Universidade de Trás-os-Montes e Alto Douro

Sistema Integrado de Comunicações para uma Rede de Monitorização e Controlo de Tráfego

Dissertação de Mestrado em Engenharia Electrotécnica e de Computadores

Dário Jorge dos Santos

Orientador: Doutor Pedro Miguel Mestre Alves da Silva

Co-orientador: Doutor Carlos Manuel José Alves Serôdio

Vila Real, 2014

Sistema Integrado de Comunicações para uma Rede de Monitorização e Controlo de Tráfego

Por

Dário Jorge dos Santos

Orientador: Doutor Pedro Miguel Mestre Alves da Silva

Co-orientador: Doutor Carlos Manuel José Alves Serôdio

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
MESTRE
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
DR – I série – A, Decreto-Lei n.º 74/2006 de 24 de Março com as alterações introduzidas pelos
Decretos-Leis n.o 107/2008, de 25 de Junho, e 230/2009, de 14 de Setembro, e demais legislação
aplicável e no Regulamento de Ciclo de Estudos Conducente ao Grau de Mestre da
Universidade de Trás-os-Montes e Alto Douro
DR, 2.ª série – n.º 149/2011 de 4 de Agosto

Sistema Integrado de Comunicações para uma Rede de Monitorização e Controlo de Tráfego

Por

Dário Jorge dos Santos

Orientador: Doutor Pedro Miguel Mestre Alves da Silva

Co-orientador: Doutor Carlos Manuel José Alves Serôdio

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
MESTRE
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
DR – I série – A, Decreto-Lei n.º 74/2006 de 24 de Março com as alterações introduzidas pelos
Decretos-Leis n.o 107/2008, de 25 de Junho, e 230/2009, de 14 de Setembro, e demais legislação
aplicável e no Regulamento de Ciclo de Estudos Conducente ao Grau de Mestre da
Universidade de Trás-os-Montes e Alto Douro
DR, 2.ª série – n.º 149/2011 de 4 de Agosto

Orientação Científica :

Doutor Pedro Miguel Mestre Alves da Silva

Professor Auxiliar do
Departamento de Engenharias da Escola de Ciências e Tecnologia
Universidade de Trás-os-Montes e Alto Douro

Doutor Carlos Manuel José Alves Serôdio

Professor Associado com Agregação do
Departamento de Engenharias da Escola de Ciências e Tecnologia
Universidade de Trás-os-Montes e Alto Douro

Acompanhamento do trabalho :

José Carlos Fernandes

Douro Alliance

"Não há problema em não saber todas as respostas, é melhor admitir a nossa ignorância que acreditar em respostas que possam estar erradas. Fingindo saber tudo, fechamos a porta à descoberta do que realmente está lá."

Neil deGrasse Tyson

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Mestrado em Engenharia Electrotécnica e de Computadores

Os membros do Júri recomendam à Universidade de Trás-os-Montes e Alto Douro a aceitação da dissertação intitulada "**Sistema Integrado de Comunicações para uma Rede de Monitorização e Controlo de Tráfego**" realizada por **Dário Jorge dos Santos** para satisfação parcial dos requisitos do grau de **Mestre**.

Dezembro 2014

Presidente: **Doutor José Paulo Barroso de Moura Oliveira**,
Professor Associado com Agregação do Departamento de Engenharias da Escola de Ciências e Tecnologia da Universidade de Trás-os-Montes e Alto Douro

Vogais do Júri: **Doutor José Araújo Mendes**,
Professor Auxiliar do Departamento de Eletrónica Industrial da Escola de Engenharia da Universidade do Minho

Doutor Pedro Miguel Mestre Alves da Silva,
Professor Auxiliar do Departamento de Engenharias da Escola de Ciências e Tecnologia da Universidade de Trás-os-Montes e Alto Douro

Sistema Integrado de Comunicações para uma Rede de Monitorização e Controlo de Tráfego

Dário Jorge dos Santos

Submetido na Universidade de Trás-os-Montes e Alto Douro
para o preenchimento dos requisitos parciais para obtenção do grau de
Mestre em Engenharia Electrotécnica e de Computadores

Resumo — Torna-se cada vez mais difícil falar de sustentabilidade urbana sem abordar o problema do tráfego rodoviário. É reconhecida a falta de mobilidade que existe quer nas deslocações quer no ato de estacionar, daí a importância de se criarem sistemas que visem facilitar a circulação rodoviária em cidades. Uma das possíveis soluções passa pela monitorização dos lugares de estacionamento recorrendo a tecnologias de redes de comunicação. O CAN e o IEEE 802.15.4 são dois protocolos de comunicação que se têm revelado bastante flexíveis na implementação de sistemas de monitorização e controlo. Tendo por base estes protocolos, foi implementado um sistema de comunicações que permite uma monitorização dos lugares de estacionamento em áreas urbanas. Ao longo do documento é apresentado o processo de implementação do sistema, utilizando cada um dos protocolos, assim como o desenvolvimento do *hardware*, *firmware* e *software* necessários ao seu funcionamento.

Palavras Chave: Tráfego Rodoviário, Monitorização de Estacionamentos, Redes de Comunicação, CAN, IEEE 802.15.4.

Integrated Communication System for Network Traffic Monitoring and Control

Dário Jorge dos Santos

Submitted to the University of Trás-os-Montes and Alto Douro
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering and Computers

Abstract — It is almost impossible to talk about urban sustainability without addressing the problem of road traffic. As it is well known there is a lack of mobility, both during the displacements and at the time of parking. Therefore it is important to develop systems that aim to help road traffic in cities. One possible solution involves the monitorization of parking spaces using communication networks technologies. CAN and IEEE 802.15.4 communication protocols have been quite flexible for implementing monitorization and control systems. Based on these protocols, a communications system which allows to monitor parking spaces in urban areas has been developed. Throughout this document it is presented the implementation of the system, using these protocols, as well as the development of the hardware, firmware and software needed for the correct system operation.

Key Words: Road Traffic, Monitorization of Parking Spaces, Communications Networks, CAN, IEEE 802.15.4.

Agradecimentos

Quero expressar os meus agradecimentos ao Magnífico Reitor da Universidade de Trás-os-Montes e Alto Douro, Professor Doutor António Augusto Fontaínhas Fernandes, por institucionalmente ter permitido a realização dos meus estudos conducentes ao grau de Mestre.

Ao Professor Doutor Pedro Miguel Mestre Alves da Silva, Professor Auxiliar do Departamento de Engenharias da Escola de Ciências e Tecnologia da Universidade de Trás-os-Montes e Alto Douro, orientador deste trabalho, pela sua motivação, pelas suas sugestões, ideias inovadoras e orientações.

Ao Professor Doutor Carlos Manuel José Alves Serôdio, Professor Associado com Agregação do Departamento de Engenharias da Escola de Ciências e Tecnologia da Universidade de Trás-os-Montes e Alto Douro, na qualidade de co-orientador, pelas suas observações e orientações.

À direção, e restantes docentes, do curso de Engenharia Electrotécnica e de Computadores da Universidade de Trás-os-Montes e Alto Douro, pela disponibilidade e simpatia demonstradas ao longo de todo o meu percurso académico nesta instituição.

A todos os meus colegas do curso de Engenharia Electrotécnica e de Computadores da Universidade de Trás-os-Montes e Alto Douro pela sua amizade e simpatia.

Aos meus colegas de trabalho, pelas suas opiniões e ajudas que em muito contribuíram para o crescimento deste projeto.
Aos meus pais e familiares, pelo apoio incondicional que me deram, sem o qual teria sido impossível a caminhada até aqui.
À minha namorada, Sara Rafaela, pela sua dedicação caraterística e palavras de incentivo e motivação que ajudam a superar-me diariamente.

A todos, um sincero obrigado!

UTAD, Dário Jorge dos Santos

Índice geral

Índice de tabelas

Índice de figuras

Glossário, acrónimos e abreviaturas

Glossário de termos

Host — Um *host* pode ser descrito como um nó da rede que permite oferecer informações, recursos, serviços e aplicações aos utilizadores e a outros nós.

Ad-hoc — Rede de comunicação que não possui um nó base, normalmente designado de *Access Point* (AP), cujas funções seriam de encaminhar as comunicações para o respetivo destino. No modo ad-hoc, essas funções ficam ao encargo dos dispositivos na rede.

Broadcast — Processo pelo qual se transmite determinada informação para todos os possíveis recetores no mesmo intervalo de tempo.

Checksum — Código utilizado para verificar a integridade dos dados transmitidos através de algum meio com presença de ruído.

Firmware — Conjunto de instruções operacionais programadas diretamente no *hardware* de um equipamento eletrónico.

Ponto-a-Ponto — Arquitetura de redes de comunicação onde cada um dos nós

na rede assume tanto o papel de cliente como de servidor, permitindo a transferência de dados sem a existência de um servidor central.

Lista de acrónimos

Sigla	**Expansão**
ACL	*Asynchronous Connection-Less*
AP	*Access Poit*
API	*Application Programming Interface*
BSS	*Basic Service Set*
CAN	*Controller Area Network*
CCK	*Complementary Code Keying*
CCO	Centro de Controlo e Operação
CPS	*Common Part Sublayer*
CRC	*Cyclic Redundancy Check*
CS	*Convergence Sublayer*
CSMA/CA	*Carrier Sense Multiple Access With Collision Avoidance*
CSMA/CD	*Carrier Sense Multiple Access With Collision Detection*
DCF	*Distributed Coordination Function*
DLC	*Data Length Code*
DSSS	*Direct Sequence Spread Spectrum*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
EOF	*End of Frame*
ESS	*Extended Service Set*
FDL	*Fieldbus Data Link*
FHSS	*Frequency Hopping Spread Spectrum*

Sigla	**Expansão**
FSK	*Frequency-Shift Keying*
GTS	*Guaranteed Time Slots*
HER	*Header Error Control*
HR-DSSS	*High Rate-Direct Sequence Spread Spectrum*
HTML	*HyperText Markup Language*
IDE	*Identifier Extension*
IEEE	*Institute of Electrical and Electronic Engineers*
IFS	*Interframe Space*
IP	*Internetworking Protocol*
ISO	*International Organization for Standardization*
JSP	*JavaServer Pages*
LAN	*Local Area Network*
LED	*Light Emitting Diode*
LLC	*Logic Link Control*
LOS	*Line of Sight*
MAC	*Media Access Control*
MAN	*Metropolitan Area Network*
MFR	*MAC Footer*
MHR	*MAC Header*
MSDU	*MAC Service Data Unit*
MPDU	*MAC Protocol Data Unit*
OFDM	*Orthogonal Frequency-Division Multiplexing*
OSI	*Open Systems Interconnection*
PAN	*Personal Area Network*
PCB	*Printed Circuit Board*
PCF	*Point Coordination Function*
PHR	*Physical Header*

Sigla	Expansão
PPDU	*Physical Protocol Data Unit*
PPM	*Pulse Position Modulation*
PROFIBUS	*Process Field Bus*
PSDU	*Physical Service Data Unit*
PSK	*Phase-Shift Keying*
RSSI	*Received Signal Strength Indicator*
RTR	*Remote Transmission Request*
SCO	*Synchronous Connection-Oriented*
SOF	*Start of Frame*
SHR	*Synchronization Header*
TCP	*Transmission Control Protocol*
TDD	*Time Division Duplex*
UDP	*User Datagram Protocol*
WAN	*Wide Area Network*
WLAN	*Wireless Local Area Network*
WMAN	*Wireless Metropolitan Area Network*
WPAN	*Wireless Personal Area Network*
XML	*Extensible Markup Language*

Lista de abreviaturas

Abreviatura	Significado(s)
e.g.	por exemplo
et al.	e outros (autores)

(continua na página seguinte)

(continuação)

Abreviatura	**Significado(s)**
etc.	etecetera, outros

1 Introdução

A busca por melhores condições de vida, fez com que muitos habitantes deixassem os seus lares em zonas rurais para se estabelecerem em zonas urbanas. Nestes centros urbanos, as pessoas procuram essencialmente oportunidades profissionais que são sem dúvida mais abundantes. No entanto, este êxodo rural pode criar cidades insustentáveis, com excesso de população, que muitas vezes retira a qualidade de vida que muitos procuravam alcançar.

Um dos fatores que melhor descreve esta insustentabilidade é o tráfego rodoviário (Álvaro Jorge da Maia Seco et al., 2008). Os habitantes deste tipo de cidades, enfrentam este problema diariamente quer nas suas deslocações entre a periferia e a área metropolitana quer nas dificuldades de estacionamento, cujo tempo requerido é por vezes superior ao tempo gasto na tarefa que originou a deslocação.

Apesar de vivermos numa era caracterizada pelo alto débito das comunicações, a falta de mobilidade em meios urbanos é um dos fatores que mais afeta negativamente a vida das pessoas, acarretando consigo consequências a nível social e ambiental que tornam o tráfego urbano num dos principais fatores para o abandono das cidades e o regresso às zonas rurais (Álvaro Jorge da Maia Seco et al., 2008).

1.1 Motivação e objetivos

Todos reconhecemos os esforços que se têm feito para minimizar o tráfego rodoviário nas cidades, sobretudo pela aposta nos meios de transporte público e no reforço da rede rodoviária urbana, no entanto, ainda existe um longo caminho a percorrer para se conseguir a tão aclamada sustentabilidade citadina.

Numa tentativa de fazer face a este problema, surge o presente trabalho enquadrado no projeto de uma *Smart City*, desenvolvido pela Universidade de Trás-os-Montes e Alto Douro e promovido pela associação Douro Alliance. Esse projeto tem como objetivo criar várias ferramentas que tornem possível a agregação de serviços dentro de uma cidade, tentando elevar substancialmente a qualidade de vida na mesma.

Um sistema integrado de comunicações de suporte a uma rede de monitorização e controlo de tráfego, enquadrar-se-ia nesta temática, tornando possível um vasto conjunto de comunicações dinamizadoras do fluxo de tráfego urbano. Este seria o ponto de partida do trabalho, que se centrou na monitorização de estacionamentos como caso de estudo.

O desenvolvimento de um sistema capaz de monitorizar e sinalizar o estado dos estacionamentos rodoviários, permite maior mobilidade e agilidade no momento de estacionar, o que necessariamente irá facilitar a circulação rodoviária.

O presente trabalho tem como objetivo o desenvolvimento de um sistema de comunicações que permite a troca de mensagens entre os nós presentes nos estacionamentos e as aplicações de controlo e monitorização. Para a implementação desta solução recorre-se a tecnologias de redes de comunicação como o CAN e o IEEE 802.15.4, que são uma mais valia pelo seu baixo custo, simplicidade de implementação, e apresentam a versatilidade necessária para este tipo de aplicação. É feito o dimensionamento dos nós terminais de comunicação, que monitorizam o estado do estacionamento, assim como os nós de interface, que funcionam como intermediários entre os nós terminais e as aplicações de controlo. Estas, têm como objetivo analisar a informação recebida e responder com a informação de sinalização correspondente.

A informação do estado dos estacionamentos é ainda guardada numa base de dados, e pode ser utilizada para apresentação dos lugares de estacionamento livres em determinados locais, o que se tornará numa mais valia para os condutores no ato de estacionar.

1.2 Organização da dissertação

Esta dissertação encontra-se estruturada em cinco capítulos. No presente Capítulo fez-se uma introdução de enquadramento, apresentou-se o problema, a motivação e os objetivos do trabalho.

O Capítulo 2 inicia-se com uma abordagem às redes de comunicação, onde se enumeram os seus diferentes tipos e modelos de suporte. Seguidamente faz-se um levantamento de alguns protocolos de comunicação, sejam eles *wireless* ou protocolos guiados industriais. O Capítulo é finalizado com uma análise comparativa destes protocolos.

A arquitetura do sistema proposto é apresentada no Capítulo 3, onde se descreve a sua conceção e implementação. Aqui são abordados os protocolos utilizados, bem como as implementações de *hardware*, *firmware* e *software* correspondentes.

Os testes realizados ao sistema, bem como os resultados obtidos, são apresentados no Capítulo 4.

A finalizar a dissertação, no Capítulo 5, faz-se uma análise dos resultados obtidos e apresentam-se algumas perspetivas de trabalho futuro.

2 Tecnologias de Redes de Comunicação

Quando olhamos pinturas e gravuras rupestres percebemos a necessidade que os humanos pré-históricos tinham de registar acontecimentos e transformações que viviam, tornando este tipo de arte numa das mais antigas fontes de informação conhecidas. Hoje em dia, a nossa apetência pelo registo de informação mantêm-se e vivemo-la na sua plenitude. A nossa era é considerada a era da informação, e deve este título à Internet que permitiu trocas de informação a uma escala global num ritmo nunca antes visto. Este sistema de trocas de informação tão difundido na nossa sociedade tem por base um processo relativamente simples denominado em engenharia por comunicação de dados.

Chamamos comunicação de dados à troca de informação entre dois dispositivos através de algum meio, por exemplo, fios e ar. Para que a comunicação aconteça, é necessário que os dispositivos envolventes façam parte de um sistema que combine *hardware* e *software*. Sempre que temos mais de dois dispositivos deste tipo, ligados entre si, dizemos que temos uma rede de comunicação (Forouzan, 2008).

As redes podem ser classificadas segundo a sua área de cobertura em PAN, LAN, MAN e WAN. Uma PAN (*Personal Area Network*) é uma rede pessoal, cujos dispositivos se encontram muito próximos uns dos outros, normalmente na ordem da

dezena de metros (Fernandes, 2011). Uma LAN (*Local Area Network*) é uma rede local, que interliga dispositivos por exemplo num escritório ou prédio, normalmente numa área geográfica inferior a 3 km. Uma MAN (*Metropolitan Area Network*) é uma rede intermediária normalmente usada na área metropolitana de uma cidade. Por último, a rede global WAN (*Wide Area Network*) destina-se a uma grande área geográfica, como um país ou continente (Forouzan, 2008).

2.1 Modelos de Comunicação

Quando criadas por diferentes entidades, as redes de comunicação poderiam tornam-se incompatíveis entre si, daí a importância dos modelos de rede, definidos em camadas cujas tarefas tornam possível a comunicação entre redes heterógeneas e os seus dispositivos. Existem dois grandes modelos de rede, o OSI (*Open Systems Interconnection*), modelo utilizado como referência, e o conjunto de protocolos TCP/IP, que apesar de anterior ao modelo OSI, se tornou a arquitetura predominante devido a ser usado e testado intensamente na Internet (Forouzan, 2008).

2.1.1 Modelo OSI

Criado pela ISO (*International Organization for Standardization*) no final da década de 1970, o modelo OSI tem como objetivo facilitar a comunicação entre sistemas diferentes, independentemente da lógica de *hardware* e *software* subjacentes a cada um deles (Forouzan, 2008).

Este modelo é constituído por sete camadas relacionadas entre si, cujo objetivo é lidar com o processo de transferência de informação através da rede. Na figura 2.1 temos dois dispositivos que comunicam entre si, o dispositivo A é o emissor, e o dispositivo B é o recetor. A comunicação tem inicio nas camadas do emissor, onde cada uma delas acrescenta as suas informações à mensagem que recebe da camada superior e a entrega à camada inferior. Na camada física, a *frame* é convertida em *bits* para que possa ser transmitida ao recetor. Camada por camada a mensagem é

aberta, são retiras as informações correspondestes a cada uma delas e o restante é enviado para a camada superior (Forouzan, 2008).

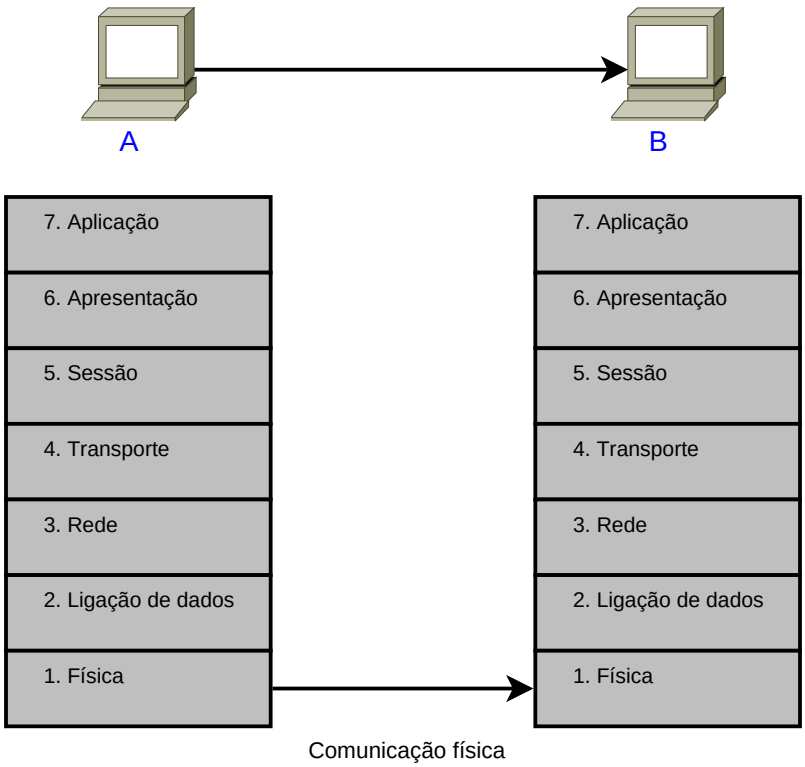

Figura 2.1 – As camadas do modelo OSI.

Cada camada do modelo OSI desempenha as seguintes funções (Forouzan, 2008):

- A **Camada Física** é responsável pela transmissão de *bits* pelo meio físico. Também provê as especificações mecânicas e elétricas;
- A **Camada de Ligação de Dados** organiza os *bits* em *frames*, e faz a sua entrega nó a nó;
- A **Camada de Rede** tem como missão transmitir os pacotes individuais desde o *host* de origem até ao *host* de destino;
- A **Camada de Transporte** está encarregue da transmissão confiável de uma mensagem processo a processo. Lida também com a recuperação de erros;

- A **Camada de Sessão** é o controlador de diálogo da rede. Estabelece, mantém e sincroniza a interação entre os sistemas em comunicação;
- A **Camada de Apresentação** ocupa-se da tradução, criptografia e compressão dos dados;
- A **Camada de Aplicação** possibilita o acesso a serviços por parte das aplicações do utilizador.

2.1.2 Conjunto de Protocolos TCP/IP

Desenvolvido antes do modelo OSI, o conjunto de protocolos TCP/IP está definido em quatro camadas: Acesso à Rede, Internet, Transporte e Aplicação. Lidando com transmissões no meio físico, podemos dizer que a primeira camada é equivalente às camadas Física e Ligação de Dados do modelo OSI. As camadas Internet e Transporte correspondem às camadas Rede e Transporte do modelo OSI, e lidam com interfaces de rede, ligação entre redes e transporte. A camada da Aplicação, realiza, a grosso modo, as funções das últimas três camadas do modelo OSI (Forouzan, 2008).

O TCP/IP é um conjunto de protocolos hierárquicos com funções específicas. Na camada Internet, o TCP/IP suporta o IP (*Internetworking Protocol*), que se encarrega de transmitir uma mensagem até ao seu destino, contudo, sem garantia que o conseguirá fazer. Na camada de Transporte suporta dois protocolos, o UDP (*User Datagram Protocol*) e o TCP (*Transmission Control Protocol*), que são responsáveis pela entrega de uma mensagem de processo a processo (Forouzan, 2008).

Existem três tipos de endereços no TCP/IP: endereços físicos, endereços lógicos e endereços de portas. O endereço físico, usado na camada Acesso à Rede, é o endereço de um nó definido pela sua rede. O endereço lógico, usado na camada Internet, define exclusivamente um *host*, independentemente da rede física subjacente. Por último, o endereço de porta identifica um processo num *host* e é utilizado na camada de Transporte (Fourozan and Fegan, 2008).

2.2 Protocolos de Comunicação sem fios

Os modelos de rede abordados até aqui, possibilitam o fluxo de dados entre redes e os respetivos dispositivos, cuja arquitetura em camadas serve como referência para diversos protocolos de comunicação *wireless*. Dentre estes, abordar-se-á os protocolos IEEE 802.11, IEEE 802.15.1, IEEE 802.15.4 e IEEE 802.16.

2.2.1 IEEE 802.11

Hoje em dia estamos mais que habituados a aceder à Internet onde quer que estejamos por intermédio de redes WLAN (*Wireless Local Area Network*). Através dos nossos dispositivos, fixos ou móveis, temos Internet em casa, no local de trabalho, na escola, na universidade, nos locais públicos como cafés, hotéis, aeroportos, centros comerciais e até já algumas avenidas citadinas e praias.

Através do protocolo IEEE 802.11, aprovado em 1997 pelo *Institute of Electrical and Electronic Engineers* (IEEE), o conhecido *WiFi* define especificações para implementar redes LAN sem fio (WLAN). A comunicação é feita através de ondas eletromagnéticas que se propagam no ar, o que minimiza os elevados custos com fios, dando uma grande mobilidade à rede e aos seus utilizadores (Fernandes, 2011). Na realidade o IEEE 802.11 é constituído por um conjunto de padrões, caracterizados pela frequência de operação, técnicas de modulação e taxas de transmissão de dados, como o IEEE 802.11a, IEEE 802.11b, IEEE 802.11g e IEEE 802.11n (da Silva Carissimi et al., 2009).

Arquiteturas do IEEE 802.11

São definidos dois tipos de arquitetura para o protocolo IEEE 802.11, a *basic service set* (BSS) e a *extended service set* (ESS). A BSS define a organização fundamental de uma rede sem fios, e é composta por duas ou mais estações *wireless* ligadas entre si formando uma rede instantânea e isolada a que chamamos ad-hoc. No caso de

também possuir uma estação base, denominada AP (*Access Point*), temos uma rede de infraestrutura. A ESS engloba duas ou mais arquiteturas BSSs, com a presença de vários AP, conectadas por intermédio de um sistema de distribuição, tipicamente uma rede LAN. Esta arquitetura permite a comunicação entre os dispositivos dentro da BSS, e a comunicação entre diferentes BSSs com recurso aos APs (Forouzan, 2008). A figura 2.2 ilustra as duas arquiteturas referidas.

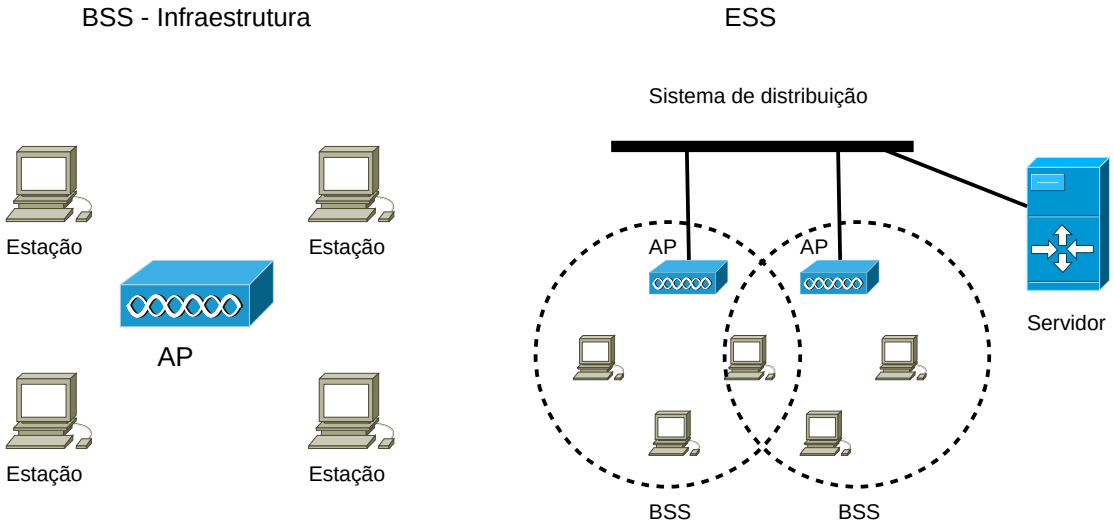

Figura 2.2 – Arquiteturas de serviço básico (BSS) e alargado (ESS).

Camada física

O protocolo IEEE 802.11 define especificações para a conversão dos *bits* em sinais na camada física. Essas especificações baseiam-se em diversas técnicas de codificação, sendo elas, FHSS (*Frequency Hopping Spread Spectrum*), DSSS (*Direct Sequence Spread Spectrum*) e OFDM (*Orthogonal frequency-division multiplexing*) (Forouzan, 2008).

- O **IEEE 802.11 FHSS** usa o método de espalhamento espectral por saltos em frequência. Utiliza a faixa de frequência 2,4 GHz que é dividida em 79 sub-faixas de 1 MHz. A sequência de saltos é selecionada por um gerador de números pseudo-aleatórios. A técnica de modulação usada é o

FSK (*Frequency-Shift Keying*) de dois níveis, ou de quatro níveis com 1 ou 2 bits/baud, o que origina velocidades totais de 1 ou 2 Mbps (Forouzan, 2008);

- O **IEEE 802.11 DSSS** usa o método de espalhamento espectral de sequência direta na faixa de frequências de 2,4 GHz. A técnica de modulação utilizada é o PSK (*Phase-Shift Keying*) a 1 Mbaud/s. O sistema permite 1 ou 2 bits/baud que origina velocidades totais de 1 ou 2 Mbps. O padrão IEEE 802.11b usa o método HR-DSSS (*High Rate*-DSSS), similar ao DSSS, exceto a codificação, pois é usado o CCK (*Complementary Code Keying*) que codifica 4 ou 8 *bits* por símbolo CCK. O HR-DSSS define quatro velocidades de transmissão de dados: 1, 2, 5.5 e 11 Mbps (Forouzan, 2008);

- O **IEEE 802.11 OFDM** utiliza o método ortogonal de multiplexação por divisão de frequência, para geração de sinais nas faixas de frequências 2,4 ou 5 GHz. Todas as sub-faixas são usadas simultaneamente pela mesma estação de origem num dado instante. As estações disputam entre si o acesso ao meio físico de transmissão na camada de Ligação de Dados. A banda é dividida em sub-faixas para transmissão de *bits* e informações de controlo, o que reduz efeitos de interferência e aumenta a segurança. As velocidades típicas para transmissão de dados são 18 e 54 Mbps. O padrão IEEE 802.11a usa o OFDM na faixa de frequências 5 GHz enquanto que o IEEE 802.11g opera na frequência 2,4 GHz (Forouzan, 2008).

Sub-camada de controlo de acesso ao meio

A segunda camada do modelo OSI, Ligação de Dados, é na realidade composta de duas sub-camadas, MAC (*Media Access Control*) e LLC (*Logic Link Control*). Por sua vez, o IEEE 802.11 define dois modos de controlo de acesso ao meio, na sub-camada MAC, o DCF (*Distributed coordination function*) e o PCF (*Point coordination function*). O DCF usa o método de acesso CSMA/CA (*Carrier sense multiple access with collision avoidance*), onde cada estação tem acesso igual ao meio e analisa-o antes de enviar dados. Para evitar o problema do terminal oculto,

caso de duas estações que tentam enviar simultaneamente dados para uma terceira, mas que, devido a alguma obstrução no meio, não se ouvem e ocorre uma colisão de pacotes, cada estação define um período de tempo de acesso ao meio e informa à outra para que não envie dados nesse período. O PCF é um método de acesso opcional, que é implementado sobre o DCF apenas em redes de infraestrutura, para transmissão de dados sensíveis a atrasos. É feito um varrimento (*polling*) em todas as estações, de modo que estas enviem para o AP quaisquer dados que possuam. Uma estação deste tipo, tem prioridade no acesso ao meio em relação a uma estação que apenas use o DCF (Fourozan and Fegan, 2008).

Existem três tipos de *frames* no protocolo IEEE 802.11, *frames* de gestão, de controlo e de dados. De maneira geral, as *frames* da sub-camada MAC são constituídas pelos seguintes campos (Fourozan and Fegan, 2008):

- **Controlo de *frame* (FC)**. Com 2 *bytes* de comprimento, define o tipo de *frame* e algumas informações de controlo;
- **D**. Este campo define a duração da transmissão. Pode ainda definir o ID da *frame*, caso seja de controlo;
- **Endereços**. Existem quatro campos para endereços, cada um com 6 *bytes* de comprimento;
- **Controlo de sequência (SC)**. Usando o controlo de fluxo, este campo define o número de sequência da *frame*;
- **Corpo da *frame***. Pode ter até 2.312 *bytes* de comprimento e contém informações baseadas no tipo de *frame* definido no FC;
- **FCS**. Com 4 *bytes* de comprimento, este campo contém uma sequência de deteção de erros.

2.2.2 IEEE 802.15.1

O conhecido Bluetooth, cujo nome se deve a um rei viking, é uma tecnologia wireless PAN (WPAN), que surgiu devido à necessidade de conectar telefones móveis a computadores e outros equipamentos pessoais. A sua primeira versão foi publicada em 1999, e então o grupo de trabalho do IEEE 802.15 o adotou como sendo o padrão IEEE 802.15.1, onde especificou as camadas Física e Ligação de Dados (Coulouris et al., 2007).

Este protocolo, que opera na faixa de frequência 2.4 GHz, foi projetado para permitir a comunicação sem fios entre pequenos dispositivos, de baixo custo e consumo. Inicialmente permitia taxas de transmissão de dados de 1 Mbps, mas versões posteriores aumentaram essa taxa, primeiro para 2,1 Mbps, e depois para 24 Mbps. A última versão desenvolvida é dedicada ao mercado de aplicações de baixa potência. Tipicamente os dispositivos Bluetooth têm uma potência de saída entre 1 e 100 miliwatt, e áreas de cobertura entre 10 e 100 metros (Gupta, 2013). De modo a minimizar interferências, os saltos em frequência ocorrem a uma taxa de 1600 vezes por segundo entre 79 sub-faixas de 1 MHz da faixa de frequência permitida (Coulouris et al., 2007).

Arquitetura do IEEE 802.15.1

No Bluetooth, os nós associam-se dinamicamente em pares, sem pré-conhecimento da rede. Após efetuada a associação, o nó iniciador tem a função de mestre, e o outro nó assume o papel de escravo. Sempre que uma rede é composta de um mestre e escravos ativos, no máximo sete, temos uma Piconet. O mestre controla o uso do canal de comunicações, alocando períodos de tempo para cada escravo. Nós que estejam presentes em mais de uma Piconet, podem atuar como ponte de comunicação entre mestres. Desta forma, duas ou mais Piconets ligadas entre si dão origem a uma Scatternet (Coulouris et al., 2007).

Para entrar em contacto com um nó desconhecido, o nó iniciador transmite uma

sequência de mensagens de procura. Estabelecida a ligação, é feita uma autenticação, para garantir que a associação se dá com o nó pretendido. O escravo mantém-se sincronizado com o mestre, mas caso esteja inativo, pode ser relegado para um estado estacionário, libertando assim uma posição ativa na Piconet. Podem existir até 255 nós no estado estacionário numa Piconet. Neste modo de baixo consumo energético, os nós continuam à espera de um sinal de ativação por parte do mestre (Coulouris et al., 2007).

A comunicação síncrona, entre mestre e escravos, é garantida pelo uso do procolo de comunicação bidirecional SCO (*Synchronous connection-oriented*), utilizado frequentemente na transmissão de voz. Já a comunicação assíncrona é garantida pelo protocolo ACL (Asynchronous Connection-Less), onde o mestre envia *frames* de consulta periódicas, e os escravos somente transmitem após a sua receção. Este protocolo é utilizado para a transmissão de pacotes de dados (Coulouris et al., 2007).

Formato da *Frame*

O formato típico das *frames* no IEEE 802.15.1 é mostrado na figura 2.3. O código de acesso contém, além de *bits* de sincronização, a identificação da estação mestre, que permite diferenciar *frames* vindas de diferentes Piconets. Como o meio de transmissão pode conter ruído, e a comunicação em tempo real não conta com retransmissões, cada *bit* do campo cabeçalho é transmitido de forma triplicada e contém: O endereço, que define até sete estações escravo, caso seja zero é um *broadcast*; O tipo, que define os dados provenientes de camadas superiores; O sub-campo F, que se destina ao controlo do fluxo, quando ativo, indica que o dispositivo não pode receber mais *frames*; O sub-campo A, que se destina à confirmação de mensagens; O sub-campo S, que define o número de sequência; O HEC (*Header Error Control*) que permite detetar erros no cabeçalho recorrendo à técnica *checksum*; O último campo da *frame*, contem os dados e informações de controlo das camadas superiores e é denominado de Payload (Forouzan, 2008).

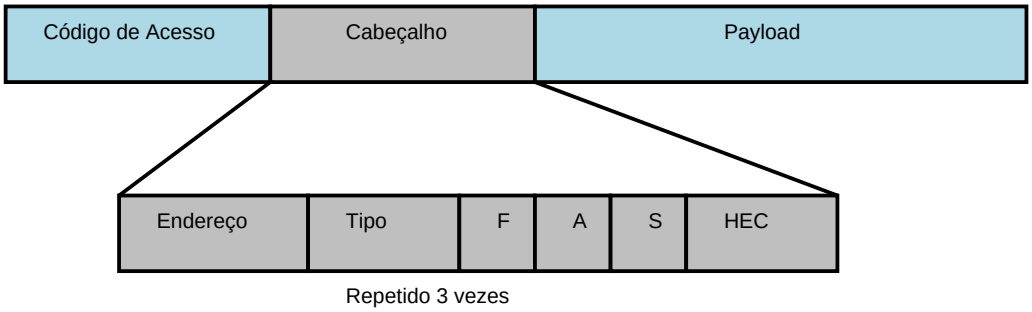

Figura 2.3 – Formato de uma *frame* no IEEE 802.15.1.

2.2.3 IEEE 802.15.4

O IEEE 802.15.4 é um padrão para comunicações sem fios (WPAN), que se carateriza pela baixa potência de operação, resultando no baixo consumo de energia por parte dos dispositivos. Possui uma baixa taxa de transmissão (até 250 Kbps), um alcance típico de 30 m, e suporta uma elevada densidade de nós por rede. Normalmente está associado a redes de baixo custo e complexidade reduzida. Criado pelo IEEE em 2003, este protocolo especifica as duas primeiras camadas do modelo OSI, Física e Ligação de Dados. Tornou-se conhecido devido às suas aplicações em redes de sensores sem fios, onde podem existir 65534 nós quando utilizados endereços de 16 *bits* (Fernandes, 2011),(IEEE, 2006).

Camada Física

A camada Física, a mais próxima do *hardware*, é especificada pelo protocolo IEEE 802.15.4, que opera numa das três bandas de frequência apresentadas na tabela 2.1. Para aplicações europeias é usada a frequência 868 MHz, para aplicações norte americanas é usada a faixa de frequências 902-928 MHz, e para aplicações globais é usada a frequência 2.4 GHz. Existe um único canal entre a faixa 868-868.6 MHz, 10 na faixa 902-928 MHz e 16 entre 2400-2483.5 MHz (Fernandes, 2011),(IEEE, 2006).

Resumidamente, a camada Física é responsável pela ativação/desativação do transmissor de rádio, deteção de potência dentro do canal, indicação da qualidade de

Tabela 2.1 – Caraterísticas das bandas de frequência no IEEE 802.15.4 (Fernandes, 2011).

Bandas de frequência	Parâmetros de Espalhamento		Parâmetros de Dados		
	Taxa de Espalhamento	Modulação	Taxa de Bit	Taxa de símbolo	Símbolos
868-868.6 (MHz)	300 (Kchip/s)	BPSK	20 (Kb/s)	20	Binário
902-928 (MHz)	600 (Kchip/s)	BPSK	40 (Kb/s)	40	Binário
2400-2483.5 (MHz)	2000 (Kchip/s)	O-QPSK	250 (Kb/s)	62.5	16 Símbolos

ligação dos pacotes recebidos, seleção da frequência do canal e receção/transmissão de dados (Fernandes, 2011).

Sub-camada de controlo de acesso ao meio

O protocolo IEEE 802.15.4 define ainda as especificações da sub-camada de controlo de acesso ao meio (MAC), que fornece uma interface entre a camada Física e a camada de Rede e é responsável por dois modos de operação, *beacon* e *non-beacon*. No modo *non-beacon* a camada MAC utiliza o método CSMA/CA, onde cada nó verifica o canal antes de iniciar a sua transmissão, diminuindo assim a possibilidade de transmissões simultâneas. Caso o canal esteja livre, o nó inicia a transmissão, no caso de uma possível colisão, aborta a transmissão e aguarda um período de tempo aleatório até voltar a tentar. O modo *beacon* tem como objetivo economizar energia por parte dos nós, onde estes enviam um sinal (*beacon*) para alertar que estão ativos. Entre *beacons*, e usando uma boa sincronização, os nós podem permanecer inativos. Delimitada por *frames beacon*, é usada uma outa *frame* no acesso exclusivo ao canal. Esta, contém um período de contenção de acesso onde todos os dispositivos, usando o método CSMA/CA, competem entre si pelo acesso ao meio, e um período livre de contenção, que garante *slots* de tempo para cada dispositivo. Após isso, e até ao próximo *beacon*, o dispositivo fica inativo, diminuindo assim o seu consumo

energético (Fernandes, 2011).

As *frames* do IEEE 802.15.4, definidas nesta sub-camada, obedecem ao formato apresentado na figura 2.4. No processo de comunicação, os dados chegam à sub-camada MAC e são referenciados como MSDU (*MAC Service Data Unit*). Aqui, é adicionado o cabeçalho MHR (*MAC Header*) e o rodapé MFR (*MAC Footer*), formando a *frame* MPDU (*MAC Protocol Data Unit*). Esta é enviada para a camada física como uma *frame* de dados dessa camada (PSDU), onde é adicionado o cabeçalho de sincronização SHR e o PHR, que contém o seu comprimento. Os campos SHR, PHR e PSDU formam uma *frame* de dados da camada Física denominada PPDU (*Physical Protocol Data Unit*) que é convertida em *bits* e enviada pelo meio de transmissão (Fernandes, 2011).

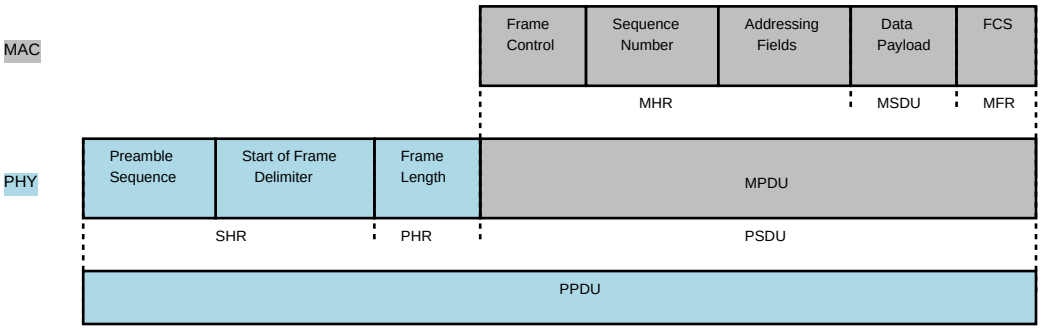

Figura 2.4 – Formato de uma *frame* de dados no IEEE 802.15.4.

2.2.4 IEEE 802.16

Em 2002 foi publicado o protocolo IEEE 802.16, concebido para ligações ponto a ponto, através de ondas rádio com linha de vista, *Line of sight* (LOS), entre o emissor e recetor. Este protocolo surgiu com o principal objetivo de cobrir grandes distâncias, usando para isso frequências dentro da banda de 10 a 66 GHz. No ano seguinte, foi lançado o IEEE 802.16a, conhecido como WiMax, que opera no intervalo de frequências de 2 a 11 GHz. Este protocolo pode ser usado em topologias de ponto-multiponto e redes em malha, não sendo necessária a linha de vista anteriormente especificada. Com uma taxa de transmissão máxima de 75 Mbps e

cobertura de aproximadamente 50 Km, o IEEE 802.16 destina-se a redes sem fios de área metropolitana (WMAN), sendo uma solução para prover serviços de rede em cidades inteiras (Matos et al., 2011).

Camada Física

No protocolo IEEE 802.16 os sinais utilizam a modulação OFDM e são constituídos por 256 portadoras (norma 802.16d) ou 128 a 2048 (norma 802.16e) consoante a largura de banda. Dessas portadoras, apenas 3/4 são usadas para dados, as restantes são utilizadas como bandas de guarda e bandas piloto, necessárias no sincronismo. Neste método, as portadoras são independentes umas das outras, contêm uma parcela da informação e são transportadas em diferentes frequências. O facto de utilizar modulação OFDM, garante uma certa robustez contra transmissão de fluxos independentes feitos em paralelo por antenas diferentes e alguma flexibilidade para incluir algoritmos e técnicas como codificação, modulação adaptativa e multi-acesso (Araújo, 2008).

Tipicamente a comunicação é implementada pela técnica TDD (Time Division Duplex), que usando um único canal para *uplink* e *downlink*, garante mais eficiência no espetro. Para evitar interferências, as comunicações da estação base e dos utilizadores ocorrem em períodos de tempo diferentes. Primeiro, a estação base transmite uma sub-*frame* seguida de um período de silêncio (*gap*). Após esse período, os utilizadores transmitem as suas sub-*frames* seguidas de outro *gap*. Por fim, a estação base volta a transmitir, recomeçado o ciclo (Araújo, 2008).

Sub-camada de controlo de acesso ao meio

A sub-camada MAC é responsável pela troca de pacotes de dados dentro da rede através de um canal partilhado. É uma camada orientada à ligação, pelo que se torna necessário a implementação de um mecanismo de classificação de pacotes, ao encargo da CS (*Convergence Sublayer*) que classifica os pacotes para serem enviados

para a respetiva ligação. Existe ainda a CPS (*Common Part Sublayer*), responsável pelo endereçamento e mecanismos de criação de ligações e a *Privacy Sublayer* que lida com os aspetos de segurança (Araújo, 2008).

No IEEE 802.16, a sub-camada MAC implementa um algoritmo que permite às estações subscritoras competirem uma única vez pelo acesso ao AP. Uma vez conseguido este acesso, é atribuído um *slot* de tempo variável à estação subscritora que o conseguiu. Esta camada, permite também diferenciar a qualidade de serviço consoante a aplicação em questão, garante avanços nos sistemas de segurança, e tem opções para poupança de bateria e mobilidade (Araújo, 2008).

Existem duas possíveis arquiteturas MAC, a arquitetura ponto-multiponto e a arquitetura *Mesh*. Na arquitetura ponto-multiponto toda a comunicação de uma estação subscritora passa pela estação base. É uma arquitetura relativamente barata, pois reduz a complexidade das estações subscritoras. Na topologia *Mesh* (malha), cada estação funciona como um nó repetidor, distribuindo tráfego para as estações vizinhas. Uma estação subscritora pode ligar-se a uma ou mais estações subscritoras intermediárias até atingir a estação base. Esta topologia é particularmente útil em grandes cidades com limitações de linha de vista, uma vez que permite aumentar a quantidade de utilizadores sem acrescentar novas estações base (Araújo, 2008).

2.3 Barramentos Industriais

Tal como no caso dos protocolos de comunicação *wireless*, existem protocolos guiados também eles baseados no modelo de referência OSI. Dentre estes, abordar-se-á alguns barramentos industriais muito utilizados, como o CAN (*Controller Area Network*), o LonWorks e o Profibus.

2.3.1 *Controller Area Network*

Introduzido oficialmente no mercado em 1986, o CAN teve a sua origem na Bosch em 1983, cujo desenvolvimento visava a industria automóvel, conforme as suas especificações publicadas mais tarde em 1991 (GmbH (1991)). Devido ao seu baixo custo e bom desempenho é um protocolo muito utilizado na industria em geral (Borges, 2007). O CAN é um barramento multi-mestre, caraterizado pela existência de vários nós mestre no barramento. O sistema de mensagens utilizado é o *broadcast*, definido pelo envio das mensagens a todos os nós presentes no barramento. Este método garante a consistência dos dados em cada nó do sistema (Corrigan, 2008).

A especificação do CAN garante uma taxa máxima de transmissão de dados de 1 Mbps para um barramento com 40 m, constituído no máximo de 30 nós. O cabo deve ser do tipo par entrançado com o par CANH e CANL, terminado nas extremidades com resistências de 120 Ω (Ohm), formando a impedância caraterística da linha que evita reflexões do sinal (Corrigan, 2008). Esta topologia de rede é mostrada na figura 2.5.

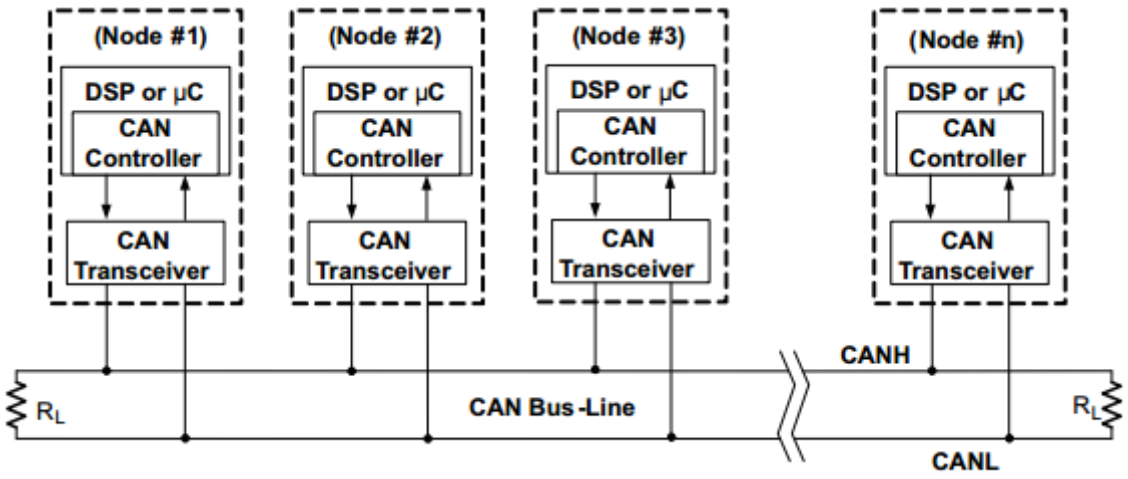

Figura 2.5 – Topologia do barramento CAN (Corrigan, 2008).

Modelo de Comunicação

O CAN define as duas primeiras camadas do modelo OSI, Física e Ligação de Dados. A comunicação entre os nós é caraterizada por ser do tipo série, ou seja, a informação é enviada bit-a-bit no barramento. É usado o protocolo de comunicação CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*), isto significa que cada nó do barramento deve aguardar por um período de inatividade antes de enviar uma mensagem. Este protocolo também resolve as possíveis colisões através da arbitragem *bit-wise*, com base na prioridade estabelecida para cada mensagem no campo identificador (Corrigan, 2008).

Existem duas versões do CAN, a *standard* que possui um identificador de 11 bits, permitindo a existência de 2048 mensagens diferentes e a versão alargada que possui um identificador de 29 bits, permitindo aproximadamente 537 milhões de mensagens diferentes. O formato das mensagens é semelhante em ambas as versões (Corrigan, 2008). Na figura 2.6 é apresentado o formato de uma *frame* na versão *standard*.

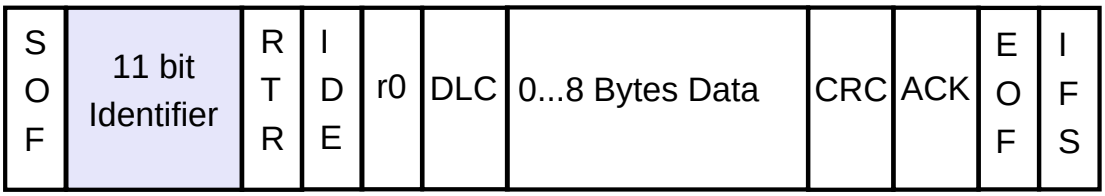

Figura 2.6 – Formato de uma *frame* CAN *STANDARD*.

O campo SOF (*Start of Frame*) indica o início de uma mensagem, e é utilizado para sincronizar os nós no barramento após ficar inativo. O campo Identificador estabelece a prioridade da mensagem, quanto menor o seu valor, maior é a prioridade. O campo RTR (*remote transmission request*) indica se a mensagem diz respeito a um pedido ou a uma resposta. O campo IDE (*Identifier Extension*) indica que está a ser transmitida uma *frame* CAN sem extensão. O campo r0 é um *bit* reservado a uma possível alteração *standard*. O campo DLC (*Data Length Code*) indica o tamanho dos dados transmitidos. O campo Data possui um comprimento máximo de 8 *bytes* e contém os dados a serem transmitidos. O campo CRC (*Cyclic Redundancy Check*) contém o número de *bits* transmitidos nos dados de aplicações anteriores para

deteção de erros. No campo ACK, cada nó substitui o *bit* recessivo original por um dominante caso receba a mensagem livre de erros, garantindo assim a integridade dos dados. O campo EOF (*End of Frame*) marca o final da mensagem CAN. E por último, o campo IFS (*Interframe Space*) contém o tempo necessário para o controlador mover a *frame* recebida para a posição correta do buffer de mensagens (Corrigan, 2008).

A alocação de prioridades para as mensagens é uma caraterística que torna o CAN particularmente interessante no controlo em tempo real. Quanto menor for o valor do identificador da mensagem, maior será a sua prioridade. No caso de ser inteiramente constituído por zeros, a mensagem passa a ser a mais prioritária na rede, detendo o barramento dominante por mais tempo. No caso de dois nós transmitirem em simultâneo, o nó A envia o último *bit* identificador '0' (dominante) enquanto o nó B envia o *bit* '1' (recessivo), então o nó A detém o controlo do barramento e completa o envio da sua mensagem. As mensagens são enviados para todos os nós da rede, onde são filtradas para descobrir quais lhe são dirigidas (Corrigan, 2008).

2.3.2 LonWorks

Criado em 1988 pela corporação Echelon, o LonWorks provê um conjunto de serviços de comunicação que permitem que a aplicação de um dispositivo envie e receba mensagens de outro dispositivo sem que tenha conhecimento da sua topologia da rede, nomes, endereços ou funções. É um protocolo de comunicação ponto-a-ponto que tem por base o modelo OSI. Fornece uma solução específica para aplicações de controlo e é caraterizado pela sua confiabilidade, desempenho e robustez (Echelon, 1999).

Modelo de Comunicação

O protocolo LonWorks define as sete camadas do modelo OSI (Echelon, 1999):

- A camada Física define a transmissão de *bits* pelo canal de comunicação;

- A camada de Ligação de Dados define métodos de acesso ao meio e codificação de dados para garantir a utilização eficiente de um único canal de comunicações. Os *bits* da camada Física são divididos em *frames* de dados variáveis. Esta camada define quando um dispositivo de origem pode transmitir uma *frame* e como o dispositivo de destino recebe essa *frame*. Está encarregue também de detetar erros de transmissão e implementar um mecanismo de prioridades para garantir entrega de mensagens importantes;

- A camada de Rede define como os pacotes são encaminhados desde a fonte até ao destino. São definidos nomes e endereços para os dispositivos de modo a garantir a entrega correta dos pacotes. No caso dos dispositivos pertencerem a diferentes canais de comunicação, são definidos nesta camada processos de roteamento para as mensagens;

- A camada de Transporte garante a entrega correta das mensagens. As mensagens podem ser trocadas usando um serviço de reconhecimento (*acknowledge*), em que o dispositivo de origem reenvia a mensagem caso não receba a confirmação por parte do dispositivo de destino;

- A camada de Sessão controla os dados transmitidos pelas camadas inferiores. Suporta ações remotas e define um protocolo de autenticação que indica aos recetores de uma mensagem se o remetente está realmente autorizado ao envio da mesma;

- A camada de Apresentação define a codificação dos dados, acrescentando assim estrutura ao corpo da mensagem;

- A camada de Aplicação adiciona compatibilidade entre os dados da mensagem e as aplicações, garantindo que estas usem uma interpretação semântica comum em relação aos dados trocados. É também responsável por um protocolo de transferência de ficheiros usado para transferir fluxos de dados entre aplicações.

Sendo independente ao meio, o LonWorks permite que os dispositivos comuniquem em qualquer meio de transporte físico, como o par entrançado, linhas de energia, topologias de barramento, etc. A maior taxa de transmissão de dados possível é de 1,25 Mbps para uma distância máxima de 125m com, no máximo, 64 dispositivos. É usado o algoritmo CSMA, de modo a controlar o acesso ao meio, evitando colisões entre pacotes. Neste algoritmo cada dispositivo aguarda um período de tempo de inatividade até enviar o seu pacote, que pode ser enviado a um dispositivo, a um grupo de dispositivos ou a todos os dispositivos disponíveis. Cada pacote contém o endereço do dispositivo de origem, e o endereço do dispositivo de destino que poderá ser um endereço físico (endereço de fábrica), endereço de dispositivo (endereço do dispositivo na rede), endereço de grupo (endereço caraterístico de um grupo na rede) ou endereço *broadcast* (endereço de todos os dispositivos na rede). Os dispositivos presentes no canal, analisam cada pacote transmitido de modo a averiguar se são eles os possíveis destinatários da mensagem (Echelon, 1999).

2.3.3 Profibus

O Profibus (*Process Field Bus*) teve origem na Alemanha em 1987 e é hoje um dos protocolos mais difundidos na Europa e América. As suas capacidades em termos de velocidade, distância e gestão de dados são muito úteis em controlo de processos e outras aplicações industriais (Borges, 2007).

Modelo de Comunicação

No Profibus todos os nós ativos (mestres), formam uma topologia lógica em anel, onde cada nó conhece os restantes, bem como as suas posições fixas no anel. É passado um *token*, *frame* que circula entre os nós ativos do anel segundo a sua ordem lógica, que permite aos nós enviar *frames* após a sua receção. O *token* especifica o tempo disponibilizado ao nó para o envio de dados, caso este tempo expire, o nó só poderá enviar mensagens de alta prioridade. Se o nó com permissão para transmitir dados, não possuir qualquer mensagem para enviar, deverá passar o *token* para o

próximo nó no anel. Os nós passivos (escravos) nunca têm acesso ao *token*. Quando um nó mestre tem o *token* e tem uma conexão a um nó escravo, este é solicitado para transmitir os seus dados (e.g. valores de leitura) para o nó mestre. Descrita na figura 2.7, a técnica *token ring* numa topologia mestre/escravo (*master*/*slave*), permite tanto a entrada como a saída de nós durante o processo (Siemens, 2009).

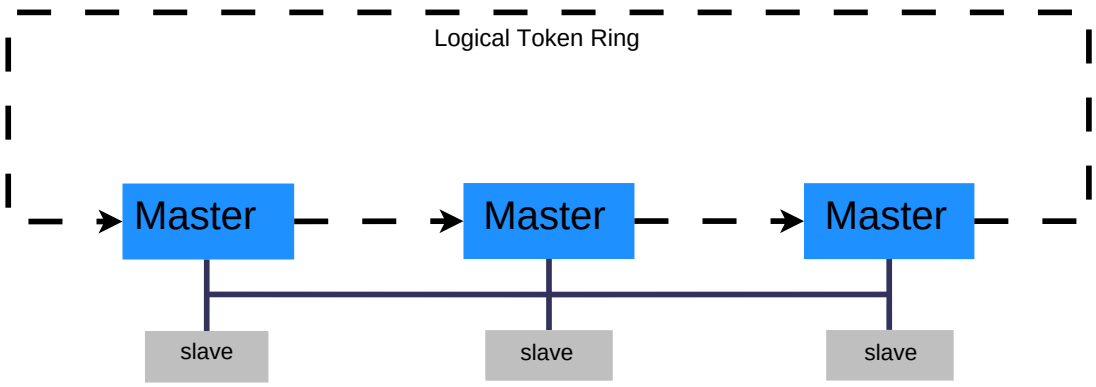

Figura 2.7 – Técnica *token ring* no Profibus.

O protocolo Profibus implementa a camada Física do modelo OSI, que diz respeito à transmissão de *bits* pelo meio físico (Siemens, 2009). Este meio pode ser um cabo de par entrançado, com um máximo de 6 km e uma taxa de transmissão máxima de 12 Mbps para um comprimento de 40 m, ou um cabo de fibra ótica, com uma distância máxima de 24 km e uma taxa de transmissão máxima de 12 Mbps (Borges, 2007). A segunda camada do modelo OSI é conhecida no Profibus como FDL (*Fieldbus Data Link*), e garante a confiabilidade na transmissão de dados. Acima desta camada, existe uma interface específica que pode, de um modo geral, representar a camada de Transporte do modelo OSI. As restantes camadas não são implementadas neste protocolo (Siemens, 2009).

2.4 Análise Comparativa

Existe uma tendência de evolução acentuada nas tecnologias de redes de comunicação, onde parâmetros como o custo, a implementação e a eficiência energética desempenham um papel importantíssimo (Borges, 2007). Foram abordados até aqui,

alguns protocolos de comunicação usados hoje em dia por diversas tecnologias nas mais variadas aplicações. Enumeramos algumas das mais conhecidas tecnologias *wireless* que nos remetem para a flexibilidade e mobilidade das redes, e algumas das mais utilizadas tecnologias guiadas que garantem comunicações seguras e fiáveis em diversos setores, nomeadamente no industrial.

2.4.1 Tecnologias de Redes sem fios

Os protocolos de redes sem fios abordados, garantem a troca de informação entre diversos dispositivos numa rede, definindo normas para as camadas Física e Ligação de Dados do modelo OSI (Fernandes, 2011). A tabela 2.2 enumera as principais caraterísticas de cada um deles.

Tabela 2.2 – Caraterísticas dos protocolos de comunicação *wireless*.

Protocolo	Tipo de Rede	Cobertura (Típica)	Taxa de Transmissão (Típica)
IEEE 802.11	WLAN	100 m	54 Mbps
IEEE 802.15.1	WPAN	10 m	1 Mbps
IEEE 802.15.4	WPAN	30 m	250 Kbps
IEEE 802.16	WMAN	50 Km	75 Mbps

Alguns problemas no desempenho do sinal junto do elevado consumo energético têm ditado a pouca aceitação do IEEE 802.16. Já o IEEE 802.11 é um fenómeno no que às tecnologias de redes diz respeito, muito embora apresente custos de implementação elevados face aos protocolos IEEE 802.15.1 e IEEE 802.15.4, que apresentam baixo consumo energético derivado da baixa área de cobertura abrangida e das baixas taxas de transmissão. O elevado tempo de associação entre dispositivos, que pode prejudicar o bom funcionamento de algumas aplicações, fez com que o IEEE 802.15.1 fosse preterido. Por outro lado, o baixo consumo energético, aliado ao baixo custo

e facilidade de implementação, fez do IEEE 802.15.4 a tecnologia *wireless* escolhida na realização deste projeto.

2.4.2 Tecnologias de Redes Guiadas

Os protocolos de redes com fios abordados, são protocolos industriais abertos muito utilizados no controlo de processos e noutras aplicações industrias. Estes sistemas de barramento de campo, permitem, interligação com vários dispositivos, maiores velocidades nos tempos de comando e resposta e maior fiabilidade e disponibilidade do sistema. O CAN, o LonWorks e o Profibus possuem provas dadas de bom desempenho em diversas aplicações, e são cada vez mais os equipamentos que comunicam ou têm possibilidade de comunicar através destes protocolos (Borges, 2007). A tabela 2.3 apresenta a taxa de transmissão e número de nós máximos para o respetivo comprimento do barramento em cada protocolo.

Tabela 2.3 – Caraterísticas dos protocolos de comunicação guiados.

Protocolo	Taxa Transmissão (Máx)/Comprimento	Nº nós (Máx)/Comprimento
CAN	1 Mbps/40 m	30/40 m
LonWorks	1,25 Mbps/125 m	64/125 m
Profibus (Par entrançado)	12 Mbps/40 m	32/1.900 m

Não menosprezando os restantes protocolos, o baixo custo e facilidade na implementação aliados a alguma familiaridade, fizeram do CAN a escolha da tecnologia guiada na realização deste projeto.

3 Arquitetura do Sistema

Apesar de se pretender criar um sistema de comunicações o mais versátil possível, com vista à integração em diversas situações de monitorização e controlo de tráfego citadino, o nosso caso de estudo diz respeito à monitorização de estacionamentos. Digamos que esta será a nossa árvore numa imensa floresta de técnicas que visam a melhoria do tráfego urbano. É sem dúvida uma das grandes problemáticas das cidades e surge aqui como ponto de partida para a criação de um sistema que agilize, através da monitorização, o processo do estacionamento.

Recorreu-se às tecnologias de redes de comunicação abordadas anteriormente, nomeadamente o CAN e o IEEE 802.15.4, tirando assim proveito dos seus baixos custos e simples implementações. As suas topologias permitem a flexibilidade necessária à implementação de um sistema deste tipo e as velocidades de comunicação cumprem os requisitos pretendidos. O CAN pode ser instalado num ambiente cujos encargos com a remoção do pavimento não sejam exagerados (e.g. paralelo), o IEEE 802.15.4 torna-se útil nas situações onde esses encargos não compensariam o investimento (e.g. asfalto). As questões de alimentação dos nós não são problemáticas, quer no CAN, por ser um protocolo guiado, quer no IEEE 802.15.4, cujo baixo consumo energético prolonga a vida útil das baterias utilizadas para o efeito.

3.1 Conceção do Sistema

O sistema de monitorização de estacionamentos tem como objetivo informar os condutores acerca do estado do estacionamento, livre ou ocupado. Como exemplificado na figura 3.1, esta informação é apresentada através de sinalização luminosa percetível ao condutor a alguns metros de distância, o que facilita a procura de lugar no momento deste estacionar.

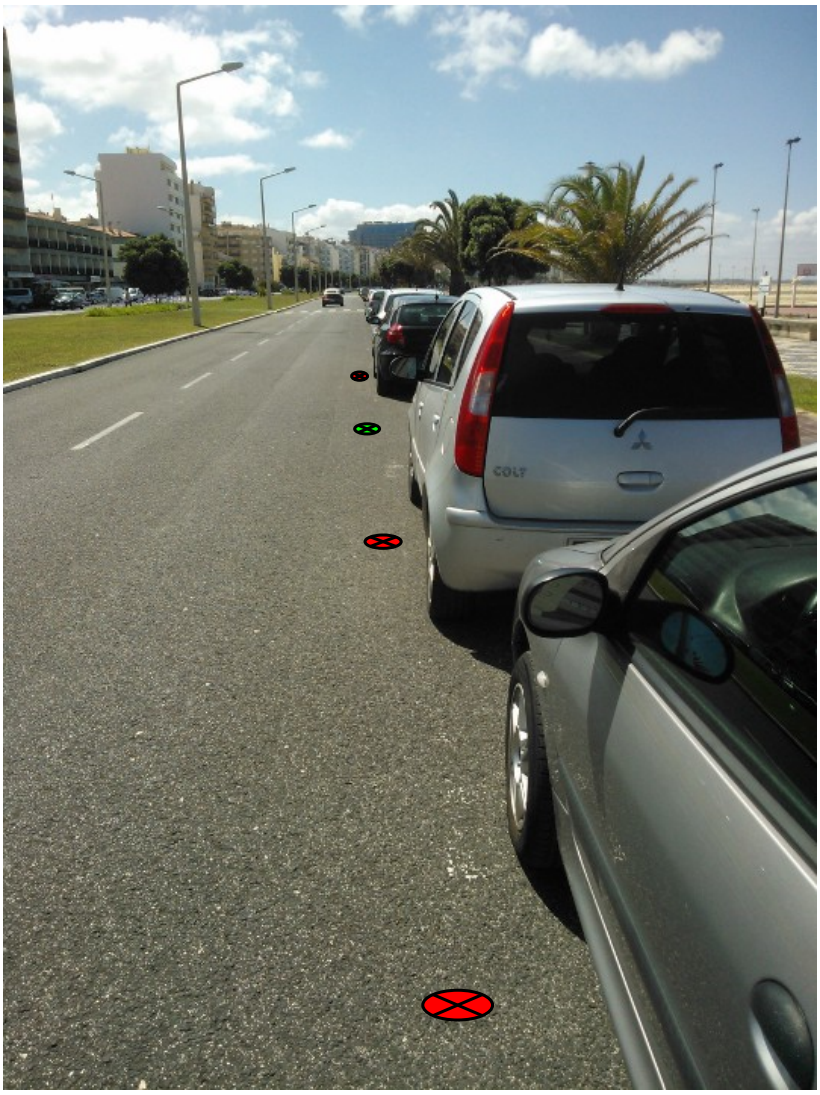

Figura 3.1 – Visão ilustrativa do funcionamento do sistema.

3.1.1 Principio de funcionamento

Conforme descrito na figura 3.2, o sistema é baseado em nós que são colocados em cada estacionamento. Estes nós são capazes de detetar e sinalizar a presença de veículos recorrendo a sensores para a deteção e a iluminação LED (*Light Emitting Diode*) para a sinalização. Feita a leitura do sensor, é gerada uma mensagem que indica o preciso estado do estacionamento. No caso de ser um nó CAN a mensagem percorre o barramento até ao nó de interface CAN, similar a um nó CAN, mas capaz de enviar a mensagem para a aplicação Driver CAN. Caso se trate de um nó IEEE 802.15.4, a mensagem é enviada via *wireless* para o nó de interface IEEE 802.15.4, cujo objetivo é de igual forma fazê-la chegar a aplicação Driver IEEE 802.15.4. Em ambos os casos, os Drivers fazem a comunicação entre o sistema operativo da máquina e o *hardware*.

Para a troca de mensagens entre aplicações, e também por ser utilizado em outras facetas do projeto *Smart City*, recorreu-se ao *middleware* ActiveMQ, serviço de troca de informação entre processos distribuídos. Os Drivers comunicam a mensagem via TCP/IP à aplicação CCO (Centro de Controlo e Operação), que realiza as seguintes tarefas:

1. Analisa o conteúdo da mensagem, verificando o estado do estacionamento, e cria uma mensagem de resposta com a sinalização correspondente ao estado. Se o estado for ocupado a sinalização será de cor vermelha, se o estado for livre a sinalização será de cor verde. Esta mensagem é consumida pelo Driver em questão e enviada ao nó correspondente via nó de interface. Após verificar o conteúdo da mensagem, o nó sinaliza o estacionamento;

2. Transmite o conteúdo da mensagem à aplicação DataBase, que através do servidor MySQL atualiza o estado do estacionamento numa base de dados previamente criada. Posteriormente é invocada uma JSP (*JavaServer Pages*) que informa os estacionamentos livres para cada local (e.g. Rua, Avenida, Parque) presente na base de dados. Esta informação pode ser apresentada em

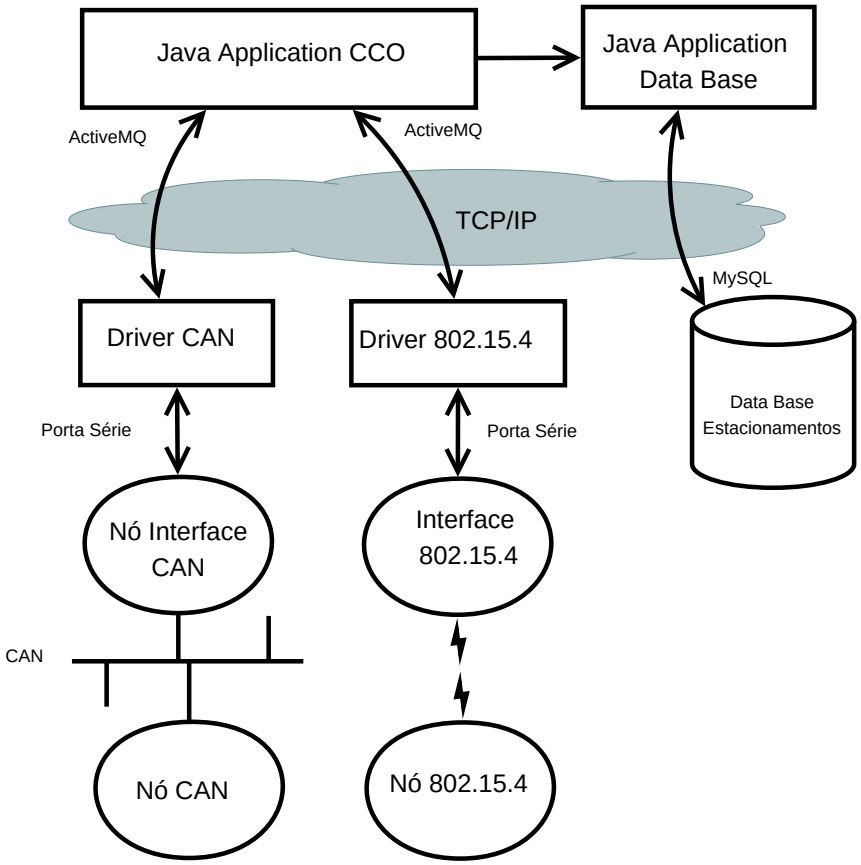

Figura 3.2 – Modelo de funcionamento do sistema.

painéis informativos junto dos respetivos locais, que informarão os condutores acerca dos lugares disponíveis.

3.2 Implementação das comunicações CAN

O processo de implementação do sistema, usando o protocolo de comunicação CAN, engloba todo o sistema de comunicações que interliga desde os nós do estacionamento às aplicações CCO e DataBase. Esta descrição terá agora inicio com uma abordagem ao protocolo CAN.

3.2.1 Protocolo CAN

A troca de mensagens entre os nós do barramento é possível graças à implementação do protocolo CAN. O formato destas mensagens é descrito na figura 3.3.

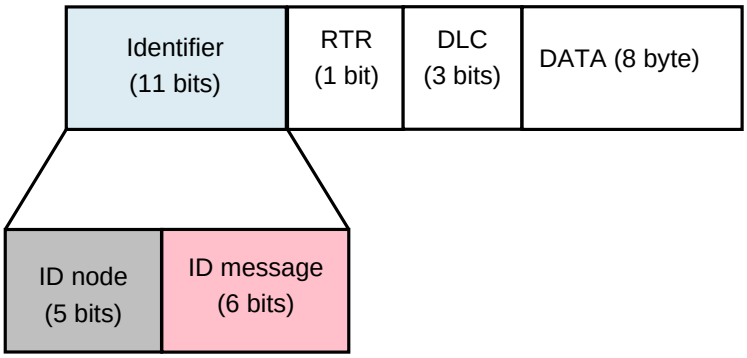

Figura 3.3 – Formato das mensagens CAN no barramento.

O identificador está dividido em dois sub-campos, o ID do nó, que identifica os nós CAN dentro do sistema, e o ID da mensagem que identifica as mensagens CAN utilizadas. Segue-se o campo RTR que indica se a mensagem se trata de um pedido ou de uma resposta, utilizando para isso os valores '1' e '0' respectivamente. O campo DLC indica o tamanho dos dados transportados no campo Data, o último da composição da *frame*.

Para a troca de mensagens entre o nó de interface CAN e o Driver CAN foi criado um protocolo de comunicação orientado ao *bit* onde se encontra o campo SOF, marcando o inicio da mensagem e permitindo identificar se esta faz parte do protocolo, acoplado aos restantes parâmetros da mensagem CAN. Este formato é descrito na figura 3.4.

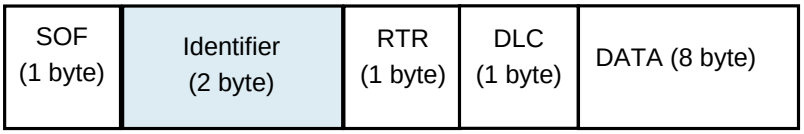

Figura 3.4 – Formato das mensagens CAN no *software*.

Tipos de Mensagens CAN

Foram projetados quatro tipos de mensagens consideradas fundamentais ao correto funcionamento do sistema. A tabela 3.1 apresenta essas mensagens bem como o seu percurso e formato no protocolo.

Tabela 3.1 – Tipos de mensagens CAN.

Mensagem	Origem-Destino	Formato da Mensagem CAN				
		ID NÓ	ID Mensagem	RTR	DLC	DATA
SendMAC	Nó CAN-Driver CAN	Não definido	0	0	6	Endereço MAC do nó
SendID	Driver CAN-Nó CAN	ID do nó	1	0	6	Endereço MAC do nó
SendState	Nó CAN-CCO	ID do nó	4	0	1	0/1 (livre/ocupado)
SendLeds	CCO-Nó CAN	ID do nó	5	0	1	0/1 (verde/vermelho)

Ao ligarmos um nó ao barramento CAN, este necessita ser identificado pelo sistema. Cada nó possui um identificador de fábrica, endereço MAC, que apesar de o identificar fisicamente, não o identifica como nó do sistema. Assim, cada vez que este se liga, é criada a mensagem `SendMAC`, que envia o endereço MAC do nó para que possa receber uma identificação na rede. A mensagem é transmitida no barramento pelo nó CAN, e chegada ao nó de interface é encaminhada para o Driver.

Após a sua receção, o Driver cria uma mensagem de resposta fornecendo ao nó o desejado ID. Esta mensagem, `SendID`, é enviada pelo Driver para o barramento CAN, através do nó de interface, e contém o ID atribuído ao nó e o respetivo endereço MAC no campo de dados, permitindo ao nó confirmar se é o devido destinatário da mensagem. Após a sua receção, o nó guarda o seu ID e passa a estar identificado na rede de comunicação.

A partir deste momento, o nó pode enviar o estado do estacionamento (livre/ocupado),

através da mensagem `SendState`, que chegada ao Driver é encaminhada para o CCO. Este, verifica o seu conteúdo e responde com a sinalização adequada através da mensagem `SendLeds`, que chegada ao Driver é transmitida para o barramento CAN. O nó, lê a mensagem, verifica que é o devido destinatário, após confirmação do ID, e ativa a iluminação correspondente (verde/vermelho).

3.2.2 Implementação do *Hardware*

Os recursos de *hardware* utilizados na implementação deste sistema, dizem respeito aos dois tipos de nós dimensionados, nós CAN e nós de interface CAN.

Utilizados como terminal de comunicação, os nós CAN são colocados nos lugares de estacionamento com a finalidade de monitorizar o seu estado. Foram implementados como um módulo completo constituído pelos elementos descritos na figura 3.5, onde identificamos os seguintes blocos:

- Microcontrolador PIC18F2680 (Microchip, 2007), escolhido por possuir controlador CAN. É programado para lidar com os periféricos de entrada/saída, como o interruptor e os LEDs de sinalização;
- Interruptor do tipo *on/off*, utilizado na simulação do estado do estacionamento. Gera valores de tensão de 0V, estado livre, e 5V, estado ocupado;
- LEDs vermelho e verde que nos indicam o estado do lugar, ocupado e livre respetivamente;
- Interface PCA82C250 (Philips, 2000), utilizada entre o barramento CAN e o microcontrolador.

Os nós de interface CAN têm como principal objetivo fazer a ligação entre os restantes nós CAN presentes no barramento e o Driver CAN. Essa ligação, consiste na permuta de mensagens entre os dois sistemas e é assegurada pela porta série. Este

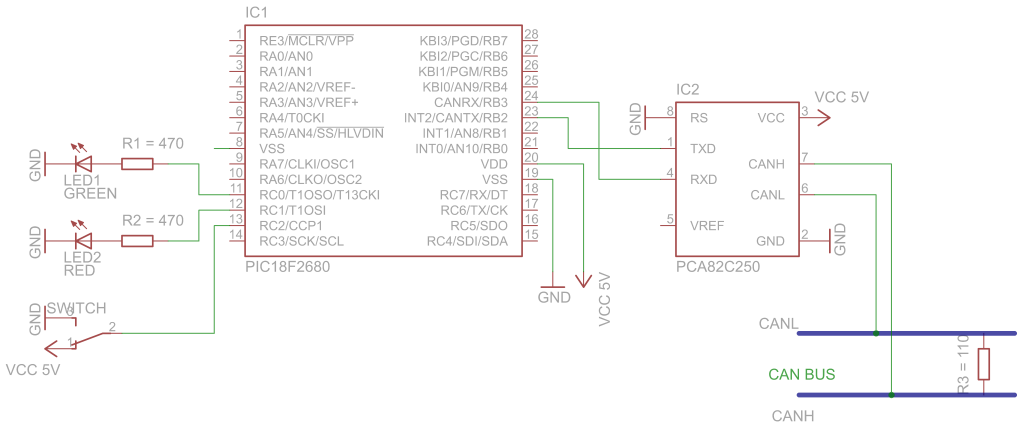

Figura 3.5 – Esquemático do nó CAN.

tipo de nó foi de igual forma implementado como um módulo completo constituído pelos elementos descritos na figura 3.6, onde identificamos os seguintes blocos:

- Microcontrolador PIC18F2680 (Microchip, 2007), programado para lidar com a troca de mensagens entre os nós do barramento e o Driver CAN;
- Interface PCA82C250 (Philips, 2000), utilizada entre o barramento CAN e o microcontrolador;
- Interface série RS232 constituída pelo MAX232 (Instruments, 2004) e a ficha conectora DB9, permitindo a conversão dos níveis dos sinais vindos do microcontrolador em níveis adequados à porta série e vice versa.

3.2.3 Implementação do *Firmware* e *Software*

Definido o protocolo e as implementações do *hardware*, é altura de abordar a implementação do *firmware* e *software* utilizados no desenvolvimento do sistema. No que diz respeito aos dois tipos de nós, o *firmware* foi programado na linguagem C utilizando a ferramenta MPLAB. O *software* das aplicações das camadas superiores foi implementado na linguagem Java com recurso ao NetBeans. Tentar-se-á aproximar

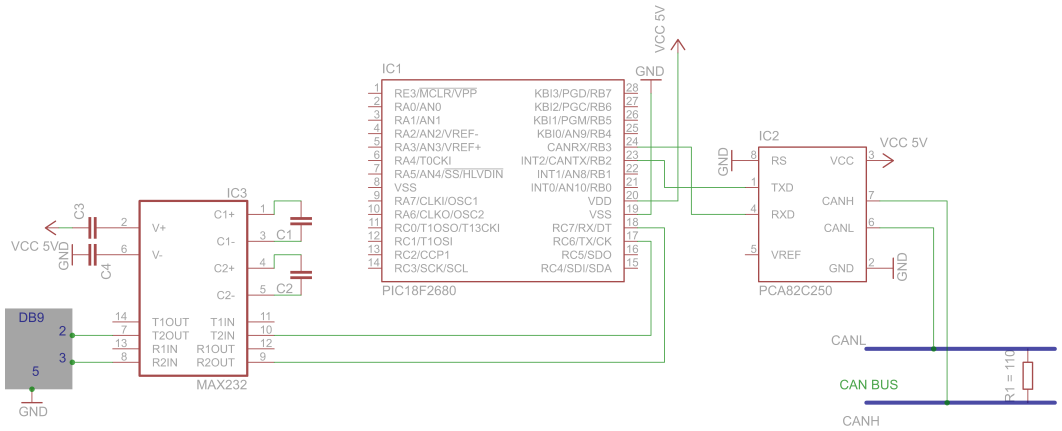

Figura 3.6 – Esquemático do nó de Interface CAN.

esta descrição o mais possível ao processo de comunicação, obedecendo assim à sua ordem de funcionamento.

Pedido e Atribuição de ID

O processo de comunicação na rede parte do pressuposto que os nós CAN estão devidamente identificados, portanto essa é a primeira tarefa realizada no momento da inserção de um novo nó. Com recurso ao MPLAB, é guardado um endereço MAC de 6 *bytes* na memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory*) do microcontrolador que servirá como identificador físico do nó.

Após cada nó possuir o seu endereço físico, poderá comunicar e fazer o pedido de um ID que o identifique na rede. Isso é realizado pela função SendMAC que é executada sempre que o nó se liga ao barramento. Esta função lê o endereço MAC guardado na memória EEPROM e cria a mensagem `SendMAC`, cujo ID de mensagem é '0' e no campo de dados contém o endereço MAC do nó. A mensagem é colocada num *buffer* de envio que vai guardando as mensagens a serem enviadas. Paralelamente existe a execução da função CANTask, que tem como objetivo enviar as mensagens presentes no *buffer* para o barramento CAN, recorrendo à função SendMessage.

Após ser transmitida no barramento, a mensagem será recebida pelo nó de interface

CAN, que executa também ele a função CANTask. Esta, verifica recorrentemente se recebeu alguma mensagem, através da função ReceivedMessage. Caso tenha recebido, a mensagem é colocada num *buffer* de receção e lida, nomeadamente o seu ID. Caso este ID seja '0', trata-se da mensagem `SendMAC`, o que significa um pedido de ID por parte de um nó CAN e necessita ser enviada para o Driver CAN. Portanto, é executada a função SendToPCTask, que transmite a mensagem pela porta série, chegando assim ao Driver, aplicação em Java que é executada num computador. Este processo é ilustrado na figura 3.7.

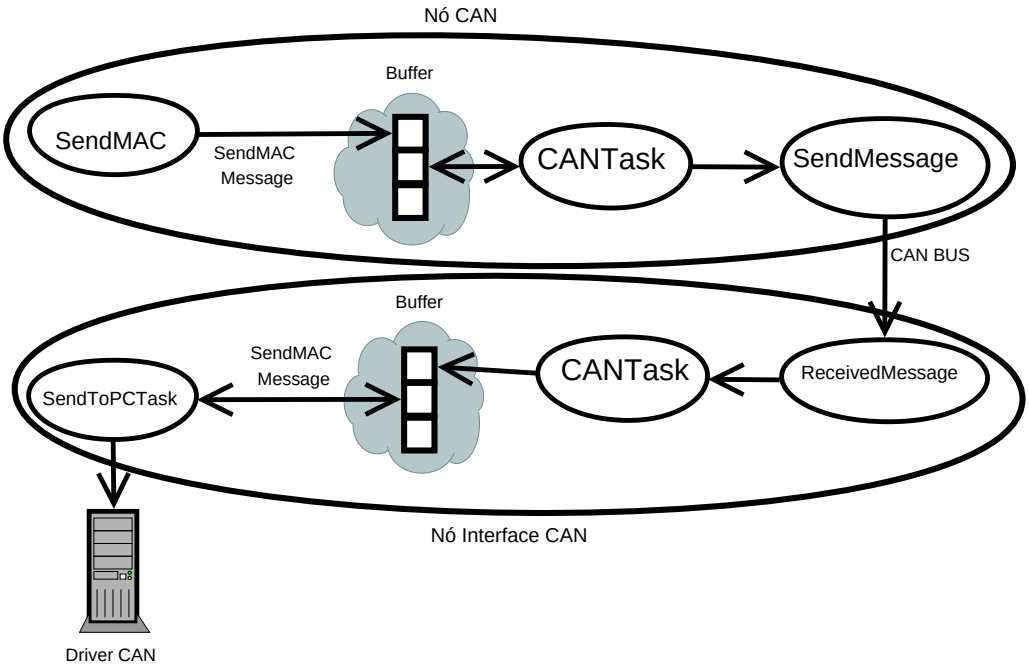

Figura 3.7 – Processo de pedido de ID nos nós.

A aplicação Driver CAN inicia a sua execução por abrir uma ligação à porta série, de modo que possa receber as mensagens vindas desse meio. Após efetuada a ligação, verifica recorrentemente a receção de algum evento vindo da porta série. Caso receba alguma mensagem, analisa o primeiro campo, SOF, de modo a verificar se esta se trata de uma mensagem do protocolo. Após esta verificação, a mensagem é guardada e verificado o seu ID, indicativo do tipo de mensagem. Neste caso, por tratar-se da mensagem `SendMAC`, o ID será '0'. É lido o campo de dados da mensagem, onde

consta o MAC do nó, e este é guardado numa lista de ID. A posição da lista onde este é guardado é atribuída como sendo o ID do nó na rede. Em futuros pedidos de ID por parte daquele nó, será apenas necessário verificar a posição na lista que corresponde ao seu MAC, uma vez que este já consta da lista. É então criada a mensagem `SendID`, com o ID de mensagem '1', o ID do nó na rede atribuído e o campo de dados continuará a ter o MAC do nó. Esta mensagem é transmitida pela porta série até ao nó de interface CAN. Este processo é ilustrado na figura 3.8.

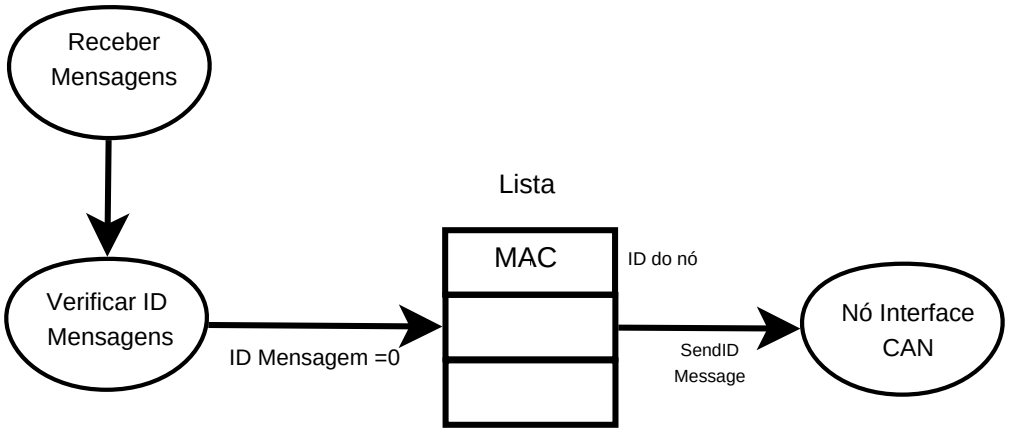

Figura 3.8 – Pedido e atribuição de ID no Driver.

O nó de interface CAN executa recorrentemente a função vSerial que verifica a receção de mensagens vindas da porta série, ou seja, do Driver. Caso receba uma mensagem CAN, esta é colocada no *buffer* de envio e transmitida para o barramento através da função CANTask. É recebida pelos nós CAN existentes, colocada num *buffer* de receção e lido o seu ID de mensagem. Como se trata da mensagem `SendID`, é recebido um ID para um determinado nó CAN que o passará a utilizar na rede. De modo a que apenas o nó que efetuou o pedido guarde o ID, é feita uma verificação através do MAC presente no campo de dados da mensagem, que corresponde ao endereço MAC do dito nó. Os nós irão comparar o seu MAC com o do campo de dados, o nó cujo MAC for igual é o correto destinatário e guarda o seu ID de rede, atribuído pelo Diver. Este processo é ilustrado na figura 3.9.

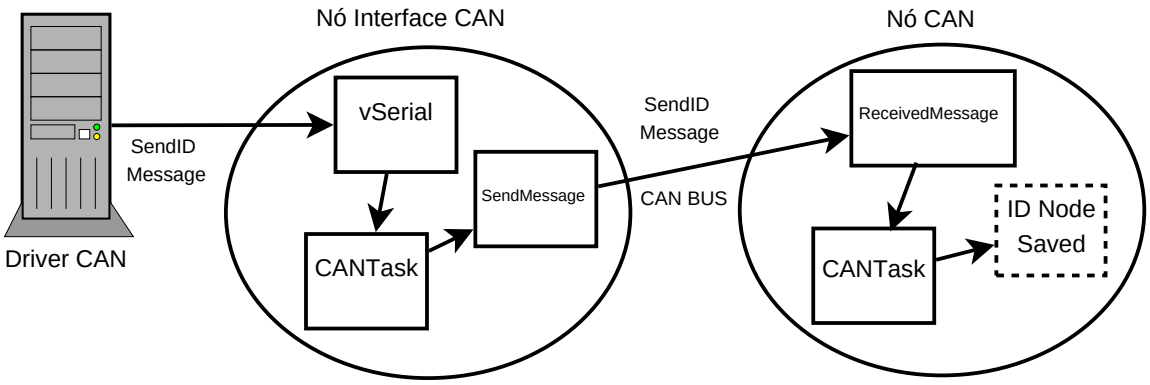

Figura 3.9 – Atribuição de ID ao nó.

Monitorização e Sinalização

Após o processo de pedido e atribuição de ID mencionado, os nós CAN podem comunicar o estado do estacionamento com o sistema. Este estado foi simulado por um interruptor do tipo *on/off* com duas posições, '0' e '1', que indicam o estado livre e ocupado respetivamente. Esta simulação é feita recorrendo à função SwitchState, máquina de estados que é executada recorrentemente nos nós CAN. Ilustrados na figura 3.10, esta função possui o seguintes estados:

1. Neste estado é verificado o valor de entrada que chega do interruptor. Caso seja '0', estacionamento livre, é criada a mensagem `SendState`, com o ID do nó em questão, o ID de mensagem '4' e nos dados o valor '0' que indica o estado livre. A mensagem é colocada no *buffer* de envio e passamos ao estado 2. Caso o valor de entrada seja '1', estacionamento ocupado, é criada a mensagem `SendState`, com o ID do nó em questão, o ID de mensagem '4' e nos dados o valor '1' que indica o estado ocupado. A mensagem é colocada no *buffer* de envio e passamos ao estado 3;

2. Aqui é verificado se o estado do interruptor mudou para a posição '1', indicando que o estacionamento passou a estar ocupado. Caso isto aconteça, retomamos ao estado 1;

3. Aqui é verificado se o estado do interruptor mudou para a posição '0', indicando

que o estacionamento passou a estar livre. Caso isto aconteça, retomamos ao estado 1.

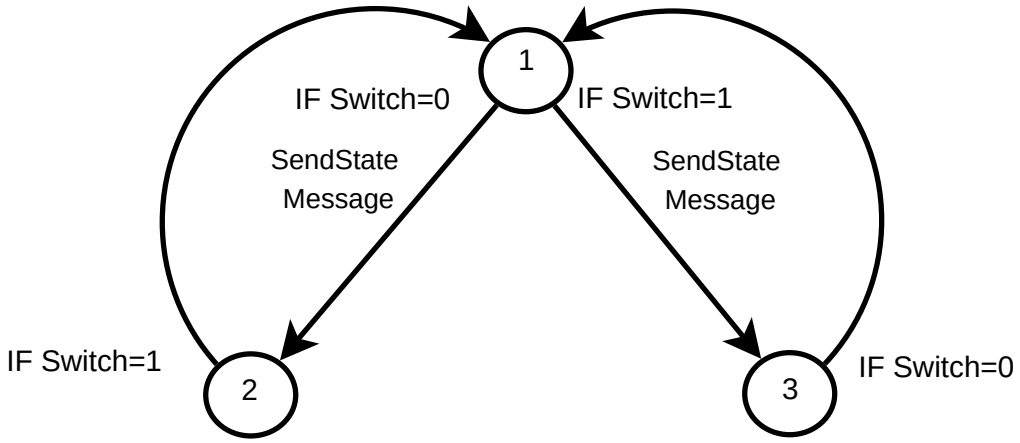

Figura 3.10 – Máquina de estados da função SwitchState.

Tal como no processo do pedido e atribuição de ID, a mensagem `SendState` é enviada pelo barramento até ao nó de interface CAN. Após a sua receção, este lê o valor do ID de mensagem, de modo a verificar de que tipo de mensagem se trata. Concluindo tratar-se da mensagem do tipo `SendState` é realizada a sua transmissão via porta série com destino o Driver CAN.

Este, verifica mais uma vez o ID da mensagem, que corresponde à mensagem `SendState`, contendo informação do estado do estacionamento num determinado nó. Neste caso, a informação deve ser comunicada à aplicação CCO para que possa ser analisada. Esta comunicação é realizada com recurso ao serviço ActiveMQ, tornando o Driver e o CCO em produtores e consumidores de mensagens. O Driver efetua assim uma conexão com este serviço, e produz a mensagem para uma lista denominada EstadoEstacionamento. A mensagem produzida é a `SendState`, onde se acrescenta o ID do Driver, para posterior identificação do Driver correto, e o endereço MAC do nó remetente da mensagem, retirado da lista de ID, para posterior utilização na base de dados.

Em constante execução, o CCO consome a mensagem da lista EstadoEstacionamento. Todos os parâmetros da mensagem são guardados, e os dados são analisados.

Caso este campo seja igual a '0', estado livre, é criada a mensagem `SendLeds`, com o ID de mensagem '5' e os dados '0', que simbolizam a sinalização verde a enviar ao nó. Caso os dados sejam '1', estado ocupado, é criada a mensagem `SendLeds`, com o ID de mensagem '5' e os dados '1', que simbolizam a sinalização vermelha a enviar ao nó. Funcionando como produtor, o CCO envia a mensagem `SendLeds` para uma lista denominada Sinalizacao.

O Driver, que na transação anterior foi produtor, será agora um consumidor da lista Sinalizacao. Guarda os parâmetros da mensagem `SendLeds` consumida e verifica se o seu ID, ID de Driver, é igual ao contido na mensagem, confirmando se é o Driver que produziu a anterior mensagem `SendState`. Em caso afirmativo, a mensagem `SendLeds` é transmitida para o nó de interface CAN, via porta série.

Tal como no processo do pedido e atribuição de ID, a mensagem `SendLeds` é recebida no nó de interface CAN e transmitida para os nós no barramento. Ao recebê-la, estes leêm o seu ID de mensagem, que neste caso será '5', o que significa que contém informação de sinalização para o nó que informou o seu estado do estacionamento. Os nós comparam o seu ID de nó, com o ID de nó incluido na mensagem, de modo a verificar quem é o real destinatário da mesma. O nó cujo ID seja igual, ativa a sinalização LED verde, caso os dados sejam '0', ou a sinalização LED vermelha, caso os dados sejam '1'. Este processo é ilustrado na figura 3.11.

Base de Dados

A informação do estado dos estacionamentos é guardada numa base de dados, para que possa ser utilizada em análises posteriores. Numa tabela de locais, com os campos ID do Local e Nome do Local, são guardados locais a monitorizar (e.g. Ruas, Avenidas, Parques). Na tabela CAN, com os campos ID do Local, ID do Estacionamento e Estado do Estacionamento, é guardado o estado de cada estacionamento presente num determinado local a monitorizar pela tecnologia CAN. Um local poderá ser, a título de exemplo, a Avenida Carvalho Araújo, que tem o ID de Local '1', possui 10 lugares de estacionamento, ou seja 10 ID de estacionamento, e para

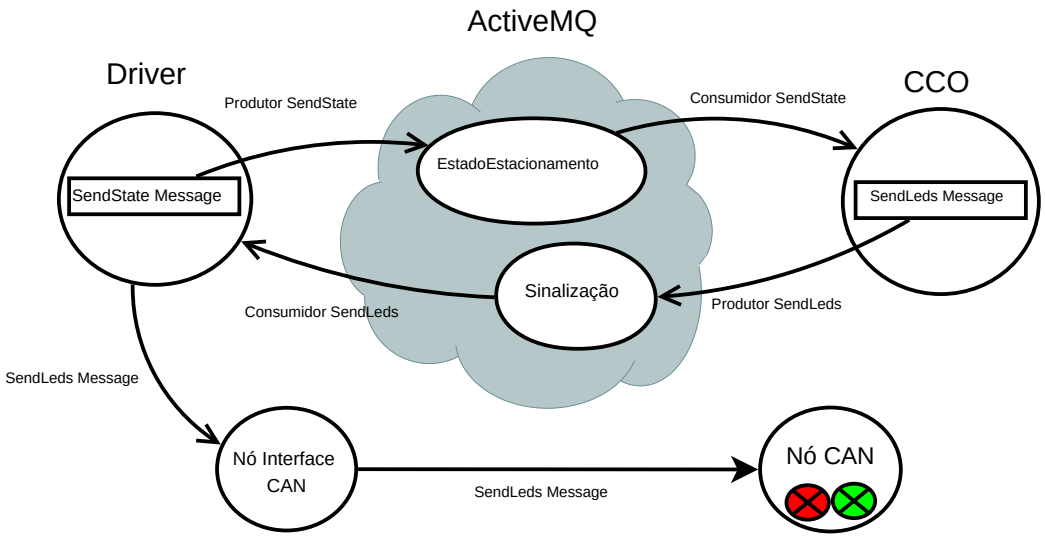

Figura 3.11 – Sinalização dos estacionamentos usando CAN.

cada um deles é guardado o estado. Os ID de estacionamento serão identificados a partir do endereço MAC dos nós presentes nos estacionamentos.

Após consumir a mensagem `SendState` da lista EstadoEstacionamento, o CCO instancia uma classe de uma outra aplicação, denominada DataBase, comunicando-lhe o endereço MAC do nó CAN em questão, assim como o seu estado, presente no campo de dados. A aplicação DataBase irá inicialmente fazer uma conexão à base de dados e a partir daí faz uma atualização do estado do nó, segundo o endereço MAC e o campo de dados recebido.

Após a base de dados ser atualizada, pretende-se mostrar os estacionamentos livres em cada local. Portanto, é criada uma página JSP que acede à tabela CAN da base de dados e conta os lugares livres para determinado local, que lhe é enviado como parâmetro. Para teste é criada uma página HTML (*HyperText Markup Language*) dinâmica que informa os lugares livres naquele local. Através do formato XML (*Extensible Markup Language*), esta informação poderá ser apresentada em painéis informativos junto dos locais, permitindo aos condutores identificar se existem lugares livres nos mesmos (e.g. Avenida Carvalho Araújo, Estacionamentos Livres: 9). Este processo é ilustrado na figura 3.12.

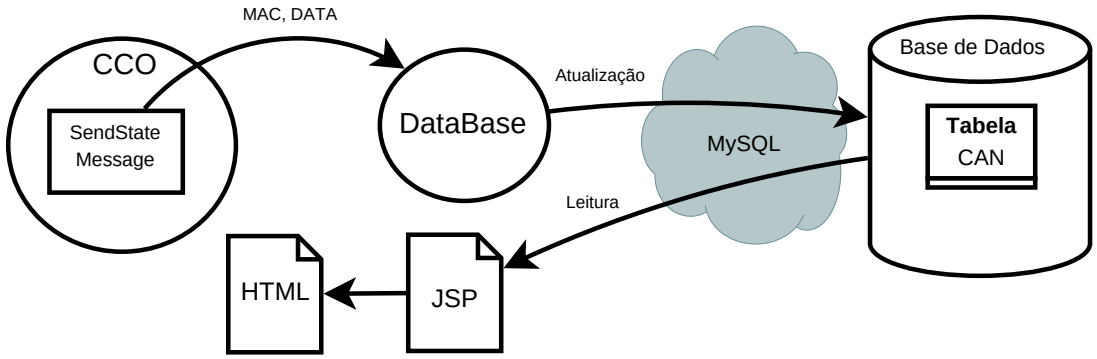

Figura 3.12 – Atualização e leitura da tabela CAN.

3.3 Implementação das comunicações IEEE 802.15.4

O processo de implementação do sistema, utilizando o protocolo de comunicação IEEE 802.15.4, engloba todo o sistema de comunicações que interliga desde os nós do estacionamento às aplicações CCO e DataBase. Esta descrição terá agora inicio com uma abordagem ao protocolo IEEE 802.15.4 utilizado.

3.3.1 Protocolo IEEE 802.15.4

A troca de mensagens no sistema de monitorização de estacionamentos é possível graças ao dispositivo XBee, comercializado pela Digi, que implementa o protocolo IEEE 802.15.4. Os XBee foram configurados no modo API (*Application Programming Interface*) que nos permite saber o endereço de origem do nó remetente de uma mensagem. No XBee existem dois tipos de *frames* de mensagens, as de transmissão, e as de receção, que foram utilizadas quer na comunicação entre os nós IEEE 802.15.4 e os nós de interface IEEE 802.15.4, quer na comunicação entre os nós de interface IEEE 802.15.4 e as aplicações.

A *frame* de transmissão utilizada, apresentada na figura 3.13, é composta pelos seguintes campos (Digi, 2009):

Start Delimiter *Byte* que carateriza o inicio da *frame* enviada;

Length *Bytes* mais e menos significativos do tamanho dos dados a enviar;

API *Identifier* Função do modo API;

***Frame* ID** *Byte* que identifica a *frame*, neste caso está a '0';

Destination Address *Bytes* mais e menos significativos do endereço de destino;

Options *Byte* com opções de *acknowledgement* e *broadcast*, neste caso está a '1';

Data *Bytes* de dados, neste caso são utilizados apenas dois. O *byte* [0] onde é identificado o tipo de mensagem e o *byte* [1] com informações de monitorização dos estacionamentos;

Checksum *Byte* que verifica a integridade da mensagem.

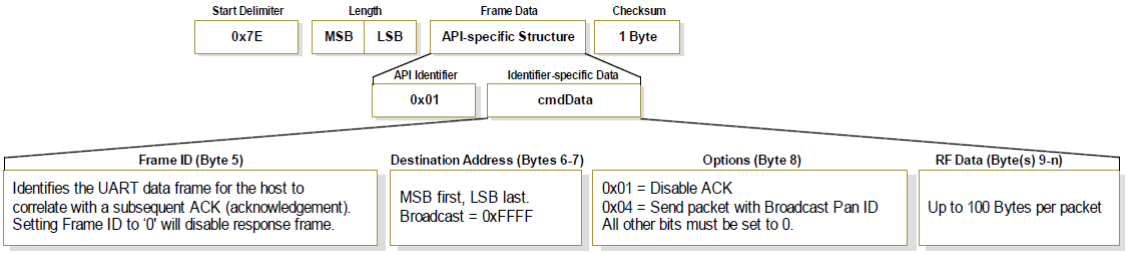

Figura 3.13 – *Frame* de transmissão utilizada no XBee (Digi, 2009).

A *frame* de receção utilizada, apresentada na figura 3.14, é composta pelos seguintes campos (Digi, 2009):

Start Delimiter *Byte* que carateriza o inicio da *frame* recebida;

Length *Bytes* mais e menos significativos do tamanho dos dados recebidos;

API *Identifier* Função do modo API;

Source Address *Bytes* mais e menos significativos do endereço de origem;

RSSI Tal como o nome indica, *Received Signal Strength Indicator*, este *byte* apresenta a potência do sinal recebido;

Options *Byte* com opções de *broadcast*;

Data *Bytes* de dados, neste caso são utilizados apenas dois. O *byte* [0] onde é identificado o tipo de mensagem e o *byte* [1] com informações de monitorização dos estacionamentos;

Checksum *Byte* que verifica a integridade da mensagem.

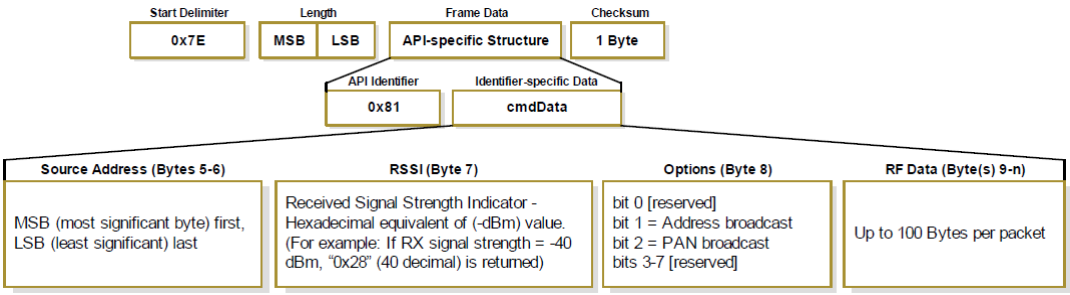

Figura 3.14 – *Frame* de receção utilizada no XBee (Digi, 2009).

Tipos de Mensagens IEEE 802.15.4

Foram projetados cinco tipos de mensagens consideradas fundamentais ao correto funcionamento do sistema. A tabela 3.2 apresenta essas mensagens bem como o seu percurso e campos relevantes da *frame* no protocolo.

Todos os nós são previamente identificados pelo seu endereço físico, que corresponde ao endereço atribuído ao módulo XBee. Ao ligá-los à rede, estes necessitam identificar o nó de interface, cuja ligação ao Driver IEEE 802.15.4 permite um sistema de comunicação entre o CCO e o próprio nó. Deste modo, foi criada a mensagem `FindInterface`, enviada para todos os nós (*Broadcast*), na tentativa de obter uma resposta do nó de interface IEEE 802.15.4. A mensagem é transmitida via *wireless*, e chegada ao nó de interface é diretamente encaminhada para o Driver.

Tabela 3.2 – Tipos de mensagens IEEE 802.15.4.

Mensagem	Origem-Destino	Campos da *frame*		
		Endereço de Destino	DATA[0]	DATA[1]
FindInterface	Nó-Driver	0xFFFF (*Broadcast*)	0	Não definido
SendInterface	Driver-Nó	Endereço do nó	1	Não definido
ActiveNode	Nó-Driver	Endereço Interface	2	Não definido
SendState	Nó-CCO	Endereço Interface	4	0/1 (livre/ocupado)
SendLeds	CCO-Nó	Endereço do nó	5	0/1 (verde/vermelho)

Após a sua receção, o Driver, cria uma mensagem de resposta, `SendInterface`, que é enviada para o nó de interface, e deste para o nó que a solicitou, fornecendo-lhe assim o endereço do nó de interface desejado.

O nó IEEE 802.15.4 guarda o endereço do nó de interface e envia uma mensagem de ativação, `ActiveNode`. Esta mensagem é transmitida para nó de interface, e diretamente encaminhada para o Driver, e tem como objetivo ativar o nó na rede.

A partir deste momento, o nó IEEE 802.15.4, pode enviar o estado do estacionamento (livre/ocupado), através da mensagem `SendState`, que chegada ao Driver é encaminhada para o CCO. Este, verifica o seu conteúdo e responde com a sinalização adequada através da mensagem `SendLeds`, que chegada ao Driver é transmitida para o nó de interface com destino ao nó IEEE 802.15.4. O nó, lê a mensagem, verificando o seu tipo, e ativa a iluminação correspondente (verde/vermelho).

3.3.2 Implementação do *Hardware*

Os recursos de *hardware* utilizados na implementação deste sistema, dizem respeito aos dois tipos de nós dimensionados, nós IEEE 802.15.4 e nós de interface IEEE 802.15.4.

Utilizados como terminal de comunicação, os nós IEEE 802.15.4 são colocados nos lugares de estacionamento com a finalidade de monitorizar e sinalizar o seu estado. Foram implementados como um módulo completo constituído pelos elementos descritos na figura 3.15, onde identificamos os seguintes blocos:

- Microcontrolador PIC18F2620 (Microchip, 2008), que permite comunicações com o módulo XBee e é programado para lidar com os periféricos de entrada/saída, como o interruptor e os LEDs de sinalização;
- Interruptor do tipo *on/off*, utilizado na simulação do estado do estacionamento. Gera os valores lógicos '0' e '1', que representam estado livre e ocupado respetivamente;
- LEDs vermelho e verde que nos indicam o estado do lugar, ocupado e livre respectivamente;
- Módulo de comunicações XBee-Pro (Digi, 2009), que implementa o protocolo IEEE 802.15.4. Este módulo, opera a 3.3V e permite comunicações na ordem da centena de metros.

Os nós de interface IEEE 802.15.4 têm como principal objetivo fazer a ligação entre os nós IEEE 802.15.4 e o Driver IEEE 802.15.4. Essa ligação, consiste na permuta de mensagens entre os dois sistemas e é assegurada pela porta série. Foi utilizada a placa de interface XBee comercializada atualmente pela Digi, conforme ilustrado na figura 3.16, onde podemos identificar nomeadamente o XBee-Pro responsável pelas comunicações entre os nós IEEE 802.15.4 e o Driver IEEE 802.15.4.

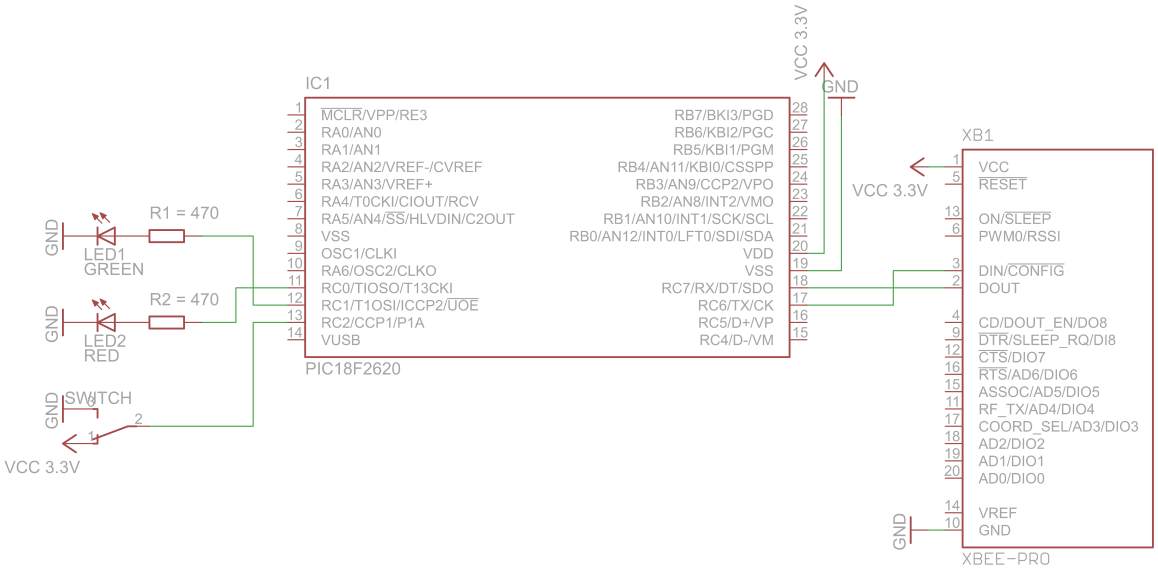

Figura 3.15 – Esquemático do nó IEEE 802.15.4.

Figura 3.16 – Nó de Interface IEEE 802.15.4 utilizado.

3.3.3 Implementação do *Firmware* e *Software*

Definido o protocolo e as implementações do *hardware*, é altura de abordar a implementação do *firmware* e *software* utilizados no desenvolvimento do sistema. No que diz respeito aos nós IEEE 802.15.4, o *firmware* foi programado na linguagem C

utilizando a ferramenta MPLAB. O *software* das aplicações das camadas superiores foi implementado na linguagem Java com recurso ao NetBeans. Tentar-se-á aproximar esta descrição o mais possível ao processo de comunicação, obedecendo assim à sua ordem de funcionamento.

Procura do endereço do nó de Interface

O processo de comunicação na rede parte do pressuposto que todos os nós estão devidamente identificados, portanto essa é a primeira tarefa realizada no momento que antecede a inserção de um novo nó. Utilizando a ferramenta XCTU, da Digi, foi configurado um endereço MAC de 16 *bits* para cada XBee utilizado. Este endereço, por sua vez, identificará o próprio nó IEEE 802.15.4.

Para poder iniciar o processo de comunicação, o nó IEEE 802.15.4, necessita ainda de saber o endereço do nó de interface, que assegurará as comunicações com o Driver. A função FindInterface é executada sempre que um nó se liga e cria uma mensagem, também denominada `FindInterface`, cujo endereço de destino é '0xFFFF' (*Broadcast*) e o seu ID de mensagem, indicado no campo Data[0], é '0'. A mensagem é colocada num *buffer* de envio que vai guardando as mensagens a serem enviadas. Paralelamente existe a execução da função Task, que tem como objetivo enviar as mensagens em espera no *buffer*, recorrendo à função Sendframe.

Após ser transmitida no meio, a mensagem será recebida, entre outros, pelo nó de interface IEEE 802.15.4, que a encaminha para o Driver, aplicação em Java que é executada num computador. Esta aplicação, começa a sua execução por abrir uma ligação à porta série de modo que possa receber as mensagens vindas desse meio. Após efetuada a ligação, verifica recorrentemente a receção de mensagens vindas da porta série, caso isto aconteça, analisa o primeiro campo da mensagem (*Start Delimiter*), de modo a averiguar se esta se trata de uma mensagem do protocolo. Após esta verificação, a mensagem é guardada e lido o seu ID, indicativo do tipo de mensagem. Por tratar-se da mensagem `FindInterface`, o ID será '0'. É criada a mensagem `SendInterface`, com o ID de mensagem '1', e cujo endereço de destino

corresponde ao endereço do nó que efetuou o pedido. Esta mensagem é enviada pela porta série até ao nó de interface IEEE 802.15.4, que a transmite via *wireless* para o nó de destino.

O nó IEEE 802.15.4, ao executar a função Task, deteta a receção de uma mensagem, recorrendo a uma outra função denominada Receiveframe. A mensagem é colocada num *buffer* de receção e lido o seu ID, cujo valor '1' indica tratar-se da mensagem `SendInterface`. O endereço de origem desta mensagem é o do nó de interface IEEE 802.15.4, que é guardado pelo nó para posteriores comunicações. Este processo é ilustrado na figura 3.17.

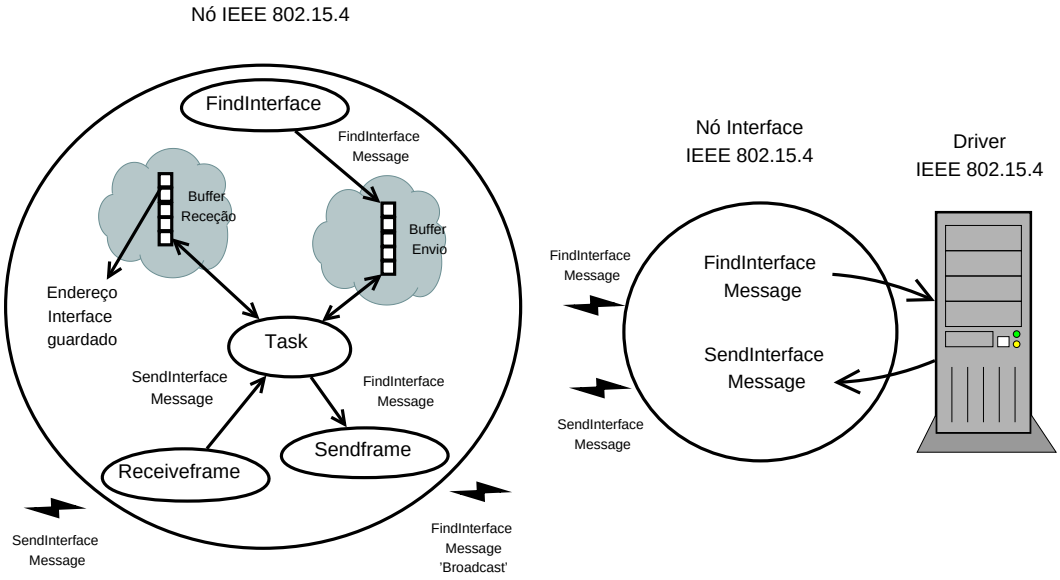

Figura 3.17 – Procura e receção do endereço do nó de Interface.

Com o objetivo de se tornar ativo na rede, o nó executa a função ActiveNode, que cria a mensagem `ActiveNode`, cujo ID de mensagem é '2' e o endereço de destino é o do nó de interface guardado anteriormente. Esta mensagem é de igual forma colocada no *buffer* e transmitida para o nó de interface IEEE 802.15.4 que a encaminha para o Driver. Após analisar a mensagem, o Driver introduz o endereço de origem, o do nó remetente da mensagem, numa lista de ID onde se encontram os nós ativos da rede. Posteriores comunicações só serão respondidas pelo Driver caso o nó em questão se encontre nesta lista.

Monitorização e Sinalização

Após o processo de procura do endereço do nó de interface, os nós IEEE 802.15.4 podem comunicar o estado do estacionamento com o sistema. Tal como no caso do sistema CAN, este estado foi simulado por um interruptor do tipo *on/off* com duas posições, '0' e '1', que indicam o estado livre e ocupado respetivamente. Esta simulação é feita recorrendo à função SwitchState, máquina de estados que é executada recorrentemente nos nós IEEE 802.15.4, caso estes já possuam o endereço do nó de interface guardado. A estrutura da máquina de estados é igual à apresentada no sistema CAN, figura 3.10, e possui os seguintes estados:

1. Neste estado é verificado o valor de entrada que chega do interruptor. Caso seja '0', estacionamento livre, é criada a mensagem `SendState`, cujo endereço de destino é o endereço do nó de interface, o ID da mensagem no campo Data[0] é '4' e o estado do estacionamento no campo Data[1] é '0'. A mensagem é colocada no *buffer* de envio e passamos ao estado 2. Caso o valor de entrada seja '1', estacionamento ocupado, é criada a mensagem `SendState`, cujo endereço de destino é o endereço do nó de interface, o ID da mensagem no campo Data[0] é '4' e o estado do estacionamento no campo Data[1] é '1'. A mensagem é colocada no *buffer* de envio e passamos ao estado 3;

2. Aqui é verificado se o estado do interruptor mudou para a posição '1', indicando que passou a estar ocupado. Caso isto aconteça, retomamos ao estado 1;

3. Aqui é verificado se o estado do interruptor mudou para a posição '0', indicando que passou a estar livre. Caso isto aconteça, retomamos ao estado 1.

Tal como no processo de procura do endereço do nó de interface, a mensagem `SendState` é transmitida via *wireless* para o endereço de destino, o nó de interface IEEE 802.15.4, que a encaminha para o Driver. Este, verifica mais uma vez o ID da mensagem, que é '4', tratando-se da mensagem que contém a informação do estado do estacionamento num determinado nó. Como no processo CAN, esta

informação será comunicada ao CCO através do ActiveMQ, tornando o Driver IEEE 802.15.4 e o CCO em produtores e consumidores de mensagens. Como o endereço do nó IEEE 802.15.4 se encontra na lista de nós ativos, o Driver efetua uma conexão com o ActiveMQ, e produz a mensagem na lista EstadoEstacionamento. A mensagem produzida é a `SendState`, desprovida de alguns campos que nesta fase não serão utilizados, como o *Start Delimiter*, o *Length*, o API *Identifier*, o RSSI, o *Options* e o *Checksum*. É assim composta por, ID do Driver, para posterior identificação do Driver correto, endereço do nó remetente da mensagem e campo de dados, que indica o ID da mensagem e o estado do estacionamento.

Em constante execução, o CCO consome a mensagem da lista EstadoEstacionamento. Todos os parâmetros da mensagem são guardados, e os dados são analisados. Caso o campo Data[1] seja igual a '0', estado livre, é criada a mensagem `SendLeds`, cujo ID da mensagem é '5' e o campo Data[1] se mantém '0', que simboliza a sinalização verde a enviar ao nó. Caso o campo Data[1] seja igual a '1', estado ocupado, é criada a mensagem `SendLeds`, cujo ID da mensagem é '5' e campo Data[1] se mantém '1', que simboliza a sinalização vermelha a enviar ao nó. Funcionando como produtor, o CCO envia a mensagem `SendLeds` para a lista Sinalizacao.

Nesta fase, o Driver consome a mensagem `SendLeds` da lista Sinalizacao, guarda os seus parâmetros e verifica se o ID de Driver presente na mensagem é igual ao seu, confirmando que é o Driver destinatário daquela mensagem. A mensagem `SendLeds` é enviada para o nó de interface IEEE 802.15.4, e transmitida por este, ao endereço de destino, que corresponde ao nó que informou o estado do estacionamento.

A mensagem `SendLeds` é recebida no nó IEEE 802.15.4, que verifica o seu ID de mensagem, neste caso é '5', e ativa a sinalização LED verde, caso o campo Data[1] seja '0', ou a sinalização LED vermelha, caso o campo Data[1] seja '1'. Este processo é ilustrado na figura 3.18.

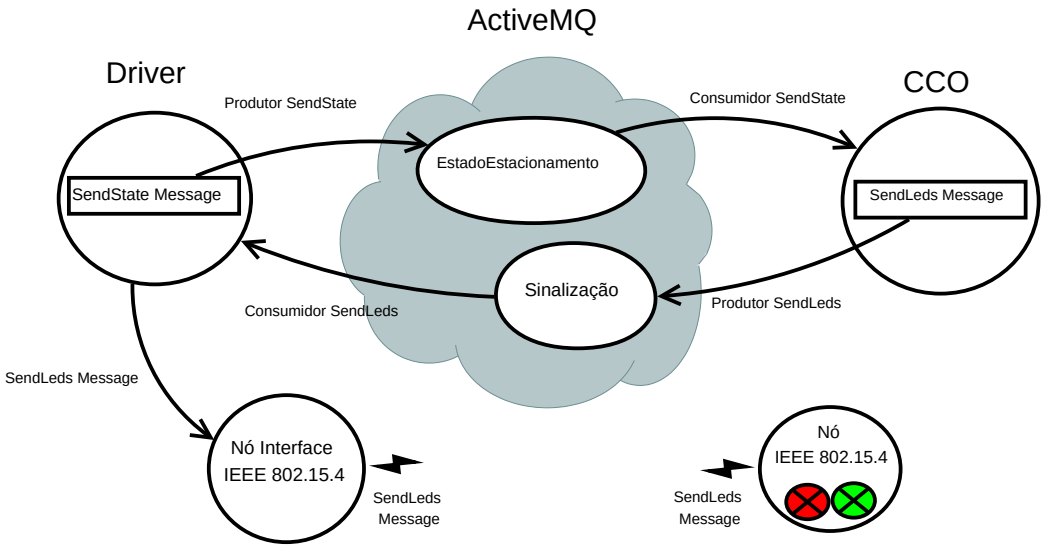

Figura 3.18 – Sinalização dos estacionamentos usando IEEE 802.15.4.

Base de Dados

Tal como no sistema CAN, a informação do estado dos estacionamentos é guardada na base de dados, para que possa ser utilizada em análises posteriores. Na tabela IEEE 802.15.4, com os campos ID do Local, ID do Estacionamento e Estado do Estacionamento, é guardado o estado de cada estacionamento presente num determinado local a monitorizar pela tecnologia IEEE 802.15.4. Os ID de estacionamento são identificados a partir do endereço do nó IEEE 802.15.4, programado no módulo XBee.

Após consumir a mensagem `SendState` da lista EstadoEstacionamento, o CCO instancia uma classe de uma outra aplicação, denominada DataBase, comunicando-lhe o endereço do nó IEEE 802.15.4 em questão, assim como o seu estado presente no campo Data[1]. A aplicação DataBase irá inicialmente fazer uma conexão à base de dados e a partir daí faz uma atualização do estado do nó, segundo o endereço e o campo de dados recebido.

Após a base de dados ser atualizada, pretende-se mostrar os estacionamentos livres em cada local. Portanto, é criada uma página JSP que acede à tabela IEEE 802.15.4

da base de dados e conta os lugares livres para determinado local, que lhe é enviado como parâmetro. Para teste é criada uma página HTML (*HyperText Markup Language*) dinâmica que informa os lugares livres naquele local. Através do formato XML (*Extensible Markup Language*), esta informação poderá ser apresentada em painéis informativos junto dos locais, permitindo aos condutores identificar se existem lugares livres nos mesmos (e.g. Avenida Europa, Estacionamentos Livres: 10). Este processo é ilustrado na figura 3.19.

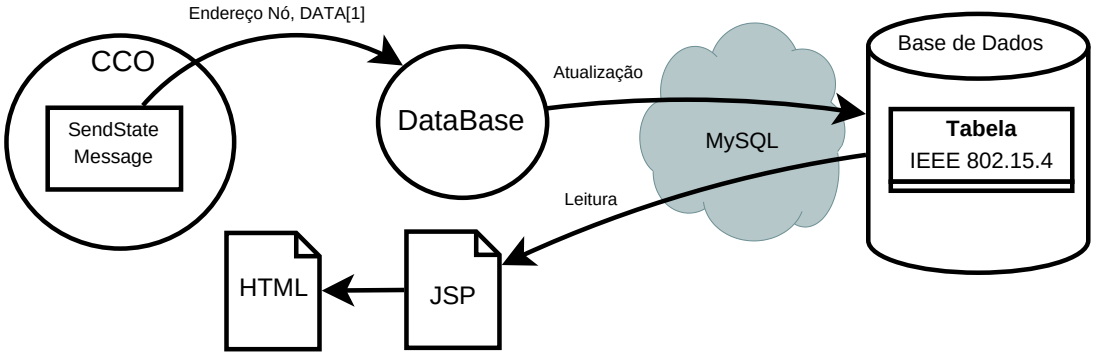

Figura 3.19 – Atualização e leitura da tabela IEEE 802.15.4.

4 Testes e Resultados

Com base nos protótipos de *hardware*, *firmware* e *software* desenvolvidos, foram efetuados diversos testes, cujos resultados são agora apresentados tendo em conta cada um dos sistemas.

4.1 Sistema CAN

Foi implementado um barramento com nós CAN e um nó de interface CAN. Sempre que um nó CAN se liga ao barramento, envia a mensagem `SendMAC` para o nó de interface CAN que a encaminha para o Driver. Nesta aplicação, a mensagem é identificada e é atribuído ID ao nó, segundo a posição em que este é guardado na lista de ID. É criada a mensagem `SendID`, e enviada como resposta ao nó CAN. A figura 4.1 ilustra o processo descrito a ocorrer no Driver CAN.

```
Mensagem CAN Recebida:
7E000600000100   Pedido de ID, para MAC:1
Mensagem CAN Enviada:
7E0410600000100   Envio de ID:1
```

Figura 4.1 – Teste de pedido e atribuição de ID.

Após o processo de pedido e atribuição de ID, os nós podem comunicar o estado do estacionamento. Simulado no interruptor, este estado é enviado na mensagem `SendState`, que é transmitida para o Driver através do nó de interface CAN. Funcionando como produtor, o Driver produz a mensagem `SendState` na lista Estado-Estacionamento. Por sua vez, a aplicação CCO, consome a mensagem da referida lista, analisa-a e cria a mensagem de resposta `SendLeds` com a informação de sinalização. Esta mensagem é produzida na lista Sinalizacao e consumida pelo Driver, que a envia para o nó CAN, via nó de interface CAN. A figura 4.2 ilustra o processo descrito, para o nó de ID '1', a ocorrer no Driver CAN e no CCO.

```
Mensagem CAN Recebida:
7E0440110000100   Estado Estacionamento (1) no nó:1
DRIVER_CAN -> Mensagem: 'IDDRIVER:1 IDH:0 IDL:68 RTR:0 DLC:1 DATA[0]:1 MAC:1' enviada para lista: ESTADOESTACIONAMENTO

        CCO -> Mensagem: 'IDDRIVER:1 IDH:0 IDL:68 RTR:0 DLC:1 DATA[0]:1 MAC:1' recebida da lista: ESTADOESTACIONAMENTO
        CCO -> Mensagem: 'IDDRIVER:1 IDH:0 IDL:69 RTR:0 DLC:1 DATA[0]:1' enviada para lista: SINALIZACAO

DRIVER_CAN -> Mensagem: 'IDDRIVER:1 IDH:0 IDL:69 RTR:0 DLC:1 DATA[0]:1' recebida da lista: SINALIZACAO
Mensagem CAN Enviada:
7E0450110000100   Envio de Sinalização (1) para nó:1
```

Figura 4.2 – Teste de monitorização e sinalização no sistema CAN.

Chegada ao nó CAN, a mensagem `SendLeds` é lida e ativada a sinalização correspondente. Neste teste, foi enviado o estado ocupado, portanto é ativada a sinalização vermelha como ilustrado na figura 4.3.

Recorrendo à aplicação DataBase, o CCO atualiza também o estado do estacionamento do nó em questão, na tabela CAN da base de dados. A figura 4.4, ilustra este processo a ocorrer no CCO e na referida tabela.

Após a atualização, e recorrendo a uma JSP, é feita uma contagem dos estacionamentos livres para cada local da tabela CAN. Este local é enviado como parâmetro, e a informação é apresentada sob a forma de uma página HTML dinâmica conforme ilustrado na figura 4.5.

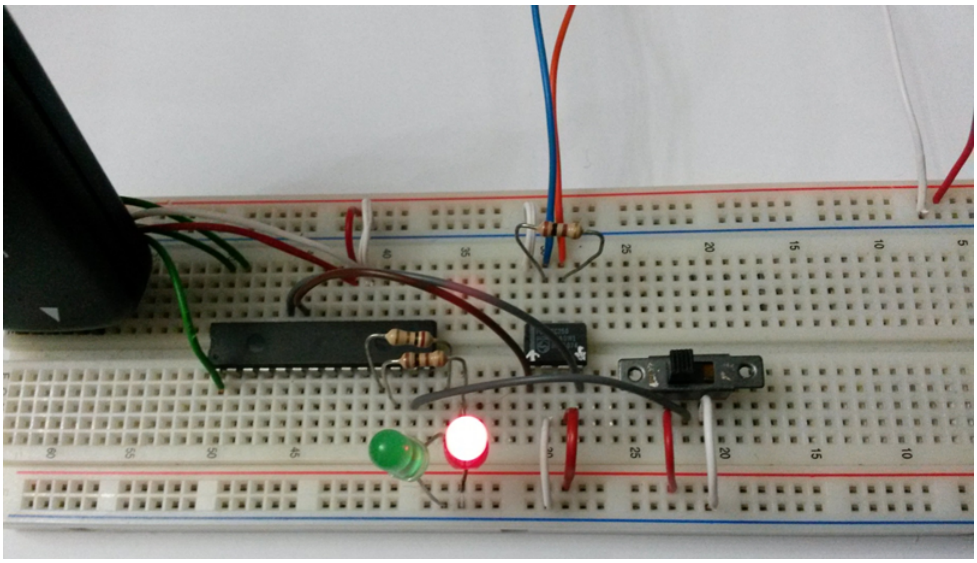

Figura 4.3 – Nó CAN sinalizado.

```
CCO -> Mensagem: 'IDDRIVER:1 IDH:0 IDL:68 RTR:0 DLC:1 DATA[0]:1 MAC:1' recebida da lista: ESTADOESTACIONAMENTO
Tabela CAN atualizada com sucesso. Nó:1 Estado:1
```

ID_LOCAL	ID_ESTACIONAMENTO	ESTADO_ESTACIONAMENTO
1	1	1
1	2	0
1	3	0

Figura 4.4 – Atualização do estado do nó na tabela CAN.

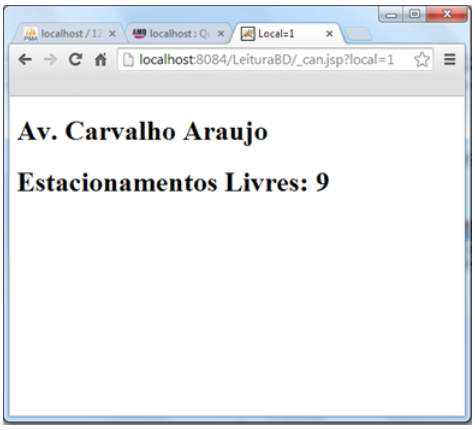

Figura 4.5 – Lugares disponíveis no local de ID '1' da tabela CAN.

4.2 Sistema IEEE 802.15.4

A comunicação neste sistema é feita através dos módulos XBee presentes em cada nó, cujos endereços atribuídos identificam o próprio nó. Sempre que um nó IEEE 802.15.4 se liga ao sistema, envia a mensagem `FindInterface` em *broadcast*. Esta é recebida pelos nós, incluindo o nó de interface IEEE 802.15.4 que a encaminha para o Driver. Nesta aplicação, a mensagem é identificada e criada a resposta `SendInterface`, enviada para o nó IEEE 802.15.4 via nó de interface. Chegada ao nó IEEE 802.15.4, a mensagem é lida e guardado o endereço do nó de interface para futuras comunicações. A figura 4.6 ilustra o processo descrito a ocorrer no Driver IEEE 802.15.4.

```
Mensagem IEEE 802.15.4 recebida:
7E0781013220049 -> Nó 1 procura Interface
Mensagem IEEE 802.15.4 Enviada:
7E071001110FB -> Envio de endereço de Interface para Nó 1
```

Figura 4.6 – Teste de pedido de endereço de interface.

Uma vez que já pode comunicar diretamente com o nó de interface, o nó IEEE 802.15.4, envia a mensagem `ActiveNode`, com o objetivo de se tornar um nó ativo na rede. O Driver introduz o endereço do nó na lista de ID, ativando este nó para posteriores comunicações. A figura 4.7 ilustra este processo a ocorrer no Driver IEEE 802.15.4.

```
Mensagem IEEE 802.15.4 recebida:
7E0781012E0204D -> Nó 1 ativado
Adiconado à lista IDs
```

Figura 4.7 – Teste de ativação de nó IEEE 802.15.4.

Após possuir o endereço do nó de interface e passar ao estado ativo, o nó pode comunicar o estado do estacionamento com o Driver, processo este, semelhante ao ocorrido no CAN. Simulado no interruptor, este estado é enviado na mensagem `SendState`, que é transmitida para o Driver através do nó de interface IEEE

802.15.4. Funcionando como produtor, o Driver produz a mensagem `SendState` na lista EstadoEstacionamento. Por sua vez, o CCO, consome a mensagem da referida lista, analisa-a e cria a mensagem de resposta `SendLeds` com a informação de sinalização. Esta mensagem é produzida na lista Sinalizacao e consumida pelo Driver, que a envia para o nó IEEE 802.15.4, via nó de interface. A figura 4.8 ilustra o processo descrito a ocorrer no Driver IEEE 802.15.4 e no CCO.

```
Mensagem IEEE 802.15.4 recebida:
7E0781012F04149 -> Estado de estacionamento no Nó 1 é 1
DRIVER IEEE 802.15.4 -> Mensagem: 'IDDRIVER:2 SourceAddressH:0 SourceAddressL:1 DATA[0]:4 DATA[1]:1' enviada para lista: ESTADOESTACIONAMENTO

                CCO -> Mensagem: 'IDDRIVER:2 SourceAddressH:0 SourceAddressL:1 DATA[0]:4 DATA[1]:1' recebida da lista: ESTADOESTACIONAMENTO
                CCO -> Mensagem: 'IDDRIVER:2 SourceAddressH:0 SourceAddressL:1 DATA[0]:5 DATA[1]:1' enviada para lista: SINALIZACAO

DRIVER IEEE 802.15.4 -> Mensagem: 'IDDRIVER:2 SourceAddressH:0 SourceAddressL:1 DATA[0]:5 DATA[1]:1' recebida da lista: SINALIZACAO
Mensagem IEEE 802.15.4 Enviada:
7E071001151F6 -> Envio de Sinalização 1 para nó:1
```

Figura 4.8 – Teste de monitorização e sinalização usando o IEEE 802.15.4.

Chegada ao nó IEEE 802.15.4, a mensagem `SendLeds` é lida e ativada a sinalização correspondente. Neste teste, como foi enviado o estado ocupado, é ativada a sinalização vermelha como ilustrado na figura 4.9.

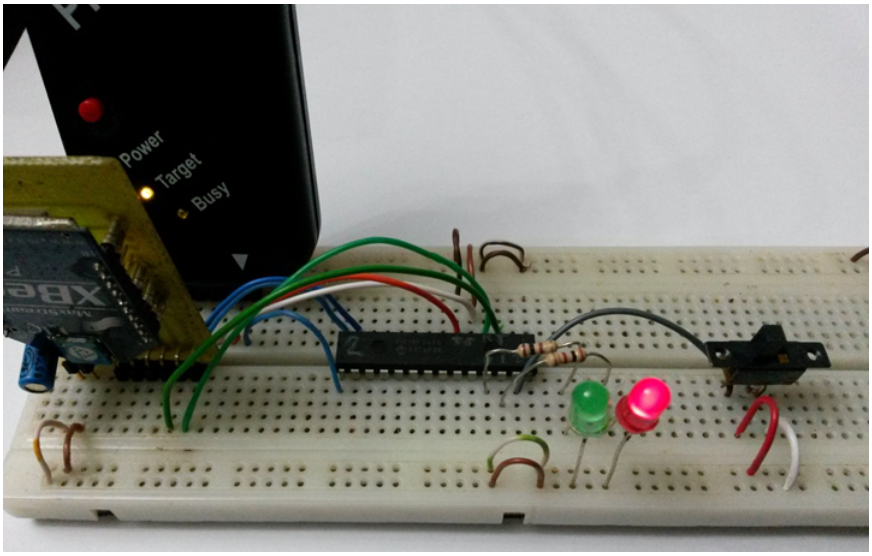

Figura 4.9 – Nó IEEE 802.15.4 sinalizado.

Recorrendo à aplicação DataBase, o CCO atualiza também o estado do estacionamento do nó em questão na tabela IEEE 802.15.4 da base de dados. A figura 4.10, ilustra este processo a ocorrer no CCO e na referida tabela.

```
CCO -> Mensagem: 'IDDRIVER:2 SourceAddressH:0 SourceAddressL:1 DATA[0]:4 DATA[1]:1' recebida da lista: ESTADOESTACIONAMENTO
Tabela IEEE 802.15.4 atualizada com sucesso. Nó:1 Estado:1
```

ID_LOCAL	ID_ESTACIONAMENTO	ESTADO_ESTACIONAMENTO
2	1	1
2	2	0

Figura 4.10 – Atualização do estado do nó na tabela IEEE 802.15.4.

Após a atualização, e tal como no sistema CAN, é feita uma contagem dos estacionamentos livres para cada local da tabela IEEE 802.15.4. Esta contagem é feita por uma JSP, que recebe o local como parâmetro. A informação é apresentada sob a forma de uma página HTML dinâmica conforme ilustrado na figura 4.11.

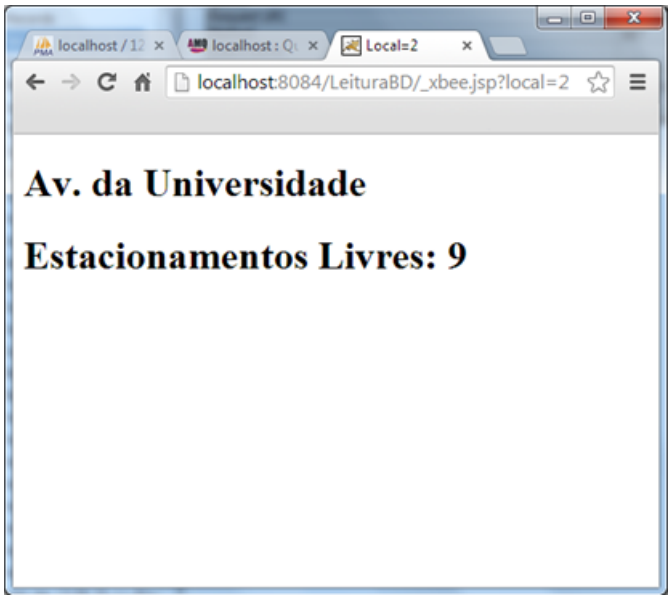

Figura 4.11 – Lugares disponíveis no local de ID '2' da tabela IEEE 802.15.4.

5 Conclusão e trabalho futuro

Sendo o tráfego urbano, um dos grandes problemas com que se deparam as cidades na atualidade, sentiu-se a necessidade de aplicar e aprofundar conhecimentos na criação de um sistema que minorasse o problema. Desse modo, o trabalho efetuado, descrito ao longo do documento, teve como objetivo o desenvolvimento de um sistema de comunicações para monitorizar os estacionamentos numa rede urbana.

Foram criados dois sistemas, baseados em dois protocolos, que permitem a comunicação entre os nós presentes nos estacionamentos e aplicações de controlo dos mesmos. A versatilidade destes dois protocolos permite ainda a implementação do sistema tendo em conta a arquitetura do local (e.g paralelo e asfalto).

Recorrendo aos protocolos CAN e IEEE 802.15.4, foi possível o desenvolvimento de sistemas com implementações relativamente simples. Os testes efetuados revelaram uma escolha acertada no que aos protocolos diz respeito, muito embora tenham sido efetuados em condições favoráveis, próprias deste tipo de implementações. O baixo comprimento e baixo número de nós no barramento CAN, não permitiram testar possíveis problemas na velocidade de entrega de mensagens. Já no IEEE 802.15.4, os testes foram realizados com os nós relativamente perto uns dos outros e alimentados pelo PICkit, pelo que não se testou o raio de cobertura e durabilidade da bateria

nos mesmos.

Ainda assim, conclui-se que este é um sistema que reúne as condições necessárias à implementação num cenário real, permitindo uma monitorização sobre os estacionamentos, que agilizaria este processo e diminuiria o tráfego que dele advém.

Este trabalho ofereceu ao seu autor uma excelente sensibilidade para o conhecimento científico-tecnológico multidisciplinar, bem como o aprofundamento e exploração de conhecimentos adquiridos nos ciclos de estudos realizados até aqui.

5.1 Trabalho Futuro

Tendo em conta que este trabalho surge num projeto mais abrangente, existem especificações como a deteção de veículos recorrendo a sensores, processamento de dados no CCO e sinalização LED, realizadas por outros intervenientes, que seriam uma mais valia se integradas neste projeto.

Como trabalho futuro é apontada também a integração dos dois sistemas criados, que permitira, por exemplo, a um nó IEEE 802.15.4 comunicar com o CCO através do barramento CAN. Isto, na tentativa de desenvolver um sistema o mais genérico, dinâmico e abrangente possível.

Futuramente poder-se-á adaptar a rede de comunicações implementada para usufruto de outra qualquer ocorrência citadina a controlar. Por exemplo informação do estado do piso, vias cortadas, e quaisquer outras informações que facilitem a circulação.

Como parte integrante deste subsistema, é importante o melhoramento do *firmware* implementado nos nós, e o *software* das aplicações, de modo a que o sistema consiga lidar com o mais variado leque de situações de possível ocorrência. Surge também a necessidade do acréscimo da componente segurança, nomeadamente na troca de mensagens dentro do sistema. No que ao *hardware* diz respeito, o objetivo passará por criar módulos PCB (*Printed Circuit Board*) adaptáveis aos estacionamentos a

controlar.

Referências bibliográficas

Araújo, M. P. V. (2008). Planeamento e Dimensionamento de Redes WiMax. Master's thesis, Universidade de Aveiro, Departamento de Electrónica, Telecomunicações e Informática, Aveiro, Portugal. 18, 19

Borges, F. (2007). *Redes de Comunicação Industrial*. Schneider Electric. 20, 24, 25, 27

Corrigan, S. (2008). *Introduction to the Controller Area Network (CAN)*. Texas Instruments. 20, 21, 22

Coulouris, G., Dollimore, J., and Kindberg, T. (2007). *Sistemas Distribuidos - Conceitos e Projeto*. Bookman, fourth edition. 13, 14

da Silva Carissimi, A., Rochol, J., and Zambenedetti, L. (2009). *Redes de Computadores*. Bookman, Porto Alegre. 9

de Sousa, J. A. C. (2014). Desenvolvimento de um Sistema de Controlo de Iluminaçao Pública e de Sinalizaçao Rodoviária. Master's thesis, Universidade de Trás-os-Montes e Alto Douro, Departamento de Engenharias, Vila Real, Portugal.

Digi (2009). *XBee / XBee-Pro Data Sheet*. Digi International Inc. 44, 45, 46, 48

Echelon (1999). *Introduction to the LONWORKS System*. Echelon Corporation. 22, 24

Fernandes, A. S. L. (2011). Wirelessteams: Comparação de Tecnologias sem Fios em Equipas de Robôs Movéis. 6, 9, 15, 16, 17, 26

Forouzan, B. A. (2008). *Comunicação de Dados e Redes de Computadores*. McGraw Hill, fourth edition. 5, 6, 7, 8, 10, 11, 14

Fourozan, B. A. and Fegan, S. C. (2008). *Protocolo TCP/IP*. McGraw Hill Brasil, third edition. 8, 12

GmbH, R. B. (1991). *CAN Specification - version 2.0*. Bosch. 20

Gupta, N. (2013). *Inside Bluetooth Low Energy*. Artech House. 13

IEEE (2006). *IEEE Standard for Information Technology Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. The Institute of Electrical and Electronics Engineers, Inc. 15

Instruments, T. (2004). *MAX232 Data Sheet*. Texas Instruments Incorporated. 36

Álvaro Jorge da Maia Seco, Gonçalves, J. H. G., and da Costa, A. H. P. (2008). *Manual do Planeamento de Acessibilidades e Transportes*. Comissão de Coordenação e Desenvolvimento Regional do Norte. 1

Matos, L., Rossi, N., Gnidarchichi, T., Chiotolli, F., Summa, G. L., Lin, S., and Mazzei, M. (2011). *Curso Essencial de Redes Wireless*. Digerati Books, São Paulo. 18

Mestre, P., Serôdio, C., Matias, J., Couto, P., Guedes, R., and Fernandes, J. (2013). Vehicle Detection for Outdoor Car Parks using IEEE802.15.4.

Mestre, P., Serôdio, C., Matias, J., Monteiro, J., and Couto, C. (2010a). Java in the Loop of Data Acquisition Systems.

Mestre, P., Serôdio, C., and Monteiro, J. (2008). A Java-based Controller Area Network Device Driver for Utilization in Data Acquisition and Actuation Systems.

Mestre, P., Serôdio, C., Morais, R., Azevedo, J., and Pinto, P. M. (2010b). Vegetation Growth Detection Using Wireless Sensor Networks.

Mestre, P., Serôdio, C., Ribeiro, J., and Monteiro, J. (2011). Propagation of IEEE802.15.4 in Vegetation.

Microchip (2007). *PIC18F2680 Data Sheet*. Microchip Technology Inc. 35, 36

Microchip (2008). *PIC18F2620 Data Sheet*. Microchip Technology Inc. 48

Morais, R., Serôdio, C., and Valente, A. (2005). A Wireless Sensor Network for Smart Irrigation and Environmental Monitoring: A position article.

Nogueira, T. A. (2009). Redes de Comunicação para Sistemas de Automação Industrial. Master's thesis, Universidade federal de Ouro Preto, Escola de Minas, Engenharia de Controlo e Automação.

Philips (2000). *PCA82C250 CAN Controller Interface*. Philips Semiconductors. 35, 36

Siemens (2009). *PROFIBUS Network Manual*. Siemens AG. 25