Instituto Politécnico de Tomar
Escola Superior de tecnologia de Tomar

Tiago Franco Silva

Desenvolvimento de um Sistema Domótico: monitorização e controlo das infraestruturas eletrotécnicas de uma habitação

Dissertação de Mestrado

Orientado por:

Doutor Mário Helder Rodrigues Gomes
Instituto Politécnico de Tomar

Prof. Doutor Paulo Coelho, IPT
Prof. Doutor Gabriel Pires, IPT
Prof. Doutor Mário Gomes, IPT

Dissertação apresentada ao Instituto Politécnico de Tomar
para cumprimento dos requisitos necessários
à obtenção do grau de Mestre
em Controlo e Eletrónica Industrial

RESUMO

Com o intuito de melhorar a qualidade de vida, aumentar o bem-estar e a segurança em residências habitacionais, a Domótica, também conhecida como edifícios inteligentes, é uma área em desenvolvimento que está a merecer grande destaque. Fazendo parte do ramo da automação industrial, esta área tem procurado oferecer soluções cada vez mais avançadas que proporcionam maior conforto e, simultaneamente, consumos de energia elétrica e de aquecimento cada vez menores. Isto deve-se ao facto de estas adotarem arquiteturas mais eficientes e sistemas de monotorização com auxílio de sensores e atuadores, tornando cada vez mais autónomas as casas do seculo XXI.

Esta dissertação apresenta um estudo e implementação de um sistema de monotorização e controlo de um protótipo para uma habitação. O projeto passou pela elaboração de alguns módulos que normalmente estão inseridos em infraestruturas de domótica. Este sistema é constituído por um barramento de comunicação CAN para a ligação de todos os sensores e atuadores que constituem a rede de Domótica. A interação homem - máquina consiste em duas aplicações desenvolvidas na ferramenta *Microsoft Visual* Studio, uma para Windows e outra para ambiente Web.

PALAVRA-CHAVE: Domótica, CAN, Sensores, Atuadores, Interfaces

ABSTRACT

In order to improve the quality of life, increase well-being and safety in residential homes, the Domotic, also known as intelligent building, it is a developing area and which has received great featured. Part of the branch of industrial automation, this area has sought to offer increasingly advanced solutions that provide added comfort and simultaneously consumption of electric energy and heating increasingly smaller. This is due to the fact that they adopted more efficient architectures and monitoring systems, with the help of sensors and actuators, making the houses of the twenty-first century are increasingly autonomous.

This dissertation presents a study and implementation of a system up monitoring and control of a prototype for a dwelling. The project went through the preparation of some modules that are typically entered in infrastructure of Domotic. This system comprises a CAN bus for connecting all the sensors and actuators which constitute the network of Domotic. The interaction man - machine consists of two applications developed in a tool Microsoft Visual Studio, one for Windows and one for Web environment.

Keywords: Domotic, CAN, Sensors, Actuators, Interfaces

AGRADECIMENTOS

Ao longo deste trabalho foram várias as pessoas que contribuíram para que fosse possível a realização deste projeto.

Quero começar por agradecer ao Professor Orientador Doutor Mário Gomes por todo o apoio, orientação e disponibilidade que me concedeu.

Ao Professor Doutor Gabriel Pires e ao Professor Doutor Carlos Ferreira pela disponibilidade e partilha de saberes que tão fundamentais foram para a elaboração deste projeto.

Ao Engenheiro Pedro Neves pelo companheirismo e entreajuda em aspetos que foram necessários para a concretização do sistema implementado.

Gostaria também de agradecer ao Jorge Cunha e ao Telmo Silva pelos esclarecimentos de algumas dúvidas que surgiram durante esta fase.

Aos meus pais pelo esforço a que se propuseram para que tornassem possível a realização desta etapa da melhor forma e pelo apoio incondicional que me deram no decorrer do mesmo.

Um especial agradecimento à Ana Fernandes por toda a compreensão, apoio e incentivo em todas as fases deste percurso.

A todos os professores da Escola Superior de Tecnologia Tomar que contribuíram para o meu percurso académico, partilhando os seus saberes.

A todos os meus familiares, amigos, colegas e conhecidos que, cada um da sua forma, me acompanharam durante este percurso de formação e de crescimento pessoal.

Índice

Índice de figuras

Índice de tabelas

Lista de Abreviaturas e Siglas

A/D: Analógico/Digital;

ADC: *Analogic Digital Converter* (Conversor Analógico Digital)*;*

ASP: *Active Server Pages* (Páginas de Servidor Ativo);

CAN: *Controller Area Network* (Rede Área Controlada);

CEBus: *Consumer Electronic Bus* (Barramento Eletrónico Consumidor);

CRC: *Cyclic Redundancy Check* (Verificação de Redundância Cíclica);

CRT: *Cathode Ray Tube* (Tubo de raios catódicos);

EEPROM: *Electrically Erasable and Programmable ROM* (ROM programável e eletricamente apagável);

EIA: *Electronic Industries Association* (Associação de Industrias Eletrónicas);

EIB: *European Installation Bus* (Instalação de Barramento Europeia);

IP: *Internet Protocol* (Protocolo Internet);

I/O: *Input/Output* (Entrada/Saída);

I2C: *Inter-Integrated Circuit* (Circuito Inter-Integrado)*;*

LAN: *Local Area Network* (Rede de Área Local);

PIC: *Peripheral Interface Controller* (Interface Periférica Programável);

PWM: *Pulse-Width Modulation (* Modulação por largura de pulso)*;*

RTR: *Remote transmission request* (Solicitação de transmissão remota);

SGBD: Sistema de Gestão de Base de Dados;

SPI: *Serial Peripheral Interface* (Interface Periférica Série);

SOF: *Start of frame* (Início da estrutura);

SQL: *Structured Query Language* (Linguagem de Consulta Estruturada);

USART: *Universal Synchronous Asynchronous Receiver Transmitter* (Transmissor Recetor Síncrono Assíncrono Universal);

UMTS: *Universal Mobile Telecommunication System* (Sistema Universal de telecomunicações móveis);

USB: *Universal Serial Bus* (Barramento Série Universal);

WAN: *Wide Area Network* (Rede Área Ampla);

XML: *Extensible Markup Language* (Linguagem de Marcação Extensível);

Capítulo 1

1 Introdução

1.1 Enquadramento e Motivações

A alteração do perfil demográfico das sociedades, o avanço tecnológico e os novos modos de vida tendem a provocar mudanças substanciais tanto nos edifícios públicos, como nas residências habitacionais. Todas estas transformações têm ocorrido nas últimas décadas, exigindo às novas infraestruturas habitacionais, uma nova arquitetura que atendam às novas exigências das famílias e à melhoria da eficiência energética.
Tendo em conta os novos requisito, têm sido estudadas e instaladas tecnologias mais sofisticadas, muitas vezes associadas a materiais de construção civil mais evoluídos. Estas questões só serão bem-sucedidas e os objetivos totalmente alcançáveis na presença de sistemas de domótica, cumprindo com as exigências dos consumidores atuais.

A motivação principal para a realização deste trabalho resulta da importância crescente que as designadas "casas inteligentes" têm vindo a ganhar numa sociedade moderna. Contudo, os custos de uma instalação de domótica são elevados e, por vezes, são utilizados sistemas proprietários, muito pouco flexíveis, tornando o custo de implementação muito elevado.

Com a constante evolução da eletrónica no campo dos sensores e atuadores e das redes de comunicação de dados, é-nos possível, agora, dispor de uma grande variedade de equipamentos com preços mais acessíveis e com a capacidade de se integrarem numa rede de comunicação. Tudo isto, juntamente com o grande crescimento dos serviços de internet, banda larga e a evolução das redes móveis 3G, 4G, UMTS (*Universal Mobile Telecommunication System*) e HSDPA (*High-Speed Downlink Packet Access*) que disponibilizam o acesso à internet em qualquer lugar, a partir de *SmartPhones*, telemóveis, *tablets* e computadores. Estes avanços abrem caminho para a possibilidade de aceder aos sistemas domóticos implementados através do acesso à internet com um simples Web *browser*.

1.2 Arquitetura do cenário DomoCenter

DomoCenter é o nome atribuído ao trabalho desenvolvido neste projeto, e refere-se à construção de um sistema domótico com interligação a uma rede de comunicação e de interface com diversos sistemas sensoriais e de atuação. O objetivo deste trabalho consiste no estudo de tecnologias na área da domótica e com o âmbito de criar um sistema de baixo custo e de forma a permitir a interligação de dispositivos e equipamentos, objeto de monitorização e controlo em ambiente habitacional.

Para este efeito, utilizo um conjunto de tecnologias de modo a construir um sistema para funcionar de forma integrada, permitindo uma gestão dos diferentes recursos que uma habitação pode dispor.

Para a construção de uma rede domótica é necessário ter em conta diversos aspetos, dos quais se salientam:

- Os sensores registam os valores e informações do local, como a temperatura, luminosidade, humidade e grandezas elétricas.
- Os microcontroladores associados aos atuadores, realizam o controlo de motores (estores, portas), como também, ligar, desligar e regular a iluminação. Outro parâmetro que pode ser controlado é a temperatura através da climatização.
- A interface disponibiliza visualmente a informação recebida pela rede de domótica para o utilizador.
- Os dispositivos específicos, tais como, *modems* ou *routers* permitem o envio de informação para o utilizador que esteja no domínio da habitação ou, até mesmo, fora dela.

Este sistema poderá ser acedido através de qualquer computador, *tablet* ou telemóvel com ligação à internet.

1.3 Estrutura da dissertação

A presente dissertação está estruturada através de seis capítulos que descrevem os aspetos essenciais desta temática, estado da arte, e a aplicação desenvolvida no âmbito do projeto final de curso.

No capítulo 2 descrevem-se as características de um sistema domótico, exemplificando as suas possíveis aplicações. Além de se apresentar de forma resumida as mais conhecidas tecnologias existentes no mercado, descrevendo, para cada uma delas, as principais características e analisando as suas vantagens e desvantagens.

No capítulo 3 são referidas as tecnologias utilizadas neste projeto. Desde a comunicação usada na rede de domótica, bem como os microcontroladores usados na mesma. Também são descritos as ferramentas (Visual Studio, .NET, IIS7, SQL Server) usadas para a realização da aplicação.

No quarto capítulo é descrito o funcionamento de todo o sistema domótico desenvolvido, desde o barramento de comunicação CAN (*Controller Area Network*), os dispositivos e a interface que realiza o fluxo de informação entre a rede domótica e o computador.

No capítulo cinco está especificado o funcionamento das interfaces gráficas, tais como a aplicação Windows e em ambiente Web. Está ainda descrito e explicado o funcionamento com a Base de Dados que é o suporte de toda a aplicação.

Por fim, no último capítulo são apresentadas as conclusões mais importantes deste trabalho.

Capítulo 2

2 Domótica

2.1 Introdução

O termo "domótica" resulta da fusão entre a palavra de origem latina "*domus*" (casa) com "telemática" (eletrónica + informática), ou seja, a domótica define-se como a possibilidade de controlo de forma automática das infraestruturas, tornando-as no que se costuma designar por casas inteligentes [1].

Do ponto de vista de um utilizador, a domótica pode ser vista como um sistema inteligente que lhe proporciona uma melhor qualidade de vida com recurso às novas tecnologias existentes. Deste modo, oferece uma redução do trabalho doméstico, um acrescento bem-estar, maior segurança dos habitantes e um melhor controlo a nível do consumo energético da habitação.

Do ponto de vista tecnológico, a definição de domótica pode ser caracterizada pela integração de diversos equipamentos domésticos em um sistema, com o intuito de comunicarem entre si através de um canal de comunicação, para que possam realizar tarefas de forma programada.

2.2 Funcionalidades

2.2.1 O Conforto

Quando é instalado um sistema de Domótica em uma habitação, os residentes pretendem que esta lhe dê algum conforto. Este conceito é destinado essencialmente às instalações AVAC (aquecimento, ventilação e ar condicionado), embora também se possa incluir nesta área todos os sistemas que possam contribuir para a comodidade e bem-estar dos utilizadores dos edifícios que tenham redes de domótica.

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

- Controlo e supervisão de todos os sistemas do edifício;
- Automatização da irrigação de jardins;

- Controlo por infravermelhos e transmissão RF dos vários automatismos presentes na habitação;
- Acionamento automático de sistemas com base em dados do meio ambiente, como por exemplo a recolha de toldos ou o fecho de persianas e janelas em função da temperatura ou das condições meteorológicas.

2.2.2 A Eficiência Energética

Falar em eficiência energética significa falar no consumo de energia elétrica, a sua gestão e racionalização, mantendo os mesmos serviços energéticos sem diminuir o conforto e a qualidade de vida.

O consumo energético numa habitação depende de diversos fatores, tais como a zona onde se situa, a qualidade de construção, o nível de isolamento, o tipo de equipamentos utilizados e o uso que lhes damos. É certo que com o uso de sistemas de domótica, estes consumos serão reduzidos significativamente.

Em Portugal, o setor residencial, com cerca de 3,9 milhões de alojamentos, contribui com uma parcela de 17,7% do consumo de energia final em termos nacionais, representando cerca de 30% do consumo de eletricidade, o que evidencia, desde logo, a necessidade de moderar especialmente o consumo elétrico [2].

Outra causa para o aumento do consumo de energia reside na ineficiência dos próprios equipamentos utilizados no sector, edifícios, procedimentos e hábitos de utilização desses mesmos equipamentos. Isto deve-se, não só a razões comportamentais dos consumidores, como também ao período necessário para a substituição dos equipamentos e progressiva recuperação dos edifícios

Nas habitações portuguesas, o consumo energético das mesmas têm vindo a registar um crescimento significativo, em parte, também devido ao aumento da aquisição de equipamentos consumidores de energia.

Em relação ao consumo elétrico de uma habitação média, de acordo com [2] esta consome cerca de 3.700kWh por ano, divididos da forma como se indica na Figura 1.

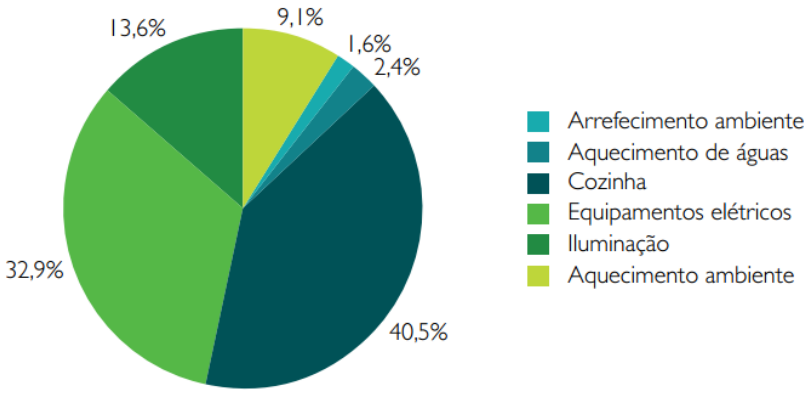

Figura 1 – Consumo de eletricidade discriminado em uma habitação média em Portugal

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

- Agendamento do uso de equipamentos domésticos com cargas mais elevadas, como por exemplo, máquinas de lavar roupa e lavar loiça, utilizando as tarifas bi-horárias.
- Aproveitamento da luz solar, com a atuação das persianas, no caso do pré-aquecimento de uma divisão com o sistema de ar condicionado ou simplesmente para iluminação do espaço.
- Utilização de sensores de presença para redução de consumos, tais como, a suspensão dos sistemas de climatização ou iluminação na ausência dos utilizadores.
- Monitorização dos consumos de água e/ou de gás por zonas, podendo, desta forma, detetar possíveis fugas ou atos de vandalismo num edifício.

2.2.3 A Segurança

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

A segurança é uma das vantagens inerentes ao conceito de domótica, podendo este ser dividido em duas áreas: a segurança das pessoas e a segurança dos bens.

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

- Iluminação automática em zonas de risco, na ausência do(s) habitante(s).

- O controlo de acesso à habitação, pode ser feito através da leitura de padrões biométricos (impressão digital, padrão retinal, padrão de voz), ou de forma mais elementar, através de acesso por palavra-chave.
- Alarmes de emergência médica. Em caso da existência de pessoas com necessidades especiais (como idosos e pessoas incapacitadas), existem acionamentos de emergência cuja ativação gera um aviso (pré-definido) para o telemóvel de um familiar ou para os serviços de emergência.

Na categoria da segurança de bens podem-se enumerar tarefas como:

- Deteção de eventuais focos de incêndio no interior de uma habitação, atuando sobre os aspersores de emergência na divisão onde foi detetada a anomalia.
- Simulação de presença na habitação, através de um modo aleatório que altere o estado da iluminação, dos estores e de outros equipamentos.
- Alarmes de fuga de gás e inundações.

2.3 Arquitetura

Independentemente do tipo de sistema de automação aplicado, habitações de pequena escala ou em habitações de elevadas dimensões (como hotéis e edifícios comercias) o objetivo será sempre o melhoramento da interação e da comunicação entre os dispositivos instalados nos edifícios.

Certamente que os requisitos, o nível de complexidade e o tipo de arquiteturas inerentes a estas duas situações são diferentes, mas em ambos os casos existirá sempre um ponto em comum, a comunicação entre os sensores e os respetivos atuadores. Para este efeito utilizam-se barramentos de comunicação de dados, com algumas semelhanças das já bastante comuns redes de sensores (utilizadas no domínio da automação industrial).

A arquitetura dos sistemas de automação pode ser classificada com base no local onde se situa o "cérebro" do sistema. Desta forma, pode-se conceber um sistema domótico com uma arquitetura centralizada, uma arquitetura descentralizada, uma arquitetura distribuída ou uma arquitetura mista (combinação das anteriores).

A arquitetura centralizada de um sistema de domótica (Figura 2), significa que existe um controlador central que está programado para receber informação dos sensores e a

informação introduzida pelo utilizador através da interface, atua nas saídas do controlador – os atuadores.

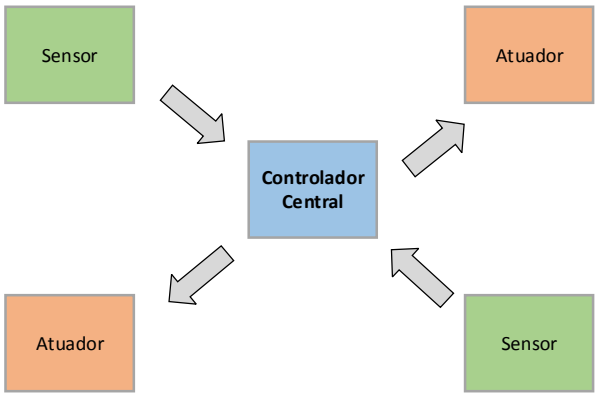

Figura 2 - Arquitetura Centralizada

Ao contrário da arquitetura anterior, na arquitetura descentralizada existem vários controladores distribuídos (cada um com os seus sensores e atuadores locais), interligados entre si por um barramento de dados para trocarem informação, conforme representado na Figura 3.

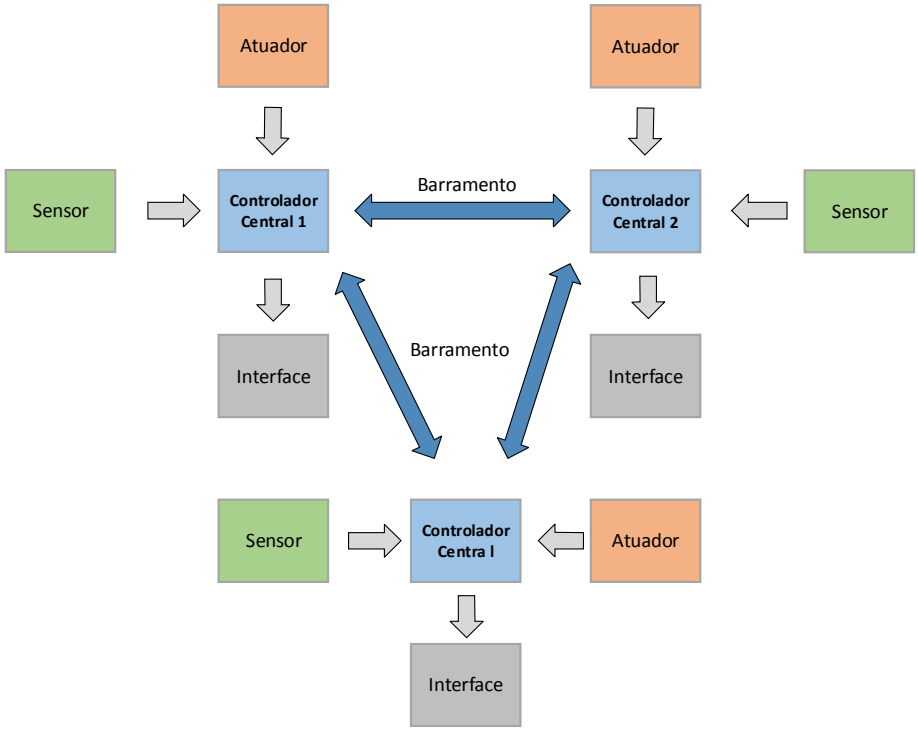

Figura 3 - Arquitetura descentralizada

Num sistema de domótica, a arquitetura distribuída caracteriza-se pela particularidade de cada elemento do sistema (um atuador, um sensor, uma simples interface, e um controlador) ser capaz de atuar e enviar informação para o barramento de dados, de acordo com o seu algoritmo e o tipo de comunicação do barramento [3].

2.4 As Comunicações

O avanço das novas tecnologias no domínio computacional, das telecomunicações e das redes de transmissão de dados, faz com que os sistemas de domótica sigam estes avanços, usando tecnologias baseadas nestas áreas. Existem empresas com departamentos específicos que se dedicam exclusivamente à investigação e ao desenvolvimento de novas arquiteturas e sistemas de integração.

Como foi referido anteriormente, a evolução da microeletrónica, das redes de transmissão de dados, com um elevado nível de desenvolvimento na área dos semicondutores e o aparecimento de novas metodologias destinadas à interface Web de serviços, contribuem para o aparecimento de novas possibilidades no meio de transmissão na domótica.

Dentro das possíveis soluções de transmissão de dados abrangidas nesta área, as que têm vindo a desenvolver-se são iniciativas de supervisão, controlo e monitorização de instalações domóticas à distância [3].

Dentro desta área destacam-se dois métodos:

- O controlo remoto dos sistemas de domótica pode ser feito pela implementação da tecnologia GSM/GPRS através do envio de mensagens SMS. O sistema poderá responder para o telemóvel do utilizador com informação de vários alarmes. No caso de intrusão nas habitações, pode ser enviado uma mensagem de alarme ou até mesmo uma imagem da câmara de segurança da divisão onde foi detetada.
- Outra metodologia para a comunicação remota pode ser realizada através da internet, via TCP/IP usando um *browser* e tecnologia Web para a implementação deste tipo de serviços. Este tipo de comunicação foi escolhido para o trabalho realizado neste projeto final de curso.

2.5 Tecnologias existentes

Atualmente, existem inúmeras soluções comerciais baseadas em vários protocolos criados para sistemas de automação de edifícios.

Existem sistemas desenvolvidos nos Estados Unidos, como o X10, o CEBus (*Consumer Electronics Bus*), o LonWorks, o Smart House, e sistemas inicialmente desenvolvidos na europa, como o BatiBUS, o EIB (*European Installation Bus*) e o EHS (*European Home Systems*).

No final da década de 90 surge a associação *Konnex*, que resulta da fusão entre a EIBA (*European Installation Bus Association*), BCI (*Batibus Club International*) e EHSA (*Europe Home System Association*), associações responsáveis pelo sistema EIB, BatiBUS e EHS, respetivamente. O objetivo desta associação seria definir um único protocolo, aberto e padronizado internacionalmente, para os sistemas de automação de habitações e edifícios, o KNX.

Existem sistemas desenvolvidos em Portugal, tais como o VIVIMAT, Domus-int e o CARDIO, capazes de oferecer a monotorização e o controlo da instalação de domótica via Web [3].

2.5.1 X-10

O X10 é o protocolo mais antigo e usado nas aplicações domóticas. Foi desenvolvido em 1976 pela empresa *Pico Electonics* com o objetivo de transmitir dados por circuitos elétricos de baixa tensão (110V nos EUA e 230V na Europa) a muito baixa frequência (60 Hz nos EUA e 50 Hz na Europa) e com custos muito baixos [4]. A vantagem da utilização da rede elétrica consiste em não ser necessário a colocação de novos cabos, sendo este o principal meio de comunicação. Além disso, a facilidade de implementação e a possibilidade de utilizar uma arquitetura descentralizada constituem alguns dos aspetos que contribuíram para o seu sucesso.

Os controladores enviam códigos/sinais pela rede elétrica dirigidos aos módulos recetores que se pretendem controlar. Deste modo é necessário definir os endereços quer dos controladores, quer nos módulos recetores.

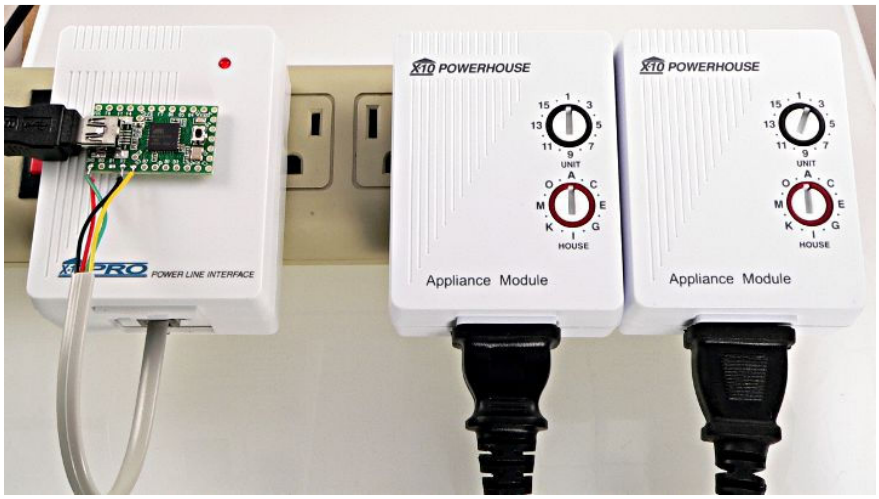

Figura 4 - Módulos X10

Os módulos da Figura 4 apresentam um conjunto de 16 letras (A-P) e 16 números por letra no que resulta um total de 256 combinações possíveis. Existe a possibilidade de vários módulos recetores partilharem o mesmo endereço, constituindo uma zona, ou seja, atuam em simulamento à mesma mensagem proveniente do módulo emissor.

Cada controlador tem um *House Code* (código de casa) até um máximo de 16 endereços, correspondente ao comando de 16 grupos de módulos recetores. Por seu lado, os recetores além do *House Code*, são endereçados pelo *Unit Code* (código de unidade) que corresponde ao número da zona de que o módulo faz parte [5].

Neste protocolo qualquer ação que se faça inclui duas mensagens. A mensagem inicial identifica o dispositivo e a outra a ordem a executar. As mensagens são enviadas duas vezes de modo a minimizar falhas.

Em relação ao meio de transmissão consiste na inserção de sinais de alta frequência (120kHz) na rede elétrica, representando sinais binários. A inclusão do sinal é feita logo após a passagem da onde sinusoidal de 50Hz pela origem, obtendo no máximo um atraso de 200 microssegundos. Posteriormente, o sinal é enviado pela rede elétrica para os módulos recetores. Na figura abaixo está ilustrada uma onda sinusoidal com a introdução de um sinal X10 [6].

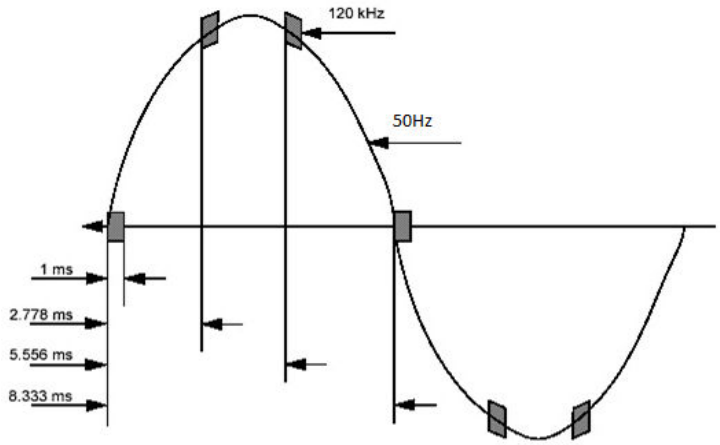

Figura 5 - Onda sinusoidal com a inserção de um sinal X10 [7]

A principal desvantagem deste protocolo tem a ver com a baixa velocidade de comunicação e com o facto de não conter controlo na receção da mensagem por parte dos dispositivos recetores.

Em contrapartida, graças ao amadurecimento (mais de 30 anos de mercado) e à tecnologia implementada, os produtos X-10 têm um preço muito competitivo sendo líder no mercado residencial Norte-americano. As instalações podem ser realizadas por eletricistas sem conhecimento de automação ou informática ou até pelos próprios utilizadores.

2.5.2 *LonWorks*

A tecnologia *LonWorks* é uma topologia de rede criada pela *Echelon Corporation* em 1992, com o intuito de solucionar os problemas de controlo de sistemas e encontra-se atualmente na versão 2.0 [6].

Apesar de esta tecnologia estar desenhada para todas as aplicações, esta é no entanto maioritariamente usada pelas indústrias, edifícios administrativos e hotéis, permitindo a implementação de redes de controlo distribuídas de uma forma automática. Isto deve-se ao facto de permitir que uma instalação suporte a monitorização de cerca de 3200 dispositivos (Figura 6). Os meios de comunicação poderão ser feitos através de cabos de fibra ótica, cabos coaxiais, infravermelhos, radiofrequência ou par entrelaçado.

Figura 6 - Rede LonWorks

Este protocolo corresponde a um sistema aberto, implementando todas as camadas do modelo OSI (*Open Systems Interconnection*), através de processador designado de *Neuron*, registado pela *Echelon*. Este Chip (Figura 7) é um identificador único, que permite controlar todos os dispositivos inseridos na topologia da rede. O identificador possui 48 *bit* ficando registado na memória *EEPROM* durante o processo de fabrico.

Figura 7 - Chip *Neuron* da série 5000

Para reduzir as colisões entre transferências é utilizado um mecanismo de acesso ao meio de transmissão denominado de *Predictive p-persistent CSMA protocol*. Este mecanismo permite obter excelentes desempenhos, mesmo em condição de sobrecarga da rede, pois atribui o acesso ao meio a cada uma das estações de uma forma aleatória. No protocolo LonWorks são utilizados no mínimo 16 níveis de diferentes prioridades de acesso ao meio e no máximo 1008, este valor é ajustado dinamicamente, consoante a estimativa de carga na rede.

Para o endereçamento dos pacotes podem ser usados três tipo de endereços, *Device Address*, *Group Address* e *Broadcast Address*, consoante se queira transmitir para um único módulo, para um grupo de módulos ou para todos os módulos, respetivamente.
Este protocolo prevê ainda a utilização de quatro tipos de mensagens para otimização da qualidade do serviço, designados por *Acknowledged Messaging*, *Repeated Messaging*, *Unacknowledged Messaging* e *Authenticated Messaging* [8].
Para finalizar pode-se salientar que é uma tecnologia fiável e robusta, facilitando a implementação de dispositivos de vários fabricantes.
A implementação doméstica desta tecnologia não é viável, devido ao custo elevado do chip *Neuron*, havendo tecnologias disponíveis com custos mais baixos.

2.5.3 CEBus

O sistema CEBus (*Consumer Electronic Bus*) foi normalizado pela EIA (*Electronic Industries Association*), com o objetivo de desenvolver uma rede de comunicação para ambiente residencial.
O protocolo CEBus utiliza um modelo de comunicação de dispositivo a dispositivo sem hierarquias nem restrições, usando um barramento comum.
O CEBus define dois tipos de canais de comunicação: um canal de controlo para mensagens entre dispositivos, e um conjunto de canais de dados para distribuição de áudio, vídeo ou outro qualquer sinal de banda larga.
A norma CEBus disponibiliza vários meios físicos de transmissão:

- Linha de energia elétrica (PL – *Power Line*);
- Cabo de par entrançado (TP – *Twisted Pair*);
- Cabo Coaxial (CX – *Coaxial*), preferencialmente para dispositivos que utilizam sinais de áudio e vídeo;
- Rádio frequência (RF – *Radio-Frequency*);
- Infravermelhos (IR – Infra-Red).

Com os diversos meios físicos de transmissão pode-se encontrar nesta tecnologia a melhor solução para as diferentes implementações. Existe apenas a necessidade de implementar

um dispositivo capaz de comunicar entre os dois meios de transmissão (*Router*), como se pode ver na Figura 8.

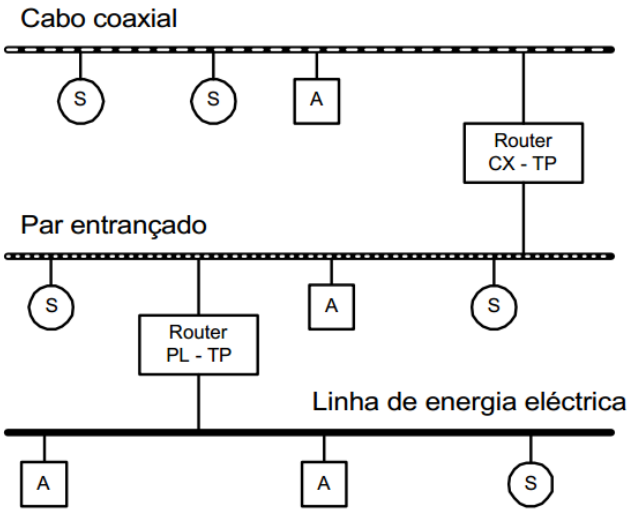

Figura 8 - Ligação entre diferentes meios de transmissão (S-sensor e A-Atuador) [9]

Em cada nó onde está inserido o dispositivo CEBus pode ser dividido em três partes, ver Figura 9. A parte designada por PROTOCOLO é igual em todos os dispositivos, sendo esta a responsável pela receção e transmissão das mensagens pelo meio físico. O dispositivo interpreta as mensagens Cal provenientes do barramento, através da camada CAL. Por último, a parte designada de APLICAÇÃO representa a aplicação do nó e é constituída pelo *hardware* e *software* que definem a operação do dispositivo.

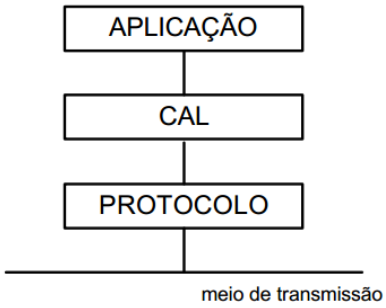

Figura 9 - Camadas constituintes de um dispositivo CEBus [10]

Em relação ao modelo de mensagens CEBus, estes possuem um *start code*, campo de controlo, endereço de destino, endereço de origem, mensagem Cal e um campo chamado

de *Checksum*. O *start code* é constituído por um valor aleatório de 8 *bit* que tem como função efetuar a deteção e resolução de colisões durante o acesso ao canal. O campo de controlo identifica o tipo de serviços da ligação de dados (DLL - *Data Link Level*) que o pacote contém, sendo a sua dimensão de 8 bit. O campo de destino e origem têm uma dimensão de 32 bit, em que 16 bit correspondem ao endereço de sistema e os restantes 16 bit correspondem ao endereço do nó. A mensagem Cal possui o tamanho máximo de 32 bytes. O último campo da mensagem possui um tamanho variável de 8 bit e serve para determinar a integridade do conteúdo da mesma. Podemos visualizar na Figura 10 o formato das mensagens CEBus [10].

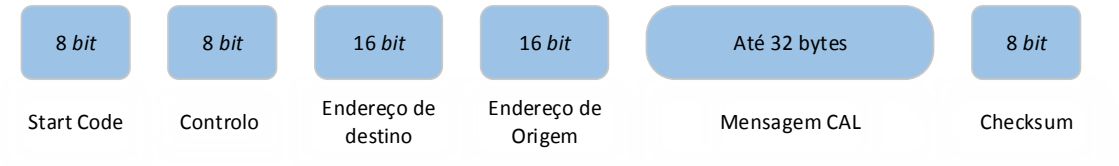

Figura 10 - Formato das mensagens CeBus

2.5.4 KNX

Como já foi referido, o KNX é um protocolo de comunicação desenvolvido por um conjunto de empresas líderes no mercado europeu de material elétrico, com o intuito de criar um padrão europeu para permitir a comunicação entre todos os dispositivos de uma instalação de uma habitação ou num edifício. Este sistema garante que os produtos (de diferentes fabricantes) utilizados em diversas aplicações operem e comuniquem entre si sem seguir qualquer hierarquia. Isto permite assegurar um importante grau de flexibilidade na extensão e na modificação de instalações [3].

O KNX define-se como uma rede totalmente distribuída, uma vez que não necessita de um controlador central na instalação. Todos os dispositivos que se ligam ao barramento de comunicação de dados têm o seu próprio microprocessador integrado e toda a eletrónica de acesso ao barramento (Figura 11).

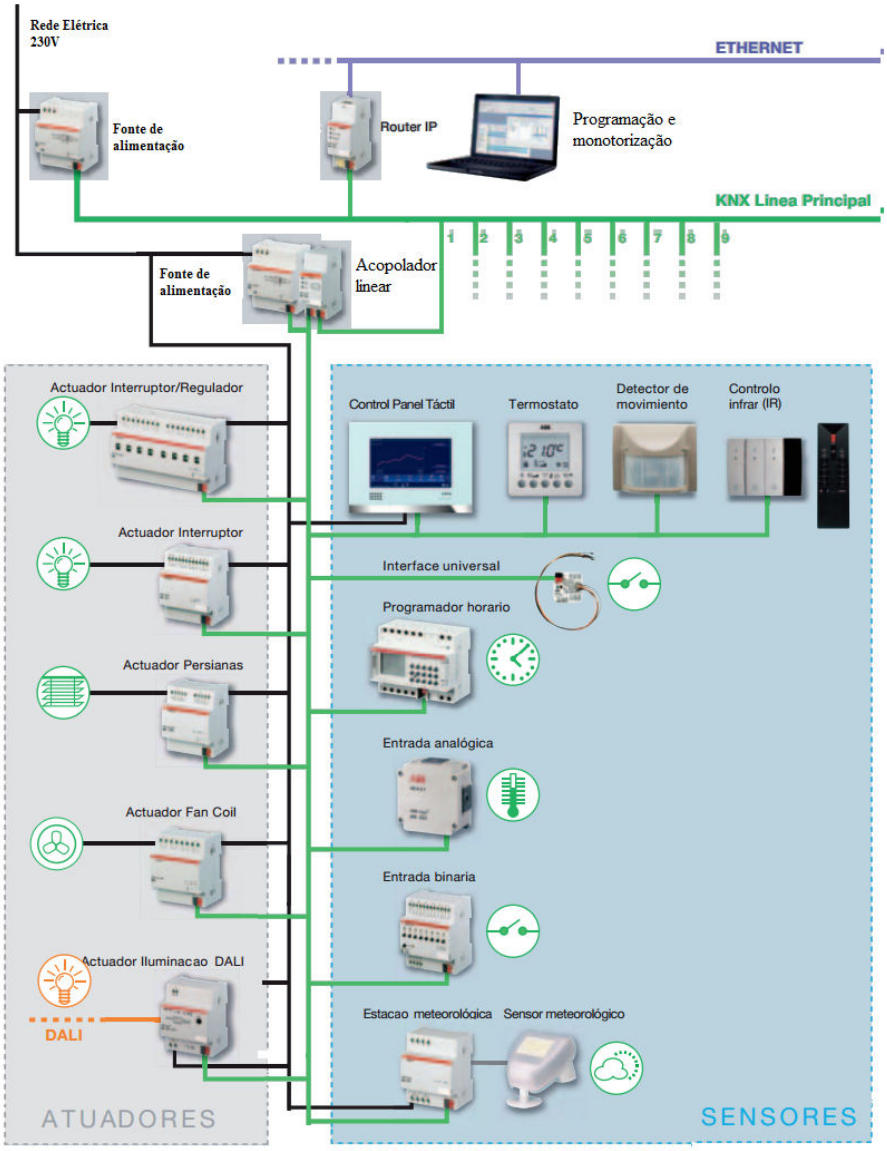

Figura 11 - Sistema KNX

Como se pode ver na figura, as instalações KNX usam a rede elétrica para alimentar os dispositivos e o barramento de comunicação por linha é alimentado por um dispositivo que faz a *Gateway* para a camada principal.

Um dispositivo KNX é composto por uma unidade de acoplamento ao barramento (designada por BCU – *Bus Coupling Unit*) e por um módulo de aplicação (AM - *Application Module*). A entidade BCU é responsável pela interface com a rede e tem capacidade de armazenamento de informação, como por exemplo o endereço físico, um ou mais endereços de grupo e parâmetros do nó. Enquanto a entidade AM é unicamente

responsável pela interface com o utilizador ou dispositivo, como por exemplo, botão de pressão, interruptores, sensores de temperatura, sensores de presença, etc [10].

A *frame* utilizada pelo protocolo KNX é composta por 7 campos, como representado na Figura 12.

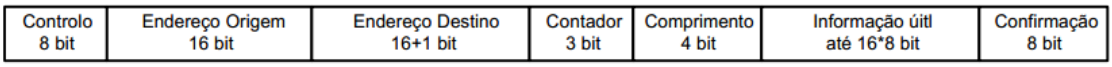

Figura 12 - Formato da *frame* KNX

A *frame* é iniciada pelo campo de controlo que tem como objetivo atribuir a prioridade de transmissão em função dos dados. O campo designado por endereço de origem contém o endereço físico do dispositivo. Este é composto por 4 *bit* que definem a área em que se encontra o dispositivo, 4 *bit* que definem a linha e os últimos 8 *bit* caracterizam o próprio dispositivo. O campo de endereço de destino tem mais um bit do que o endereço de origem para permitir que se enderece um grupo de dispositivos, sendo o 17º *bit* utilizado para indicar se o endereço representado pelos outros 16 *bit* são de um endereço físico ou de grupo.

O campo de comprimento, como o próprio nome indica, informa o número de *byte* que o campo de informação útil contém. O campo de confirmação é utilizado como mecanismo de deteção de erros na transmissão dos dados. Este campo é utilizado no dispositivo de receção para validar a mensagem recebida e poder enviar ao emissor um aviso de confirmação de receção correta.

Cada 8 bit de dados são transmitidos em grupo, formando uma palavra. A palavra é constituída pelos 8 bit mais um bit inicial ST (*Start* bit), um bit de paridade Pz (*Parity bit*) e um bit final SP (*Stop* bit). Entre palavras deverá existir uma pausa de 2 bit [10], Figura 13.

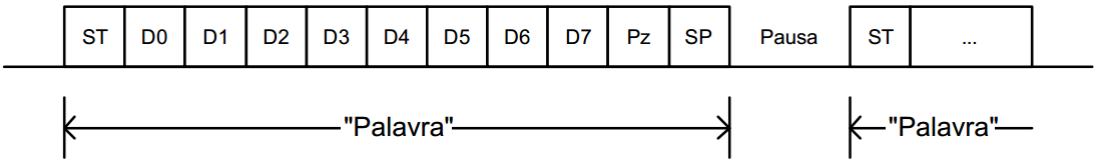

Figura 13 – Estrutura de uma palavra KNX

Com as condicionantes atrás referidas e com um ritmo de transmissão de 9600 bps (*bit per second*), o tempo que demora a transmitir uma palavra é 1,35 ms.

Em relação à qualidade deste produto, a associação *Konnex* requer um alto nível de controlo de qualidade durante todas as etapas da vida do equipamento. Assim, todos os produtos desenvolvidos pelos membros da *Konnex*, que assumem a marca KNX, têm de demonstrar compatibilidade com a norma ISO 9001.

Devido ao facto de a norma KNX ter sido uma fusão de três sistemas já existentes anteriormente com as suas respetivas vantagens, não apresenta grandes desvantagens.

Os pontos mais desfavoráveis que se podem destacar desta tecnologia, são a relativa complexidade das instalações e o custo também relativamente elevado equipamento (que é certificado).

Capítulo 3

3 Tecnologias adotadas

Para a elaboração deste projeto foi realizada, numa primeira fase, uma análise dos tipos de tecnologias e componentes disponíveis de modo a ir ao encontro dos objetivos do trabalho. Neste âmbito descrevem-se as razões pela qual se adotam umas tecnologias comparativamente com outras na aplicação desenvolvida.

3.1 Rede de Domótica

Sendo este um projeto puramente académico mas com bases realistas, houve a necessidade de minimizar sempre os custos associados com a tecnologia a utilizar. Desta forma, e tentando aplicar conhecimentos adquiridos ao longo do curso de mestrado, entendi ser uma boa solução implementar um barramento de comunicação utilizando microcontroladores ligados ao mesmo.

3.1.1 Protocolo de comunicação utilizado

Como a escolha da comunicação entre os dispositivos da rede de domótica recaiu na opção de usar microcontroladores, a primeira solução estudada foi a de utilizar a comunicação TCP/IP para aceder aos dispositivos. Após algum tempo a testar esta hipótese, verifiquei que não reunia os requisitos para este projeto. Esta conclusão deve-se ao facto de cada microcontrolador ter de estar ligado à rede Ethernet, limitando muito o local em que estes teriam que ser instalados.
Parti para outra hipótese, tendo sido o protocolo CAN (*Controller Area Network*) aquele que mereceu maior destaque, sendo este adotado como protocolo de comunicação.
É certo que este protocolo não foi desenhado para ser utilizado em sistemas de domótica. Mas uma vez que contém todo um conjunto de características necessárias para um sistema deste tipo, como: elevada fiabilidade, rapidez e enorme versatilidade em conectar-se com os microcontroladores, foi eleito para este projeto.

3.1.2 Microcontroladores PIC18FX58 e *Transceiver* MCP2551

Como já foi referido, o desenvolvimento deste projeto incidiu no uso de microcontroladores (fabricante *Microchip Technology)*.

Cada PIC (*Peripheral Interface Controller*) da vasta família de microcontroladores tem as suas características próprias que lhe conferem a sua potencialidade. É errado dizer que existe um PIC melhor que outro. O melhor PIC é, sem dúvida, aquele que satisfaz os requisitos do projeto em questão. Assim sendo, cabe ao utilizador escolher o dispositivo com as características que mais se adequa ao trabalho a desenvolver.

Neste projeto, a escolha do PIC recaiu nos seguintes parâmetros:

- Número de entradas/saídas (I/O) necessárias;
- Comunicações (USART, USB);
- Temporizadores;
- Conversores A/D;
- Interrupções;
- Preço.

De acordo com estes requisitos, e dos conhecimentos adquiridos, os PIC mais apropriados são os PIC18F258 e PIC18F458. Como o nome indica, ambos os microcontroladores são da série PIC18F e partilham características muito idênticas. A diferenciação mais óbvia está no aspeto físico, devido ao número de pinos. Estes dois dispositivos contêm funcionalidades como módulo CAN, módulo USART (*Universal Synchronous Asynchronous Receiver Transmitter*), temporizadores, PWM (*Pulse-Width Modulation*), SPI (*Serial Peripheral Interface*), I2C (*Inter-Integrated Circuit*) e conversores analógicos digitais (ADC - *Analogic Digital Converter*).

Na tabela seguinte indicam-se algumas das especificações destes dois microcontroladores [11].

Tabela 1 – Características dos PIC: PIC18F258 e PIC18F458

	PIC18F258	**PIC18F458**
Frequência de operação	Até 40MHz	Até 40MHz
Memória Flash (Bytes)	32K	32K
Memória Dados (Bytes)	1536	1536
EEPROM (Bytes)	256	256
10 bit ADC	5	8
Comparador Analógico	-	2

A tensão de alimentação suportada por ambos está compreendida entre os +4.2V e +5.5V. Na Figura 14 apresenta-se as fotografias destes dois PIC.

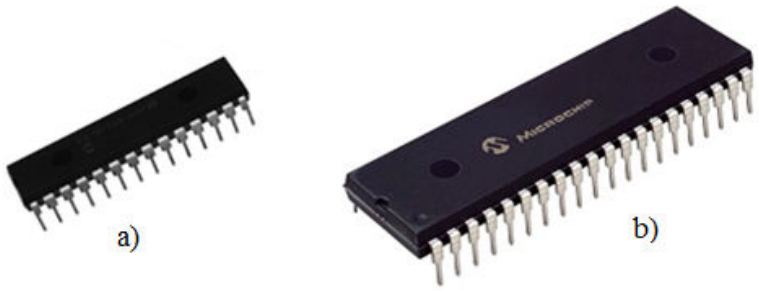

Figura 14 – Imagens dos PIC: a) PIC18F258 e b) PIC18F458

Como estes PIC contêm internamente o módulo CAN e como o protocolo de comunicação adotado é o CAN, então a razão mais importante para escolha deste modelo, da vasta família PIC18F, foi esta característica. Foram utilizados os dois PIC, sendo que a escolha do microcontrolador de 40 pinos (PIC18F458) deve-se ao facto de um dos módulos da rede de domótica deste projeto necessitar de mais de 5 ADC (concretamente 6). Para ligar estes microcontroladores ao barramento de comunicação, para além do módulo CAN já implementado internamente no PIC, é necessário introduzir um dispositivo entre o PIC e o barramento CAN. Para tal, é utilizado um *transceiver* também da *Microchip* designado de IC MCP2551 (Figura 15) que é um controlador de alta velocidade. Este componente fornece transmissão diferencial e receção compatível com o controlador CAN. O MCP2551 permite velocidades até 1MBps de transmissão de dados em implementação de redes [12].

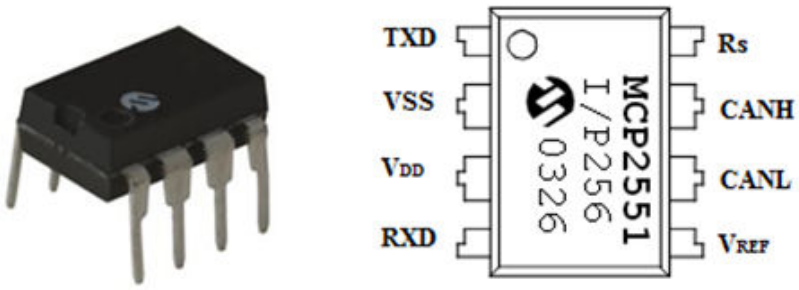

Figura 15 – Imagem e identificação dos pinos do IC MCP2551

Outro PIC da família PIC18F utilizado neste projeto é o PIC18F2550 (Figura 16) devido a ser um microcontrolador extremamente completo [13]. Uma das particularidades que o diferencia dos outros PIC já referenciados é o facto de este conter o módulo de comunicação USB, que é utilizado neste projeto.

Figura 16 - PIC18F2550

3.2 Software utilizado

Após ter sido definido o protocolo CAN como meio de comunicação da rede de domótica, é necessário determinar as ferramentas de software para a implementação da aplicação, neste caso em ambiente Windows. O interesse na escolha destas ferramentas para este projeto descritas acima incidiu na grande potencialidade que estes produtos dispõem.

Para a criação da aplicação, a escolha recaiu na utilização de uma vasta ferramenta da Microsoft, o *Visual Studio*. Para a programação da aplicação são utilizadas linguagens como o *Visual Basic*, C# e uma plataforma de desenvolvimento, como o ASP.NET. Para dar suporte à aplicação Web desenvolvida, é utilizado o SQL *Server* como servidor de base

de dados e o IIS (*Internet Information Service*) como servidor Web da aplicação. Todas estas ferramentas estão descritas nos seguintes pontos.

A - *Microsoft Visual Studio*

O *Microsoft Visual Studio* é um pacote de programas da empresa *Microsoft*, para o desenvolvimento de software, dedicado às linguagens *Visual Basic*, C, C++, C# e J# e à tecnologia .NET *Framework*. É também uma grande ferramenta para o desenvolvimento de produtos para a área da internet, usando ASP.NET.

B - .NET

Refere-se a uma plataforma única para o desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código criado para .NET pode ser executado em qualquer dispositivo ou plataforma que possua uma *Framework*. Uma *framework* consiste em um conjunto de classes, interfaces e padrões dedicados a solucionar um conjunto de problemas através de uma arquitetura de programação flexível e extensível [22].

A plataforma .NET baseia-se num dos princípios utilizados na tecnologia Java, os programas desenvolvidos são compilados duas vezes, uma na distribuição e outra na execução [14].

Um programa pode ser escrito em mais de vinte linguagens de programação disponíveis para a plataforma. O código criado pelo programador é compilado pela linguagem escolhida, criando um código intermédio numa linguagem chamada de MSIL (*Multi Service Interconnect Link*).

C - ASP.NET

É a plataforma da *Microsoft* para o desenvolvimento de aplicações Web. É um componente do IIS que permite criar páginas dinâmicas através de uma linguagem de programação .NET.

O ASP.NET é baseado na *Framework* .NET herdando todas as suas características, por isso, como qualquer aplicação .NET, as ferramentas para essa plataforma podem ser escritas em várias linguagens, como C# e Visual Basic.NET.

As aplicações Web desenvolvidas em ASP.NET necessitam da *Framework* .NET e do servidor IIS para serem executadas.

D - AJAX

AJAX provém do acrónimo em inglês de *Asynchronous Javascript and XML* que em português significa *Javascipt* Assíncrono e XML (*Extensible Markup Language*). É o uso metodológico de tecnologias como *Javascipt* e XML, para tornar páginas Web mais interativas, usando solicitações assíncronas de informação. Não é caracterizada como uma tecnologia mas sim por um conjunto de várias tecnologias bem conhecidas que trabalham em conjunto, oferecendo assim novas funcionalidades.
Ao usar AJAX no desenvolvimento de serviços para Web, a informação é carregada de maneira mais simples e precisa. O utilizador não precisa de aguardar que a página seja totalmente carregada quando faz uma requisição, pois o servidor só irá enviar os dados relevantes, reduzindo o tráfego de dados pela rede.

E - C#

C# (C Sharp) é uma linguagem de programação orientada a objetos criada pela *Microsoft* e faz parte da sua plataforma .NET. A companhia baseou C# na linguagem C++ e Java.
Esta linguagem foi criada com a arquitetura .NET, e embora existam várias outras linguagens que suportam esta tecnologia (VB.NET, C++, J#), C# é considerada a linguagem símbolo da plataforma .NET.

F- Servidor Web

IIS (*Internet Information Service*) - anteriormente denominado *Internet Information Server* - é um servidor Web criado pela Microsoft para os servidores dos seus sistemas operativos [15].
Uma das suas características mais utilizadas é a criação de páginas HTML dinâmicas que, ao contrário de outros servidores Web, usa tecnologia proprietária, o ASP (*Active Server Pages*), mas também pode usar outras tecnologias com adição de módulos de terceiros. Como no sistema operativo do servidor que corre a aplicação desenvolvida é o Windows 7, a versão do IIS é a 7.5.

G - SGBD

Relativamente ao SGBD (Sistema de Gestão de Base de Dados) usei a ferramenta *SQL Server* 2008 R2, também da *Microsoft*, para a criação das várias tabelas presentes na base de dados. O suporte a este tipo de base de dados (com a informação essencial) permite desenvolver aplicações que funcionem corretamente.

Capítulo 4

4 Sistema de Domótica desenvolvido

4.1 Rede de Domótica

4.1.1 Comunicação CAN

Com a evolução no ramo da eletrónica e das funcionalidades de sistemas nos anos 80, a metodologia de ligação ponto-a-ponto não conseguiu competir com o crescimento dos sistemas eletrónicos. Dando lugar a sistemas interativos onde a informação é partilhada através de um barramento comum. Uma vez que estas ligações aumentaram a necessidade de interfaces série e de um protocolo de barramento aberto, estes sistemas de comunicação adquiriram bastante importância nesta época.

Desta forma, o modelo de *Controller Area Network* (CAN) desenvolvido por *Robert Bosh* na Alemanha em 1980, para a aplicação na indústria automóvel, com o objetivo de simplificar os complexos sistemas de fios em veículos com sistemas de controlo composto por inúmeros microcontroladores para a gestão dos diversos sistemas do veículo. Em 1996, *Robert Bosh* apresentou a versão 1.0 deste protocolo. Devido a este ter um grande potencial, tornou este produto num protocolo mais flexível, o que originou a publicação, em 1991, da versão 2.0. Durante a década de 90 este protocolo foi introduzido como standart pela *International Standard Organization* (ISO), através do ISO 11898 [16].

Devido às excelentes características, o protocolo CAN tem vindo a tornar-se cada vez mais popular. Ao longo dos anos, o CAN evoluiu de aplicações dedicadas à indústria automóvel para outras áreas de uso industrial e produtos que envolvem microcontroladores. É considerado uma solução para implementar comunicações em rede de uma forma simples mas robusta, fiável e de baixo custo.

Esta comunicação é baseada na topologia de barramento, onde apenas são utilizados dois fios, CAN_H (CAN *High*) e CAN_L (CAN *Low*) para fazer a transmissão de dados nesta rede. Apesar de este protocolo ser caraterizado pela sua robustez devido ao facto de ser planeado para ambientes com grande nível de ruído e grandes oscilações de alimentação, permite a incorporação de um terceiro fio. A utilização deste fio que estará ligado ao GND tem o objetivo de reduzir os efeitos eletromagnéticos que possam surgir no barramento. Como os fios CAN_H e CAN_L são um par entrançado, a indução de um ruído que eleve o

nível de tensão em um dos fios, também afeta o outro pelo mesmo nível de tensão. Os dados enviados pelo barramento são interpretados pela diferença de potencial entre os dois condutores CAN_H e CAN_L. O barramento tem uma estrutura onde cada dispositivo ligado pode enviar ou receber dados (Figura 17).

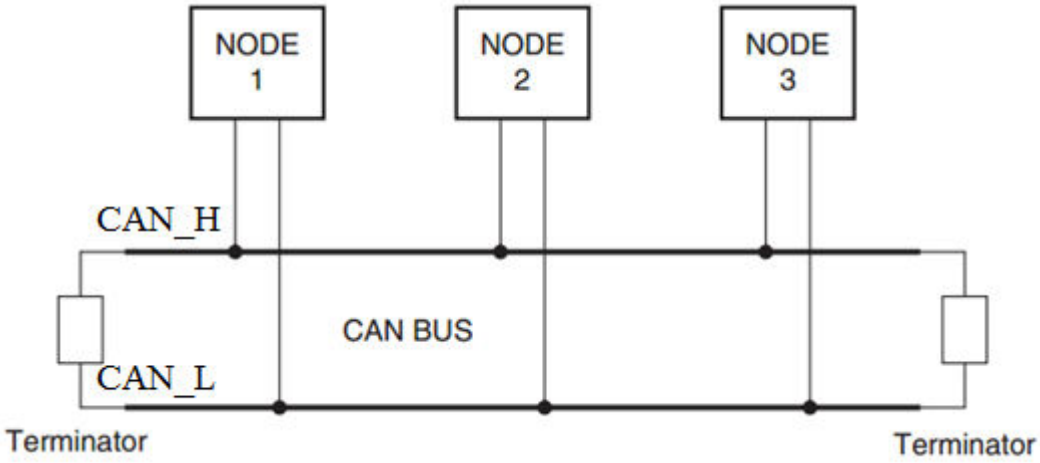

Figura 17 - Barramento CAN

O protocolo CAN é baseado no protocolo CSMA/CDþAMP (*Carrier-Sense Multiple Access/Collision Detection with Arbitration on Message Priority*), que é similar ao protocolo usado nas redes Ethernet. Este método de arbitragem tem a função de evitar colisões de dados durante o decorrer das transmissões entre os dispositivos ligados no barramento.

A ISO-11898 especifica que um dispositivo no barramento deve ser capaz de comunicar com outro nó a uma distância de 40 metros a uma taxa de transmissão de 1Mb/s. Ao baixar a velocidade de transmissão consegue-se distâncias maiores, como mostra a figura seguinte.

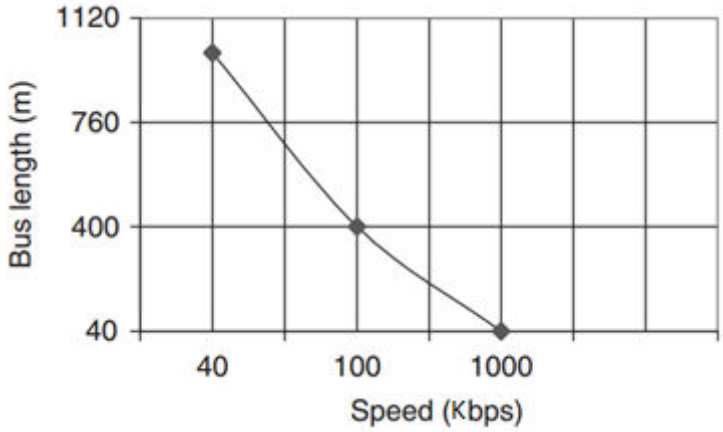

Figura 18 - Distância versus velocidade no barramento CAN [16]

Em cada extremidade do barramento CAN, o nível alto (CAN *High*) e o nível baixo (CAN *Low*) são ligados por uma carga resistiva de modo a minimizar os sinais de reflexão no barramento. A ISO-11989 exige que o barramento tenha uma impedância característica de 120 Ohm. O terminal pode ser constituído através dos seguintes circuitos (Figura 19):

- Terminal padrão;
- Terminal *split*;
- Terminal *biased split*.

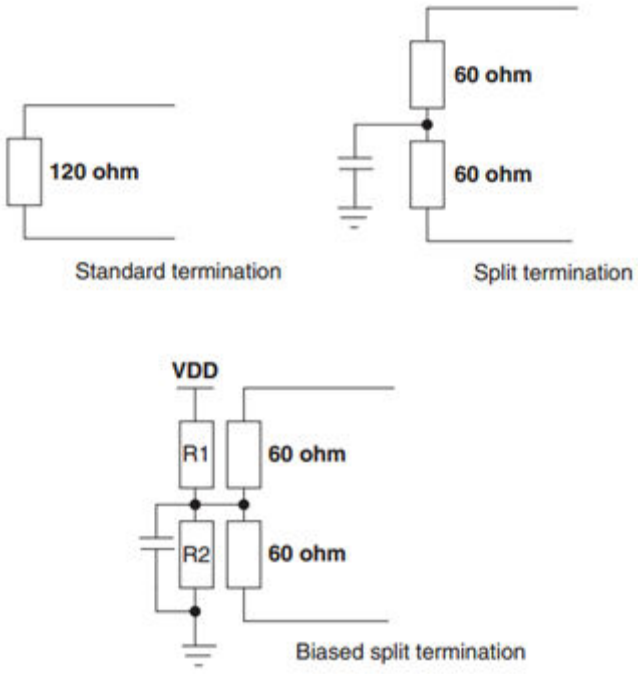

Figura 19 - Terminais possíveis para o barramento CAN

O método padrão é o mais comum (o eleito para a aplicação neste projeto), utiliza uma resistência de 120 Ohm nas extremidades do barramento. No segundo circuito, com um terminal *split*, as extremidades são divididas por uma resistência de 60 Ohm, permitindo a redução de emissões de ruído. Por fim, o terminal *Biased split* é similar à configuração em *split* mas com o acrescento de um divisor de tensão e um condensador. Estes métodos melhoram significativamente o desempenho do barramento.

As mensagens que circulam no barramento CAN são designadas de *frames*. Estas podem ser de dois formatos: *Standard* (11-bits ID) ou *Extended* (29-bits ID). Quanto menor o valor deste campo, maior é a prioridade da mensagem. A *frame* de dados é usada no dispositivo de transmissão para o de receção contendo a informação que o utilizador solicitou. A *frame* é iniciada com o bit SOF (*Start Of Frame*), a qual é seguida pelo bit identificador e transmissão remota RTR (*Remote Transmission Request*). O identificador e o RTR são constituídos por 12 bit e o campo de controlo é de 6 bit e indica quantos dados estão no campo de dados. O campo de dados pode ser preenchido até 8 bytes. Na Figura 20 ilustra-se a estrutura da mensagem.

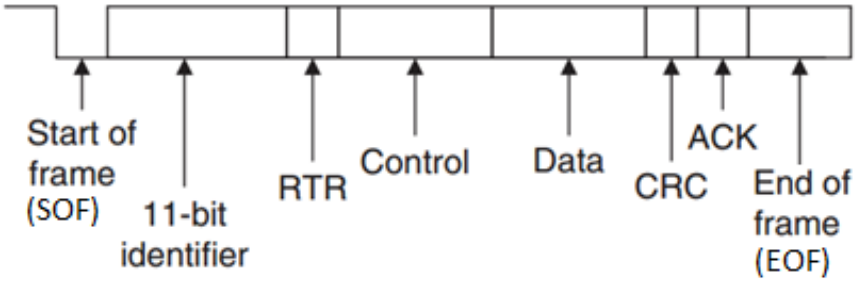

Figura 20 – Mensagem *Frame* CAN

O CRC (*Cyclic Redundancy Check*) verifica se a correspondência de bits está corrompido. O campo ACK é de 2 bit e é utilizado pelo transmissor para receber a confirmação de uma *frame* válida a partir de qualquer recetor. O final da mensagem é indicado pelos últimos 7 bit EOF.

Na *frame* extensa, o campo de arbitragem é de 32 bit: 29 bit do identificador, 1 bit IDE para definir a mensagem como *frame* extensa, 1 bit SRR que não é utilizado e mais 1 bit RTR.

4.1.1.1 Interface PIC – CAN

Na maioria dos casos, qualquer tipo de microcontrolador pode usar o barramento CAN, mas só alguns têm integrado o módulo CAN, como o PIC18F258, simplificando o hardware. Microcontroladores que não possuem este módulo também podem ser utilizados, adicionando hardware e software correspondente, mas aumentando a complexidade do sistema, bem como o custo.
A Figura 21 representa a ligação de um microcontrolador a um nó do barramento CAN. Como o PIC18F258 contém internamente o módulo controlador CAN, apenas é necessário inserir o CAN *transceiver*.

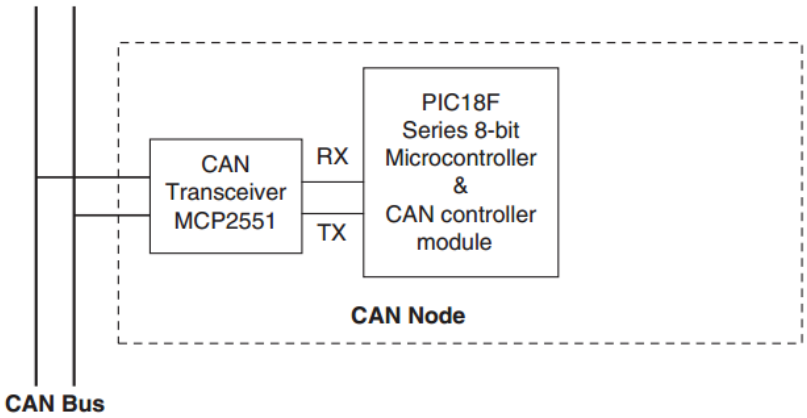

Figura 21 - Ligação PIC18F com o *transceiver* MCP2551

O módulo CAN do microcontrolador utiliza os pinos RB3/CANRX e RB2/CANTX para a receção e respetiva transmissão. Estes pinos são ligados ao barramento através do chip *transceiver* MCP2551.

Como já foi referido, os microcontroladores PIC18F258 e PIC18F458 têm incorporado o módulo CAN, mas para poder usar esta comunicação é necessário aceder aos registos certos. Assim, a *Microchip* disponibiliza o software *Application Maestro* que cria código dependendo dos parâmetros introduzidos nesta aplicação (Figura 22). Pelo que, para receber ou enviar mensagens no barramento CAN, basta chamar as funções que estão alojadas dentro da biblioteca criada.

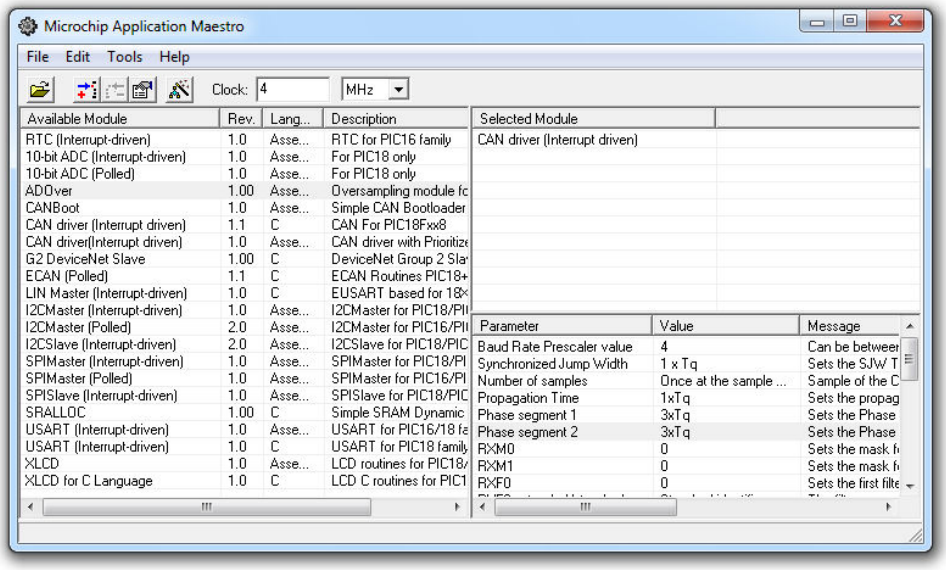

Figura 22 - Programa Microchip *Application Maestro*

Para implementar o barramento CAN com uma velocidade de transmissão de 125kbps e com um oscilador de 4Mhz nos microcontroladores (utilizados neste projeto), foi imprescindível utilizar o software *Microchip CAN bit Timing Calculator* da *Intrepid Control Systems* (Figura 23). Este devolve os valores dos parâmetros *Baud Rate Prescaler*, o *Propagation Time*, *Phase Segment* 1, *Phase Segment* 2 e do *Synchronized Jump Width* conforme se ilustra na Figura 24.

Figura 23 - Programa *Microchip CAN bit Timing Calculator*

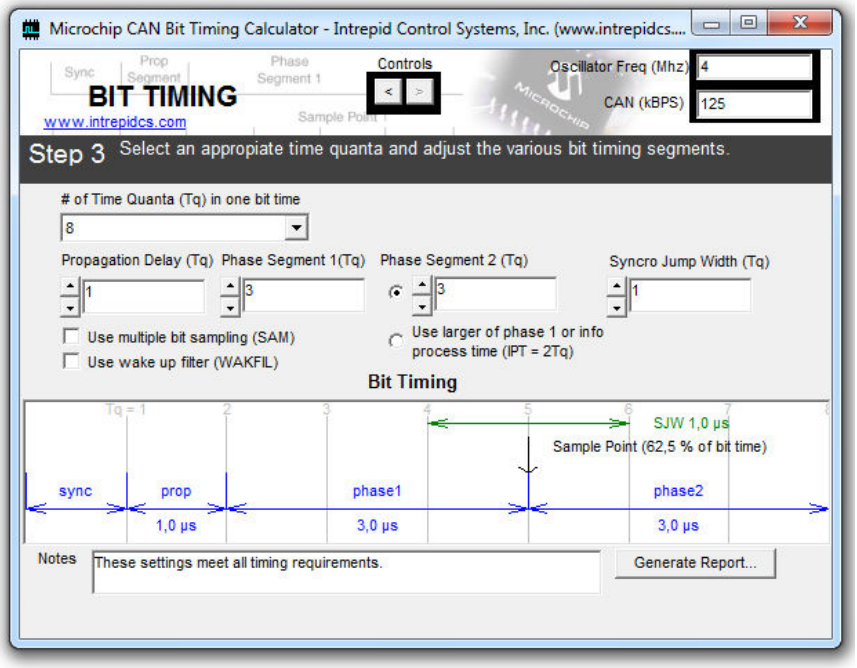

Figura 24 - Parâmetros usados no *Microchip CAN bit Timing Calculator*

Como se pode ver na Figura 24, com uma frequência de relógio de 4MHz e um *baud rate* de 125kbps, obtêm-se os parâmetros apresentados na Tabela 2.

Tabela 2 - Parâmetros do CAN

4 Mhz / 125kbps	
Baud Rate Prescaler (BRP-1)	1
Propagation Delay (Prop)	1 (µs)
Phase Segment 1 (Phase 1)	3 (µs)
Phase Segment 2 (Phase 2)	3 (µs)
Syncronization Jump Width (SJW)	1 (µs)
Error of target Baud Rate	0,0%

Para uma temporização correta é exigido que os parâmetros anteriores correspondam às seguintes condições:

- Prop_Seg + Phase_Seg1 >= Phase_Seg2;
- Phase_Seg2 >= SJW;

Como se pode ver na Tabela 2, estas condições são cumpridas.

Outros dos parâmetros a serem fornecidos ao *Application Maestro*, são os valores para as mascaras e filtros. Para que cada microcontrolador apenas aceite as mensagens direcionadas a este, foi necessário utilizar na máscara RXM0 e RXM1, o valor 536870911 ($2^{29} - 1$). Todos os microcontroladores foram configurados desta forma, diminuindo o esforço de cada PIC cuja mensagem não seja direcionada a este. O valor dos endereços de cada módulo inserido no barramento, estão ilustrados na Tabela 3.

Tabela 3 – Endereço dos módulos no barramento CAN

Dispositivos CAN	**Endereço CAN**
USB - CAN	1
Módulo de deteção de gás	2
Módulo de temperatura	3
Módulo de iluminação	4

4.1.2 Interface PC-CAN

Nesta secção descrevem-se as duas etapas do trabalho desenvolvido para a interface que liga o barramento CAN ao computador. A primeira implementação consiste em utilizar a comunicação série entre o microcontrolador e o computador, utilizando um conversor (USB-RS232). A segunda implementação usa o protocolo USB como meio de comunicação entre o computador e o PIC18F2550, que por sua vez está ligado ao PIC18F258 através do USART.

4.1.2.1 Primeira Fase (RS232-PC)

De referir que um dos objetivos deste projeto é a realização de uma aplicação para computador que interaja com os sistemas desenvolvidos.
Após ter definido o CAN como protocolo de comunicação para a rede de domótica, foi necessário implementar uma placa para a ligação do microcontrolador ao PC. A escolha da

comunicação dependeu do microcontrolador e como o PIC18F258 suporta USART, seria a hipótese mais adequada para o fazer.

Para fazer a ligação dos pinos RC6 (transmissor) e RC7 (recetor) do microcontrolador, com a porta série do computador, é necessário introduzir um elemento que converta os sinais para TTL e vice-versa. Este elemento é designado de MAX232 e consiste num integrado que tem essa função (Figura 25).

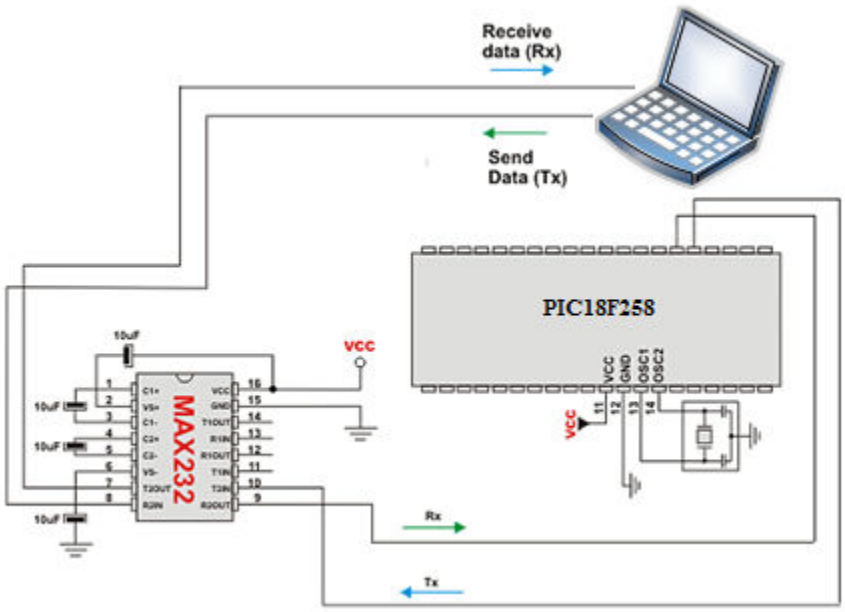

Figura 25 - Esquema de ligação do MAX232

Como o computador usado neste projeto não possui porta série, foi necessário utilizar um conversor Série para USB igual ao da Figura 26.

Figura 26 - Conversor USB - Série

O módulo USART é um dos 3 módulos de comunicação série que vem incorporado no PIC18F258 e PIC18F458. O USART pode ser configurado em regime *full-duplex* de forma assíncrona, pode comunicar com dispositivos periféricos, tais como terminais CRT (*Cathode Ray Tube*) e computadores pessoais. Pode também ser configurado como *half-duplex* de forma síncrono e pode comunicar com dispositivos periféricos do tipo integrados A/D ou D/A, EEPROMs séries, etc.

Para usar a comunicação USART no PIC18F258 é necessário efetuar algumas configurações a nível de registos no microcontrolador, como se descreve a seguir.

A- TXSTA

O registo TXSTA é responsável pelo estado da transmissão, sendo um registo de controlo, Figura 27.

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

Figura 27 - Registo TXSTA

A lista seguinte apresenta os bits deste registo com os valores correspondentes:

- Bit 7: TXSTAbits.CSRC = 0; - Indiferente em modo assíncrono;
- Bit 6: TXSTAbits.TX9 = 0; -Transmissões de 8 bits;
- Bit 5: TXSTAbits.TXEN = 1; - Habilitada a transmissão;
- Bit 4: TXSTAbits.SYNC = 0; - Modo assíncrono;
- Bit 2: TXSTAbits.BRGH = 1; - Alta velocidade.

B- *Baud Rate*

Uma das mais importantes configurações é a taxa de transmissão, designada por *Baud Rate*. Para conseguir uma boa comunicação entre os dois dispositivos, este parâmetro deverá ser o mais próximo para ambos. O registo que trata deste parâmetro é o registo SPBRG.

No lado do dispositivo PIC, a *Buad Rate* pode tomar diversos valores. Para calcular este parâmetro é necessário usar a seguinte fórmula (para BRGH igual a 1), disponibilizada no *datasheet* do mesmo [11].

$$Baud\ Rate\ = \frac{\mathrm{Fosc}}{16\ .\ (\mathrm{X}+1)} \quad (1)$$

Em que: *Fosc* é a frequência de relógio, sendo igual a 4MHz;

X é o valor que será inserido no registo.

Como esta comunicação é para ser utilizada com um computador que contém taxas de transmissão *standard*, é necessário calcular uma *Baud Rate* no microcontrolador que se aproxime o mais possível à taxa de transmissão do computador (Tabela 4).

Tabela 4 – Valores de *Baud Rate* normalizados

Baud Rate normalizados (bps)
2400
4800
9600
19200
38400
57600
115200

Para obter um *Baud Rate* de aproximadamente de 9600 bps (taxa de transmissão escolhida) é necessário um SPBRG de 25, como se determina abaixo.

$$Baud\ Rate\ = \frac{4\mathrm{Mhz}}{16\ .\ (25+1)} = 9615.39\ \mathrm{bps} \quad (2)$$

O erro obtido nesta situação é dado por (3) e representa o erro associado por utilizar um oscilador de 4MHz com um SPBRG de 25.

$$Erro = \frac{\text{Baud Rate Calculado} - \text{Baud Rate Desejado}}{\text{Baud Rate Desejado}} = \left(\frac{9615{,}39-9600}{9600}\right) * 100 = 0.160\% \quad (3)$$

Com um erro de aproximadamente 0,16%, significa que para uma comunicação série com esta taxa de transmissão (9600 bps) o erro associado a falhas é quase nulo.

C- RCSTA

Outro registo que carece de configuração é o RCSTA (Figura 28). Este registo é responsável pela receção dos dados do USART.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Figura 28 - Registo RCSTA

De seguida é apresentada uma lista de cada bit deste registo com os valores especificados:

- Bit 7: RCSTAbits.SPEN = 1; - Serial port enable;
- Bit 6: RCSTAbits.RX9 = 0; - Transmissões de 8 bits;
- Bit 5: RCSTAbits.SREN = 0; - Indiferente no modo assíncrono;
- Bit 4: RCSTAbits.CREN = 0; - Receção contínua;
- Bit 3: RCSTAbits.ADDEN = 0; - Endereço detetado;
- Bit 2: RCSTAbits.FERR = 0; - *No Framing Error;*
- Bit 1: RCSTAbits.OERR = 0; - *No overrun error;*
- Bit 0: RCSTAbits.RX9D = 0; - 9º bit de receção.

Ainda na configuração do módulo USART, os pinos RX e TX do microcontrolador têm que ser configurados como entrada e saída, respetivamente. Assim, o pino 7 do porto C (RC7) tem de ser colocado a 1 e TRISC6 tem de ser colocado a 0 que é o bit de transmissão de dados (RC6), Figura 29.

```
//PORTC Bit 6 - TX
TRISCbits.RC6 = 0;
//PORTC Bit 7 - RX
TRISCbits.RC7 = 1;
```

Figura 29 - Configuração pinos RX e TX do USART

Por fim, é necessário "chamar" a função *OpenUSART* e configurar alguns parâmetros. A interrupção de transmissão (USART_TX_INT_OFF) foi desligada, ao contrário da receção de dados (USART_RX_INT_ON). O modo foi definido como assíncrono (USART_ASYNCH_MODE), a largura de banda é de 8-bit (USART_EIGHT_BIT), o modo de receção é contínuo (USART_CONT_RX), o baud rate é referido como alto (USART_BRGH_HIGH) e o SPBRG é de 25 para este microcontrolador, Figura 30.

```
//Configuração do módulo USART
OpenUSART( USART_TX_INT_OFF &
USART_RX_INT_ON &
USART_ASYNCH_MODE &
USART_EIGHT_BIT &
USART_CONT_RX &
USART_BRGH_HIGH,
25);
```

Figura 30 - Configuração da função OpenUSART

Para testar esta comunicação desenvolvi um pequeno programa de testes (em Visual Basic), onde o evento *SerialPort_DataReceived* ficaria à "escuta" na porta série. Como se trata de uma comunicação série, só é possível o envio de 1 byte de cada vez. Assim sendo, o microcontrolador enviava os bytes de forma sequencial através de uma temporização definida (50 milissegundos).

Dado que a comunicação começou a apresentar alguma instabilidade para temporizações desta ordem do envio dos dados, concluí que esta abordagem não poderia ser adotada desta maneira.

Uma vez que esta hipótese de solução não cumpre os requisitos da aplicação em termos de tempos de comunicação, foi necessário ultrapassar esta limitação cuja explicação se apresenta na subsecção seguinte.

4.1.2.2 Segunda Fase: PC-USB

Para não abdicar do trabalho já desenvolvido, foi dedicado algum tempo de pesquisa e uma das soluções encontradas, para a limitação referida na fase anterior é a utilização do microcontrolador PIC18F2550 que contém o módulo USB 2.0 integrado. Para usar esta comunicação é utilizado uma biblioteca em C e uma classe em C# existentes em [17] que utiliza o microcontrolador PIC18F4550 para a interface.

Após uma avaliação cuidada, verifiquei que tinha grande potencialidade para a aplicação que estava a desenvolver, visto que este microcontrolador também possibilita a comunicação USART. O passo seguinte consistiu na análise ao *firmware* disponível e na sua adaptação para um PIC18F2550 (que também tem o módulo USB internamente, mas com a vantagem de ter um tamanho mais reduzido).

Outra das modificações efetuadas foi a configuração dos registos para poder usar a comunicação USART e a respetiva interrupção, para conseguir comunicar com o PIC18F258, como se mostra no diagrama da Figura 31.

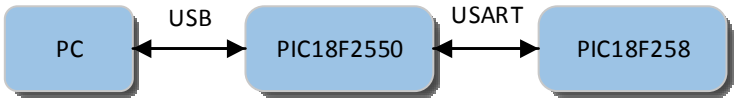

Figura 31 – Diagrama de comunicação da interface PC-CAN

Assim, a matéria descrita na primeira fase desta sessão 4.1.2 passa a ser apropriada para a comunicação entre os dois microcontroladores.

Esta metodologia, apesar de aparentemente parecer pouco ortodoxa, apresenta uma estrutura que funciona bastante bem e que cumpre com os requisitos deste projeto.

Para a comunicação entre o PIC18F2550 e computador através do USB é utilizada uma biblioteca designada de "*USBFunctions.c*" [17], que contém funções em *C* para ler e escrever a informação no USB. Esta biblioteca contém funções que permite a comunicação com o sistema operativo Windows que reconhece o microcontrolador como um dispositivo HID (*Human Interface Device*). Assim, para enviar dados para o USB é usada a função designada de *sendViaUSB*, que faz o envio de um *arrray* de 65 caracteres. Para especificar cada caractere desse *array* é utilizada a estrutura *g_fromDeviceToHostBuffer[n]*. No lado

da receção de dados é chamada a função *receiveViaUSB* e cada caractere está especificado na estrutura *g_fromHostToDeviceBuffer[n]*, em que "n" é o número do byte do *array*.
A Figura 32 apresenta o fluxo referente ao processo neste microcontrolador.

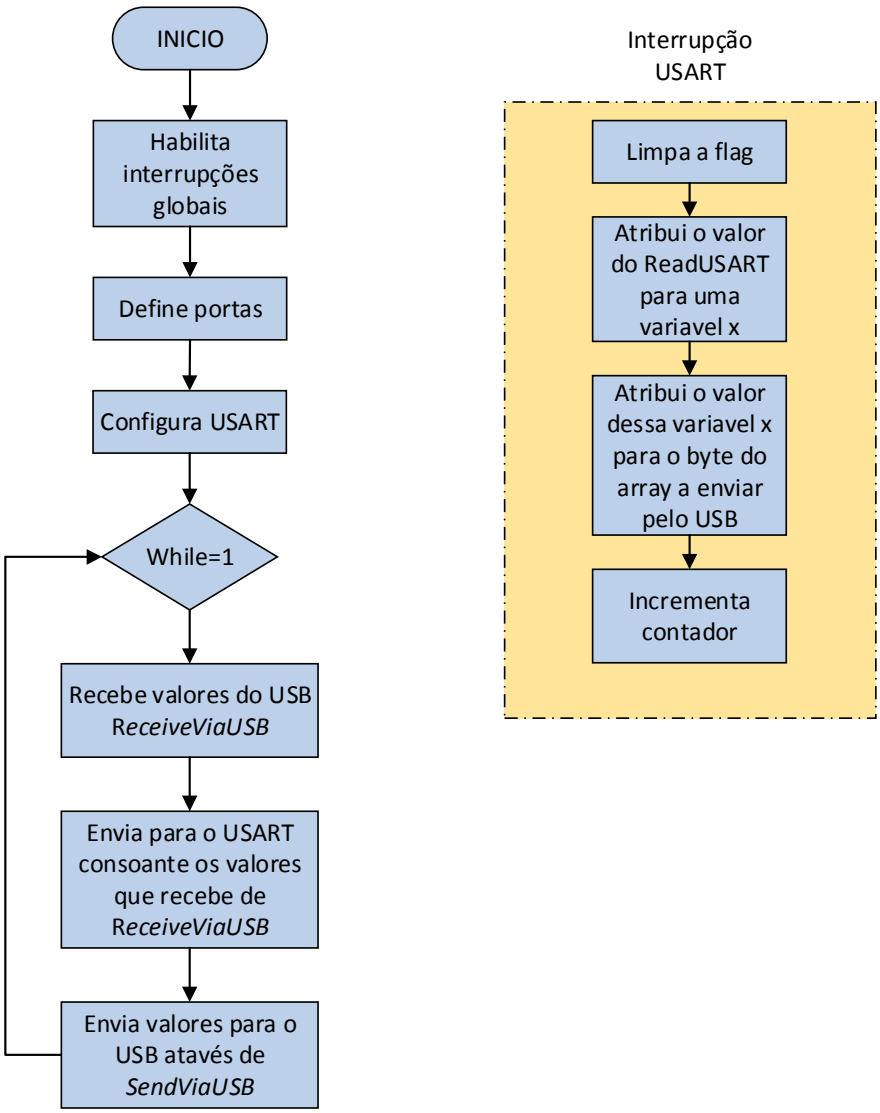

Figura 32 – Diagrama do PIC18F2550

Para a comunicação funcionar, as funções *sendViaUSB e receiveViaUSB* são chamadas no ciclo *while* do microcontrolador. Sempre que recebe dados atualizados provenientes da função *receiveViaUSB* é enviada essa informação para o PIC18F258.

Para este microcontrolador receber informação do PIC18F258, foi necessário configurar a interrupção USART para que sempre que recebe dados desta comunicação seja ativada a *flag* desta interrupção e guardado o valor dessa receção.

O microcontrolador PIC18F258 que está inserido na mesma placa (como se apresenta na Figura 38) é o microcontrolador *Master* do barramento CAN. Este PIC tem a função de "pedir" aos restantes dispositivos ligados ao barramento, designados de *slaves*, os valores atualizados dos seus estados. Para isso, é utilizada a função *CANPut* que recebe como argumento a mensagem a ser enviada para o *buffer* (TXBUF). Esta estrutura (Figura 33) contém o endereço associado à mensagem (*TX_Message.Address*), se a mensagem é do tipo extensa (*TX_Message.Ext*), o número de bytes da mensagem (*TX_Message.NoOfBytes*), o array de bytes (*TX_Message.Data*), a prioridade (*TX_Message.Priority*), e se é uma *frame* remota (*TX_Message.Remote*). Para o PIC *Master* receber as mensagens vindas dos *slaves*, é utilizada a função *CANRXMessageIsPending* que verifica a existência de alguma mensagem no *buffer* de receção (RXBUF). Caso exista, é chamada a função *CANGet* que devolve a mensagem. Todo este processo é utilizado nos restantes microcontroladores.

```
struct CANMessage {
       unsigned long Address;
       unsigned char Data[8];
       unsigned char NoOfBytes;
       unsigned char Priority;
       unsigned Ext:1;
       unsigned Remote:1;
};
```

Figura 33 - Estrutura CAN *Message*

Como um dos objetivos do PIC *Master* é recolher toda a informação dos estados dos sensores e variáveis que os *slaves* dispõem, é necessário configurar o *Timer0* do PIC para que a cada 50 milissegundos (temporização escolhida) envie uma mensagem a cada um dos *slaves* (Figura 34). Após o PIC *Master* receber a mensagem do *slave* requisitado, é enviada essa informação para o PIC18F2550 através da comunicação USART. Para isso, é enviada uma sequência de bytes com toda a informação do PIC *slave* em causa.

```
// Timer
OpenTimer0(TIMER_INT_ON & T0_16BIT
& T0_SOURCE_INT & T0_PS_1_1);
WriteTimer0(15535);
```

Figura 34 - Configuração do Timer0

Como a comunicação se processa através do envio sequencial de dados, é necessário enviar um caractere "#" que indica o início do envio, de seguida é enviado o ID do PIC *slave* para o PIC18F2550 identificar qual a origem da mensagem. É também enviado o número de bytes que a sequência tem, seguindo os bytes referentes aos dados e o fecho da mensagem é feita com um caractere "$", conforme se indica na Figura 35.

Para que o PIC *Master* possa gerir toda a informação requisitada pelo utilizador (vinda do PIC18F2550), é necessário que na mesma mensagem direcionada aos *slaves* sejam enviados os pedidos da aplicação.

```
[#][PIC_ID][nº de bytes][bytes][$]
```

Figura 35 - Formato de envio para o USART

O próximo diagrama ilustra o processo do microcontrolador *Master* do barramento CAN.

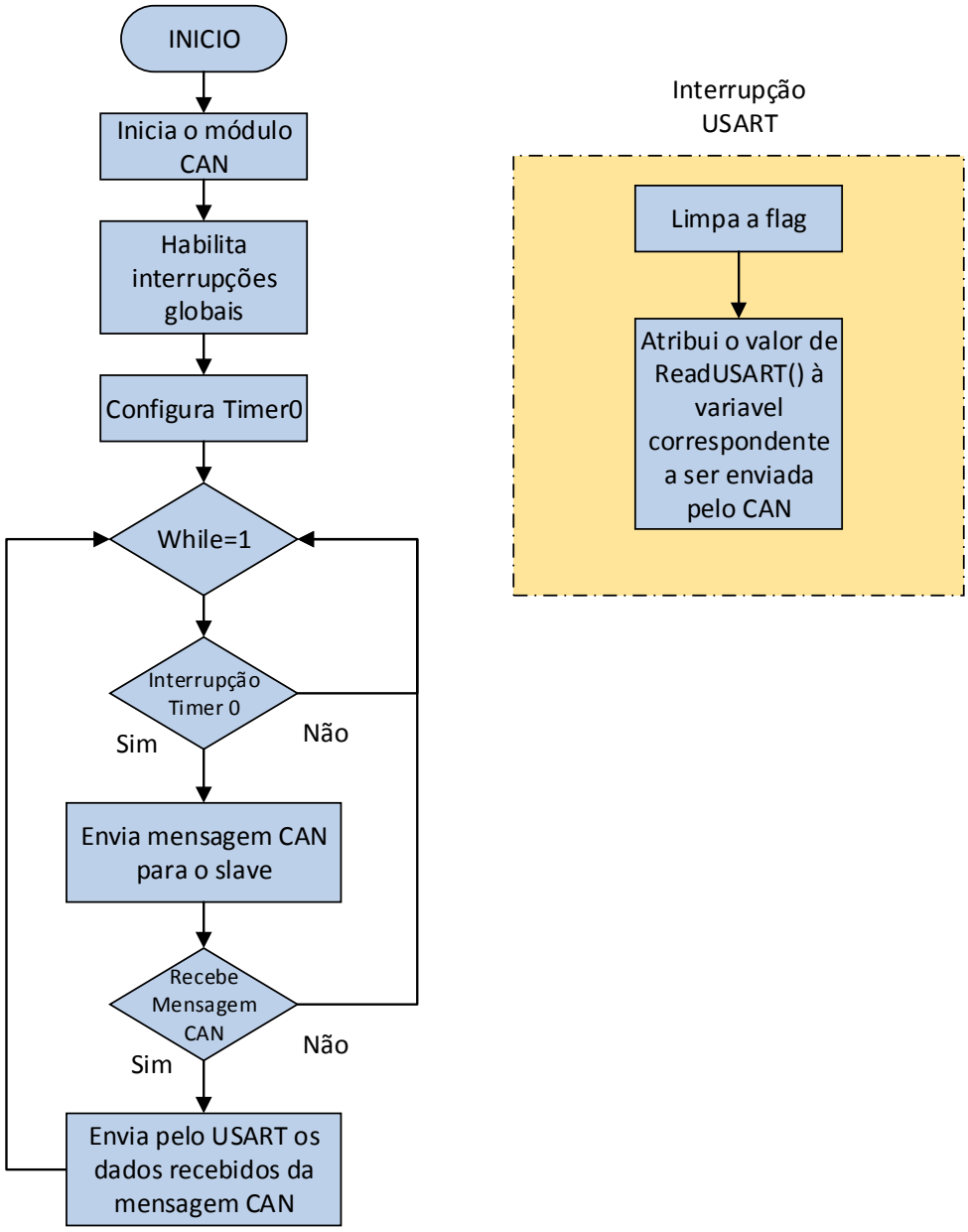

Figura 36 – Diagrama do PIC *Master*

A Figura 37 ilustra o esquema da placa PC-CAN composta pelo PIC18F258 que está ligado ao barramento CAN através do *transceiver* MCP2550. Por sua vez este microcontrolador está ligado pela comunicação USART ao PIC18F2550 que contém a ligação para o USB.

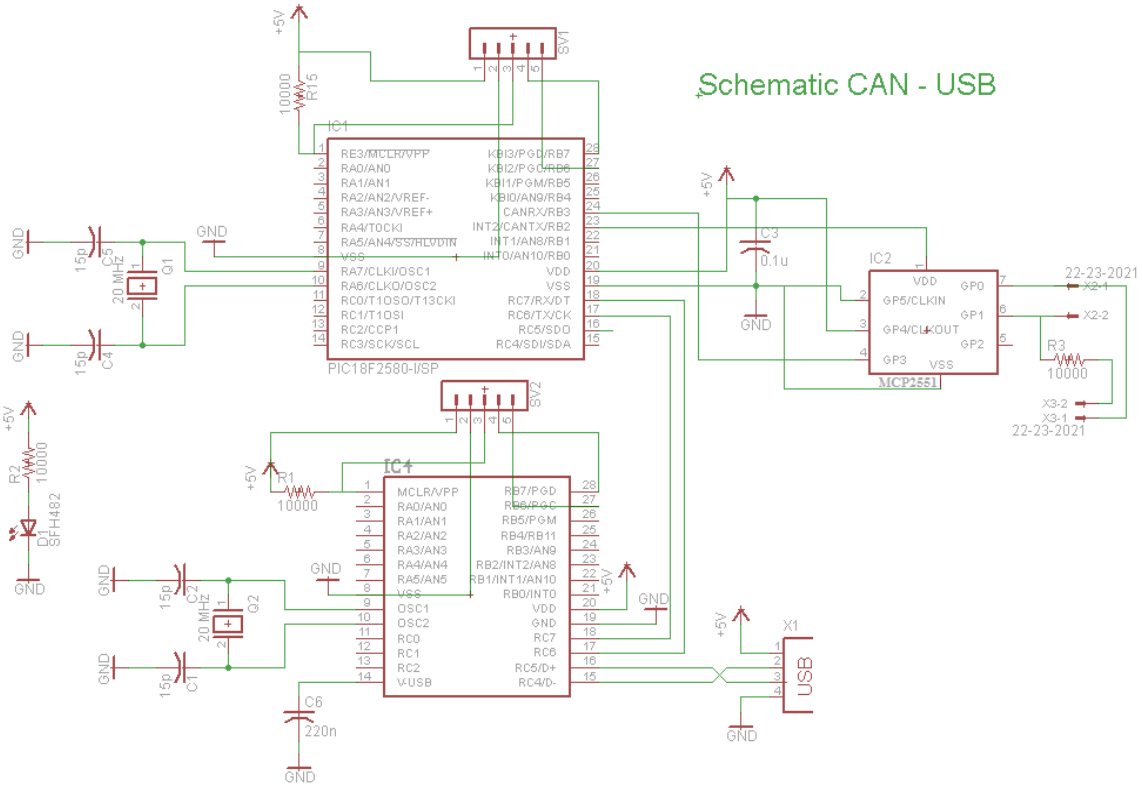

Figura 37 - Esquema da placa PC-CAN em *Eagle*

A Figura 38 apresenta a fotografia da placa PC-CAN desenvolvida em circuito impresso. Os dois microcontroladores foram colocados lado a lado, de forma reduzir o tamanho da placa e com orientações opostas para que os pinos da comunicação USART ficassem o mais perto possível de ambos. O *transceiver* também foi posicionado de maneira a ficar próximo da ligação ao barramento CAN. No lado oposto é visível a ligação USB através de um conetor do tipo B.

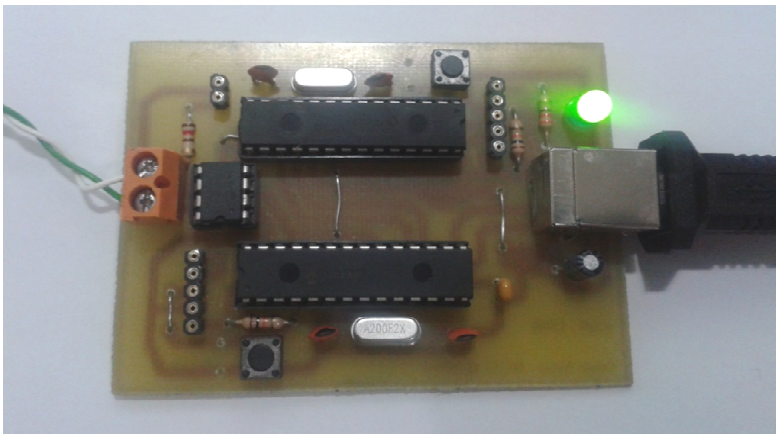

Figura 38 - Placa PCB PC-CAN

4.2 Módulos Desenvolvidos

Nesta secção faz-se a descrição dos módulos (temperatura, iluminação e deteção de gás) desenvolvidos neste projeto com ligação ao barramento CAN.

4.2.1 Módulo de Temperatura

O módulo de temperatura utiliza um sensor de precisão, designado de LM35 (Figura 39) fabricado pela *National Semicondutor*. Este sensor apresenta uma saída analógica linearmente proporcional à temperatura do local em que se encontra inserido. Pode ser alimentado com uma tensão entre os +4V e os +20V, tem um consumo de apenas 60μA e a sua precisão é de 10mV por cada grau centígrado. Outra das características deste sensor é o facto de fornecer uma precisão típica de ¼ºC dentro da faixa de temperatura entre os -55ºC e os +150ºC. Todos as suas características tornam este sensor ideal para este tipo de aplicação.

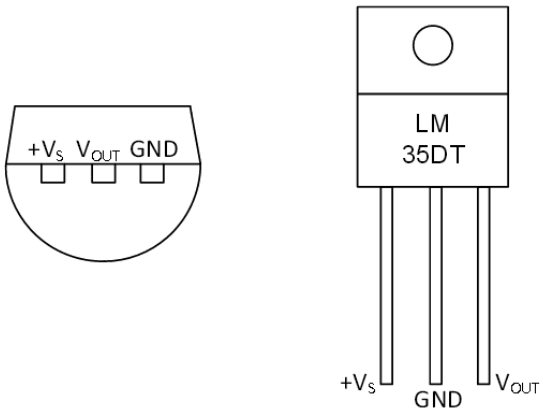

Figura 39 - Sensor de temperatura LM35

O esquema da Figura 40 corresponde ao circuito elaborado para este módulo de temperatura. Foi introduzido um microcontrolador PIC18F258 para fazer a leitura da tensão de saída do LM35, através do conversor analógico digital, de forma a enviar o valor da temperatura através do barramento CAN.

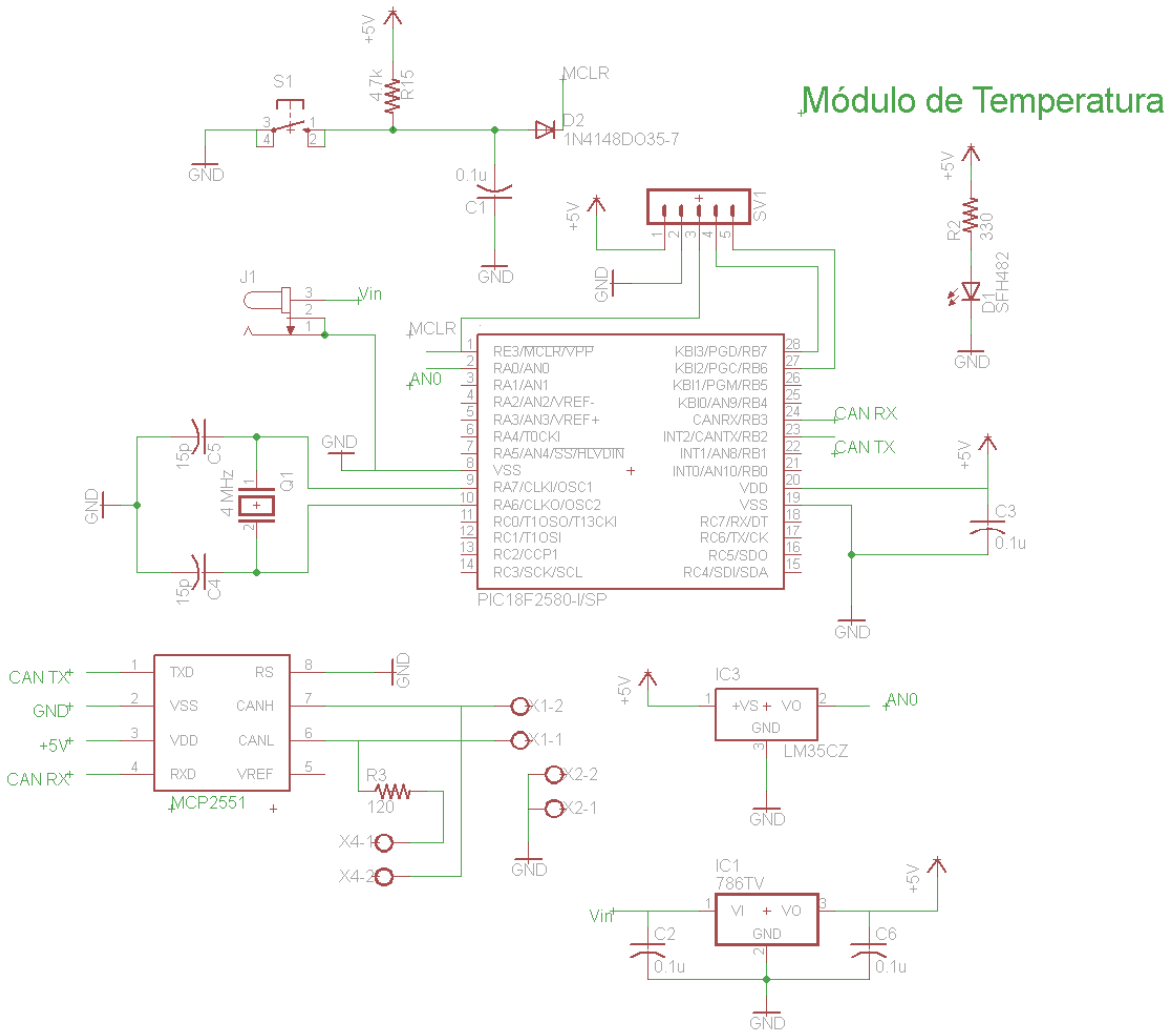

Figura 40 - Esquema em *Eagle* do módulo de temperatura

Como se pode verificar na Figura 41, o sensor de temperatura foi estrategicamente colocado, de modo a ficar o mais afastado possível do circuito de alimentação. Desta forma, elimina-se quaisquer perturbações na medição do LM35 devido ao aquecimento proveniente do regulador de tensão IC LM7805. Este módulo é alimentado por um transformador de +12V, e o regulador garante que o circuito permanece com uma tensão de +5V (valor ideal para alimentar os componentes).

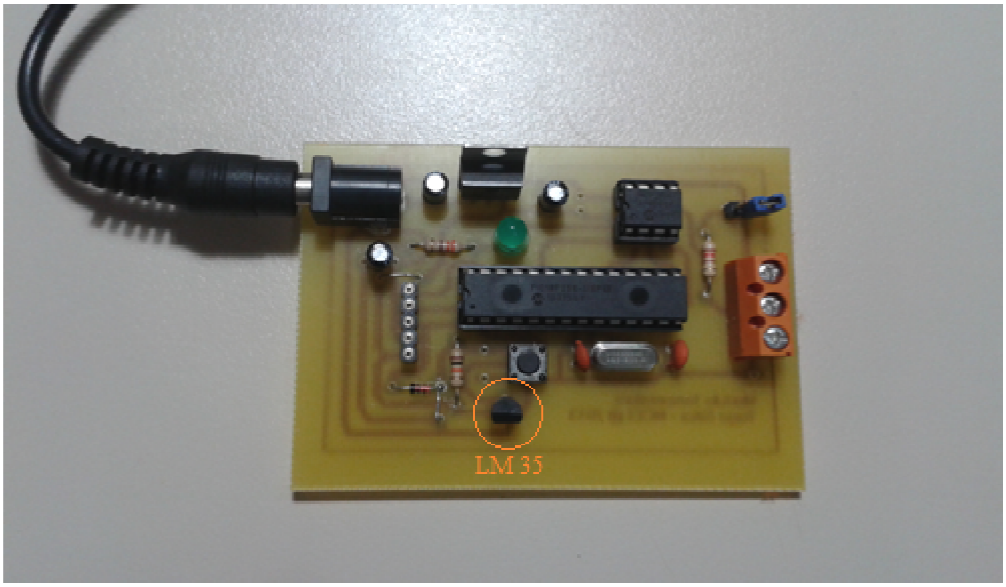

Figura 41 – Placa em PCB do módulo de temperatura

Para o microcontrolador conseguir "ler" um valor analógico, associado a uma das suas portas, foi necessário configurar esta entrada. Para tal, configurei o pino RA0 para entrada analógica, da seguinte forma (Figura 42).

```
ADC(ADC_FOSC_8
      & ADC_RIGHT_JUST
      & ADC_1ANA_0REF, ADC_CH0
      & ADC_INT_OFF);
```

Figura 42 - Configuração do ADC

De maneira a ter sempre atualizado o valor associado da porta analógica, é necessário que a cada período de tempo o microcontrolador faça essa leitura. Para criar esta interrupção temporizada, o *Timer0* está configurado para que a cada 50 milissegundos execute a interrupção, de modo a permitir ao PIC a leitura do valor de tensão proveniente do sensor de temperatura.

Para converter o valor lido do ADC em graus Celcius, procede-se como indicado na Figura 43.

```
temp_LM35 = 5.0 * valor_adc * 100.0/1023.0; //graus Cº
```

Na Figura 44 apresenta-se o fluxograma simplificado do microcontrolador inserido neste módulo.

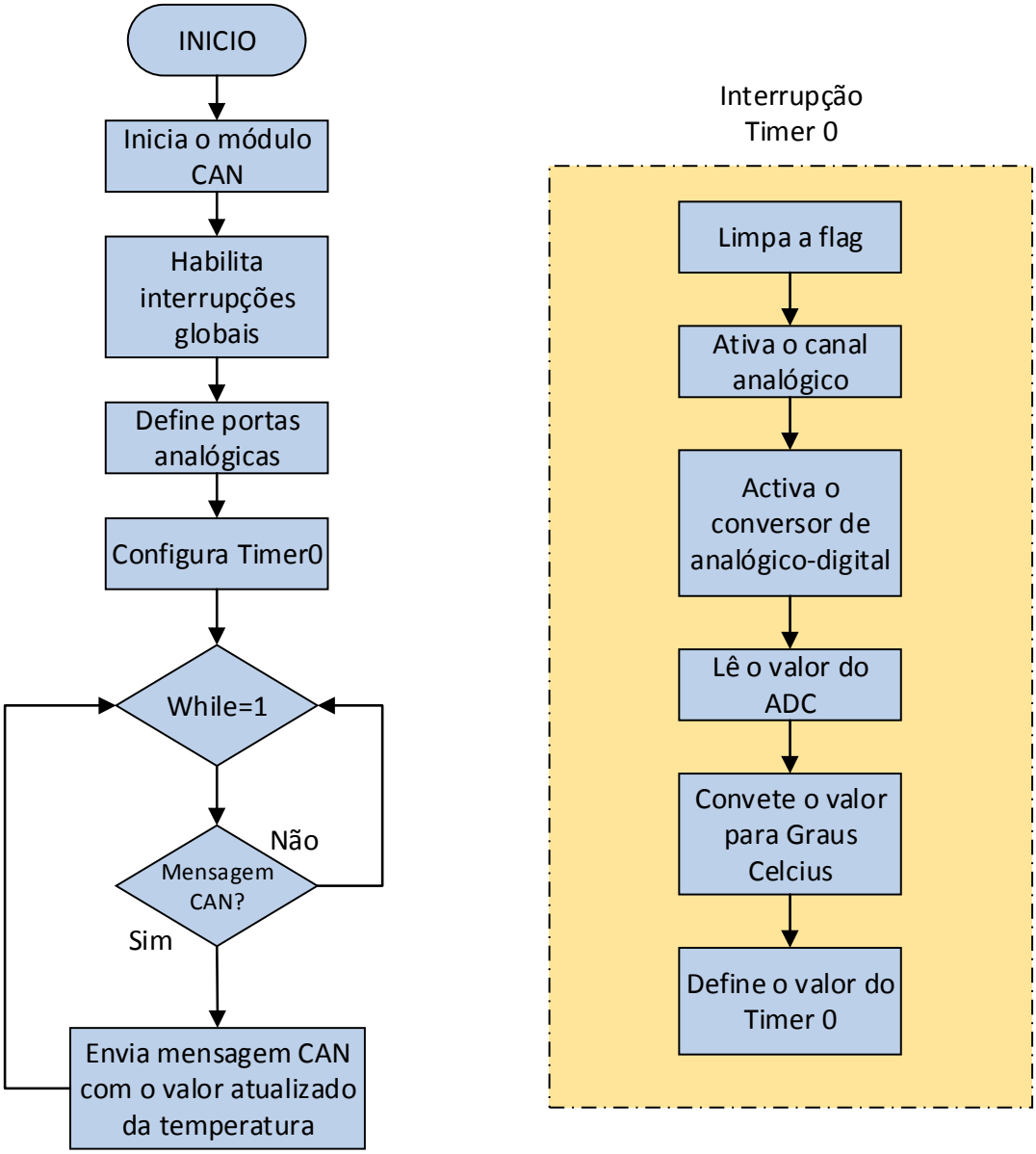

Figura 44 - Diagrama do módulo de temperatura

4.2.2 Módulo de Iluminação

O módulo de Iluminação é o que requer maior destaque. Para implementar este módulo foi utilizado um sistema de iluminação desenvolvido anteriormente em duas unidades curriculares do mestrado. Este sistema consiste em fazer o controlo automático ou o

comando manual do nível de iluminação de um protótipo para uma habitação. A estrutura é constituída por duas divisões que estão separadas por uma janela onde está inserida uma persiana, como se mostra na Figura 45.

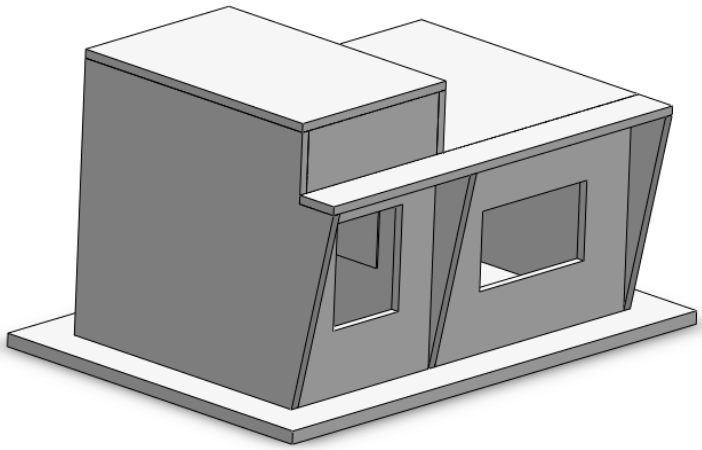

Figura 45 - Módulo Iluminação em *SolidWorks*

Na divisão do lado esquerdo está inserida uma lâmpada do tipo incandescente que pode ser regulada através da alteração da tensão por meio de um TRIAC. O intuito deste sistema é simular a iluminação do sol. A outra divisão simula o interior de uma habitação e nela foi colocado um conjunto de Leds que são alimentados por uma fonte de corrente. Para que esta iluminação seja variável, é ligado o sinal PWM do microcontrolador, associado a este módulo, à fonte de corrente que alimenta os Led.

Para poder usar este módulo neste projeto foi necessário efetuar várias alterações, tais como, inserir a comunicação CAN, um sensor de presenças e a programação do PIC.
Este módulo é composto por duas categorias, sensores e atuadores.
No lado dos sensores tem-se:

- Sensor de luminosidade exterior;
- Sensor de luminosidade interior;
- Sensor de presenças;
- Sensor de posição da persiana (potenciómetro linear);
- Sensor de regulação da persiana pretendida (potenciómetro rotativo);
- Sensor de regulação da iluminação pretendida (potenciómetro rotativo).

No lado dos atuadores tem-se:

- Motor DC;
- Iluminação LED;
- Leds sinalizadores;

A razão de utilizar o PIC18F458 neste módulo é o facto de necessitar de 6 ADC, descritos na secção 3.1.2 do Capítulo 3. Nos pontos seguintes apresentam-se as características dos componentes utilizados.

A- Sensores de Luminosidade

Para captar a iluminação em ambas as divisões foi instalado um fotodíodo (Figura 46) que é um sensor de luminosidade analógico. Cada sensor está ligado a uma entrada ADC do PIC18F458.

Figura 46 - Fotodíodo

B- Sensor de presença (HC-SR501)

O sensor de presença PIR HC-SR501 (Figura 47) é um dispositivo que permite a deteção de pessoas quando percorrem o seu campo de "visão". É ideal para aplicações com microcontroladores, pois é pequeno, tem um custo reduzido, baixo consumo (50µA) e é fácil de usar. Este dispositivo pode ser alimentado a uma tensão de +4,5V a +20V e tem uma saída digital referida para 0-3,3V.

Figura 47 - Sensor de presença PIR HC-SR501

Como este dispositivo contém internamente um temporizador, que pode ser regulado por meio de um potenciómetro, é possível definir o tempo em que a saída se encontra no nível lógico 1. Sendo assim, é possível manter a saída com os 3,3V durante um período de tempo, que por defeito é de 5 segundos. Com este procedimento é exequível ligar a saída digital do PIR HR-501 a uma das entradas analógicas ADC do microcontrolador.
Ao inserir este dispositivo no módulo de iluminação é possível, através da aplicação criada, definir o período de tempo que a iluminação permanece ligada.

C- Potenciómetro linear

Foi acoplado à persiana um potenciómetro do tipo *slider* (Figura 48) que acompanha todo o movimento vertical da mesma. É usada uma das entradas analógicas do microcontrolador para ler o valor deste potenciómetro.

Figura 48 - Potenciómetro linear

D- Potenciómetro rotativo

Este módulo possui dois potenciómetros rotativos que permite ao utilizador regular o nível da iluminação dos Leds e a posição da persiana (modo manual).

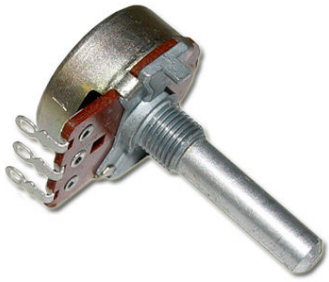

Figura 49 - Potenciómetro linear

E- Atuadores

Do lado dos atuadores, foram utilizados dois conjuntos de quatro Led (1W cada conjunto) na divisão que se pretende um controlo da iluminação.

Figura 50 - Iluminação LED

O microcontrolador responsável pelo funcionamento do módulo de iluminação tem vários processos na sua programação, como se pode ver na Figura 51. Este PIC está configurado para despoletar duas interrupções externas (*Interrupt1* e *Interrupt2*). A interrupção 1 existe para o utilizador poder alterar o estado de manual para automático e vice-versa. O utilizador também tem a possibilidade de neste módulo, ligar ou desligar a iluminação interior, através da segunda interrupção externa.

O ciclo *while* do microcontrolador está configurado para ler todos os valores das portas ADC ativadas. De seguida o PIC18F458 define o valor dos seus *outputs* (PWM, Motor, Sentido motor) consoante os valores que tem nas variáveis associadas, podendo ser alteradas tanto no módulo como nas aplicações desenvolvidas. Para este microcontrolador receber a mensagem CAN vinda do PIC *Master*, este tem uma terceira interrupção configurada.

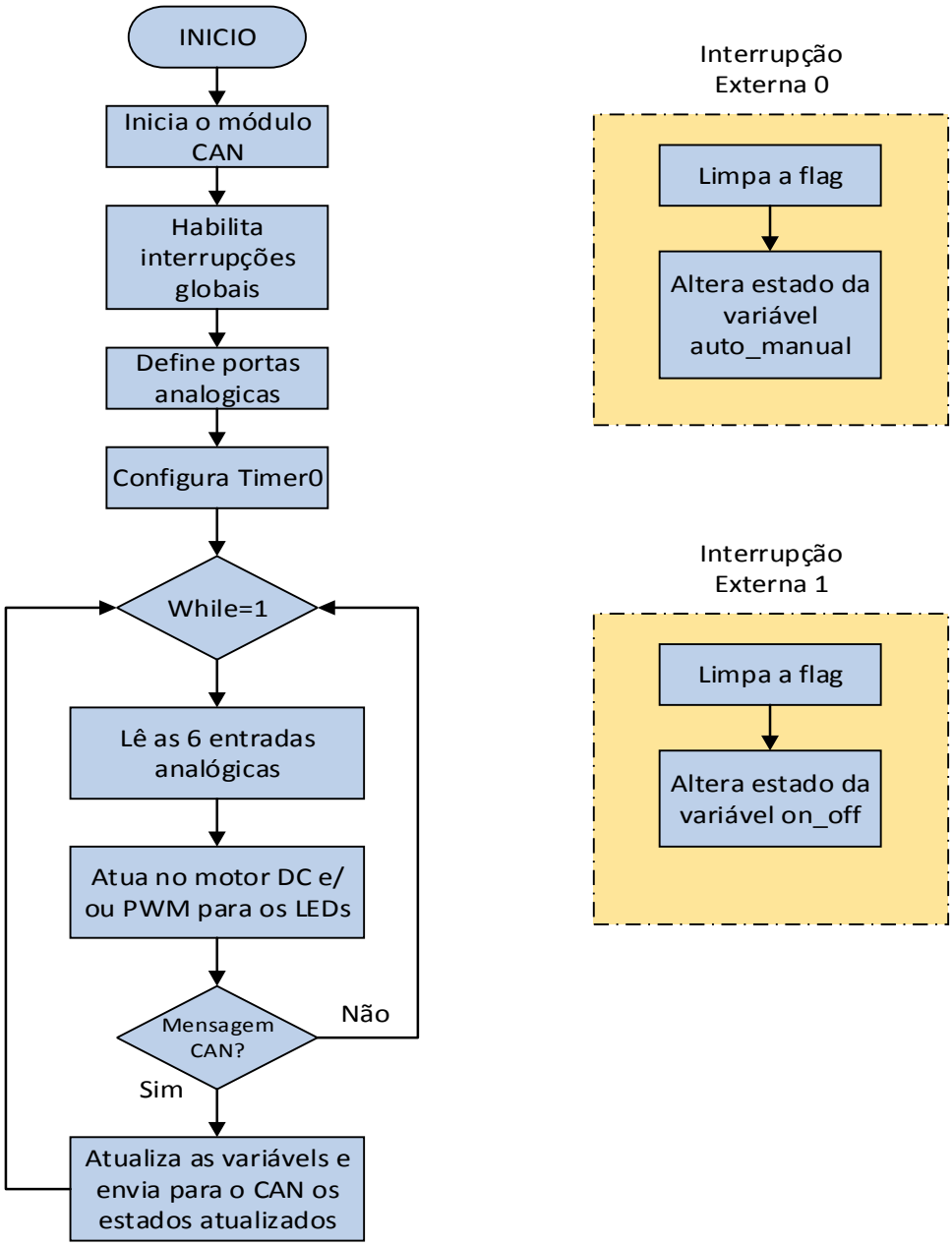

Figura 51 – Diagrama referente ao módulo de iluminação

4.2.3 Módulo de deteção de Gás

O módulo de deteção de gás, como o nome indica, é um circuito que tem como objetivo obter a leitura de um sensor sensível a vários tipos de gás, correspondendo assim este módulo à categoria de segurança. É utilizado o sensor MQ-2 (Figura 52) que é altamente sensível a GPL (gás liquefeito de petróleo), butano, propano, metano, e também pode ser usado para detetar níveis de álcool e hidrogénio.

Figura 52 - Sensor de Gás MQ-2

A escolha deste sensor deveu-se ao facto de ser adequado para o tipo de gás que é habitual ser usado no meio doméstico, de apresentar uma boa rapidez de resposta e ter um preço de aquisição baixo. Como a sua tensão de alimentação é de +5V, facilmente se consegue integrar em um circuito com microcontroladores, Figura 53.

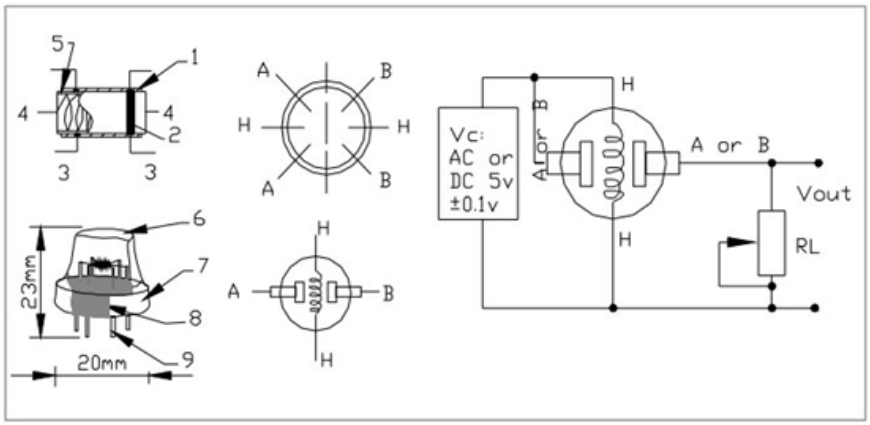

Figura 53 - Esquema de ligação do sensor MQ-2

Ao aplicar 5V no pino H e consecutivamente no pino A ou B, este sensor disponibiliza na sua saída uma tensão referente à sua leitura, o que é ideal para se ligar ao microcontrolador. A sensibilidade do MQ-2 pode ser ajustada através da inserção de uma resistência variável entre a saída e o GND.

À semelhança da rotina do módulo de temperatura, nesta também foi configurado o *Timer0* para que a cada 50 milissegundos execute uma interrupção e faça a leitura do valor que o sensor fornece (Figura 55). O valor lido é guardado numa variável para que possa ser envida para o barramento CAN.

```
ConvertADC();
while(BusyADC());
resultado = ReadADC();
```

Figura 54 - Leitura do ADC efetuado na interrupção

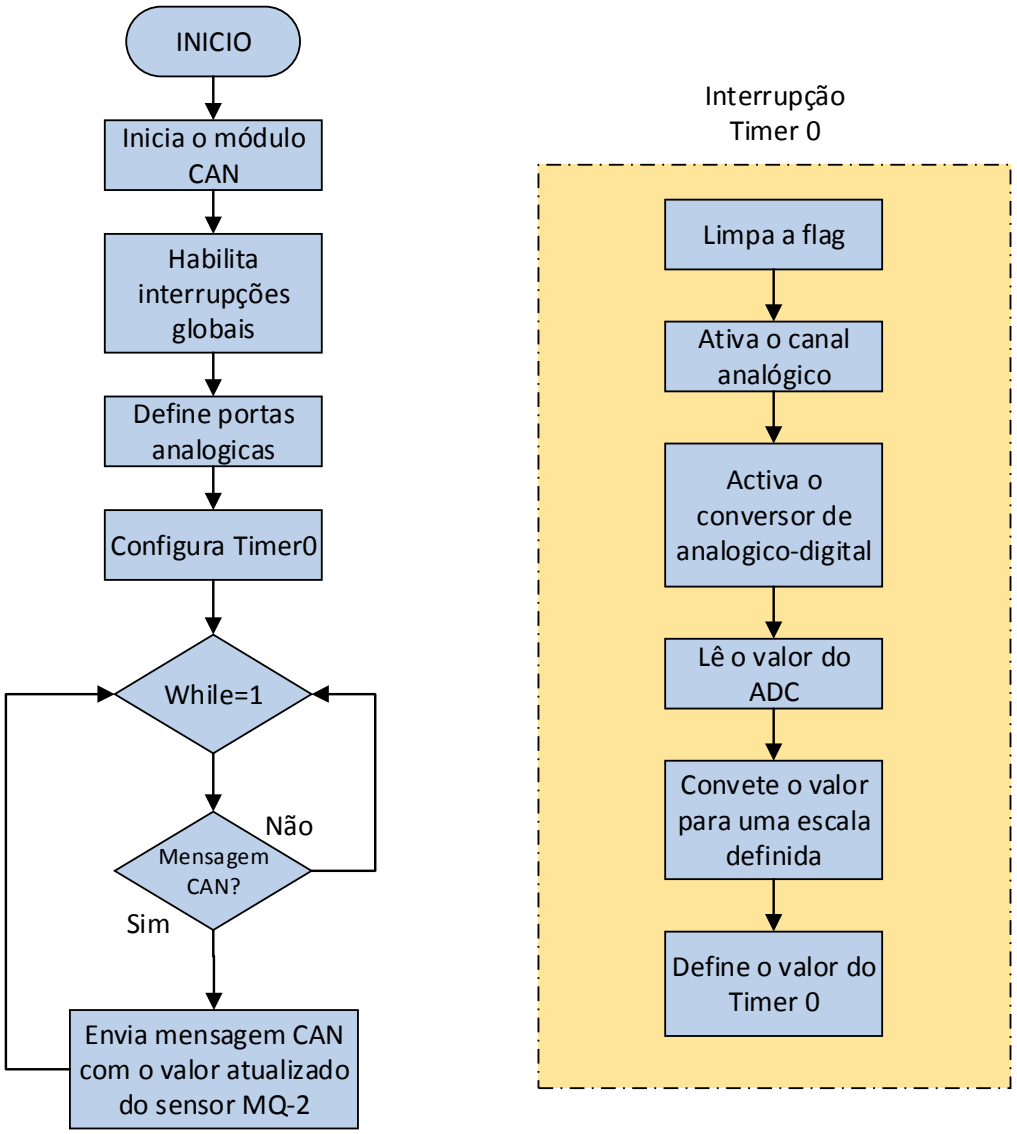

Figura 55 - Diagrama do módulo de deteção de gás

O esquema da Figura 56 mostra o circuito elaborado para este módulo, sendo desenhado para ser ligado ao sensor MQ-2 através de um conjunto de pinos. Um microcontrolador PIC18F258 e o *transceiver* que está ligado ao barramento CAN. Todo o circuito é alimentado por um transformador de +12V com um regulador IC LM7805 para manter a tensão nos +5V.

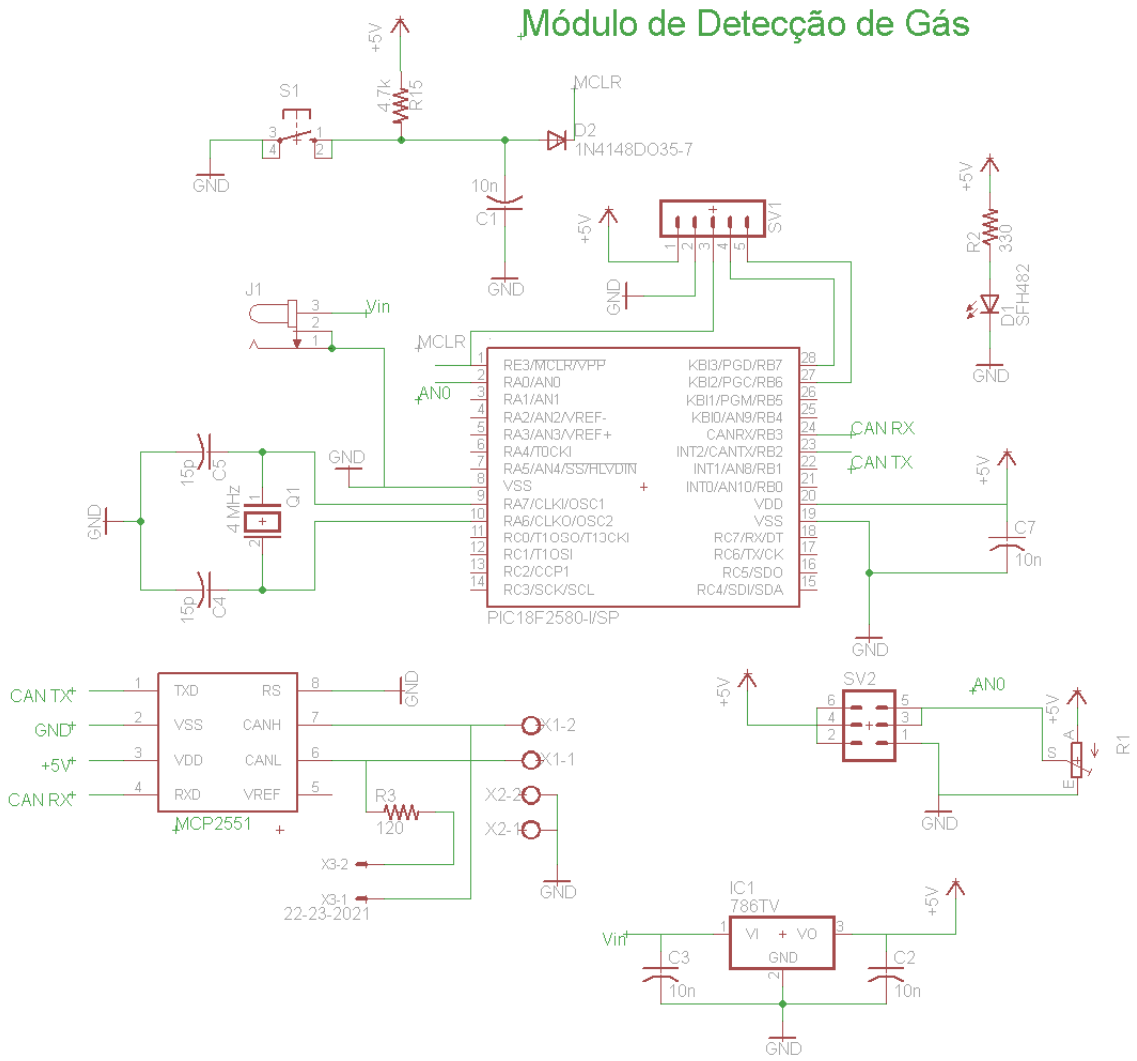

Figura 56 - Esquema em *Eagle* do módulo de Deteção de Gás

A Figura 57 mostra a montagem final deste módulo onde é possível ligar o sensor MQ-2 a partir de um conjunto de pinos de forma a instalar o sensor no local mais adequado para as medições.

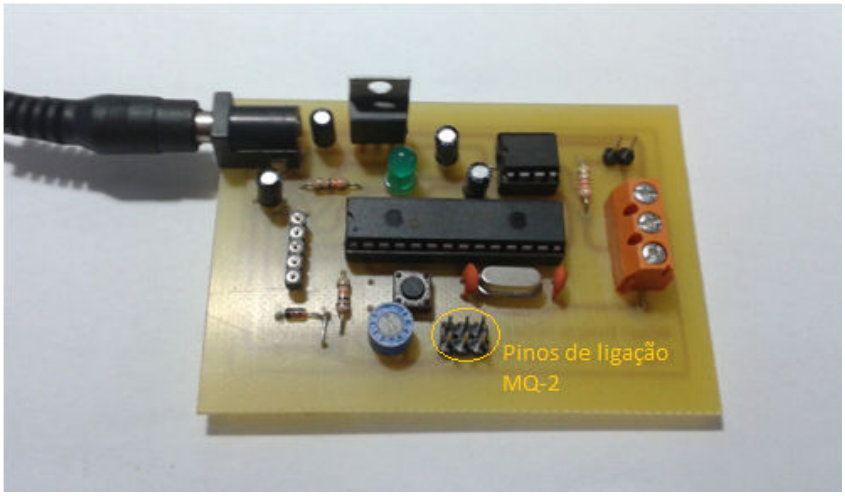

Figura 57 – Placa em PCB do módulo de deteção de gás

Capítulo 5

5 Aplicação: desenvolvimento do software

Este capítulo descreve todo o trabalho desenvolvido a nível da aplicação em *Microsoft Visual Studio*. Esta aplicação é desenvolvida para duas vertentes, uma para ambiente Windows e outra para plataforma Web como é descrito nos seguintes pontos.

5.1 Primeira Implementação: Aplicação Windows

5.1.1 Primeira fase

Como foi referido no capítulo anterior, independentemente da comunicação usada entre a rede de domótica e o computador, um dos objetivos seria a criação de uma interface gráfica para que o utilizador pudesse interagir com o sistema domótico. Neste sentido, na primeira fase foi desenvolvido um programa em *Visual Basic* para testar a comunicação série USART com o computador.

Através da função *SerialPort_ReadByte*, a aplicação atribuía o valor que recebesse da porta série para a variável que dependia da sequência de receção.

As variáveis que estavam agregadas aos dados de receção foram associadas a *Label* que eram atualizadas através do método *Timer1_Tick*.

5.1.2 Segunda fase

Como foi adotado a comunicação do sistema [17], a necessidade para realizar a interface partiu desse prossuposto. Sendo assim, a interface gráfica foi desenvolvida em *C#* como linguagem de programação.

Da mesma forma que é realizada a comunicação do microcontrolador com a porta USB (descrito no capítulo 4), a aplicação gráfica funciona com o mesmo princípio. Uma classe chamada de *USBclass.cs,* contém todas as funções em *C#* para "ler" e "escrever" no USB, *receiveViaUSB* e *sendViaUSB,* respetivamente.

Estas funções são chamadas através de um temporizador que despoleta o evento *Timer1_Tick* a cada 50 milissegundos, chamando a função *receiveViaUSB* em primeiro lugar. Para aceder a cada byte do *array* recebido, é utilizada a estrutura *USBObject.fromDeviceToHostBuffer(x)*, em que x é o número do byte. A mesma forma é usada para atribuir os valores ao *array* de bytes enviados, *USBObject.fromHostToDeviceBuffer(x)*, finalizando a função do *Timer* com *sendViaUSB*.
Esta comunicação, entre a rede de Domótica e a interface gráfica, funcionou bastante bem mas seria um sistema muito limitado em termos de flexibilidade de acesso. Esta limitação obriga o utilizador a ter que se deslocar ao computador onde o sistema está inserido para poder interagir com a rede de Domótica.
Para ultrapassar esta situação, decidi implementar uma aplicação Web de modo a que o utilizador possa ter acesso à aplicação gráfica e interagir com o sistema domótico a partir de qualquer outra máquina, dentro ou fora da rede local (bastando ter acesso à internet). Sendo assim, o próximo subcapítulo vai abordar todo o processo de realização da aplicação Web.

5.2 Segunda Implementação: Aplicação Web

5.2.1 Primeira fase

Para realizar a aplicação Web foi utilizada a tecnologia ASP.NET com recurso ao AJAX, atualmente um dos mais comuns para a criação dinâmica de páginas Web. O ambiente de desenvolvimento é também o Visual Studio 2010, já que este possui boas características para este tipo de aplicações. Como linguagem de programação optei por usar o C#, por ser considerada a linguagem mais comum para este tipo de aplicações e também por ser a linguagem utilizada na interface com o USB.
Como tinha poucos conhecimentos sobre este assunto, comecei por iniciar um projeto que conseguisse o mesmo resultado que a aplicação desenvolvida anteriormente em ambiente Windows. Posto isto, este processo está dividido em várias fases de desenvolvimento.
Em termos de *design* gráfico, usou-se como base o *template* que vem na ferramenta, pelo facto que não era prioridade neste trabalho primar o aspeto visual, mas sim a

funcionalidade do mesmo. A primeira alteração a ser executada foi na página *default.aspx* que é a página a ser criada por defeito.

De modo a recriar as mesmas funcionalidades da aplicação Windows, as primeiras alterações realizadas consistiram na introdução do evento *Timer* para a atualização das várias *Label* e para a realização da comunicação USB de acordo com o processo usado na aplicação Windows, como se ilustra na Figura 58.

```
<asp:Timer ID="Timer1" runat="server"
Interval="500"    OnTick="Timer1_Tick">
</asp:Timer>
```

Figura 58 - Timer1 em aspx

Para que a página não seja completamente atualizada, usei o controlo *Update Panel* AJAX. Visto que este controlo, quando existe um *PostBack* criado por um evento no seu interior, faz com que não se visualize a atualização da página. Ou seja, sempre que o evento *Timer1* é ativado, o conteúdo que estiver abrangido pelo *Update Panel* é atualizado. A estrutura do *Update Panel* é a seguinte:

```
<asp:UpdatePanel ID="UpdatePanel_On_Off" runat="server">
        <ContentTemplate>

            <asp:Label ID="Label_On_off" runat="server"> </asp:Label>

        </ContentTemplate>
        <Triggers>
            <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />
        </Triggers>
</asp:UpdatePanel>
```

Figura 59 - Update Panel em aspx

O *Timer1* é responsável por chamar a função *ReceiveViaUSB* da classe USB e refrescar os valores das *Label*. Os botões disputam eventos que modicam o *array* de *bytes* de envio para o USB.

Após recriar estes processos conseguiu-se um resultado semelhante ao ambiente criado para a aplicação Windows.

A Figura 60 mostra a arquitetura realizada nesta primeira fase do desenvolvimento da aplicação Web.

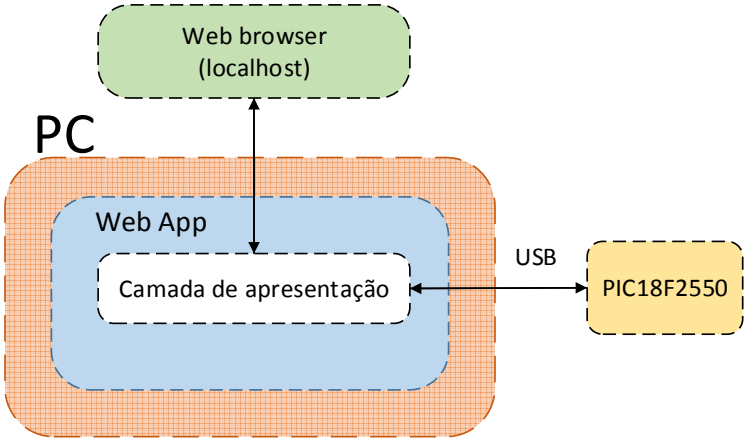

Figura 60 - Arquitetura da primeira fase da aplicação *Web*

Esta implementação em ambiente Web foi a base para teste dos processos desenvolvidos na aplicação Windows e a classe USB. Após alguns testes a esta solução, verificou-se que era funcional mas que não correspondia ao melhor formato para implementar este tipo de soluções.

No próximo ponto explica-se a evolução conseguida para esta aplicação.

5.2.2 Segunda fase

Em aplicações Web é usual utilizar uma estrutura bem mais sólida do que a que foi utilizada na fase anterior. Desta forma, efetuei um trabalho de pesquisa e verifiquei que é utilizada uma arquitetura a que se designa de "*Three-Tier*". Esta estrutura é constituída por três camadas, denominada de arquitetura distribuída Cliente - Servidor.

A constituição das camadas é a seguinte:

- Apresentação;
- Negócio;
- Acesso a dados.

A Figura 61 ilustre a topologia implementada na aplicação desenvolvida.

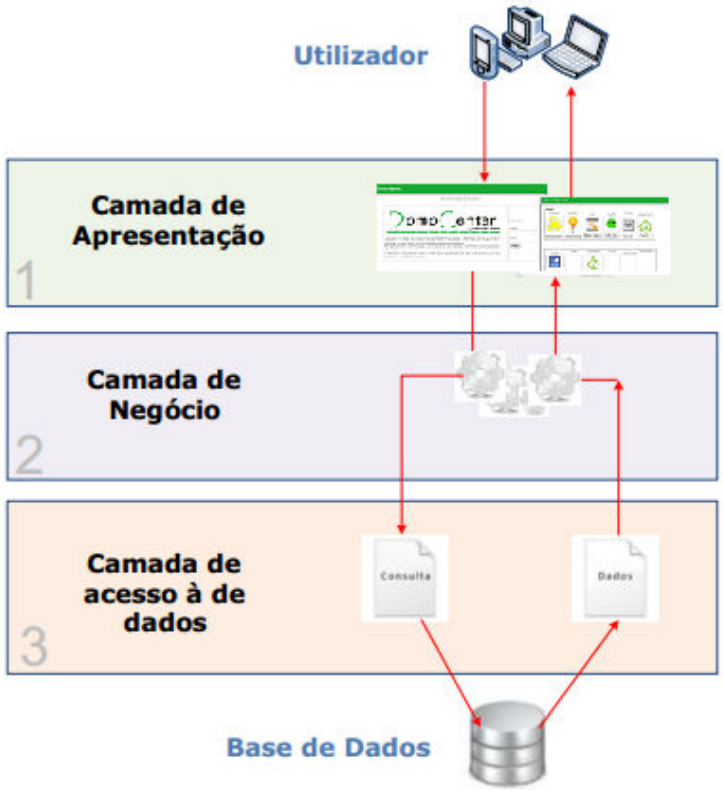

Figura 61 – Topologia da arquitetura *Three-Tier*

Com esta arquitetura tem-se diversas vantagens, mas também algumas desvantagens:

As principais vantagens são:

- Aplicação de regras de negócio;
- Flexibilidade;
- Maior segurança;
- Tratamento de Erros;
- Escalabilidade;
- Fácil Manutenção;
- Controlo de transações e consultas.

Em relação às desvantagens podem-se destacar:

- Implementação mais demorada e complexa;
- Separação da interface do utilizador, a gestão de processos e dados lógicos nem sempre é óbvia.

Para melhor esclarecimento da arquitetura usada, descrevo cada uma das camadas nos parágrafos seguintes:

A- Camada de apresentação

É a interface com a qual o utilizador vai interagir diretamente, ou seja, todas as páginas Web onde se insere ou se consulta informação com destino ou origem da camada de negócio.

B- Camada de negócio (BLL)

Também designada por camada Lógica Empresarial ou do inglês "*Business Logic Layer*", é nela que estão alocadas as funções de processamento da aplicação. O objetivo é fazer a ligação entre a camada superior – **Camada de apresentação** – e a inferior que interage diretamente com base de dados – **Camada de Acesso a Dados**.
Esta camada tem um papel muito importante para a aplicação, pois é dentro desta que estão alojadas as funções para a comunicação USB.
Sendo esta a camada que faz a ligação com a camada inferior (camada de acesso à base de dados), os métodos existentes nas classes desta camada que tratam da interligação são do seguinte tipo (disponível no Anexo 2):

- ***Insert*** – Método que insere registos novos;
- ***Refresh*** – Método que modifica dados já existentes;
- ***Get –*** Método que realiza a consulta à base de dados.

Com esta estrutura mais sólida, é possível ao utilizador consultar a Base de Dados através da camada de negócio utilizando estes métodos.

C- Camada de Acesso a Dados (DAL)

Designada por DAL (*Data Acess Layer)*, consiste no acesso ao sistema de base de dados. Recebe e envia dados de origem ou destino da camada de negócio (BLL).
De modo a facilitar a comunicação à base de dados, atualmente está implementado para este tipo aplicações uma ferramenta chamada de ORM (*Object-Relational Mapping*). É

uma técnica de desenvolvimento utilizada para reduzir o esforço da programação orientada a objetos utilizando base de dados. Assim as tabelas da base de dados são representadas através de classes e os atributos de cada tabela são representados como instâncias dessas classes. Com esta técnica não é necessário interpretar a base de dados com comandos SQL (*Structure Query Language*) sempre que se queira comunicar com a mesma.

D- Tabela *tData*

Nesta fase da aplicação, a base de dados apenas contém a tabela *tData*. Como se pode ver na Figura 62, a tabela é constituída por três atributos: ID, Nome e Valor. O ID (identificador) é a chave primária da tabela, o' Nome' é a descrição da linha e 'Valor' é onde será colocada a informação inerente ao registo.

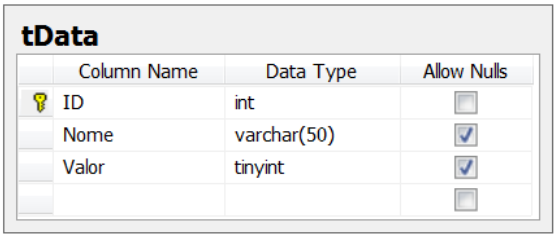

Figura 62 - Tabela SQL *tData*

Na tabela *tData* está alojada toda a informação referente aos estados da rede de domótica. Como foi explicado anteriormente, o *array* de bytes de receção do USB é composto por 65 bytes, ou seja, cada registo da tabela *tData* representa 1 *byte* desse *array*. Na Figura 63 é ilustrada a tabela *tData* com alguns registos provenientes da rede de domótica desenvolvida.

TIAGO-PC\SQLEXPRESS.DC - dbo.tData

ID	Nome	Valor
1	0	0
2	Auto_Manual	0
3	Iluminação pret	8
4	On_Off	1
5	Sensor Exterior	251
6	Sensor Interior	112
7	Sensor Presenças	0
8	Sensor Persiana	156
9	Posição Persiana	7
10	Não Atribuído	0
11	LM35	16

Figura 63 - Tabela SQL *tData* com registos da rede de domótica

Apesar da comunicação USB ser tratada na camada de negócio, todos os pedidos feitos pelo utilizador incidem diretamente na classe *Manager* (disponibilizado no Anexo 2) associada ao tratamento de dados de envio e receção do USB, Figura 64.

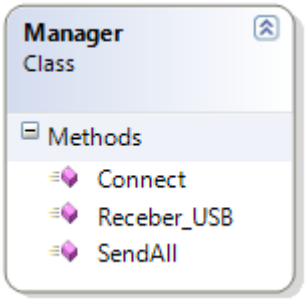

Figura 64 - Classe *Manager*

Para a atualização das *Label*, o método "Receber_USB" desta classe também é chamada dentro do *Timer* associado à página em desenvolvimento.

```
byte[] data = DC.BLL.USB.Manager.Receber_USB();
```

Figura 65 - Método Receber_USB

A ligação USB começou a mostrar alguns problemas na sua comunicação e algumas das razões aparentes são:

- O facto de ter sido alterado o processo de envio no USB no lado da aplicação.
- Tantos pedidos quantos possíveis, na classe *Manager*, dependendo da quantidade de páginas HTML abertas em *Browser*.

Na figura abaixo está representada a arquitetura da aplicação Web na segunda fase de desenvolvimento. Como se pode ver, a aplicação ainda era testada num *localhost* gerado pela aplicação ASP.NET.

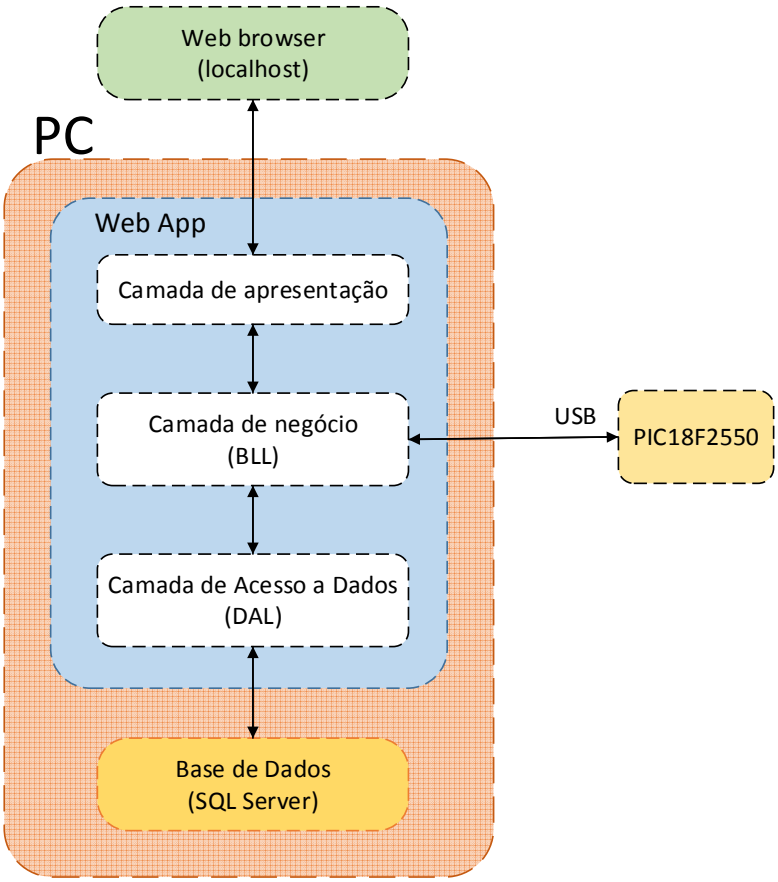

Figura 66 - Arquitetura da segunda fase da aplicação Web

5.2.3 Terceira fase

De maneira a contornar os problemas encontrados na fase anterior, houve a necessidade de alterar o processo entre os pedidos do utilizador e a comunicação USB. De forma a não ser chamado o método 'Receber_USB' e o método de envio a cada página *HTML* ou a cada pedido do utilizador, teve que ser criado um tipo de serviço único que tratasse de todos os pedidos provenientes do cliente.

Para isso foi necessário criar uma tabela na base de dados, chamada de *tQueues*, que contém todos os pedidos do cliente, como se apresenta na Figura 67.

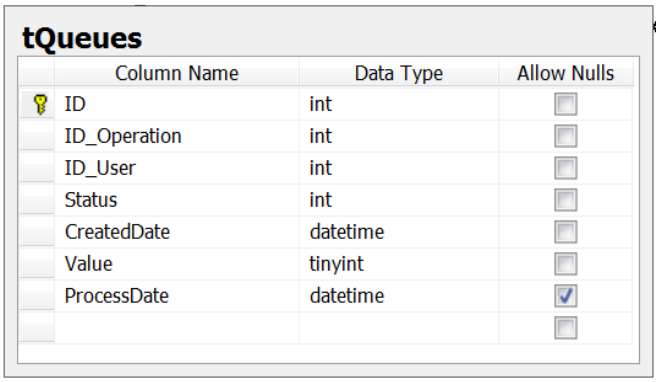

Figura 67 - Tabela SQL *tQueues*

Esta tabela é constituída por diversos atributos, dos quais se elencam:

- *ID* - Identificador único;
- *ID_Operation* - Identificador da operação;
- *ID_User* - Identificador do utilizador;
- *Status* – Estado do pedido;
- *CreatedDate* - Data de criação do mesmo;
- *Value* – Valor do pedido;
- *ProcessDate* - Data de processo do registo.

Todos os pedidos efetuados pelo utilizador da aplicação são enviados para a tabela *tQueues* com o valor de *status* 0. Para enviar os pedidos para esta tabela, todos os botões da aplicação têm associados no respetivo método, um outro chamado de *Insert,* que faz precisamente o envio do pedido para a tabela *tQueues* com a informação do *ID* do utilizador, *ID* da operação e o respetivo valor (Anexo 2 – Classe *Queues*).

Para o serviço USB identificar quais os pedidos que deve executar, a tabela *tData* está ligada pela chave primária a uma outra tabela, chamada de *tOperations* (Figura 68).

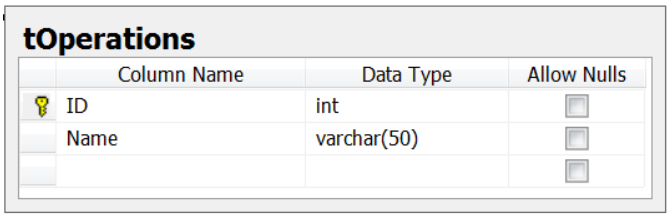

Figura 68 - Tabela SQL *tOperations*

Esta tabela tem todos os nomes referente às diversas operações que a aplicação Windows e Web contêm, como mostra a Figura 69:

TIAGO-PC\SQLEXPR... dbo.tOperations

	ID	Name
	1	Definir Manual_Auto
	2	Definir Luminosidade
▶	3	On_Off
*	NULL	NULL

Figura 69 – Registos data tabela SQL *tOperations*

O serviço USB é composto essencialmente por dois métodos responsáveis pelo envio e receção, *Process* e *Refresh*, respetivamente (Anexo 2 - Classe *Queues*, Classe *Data*).
O método *Process* que se encontra na classe *Queues* é responsável por fazer uma *Query* à tabela *tQueues* e verificar todos os registos da mesma e identificar quais os que têm o *status* igual a 0. De seguida é criada uma lista para todos esses pedidos pendentes. Para cada pedido, identifica o tipo de operação e vai construir uma outra lista chamada de *data* que será enviada para o USB através do método *SendAll* que está alocado na classe *Manager*. Por fim, aos registos processados são atribuídos o *status* com o valor 1.

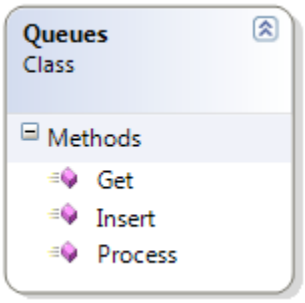

Figura 70 - Classe *Queues*

O método responsável por receber os dados provenientes da comunicação USB é o método *Refresh* que se encontra na classe *Data*. Neste é criado uma variável *array* em que é atribuído o conteúdo que está no método *Receber_USB* na classe *Manager*. Este *array* através de um ciclo *for* preenche todos os campos na tabela *tData*.

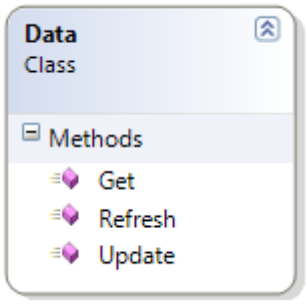

Figura 71 - Classe *Data*

5.2.3.1 Proteção da aplicação

A parte mais sensível deste desenvolvimento e que merece algum cuidado é a que diz respeito à proteção da aplicação Web. Como o objetivo final seria disponibilizar a aplicação fora da rede doméstica foi necessário criar um processo de autenticação que pudesse proteger os recursos de todo o sistema.

Sempre que um pedido chega à aplicação, através do servidor IIS, esta verifica se o pedido está ou não autenticado. Caso não esteja, a aplicação encaminha o pedido de imediato para a página *login.aspx*.

Na Figura 72 apresenta-se a página inicial da aplicação DomoCenter a que o utilizador não registado tem acesso.

Figura 72 - Página inicial *DomoCenter*

Esta página é composta por duas *textbox* e um botão que faz a submissão das credenciais.
Textbox para o nome:

```
<asp:TextBox ID="TextBox_User" runat="server"></asp:TextBox>
```

Figura 73 - Caixa de texto *TextBox_User* em aspx

Textbox para a palavra-chave:

```
asp:TextBox ID="TextBox_Pass" runat="server"
    TextMode="Password"></asp:TextBox>
```

Figura 74 - Caixa de texto *TextBox_Pass* em aspx

A página inicial contém um botão para o utilizador submeter as credenciais:

```
<asp:Button ID="Button_Submit" runat="server"
Text="Login" onclick="Button_Submit_Click" />
```

Figura 75 - Botão *Button_Submit* em aspx

Para a aplicação submeter a informação nas caixas de texto é necessário que no método associado ao botão *Button_Submit* tenha uma rotina que valide as credenciais, Figura 76.

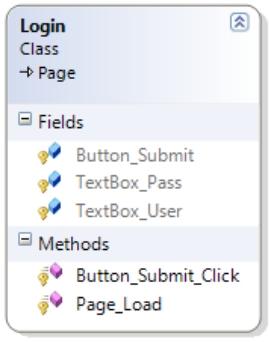

Figura 76 - Classe Login

Sempre que um novo pedido chega ao servidor Web, a aplicação verifica se o cliente está ou não autenticado, para assim atribuir os conteúdos necessários.
Para a aplicação verificar se está perante um utilizador com ou sem permissões é realizado o processo da Figura 77.

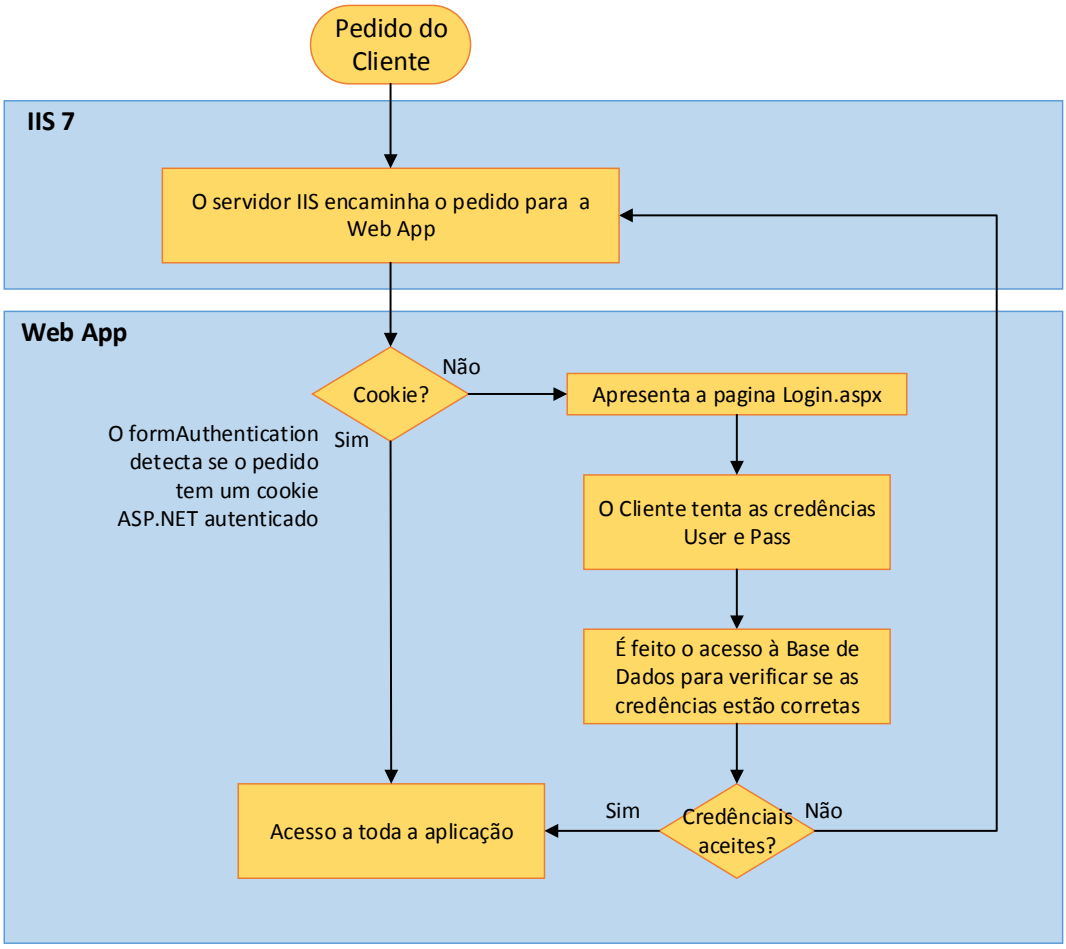

Figura 77 - Diagrama autenticação Login

Assim sendo, é "chamado" no seu método *Button_Submit_Click* um outro método designado de *Validate* (Figura 78). Este método inserido na classe da camada de negócio (BLL), com o nome de *Users* (Anexo 2 – Classe *Users*), é responsável por verificar na base de dados a existência de algum registo com as mesmas características daquelas que foram submetidas [18] [19].

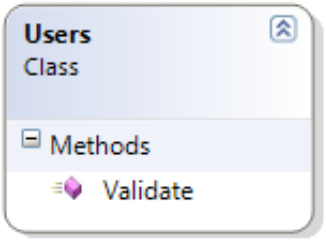

Figura 78 - Classe *Users*

Para a aplicação poder tomar esta decisão, criei uma outra tabela na base de dados, com o nome de *tUsers*, composta por diversos atributos, como mostra a Figura 79.

O método *Validade* vai comparar as credências submetidas pelo utilizador com as dos campos existentes nos atributos *Loginname* e *Password*. Caso a pesquisa à tabela *tUers* encontre um registo igual à informação submetida pelo utilizador, é criado um *ticket* com a informação do utilizador e com um tempo de sessão (neste caso é de 30 minutos). Após a criação deste *ticket*, é enviado um *cookie* autenticado com essa informação para o cliente [20].

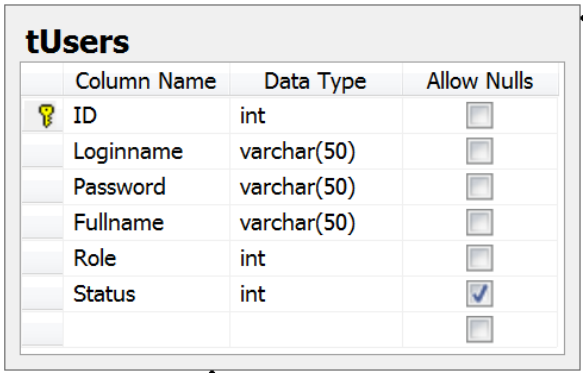

Figura 79 - Tabela SQL *tUsers*

Para que haja algum controlo nos acessos à aplicação e até mesmo dentro da mesma, criei uma tabela designada de *tLogs* que vai armazenar a informação de alguns acontecimentos do sistema, Figura 80. No caso dos acessos login a que a aplicação está sujeita, sempre que exista uma tentativa de submissão na página login, fica registado nesta tabela os conteúdos dos campos *TextBox_User* e *TextBox_Pass*. Para isso, no método relacionado com o botão de submeter, é "chamado" o método *Insert*, alocado na camada de negócio com o nome de *Logs* (Anexo 2 – Classe *Logs*). Assim, o proprietário da aplicação tem a capacidade de saber quais os logins a que a aplicação foi submetida.

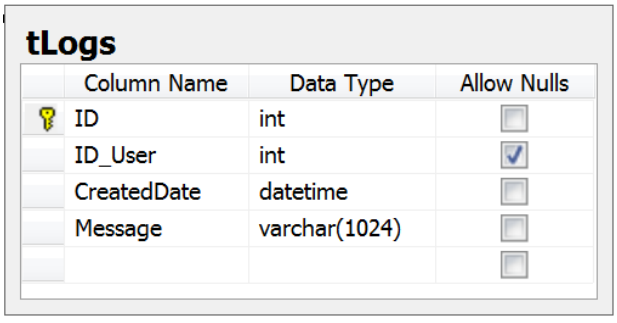

Figura 80 - Tabela SQL *tLogs*

A figura abaixo representa todas as tabelas usadas nesta aplicação bem como as relações existentes.

Figura 81 - Tabelas criadas em SQL *Server*

De forma a resumir todos os assuntos implementados nesta última fase da aplicação Web, a Figura 82 representa a arquitetura final resultante da implementação que foi realizada. A aplicação Windows foi integrada na arquitetura final para que o utilizador possa interagir com o sistema DomoCenter a partir do PC servidor sem ter que utilizar um Web *Browser*.

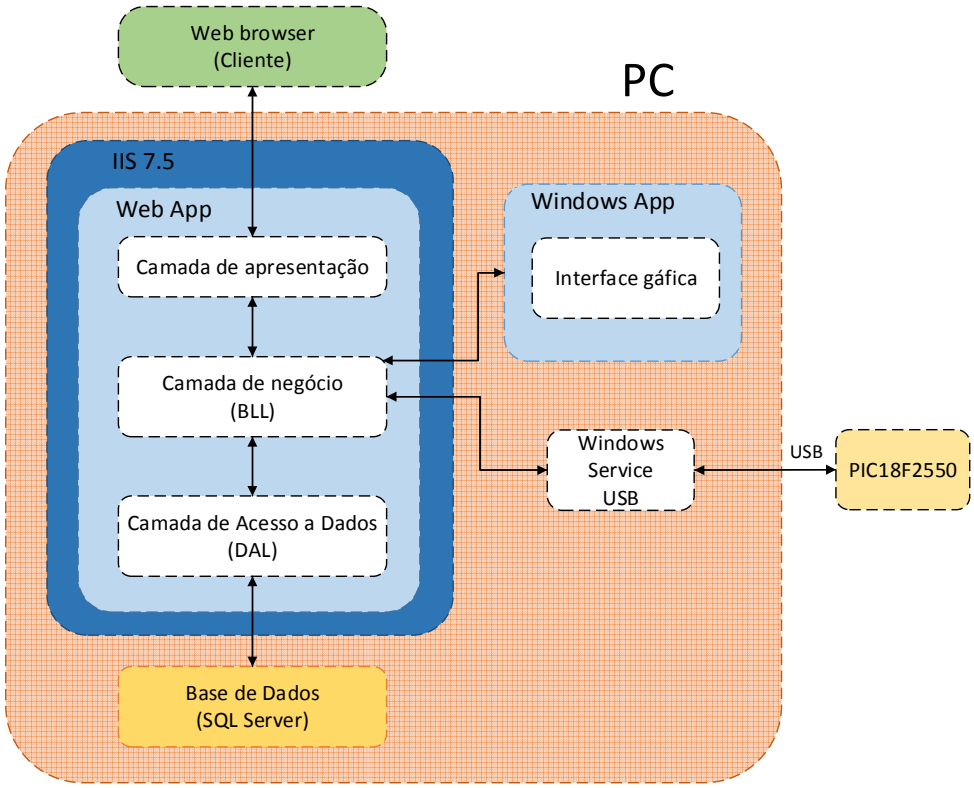

Figura 82 - Arquitetura final da aplicação DomoCenter

5.2.4 Servidor Web

Como foi descrito no Capítulo 3, o servidor Web utilizado para este projeto foi o IIS 7.5 da *Microsoft*. Este produto é uma funcionalidade que vem introduzido nos sistemas operativos Windows, que está desativada por defeito, sendo necessário ativá-la para poder usufruir da mesma. A ativação faz-se através da janela "painel de controlo", clicando em "ativar ou desativar funcionalidades do Windows" e procurando por "Serviços de Informação Internet (IIS) ".

Após ativar esta funcionalidade, é criada na raiz do disco uma pasta que contém a página Web por defeito. Nesta fase, todos os pedidos *HTML* recebidos por esta máquina têm uma resposta idêntica à da Figura 83.

Figura 83 - Página *Default* do IIS 7.5

Para o servidor Web carregar a aplicação DomoCenter, e não a página ilustrada acima, foi necessário adicionar um novo *Website* no Gestor de Serviços de Informação Internet (IIS) e configurar alguns parâmetros. Tais como a diretoria onde se situa a aplicação no computador e dar permissões à pasta onde esta se encontra.

5.2.5 Port Forwarding

Port Forwarding, ou, em português, redireccionamento de portas, é a técnica utilizada para ativar diversos serviços quando se está a usar *Router*.

De forma a disponibilizar a aplicação fora da rede local da habitação, é necessário fazer uma configuração no *modem/router* para que este dê acesso a pedidos provenientes do exterior. Uma vez que, associado ao *modem*, há um endereço de IP público, atribuído pelo ISP (*Internet Service Provider*), é necessário redirecionar todos os pedidos que chegam ao

modem para a porta 80, com destino ao IP do Servidor da aplicação *DomoCenter*, Figura 84.

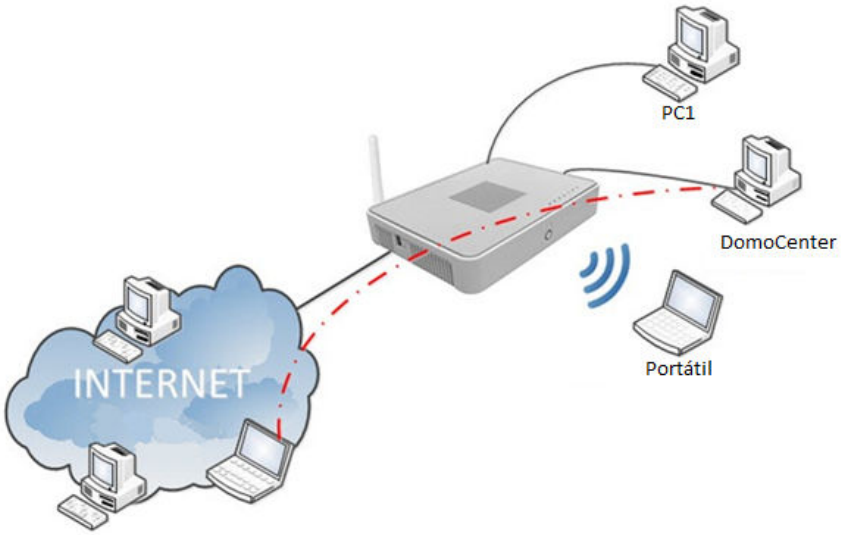

Figura 84 - *Port Forwarding* do *modem*

5.2.6 Servidor DNS

DNS (*Domain Name System),* que em português significa, Sistema de Nomes de Domínios, é uma das ferramentas fundamentais para o funcionamento da internet. Este sistema efetua a resolução de nomes de domínios em endereços IP (sejam eles *IPV4* ou *IPV6*) e vice-versa [21].

Este sistema garante dois objetivos essenciais:

- A possibilidade que dá ao ser humano de se abstrair de endereços de rede (endereços IP) cuja memorização é complexa, ao mesmo tempo que permite alterações desses endereços IP sem que o utilizador tenha que conhecer essa alteração para continuar a usar um serviço;
- A garantia de que as máquinas e os seus nomes sejam geridos de forma hierárquica e distribuída com o *Root Server* mundial no topo da hierarquia e com a informação distribuída por milhares de servidores de nomes existentes na internet (Figura 85). Estas características estão na base do seu sucesso enquanto rede global – não sendo necessário contactar uma entidade central sempre que se efetue uma alteração ou uma adição de novos dispositivos na Internet.

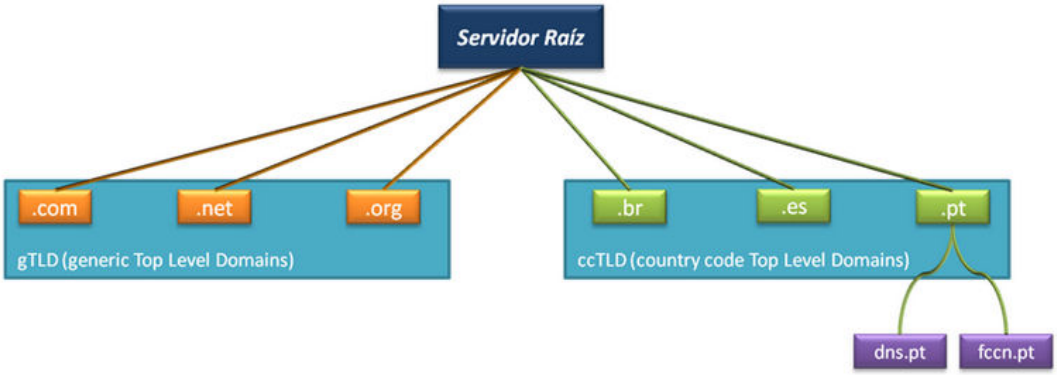

Figura 85 - Distribuição hierárquica do sistema DNS mundial

Para poder usufruir e ter acesso à aplicação fora da rede doméstica, sem ter a preocupação de saber qual o IP público do *modem*, tive que criar um alojamento num serviço DNS para o endereço do mesmo. Um dos pontos a ter em conta é o facto de o endereço IP público atribuído ao *modem* não ser fixo, logo existiu a necessidade de escolher um serviço DNS que fosse dinâmico. Isto significa que, mesmo havendo uma alteração do endereço do *modem*, o servidor de DNS dinâmico garante que o alojamento (*hostname)* associado à conta DNS, aponta sempre para o IP do *modem* em questão.

Uma vez que este projeto tem caracter essencialmente académico, houve a necessidade de criar uma conta num servidor DNS que oferecesse o alojamento sem custos. A escolha recaiu no serviço da *no-ip* que é um dos mais populares para este tipo de aplicações, oferecendo até 3 alojamentos (*hosts)* por conta de utilizador.

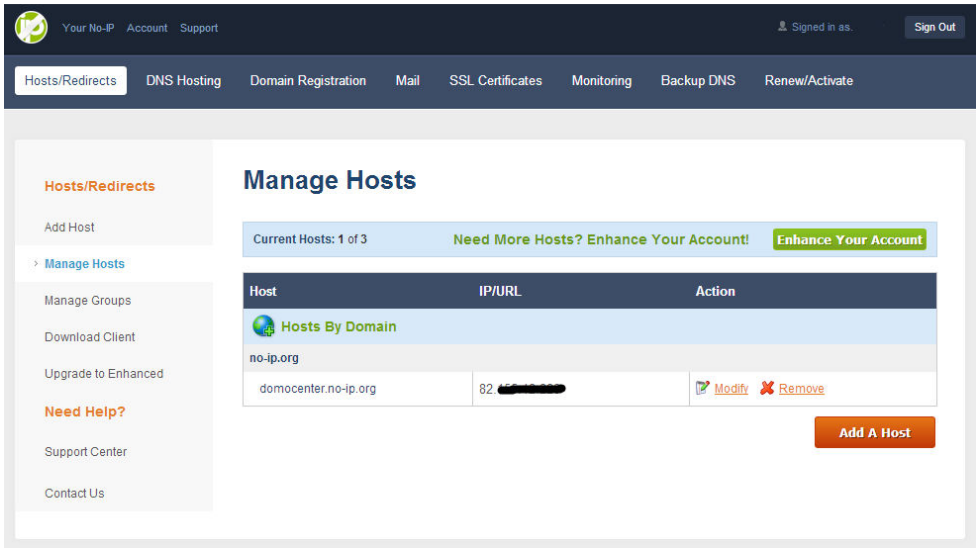

Figura 86 - Alojamento domocenter.no-ip.org

Como se pode ver na Figura 86, ao criar o servidor DNS, é criado em simultâneo o domínio para o endereço em causa. O domínio para a aplicação ficou com o nome de *domocenter.no-ip.org*.

Numa primeira fase, para o servidor DNS ser dinâmico foi necessário instalar no computador, onde a aplicação DomoCenter está alojada, um programa disponibilizado pela *no-ip* que tem o nome de DUC (DNS *Update Cliente*) (Figura 87). Este pequeno programa tem como funcionalidade verificar num intervalo periódico, qual o IP público associado à rede onde está instalado e atualizar este endereço no servidor DNS associado à sua conta.

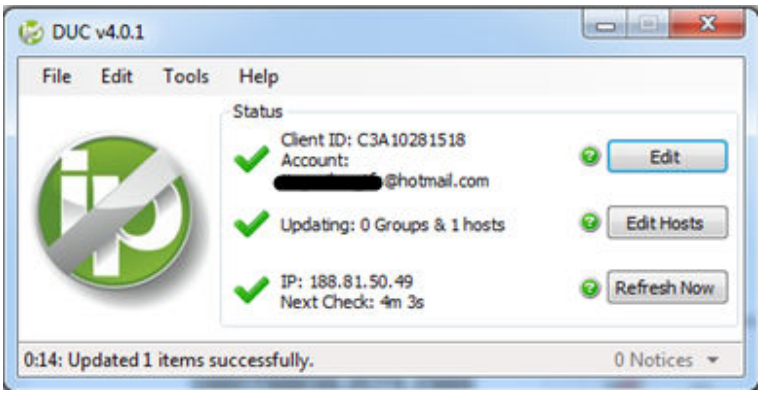

Figura 87 - Software DUC v4.01

Após uma pesquisa às funcionalidades do *modem* em estudo, verificou-se que este possui uma ferramenta de Serviço DNS dinâmico (Figura 88). Com esta ferramenta é dispensável o uso do software DUC dentro do servidor da aplicação, visto que o *modem* passa a tratar da atualização do seu endereço público com a conta associada ao serviço DNS.

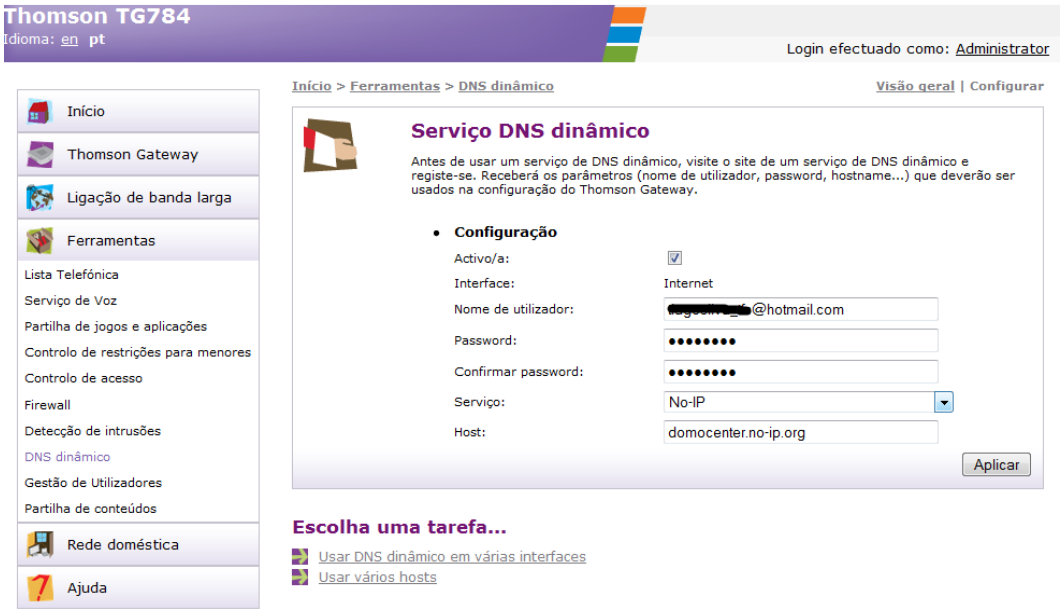

Figura 88 - Serviço DNS dinâmico no *modem*

A Figura 89 ilustrar como é realizado o acesso do cliente à rede de domótica CAN, através da internet.

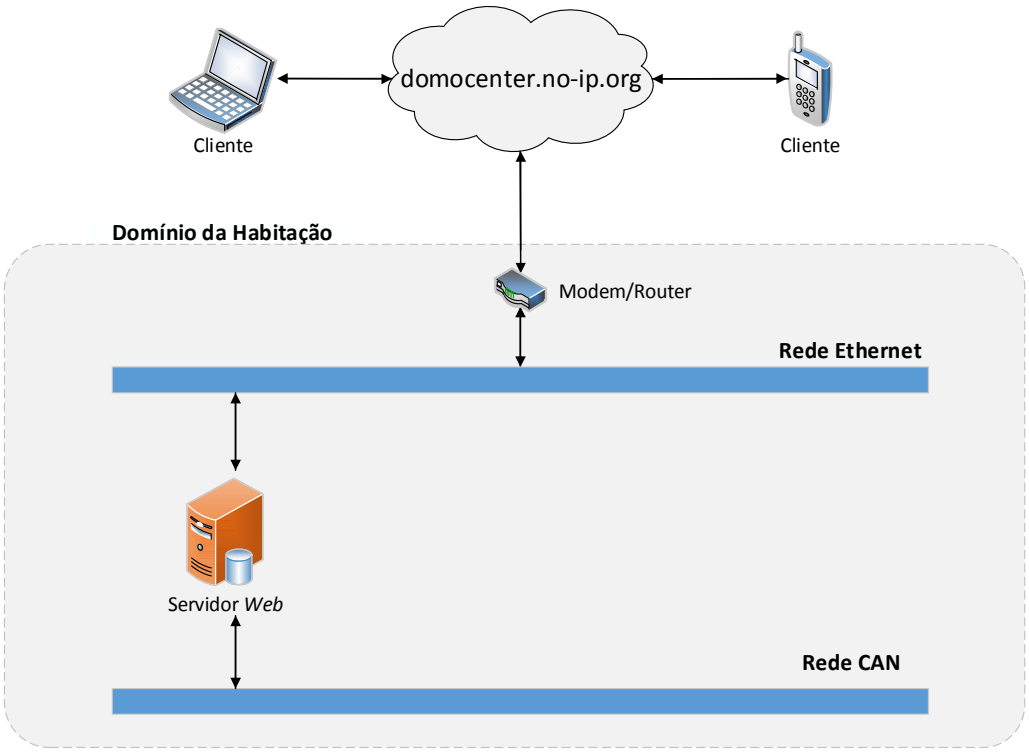

Figura 89 – Cliente/Rede Domótica

5.3 Aplicação desenvolvida

Como se pode ver na Figura 90, a aplicação desenvolvida está composto pelo módulo de iluminação e de temperatura. Também é visível o adaptador USB-CAN que está ligado ao computador através da conexão USB. O módulo de deteção de gás não foi inserido devido ao atraso na aquisição do sensor MQ-2, impossibilitando ligar este módulo ao barramento CAN.

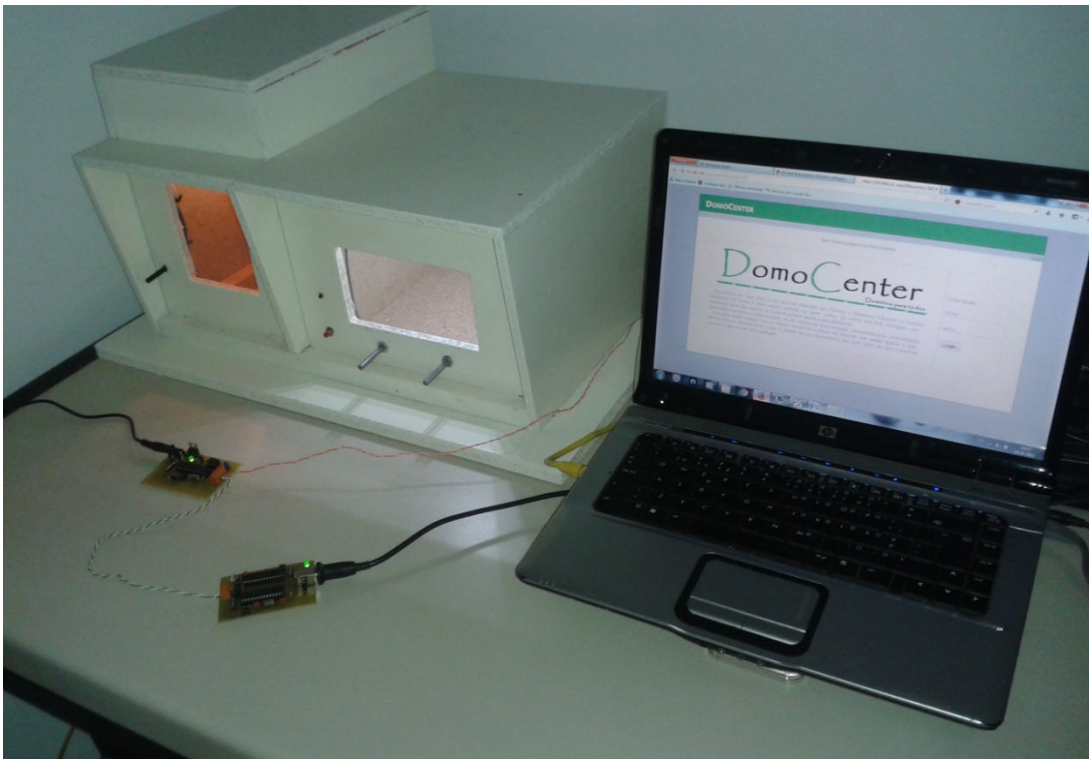

Figura 90 - Aplicação desenvolvida

A Figura 91 apresenta a página html referente ao módulo de iluminação da aplicação em ambiente Web. O utilizador autenticado tem a possibilidade de visualizar os estados atualizados dos sensores e variáveis provenientes dos módulos. É possível modificar o estado manual/auto, ligar/desligar a iluminação, alterar o nível de iluminação numa escala de 0 a 10, bem como a posição da persiana. Através de um temporizador é possível agendar a alteração dos estados tanto na iluminação como na posição da persiana.

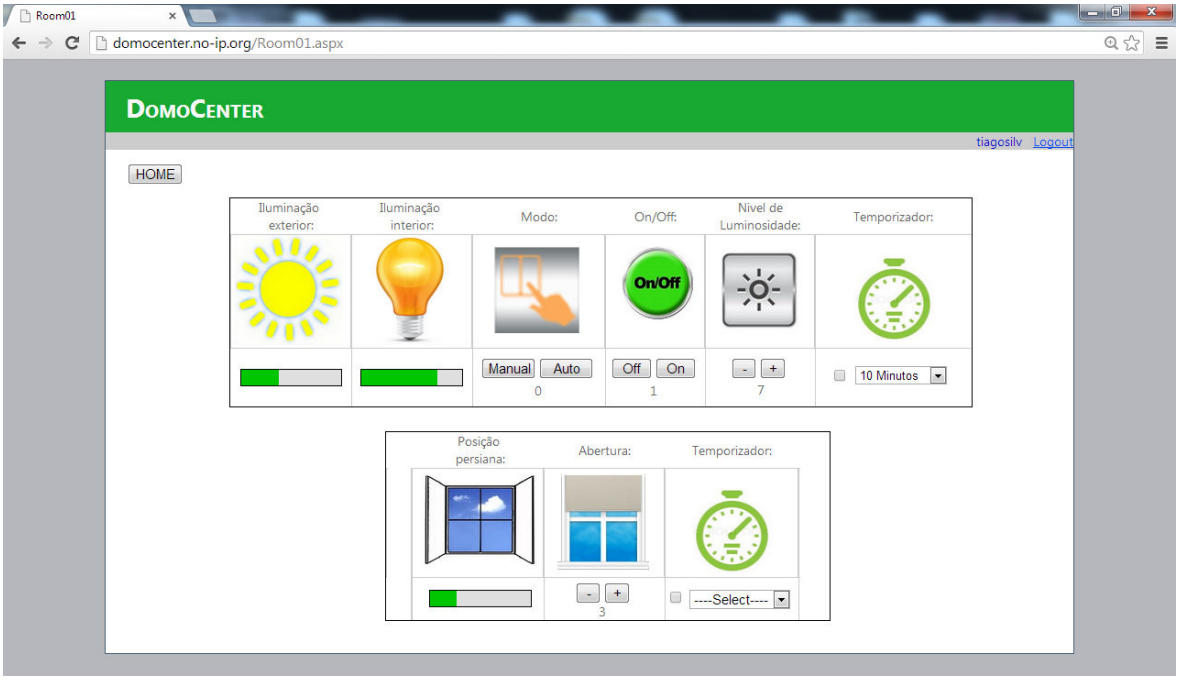

Figura 91 - Aplicação DomoCenter em ambiente Web

6 Conclusões

Neste trabalho foi desenvolvido um sistema de monotorização e controlo de um protótipo para uma habitação com recurso a vários microcontroladores e algumas ferramentas de software.

Numa primeira fase do desenvolvimento do projeto houve alguma dificuldade na escolha da arquitetura a adotar na rede de sensores e atuadores. Um dos objetivos para a criação da rede de domótica consistia em usar microcontroladores, pelo que todas as hipóteses se apoiariam neste princípio.

Como foi referido no capítulo 3, nem todas as escolhas foram as mais acertadas, houve um período de esforço e dedicação a um tipo implementação que acabou por ter que ser abandonado. De maneira a contornar esta adversidade, a comunicação adotada foi o protocolo CAN, tendo sido construído um barramento de comunicação para ligar todos os microcontroladores usados neste projeto.

No caso da *Gateway* que faz o fluxo de informação entre o barramento CAN e as aplicações desenvolvidas em ambiente Windows, houve algum tempo de pesquisa e testes em possíveis comunicações. Na primeira implementação foi utilizada a comunicação Série entre o PIC *Master* da rede domótica e o computador. Esta solução acabou por ser alterada, incluindo o PIC18F2550 entre o PIC *Master* e a ligação com o computador.

Outro dos objetivos principais deste trabalho correspondia à criação de uma interface gráfica na plataforma Windows. Este ponto foi bem sucedido e diria até que foi superado, pelo facto de ter sido implementado juntamente com esta aplicação, uma outra em ambiente Web. Para desenvolver estas aplicações, optei pela plataforma de desenvolvimento disponibilizada pela *Microsoft*. Esta fornece atualmente umas das melhores plataformas de desenvolvimento e execução de sistemas e aplicações, pois consegue reunir dezenas de linguagens. Isso é possível devido à *Framework* comum a todos os sistemas *Microsoft*. Esta ferramenta tem o mesmo princípio que a plataforma Java, ou seja, um programador deixa de escrever código para um sistema ou dispositivo específico e passa a escrever código generalizado.

Em relação ao número de módulos a monitorizar/controlar na rede de domótica, inicialmente propostos, não foi possível cumprir com o esperado. A principal razão foi a

dificuldade na aquisição dos sensores necessários para a criação tanto do módulo de iluminação como do de segurança.

Após a conclusão deste projeto posso afirmar que a maioria dos objetivos mais relevantes para a aplicação foram atingidos. Apesar das adversidades encontradas, é com um sentimento de satisfação que finalizo esta etapa, em termos globais, pelo trabalho efetuado e pelas dificuldades de caracter académico ultrapassadas.

6.1 Melhoramentos futuros

Como qualquer outro projeto, existe um conjunto de processos a serem desenvolvidos com vista a melhorar globalmente o sistema implementado.

Em termos de desenvolvimentos futuros do sistema DomoCenter, gostaria de registar que podem ser melhorados os módulos desenvolvidos e implementados outros que são importantes para uma rede de domótica. No caso do módulo de temperatura, pode ser desenvolvido um sistema de climatização através do uso de mais sensores e também atuadores. Na categoria de segurança pode ser implementado um sistema de segurança com recurso a várias camaras de vigilância ligadas ao sistema DomoCenter.

Em ambas as aplicações (software) podem ser introduzidas mais funcionalidades e melhorado o aspeto visual. Por exemplo o caso da aplicação Web apresentar um ambiente customizado mais apelativo e moderno.

7 Referências

[1] http://www.electronica-pt.com/index.php/content/view/67/43/ - Acedido em Fevereiro de 2013.

[2] http://www.adene.pt/pt-pt/Publicacoes/Documents/GuiaEE_v1310.pdf - Acedido em Março de 2013.

[3] J. Ferreira, Interface homem-máquina para domótica baseada em tecnologias Web, 2008.

[4] http://home.planet.nl/~lhendrix/x10_history.htm - Acedido em Agosto de 2013

[5] http://digitat.info/modules/sections/index.php?op=viewarticle&artid=5 - Acedido em Agosto de 2013

[6] D. Cardoso, Domótica Inteligente – Um Contributo Prático, 2009.

[7] http://technologytwodotzero.wordpress.com/2011/03/12/ - Acedido em Outubro de 2013.

[8] *Echelon Corporation. Introduction to Lonworks System, Version 1.0.*

[9] http://hometoys.com/emagazine.php?url=/htinews/aug97/articles/kwacks/kwacks.htm – Acedido em Setembro de 2013.

[10] P. Matutino, Conceção e Desenvolvimento de uma Rede Domótica, 2001.

[11] *Microchip*, PIC18FXX8 *Data sheet*, 2004.

[12] Microchip, MCP2551 *Data sheet*, 2003.

[13] Microchip, PIC18F2455/2550/4455/4550 *Data sheet*, 2006.

[14] http://msdn.microsoft.com/en-us/vstudio/ – Acedido em Setembro de 2013.

[15] http://www.iis.net/learn – Acedido em Setembro de 2013.

[16] Advanced PIC18F Projects – CAN Bus projects, Cap. 9.

[17] www.18f4550.com – Acedido em Fevereiro de 2013.

[18] http://www.howtoasp.net/asp - net - security - tutorials/how – to – create – a-forms-authentication-custom-login-page-in-asp-net-in-vb-net/ – Acedido em Outubro de 2013.

[19] http://www.howtoasp.net/asp-net-security-tutorials/how-does-forms-authentication - work-in-asp-net/ – Acedido em Outubro de 2013.

[20] http://www.howtoasp.net/asp-net-security-tutorials/how-to-use-persistent -cookies-with-forms-authentication-in-asp-net-in-c/ – Acedido em Outubro de 2013.

[21] https://www.dns.pt/o-sistema-dns – Acedido em Outubro de 2013.

[22] Govoni, Darren: Java Application Frameworks, Wiley New York, 1999.

8 Anexos

8.1 Anexo1: *Boards* em *Eagle*

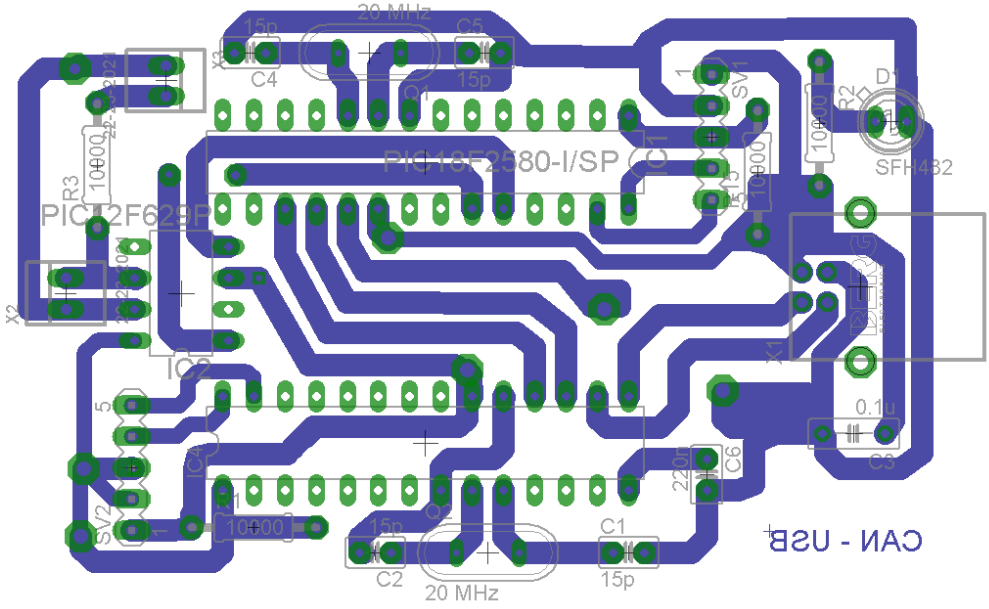

Figura 92 - Adaptador USB-CAN em *Eagle*

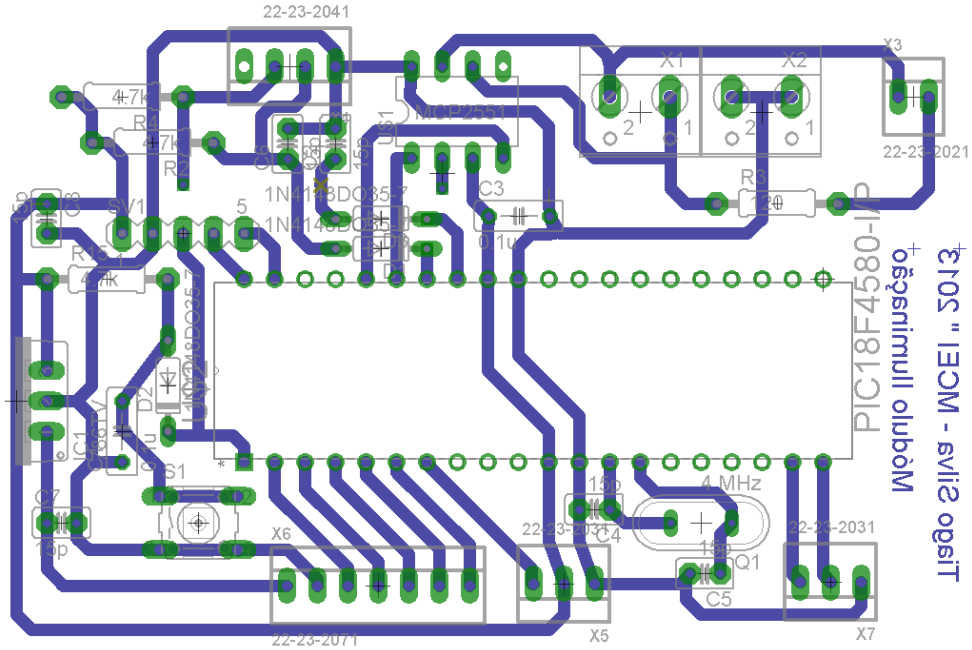

Figura 93 - Módulo de Iluminação em *Eagle*

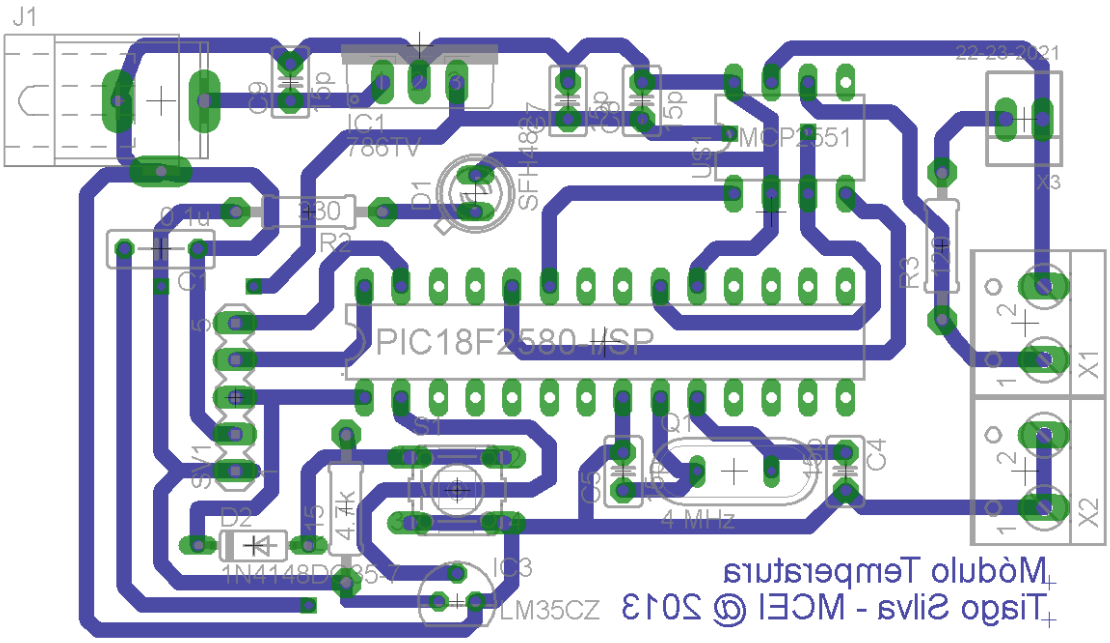

Figura 94 - Módulo de Temperatura em *Eagle*

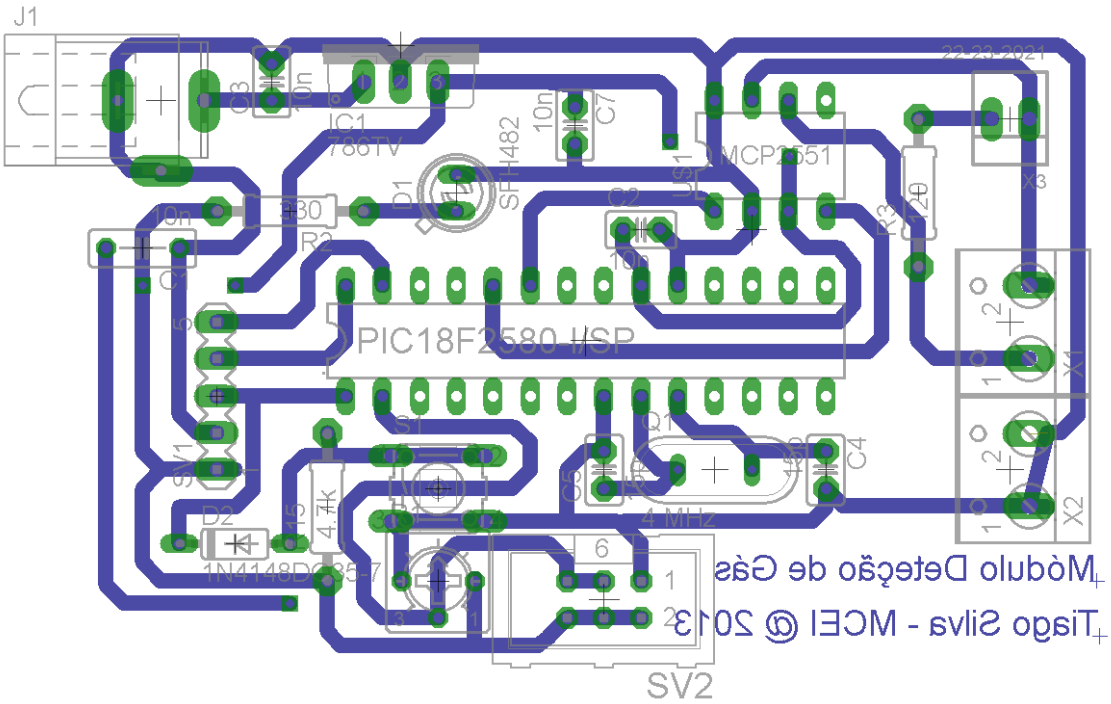

Figura 95 - Módulo de deteção de Gás em *Eagle*

8.2 Anexo 2 – Classes em C#

A- Classe *Data*

```
public class Data
    {

        // Método responsável por procurar um registo na tabela tdata dependendo do
        parâmetro de entrada ID
        public static DC.DAL.tData  Get(int ID)
        {
            using (DC.DAL.DCEntities bd = new DAL.DCEntities())
            {

              return bd.tDatas.FirstOrDefault(j => j.ID == ID);
            }
        }


     // Método que recebe os dados do USB e faz o update na tabela tData
        public static void Refresh(DC.BLL.USB.Proxy usb)
        {
            using (DC.DAL.DCEntities bd = new DAL.DCEntities())
            {
                //recebe o Byte Array do USB e chama-o 'newdata'
                byte[] newdata = DC.BLL.USB.Manager.Receber_USB(usb);

               //cria a lista 'data' para inserir na tabela tData
               List<DC.DAL.tData> data = (from td in bd.tDatas
                                          select td).ToList();

              // preenche todos os campos da tabela tData
               for (int i = 0; i < data.Count; i++)
                     data[i].Valor = newdata[i];

          bd.SaveChanges(); //grava a BD


            }
        }
    }
```

B- Classe *Queues*

```
public class Queues
   {
      // Método para inserir pedidos na tabela tQueues
      public static void Insert(int UserID, int OperationID, byte Value)
       {
           using (DC.DAL.DCEntities bd = new DAL.DCEntities())
           {
               DC.DAL.tQueue Queue = new DAL.tQueue();
               Queue.CreatedDate = DateTime.Now; //Data de criação do registo

               Queue.ID_Operation = OperationID; //ID da operação
               Queue.ID_User = UserID; //ID do User
               Queue.Value = Value; //Valor do registo

               bd.AddTotQueues(Queue); // Adiciona um novo registo
               bd.SaveChanges(); // grava a Base de dados
           }
       }

       // Método para despachar pedidos pendentes
       public static void Process(DC.BLL.USB.Proxy usb)
       {
           using (DC.DAL.DCEntities bd = new DAL.DCEntities())
           {

      // Vai à tQueues procurar os pedidos a processar com status a '0'

               List<DC.DAL.tQueue> queues = (from tq in bd.tQueues
                                             where tq.Status == 0
                                             select tq).ToList();

              foreach (DC.DAL.tQueue queue in queues) //para cada pedido
               {
                   try
                   {
                       List<DC.DAL.tDataSend > data = (from td in bd.tDataSends
                                                       select td).ToList();


                       switch (queue.tOperation.Name)
                       {

                           case "Definir Manual_Auto":
                               data[1].Valor = queue.Value;
                               break;


                          case "Definir Luminosidade":
                               data[2].Valor = queue.Value;
                               break;
```

```
                        case "On_Off":
                            data[3].Valor = queue.Value;
                            break;

                        case "Definir Posição Persiana":
                            data[4].Valor = queue.Value;
                            break;


                        default: break;
                    }

                    byte[] newdata = new byte[65];
                    for (int i = 0; i < data.Count; i++)
                        newdata[i] = data[i].Valor;

                    DC.BLL.USB.Manager.SendAll(newdata,usb);// Envia o array para o USB

                    queue.Status = 1; // Altera o estado para 1
                    queue.ProcessDate = DateTime.Now; // grava a data em que o registo
                    //foi processado
                     bd.SaveChanges();// Grava a BD
                }
                catch (Exception)
                {

                    queue.Status = 10;
                    queue.ProcessDate = DateTime.Now;
                    bd.SaveChanges();
                }
            }
        }
    }
}
```

C- Classe *Manager*

```
public class Manager
    {

        /// <summary>
        /// Efectua a ligação
        /// </summary>
        /// <returns>Devolve um objecto do tipo Proxy já com a ligação
        efectuada, ou então um erro para ser tratado no método pai</returns>
        public static Proxy Connect()
        {
            Proxy usb = new Proxy();
              // attempt to connect to USB board
            usb.connectionState = usb.attemptUSBConnection();

            if (usb.connectionState == Proxy.CONNECTION_SUCCESSFUL)
                return usb;
            else throw new Exception("Ligação não estabelecida!");

        }

       // Função que recebe dados do USB
       public static byte[] Receber_USB(Proxy usb)
        {
            usb.connectionState = usb.attemptUSBConnection();

            usb.receiveViaUSB();

            return usb.fromDeviceToHostBuffer;

        }

       // Método para enviar para o USB
       public static void SendAll(byte[] data, Proxy usb)
        {
            usb.connectionState = usb.attemptUSBConnection();
            usb.fromHostToDeviceBuffer = data;
            usb.sendViaUSB();
        }
}
```

D- Classe *Users*

```
public class Users
    {
       // Método para validar as credenciais
       public static DC.DAL.tUser Validate(string Loginname, string Password)
        {
            using(DC.DAL.DCEntities bd = new DAL.DCEntities())
            {
             return bd.tUsers.FirstOrDefault(a => a.Loginname == Loginname &&
             a.Password == Password);

            }
        }
    }
```

E- Classe *Logs*

```
public class Logs
    {
        // Método para criar os registos de Logs na tabela tLog
        public static void Insert(string Message, DC.DAL.tUser User)
        {

             using(DC.DAL.DCEntities bd = new DAL.DCEntities())
            {
                DC.DAL.tLog log = new DAL.tLog();
                log.CreatedDate = DateTime.Now;

                if(User != null)
                    log.ID_User = User.ID;
                    log.Message = Message;

                bd.AddTotLogs(log);
                bd.SaveChanges();


            }
        }
    }
```