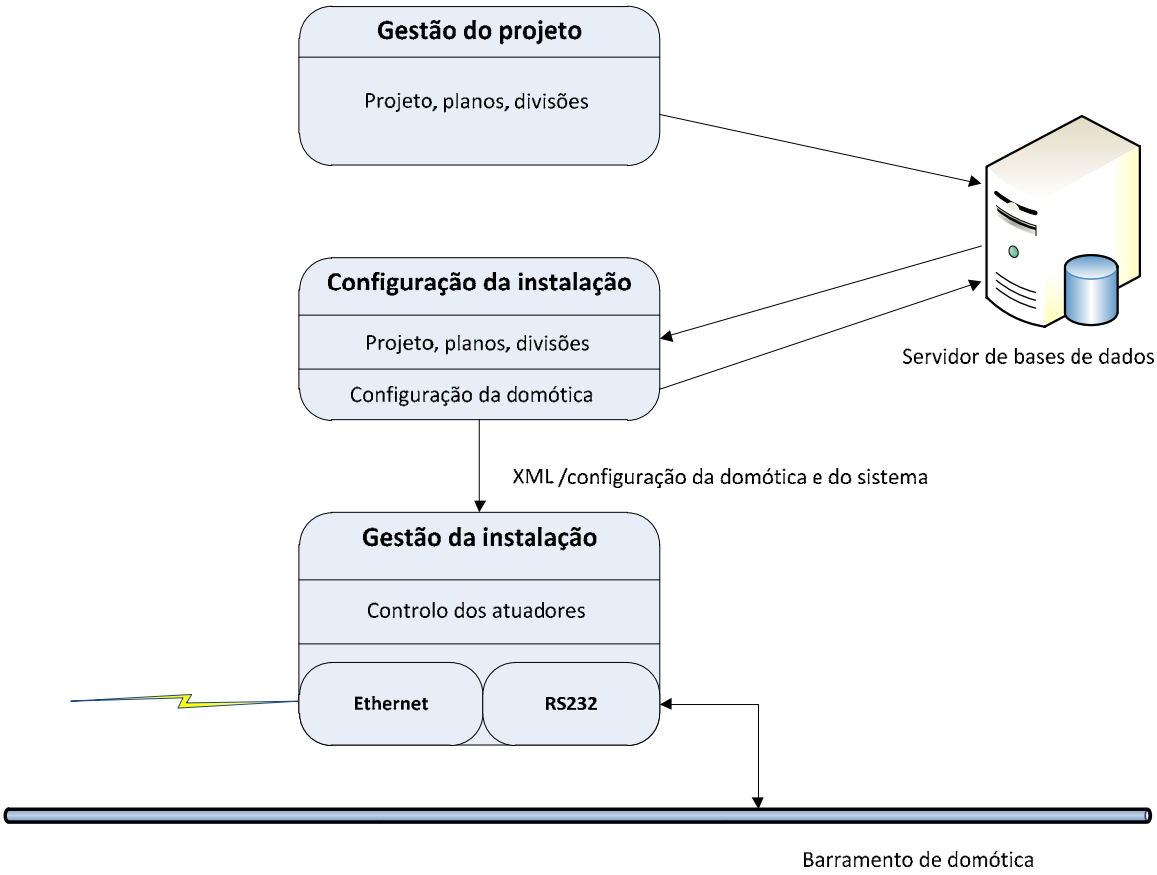

Figura 18 **Arquitetura do sistema**

4.2. GESTÃO DO PROJETO

O módulo de Gestão do Projeto permite a criação e atualização de projetos mediante o armazenamento dos dados dos diferentes clientes e da definição da arquitetura do imóvel, ou seja, dos diferentes planos e divisões por plano.

Embora o conceito de plano e divisão seja intuitivo, a aplicação é flexível, permitindo ao utilizador aplicar diferentes conceitos desde que tenha em conta as restrições da utilização do *hardware* (módulos Only) definidas pelo fabricante.

Este módulo foi desenvolvido em C# (`LSprj`) e é responsável não só pela interface gráfica com o utilizador, mas, também, pelo armazenamento e tratamento da informação no servidor de base de dados. As principais funções que oferece ao utilizador são a visualização, edição e eliminação de projetos.

O utilizador, quando cria um novo projeto, introduz o nome do projeto e especifica um conjunto de dados complementares que incluem a entidade, o endereço, o contacto e ainda um campo opcional de observações. Ao novo projeto é automaticamente atribuído um código numérico (único) que é composto pelo ano em curso, seguido de um número sequencial composto por 3 dígitos. Por exemplo, ao terceiro projeto criado no ano de 2012 corresponde o código 2012003. Por último, atribui-se ao projeto um estado. Os estados possíveis são 'Adjudicado', Concluído', 'Em execução' e 'Orçamentado'. Convencionou-se que o estado inicial é, por omissão, 'Orçamentado'.

Na opção de visualização é possível selecionar um projeto da lista de projetos gravados ou aplicar filtros que permitam selecionar os projetos que obedeçam a um determinado estado. Depois de selecionado um projeto, o utilizador pode visualizar todos os dados do projeto assim como o histórico das suas mudanças de estado.

A edição de um projeto permite ao utilizador alterar qualquer um dos atributos do projeto com a exceção do código do projeto que, uma vez automaticamente atribuído, não é alterável.

A opção de eliminação permite remover da base de dados toda a informação relativa a um projeto, incluindo os registos relativos aos planos e divisões. À semelhança dos projetos, também os planos e as divisões podem ser alterados ou eliminados. A eliminação de um plano tem como consequência a eliminação de todos as divisões associadas ao plano.

Depois de criado um projeto, o utilizador pode proceder à caracterização do imóvel, definindo os diversos planos e, caso pretenda, adicionando uma imagem a cada plano criado. Para cada um dos planos criados podem ser definidas divisões, sendo igualmente possível adicionar uma imagem a cada divisão. A adição de imagens aos planos e divisões permite associar as respetivas plantas ao projeto. Só depois do imóvel ter sido caracterizado é que é possível configurar a instalação.

A aplicação de gestão de projetos é constituída essencialmente por seis ficheiros:

– `frmSimply.cs` – *form* principal;

– `frmProject.cs` – dedicado à introdução e alteração de dados dos projetos;

– `frmPlan.cs` – dedicado à introdução e alteração de dados dos planos;

– `frmDivision.cs` – dedicado à introdução e alteração de dados das divisões;

– `SQLutils.cs` – dedicado à interface com a base de dados;

– `appSettings.settings` – ficheiro de configuração da aplicação – contém parâmetros relevantes para a aplicação como o nome do servidor SQL e o nome da base de dados.

A Figura 19 apresenta o projeto com o código 2012001 no estado 'Orçamentado'.

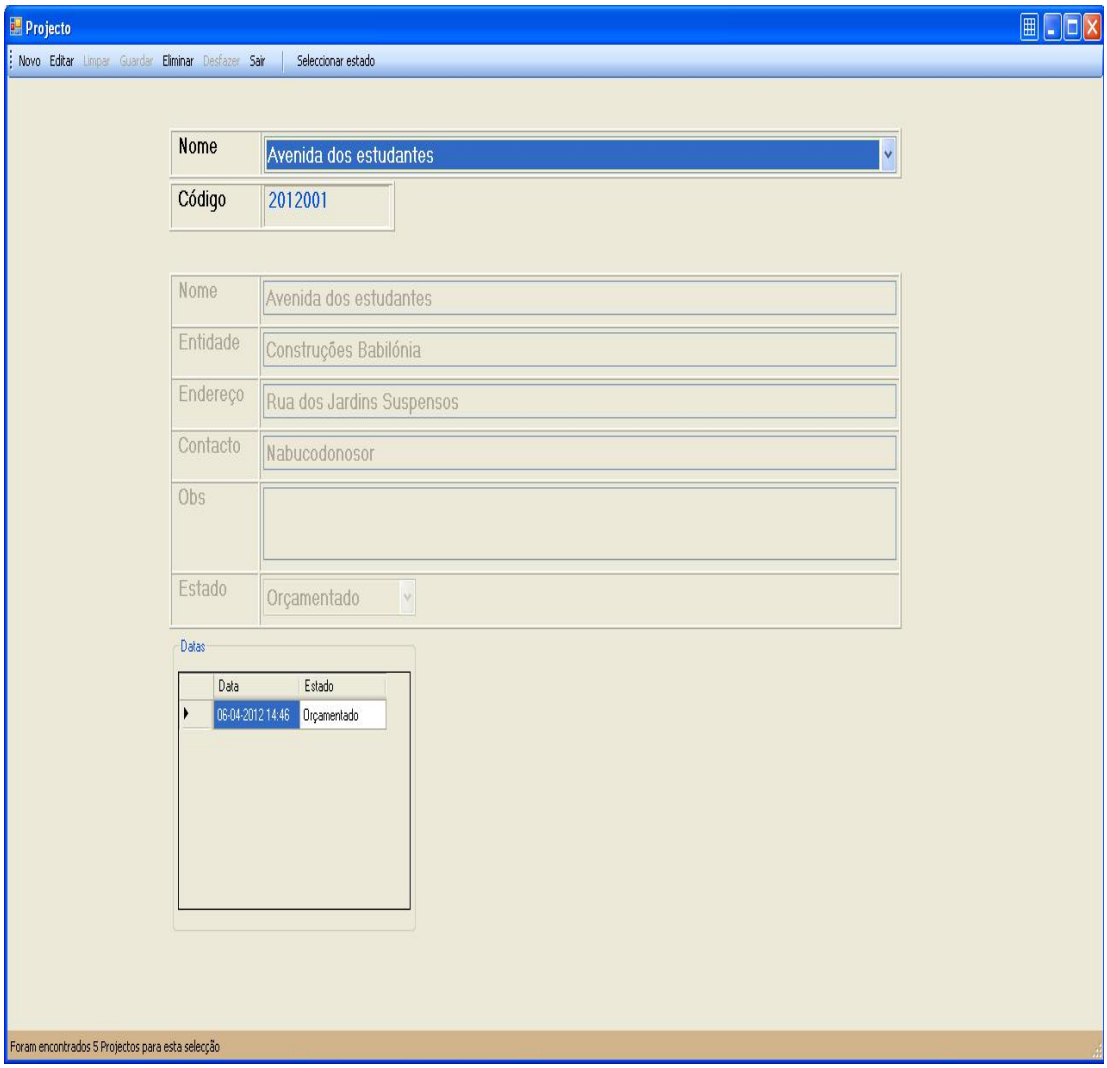

Figura 19 **Projeto**

A Figura 20 mostra o mesmo projeto já com o plano do R/C criado e a respetiva imagem.

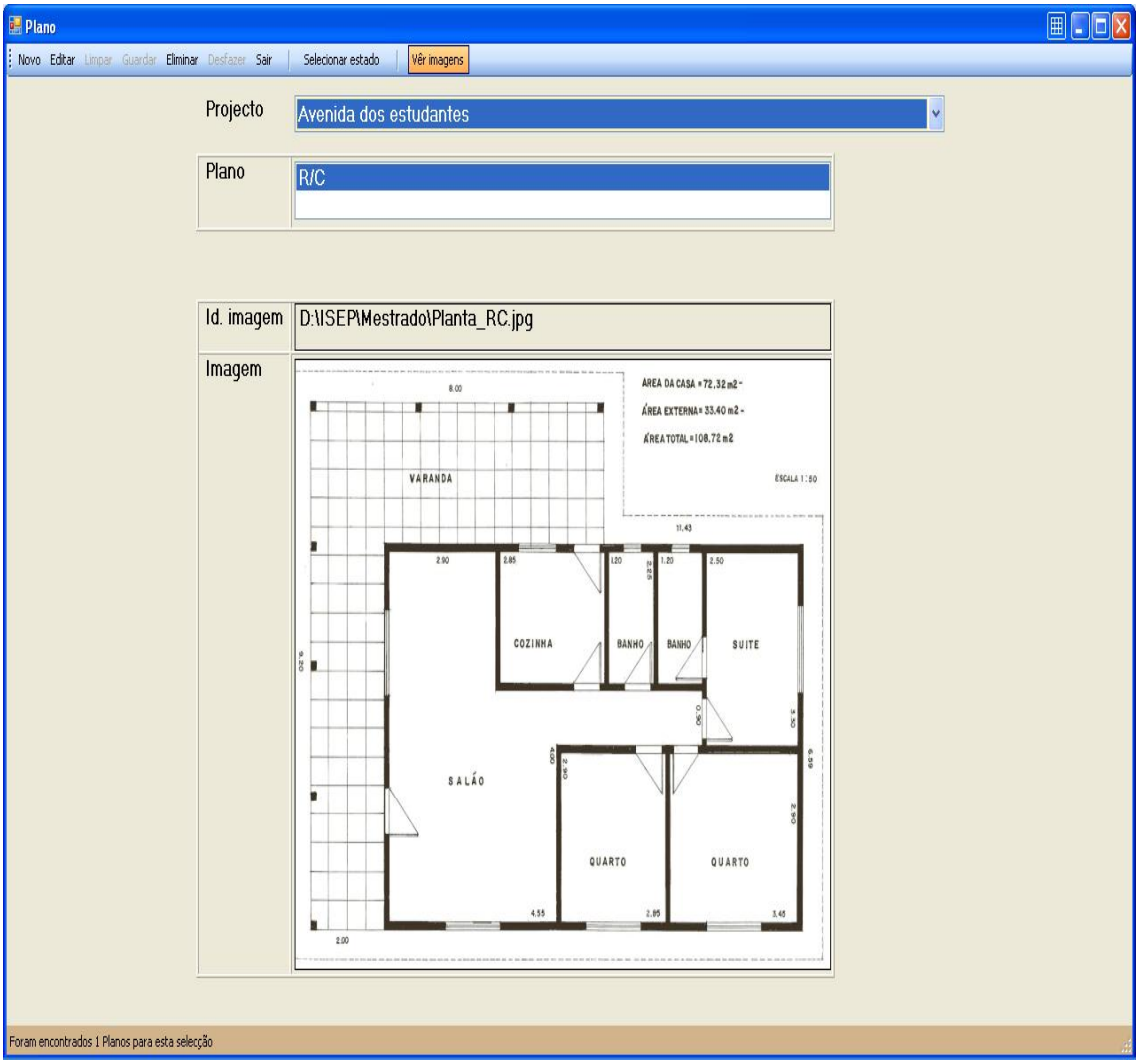

Figura 20 **Plano**

As imagens são guardadas no disco duro ou noutro suporte indicado na altura da criação dos planos e das divisões, sendo guardados na base de dados apenas o caminho e o nome do ficheiro.

A classe '`SQLutils`' oferece um conjunto de funcionalidades que estabelecem a interligação entre a aplicação e a base de dados. Estas funcionalidades foram implementadas através de *stored procedures*.

No Excerto de Código 1 apresenta-se o método '`UpdateDivision`' que atualiza na base de dados os dados relativos à divisão que é passada como parâmetro de entrada. O nome da divisão é único por plano assim como o nome do plano é único por projeto. No entanto, é possível utilizar o mesmo nome de plano e de divisão em diferentes projetos. Por esta

razão, este método inclui entre os seus parâmetros de entrada o nome do plano e do projeto. O método devolve um valor do tipo byte que corresponde ao resultado retornado pela *stored procedure* que é zero no caso de ter sucesso ou outro valor nos restantes casos.

```
public byte UpdateDivision(string DivisionName, string
   OldDivisionName, string PlanName, string ProjectName,
   string ImgFileName) {
   SQLErrorInfo = "";
   try {
     SqlCommand sqlCmd = new SqlCommand("UpdateDivision", sqlconn);
     sqlCmd.CommandType = CommandType.StoredProcedure;
     sqlCmd.Parameters.Add("@myDivisionName",
           SqlDbType.VarChar, 50).Value = DivisionName;
     sqlCmd.Parameters.Add("@myOldDivisionName",
           SqlDbType.VarChar, 50).Value = OldDivisionName;
     sqlCmd.Parameters.Add("@myPlanName", SqlDbType.VarChar,
           50).Value = PlanName;
     sqlCmd.Parameters.Add("@myProjectName",
           SqlDbType.VarChar, 100).Value = ProjectName;
     sqlCmd.Parameters.Add("@myImg", SqlDbType.VarChar,
           200).Value = ImgFileName;
     SqlParameter myParam =
           sqlCmd.Parameters.Add("@RetValue",
           SqlDbType.TinyInt, 0);
     myParam.Direction = ParameterDirection.ReturnValue;
     sqlconn.Open();
     sqlCmd.ExecuteNonQuery();
     sqlconn.Close();
     byte er = Convert.ToByte(myParam.Value);
     GetErrorByNumber(er);
     return er;
   }
   catch (Exception ex) {
     if (sqlconn.State == ConnectionState.Open)
           sqlconn.Close();
     SQLErrorInfo = ex.Message;
     return 99;
   }
}
```

Excerto de Código 1 **Método '`UpdateDivision`'**

O método '`GetErrorByNumber`' determina o texto correspondente ao erro encontrado, conforme se pode ver no Excerto de Código 2. A propriedade '`SQLErrorInfo`' é atualizada em função do erro encontrado.

```
public void GetErrorByNumber(byte Error) {
   switch (Error) {
     case 0:
       SQLErrorInfo = "Sucesso";
       break;
     case 1:
       SQLErrorInfo = "Valor já existe";
       break;
     case 2:
       SQLErrorInfo = "Valor não existe";
       break;
     default:
       SQLErrorInfo = "Desconhecido";
       break;
   }
}
```

Excerto de Código 2 **Método '`GetErrorByNumber`'**

4.3. CONFIGURAÇÃO DA INSTALAÇÃO

Neste módulo, que permite definir para cada divisão os circuitos elétricos e os elementos ativos que serão utilizados, são relevantes a identificação do projeto, planos e divisões por plano definidos na aplicação `LSprj`. Os teclados (entradas do sistema) e os atuadores (saídas do sistema) são designados elementos ativos. Em relação a cada um destes elementos é possível alterar a configuração de fábrica, configurando-os para se obterem as funcionalidades pretendidas.

No âmbito deste projeto, de acordo com a terminologia adotada pelos técnicos da Live Simply, os teclados são designados painéis.

Este módulo, que consiste na aplicação `LSinst` desenvolvida em C#, exibe uma interface gráfica para facilitar a configuração da instalação. Os dados de configuração de cada módulo e as suas interligações são armazenados na base de dados e relacionados com o projeto, plano e divisão a que dizem respeito. A integridade da configuração da instalação é automaticamente verificada e o técnico é alertado sempre que sejam detetadas inconsistências.

A configuração, depois de validada, é também utilizada pelo módulo Gestão da Instalação. Por esta razão, a informação de configuração da instalação é adicionalmente armazenada num formato facilmente transportável e interpretável. Assim, a informação de configuração da instalação, além de ser armazenada na base de dados, é gravada num ficheiro XML que é, por sua vez, copiado para um cartão do tipo SD.

A aplicação é constituída por cinco ficheiros:

- `frmPlanning.cs` – *form* principal;
- `FTC232.cs` – classe que implementa o protocolo de comunicação dos módulos de domótica;
- `SQLutils.cs` – classe que implementa a interface com a base de dados;
- `Utils.cs` – classe com um conjunto de métodos utilitários de conversão entre diferentes tipos de dados;

– `appSettings.settings` – Ficheiro de configuração da aplicação que contém os parâmetros relevantes para a aplicação e cujo conteúdo se ilustra na Tabela 4.

Tabela 4 LsInst – Parâmetros

Name	Type	Scope	Value
ComPort	string	Application	COM6
ServerName	string	Application	ASUSJC\SQLEXPRESS
DBname	string	Application	Domus
ImageListPath	string	Application	D:\ISEP\Mestrado\Botoes\
InputPrgDefaultValue	string	Application	TOGGLE
AppImgsPath	string	Application	D:\ISEP\Mestrado\AppImgs\
CircuitPrefix	string	Application	Cir_
ActuatorPrefix	string	Application	Act_
tmrInterval	int	Application	200
XMLfolder	string	Application	D:\ISEP\Mestrado\Programas\Configuracao\
XSDfile	string	Application	D:\ISEP\Mestrado\Programas\Configuracao\domotica.xsd
rxTimeout	int	Application	3000
longDelayBetweenMsgs	int	Application	300

A aplicação `LSinst` induz o utilizador a seguir a sequência de trabalho mais aconselhada para a configuração da instalação.

Na Figura 21 apresenta-se o aspeto gráfico da aplicação. Neste caso foi selecionado o projeto '2012001', mas ainda não foram definidos quaisquer circuitos ou elementos ativos.

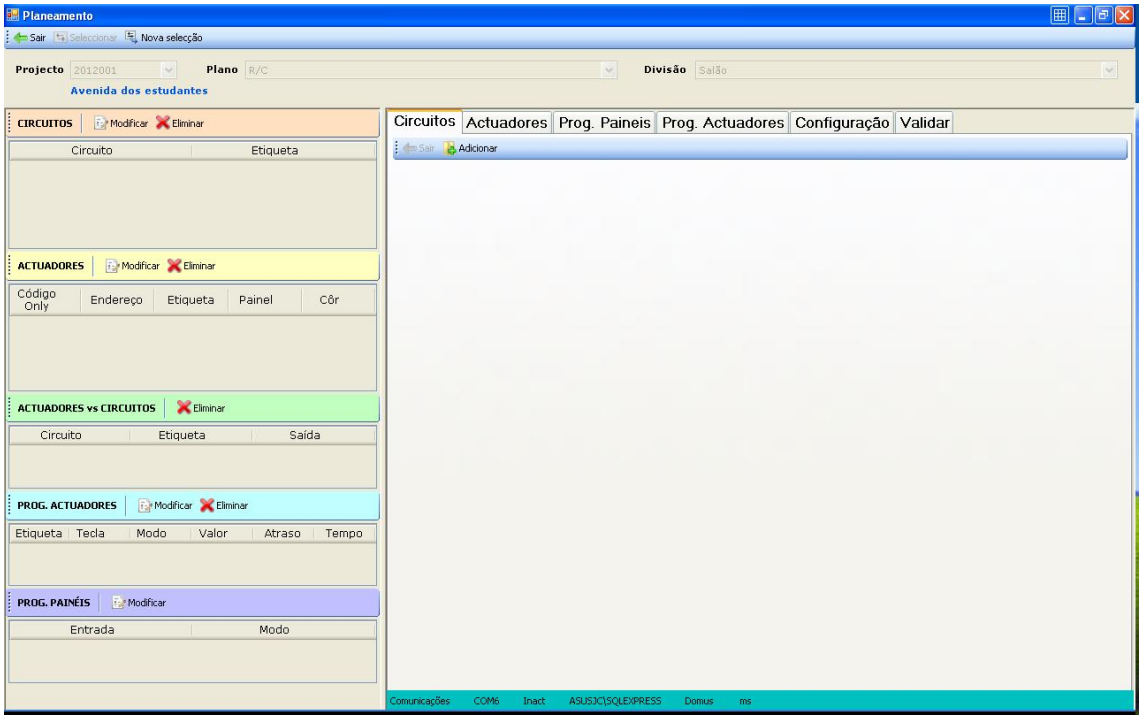

Figura 21 **Configuração vazia**

De seguida descrevem-se os diversos passos de configuração ou ‘Planeamento’ da instalação.

4.3.1. CIRCUITOS

O utilizador escolhe, para cada divisão selecionada, os circuitos que pretende controlar, utilizando o menu ‘adicionar’ e escolhendo os tipos de circuitos da lista apresentada. A lista de circuitos disponíveis é composta pelos circuitos previamente introduzidos na base de dados. A título de ajuda é exibida a imagem do item selecionado assim como o prefixo escolhido para etiqueta ou descrição do circuito. O prefixo dos circuitos, que é ‘Cir_’, está definido no ficheiro de parâmetros (Tabela 4). A seleção de um circuito para alteração ou eliminação, é efetuada na secção do lado esquerdo.

4.3.2. ATUADORES

Existem vários tipos de atuadores e a sua escolha pode ser feita com a ajuda dos diferentes filtros que a aplicação disponibiliza. Assim, para um atuador da linha ‘OnlyTouch’ e do tipo ‘On/OFF’ existem quatro escolhas possíveis como se mostra na Figura 22.

Figura 22 **Escolha do atuador**

Assim que o utilizador seleciona um tipo de atuador, a aplicação sugere automaticamente a utilização de um painel adequado de cor branca. O utilizador pode, de entre as opções apresentadas pela aplicação, escolher valores diferentes dos sugeridos. No entanto, para evitar inconsistências, a aplicação limita a escolha ao conjunto de combinações de

atuadores e painéis possíveis, impedindo o utilizador de efetuar escolhas de funcionamento incompatível.

Após a seleção e adição de um atuador ao projeto, o utilizador pode escolher os circuitos a associar às saídas do atuador. O utilizador pode verificar na secção do lado esquerdo da interface as ligações definidas entre as saídas dos atuadores escolhidos e os circuitos da instalação.

A aplicação permite modificar a etiqueta ou o painel associado a um atuador e eliminar um atuador. A remoção de um atuador implica a eliminação, por um lado, das ligações estabelecidas entre o atuador e os circuitos da instalação e, por outro lado, da programação dos painéis ligados ao atuador. A configuração dos painéis tem de ser removida porque os painéis estão fisicamente ligados a atuadores para conseguirem comunicar com o barramento de domótica. Existem módulos desprovidos de saídas que satisfazem as situações em que não são necessárias saídas e apenas é pretendida a comunicação com o barramento.

É possível, no quadro das ligações entre saídas de atuadores e circuitos da instalação, eliminar uma ou mais relações. Nesses casos, os circuitos em questão perderão a sua associação às respetivas saídas dos atuadores, ficando disponíveis para serem associados a outros atuadores.

4.3.3. PROGRAMAÇÃO DE PAINÉIS

A programação de um painel permite escolher o modo de funcionamento de cada uma das suas entradas, ou seja, das suas teclas. A Figura 23 mostra a configuração da ‘Entrada 1’ que corresponde à tecla 1 do painel.

Em relação ao painel, a aplicação apresenta ainda o atuador a que este está fisicamente ligado, permitindo que o instalador referencie os módulos e, assim, evite erros de montagem em instalações com elevado número de módulos. Na Figura 23 verifica-se que o painel ‘OT-4S-G’ (painel com quatro teclas) está fisicamente ligado ao atuador ‘CO-2OUT’ (atuador de duas saídas) e que a tecla número 1 terá a função ‘TOGGLE’.

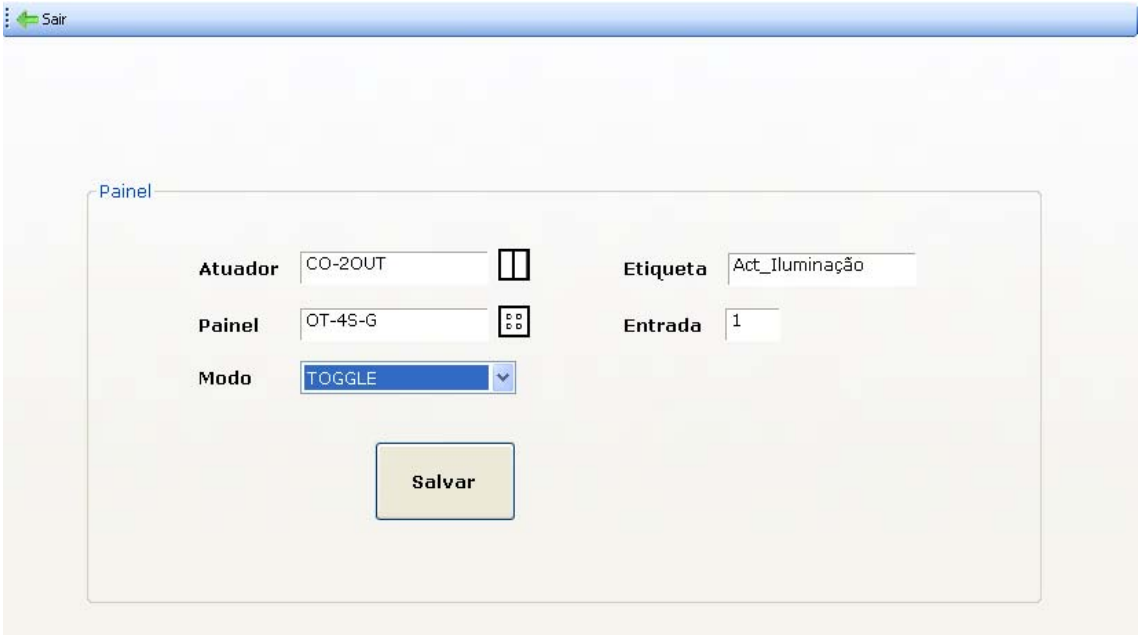

Figura 23 **Programação das teclas**

Na base de dados encontra-se definido para cada painel a função que cada tecla tem por omissão (tabela `PanelsDefaultConfig`). A aplicação permite alterar a funcionalidade de cada tecla de acordo com as necessidades da instalação.

4.3.4. PROGRAMAÇÃO DE ATUADORES

A programação de um atuador tem por finalidade definir o comportamento de cada uma das suas saídas em função das teclas que podem ser premidas. As teclas podem pertencer ao painel fisicamente associado ao atuador em questão ou a qualquer outro painel. Por exemplo, é possível ligar uma saída com a tecla 2 de um módulo e desligá-la com a tecla 4 de outro módulo. Diferentes configurações ou programações podem ser efetuadas em função dos requisitos do cliente como, por exemplo, desligar toda a iluminação e baixar todos os estores premindo apenas uma tecla.

Na Figura 24 apresenta-se a programação da saída número 1 do atuador ‘CO-1DIM’ (secção atuador presente). Caso a tecla número 1 do painel ‘OT-4S-S’ (associada ao atuador remoto) esteja programada para o estado ‘OFF’, a saída irá ser desligada. Neste exemplo o atuador presente e o atuador remoto não são o mesmo, ou seja, o painel está ligado fisicamente a um outro atuador, o ‘CO-1SHU’.

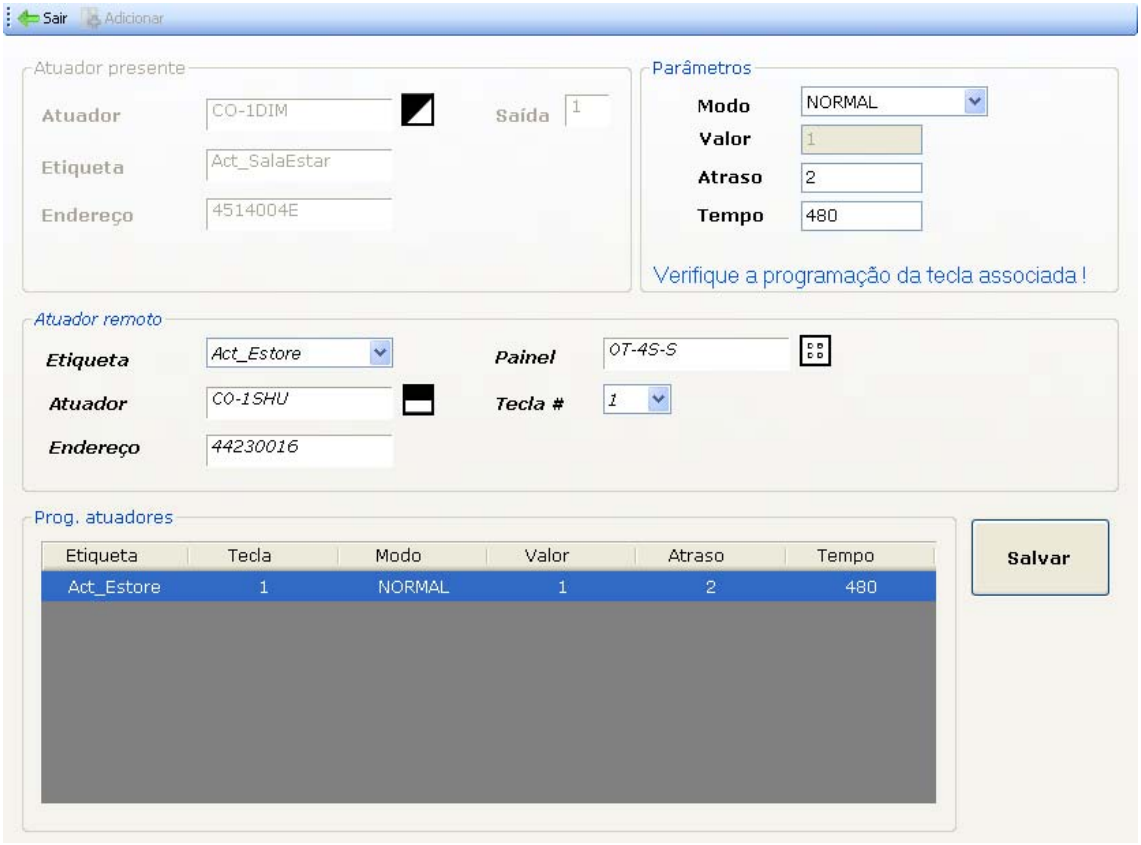

Figura 24 **Programação de uma saída**

Pode-se remover ou modificar a programação de qualquer saída de um atuador.

Os técnicos podem realizar toda esta configuração sem necessidade de se deslocarem às instalações do cliente. No entanto, no final desta fase, os campos 'Endereço' dos atuadores estão por definir.

4.3.5. CONFIGURAÇÃO

Uma vez definidos os circuitos, adicionados os atuadores, configurados os painéis e atuadores é necessário emparelhar os endereços dos atuadores definidos na aplicação com os atuadores físicos. Nesta etapa, o técnico tem de se deslocar às instalações do cliente, montar a rede de domótica, ligar os módulos e ler os endereços físicos de cada um dos atuadores previamente definidos.

A classe `FTC232.cs` é responsável pela implementação do protocolo de comunicação e providencia todas as funcionalidades necessárias aos sistemas de automação e de climatização, embora só os primeiros sejam alvo de estudo neste trabalho. Esta classe implementa uma fila onde são armazenadas as mensagens lidas do barramento. Cada vez

que chega um byte à porta série é gerado um evento que, por sua vez, invoca um método que armazena o valor recebido. Para evitar a coexistência de processos concorrentes de escrita e leitura sobre esta fila, foi implementado um processo de *locking*. As mensagens são lidas da fila através de um mecanismo de *pooling* com um intervalo de tempo definido no ficheiro de parâmetros (Tabela 4). Depois de vários testes, adotou-se o valor de 200 ms.

Na Figura 25 mostram-se as mensagens lidas do barramento depois de decompostas nos seus elementos mais relevantes.

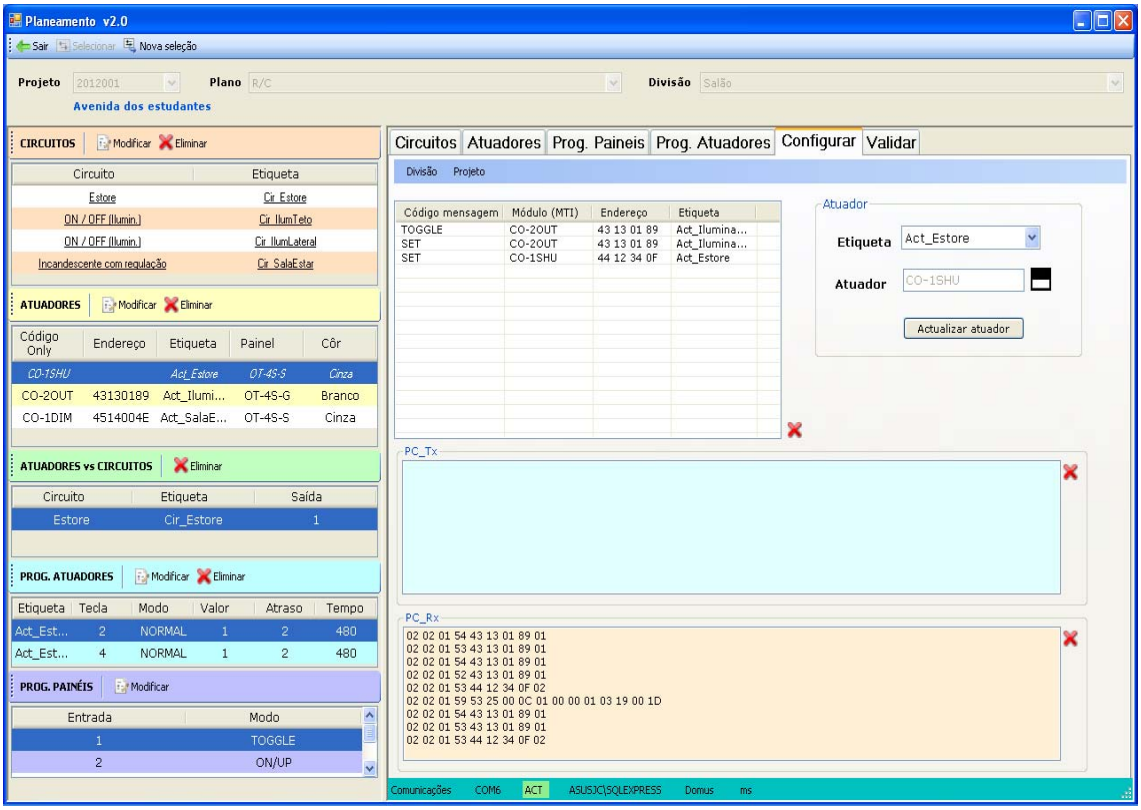

Figura 25 **Configuração (endereços)**

Para cada um dos atuadores registados na aplicação e ainda sem endereço definido (identificados na lista pela sua etiqueta), é possível encontrar o módulo físico correspondente e atribuir ao primeiro o endereço do segundo premindo o botão ‘Atualizar atuador’. Para que a mensagem correspondente a um módulo apareça no barramento e, por sua vez, na aplicação, o técnico deve premir uma tecla do módulo; ao fazê-lo o módulo envia uma mensagem para o barramento. Depois de todos os módulos da aplicação adquirirem um endereço, é possível validar a configuração, selecionando a opção ‘Divisão -> Validar’. O processo de validação termina com a criação de um ficheiro XML que contém toda a caracterização da instalação.

A título de exemplo, no Excerto de Código 3 apresenta-se uma parte do ficheiro de configuração da instalação que segue o esquema definido no ficheiro ‘`domotica.xsd`’.

```
<?xml version="1.0"?>
<Projecto xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\ISEP\Mestrado\Programas\Configura
caoInst\domotica.xsd">
    <Nome>Avenida dos estudantes</Nome>
    <Codigo>2012001</Codigo>
    <Plano>
        <Nome>R/C</Nome>
        <IdPlano>49</IdPlano>
        <Divisao>
            <Nome>Salão</Nome>
            <IdDivisao>36</IdDivisao>
            <Atuador>
                <Nome>CO-1SHU</Nome>
                <Endereco>4412340F</Endereco>
                <Etiqueta>Act_Estore</Etiqueta>
                <Painel>
                    <Nome>OT-4S-S</Nome>
                    <Cor>Cinza</Cor>
                    <Tecla>
                        <Num>1</Num>
                        <Modo>TOGGLE</Modo>
                    </Tecla>
                    <Tecla>
                        <Num>2</Num>
                        <Modo>ON/UP</Modo>
                    </Tecla>
                    <Tecla>
                        <Num>3</Num>
                        <Modo>TOGGLE</Modo>
                    </Tecla>
                </Painel>
                <Saida>
                    <Num>1</Num>
                    <Circuito>
                        <Nome>Estore</Nome>
                        <IdCircuito>128</IdCircuito>
                        <Etiqueta>Cir_Estore</Etiqueta>
                    </Circuito>
                    <Memoria>
                        <Tecla>2</Tecla>
                        <Endereco>4412340F</Endereco>
                        <Modo>NORMAL</Modo>
                        <Valor>1</Valor>
                        <Atraso>2</Atraso>
                        <Tempo>480</Tempo>
                    </Memoria>
                    <Memoria>
                        <Tecla>4</Tecla>
                        <Endereco>4412340F</Endereco>
                        <Modo>NORMAL</Modo>
                        <Valor>1</Valor>
                        <Atraso>2</Atraso>
                        <Tempo>480</Tempo>
                    </Memoria>
                </Saida>
            </Atuador>

…
```

Excerto de Código 3 **Ficheiro de configuração da instalação**

No caso da validação estar correta, é possível passar para o próximo passo, selecionando o menu ‘Projeto’. Este menu permite-nos escolher uma das seguintes funcionalidades:

– Configuração de fábrica;

– Programar;

– *Backup*;

– *Restore*.

A opção ‘Configuração de fábrica’ envia uma instrução que comanda os módulos a carregarem a configuração de fábrica. É bastante útil para recuperar de uma configuração errada ou para realizar testes com novas configurações.

A opção ‘Programar’, ilustrada na Figura 26, envia para cada módulo a programação definida pela aplicação. Esta ação desenrola-se em três etapas:

– Envio das configurações de fábrica;

– Programação das entradas (painéis);

– Programação das saídas (atuadores).

Na primeira etapa enviam-se as configurações de fábrica. Caso as seguintes etapas (mais críticas) sejam mal sucedidas, garante-se que são carregadas as configurações definidas pelo fabricante. Na segunda etapa envia-se a programação de cada um dos painéis que configura a funcionalidade pretendida em cada tecla. Por último, envia-se a configuração de cada um dos atuadores, ou seja, define-se o que deve ser transmitido através das saídas dos atuadores quando se primem as teclas dos diferentes painéis.

A opção ‘*Backup*’ guarda na memória EEPROM[15] do atuador a configuração ativa (a programação das suas saídas).

A opção ‘*Restore*’ carrega e ativa a configuração guardada na EEPROM.

15 *Electrically-Erasable Programmable Read-Only Memory* (EEPROM) designa um tipo de memória de armazenamento não volátil com a capacidade de ser programada eletricamente.

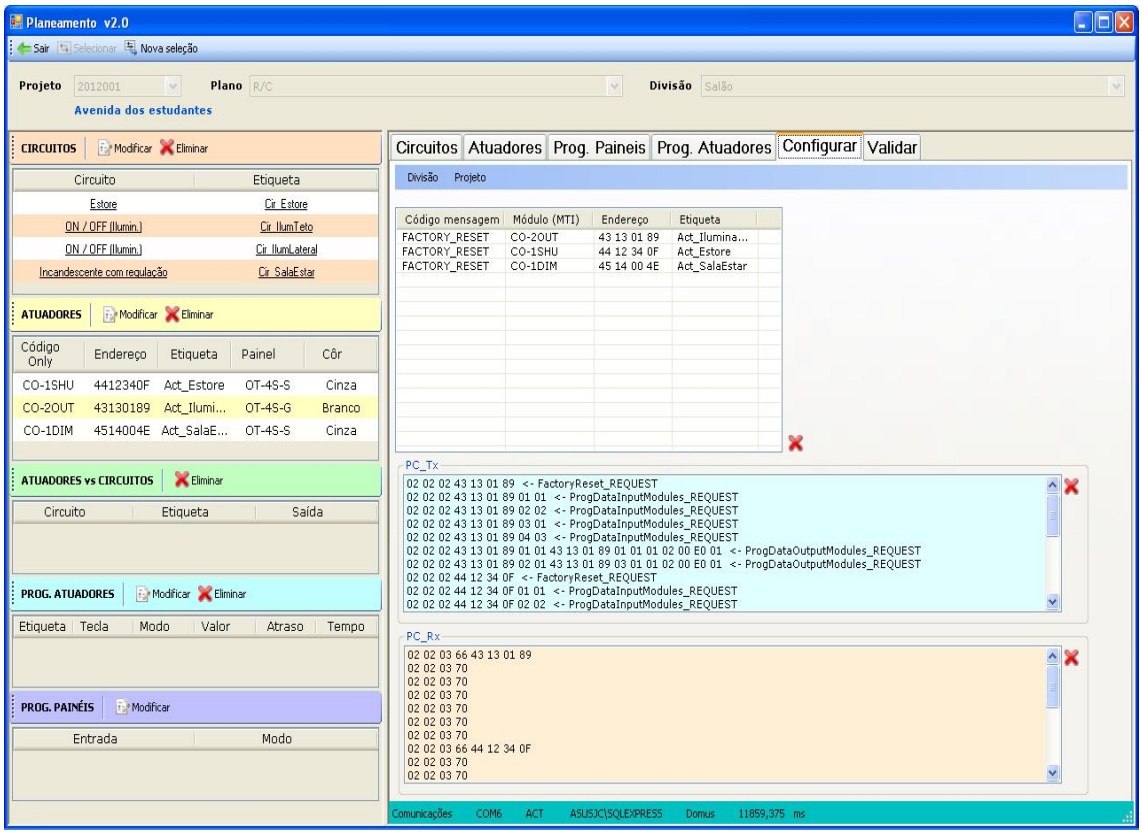

Figura 26 **Programação dos módulos**

4.3.6. VALIDAÇÃO

Como complemento à ação de 'Validação' (ver menu Divisão -> Validar na sub-subsecção 4.3.5), é possível identificar com detalhe se a configuração apresenta eventuais inconsistências com recurso ao menu 'Validar'.

Na Figura 27 apresenta-se um exemplo de uma configuração com um erro. Neste caso, o circuito de iluminação 'Cir_Ilum2' do tipo ON/OFF não está ligado a nenhum atuador. Quando se seleciona 'Guardar', esta informação é exportada para um ficheiro do tipo folha de cálculo (Microsoft Excel) conforme se ilustra na Figura 28.

O processo de validação verifica os seguintes aspetos:

- Circuitos com descrição (etiquetas);
- Atuadores com descrição;
- Atuadores com endereços;
- Circuitos ligados a uma saída de um atuador;

– Utilização das saídas dos atuadores;

– Programação das saídas dos atuadores.

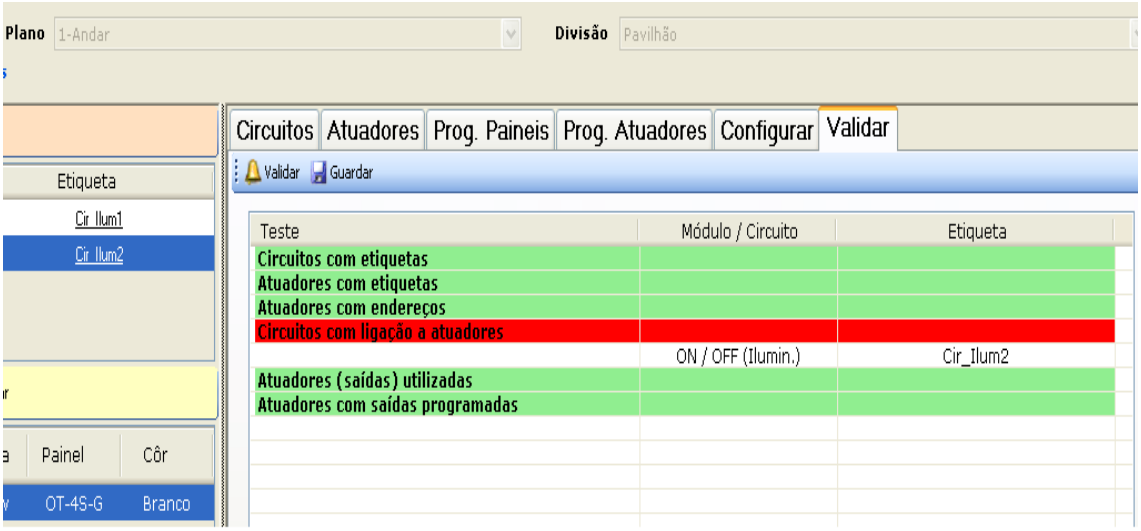

Figura 27 **Visualização do detalhe da validação**

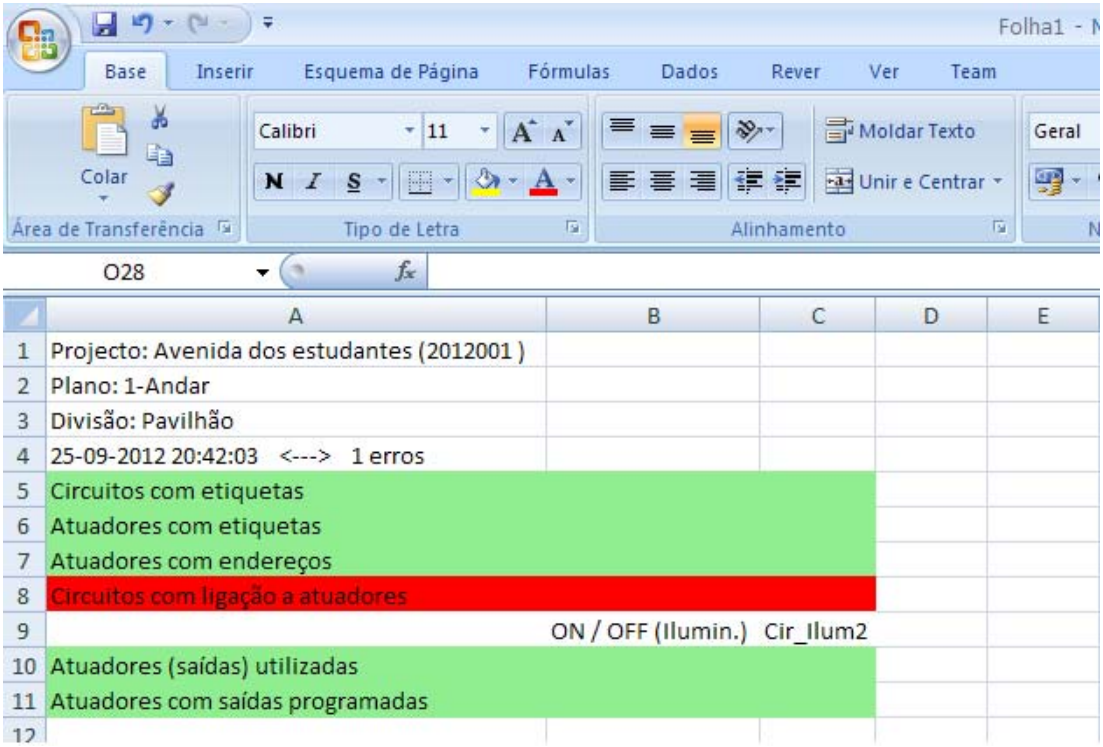

Figura 28 **Ficheiro com a análise da validação**

4.4. SERVIDOR DE BASE DE DADOS

Os dados persistentes necessários ao funcionamento das diferentes aplicações assim como os respetivos procedimentos (*stored procedures*) de acesso e processamento estão

armazenados numa base de dados alojada no servidor de base de dados Microsoft SQL Server Express.

Os dados armazenados podem pertencer a uma das seguintes categorias:

- Dados base do projeto onde se incluem a informação relativa aos planos e divisões;
- Dados base que caracterizam os módulos bem assim como as suas funcionalidades;
- Informação relativa à configuração da instalação, *i.e.*, quais os circuitos e módulos utilizados em cada uma das divisões e a sua respetiva configuração.

Na Figura 29 apresenta-se o conjunto das 25 tabelas que constituem a base de dados.

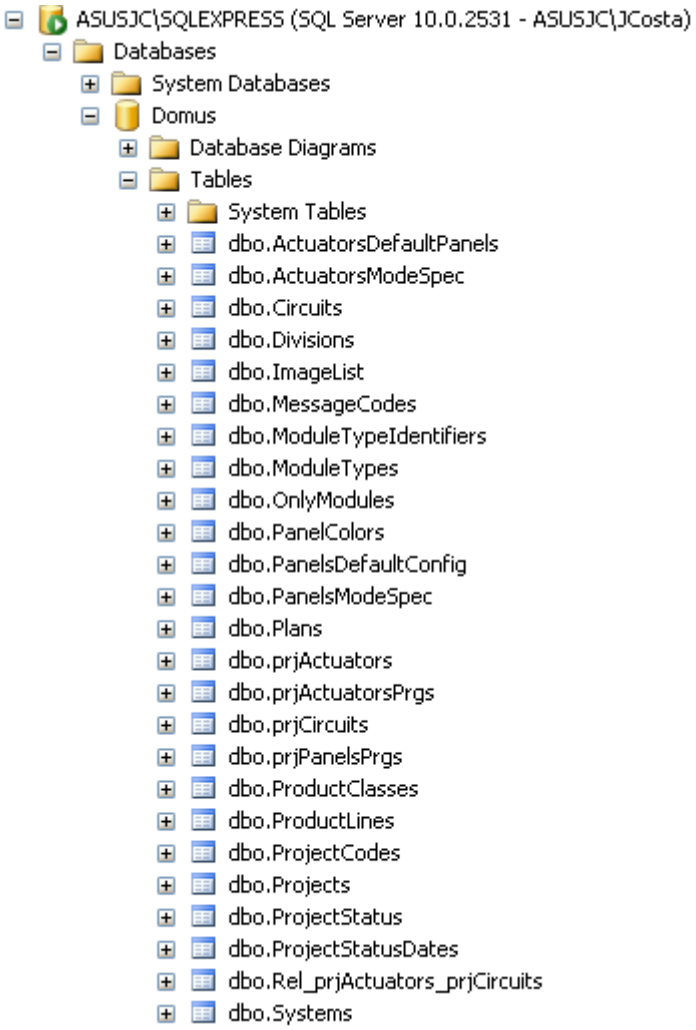

Figura 29 **Tabelas da base de dados**

Na Figura 30 lista-se o conjunto dos 44 procedimentos criados para acesso e processamento da informação armazenada na base de dados.

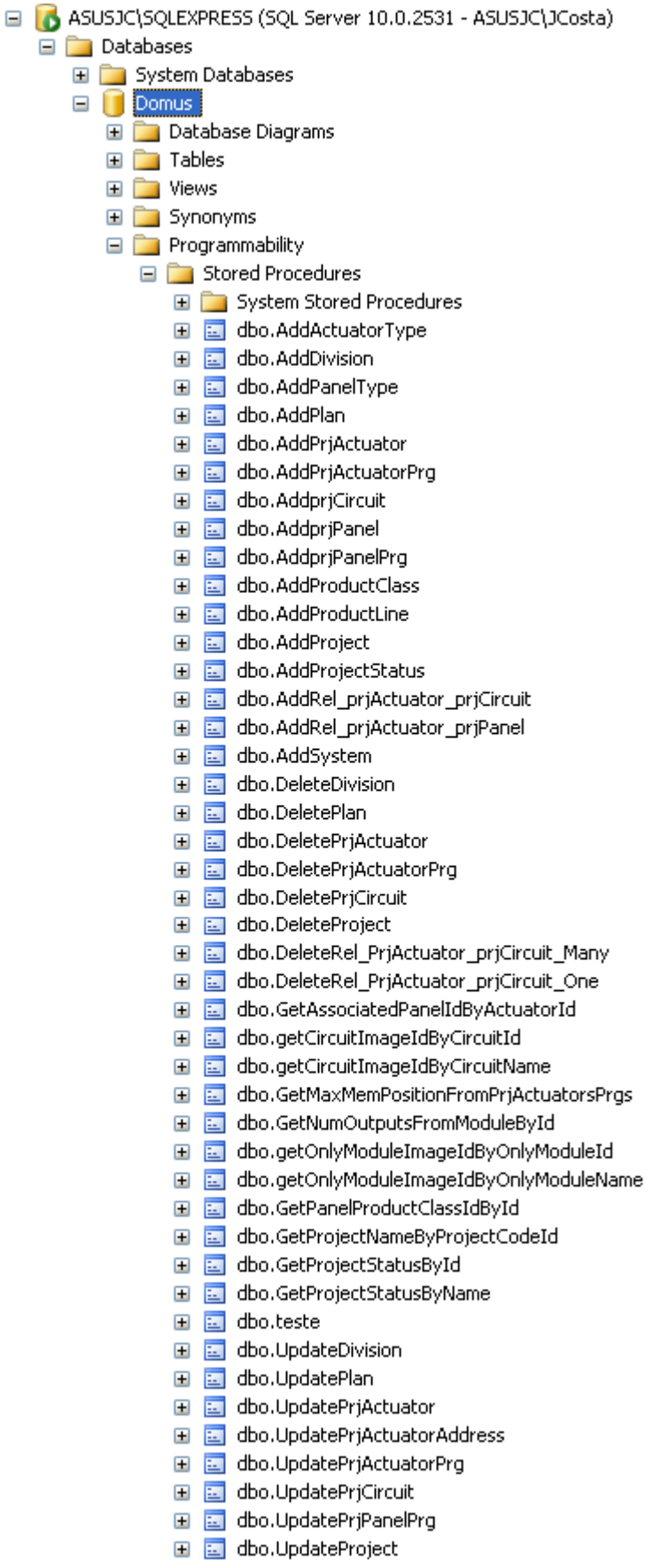

Figura 30 **Procedimentos da base de dados**

O acesso aos dados armazenados na base de dados é efetuado através da classe `SQLutils` que é partilhada pelas aplicações 'Gestão do projeto' e 'Configuração da instalação'. Deste modo, separam-se os dados da camada de apresentação da informação estratégica de negócio, permitindo uma maior abstração no desenvolvimento das camadas mais altas.

Os métodos desta classe agem sobre os dados através de procedimentos definidos na base de dados, realizando, em primeiro lugar, a ligação à base de dados, seguido de uma operação de leitura ou escrita e, por último, o encerramento da ligação.

No Excerto de Código 4 ilustra-se o procedimento que permite modificar a informação relativa a um projeto.

```
USE [Domus]
GO
/****** Object:  StoredProcedure [dbo].[UpdateProject]    Script
Date: 06/03/2012 22:58:42 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateProject]
   @myOldProjectName varchar(100),
   @myProjectName varchar(100),
   @myEntity varchar(100),
   @myAddress varchar(100),
   @myContact varchar(100),
   @myNotes varchar(200),
   @myStatus varchar(50)
AS
   IF (@myOldProjectName <> @myProjectName)
    IF EXISTS (SELECT [ProjectName] FROM dbo.Projects WHERE
[ProjectName] = @myProjectName)
           RETURN 1       -- já existe
   UPDATE dbo.Projects
   SET
           [ProjectName] = @myProjectName,
           [EntityName] = @myEntity,
           [EntityAddress] = @myAddress,
           [EntityContact] = @myContact,
           [Notes] = @myNotes,
           [Id_Status] = (SELECT Id FROM dbo.ProjectStatus WHERE
StatusName = @myStatus)
   WHERE [Id] = (SELECT [Id] FROM dbo.Projects WHERE [ProjectName]
= @myOldProjectName)

           -- Datas do estado
   INSERT INTO dbo.ProjectStatusDates
   ([Id_Project], [Id_Status])
   VALUES ((SELECT [Id] FROM dbo.Projects WHERE [ProjectName] =
@myOldProjectName),
                   (SELECT Id FROM dbo.ProjectStatus WHERE
[StatusName] = @myStatus))

   RETURN 0
```

Excerto de Código 4 **Procedimento '`UpdateProject`'**

Na Figura 31 apresenta-se o conjunto de tabelas envolvido na definição de um circuito de uma determinada divisão de uma instalação. Este conjunto é composto pelas seguintes

tabelas: `Circuits`, `prjCircuits`, `Divisions`, `Plans`, `Projects`, `ProjectCodes` e `ProjectStatus`.

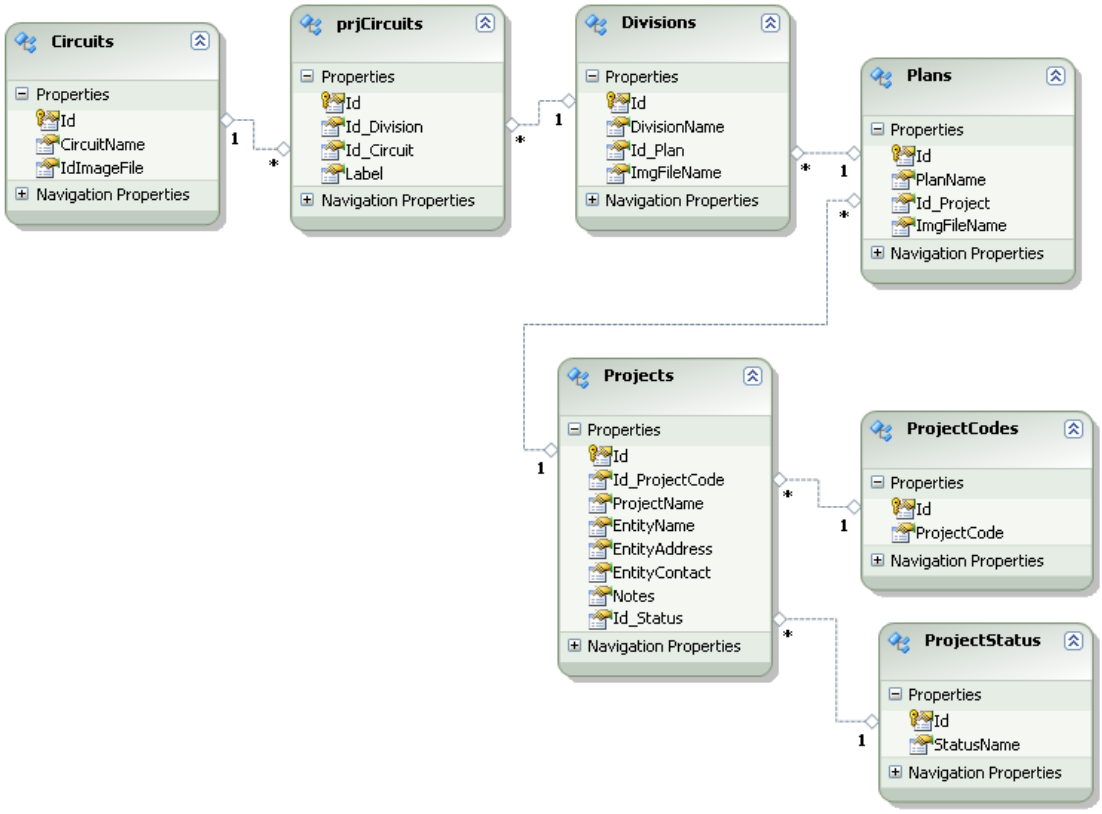

Figura 31 **Conjunto de tabelas envolvido na definição dos circuitos de um projeto**

4.5. GESTÃO DA INSTALAÇÃO

Este módulo destina-se ao controlo automático da instalação, ou seja, a comandar via *software* as saídas dos atuadores. Existem circuitos cujo controlo pode ser do tipo 'tudo ou nada' ou regulado quer em tensão, quer por tempo. Neste último tipo encontram-se, respetivamente, a iluminação com variação de intensidade e os estores.

O módulo é constituído por uma aplicação desenvolvida em C# (`LSSimplyWay`) e por um sistema do tipo embarcado (*hardware* de controlo – `SimplyWay`) já descrito em detalhe na subsecção 3.4.1.

A aplicação foi desenvolvida tendo em atenção as restrições do *hardware* utilizado assim como a versão minimalista do .Net Framework (.Net MicroFramework) disponível no sistema embarcado. Esta aplicação, que é suportada por um conjunto de bibliotecas

fornecidas pela GHI que permitem o acesso ao *hardware* deste fabricante, é constituída pelos seguintes módulos:

- `Program.cs` – classe de arranque do programa;
- `microFTC232.cs` – classe que implementa o protocolo de comunicação dos módulos de domótica;
- `microSD.cs` – classe responsável pelos acessos à informação do cartão SD;
- `FezUtils.cs` – classe com acessos específicos ao *hardware*;
- `TcpServer.cs` – classe com o protocolo Micros Fidelio;
- `WebServer.cs` – classe que implementa um servidor *Web*;
- `Cmds.cs` – classe que faz a ponte entre os comandos de alto nível e os do barramento de domótica;
- `Vars.cs` – classe com estruturas de dados.

A aplicação foi descarregada para o *hardware* de controlo utilizando uma ligação USB.

A interligação do módulo de controlo com o barramento de domótica é realizada usando uma porta série (RS232). Cabe à aplicação `LSSimplyWay` interpretar os comandos enviados pela interface Ethernet e traduzi-los no protocolo de comunicação dos módulos de domótica Only. No sentido inverso, traduz as respostas recebidas do barramento de domótica no protocolo adequado e envia-as através da interface Ethernet.

Este módulo permite processar instruções de controlo com origem quer em páginas *Web*, quer em clientes baseados no protocolo Micros Fidelio.

4.5.1. ARRANQUE DO SISTEMA

O sistema está pronto para processar comandos 5 s após se lhe ter aplicado a tensão de alimentação ou se ter efetuado *reset*. O sistema arranca de acordo com a seguinte sequência:

- Verifica a presença de um cartão SD válido;

- Lê a configuração do sistema;
- Lê a configuração da ligação *Ethernet*;
- Lê a configuração da instalação domótica;
- Inicia as comunicações com a porta RS232;
- Inicia o servidor TCP (se assim estiver definido no ficheiro de configuração) ou cria as páginas HTML e inicia o servidor *Web.*

Para auxiliar a deteção de erros na fase de iniciação do sistema, é utilizado um *Light Emitter Diode* (LED) para sinalizar o bom ou mau funcionamento do `SimplyWay`. Assim, dependendo do erro que venha a ocorrer, este LED pode piscar entre duas a oito vezes com intervalos de 300 ms cada 4 s. No caso de uma iniciação com sucesso apenas piscará uma vez. Esta sinalização, realizada através de uma tarefa concorrente, permanece ativa durante o funcionamento do sistema.

4.5.2. CONFIGURAÇÃO

O sistema desenvolvido é bastante versátil, podendo adaptar-se a qualquer configuração domótica pretendida. Esta funcionalidade baseia-se na leitura e escrita da informação correspondente num ficheiro XML no cartão SD pela aplicação `LSinst`. Este cartão, além da configuração da instalação, contém a informação relativa à configuração do próprio `SimplyWay` e dos parâmetros de funcionamento. A Figura 32 apresenta o conteúdo inicial do cartão SD.

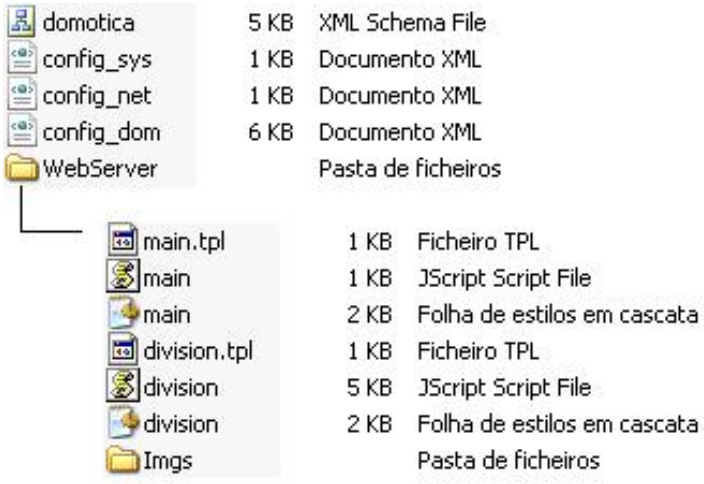

Figura 32 **Estrutura original dos ficheiros no cartão SD**

O ficheiro `config.sys` contém as definições dos intervalos de tempo necessários para ligar ou desligar um ou mais circuitos. É o caso, por exemplo, da comutação de estado da saída de um atuador que necessita de um intervalo de tempo para que a comutação de estado tenha efeito.

No Excerto de Código 5 transcreve-se o conteúdo do ficheiro `config.net`. Este ficheiro configura a ligação *Ethernet*, permitindo que o *hardware* adote o endereço IP definido ou aquele que lhe venha a ser atribuído por um servidor Dynamic Host Configuration Protocol (DHCP[16]). Nesta configuração atribui-se um endereço IP fixo. No caso das configurações que utilizam o protocolo Micros Fidelio, a anotação `<dhcp>` deve ter o valor '1' e a anotação `<clientType>` o valor 'TCP'.

```
<?xml version="1.0" encoding="utf-8"?>
<config>
  <mac>00-191-43-42-41-40</mac>
  <ip>192.168.100.100</ip>
  <port>4411</port>
  <subnet>255.255.255.0</subnet>
  <dns>192.168.100.250</dns>
  <gateway>192.168.100.254</gateway>
  <dhcp>0</dhcp>
  <clientType>WEB</clientType>
</config>
```

Excerto de Código 5 **Ficheiro `config.net`**

É possível forçar a configuração para o modo *Web* independentemente do valor do marcador `<clientType>` mantendo no nível lógico verdadeiro a entrada Di2. Esta facilidade foi implementada para facilitar o teste do sistema nas duas configurações sem recorrer à reescrita do conteúdo deste ficheiro, poupando-se tempo de desenvolvimento. Acresce ainda o facto de este ser o modo de funcionamento mais frequente.

Os ficheiros `config_dom.xml` e `domótica.xsd`, que já foram descritos na subsecção 4.3.5, representam, respetivamente, a configuração da instalação e o esquema de anotação adotado para este ficheiro XML. Durante o processo de arranque e na fase de leitura do ficheiro de configuração da instalação domótica são criadas automaticamente estruturas de dados que armazenam informação relevante acerca do projeto, dos planos, das divisões, dos circuitos e dos atuadores. Estas estruturas estão interligadas de modo relacional de modo a proporcionarem um acesso eficiente. Foram implementados ainda diversos

16 *Dynamic Host Configuration Protocol* (DHCP) é um protocolo do serviço TCP/IP que permite a uma máquina (servidor) atribuir endereços IP a outras máquinas (clientes).

métodos complementares que permitem, por exemplo, fazer a pesquisa de um endereço de um atuador e respetiva saída associada a um circuito.

Assim, sempre que é necessário processar um comando de leitura ou de acionamento de um circuito, é mais rápido fazê-lo recorrendo à memória de execução do que através da leitura do ficheiro XML. Esta vantagem é ainda acrescida do facto da leitura do ficheiro consumir mais recursos que a leitura da memória.

Estas funcionalidades estão implementadas na classe `Cmds.cs`. No Excerto de Código 6 ilustra-se a pesquisa do endereço de um atuador e da sua saída ligada a um determinado circuito.

```
private void getAddressAndOutputbyCircuitId(string circuitId, ref
byte[] address, ref byte output)
    {
      for (byte n = 0; n <= Vars.cirStruct.Length - 1; n++)
      {
        if (Vars.cirStruct[n].circuitId == circuitId)
        {
          address = Vars.cirStruct[n].address;
          output = byte.Parse(Vars.cirStruct[n].output);
          break;
        }
      }
    }
```

Excerto de Código 6 **Pesquisa na estrutura `circStruct`**

No Excerto de Código 7 mostra-se a estrutura que armazena a informação dos circuitos e que é utilizada no método `getAddressAndOutputbyCircuitId(…)` ilustrado no Excerto de Código 6.

```
    public struct circuits
    {
      public string circuitId;
      public string circuitLabel;
      public string circuitType;
      public byte[] address;
      public string output;
      public string divisionId;
    }
    public static circuits[] cirStruct;
```

Excerto de Código 7 **Estrutura com a informação dos circuitos**

4.5.3. COMUNICAÇÃO RS232

A classe `microFTC232` implementa a comunicação com o barramento de domótica. Esta classe tem como base a versão utilizada na aplicação `LSinst` (`FTC232.cs`). Como a memória disponível para programa se revelou escassa durante o desenvolvimento desta

aplicação, foram retirados os métodos irrelevantes para a comunicação com os módulos de domótica.

De modo a manter a compatibilidade e a facilidade de alteração do código destas duas classes, foram replicadas na classe `microFT232` os métodos de manipulação do objeto `ArrayList` ausentes do microFramework mas presentes na aplicação `LSinst` baseada no Framework. O objeto `ArrayList` armazena as mensagens recebidas através da porta série.

4.5.4. SERVIDOR TCP

A classe `TcpServer.cs` implementa um servidor TCP multiutilizador que recebe e interpreta pedidos dos clientes com recurso à classe `Cmds.cs` e que os envia para o barramento com recurso à classe `microFTC232.cs`. Quando a resposta a um pedido é colocada no barramento, o processo decorre no sentido inverso, sendo devolvido ao cliente o resultado do pedido que pode ser um ou mais escalares. A troca de mensagens (encapsuladas nas tramas TCP) é realizada com base no protocolo Micros Fidelio.

Quando um cliente requer uma ligação, é obrigado a executar um procedimento de início de sessão. Só depois de concluído este procedimento é possível processar pedidos e respostas entre cliente e servidor. No Anexo A encontra-se uma breve descrição do protocolo Micros Fidelio assim como um exemplo da sua utilização neste trabalho ao nível das mensagens TCP.

4.5.5. SERVIDOR *WEB*

A classe `WebServer.cs` implementa um servidor *Web* apenas com as funcionalidades necessárias para a realização dos objetivos deste trabalho. Os ficheiros necessários para a criação das páginas *Web* assim como as próprias páginas encontram-se na pasta ‘WebServer’.

Em modo *Web*, o servidor cria, numa primeira fase, os ficheiros com a extensão ‘`html`’e, por fim, cria o objeto `myWebServer` (instância da classe `WebServer`). São criadas a página principal do servidor (`main.html`) e tantas páginas *Web* quantas as divisões existentes na instalação. Deste modo, independentemente de se ter alterado a configuração da instalação (ficheiro `domótica.xml`), o sistema cria sempre, durante a fase de arranque, as páginas *Web* de acordo com a informação armazenada, tornando o sistema mais versátil.

A página principal ‘main.html’ é criada tendo como modelo o ficheiro ‘main.tpl’ que se transcreve no Excerto de Código 8. Na fase da criação da página, a aplicação copia o modelo para a nova página, substituindo um conjunto pré-definido de marcadores pelo texto correspondente. Esta operação de pesquisa e substituição decorre para os seguintes marcadores:

- [css] – refere-se ao ficheiro de estilos e é substituído por ‘main.css’;
- [js] – designa o ficheiro com código ‘JavaScript’ e é substituído por ‘main.js’;
- [prjName] – refere-se ao nome do projeto e é substituído pelo nome do projeto em questão;
- [auto] – indica o início da estrutura de planos e divisões da instalação de acordo com o ficheiro domótica.xml.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="content-type"
        content="text/html;charset=utf-8"/>
  <title>SimplyWay</title>
  <link rel="stylesheet" type="text/css" href="[css]" />
  <script type="text/javascript" src="[js]"></script>
 </head>
 <body onload="docLoad()">
  <div id="container">
   <div id="header">
    <p>[prjName]</p>
   </div>
   <div id="tree">
    <ul id="prj">
     <!--[auto]-->
    </ul>
   </div>
   <div id="content">
    <iframe id="if_contents" frameborder="0" src=""></iframe>
   </div>
   <div id="footer">
    <ul>
     <li>LiveSimply</li>
     <li>SimplyWay V1.0</li>
    </ul>
   </div>
  </div>
 </body>
</html>
```

Excerto de Código 8 **Modelo da página 'main.html'**

No Excerto de Código 9 mostra-se um exemplo da página ‘main.html’ criada no arranque do sistema, tendo como base o modelo descrito.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="content-type"
        content="text/html;charset=utf-8"/>
  <title>SimplyWay</title>
  <link rel="stylesheet" type="text/css" href="main.css" />
  <script type="text/javascript" src="main.js"></script>
 </head>
 <body onload="docLoad()">
  <div id="container">
    <div id="header">
     <p>Avenida dos estudantes</p>
    </div>
    <div id="tree">
     <ul id="prj">
      <li class="plan">R/C
       <ul class="allDivisions">
        <li id="DV36" class="division"
           onclick="changeIframeTarget('DV36.html')">Salão</li>
        <li id="DV37" class="division"
           onclick="changeIframeTarget('DV37.html')">Cozinha</li>
       </ul>
      </li>
      <li class="plan">1-Andar
       <ul class="allDivisions">
        <li id="DV38" class="division"
           onclick="changeIframeTarget('DV38.html')">Pavilhão</li>
       </ul>
      </li>
     </ul>
    </div>
    <div id="content">
     <iframe id="if_contents" frameborder="0" src=""></iframe>
    </div>
    <div id="footer">
     <ul>
      <li>LiveSimply</li>
      <li>SimplyWay V1.0</li>
     </ul>
   </div>
</div>
</body>
</html>
```

Excerto de Código 9 **Página '`main.html`'**

A criação das páginas relativas às divisões obedecem às mesmas regras descritas para a página principal, sendo neste caso utilizados os ficheiros '`division.tpl`', '`division.css`' e '`division.js`'. O Excerto de Código 10 apresenta o conteúdo da página correspondente à divisão identificada no ficheiro XML com o código 36 ('Salão' situado no plano 'R/C').

O ficheiro '`division.js`' contém o código necessário à visualização e atuação das saídas dos atuadores. O envio dos comandos de domótica para o servidor e a receção das respostas está a cargo do código AJAX definido neste ficheiro.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="content-type"
        content="text/html;charset=utf-8"/>
  <meta name="divid" content="36" />
  <title>Salão</title>
  <link rel="stylesheet" type="text/css" href="division.css" />
  <script type="text/javascript" src="division.js"></script>
 </head>
 <body onload="docLoad()">
  <h2>R/C ... Salão</h2>
  <h1>
   <img id="refresh" src="Imgs/refresh_enabled.png" alt=""
        onclick="doRefresh()" />
  </h1>
  <input id="CI128" class="cirLabel" type="text" name="P"
         value="Cir_Estore" readonly="readonly" />
  <input id="VR128" class="vRead" type="text" name="" value=""
         readonly="readonly"/>%
  <input class="minus" type="button" value="-"
         onclick="minus('VW128')" />
  <input id="VW128" class="vWrite" type="text" name="" value="0"
         readonly="readonly"/>%
  <input class="plus" type="button" value="+"
         onclick="plus('VW128')" />
  <img class="go" src="Imgs/go.png" alt="" title=""
         onclick="go('CI128', 'VW128')" /><br />
  <input id="CI129" class="cirLabel" type="text" name="D"
         value="Cir_IlumTeto" readonly="readonly" />
  <img id="IM129" class="toggle" src="Imgs/off.png" alt =""
       title="" onclick="toggleCir('CI129')" /><br />
  <input id="CI130" class="cirLabel" type="text" name="D"
         value="Cir_IlumLateral" readonly="readonly" />
  <img id="IM130" class="toggle" src="Imgs/off.png" alt =""
       title="" onclick="toggleCir('CI130')" /><br />
  <input id="CI131" class="cirLabel" type="text" name="P"
         value="Cir_SalaEstar" readonly="readonly" />
  <input id="VR131" class="vRead" type="text" name="" value=""
         readonly="readonly"/>%
  <input class="minus" type="button" value="-"
         onclick="minus('VW131')" />
  <input id="VW131" class="vWrite" type="text" name="" value="0"
         readonly="readonly"/>%
  <input class="plus" type="button" value="+"
         onclick="plus('VW131')" />
  <img class="go" src="Imgs/go.png" alt="" title=""
       onclick="go('CI131', 'VW131')" /><br />
 </body>
</html>
```

Excerto de Código 10 **Página 'dv36.html'**

Na Figura 33 mostra-se um exemplo de uma página *Web* com dois planos (R/C e 1.º Andar) e as respetivas divisões. Na divisão 'Salão' encontram-se dois circuitos de iluminação sem variação de intensidade e dois circuitos de controlo regulável: o circuito de estore e o circuito de iluminação da sala de estar.

Os circuitos de controlo regulável podem ser controlados através dos botões + e – para subir ou descer (aumentar ou diminuir) em passos de 20 %. Os circuitos do tipo tudo ou nada, ligam-se e desligam-se selecionando-se a estrela.

Na classe `WebServer.cs` apenas foram implementados os métodos de processamento dos pedidos GET e POST. O método GET é responsável pelo envio de páginas HTML para o navegador do cliente. O método POST recebe do navegador cliente o comando de domótica a processar e, recorrendo às classes `Cmds.cs` e `microFTF232.cs` para o envio do comando e receção da resposta, obtém o resultado que envia ao navegador cliente. Os comandos e as respostas com origem no método POST, têm a mesma sintaxe dos comandos utilizados no protocolo Micros Fidelio (configuração TCP).

Do lado do cliente, o código JavaScript e AJAX encarregam-se de atualizar a página em conformidade.

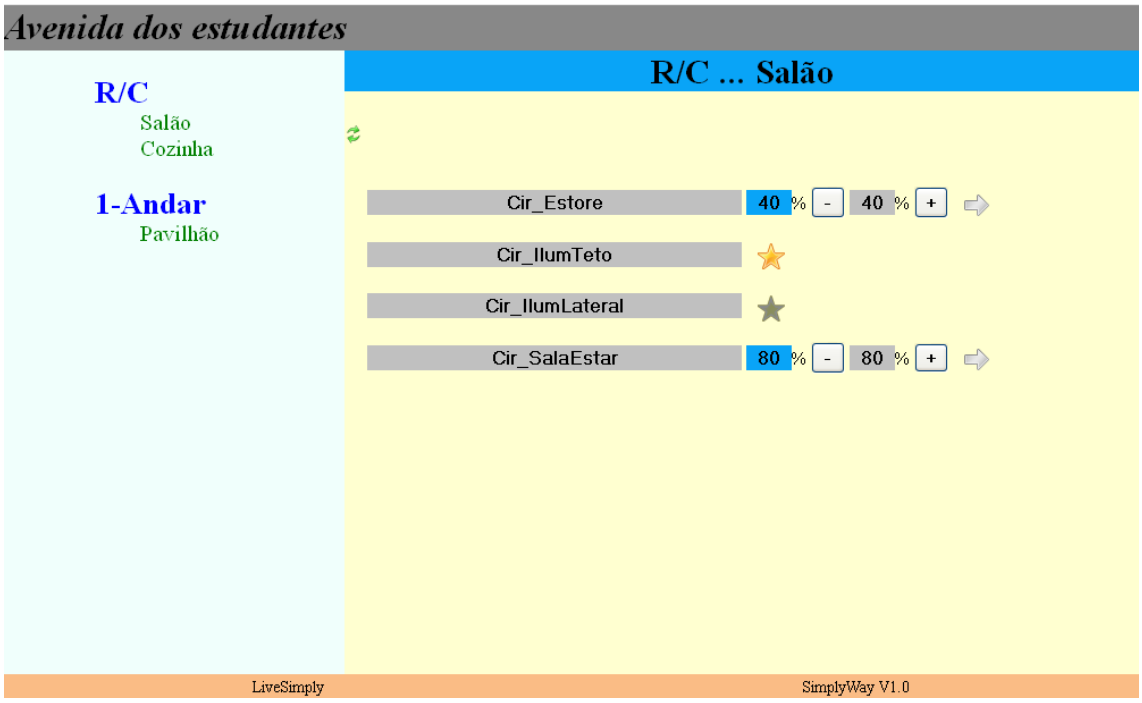

Figura 33 **Página *Web***

Todas as páginas *Web* criadas pelo servidor foram validadas pelo 'W3C Markup Validation Service'[17].

4.6. *HARDWARE*

Como já referido na subsecção 3.4 o sistema de controlo é baseado numa placa microprocessada da GHI designada Fez Domino, por uma placa da CuteDigi, que realiza a

17 Disponível em http://validator.w3.org/ [Acedido em Junho de 2012].

interface com o barramento de domótica, e por uma placa Fez Connect da GHI baseada no controlador W5100 e que permite a interface com o barramento *Ethernet*.

Os pinos Di0 e Di1 são, respetivamente, os pinos de receção e transmissão utilizados na comunicação RS232. A placa da CuteDigi realiza o condicionamento de sinal, já que a placa Fez Domino apenas disponibiliza sinais ente 0 V e 5 V.

A placa Fez Connect, além de outras funcionalidades, é responsável pelas comunicações *Ethernet*. A comunicação entre a placa Fez Domino e esta placa realiza-se através da SPI1 e utiliza os seguintes sinais da placa Fez Domino:

- Di7 – *Reset* do W5100;
- Di10 – Sinal *Chip Select* (para a SPI1);
- Di11 – Sinal *Master Output/Slave Input* (MOSI);
- Di12 – Sinal *Master Input/Slave Output* (MISO);
- Di13 – Sinal de relógio para a SPI1.

Na Figura 34 mostra-se o diagrama das ligações de apoio à comunicação descrita.

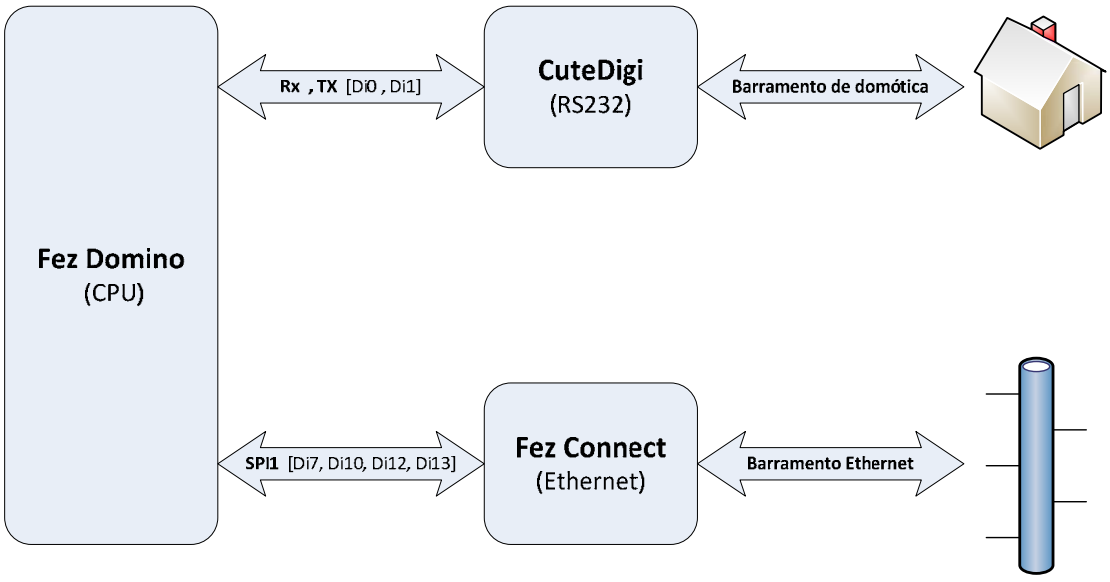

Figura 34 **Diagrama de ligação**

Na aplicação `LSSimplyWay` e nas classes `TcpServer.cs` e `WebServer.cs`, dependendo da configuração selecionada, configura-se a SPI1 do Fez Domino de acordo com o Excerto de Código 11.

```
WIZnet_W5100.Enable(SPI.SPI_module.SPI1, (Cpu.Pin)FEZ_Pin.Digital.D
i10, (Cpu.Pin)FEZ_Pin.Digital.Di7, true); // WIZnet Fez connect
```

Excerto de Código 11 **Programar SPI1 do Fez Domino**

4.7. PROBLEMAS

Vários problemas ocorreram durante o desenvolvimento do projeto. Um dos principais desafios foi a implementação do sistema de controlo no *hardware* Fez Domino. Por um lado, a gestão da memória volátil de dados e, por outro lado, a utilização dos espaços de nomes, cuja implementação não será a mais correta para módulos deste tipo, obrigaram a abordagens diferentes das que estavam originalmente previstas.

4.7.1. GESTÃO DE MEMÓRIA

Durante a fase de desenvolvimento e testes da aplicação de controlo foram encontradas com alguma frequência situações de perda de dados. Chegou-se à conclusão que as funções de gestão de memória do Fez Domino, presentes nas bibliotecas do .Net Micro Framework, não funcionavam da forma prevista. Assim, passou a proceder-se à limpeza de memória de um modo explícito, ou seja, introduzindo chamadas às funções de gestão de memória nas situações mais críticas. De notar que este problema acontece frequentemente devido à escassez da memória de dados da placa Fez Domino e, obviamente, se faz notar tanto mais quanto menor for o espaço livre para a gestão da mesma. Estes problemas foram mais evidentes nas operações de leitura ou escrita de dados em ficheiros ou através de *sockets* TCP.

As operações de leitura e escrita de dados em ficheiros são implementadas recorrendo a estruturas de dados do tipo *string*. Assim, além das operações de gestão de memória, foi necessário reduzir o tamanho das *strings* e aumentar o número de iterações para ler ou escrever em ficheiros.

Uma situação idêntica ocorreu relativamente às operações de leitura e escrita em *sockets* TCP. Neste caso, estas operações são implementadas à custa de vetores de elementos do tipo byte, tendo-se recorrido a uma solução idêntica à anterior para resolver este problema.

4.7.2. PROTOCOLO HTTP

Durante a elaboração dos métodos da classe `WebServer.cs` verificou-se que o *hardware* de controlo, configurado como servidor, fechava a conexão após uma operação GET ou POST apesar do cliente especificar a adoção da versão 1.1 do protocolo HTTP e a manutenção da ligação (*KeepAlive*). Depois de alguns testes, verificou-se que a biblioteca disponibilizada pela GHI contemplava apenas a versão 1.0 do protocolo HTTP.

Esta questão trouxe um problema adicional, já que o *hardware* da GHI apenas permite a criação em simultâneo de quatro *sockets*. Um destes *sockets* está reservado internamente pelo *hardware*, outro encontra-se reservado para o serviço DHCP, o terceiro corresponde ao *socket* de escuta e o último é o único disponível para atendimento de um cliente. Assim, se o servidor se desligar após o atendimento de cada pedido, provoca uma sobrecarga do processador, resultando em erros do lado do cliente durante o carregamento das páginas HTML de resposta.

A solução passou por implementar o protocolo HTTP 1.1 de raiz em TCP, eliminando assim os problemas identificados. Neste novo cenário foram realizados com sucesso testes envolvendo três ligações.

4.8. TESTES E RESULTADOS

Os testes realizados visaram essencialmente a verificação da correta implementação do protocolo Only. Realizou-se uma pequena montagem com três módulos e uma pequena aplicação em C# de modo a testar as funcionalidades da classe `FTC232.cs (módulo LSInst)`. Alguns dos resultados obtidos não estavam de acordo com a documentação fornecida, tendo o fabricante sido informado.

O objeto instância desta classe corre num *Thread*[18] diferente da aplicação principal, utilizando uma fila para expor as mensagens lidas do barramento de domótica. Depois de múltiplos testes, concluiu-se que uma fila dimensionada para 50 mensagens e uma periodicidade de consulta/leitura da fila de 200 ms pelo *Thread* principal era suficiente.

18 Subprocesso ou atividade que pode executar em paralelo; as *Threads* de um mesmo processo têm a particularidade de partilharem a mesma memória.

Mesmo colocando no barramento mensagens a um ritmo para além do razoável, o comportamento do sistema mostrou-se sempre estável.

Para testar as funcionalidades das classes `TcpServer.cs` e `Cmds.cs` (módulo `LSSimplyWay`), elaborou-se um pequeno programa em C# onde, fazendo uso da montagem de módulos já referida, se testaram as funções destas classes.

O teste final realizou-se recorrendo a um painel de demonstração que, para além dos módulos de domótica, também continha módulos de segurança e de climatização, procurando-se testar o sistema em condições tão próximas das condições reais quanto possível. Estes testes contemplaram as duas configurações possíveis (*Web* e Micros Fidelio) e realizaram-se percorrendo todos os passos, ou seja, começando pelo módulo de gestão do projeto, passando pelo módulo de configuração da instalação e, por último, instalando o cartão SD no *hardware* de controlo e testando as várias funcionalidades definidas na configuração. Foram medidos os tempos de resposta do sistema de controlo para uma configuração composta por dois planos e duas divisões por plano num total de oito circuitos. Na versão mais complexa (versão *Web*), o sistema demora a arrancar 5 s, sendo necessários cerca de 2 s para criar as páginas HTML.

Depois de analisados os resultados e o desempenho dos vários módulos do sistema, verificou-se tratar-se de uma solução viável e totalmente funcional.

4.9. CONCLUSÃO

Neste capítulo descreveu-se a arquitetura do sistema, detalhando-se as funcionalidades de cada um dos módulos de *software*, *hardware* e da base de dados. Apresentou-se o funcionamento dos módulos de Gestão de Projeto, Configuração da Instalação, Gestão da Instalação e os excertos de código mais relevantes. Reportaram-se ainda os principais problemas e as soluções encontradas. Por último, descreveram-se os testes realizados e comentaram-se os resultados obtidos.

5. CONCLUSÕES

Neste capítulo apresenta-se o balanço do projeto realizado no âmbito da unidade curricular de Tese/Dissertação assim como algumas sugestões de desenvolvimento e melhoramento futuros.

5.1. BALANÇO

O primeiro objetivo proposto – a criação de um sistema que permitisse aos técnicos da Live Simply projetar, configurar e gerir uma instalação domótica – foi conseguido em duas etapas: (*i*) o desenvolvimento de uma ferramenta de *software* (`LSprj`) destinada à criação do projeto e definição da tipologia da instalação domótica; e (*ii*) a realização de uma ferramenta de *software* (`LSinst`) que permite a configuração e gestão da instalação domótica, sendo a configuração armazenada num cartão de memória do tipo SD.

Quanto ao segundo objetivo – desenvolvimento de uma ferramenta que, utilizando técnicas *Web*, permitisse ao utilizador da instalação domótica interagir com a mesma – foi alcançado através da realização de um sistema composto por uma aplicação de *software* (`LSSimplyWay`) e por um conjunto de *hardware* (`SimplyWay`) que pode ser alojado no quadro elétrico da instalação do cliente e que possibilita a interação com a instalação domótica via um *front-end Web*.

Finalmente, relativamente ao pedido posterior da Live Simply de suporte do protocolo Micros Fidelio, o sistema de controlo foi implementado com a capacidade de atendimento de clientes Micros Fidelio.

5.2. MELHORAMENTOS FUTUROS

Apesar do sistema de controlo corresponder integralmente aos requisitos propostos, foram identificados alguns aspetos que podem ser melhorados.

Uma análise da tendência de mercado da área da domótica, que se encontra alinhada com os objetivos futuros da Live Simply, permite concluir que o sistema necessita aumentar as suas capacidades. Assim, e tendo em conta que o custo das placas eletrónicas do sistema atual ronda 80 € e a solução final não deve exceder os 90 €, verifica-se que é possível encontrar no mercado soluções alternativas de *hardware* que apresentam maior capacidade de processamento. É o caso, por exemplo, das soluções de *hardware* não proprietário do tipo computador de baixo consumo alojado em placa única *BeagleBone* e *Beagleboard*[19]. Estes sistemas, que comportam processadores a 700 MHz e a 1 GHz, permitem executar o sistema operativo Linux e instalar a solução apresentada neste trabalho sem grandes alterações, recorrendo, por exemplo, à plataforma Mono[20]. Neste sentido, e apenas para efeitos de teste, foi compilado no Mono uma versão ligeiramente alterada da versão instalada no sistema de controlo desenvolvido. Esta versão foi, de seguida, carregada para uma placa Beagleboard XM a correr a implementação de Linux da Ångström [49]. Os resultados positivos provaram que se trata de uma abordagem válida que permite corresponder aos desafios futuros de escalabilidade e maior capacidade de processamento.

O sistema desenvolvido dedicado ao projeto, configuração e gestão de instalações domóticas centrou-se no domínio da automação, mas pode ser alargado aos domínios do áudio, climatização e segurança. A inclusão destas áreas permitiriam transformar este sistema numa ferramenta de trabalho mais abrangente e incontornável.

Do ponto de vista do dono da instalação, o desenvolvimento de uma aplicação cliente para dispositivos móveis perfila-se como uma mais-valia interessante. Esta aplicação permitiria ao utilizador controlar os equipamentos da sua instalação através do seu dispositivo móvel.

19 Informação disponível em http://beagleboard.org/. [Acedido em Junho de 2012].
20 Informação disponível em http://www.mono-project.com. [Acedido em Junho de 2012].

Referências Documentais

[1] ADIT, *Will Domotics Finally Make a Breakthrough?*, eTech France, N. 141, 2004. Disponível em http://www.bulletins-electroniques.com/actualites/38245.htm [Acedido em Janeiro de 2012].

[2] National Institute of Standards and Technology (NIST), *The NIST Guide for the International system of Units*, NIST Special Publication 811, 2008. Disponível em http://physics.nist.gov/cuu/pdf/sp811.pdf [Acedido em Fevereiro de 2012].

[3] Renato Nunes e José Delgado, *An Architecture for a Home Automation System*, IST/INESC Lisboa, IEEE International Conference in Electronics, Circuits and Systems, Vol. 1, 259-262 1998. DOI: 10.1109/ICECS.1998.813316. Disponível em http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=813316 [Acedido em Fevereiro de 2012].

[4] César Adriano Alievi, *Automação Residencial com Utilização de Controlador Lógico Programável*, 2008. Disponível em http://tconline.feevale.br/tc/files/0001_1697.pdf [Acedido em Fevereiro de 2012].

[5] Edward B. Driscoll, Jr., *The history of X10*, Disponível em http://home.planet.nl/~lhendrix/x10_history.htm [Acedido em Fevereiro de 2012].

[6] X10, *X10 Technology Transmission Theory*. Disponível em http://x10.com/technology1.htm [Acedido em Fevereiro de 2012].

[7] Kenneth P. Wacks, *Introduction to the CEBUS Communications Protocol*, Home Technology eMagazine, 1997. Disponível em http://hometoys.com/htinews/aug97/articles/kwacks/kwacks.htm [Acedido em Fevereiro de 2012].

[8] Echelon. Disponível em http://echelon.com [Acedido em Fevereiro de 2012].

[9] Jaec, *European Home Systems*, 2008. Disponível em http://www.jaec.info/Home%20Automation/Protocols-buses-house/Ehs-Protocol/ehs-protocol.php [Acedido em Fevereiro de 2012].

[10] Yoshiyuki Honda, Masahiro Inoue, Rieko Iwatsubo e Kazunori Sakanobe, *Protocol Analyzer For Home Bus System* (HBS), IEEE Transaction on Consumer Electronics, Vol. 36, N. 3, 586-592, August 1990. DOI: 10.1109/30.103178. Disponível em http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=103178 [Acedido em Fevereiro de 2012].

[11] Fukuoka Institute of Technology, *Home Network* (HBS, IEEE1394). Disponível em http://www.fit.ac.jp/~hamabe/intro/HBS.html [Acedido em Fevereiro de 2012].

[12] Pedro Miguel de Miranda Fernandes, *Aplicações Domóticas para Cidadãos com Paralisia Cerebral*, Tecnologia Domótica, 2001. Disponível em

http://portal.ua.pt/bibliotecad/default1.asp?OP2=0&Serie=0&Obra=28&H1=2&H2=1 [Acedido em Fevereiro de 2012].

[13] KNX Association, *Configuration modes*, 2012. Disponível em www.knx.org [Acedido em Fevereiro de 2012].

[14] Tapko Tecnologies Gmbh, *How to develop an A-mode compatible device*, 2004. Disponível em www.tapko.de. [Acedido em Fevereiro de 2012].

[15] Renato Jorge Caldeira Nunes, *Análise Comparativa de Tecnologias para Domótica*, II Jornadas de Engenharia de Automação, Controlo e Instrumentação, ESTS, Setúbal, 2002. Disponível em http://domobus.net/docs/02-JEACI02.pdf [Acedido em Fevereiro de 2012].

[16] Domingos Salvador dos Santos, *KNX – Topologia*, ISEP, Porto, 2007. Disponível em http://ave.dee.isep.ipp.pt/~dss/Disciplinas/DOMOT_PT.htm [Acedido em Fevereiro de 2012].

[17] Lucínio Preza de Araújo, *Domótica – Protocolo de comunicação X10*. Disponível em http://www.prof2000.pt/users/lpa/x10.ppt [Acedido em Fevereiro de 2012].

[18] Hugo Malafaya, Luís Tomás, João Paulo Sousa, *Sensorização sem fios sobre ZigBee e IEEE 802.15.4*, Terceiras Jornadas de Engenharia de Electrónica e Telecomunicações e de Computadores, ISEL, Lisboa, 2005. Disponível em http://www.deetc.isel.ipl.pt/jetc05/JETC05/Artigos/Electronica/Poster%20E/136.pdf [Acedido em Fevereiro de 2012].

[19] Jennic, *ZigBee Topologies*, 2007. Disponível em http://www.jennic.com/elearning/zigbee/files/html/module2/module2-1.htm [Acedido em Fevereiro de 2012].

[20] Sinem Coleri Ergen, *ZigBee/IEEE 802.15.4 Summary*, 2004. Disponível em http://pages.cs.wisc.edu/~suman/courses/838/papers/zigbee.pdf [Acedido em Fevereiro de 2012].

[21] Diana Sobreiro de Costa Palma, *FEUP KNX – Domótica KNX/EIB de Baixo Custo*, Tese de Mestrado, 2008. Disponível em http://paginas.fe.up.pt/~ee05241/feupknx/Tese_Diana_Palma.pdf [Acedido em Fevereiro de 2012].

[22] Renato Nunes, *Domótica: Presente e Futuro*, 2.º Encontro de Estudantes de Informática, Évora, 2006. Disponível em http://enei.net/enei2006/documentos/apresentacoes/Dia2_Prof_Renato%20Nunes_Domotica.pdf [Acedido em Fevereiro de 2012].

[23] Valter Filipe Silva e João Alberto Fonseca, *Redes para automação de edifícios*, Workshop de Domótica, Aveiro, 2006. Disponível em http://www.aveirodomus.pt/workshop/4%20Domotica/2%20Valter%20Silva.pdf [Acedido em Fevereiro de 2012].

[24] Mono. Disponível em http://www.mono-project.com/Main_Page [Acedido em Março de 2012].

[25] dotnetpowered.com, *Language List*. Disponível em http://www.dotnetpowered.com/languages.aspx [Acedido em Março de 2012].

[26] O´Reilley, Windows devcenter, *What Is .NET?*, 2009 Disponível em http://ondotnet.com/pub/a/dotnet/2005/09/06/what-is-dotnet.html#heading2 [Acedido em Março de 2012].

[27] Jesse Liberty, *C++ -> C#: What You Need to Know to Move from C++ to C#*, MSDN Magazine, July 2001. Disponível em http://msdn.microsoft.com/en-us/magazine/cc301520.aspx [Acedido em Março de 2012].

[28] Microsoft .NET Micro Framework. Disponível em http://www.netmf.com/ [Acedido em Março de 2012].

[29] Microsoft .NET Micro Framework 4.1 Porting Kit. Disponível em http://www.microsoft.com/download/en/details.aspx?id=5868 [Acedido em Março de 2012].

[30] Microsoft .NET Framework Developer Center. Disponível em http://msdn.microsoft.com/en-us/netframework/ff962539 [Acedido em Março de 2012].

[31] W3C, *Extensible Markup Language* (XML) 1.0 (Fifth Edition), 2008. Disponível em http://www.w3.org/TR/REC-xml/ [Acedido em Março de 2012].

[32] Norman Walsh, *A Technical Introduction to XML*, O´Reily XML.com., 1998. Disponível em http://www.xml.com/pub/a/98/10/guide0.html [Acedido em Março de 2012].

[33] W3C, *XML Schema*. Disponível em http://www.w3.org/standards/xml/schema [Acedido em Março de 2012].

[34] TechTarget, *XSD* (*XML Schema Definition*). Disponível em http://searchsoa.techtarget.com/definition/XSD [Acedido em Março de 2012].

[35] W3C, *What is XHTML?*, 2002 Disponível em, http://www.w3.org/TR/xhtml1/introduction.html [Acedido em Março de 2012].

[36] TechTarget, *XHTML (Extensible Hypertext Markup Language)*, 2006. Disponível em http://searchsoa.techtarget.com/definition/XHTML [Acedido em Março de 2012].

[37] W3C, *Cascade Style Sheets*. Disponível em http://www.w3.org/Style/CSS/ [Acedido em Março de 2012].

[38] w3schools, *CSS Tutorial*. Disponível em http://www.w3schools.com/css/ [Acedido em Março de 2012].

[39] Danny Goodman, Michael Morrison, Paul Novitski, Tia Gustaff Rayl, *JavaScript Bible*, Seventh Edition, Wiley Publishing, Inc., ISBN: 978-0-470-52691-0, 2010 [Acedido em Março de 2011].

[40] w3schools, *JavaScript Tutorial*. Disponível em http://www.w3schools.com/js/ [Acedido em Março de 2012].

[41] W3C, *XMLHttpRequest Level 2*, 2012. Disponível em http://www.w3.org/TR/XMLHttpRequest/ [Acedido em Março 2012].

[42] Microsoft SQL Server 2008 Management Studio Express. Disponível em http://www.microsoft.com/downloads/details.aspx?FamilyID=08e52ac2-1d62-45f6-9a4a-4b76a8564a2b&displaylang=pt-br [Acedido em Março de 2012].

[43] Manual USBizi. Disponível em http://www.ghielectronics.com/downloads/USBizi/USBizi_Broch_Pinout.pdf [Acedido em Março de 2012].

[44] Net Framework and Languages. Disponível em http://www.startvbdotnet.com/dotnet/languages.aspx [Acedido em Junho de 2012]

[45] Fez Domino. Disponível em http://www.ghielectronics.com/downloads/FEZ/Domino/Broch_FEZ_Domino.pdf [Acedido em Junho de 2012].

[46] CuteDigi, placa RS232. Disponível em http://www.cutedigi.com/arduino-shields/rs232-shield-for-arduino.html [Acedido em Junho de 2012].

[47] Fez Connect, placa de rede. Disponível em http://www.ghielectronics.com/catalog/product/261 [Acedido em Junho de 2012].

[48] Fez Connect, funcionalidades. Disponível em http://www.ghielectronics.com/downloads/FEZ/Shield/Broch_FEZ_Connect.pdf [Acedido em Junho de 2012].

[49] The Ångström Distribution. Disponível em http://www.angstrom-distribution.org [Acedido em Junho de 2012].

[50] A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. Disponível em http://eee.guc.edu.eg/Announcements/Comparaitive_Wireless_Standards.pdf [Acedido em Junho de 2012].

Anexo A. UMA IMPLEMENTAÇÃO BASEADA NO PROTOCOLO MICROS FIDELIO

A Micros Systems, Inc é uma empresa norte americana com sede em Columbia, Maryland, que fabrica e vende computadores, *software* e serviços para as áreas da restauração, hotelaria, hospitalar e retalho, com especial enfoque em soluções de ponto de venda e gestão hoteleira. A Micros Fidelio, que é a divisão da Micros orientada para o negócio da gestão hoteleira, criou a *Fidelio Interface Application Specification* (FIAS)[21]. Esta especificação tornou-se um padrão no domínio, permitindo a comunicação entre diferentes sistemas como, por exemplo, o sistema de automação, de ar condicionado ou de gestão de energia com o sistema central de gestão de um hotel.

Neste trabalho implementa-se a variante do protocolo FIAS utilizada pela Live Simply e por alguns dos seus parceiros de negócio. O protocolo FIAS, que foi desenhado originalmente para utilizar o protocolo RS232, pode também ser encapsulado em tramas TCP. Esta foi a abordagem adotada neste trabalho.

As mensagens são trocadas entre os vários intervenientes de acordo com o seguinte padrão: STX + Conteúdo da mensagem + ETX. Os caracteres *Start of TeXt* (STX) e *End of TeXt* (ETX) são, respetivamente, o carácter de início e de fim de mensagem.

Neste trabalho, quando o `SimplyWay` está configurado como servidor (modo TCP), o sistema cliente tem de, em primeiro lugar, iniciar a sessão de acordo com o protocolo Micros Fidelio – ver a Tabela 5.

Tabela 5 FIAS – Início de sessão

Sequência	TX / RX	Mensagem
1	RX	<STX>LS|DA110601|TI170407|<ETX>
2	TX	<STX>LS|DA110601|TI170407|<ETX>
3	RX	<STX>LD|PR2011022|V#1.01|IF*interfaceID*|<ETX>
4	RX	<STX>LA| DA110601|TI170434|<ETX>
5	TX	<STX>LA| DA110601|TI170434|<ETX>

21 Para mais informação consultar http://www.micros.com/ e http://www.micros-fidelio.pt/. [Acedidos em Maio de 2012].

As siglas têm o seguinte significado:

- RX – Mensagem recebida pelo `SimplyWay`;
- TX – Mensagem enviada pelo `SimplyWay`;
- LS – *Link Start*;
- LD – *Link Description*;
- LA – *Link Alive*;
- DA – *Date*;
- TI – *Time*;
- PR e V# não fazem parte do protocolo FIAS e foram definidas pela Live Simply.

O carácter '|' é utilizado para separar os vários campos da mensagem.

O `SimplyWay` utiliza os comandos DA e TI para acertar o relógio interno. Depois do início de sessão ter decorrido com sucesso, o sistema cliente pode enviar os comandos.

A1. COMANDOS

Os comandos seguem a estrutura ilustrada na Tabela 5. Para simplificar a apresentação dos comandos criados para controlo dos atuadores suprimiram-se os caracteres STX e ETX.

Todos os comandos enviados pelo sistema cliente apresentam à cabeça a sigla DC – *Domotic Control*. A adoção desta sigla permite distinguir estes comandos dos restantes comandos do protocolo Micros Fidelio que possam eventualmente estar presentes no barramento. Segue-se uma sigla, composta por dois caracteres, que identifica um dos quatro sistemas domóticos suportados:

- AU – automação
- AD – áudio
- CL – climatização

– SG – segurança

Na Tabela 6 ilustram-se os comandos de automação.

Tabela 6 Lista completa de comandos de automação

Comando	TX / RX	Descrição
DC|AU|CI**cc1**|RD	RX	Lê o valor da saída associada ao circuito **cc1**
DC|AU|CI**cc1**|RD|ST**s**|VL**v**	TX	Resposta com estado e valor
DC|AU|CI**cc1**|WR**w**	TX	Escreve na saída associada ao circuito **cc1**, o valor **w**
DC|AU|CI**cc1**|WR**w**|ST**s**	RX	Resposta com estado
DC|AU|DV**d**	TX	Lê o valor das saídas associadas aos circuitos da divisão **d**
DC|AU|DV**d**|ST**s**|CI**cc1**|VL**v** … |CI**ccn**|VL**v**	RX	Resposta com estado e valores
DC|AU|EX**x**|DV**d**	TX	Comando estendido **x** para os circuitos da divisão **d**
DC|AU|EX**x**|DV**d**|ST**s**	RX	Resposta com estado

As siglas têm o seguinte significado:

– RX – Mensagem recebida pelo `SimplyWay`;

– TX – Mensagem enviada pelo `SimplyWay`;

– DC – Comando de domótica;

– AU – Comando do sistema de automação;

– CI**cc1** – Circuito identificado por **cc1**;

– RD – Operação de leitura;

– WR**w** – Operação de escrita onde o valor de **w** é escrito na saída do atuador associado a um determinado circuito; o valor de **w** pode variar entre 0 e 100; para circuitos do tipo tudo ou nada, 0 equivale a desligar o circuito e 100 a ligá-lo; para circuitos que sejam regulados o valor indica a proporção do controlo desejado;

– ST**s** – estado da resposta, onde o valor de **s** corresponde a 0 para um comando bem sucedido e a valores entre 1 e 99 nos casos de erro;

– VL**v** – o valor de **v** indica o valor lido na saída do atuador associado a um determinado circuito**;** para leituras com sucesso, **v** varia entre 0 e 100; para leituras com erro, **v** assume valores superiores a 100, identificando um erro ao nível do protocolo RS232;

– DV**d** – o valor de **d** identifica a divisão da instalação; este comando permite ler o estado de todos os circuitos associados à divisão **d**;

– EX**x** – executa um comando previamente definido no `SimplyWay`.

A2. RESULTADOS

De seguida, apresenta-se a lista dos resultados dos comandos associados ao campo ST:

– ST0 – operação bem sucedida;

– ST1 – a operação especificada não é de leitura ou escrita;

– ST2 – o parâmetro recebido não é válido;

– ST3 – comando inválido;

– ST10 – o circuito não existe na coleção de dados;

– ST11 – a divisão não existe na coleção de dados;

– ST20 – erro associado ao comando EX;

– ST30 – o valor passado como parâmetro não é válido.

Erros associados ao campo VL:

– VL997 – erro no protocolo RS232;

– VL998 – operação de leitura não realizada em tempo útil;

– VL999 – erro na gestão da fila de mensagens.

Apesar dos comandos descritos serem suficientes para o controlo da instalação, outros comandos podem vir a ser implementados desde que respeitem a sintaxe definida.