

Universidade do Minho
Escola de Engenharia

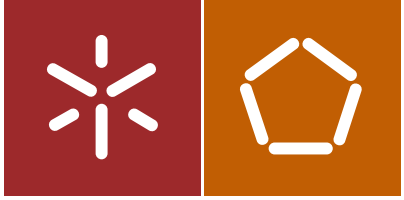
Rita Baldaia da Costa e Silva

Interface Homem-Máquina para Carro Elétrico baseada em Bluetooth Low Energy

outubro de 2016

Rita Baldaia da Costa e Silva Interface Homem-Máquina para Carro Elétrico baseada em Bluetooth Low Energy

UMinho | 2016



Universidade do Minho
Escola de Engenharia

Rita Baldaia da Costa e Silva

Interface Homem-Máquina para Carro Elétrico baseada em Bluetooth Low Energy

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia de Telecomunicações e Informática

Trabalho efetuado sob a orientação do
Professor Doutor José Augusto Afonso
Professor Doutor João Luiz Afonso

outubro de 2016

Agradecimentos

O presente trabalho requer um imprescindível agradecimento a algumas pessoas. Assim, gostaria de aproveitar esta oportunidade para o fazer.

Em primeiro lugar, gostaria de mostrar o meu reconhecimento ao Professor Doutor José Augusto Afonso pela orientação, disponibilidade e paciência durante a realização deste trabalho. Agradeço, também, a colaboração do Professor Doutor João Luiz Afonso.

Ao Engenheiro Delfim Pedrosa e Engenheiro Vítor Monteiro pela disponibilidade que demonstraram para ajudar e apoiar a realização deste projeto, o meu agradecimento.

Gostaria, também, de expressar gratidão aos meus amigos, que tiveram um papel fundamental ao longo do meu percurso, pela amizade e apoio que me souberam dar.

Ao Rui, um agradecimento especial, pela compreensão, paciência e companheirismo em todos os momentos da minha vida académica e pessoal.

Por último, nada disto teria sido possível sem o apoio incondicional dos meus pais. Agradeço os sacrifícios, confiança e apoio prestado em todas a fases do meu percurso académico e, em especial, para concretizar esta dissertação.

Resumo

Os carros hoje em dia integram geralmente uma rede cablada de controladores conhecida como Controller Area Network (CAN), que estabelece a comunicação entre o computador de bordo de um carro e pequenas unidades de controlo eletrónicas (ECU – Electronic Control Unit). A crescente complexidade dos veículos implica um aumento na cablagem para possibilitar as conexões entre as ECUs. O uso de tecnologias sem fios foi proposto para a eliminação dos cabos, proporcionando a diminuição do peso, melhorando aspetos como o desempenho e a economia de combustível. Entre as várias tecnologias existentes, a escolhida para este trabalho foi o Bluetooth Low Energy (BLE).

A finalidade desta dissertação é desenvolver um sistema de comunicação entre diversas partes de um carro elétrico e o seu utilizador através do uso de sensores ligados a módulos BLE e um dispositivo móvel. Para este propósito, efetuou-se uma série de procedimentos que visaram atingir diferentes objetivos: implementar e testar o desempenho da rede de comunicações sem fios BLE; desenvolver um protocolo de comunicação sobre UART para a recolha dos dados de sensores através de módulos BLE; desenvolver uma interface homem-máquina entre o condutor e o veículo com base num dispositivo móvel Android; integrar as partes desenvolvidas e testar o sistema no carro elétrico.

A implementação da rede BLE teve em conta duas configurações: uma tendo como base somente módulos BLE CC2540, da Texas Instruments, e outra integrando tanto estes módulos como um *smartphone* Android. A implementação foi feita modularmente, com o objetivo de facilitar a identificação de problemas e respetivas soluções. O desempenho das diferentes partes do sistema desenvolvido foi avaliado com o auxílio de testes experimentais.

Abstract

Most of modern cars include a wired network of controllers known as Controller Area Network (CAN), which establishes communication between the on-board car computer and small electronic control units (ECU). However, the increasing vehicle complexity leads to additional wiring to make connections among ECUs possible. The use of wireless technology was proposed to eliminate cables, providing a reduction in weight and improving performance and fuel economy. Among the multiple available wireless technologies, the one proposed for this work was Bluetooth Low Energy (BLE).

The purpose of this dissertation is to develop a communication system among the different sections of an electric car and its user, through the use of sensors connected to BLE modules and a mobile device. In order to achieve this goal, a set of procedures was followed to perform different tasks and attain specific objectives: implement and test the performance of the Bluetooth Low Energy network (BLE); develop a communication protocol and carry out code tests to collect data from sensors through in-vehicle BLE modules; develop a graphic interface for an Android mobile device, to play the role of human-machine Interface); integrate all the developed parts and test the system in the electric car.

The BLE network implementation was based on two configurations: one using only CC2540 BLE modules, and another integrating both these modules and an Android smartphone. The performance of the different parts of the developed system was evaluated with the aid of experimental tests.

Índice de conteúdos

Agradecimentos ..... **i**

Resumo ..... **iii**

Abstract ..... **v**

Índice de conteúdos ..... **vii**

Lista de figuras ..... **xi**

Lista de tabelas ..... **xv**

Lista de abreviaturas **16**

1. Introdução **19**

1.1 Enquadramento e motivação 19

1.2 Objetivos 21

1.3 Estrutura da dissertação 22

2. Estado da Arte **23**

2.1 Bluetooth 23

2.1.1 Arquitetura 24

2.1.1.1 Topologia de rede 25

2.1.2 Máquina de estados 26

2.1.3 Camadas superiores 28

2.2 Bluetooth Low Energy 28

2.2.1 Novas funcionalidades 30

2.2.1.1 Attribute protocol e generic attribute profile 30

2.2.1.2 Generic access profile 31

2.2.1.3 Camada de ligação 32
2.3 Controller area network 33
2.3.1 Arquitetura 34
2.4 Redes de sensores sem fios intra-veículares 35
2.5 Sistema operativo Android 37
2.5.1 Arquitetura 37
2.5.2 Blocos de construção 39
2.5.2.1 Activity 40
2.5.2.2 Serviços 41
2.5.2.3 Fornecedor de conteúdos 41
2.5.2.4 Recetor de difusão 42
2.5.3 Interface do utilizador 42
2.5.4 Android manifest 43
2.6 Trabalho relacionado 44
3. Desenvolvimento do sistema 53
3.1 Introdução 53
3.2 CEPIUM 55
3.3 Hardware BLE 55
3.3.1 Desenvolvimento do firmware 57
3.3.1.1 Estação periférica 58
3.3.1.2 Estação central 61
3.3.1.3 Indicações 62
3.3.1.4 Configurações USART 63
3.3.1.5 Conversão de um valor ADC para um valor 64

3.4 Protocolo de comunicação 67
3.4.1 Sistema da bateria 68
3.4.2 Sistema da tração 70
3.4.3 Mensagens de erro 71
3.4.4 Processamento das tramas nas estações periféricas 72
3.5 Software Android 72
3.5.1 Introdução 73
3.5.2 Lista menu 75
3.5.2.1 Bateria e tração 78
3.5.2.2 Tempo 79
3.5.3 Procura dispositivos 80
3.5.4 Informação dos sensores 82
4. Testes experimentais e resultados 85
4.1 Alcance e perdas em meio aberto 85
4.2 Interferências e obstruções de sinal 88
4.3 Recolha de dados do sistema da bateria 90
4.4 Comunicação com múltiplos módulos sensoriais 92
5. Conclusões e trabalho futuro 103
5.1 Conclusões 103
5.2 Trabalho futuro 105
Referências 106

Lista de figuras

Figura 1.1 - Carro elétrico *plug-in* da Universidade do Minho (CEPIUM) [5]. 21

Figura 2.1 – Combinações de arquiteturas com os blocos *host*, BR/EDR e AMP [6]... 24

Figura 2.2 - Ilustração de duas redes *piconets* e uma *scatternet* (adaptada de [6]). .. 25

Figura 2.3 - Máquina de estados (adaptada de [6]). 26

Figura 2.4 – Combinação dos controladores na arquitetura: a) apenas o controlador LE, b) apenas o controlador BR/EDR e c) combinação do controlador BR/EDR com o controlador secundário AMP [4]. 29

Figura 2.5 - Combinações de arquiteturas com os blocos BR/EDR e LE [4]. 30

Figura 2.6 - Máquina de estados da camada LL [4]. 32

Figura 2.7 - Comparação entre o modelo OSI e CAN [9]. 34

Figura 2.8 - Arquitetura Android. 38

Figura 2.9 - Ciclo de vida de uma *activity* [16]. 41

Figura 2.10 - Hierarquia de Views de uma aplicação [17]. 42

Figura 2.11 – Desempenho do BLE e ZigBee em redes IVWSN [21]. 48

Figura 2.12 – Arquitetura do sistema de monotorização ECG baseado na tecnologia BLE. 49

Figura 2.13 -Arquitetura geral do sistema proposto [25]. 50

Figura 2.14 – Arquitetura geral do sistema do sistema de monitorização [26]. 51

Figura 3.1 - Esquema geral do sistema de monitorização do veículo elétrico. 53

Figura 3.2 - Localização da estação sensorial da bateria. 54

Figura 3.3 - Localização da estação sensorial do motor 54

Figura 3.4 - As estações usadas no sistema de monitorização são compostas pelos seguintes componentes: CC2540EM, SmartRF05EB e uma antena. [19]. 56

Figura 3.5 – Localização das componentes na placa de desenvolvimento. 57

Figura 3.6 – Fluxograma para as estações central e periférica. 59

Figura 3.7 - Fluxograma do processador de evento PeridicSendDataTask(). 60

Figura 3.8 - Fluxograma do processador de evento PeridicReadDataTask(). 61

Figura 3.9 - Fluxograma de processador de evento SimpleBLECentralProcessGATTMsg(). 63

Figura 3.10 – Definições acrescentadas às opções do projeto. 64

Figura 3.11 – Definição ADC acrescentada às opções do projeto. 65

Figura 3.12 – Localização dos portos P18 e P20. 66

Figura 3.13 - Diagrama de sequência da comunicação. 68

Figura 3.14 - Esquema geral do sistema da bateria. 69

Figura 3.15 – Trama do sistema de bateria. 69

Figura 3.16 - Esquema geral do sistema de tração. 70

Figura 3.17 - Trama do sistema de tração. 71

Figura 3.18 - Mensagem de erro. 72

Figura 3.19 - Estrutura geral da aplicação desenvolvida. 74

Figura 3.20 – Fluxograma da *activity* do menu principal. 75

Figura 3.21 – Fluxograma da Navigation drawer. 76

Figura 3.22 – Navigation drawer da aplicação. 77

Figura 3.23 – Fragment responsável por apresentar os dados do sistema da bateria. 78

Figura 3.24 - Fragment responsável por apresentar os dados do sistema de tração. 79

Figura 3.25 – Activity mostra o estado do tempo. 80

Figura 3.26 – Exemplo de listagem de 3 dispositivos BLE encontrados. 81

Figura 3.27 – Fluxograma da *activity* para procurar dispositivos BLE. 82

Figura 3.28 - Fluxograma da *activity* responsável por obter informação das estações periféricas. 83

Figura 4.1 – Local onde se realizaram os testes. 86

Figura 4.2 – Representação gráfica dos testes de alcance. 88

Figura 4.3 – Localização das estações para o teste de interferências. 89

Figura 4.6 – Esquema como ocorreram os testes com dados emulados. 90

Figura 4.7 – Aplicação Android com dados emolados recolhidos. 92

Figura 4.8 - Esquema como ocorreram os testes. 93

Figura 4.9 - Output log dos dados recebidos. 94

Figura 4.10 - Aplicação Android com dados simulados. 94

Lista de tabelas

Tabela 2.1 – Comparação de *standards* existentes (adaptado de [10]). 45

Tabela 3.1 – Correspondência dos portos no código e os portos na placa SmartRF05EB. 66

Tabela 3.2 - Correspondência do campo tipo de trama. 68

Tabela 3.3 - Correspondência do campo endereço do sistema. 70

Tabela 3.4 - Correspondência do campo tipo de trama. 70

Tabela 4.1 – Intervalo de valores RSSI para uma potência de -23 dBm. 86

Tabela 4.2 – Intervalo de valores RSSI para uma potência de -6 dBm. 87

Tabela 4.3 – Intervalo de valores RSSI para uma potência de 0 dBm. 87

Tabela 4.4 – Intervalo de valores RSSI para uma potência de 4 dBm. 87

Tabela 4.9 – Especificação dos sensores do sistema da bateria. 91

Tabela 4.10 - Especificação dos sensores do sistema de tração. 93

.

Lista de abreviaturas

8DPSK	8-ary Differential Phase Shift Keying
ABS	Antilock Braking System
ACK	Acknowledgement
AMP	Alternate MAC/PHY
API	Application Programming Interface
ART	Android Runtime
ATT	Attribute Protocol
BD/ EDR	Basic Rate/Enhanced Data Rate
BLE	Bluetooth Low Energy
BR/EDR	Basic Rate/Enhanced Data Rate
CAN	Controller Area Network
CCC	Client Characteristic Configuration
CEPIUM	Carro Elétrico Plug-In da Universidade do Minho
dBm	Decibel Miliwatt
DLL	Data Link Layer
DQPSK	Differential Quadrature Phase Shift Keying
ECU	Electronic Control Unit
FDMA	Frequency Division Multiple Access
FHSS	Frequency-Hopping Spread Spectrum
FLC	Fuzzy Logic Controller
FM	Frequency Modulation

FSK	Frequency-Shift Keying
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GEPE	Grupo de Electrónica de Potência e Energia
GUI	Graphical User Interface
HCI	Host Controller Interface
HMI	Human-Machine Interface
IC	Intervalo de Conexão
IrDA	Infrared Data Association
ISM	Industrial, Scientific and Medical
ISO	International Standard Organization
IVWSN	Intra-Vehicular Wireless Sensor Network
L2CAP	Link Layer Control and Adaptation Protocol
LE	Low Energy
LL	Link Layer
LLC	Logic Link Control
LOS	Line of Sight
LQI	Link Quality Indicator
MAC	Media Access Control
OHA	Open Handset Alliance
OS	Operating System
OSI	Open Systems Interconnection
PEP	Período do Evento Periódico
PHY	Physical Layer

PLR	Packet Loss Rate
PRR	Packet Reception Ratio
PSK	Phase-shift keying
RSSI	Received Signal Strength Indicator
SDP	Service Discovery Protocol
SIG	Special Interest Group
SMP	Security Manager Protocol
TDD	Time division duplex
TDMA	Time Division Multiple Access
TI	Texas Instruments
UART	Universal Asynchronous Receiver/Transmitter
ULP	Ultra-Low Power
UUID	Universally Unique Identifier
UWB	Ultra-wideband
WASN	Wide Area Sensor Network

1. Introdução

1.1 Enquadramento e motivação

A produção de veículos modernos está altamente informatizada. Os carros hoje em dia integram geralmente uma rede cablada de controladores conhecida como Controller Area Network (CAN), que estabelece a comunicação entre o computador de bordo de um carro e pequenas unidades de controlo eletrónicas (ECU – Electronic Control Unit). Estas unidades possuem sensores para detetar variações em parâmetros como temperatura, pressão e tensão. Com a crescente complexidade dos veículos, o número de dispositivos aumenta, assim como a quantidade de cabos requerida para possibilitar as conexões, criando assim problemas significativos: aumento do peso e do custo causados pelos cabos e limitação da localização dos sensores [1].

A eliminação dos cabos possibilita a diminuição do peso dos carros, o que proporciona uma melhoria no desempenho e na economia de combustível, para além da redução de custos com a instalação de cabos para os fabricantes de automóveis. A fim de encontrar soluções para estas questões, o uso de tecnologias sem fios foi recentemente proposto para a comunicação entre as ECUs. No entanto, as peças em metal podem criar obstáculos à propagação dos sinais e um ambiente desafiador e único para a comunicação sem fios, comparado com ambientes em espaços abertos como aqueles que existem na maioria das comunicações sem fios [1].

Existem várias tecnologias de redes de comunicação sem fios no mercado: ZigBee, Wi-Fi, Bluetooth Low Energy (BLE), entre outras. O BLE foi desenvolvido pelo grupo Bluetooth Special Interest Group (SIG) como uma solução de baixo consumo de energia para aplicações de controlo e monitorização, sendo nesta tecnologia emergente que este trabalho se concentra.

O BLE apareceu como resposta a uma necessidade de aumento da autonomia de dispositivos sem fios alimentados através de baterias. Atualmente, o seu uso centra-se em dispositivos de cuidados de saúde, desporto, teclados e ratos, entre muitos outros, pois permite o uso prolongado dos dispositivos sem a necessidade de recarga regular. Para além do baixo consumo energético, apresenta baixos custos de implementação, compatibilidade de comunicação entre os vários fornecedores [2] e suporte nativo em grande parte dos *tablets* e *smartphones*.

O BLE está integrado na norma Bluetooth 4.0, que é a versão mais recente do Bluetooth e tem como principal característica a tecnologia Ultra-Low Power (ULP), de baixo consumo. O BLE opera na banda de frequências Industrial, Scientific and Medical (ISM) de 2,4 GHz. Possui um intervalo de conexão variável, que pode ser ajustado entre 7,5 ms e vários segundos (dependendo da aplicação desejada), um tempo de espera elevado e uma rápida conexão entre os dispositivos. Estas são características que permitem que esta tecnologia seja uma boa solução para comunicações a curtas distâncias [3], [4].

Nesta dissertação será desenvolvida uma aplicação gráfica para Android e realizada a integração de um protocolo de comunicações sem fios no sistema para comunicação com as ECUs. A aplicação gráfica será apresentada num *smartphone* que irá desempenhar o papel de display interface homem-máquina (HMI - Humana Machine Interface) do carro elétrico *plug-in* da Universidade do Minho (CEPIUM) (Figura 1.1) desenvolvido pelo Grupo de Eletrónica de Potência e Energia (GEPE). O protocolo de comunicações sem fios será implementado utilizando dispositivos BLE e terá em conta duas implementações: uma tendo como base módulos CC2540 da Texas Instruments, e a outra integrando estes módulos com o *smartphone*.



Figura 1.1 - Carro elétrico *plug-in* da Universidade do Minho (CEPIUM) [5].

1.2 Objetivos

Esta dissertação tem como finalidade desenvolver um sistema de comunicação entre um carro elétrico e o seu utilizador através do uso de sensores ligados a módulos BLE e um dispositivo móvel. A recolha periódica de informação permite ao utilizador estar atualizado sobre o estado do veículo. Neste dispositivo móvel encontrar-se-á uma aplicação Android que permite o acesso aos dados recolhidos pelos sensores. Mais especificamente pretende-se atingir os seguintes objetivos:

1. Implementar e testar o desempenho da rede de comunicações sem fios BLE em laboratório e no veículo.
2. Desenvolver e realizar testes do código para a recolha dos dados de sensores, obtidos através dos módulos BLE instalados no veículo.
3. Desenvolver uma interface gráfica para um dispositivo móvel Android com a finalidade de apresentação dos dados.
4. Integrar as partes desenvolvidas e testar o sistema no carro elétrico CEPIUM.

1.3 Estrutura da dissertação

Esta dissertação está organizada em cinco capítulos.

No capítulo 1 é realizada uma introdução ao tema da dissertação. Neste capítulo é descrito o enquadramento do trabalho assim como os seus objetivos. Por último, este capítulo é finalizado com a apresentação da estrutura desta dissertação.

No capítulo 2 (Estado da Arte) é apresentada uma descrição geral de conceitos teóricos e tecnologias que foram importantes na realização deste trabalho. No final deste capítulo realiza-se o ponto da situação sobre trabalhos de investigação que, direta ou indiretamente, contribuíram para a compreensão deste trabalho.

No capítulo 3 (Desenvolvimento do Sistema) é efetuada uma descrição da implementação da aplicação gráfica para Android e do protocolo de comunicações com as ECUs. É apresentada a plataforma de *hardware* e *software* que serviu de base para a realização de todo o trabalho, bem como certas ferramentas de apoio. A aquisição dos dados sensoriais e o formato das tramas utilizado são também descritos neste capítulo.

No capítulo 4 (Testes e Resultados) são apresentados os principais resultados obtidos. Alcance, uso de notificações e indicações, potência de transmissão, taxa de perda de pacotes e interferências são parâmetros avaliados, teórica e experimentalmente, neste capítulo. Os resultados dos testes efetuados são devidamente acompanhados das respetivas discussões.

Por fim, o capítulo 5 (Conclusão e Trabalho Futuro) contém as conclusões da realização deste trabalho de investigação assim como a discussão sobre trabalho futuro a desenvolver.

2. Estado da Arte

O presente capítulo aborda o estado atual dos temas principais que servem de base para esta dissertação. Sendo assim, é feito um resumo das noções necessárias para o desenvolvimento de dois módulos fundamentais: o *firmware* dos módulos BLE e aplicação Android. Inicialmente são abordadas as redes Bluetooth e BLE. Seguidamente, são expostos conceitos básicos das redes CAN e aspetos associados à implementação da tecnologia BLE na indústria automóvel. Posteriormente, são tratados alguns conceitos relativos ao sistema operativo Android que foram necessários para o desenvolvimento da aplicação. Por fim, é apresentado algum trabalho relacionado com a área de foco da dissertação, mais propriamente trabalhos de índole aplicacional e trabalhos mais direcionados para o desempenho do BLE e outras tecnologias idênticas.

2.1 Bluetooth

O Bluetooth é uma tecnologia usada para comunicações de rádio de curtas distâncias, com a finalidade de substituir as conexões por fios entre diversos dispositivos, como telefones, computadores e assistentes digitais pessoais. A entidade responsável pela sua especificação e promoção é denominada Bluetooth SIG (Special Interest Group).

As características principais desta tecnologia são a robustez, baixo consumo de energia, baixa complexidade e baixo custo [4]. Os seus objetivos são semelhantes para o protocolo IrDA (Infrared Data Association), no entanto o Bluetooth comunica na gama de frequências ISM de 2,4 GHz e utiliza espalhamento espectral por saltos em frequência (FHSS – Frequency Hopping Spread Spectrum) por 79 canais com espaçamento de 1 MHz, o que permite a comunicação em áreas de grande interferência eletromagnética. O Bluetooth permite a comunicação sem fios até 10

metros ou mais, no caso de a potência de transmissão ser aumentada, sem necessidade de linha de vista (LoS – Line of Sight). Esta tecnologia permite realizar comunicações ponto-a-ponto ou ponto-multiponto [6]. Existem quatro versões do Bluetooth, no entanto nesta secção são abordados os aspetos gerais desta tecnologia. Na secção 2.2 será abordada a ultima versão disponível mais detalhadamente.

2.1.1 Arquitetura

A arquitetura de um sistema Bluetooth consiste num *host* e num ou mais controladores. Um *host* é composto por todas as camadas que se encontram acima da interface HCI (Host Controller Interface). Um controlador, por sua vez, é constituído por todas as camadas abaixo da interface HCI. Segundo a versão 3.0 do Bluetooth [6] existem dois tipos de controladores: Basic Rate/Enhanced Data Rate (BR/EDR) e Alternate MAC/PHY (AMP). Este último controlador é uma nova aquisição para permitir a transmissão de grandes quantidades de tráfego. A **Figura 2.1** mostra algumas arquiteturas do Bluetooth versão 3.0.

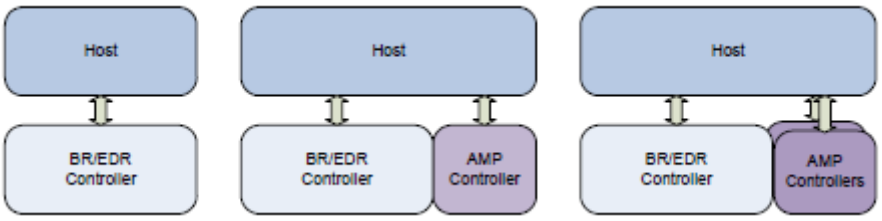


Figura 2.1 – Combinações de arquiteturas com os blocos *host*, BR/EDR e AMP [6]**.**

O controlador BR/EDR usa dois tipos de modulação: FSK (Frequency-Shift Keying), cuja implementação é obrigatória, e modulação por deslocamento de fase (PSK - Phase-Shift Keying) de implementação opcional, com duas variantes, $\frac{\pi}{4}$ DQPSK (Differential Quadrature Phase Shift Keying) e 8DPSK (8-ary Differential Phase Shift Keying). A taxa de transmissão por ar é de 1 Mbps para o primeiro tipo de modulação e 2 Mbps / 3 Mbps para a segunda modulação, usando as diferentes variantes, respetivamente. Para haver uma transmissão bidirecional é usado um esquema de divisão no tempo (TDD - Time Division Duplex) nos controladores BR/EDR.

2.1.1.1 Topologia de rede

Os dispositivos Bluetooth são geralmente organizados em redes de dispositivos interligados entre si, designadas de *piconets*. Quando dois ou mais dispositivos estabelecem uma comunicação através de um controlador BR/EDR, partilhando o mesmo canal físico, formam uma *piconet*. Nestas redes, a comunicação é sempre feita entre mestre e escravo, através de um esquema de *polling*, e nunca diretamente entre dois escravos. Numa *piconet,* podem participar simultaneamente um mestre e até sete escravos no modo ativo. A função mestre/escravo só é atribuída quando a *piconet* é estabelecida e o papel de mestre vai para unidade que formou a *piconet*. Um conjunto de várias *piconets* interligadas na mesma área é chamada de *scatternet* (Figura 2.2). Uma vez que a sincronização é definida através do relógio do dispositivo mestre e cada *piconet* exige um canal físico de comunicação diferente, os tempos e as sequências de saltos serão distintos, tornando impossível um mestre pertencer a duas ou mais *piconets* [6].

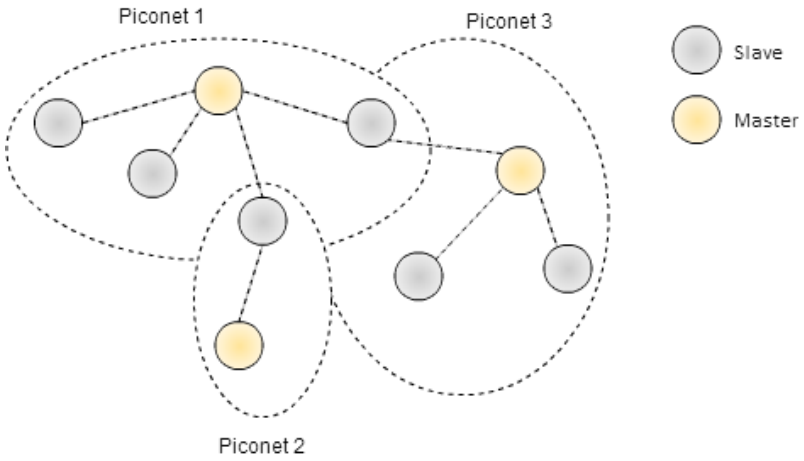


Figura 2.2 - Ilustração de duas redes *piconets* e uma *scatternet* (adaptada de [6]**).**

O canal físico é caracterizado por um salto pseudoaleatório entre os 79 canais de radiofrequência. Para determinar a sequência de salto e a fase a utilizar no mecanismo FHSS é necessário obter informação sobre o endereço e o relógio da estação mestre. Dado que os relógios são independentes estas informações devem ser atualizadas regularmente [6].

2.1.2 Máquina de estados

No Bluetooth foram definidos 3 elementos que suportam o estabelecimento de conexão: procura (*scan*), paginação (*page*) e consulta (*inquiry*). A máquina de estados da Figura 2.3 serve para mostrar como os dispositivos Bluetooth podem ser adicionados ou libertados de uma *piconet* e demonstra os três estados principais:

1. Espera (*Standby*);
2. Conectado (*Connected*);
3. Estacionado (*Parked*).

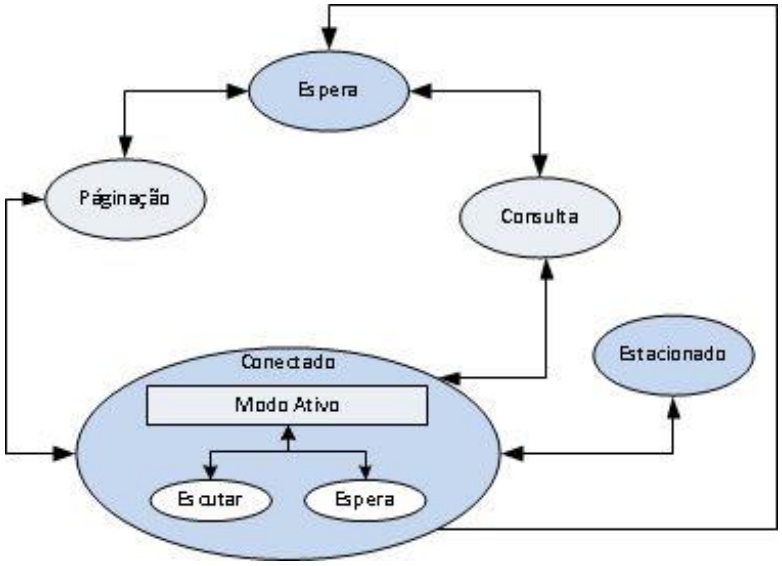


Figura 2.3 - Máquina de estados (adaptada de [6]**).**

O estado padrão de um dispositivo é o modo de espera. Neste modo, para conservar energia, o dispositivo fica inativo grande parte do seu tempo. Periodicamente, o dispositivo ativa-se para escutar os canais a verificar se existe alguma comunicação com o código de acesso que derivou da sua própria identidade. Sempre que se ativa procura num canal de frequência diferente.

A fim de estabelecer uma nova comunicação é usado o procedimento de paginação. Este procedimento requer o conhecimento da identidade do dispositivo à qual se pretende conectar. De seguida, toma conhecimento da sequência de frequência para realizar uma procura nos tempos ativos e poder gerar um código de

acesso que serve como mensagem de paginação. O dispositivo que realiza o procedimento de paginação torna-se automaticamente o mestre da conexão.

O procedimento de consulta é utilizado quando um dispositivo procura outro e envia pedidos de consulta. Os dispositivos que estão disponíveis para ser encontrados são conhecidos como dispositivos "identificáveis" (*discoverable*), e assim que ouvem pedidos de solicitação enviam respostas.

No estado de conexão, a ligação já foi estabelecida e os dispositivos estão conectados uns aos outros dentro de uma *piconet*, podendo haver troca de pacotes de forma bidirecional. Neste estado, se um dispositivo não permanecer constantemente no canal, pode ativar os modos "escuta" (*sniff*) ou "espera" (*hold*) (Figura 2.3) para conservar energia.

Quando um dispositivo se encontra no modo de escuta, a disponibilidade dos serviços é modificada através da definição de ciclos que consistem em períodos de presença e de ausência.

No modo de espera, o dispositivo repousa por um período de tempo definido e retorna para o modo ativo quando esse tempo expira. É um processo normalmente adquirido por dispositivos que não têm necessidade de enviar informação por um intervalo de tempo relativamente longo.

Por último, quando um escravo não precisa de participar numa *piconet* mas ainda necessita de permanecer sincronizado com o canal de comunicação, começa o estado de "estacionamento" (*parked*). Este estado é o mais complexo dos modos de suspensão, pois permite que haja uma troca de escravos ativos com escravos estacionados na mesma *piconet,* permitindo que o número de escravos numa topologia seja grande. Este processo consiste na remoção do endereço de membro ativo e reatribuição a outro dispositivo que foi retirado do estado estacionado pelo mestre.

2.1.3 Camadas superiores

Esta secção define as camadas Link Layer Control and Adaptation Protocol (L2CAP), Generic Access Profile (GAP) e Service Discovery Protocol (SDP). A camada L2CAP fica em cima da camada composta pelos controladores BR/EDR e faz a interface com os protocolos superiores. Deve operar em mais do que um controlador de cada vez e, durante as configurações de canal, a capacidade de protocolo de multiplexação é usada para encaminhar conexões para as camadas superiores corretas. Contudo, na transferência de dados, a multiplexação é necessária no canal lógico para distinguir entre as várias entidades, pois pode haver mais do que uma entidade com o mesmo protocolo.

A camada GAP tem como objetivo introduzir definições, recomendações e requisitos comuns relacionados com os modos e procedimentos de acesso que estão a ser utilizados pelos perfis de transporte e aplicações. Descreve como um dispositivo se comporta nos estados de espera e conectado para garantir que as ligações e os canais possam ser sempre estabelecidos entre os dispositivos Bluetooth.

Finalmente a camada SDP fornece meios de descoberta para as aplicações encontrarem os serviços disponíveis e determinar as características desses serviços. A comunicação SDP envolve um cliente e um servidor. O servidor mantem uma lista de registos de serviços que descreve as características de serviços associados e o cliente pode aceder a essas informações enviando um pedido. Haverá no máximo um servidor SDP ativo por dispositivo Bluetooth, mas é possível que um único dispositivo Bluetooth funcione tanto como um servidor como um cliente SDP.

2.2 Bluetooth Low Energy

A versão do Bluetooth utilizada no projeto é o Bluetooth 4.0. Esta versão inclui duas variantes da tecnologia: o Bluetooth Clássico (BD/EDR) e o Bluetooth Low Energy (BLE). Este último modo foi o escolhido para o desenvolvimento do trabalho.

O sistema de baixa energia (LE – Low Energy) inclui recursos projetados para permitir aplicações de baixo consumo, como aplicações de controlo e monotorização, pois estas requerem menores custos e taxa de transmissão mais baixa do que os sistemas que usam BD/EDR e os dispositivos são menos complexos [4]. Como foi visto anteriormente, o núcleo do sistema Bluetooth consiste num *host* e num ou mais controladores. O controlador inclui a camada física e a camada de ligação. O *host* contém as funcionalidades da camada superior, ou seja, o L2CAP, o Attribute Protocol (ATT), o Generic Attribute Profile (GATT), o GAP e o Security Manager Protocol (SMP). A comunicação entre o *host* e o controlador é normalizada como Host Controller Interface (HCI).

Nesta versão existem dois tipos de controladores: os controladores primários e os secundários. Uma implementação do núcleo Bluetooth tem apenas um controlador primário, que pode ser uma das seguintes configurações: somente LE, somente BR/EDR, ou um ou mais controladores secundários (um controlador AMP) em junção com o controlador primário BR/EDR (Figura 2.4).

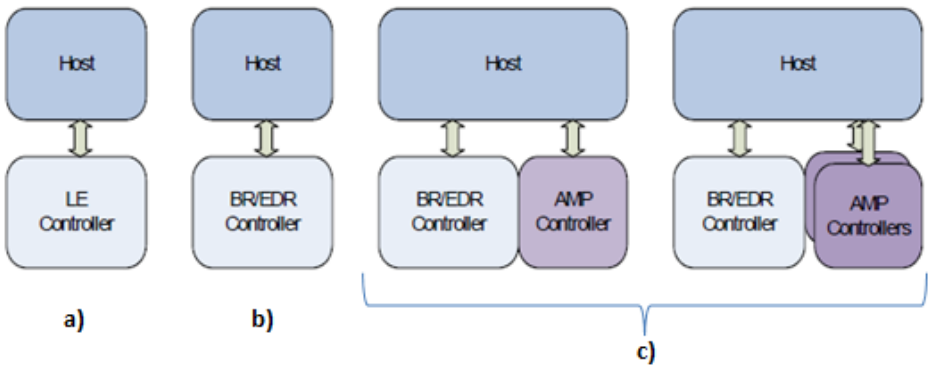


Figura 2.4 – Combinação dos controladores na arquitetura: a) apenas o controlador LE, b) apenas o controlador BR/EDR e c) combinação do controlador BR/EDR com o controlador secundário AMP [4].

A outra implementação do núcleo do sistema consiste numa combinação dos dois controladores primários ou dos dois controladores primários com um ou mais controladores secundários (Figura 2.5).

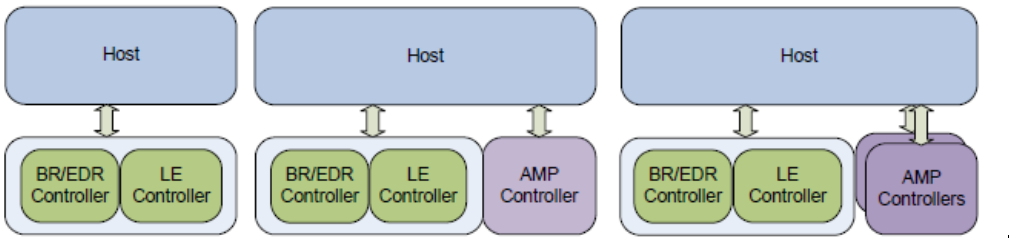


Figura 2.5 - Combinações de arquiteturas com os blocos BR/EDR e LE [4].

O controlador LE emprega dois esquemas de acesso múltiplo: por divisão de frequência (FDMA - Frequency Division Multiple Access) e por divisão de tempo (TDMA - Time Division Multiple Access). No esquema FDMA são usados 40 canais físicos, separados por 2 MHz, sendo que três são usados para canais de *advertising* (anúncio) e 37 são usados para canais de dados [4], [7].

Dispositivos de uma *piconet* podem usar um padrão de salto em frequência específico, que é um algoritmo determinado por um campo contido no pedido de conexão enviado por um dispositivo. Um mecanismo de salto em frequência adaptativo é usado em cima dos canais de dados, a fim de enfrentar problemas de interferência. Esta técnica de salto adaptativo melhora a coexistência de dispositivos Bluetooth com sistemas estáticos (sem saltos em frequência) na mesma banda ISM.

Relativamente às mudanças em relação à versão anterior, pode-se notar uma introdução de novas funcionalidades (camada GATT, ATT e LL), melhoramentos e simplificações tendo em vista o baixo consumo de energia (camada HCI, L2CAP e GAP) [4], [7].

2.2.1 Novas funcionalidades

2.2.1.1 Attribute protocol e generic attribute profile

A camada ATT (Attribute Protocol) implementa um protocolo ponto-a-ponto entre um servidor ATT e um cliente ATT. O servidor mantém um conjunto de atributos que armazena a informação numa estrutura de dados controlada pelo GATT (Generic

Attribute Profile). O papel de servidor ou cliente ATT é independente do papel mestre ou escravo.

O cliente envia comandos ATT, solicitações e confirmações para o servidor. Estes comandos e pedidos fornecem um meio para ler e escrever valores de atributos no servidor. O servidor pode enviar para um cliente dois tipos de resposta que contêm atributos: notificações, que são não confirmadas; e indicações, que exigem uma confirmação.

A camada Generic Attribute Profile (GATT) descreve a hierarquia de serviços, características e atributos usados no servidor ATT. Este bloco fornece interfaces para descobrir, ler, escrever e indicar características de serviços e atributos. As características são um conjunto de dados que inclui um valor e propriedades. Os dados relacionados com os serviços e características são armazenados em atributos. A GATT é usada por dispositivos de baixa energia para a descoberta de serviços de perfis LE.

2.2.1.2 Generic access profile

O GAP é uma interface direta com aplicações, sendo responsável pela especificação dos papéis que um dispositivo adota, modos de descoberta de dispositivos e serviços e gestão do estabelecimento de uma conexão. Existem quatro papéis: *broadcaster*, observador, periférico ou central. Apenas os dois últimos são explicados, uma vez que foram os papéis usados no desenvolvimento desta dissertação.

O papel de um dispositivo central está projetado para um dispositivo que fica encarregado de iniciar e gerir múltiplas conexões, enquanto o papel de um dispositivo periférico está desenhado para um dispositivo simples que só usa uma única conexão com o dispositivo central. Um dispositivo central opera como mestre e um dispositivo periférico como escravo.

2.2.1.3 Camada de ligação

Uma outra diferença que se pode verificar entre as versões anteriores do Bluetooth e a esta nova versão é a máquina de estados desta camada. Como se pode verificar na Figura 2.6, a máquina de estados de um dispositivo BLE é muito simples.

A cama da ligação (LL – Link Layer) opera com base em cinco estados: espera, anúncio, descoberta (*scanning*), iniciação (*initiating*) e conexão (*connection*).

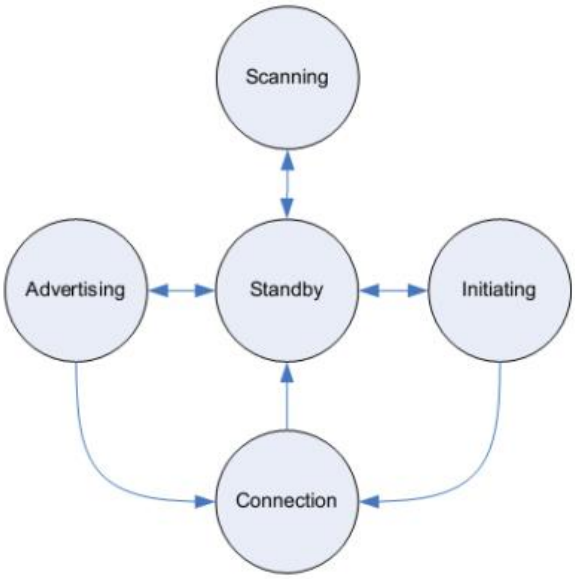


Figura 2.6 - Máquina de estados da camada LL [4].

A camada LL no estado de espera não transmite nem recebe quaisquer pacotes. No estado de anúncio, transmite pacotes nos canais de anúncio e pode ouvir e responder às respostas desencadeadas por estes pacotes. Estes pacotes nos canais de anúncios são escutados pelos dispositivos no estado de descoberta. Os dispositivos no estado de iniciação respondem a pacotes de dispositivo(s) específico(s) para iniciar uma conexão com outro dispositivo. No estado de conexão os dispositivos estão conectados. Existem duas funções neste estado: a função de mestre e a função de escravo. Quando o seu estado procedente é a inicialização, a função do dispositivo é de mestre. Quando acedido através do estado de anúncio, o papel desempenhado pelo dispositivo é o de escravo.

2.3 Controller area network

O CAN (Controller Area Network) é um protocolo de comunicação baseado em *multicast*, desenvolvido nos meados de 1980 pela Bosch para a indústria automóvel. O objetivo das redes CAN consistia na simplificação dos complexos sistemas de fios nos veículos controlados por múltiplos microcontroladores para a gestão do motor, sistema de ABS e controlo de suspensão [8].

O CAN é um barramento digital projetado para operar nas velocidades de 20 kbps a 1 Mbps e aprovada pela International Standard Organization (ISO). A taxa de transmissão depende da velocidade e comprimento do barramento recetor [8].

Nas redes CAN as mensagens transmitidas possuem um determinado identificador e não um endereço de destinatário como nas redes convencionais. Deste modo, um emissor envia uma mensagem para todos os nós CAN e cada um decide se deve ou não processar a mensagem. Para além disto, este identificador também define a prioridade da mensagem, que permite competir com outras mensagens pelo acesso ao barramento. Uma outra característica destas redes é a sua flexibilidade, uma vez que podem ser adicionados novos nós a uma rede sem fazer alterações de *software* ou *hardware* dos restantes nós. Cada estação CAN regista os erros e avalia, tornando este protocolo eficaz para ambientes com ruídos.

Esta é uma solução atraente para sistemas embebidos de controlo por causa do baixo custo, gestão de controlo leve e as suas características internas para a deteção de erros e retransmissão. Em resumo, as suas diversas características são:

- Possibilidade de atribuir prioridade às mensagens;
- Capacidade *multicast*;
- Reduzido tempo de latência;
- Partilha de dados entre as várias unidades que podem eliminar redundância de informação;
- A deteção de erros e sinalização com retransmissão automática de mensagens corrompidas;
- Menos fios no sistema, tornando a cablagem mais simples e mais barata;

- Sistema flexível, as unidades podem ser adicionadas ou removidas de forma mais simples;
- Distinção entre erros temporários e erros permanentes dos nós;
- Considerável imunidade ao ruido.

2.3.1 Arquitetura

A especificação da rede CAN, face ao modelo de sete camadas OSI, descreve apenas a camada física, a camada de ligação de dados (*hardware*) e aplicação (*software*). A Figura 2.7 representa essa comparação.

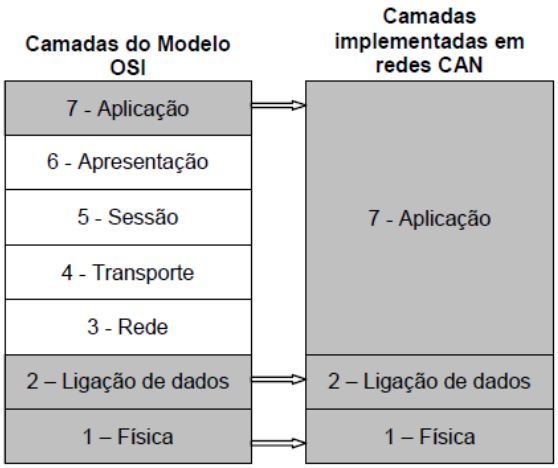


Figura 2.7 - Comparação entre o modelo OSI e CAN [9]**.**

O objetivo da camada física é definir a codificação dos bits em sinais elétricos ou eletromagnéticos que serão transmitidos através de ligações com fios a partir de um nó para outro. Na norma da Bosch, na descrição desta camada, apenas é definida a temporização dos bits, a codificação dos bits e a sincronização, deixando de fora a especificação do meio de transmissão físico, os níveis de sinal e outras características necessárias. De modo a preencher estas lacunas e oferecer soluções existem outros documentos e implementações [8].

A camada de ligação de dados (DLL – Data Link Layer) estabelece o formato das mensagens que percorrem o barramento e fornece mecanismos de deteção e prevenção de falhas. Esta camada consiste em duas subcamadas: controlo de link lógico (LLC – Logic Link Control) e controle de acesso ao meio (MAC – Media Access

Control). A subcamada LLC fornece todos os serviços para transmissão de fluxo de bits de uma fonte para um destino. Define mais precisamente:

- Filtro de aceitação de mensagens;
- Notificação de *overload*;
- Gestão de recuperação de mensagens ou de situação de erro.

A subcamada MAC é o núcleo da especificação do protocolo CAN, sendo responsável pela:

- Gestão de acesso ao meio;
- Encapsulamento / desencapsulamento das mensagens;
- Deteção e sinalização de erros;
- Codificação das mensagens;
- Confirmação da integridade das mensagens;
- Serialização/desserialização da informação.

2.4 Redes de sensores sem fios intra-veículares

Os veículos modernos são altamente informatizados e as principais funcionalidades são controladas por várias unidades ECU (Electronic Control Unit) no seu interior. Estas unidades recolhem informação sobre o estado do veículo através dos sensores para manter todas as operações necessárias do veículo. Atualmente, a maior parte dos sensores estão conectados por cabos que transmitem os dados por fios até ao seu destino. As ECUs foram introduzidas nos sistemas automóveis para controlar as regras de limite de emissões de gases e, desde então, têm evoluído e difundiram-se em várias áreas.

Devido ao aumento da complexidade dos veículos e do número cada vez maior de aplicações e dispositivos que estes abarcam, aumenta também, o número de cabos necessários para manter todas as conexões com os sensores, o que traz vários desafios, como o peso adicional, a localização limitada dos sensores e a substituição de cabos caso necessário [1]. Se o peso extra dos cabos pode ser eliminado, o peso do

veículo em si será reduzido resultando num menor consumo de combustível, melhor desempenho e menos custos de montagem [10]. Os fios limitam a localização dos sensores e, consequentemente, a gama de aplicações possíveis. A fim de abordar estas questões, a tecnologia sem fios foi recentemente proposta para as comunicações entre os sensores e ECUs [1]. Os sensores sem fios e as ECUs formam uma nova arquitetura, que é muitas vezes referida como uma rede de sensores sem fios intra-veícular (IVWSN - Intra-Vehicular Wireless Sensor Network).

A mudança de um sistema com fios para um sistema sem fios deve adotar um esquema gradual: pode-se começar por alterar os sensores que não são de segurança crítica, ou os únicos sensores com posições remotas de difícil alcance, ou os sensores mais acessíveis para serem substituídos. A principal barreira para efetuar esta substituição, para os fabricantes de automóveis, é o custo adicional do *hardware* sem fios. A escolha do tipo de sensor tem os seguintes requisitos e propriedades [1]:

- Baixo custo: menor complexidade, menor custo;
- Baixo consumo de energia: uma vez que os sistemas sem fios são alimentados por uma bateria, o seu consumo deve ser o mínimo possível para a bateria ter um tempo de vida útil razoável;
- Tempo de resposta curto e baixa taxa de transferência de dados;
- Alta confiabilidade;
- Transmissão de mensagens com várias prioridades.

A maioria das IVWSN são fixas ou com um movimento muito reduzido dentro de uma área pequena, mesmo que as redes de sensores sem fios clássicas tenham uma topologia dinâmica. Isto implica que a configuração de rotas é mais simples nas redes IVWSN. No entanto, as peças de metal, especialmente o compartimento do motor, criam um ambiente desafiador para as comunicações sem fios.

Neste trabalho a tecnologia BLE foi proposta como opção para uma arquitetura IVWSN, como pode ser observado no capítulo 3.

2.5 Sistema operativo Android

O Android é um sistema operativo livre e *open-source* baseado em Linux para dispositivos móveis de ecrã tátil como os *smartphones* e *tablets* [11]. Foi desenvolvido em 2003 por uma *startup* liderada por Andy Rubin, Rich Miner e Nick Sears. Atualmente o Android pertence à empresa norte-americana Google e é desenvolvido pela Open Handset Allience (OHA), liderada pela mesma. A OHA é uma aliança de diversas companhias internacionais com intenção de fortalecer e criar padrões abertos para o Android. O Google descreve o Android como [12]:

"*The first truly open and comprehensive platform for mobile devices, all of the software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation.*"

Existem centenas de milhões de dispositivos Android em mais de 190 países [11], [13]. Esta popularidade, que se deve ao facto do Android ser *open-source*, motivou o constante crescimento de comunidades de programadores e de consumidores. Para desenvolver o trabalho desta dissertação foi preciso adquirir conhecimento de algumas das suas funcionalidades, como: arquitetura, serviços, fornecedor de conteúdos e recetor de difusão.

2.5.1 Arquitetura

A arquitetura Android, também conhecida como pilha de *software* Android, é mostrada na Figura 2.8, sendo composta por quatro camadas e cinco componentes principais.

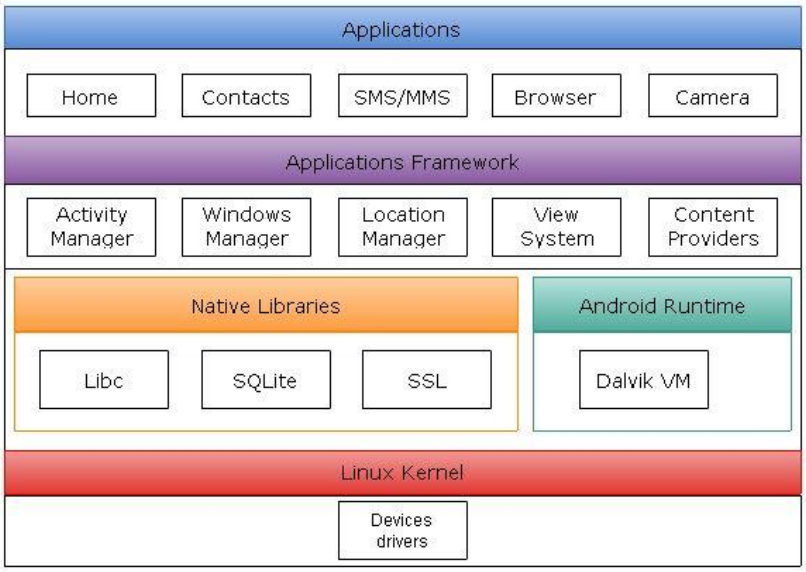


Figura 2.8 - Arquitetura Android.

Na parte inferior da hierarquia da arquitetura encontra-se a camada Linux Kernel. O *kernel* em uso é a versão 2.6 [14], modificado para as necessidades especiais como na gestão de energia, memória e processos. O *kernel* também funciona como uma camada de separação entre o *hardware* e o resto da pilha de *software*.

Acima da camada Linux Kernel encontra-se uma camada dividida em duas componentes. A primeira são as Bibliotecas Nativas (Native Libraries) como SQLite, biblioteca C (libc) e SSL. A segunda componente é o Android Runtime (ART), que inclui um conjunto de bibliotecas centrais que fornecem a maioria das funcionalidades disponíveis na biblioteca principal da linguagem de programação Java, bem como uma máquina virtual própria, denominada Dalvik. Cada aplicação é executada com o seu próprio processo e com a sua própria instância nesta máquina virtual. A Dalvik é semelhante à máquina virtual Java, mas otimizada para consumir o mínimo de memória, uma vez que o Android é executado em dispositivos com pouca memória e pouco poder de processamento [14], [15].

No topo das Bibliotecas Nativas e do ART encontra-se a camada Framework das Aplicações (Application Framework). Grandes quantidades de classes e interfaces para desenvolvimento Android são fornecidas por esta camada. O seu objetivo é fornecer

uma maneira fácil e consistente de gerir interfaces gráficas, recursos de acesso e conteúdo ou para lidar com as chamadas recebidas [15]. Os serviços essenciais são:

- Activity Manager – controla o ciclo de vida de uma aplicação;
- Content Providers – permite que as aplicações publiquem e partilhem os seus dados com outras aplicações;
- Resource Manager – fornece acesso a não código incorporado como *strings*, cores e *layouts*;
- Notification Manager – permite alertas e notificar, da aplicação para o utilizador;
- View System – usado para criar interfaces de utilizador.

Todas as aplicações Android, como contactos, *browser*, programas de SMS e outros encontram-se na camada mais alta, a camada de aplicação. As aplicações desenvolvidas são escritas na linguagem de programação Java e o Android permite que as aplicações possam ser executadas em simultâneo [14]. Existem quatro tipos de componentes essenciais para uma aplicação: Atividades (Activities), Serviços (Services), Fornecedores de Conteúdos (Content Providers) e Recetores de Difusão (Broadcast Receivers).

Como pertencem à camada mais alta da arquitetura Android, podemos concluir que as aplicações usam a camada Framework das Aplicações que utiliza o ART e as suas bibliotecas, que por sua vez utilizam o *kernel* do Linux. É nesta camada que se enquadra a aplicação desenvolvida para esta dissertação.

2.5.2 Blocos de construção

Cada componente de uma aplicação é um ponto de acesso diferente para o sistema. Nem todas as aplicações precisam de ter as quatro componentes, mas sim alguma combinação destas. As componentes, referidas na secção anterior, ajudam a definir o comportamento global da aplicação e, uma vez que já foram determinadas as componentes necessárias para aplicação, são registadas num ficheiro com o nome

de AndroidManifest.xml. É também neste ficheiro que se encontram as permissões necessárias para o funcionamento correto da aplicação desenvolvida.

2.5.2.1 Activity

Uma *activity* representa um único ecrã com uma interface que permite interação com o utilizador, ou seja, permite a construção de interfaces gráficas do utilizador (GUI – Graphical User Interface) [16]. No arranque de uma aplicação, é sempre iniciada pelo menos uma *activity*. Esta funciona como um ponto de referência na aplicação podendo, por sua vez, iniciar outras *activities*. Uma aplicação pode ter várias *activities* e a sua transação entre elas é iniciada através de *intents*. Um *intent* é uma estrutura de dados passiva que possui uma descrição abstrata de uma operação a ser realizada [16]. Através do método startActivity(Intent) é possível despoletar uma nova *activity*. O método startActivityForResult(Intent) permite passar e obter dados no final de execução da *activity*. Devido ao lugar de destaque que as atividades ocupam, o sistema contém um conjunto de mecanismos que ajudam o utilizador entre as várias *activities*. Um destes mecanismos é parar temporariamente e preservar a *activity* anterior numa *stack*. Este mecanismo é conhecido por Task Backstack [16]. Assim, o seu ciclo de vida fica a cargo do sistema operativo, que gere automaticamente as diversas fases das atividades, garantido que estas são devidamente inicializadas, suspendidas, retomadas ou destruídas. É permitido ao programador alterar a aplicação conforme a fase da *activity*. Existem sete métodos que permitem alterar o comportamento de cada fase:

- onCreate () – executado quando a *activity* é criada pela primeira vez;
- onStart () – executado quando a *activity* se torna visível para o utilizador;
- onResume () – executado quando o utilizador começa a interagir com a aplicação;
- onPause () – executado durante a transição da GUI para outra *activity*;
- onStop () – executado quando a *activity* não se encontra mais visível;
- onDestroy () - executado na destruição da *activity*;
- onRestart () – executado caso o utilizador volte a iniciar a *activity*.

A Figura 2.9 descreve em forma de diagrama as várias fases de uma *activity,* descritas anteriormente, e também as transições que podem ocorrer.

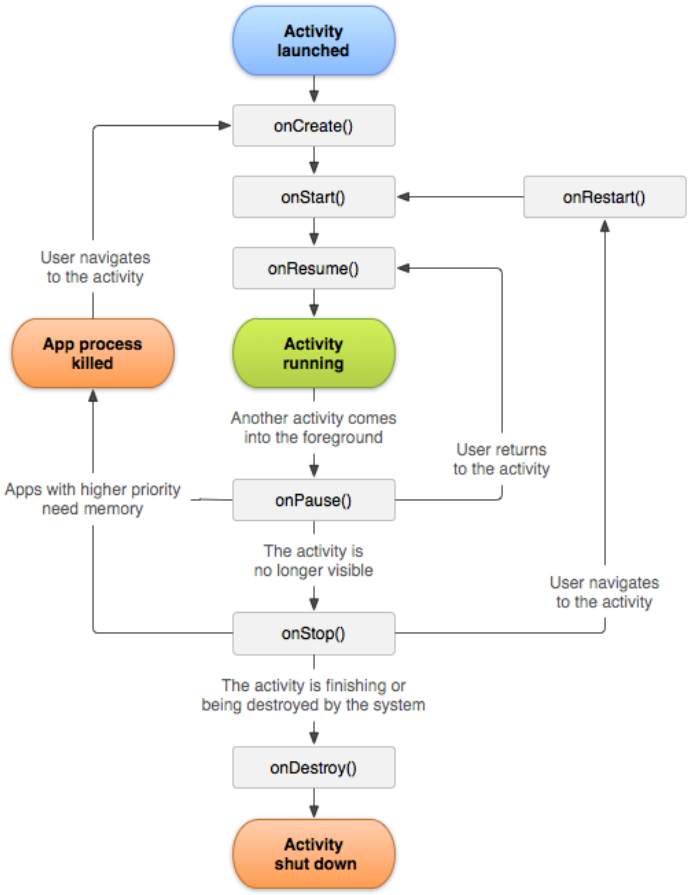


Figura 2.9 - Ciclo de vida de uma *activity* [16]**.**

2.5.2.2 Serviços

Os serviços são componentes da aplicação que podem executar operações de longa execução em segundo plano e não oferecem qualquer tipo de GUI. Uma componente de uma aplicação pode iniciar um serviço e este continuará a ser executado, mesmo que o utilizador mude para outra aplicação [16].

2.5.2.3 Fornecedor de conteúdos

Um fornecedor de conteúdos é uma classe que implementa um conjunto de métodos padrão para permitir o armazenamento e a obtenção de dados numa aplicação. É possível utilizar estes fornecedores para compartilhar dados entre várias

aplicações, assumindo que as aplicações envolvidas possuem as permissões corretas para aceder aos dados [16].

2.5.2.4 Recetor de difusão

Os recetores de difusão processam eventos de mensagens de outros aplicativos ou do próprio sistema. Estas mensagens são criadas através da definição de intents e são difundidas para o sistema. Os recetores de difusão são importantes para a performance e funcionalidade do sistema Android, pois são eles que permitem que uma aplicação registe se está apto para receber eventos e que eventos. Deste modo, sempre que ocorre um evento, o evento não necessita de ser enviado para todas as aplicações.

2.5.3 Interface do utilizador

A interface do utilizador é contruída utilizando as *activities*. Todos os elementos da interface do utilizador numa aplicação Android são construídos usando os objetos Views e ViewGroup [17]. A View é a classe básica para elementos de interface visual e disponibiliza os objetos no ecrã que permitem a interação com o utilizador. ViewGroup (Figura 2.10) é a sua subclasse, que fornece um recipiente invisível que contem outros Views e/ou ViewGroup e define as propriedades do *layout* da interface.

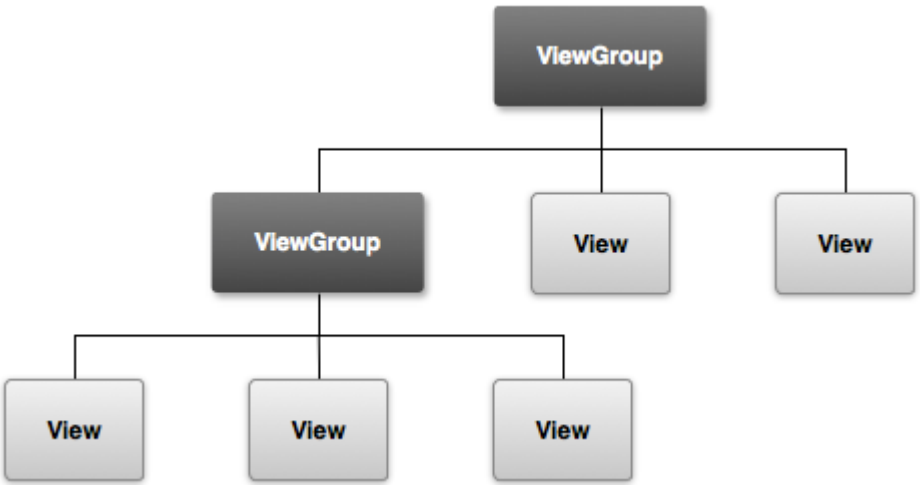


Figura 2.10 - Hierarquia de Views de uma aplicação [17].

É através da subclasse destes objetos que é possível obter diferentes tipos de *layout*s, como Linear ou Relative, e também utilizar componentes como caixas de texto, botões e imagens para proceder ao preenchimento destes *layout*s [17].

Apesar de ser possível criar *layout*s através do código Java é recomendado que estes sejam definidos em ficheiros XML [17]. O XML oferece uma estrutura legível para o *layout* semelhante ao HTML. Estes ficheiros serão invocados pelas *activities* e encontram-se na pasta res/*layout* do projeto que venha a ser desenvolvido.

É possível na programação Android utilizar estilos nas aplicações para manter uma aparência constante ao longo da navegação entre as várias *activities* [17]. Um estilo é um conjunto de propriedades que especificam a aparência e o formato para a View. O ficheiro de estilo é especificado num ficheiro XML separado do ficheiro que especifica o *layout*. Este ficheiro encontra-se numa pasta res/*values*. Quando um estilo é aplicado não só a uma View individual, mas sim à *activity* ou aplicação por completo, designa-se como tema [17]. Para mudar a aparência da aplicação com um tema específico é necessário declarar o tema no ficheiro AndroidManifest.xml.

2.5.4 Android manifest

Todas as aplicações Android precisam de ter este ficheiro que é essencial para o seu funcionamento. O sistema utiliza-o para obter informação administrativa e organizacional sobre a aplicação em questão. É neste ficheiro que estão descritas as *acitivities*, serviços, fornecedores de conteúdo e recetores de difusão usados na aplicação. É também responsável por proteger a aplicação, contendo as permissões de acesso ou mesmo para o seu correto funcionamento, como por exemplo, pelo uso da definição android.permission.BLUETOOTH para permitir o acesso ao Bluetooth. Outra função importante deste ficheiro é definir o nível mínimo da API (Application Programming Interface) do Android que a aplicação necessita [18].

2.6 Trabalho relacionado

Nesta secção é descrita a análise realizada sobre as tecnologias existentes no âmbito desta dissertação. Foram estudados vários documentos relacionados com protocolos de comunicação de curto alcance BLE, fazendo comparações entre algumas tecnologias. Por fim, são apresentadas aplicações que tiveram como objetivo a utilização de sensores para monitorização na área da saúde, desporto e domótica uma vez que BLE é uma tecnologia cada vez mais procurada.

O artigo [19] apresenta a primeira tentativa de caracterizar o desempenho da tecnologia ZigBee dentro de um veículo. Os autores apresentam resultados de experiências de transmissão de pacotes usando nós sensores ZigBee dentro de um carro sob vários cenários, em locais diferentes do carro e com motor ligado/desligado. Os resultados destas experiências sugerem que tanto o RSSSI (Received Signal Strength Indicator) como o LQI (Link Quality Indicator) apenas podem ser usados como um indicador base de um limite para avaliar a qualidade de ligação. A ultrapassagem do limite inferior indica a má qualidade da ligação. Os autores propuseram um conjunto de algoritmos de deteção com base nos padrões RSSI/LQI e uma estratégia adaptativa para aumentar o desempenho de uma ligação (no que concerne ao *goodput*) enquanto reduz o consumo energético do rádio. Para além disto, os resultados dos testes mostraram que as variações na qualidade de ligação com respeito às posições dos nós no carro são significativas. O ruído do motor pode aumentar em 2 a 4 dB os limites da sensibilidade, afetando a taxa de receção de pacotes (PRR - Packet Reception Ratio) e o *goodput*.

Em [10] são exploradas algumas tecnologias sem fios adequadas ao contexto de redes veiculares e investigam-se algumas questões em torno da substituição da cabelagem entre as ECUs e os sensores num veículo de ligações sem fios. O objetivo é explorar tecnologias sem fios adequadas para salientar os problemas principais de arquitetura de rede não otimizado para automóveis, de modo a sugerir algumas soluções. Inicialmente, fizeram um levantamento de tecnologias sem fios de curto

alcance e avaliaram a sua aptidão para as redes veiculares sem fios. A Tabela 2.1 faz uma comparação resumida das tecnologias abordadas.

Tabela 2.1 – Comparação de *standards* existentes (adaptado de [10]).

Nome do *standard* no mercado	Bluetooth	ZigBee	UWB
Taxa de dados (Kbps)	720	20-250	20-250
Alcance de Transmissão (m)	1 –10	1-100	1-100
Tamanho da rede	7	255/65536	255/65536
Topologia de rede	Estrela (fixa)	Flexível (estrela, malha, hibrido)	Flexível (estrela, malha, hibrido)
Protocolo MAC	TDMA	CSMA/CA e TDMA	CSMA/CA e TDMA
Espalhamento espectral	FHSS	DSSS	DS-UWB
Requisitos de memória	250 KB	4-32 KB	4-32 KB
Requerimentos da potência de transmissão	1 mW	<1 mW	1 mW/Mbps
Foco aplicacional	Substituição de cabos	monitorização e controlo	Localização de alta precisão, multimédia

Após uma reflexão das tecnologias abordadas na tabela e das necessidades das redes intra-veiculares (suporte para *payloads* essencialmente pequenos com uma latência inferior a milissegundos, baixo *overhead* dos protocolos, e baixa complexidade do transcetor) foi concluído que a tecnologia ZigBee era a mais apropriada, juntamente com outra proposta emergente baseada em Ultra-Wideband (UWB).

Levando em consideração as mesmas questões que o artigo anterior, os autores em [1] propuseram o uso da tecnologia BLE como uma boa opção para uma arquitetura IVWSN. Após uma comparação detalhada entre o ZigBee e o BLE, este artigo fornece informação sobre o *design* do sistema, configuração e metodologia utilizada para uma aplicação exemplo: um sistema passivo sem chave baseado no BLE. Uma preocupação importante sobre os sensores sem fios é a sua autonomia energética.

De acordo com o artigo [20], tendo em consideração a configuração e *hardware* utilizado, o consumo médio da corrente em modo ativo num evento de conexão foi de 8,2463 mA, com uma duração média de 2,675 ms. Já o consumo da corrente no modo *sleep* é de 0,01 mA. Assim, se o intervalo de conexão do sistema for de 1 s, o consumo médio de corrente será 23 µA, calculado através da seguinte equação:

$$I_c = \frac{(\,2{,}675 \times 8{,}2463) + ((1000 - 2{,}675) \times 0{,}01)}{1000} = 0{,}023\ m \qquad (1)$$

Sabendo-se a corrente média, pode-se calcular a autonomia de uma bateria com uma conexão ativa. Assumindo que se usa uma bateria comum de 3 V (CR2032) e capacidade de 230 mAh, a autonomia pode ser calculada com uma fórmula simples:

$$Tempo_{bateria} = \frac{230}{0.023} = 10\,000\ horas \cong 417\ dias \cong 1{,}14\ ano \qquad (2)$$

Este estudo permite concluir que a autonomia de um nó sensor BLE alimentado por bateria pode ser superior a 1 ano. Os resultados mostraram que o BLE é uma tecnologia promissora e fiável para aplicações que exigem soluções de baixo custo e baixa potência.

A maioria das tecnologias sem fios populares aplicáveis às redes IVWSN ocupam a mesma banda de frequências ISM que o Wi-Fi e o Bluetooth, sendo importante avaliar o desempenho das redes IVWSN sob interferência de dispositivos destas tecnologias, especialmente quando se encontram dentro do veículo. O impacto das interferências dos dispositivos Bluetooth e Wi-Fi pode ser observado claramente a partir das experiências descritas no artigo [21]. Estas experiências foram conduzidas nos dois estados do veículo: parado e em movimento. Os resultados indicam que a tecnologia BLE supera a tecnologia ZigBee em termos de *goodput*, no contexto das IVWSNs, quando existe interferência Wi-Fi no carro. Isto verifica-se devido ao facto da tecnologia BLE ser mais robusta comparativamente com a tecnologia ZigBee. Na Figura 2.11 – Desempenho do BLE e ZigBee em redes IVWSN [21].Figura 2.11 apresentam-se os resultados de quatro configurações diferentes dos sensores nos dois estados do veículo: parado (Lot) e em movimento (Dyn). No final do documento conclui-se que a tecnologia BLE pode ser uma boa proposta paras as redes veiculares, uma vez que é mais resistente a interferências, que são uma grande preocupação para este tipo de redes.

O trabalho [22] apresenta diversas contribuições inovadoras relativamente ao desempenho do protocolo BLE. Os objetivos deste artigo incluem determinar o número de nós suportados no contexto de uma aplicação real e avaliar a qualidade de serviço para o transporte de tráfego de múltiplas fontes de alta taxa de dados. Os testes conduzidos utilizaram os mesmos módulos (CC2540) escolhidos para o desenvolvimento desta dissertação e também *smartphones*. As principais contribuições deste trabalho são: uma análise teórica do desempenho de BLE com vários nós sensores e tráfego de alto débito; uma avaliação experimental da confiabilidade do BLE, com intervalos de ligação pequenos e vários nós sensores, usando duas plataformas BLE diferentes; e uma comparação de desempenho, em termos do número de escravos suportados e consumo de energia, entre os protocolos BLE e eLPRT (Enhanced Low Power Real Time)[23], baseado na norma IEEE 802.15.4. Os resultados mostram que, para o mesmo tráfego, o protocolo BLE apresenta menor consumo de energia e suporta mais nós sensores que o eLPRT.

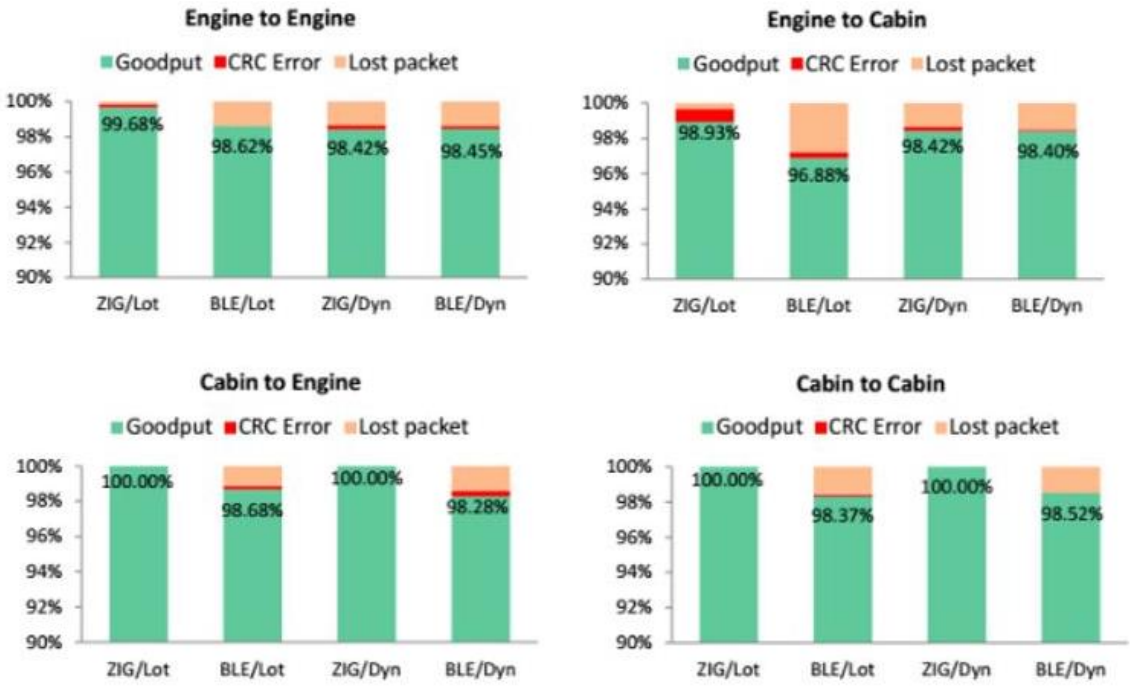


Figura 2.11 – Desempenho do BLE e ZigBee em redes IVWSN [21].

Foi proposto em [24] desenvolver um sistema de monotorização ECG (eletrocardiograma) sem fios que integra a tecnologia BLE (Figura 2.12). Os resultados mostram que a implementação da tecnologia no sistema de monitorização de ECG existente não só elimina as limitações físicas impostas pelas ligações com fios, mas também reduz o consumo de energia do sistema de monitorização de longo prazo.

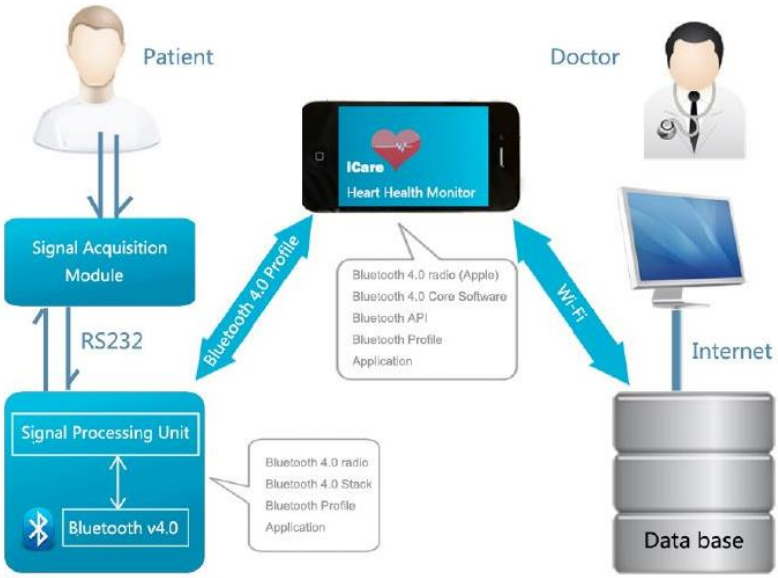


Figura 2.12 – Arquitetura do sistema de monotorização ECG baseado na tecnologia BLE [24].

O facto de a investigação e implementação da automação residencial se popularizar tão rapidamente deve-se à introdução da Internet das Coisas (IoT - Internet of Things), a qual continua a estimular a ideia da possibilidade de conceção de casas cada vez mais inteligentes através das tecnologias sem fios. A eficiência energética, sendo um requisito essencial, torna o protocolo sem fios BLE uma das soluções mais adequadas para o uso na IoT. O artigo [25] propõe um mecanismo baseado em lógica difusa para aumentar a autonomia dos dispositivos de uma rede sem fios num ambiente de automação residencial baseado em BLE, com um nó mestre para o controlo (Figura 2.13). O controlador de lógica difusa (FLC – Fuzzy Logic Controller) proposto determina o tempo de suspensão dos dispositivos em campo, de acordo com o nível da bateria e a proporção entre a taxa de transferência e o *workload* (TH/WL). Os resultados de simulação revelam que ao usar a abordagem FLC proposta, a autonomia dos dispositivos aumenta em 30% em comparação com o uso de tempos de suspensão fixos.

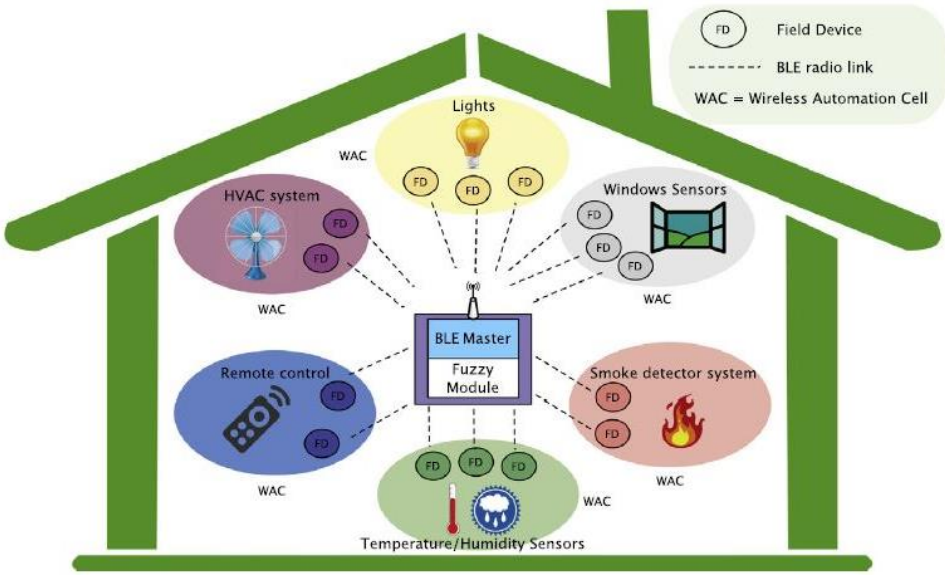


Figura 2.13 – Arquitetura geral do sistema baseado em lógica difusa proposto [25].

A dissertação [26] tem como finalidade analisar e aplicar o protocolo de comunicações sem fios BLE à monitorização da postura de um atleta durante a atividade de ciclismo (Figura 2.14). Este trabalho implementa uma infraestrutura de rede BLE integrado com múltiplos sensores. Foi escolhida uma rede de sensores sem fios para garantir maior flexibilidade e mobilidade de utilização aos atletas que estão a ser monitorizados. Os dados relativamente à postura são obtidos de forma periódica de acelerómetros, magnetómetros e giroscópios, por meio de módulos BLE CC2540, e enviados para um *smartphone*, que desempenha o papel de estação central. Foi desenvolvida uma aplicação Android que recebe, processa e apresenta os dados vindos dos vários nós sensoriais. O processamento dos dados passa pelo cálculo da postura nos segmentos corporais assinalados com base nos dados brutos recolhidos dos sensores. Foi também realizada uma análise intensiva ao protocolo BLE e discutidos os resultados obtidos por via teórica e experimental. O autor conclui que o BLE é uma tecnologia promissora no ramo das redes de sensores sem fios, com uma topologia de rede escalável, o que permite conectar diversos módulos sensoriais.



Figura 2.14 – Arquitetura geral do sistema do sistema de monitorização da postura [26].

3. Desenvolvimento do sistema

Este capítulo tem como objetivo descrever a implementação da aplicação gráfica para Android e do protocolo de comunicação entre as ECUs e os respetivos módulos sensoriais, encontrando-se dividido em cinco secções. Primeiramente apresenta-se uma descrição geral do sistema, seguida por uma breve descrição do veículo elétrico utilizado. O *firmware* usado para a comunicação entre as ECUs é aqui descrito, assim como o protocolo de comunicação para aquisição dos dados sensoriais recolhidos pelas estações. Para além disto, é abordada a aplicação Android desenvolvida e descrita a interface gráfica adotada.

3.1 Introdução

O sistema desenvolvido é um sistema de monitorização e controlo sem fios de um veículo elétrico com o objetivo de melhorar o seu desempenho, assim como auxiliar o condutor de forma mais eficiente possível. O sistema envolve um veículo elétrico e dois tipos de estações: uma estação central, junto ao condutor, e várias estações periféricas (Figura 3.1).



Figura 3.1 - Esquema geral do sistema de monitorização do veículo elétrico.

As estações periféricas foram colocadas em zonas diferentes do veículo, dependendo da sua função. A estação localizada junto à bateria fornece alguma informação necessária para o seu carregamento (Figura 3.2). Esta estação recolhe dados do respetivo módulo sensorial (sistema de bateria) através da interface UART (Universal Asynchronous Receiver/Transmitter). Também foi colocada uma estação junto ao motor (Figura 3.3) que fornece vários dados necessários para a interação com o cockpit do carro, informando o condutor da autonomia da viatura e temperatura do motor. Esta estação também recolhe os dados do respetivo módulo sensorial (sistema de tração) através da interface UART, exceto os dados referentes à temperatura do motor, uma vez que aquisição de dados neste caso é feita através de um conversor analógico-digital (ADC - Analog-to-Digital Converter).

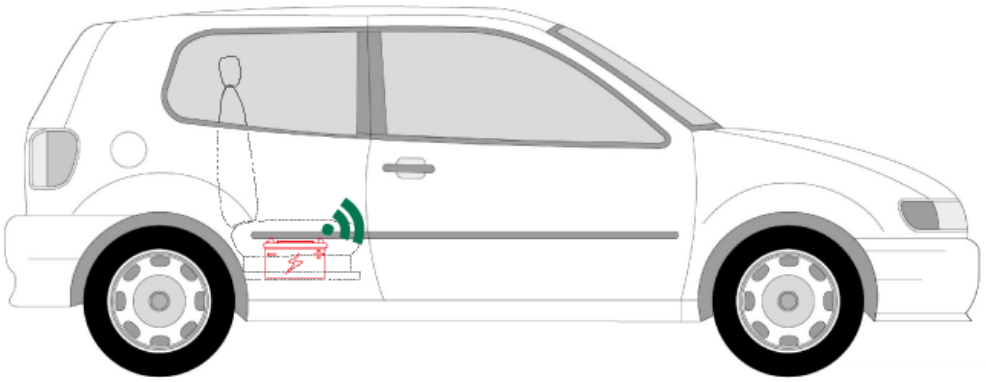


Figura 3.2 - Localização da estação sensorial da bateria.

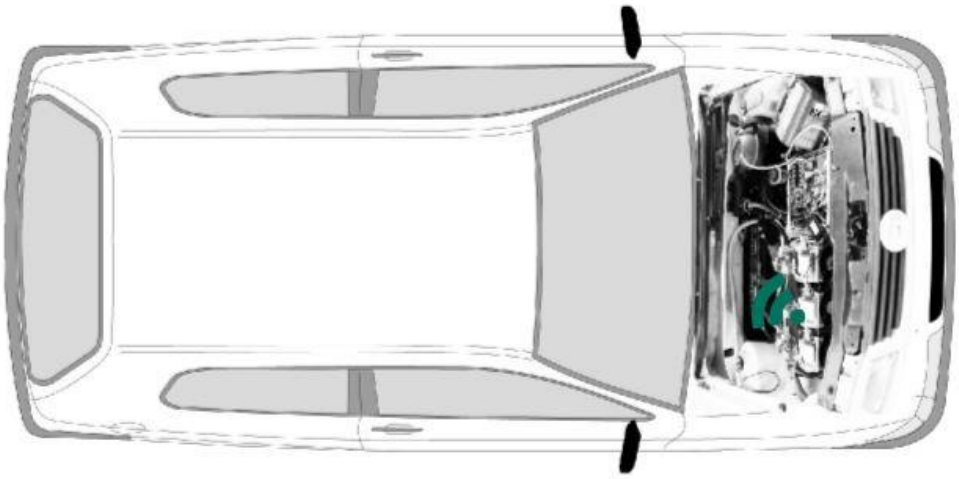


Figura 3.3 - Localização da estação sensorial do motor

A estação central recebe os dados enviados pelas estações periféricas via BLE. Após a receção, os valores dos dados são processados e convertidos para que seja possível a compreensão do seu significado utilizando unidades de medida adequadas.

Numa fase inicial foi usada uma placa de desenvolvimento SmartRF05EB com um módulo CC2540 em todas as estações. Na fase final a placa utilizada como estação central foi substituída por um *smartphone* Android, de modo a permitir apresentar os dados de forma mais clara e facilitar a visualização e controlo do estado do veículo.

3.2 CEPIUM

O sistema descrito nesta dissertação foi desenvolvido e testado utilizando um veículo elétrico desenvolvido pelo GEPE, denominado CEPIUM. Este veículo é um Volkswagen Polo do ano 1996, que sofreu as devidas alterações de modo a converter um veículo com motor de combustão interna num veículo elétrico. Todo o sistema de ligações do motor elétrico e os circuitos eletrónicos complementares necessários para a conversão foram integralmente desenvolvidos na Universidade do Minho.

O veículo possui duas baterias de gel/chumbo de 12 V com uma capacidade de 9,5 kWh, o que oferece uma autonomia de 70 km. As baterias estão localizadas em dois locais distintos: uma encontra-se no antigo espaço do depósito do combustível, e a outra encontra-se onde se situava o antigo motor de combustão interna de 1.2 L. No final destas modificações o CEPIUM ficou um pouco mais pesado, uma vez que ficou com um peso de 1200 kg, enquanto o peso original do veículo era de 850 kg [5] [27]. O veículo consegue atingir os 100 km/h em menos de 14 s, praticamente o mesmo tempo que com o motor de combustão original.

3.3 *Hardware* BLE

O *kit* de desenvolvimento CC2540 fornecido pela Texas Instruments (TI) foi o *hardware* escolhido para o desenvolvimento do sistema de comunicações sem fios [28], [29]. Este *kit* inclui dois módulos CC2540EM, duas placas de desenvolvimento

SmartRF05EB Rev. 1.8.1.1, duas antenas e um *dongle* USB CC2540 (Figura 3.4). Os dados são transmitidos através da tecnologia BLE, que opera na banda de frequências de 2,4 GHz. O circuito integrado CC2540 presente no módulo CC2540EM possui duas USARTs (Universal Synchronous/Asynchronous Receiver/Transmitter) com suporte para diversos protocolos série e um conversor analógico-digital (ADC - Analog-to-Digital Converter) de 12 bits com 8 canais de resolução.

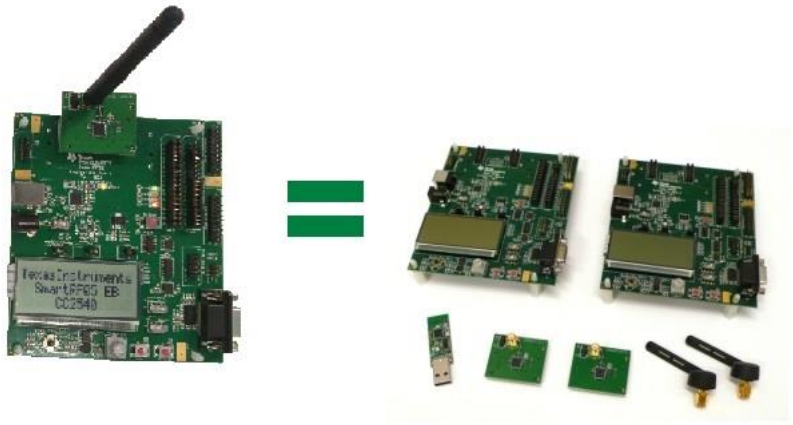


Figura 3.4 - As estações usadas no sistema de monitorização são compostas pelos seguintes componentes: CC2540EM, SmartRF05EB e uma antena [19].

As placas de desenvolvimento oferecem os componentes que se podem observar na Figura 3.5, dentre os quais se destacam as interfaces USB e RS-232, o LCD e o joystick.

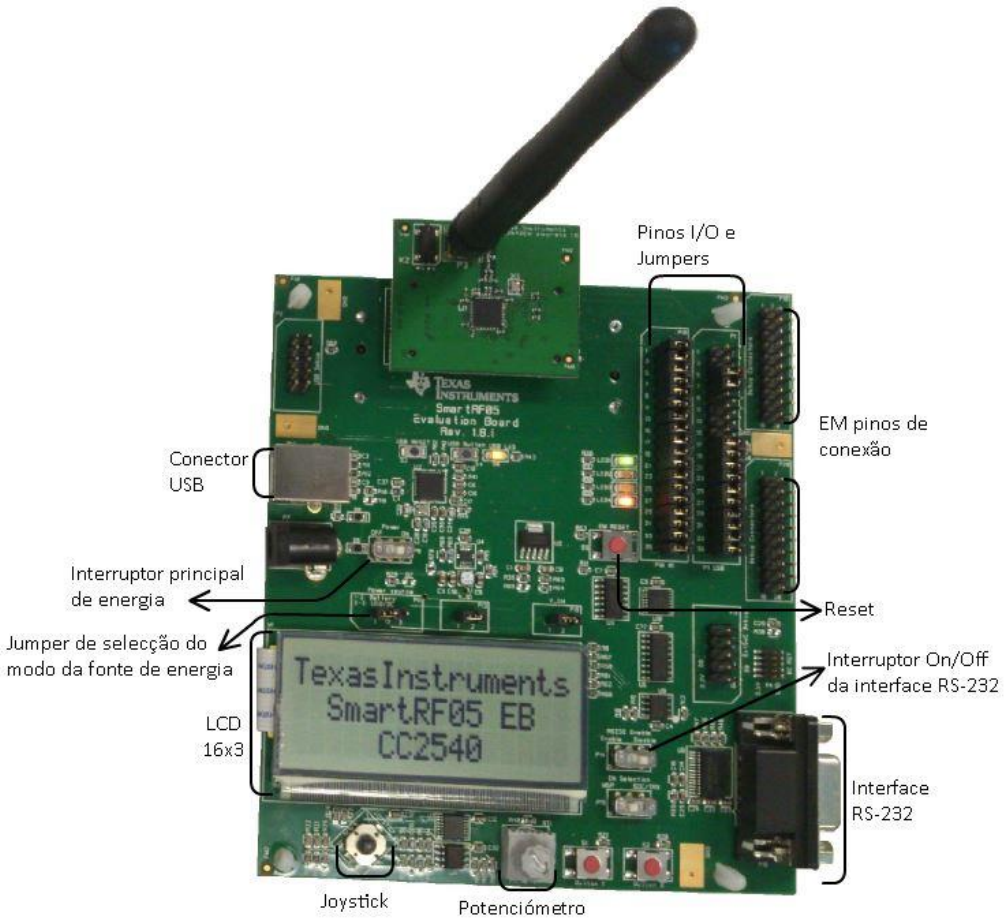


Figura 3.5 – Localização das componentes na placa de desenvolvimento.

O ambiente de *software* utilizado para programar os módulos na linguagem C foi o IAR Embedded Workbench for 8051 [31]. Por último, também foi utilizado o *software* SmartRF Packet Sniffer para testes do sistema com o uso auxiliar do *dongle* USB CC2540.

3.3.1 Desenvolvimento do *firmware*

Na fase inicial o objetivo do programa desenvolvido era estabelecer a comunicação periódica entre as estações periféricas e a estação central. Foram implementadas com sucesso duas formas de comunicação: notificações e indicações. As suas diferenças práticas são discutidas no capítulo de testes e resultados mais à frente neste documento. O código desenvolvido nesta componente teve como base dois projetos exemplo, SimpleBLECentral e SimpleBLEPeripheral, fornecidos pela TI na

instalação da *stack* versão BLE-CC2540-1.3.2 e uma adaptação do serviço "Send Data Service" desenvolvido em [26].

3.3.1.1 Estação periférica

A inicialização do programa das estações periféricas ocorre em duas fases. A função SimpleBLEPeripheral_Init define os parâmetros das camadas GAP e GATT, e define o evento SBP_START_DEVICE_EVT. Este evento despoleta a segunda fase, onde a estação periférica entra no estado de anúncio até que uma estação central entre em contacto. Deste modo, quando recebe um pedido de conexão, altera o seu estado para conectado. Caso não seja feito nenhum pedido de conexão num determinado intervalo de tempo, a estação entra no estado de espera [32].

Para o programa funcionar e transmitir informação de forma periódica foi necessário criar um perfil para disponibilizar este serviço. Este serviço encontra-se implementado na estação periférica, pois esta assume o papel de servidor GATT. O evento periódico, SBP_PERIODIC_EVT, ocorre a cada 500 ms, para o caso de uma comunicação por notificação, ou a cada 1 s, no caso de comunicação por indicação, e invoca a função para o envio dos dados via BLE (PeriodicSendDataTask()), conforme ilustrado na Figura 3.6 e na Figura 3.7. Os valores do evento são escolhidos e definidos pelo programador. Esta função obtém valores provenientes de sensores periodicamente através da função PeriodicReadDataTask(), representada na Figura 3.8. O período do evento periódico é definido por UART_PERIODIC_EVT e ocorre a cada 500 ms. Na fase inicial foi necessário gerar valores aleatórios para simular estes valores.

O bloco responsável pelo envio da informação é dominado sendDataServiceNotify e oferece um perfil que constitui uma única característica, cujo tamanho máximo na trama é de 20 bytes, e um descritor (CCC - Client Characteristic Configuration). Se a comunicação estiver configurada para indicações, o dispositivo espera por uma confirmação (ACK) para cada indicação enviada.

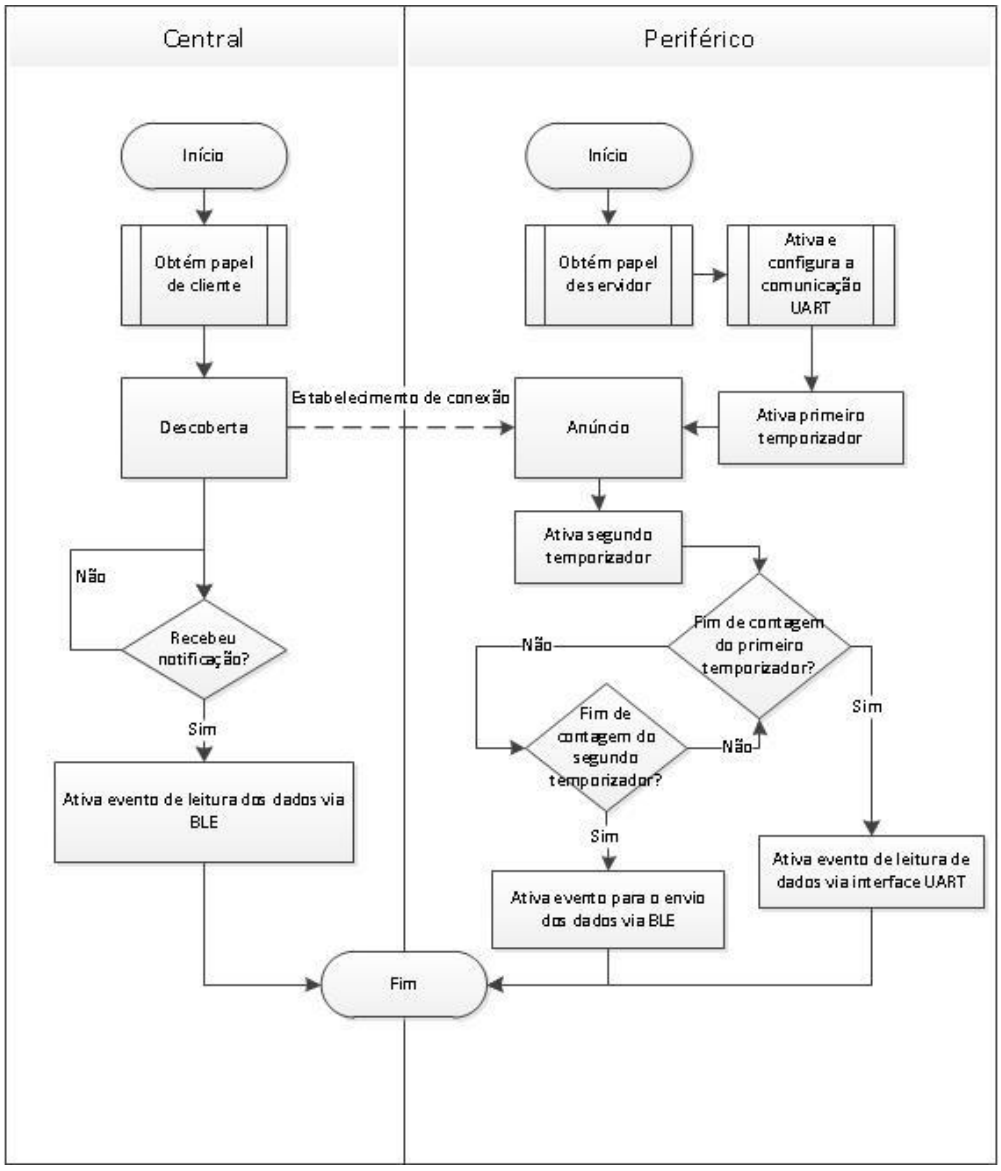


Figura 3.6 – Fluxograma para as estações central e periférica.

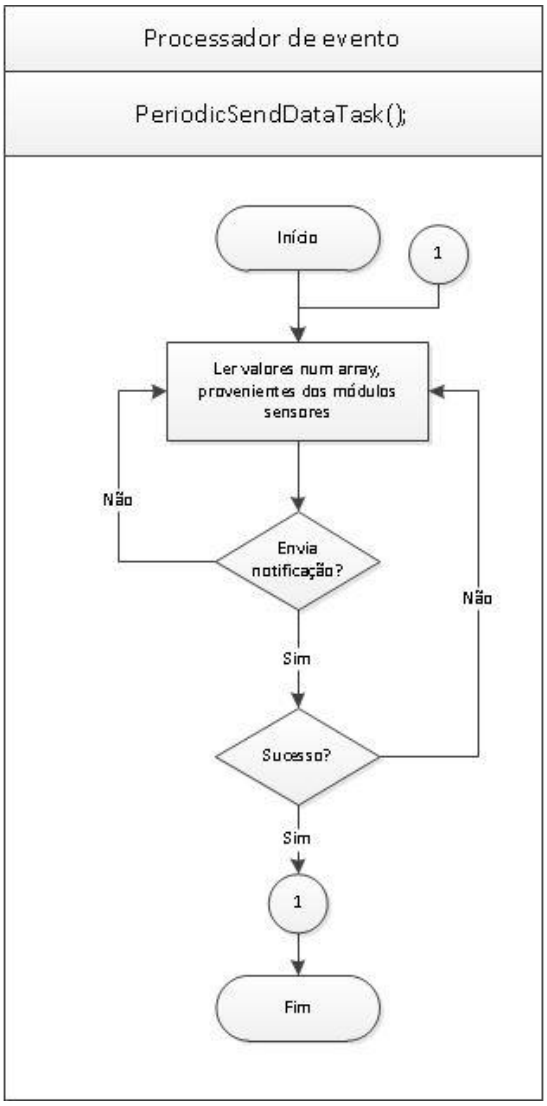


Figura 3.7 - Fluxograma do processador de evento PeridicSendDataTask().



Figura 3.8 - Fluxograma do processador de evento PeridicReadDataTask().

3.3.1.2 Estação central

A estação central inicia a descoberta de dispositivos após mover-se o *joystick* para cima, passando desta forma do estado de *espera* para o estado de descoberta. O seu movimento para a esquerda permite percorrer todos os dispositivos que se encontram no estado de *advertisement*. Para selecionar o dispositivo a que se pretende conectar, basta pressionar o *joystick*. Caso a conexão seja feita com sucesso, o seu estado é alterado para o estado de conexão.

Assim como as estações periféricas, a inicialização da estação central é feita em duas fases: a função SimpleBLECentral_Init inicia o funcionamento da camada, define os parâmetros das camadas GAP e GATT, e define o evento SBP_START_DEVICE_EVT. Para além disto, é nesta fase que as permissões para receber dados através de notificações ou indicações são ativadas. Com o evento SBP_START_DEVICE_EVT, a segunda fase é desencadeada, e através do bloco simpleBLECentralProcessGATTMsg [32] (Figura 3.9) as mensagens são processadas. Esta função recebe todas as comunicações e processa-as, sejam elas notificações ou indicações. A fim de testar a finalidade do sistema numa comunicação com notificações foi implementada uma função que calcula o Packet Loss Rate (PLR).

3.3.1.3 Indicações

Como já foi referido anteriormente, o desenvolvimento deste código teve como base dois projetos disponibilizados pela TI. Para tal, foi necessário fazer algumas alterações para ser possível uma comunicação por indicações. Estas modificações foram feitas principalmente para as placas periféricas. Foi criada uma característica que assume o valor GATT_PROP_INDICATE e registar o descritor CCC com o valor de 0x0002 para permitir o serviço avaliar se as indicações se encontram ativas. É também preciso alterar o tipo de dados que vão ser tratados para attHandleValueInd_t, e finalmente enviar os dados através da função GATT_Indication para o dispositivo central.

Já no dispositivo central não foram feitos muitos ajustes, uma vez que se encontrava preparado para receber dados tanto na forma de notificação como indicação. Apenas foi necessário acrescentar a função ATT_HandleValueCfm para enviar os respetivos ACKs por cada pacote recebido.

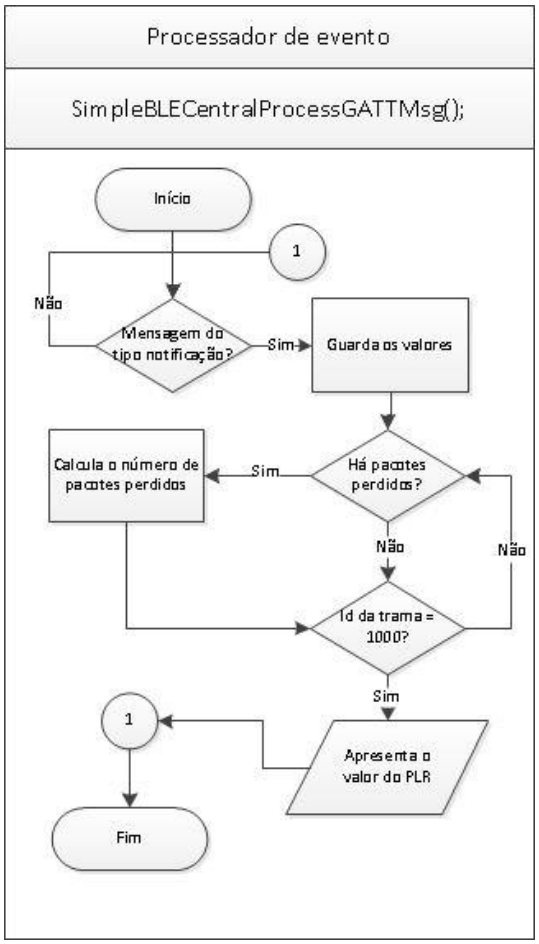


Figura 3.9 - Fluxograma de processador de evento SimpleBLECentralProcessGATTMsg().

3.3.1.4 Configurações USART

De modo a ser possível a comunicação entre a placa periférica e os sensores, foram implementadas as configurações USART necessárias. Esta comunicação foi feita por interrupções (ISR - Interrupt Service Routine). Para que tal seja possível, primeiro foi necessário acrescentar duas definições às opções do projeto: HAL_DMA=TRUE e HAL_UART = TRUE (Figura 3.10).

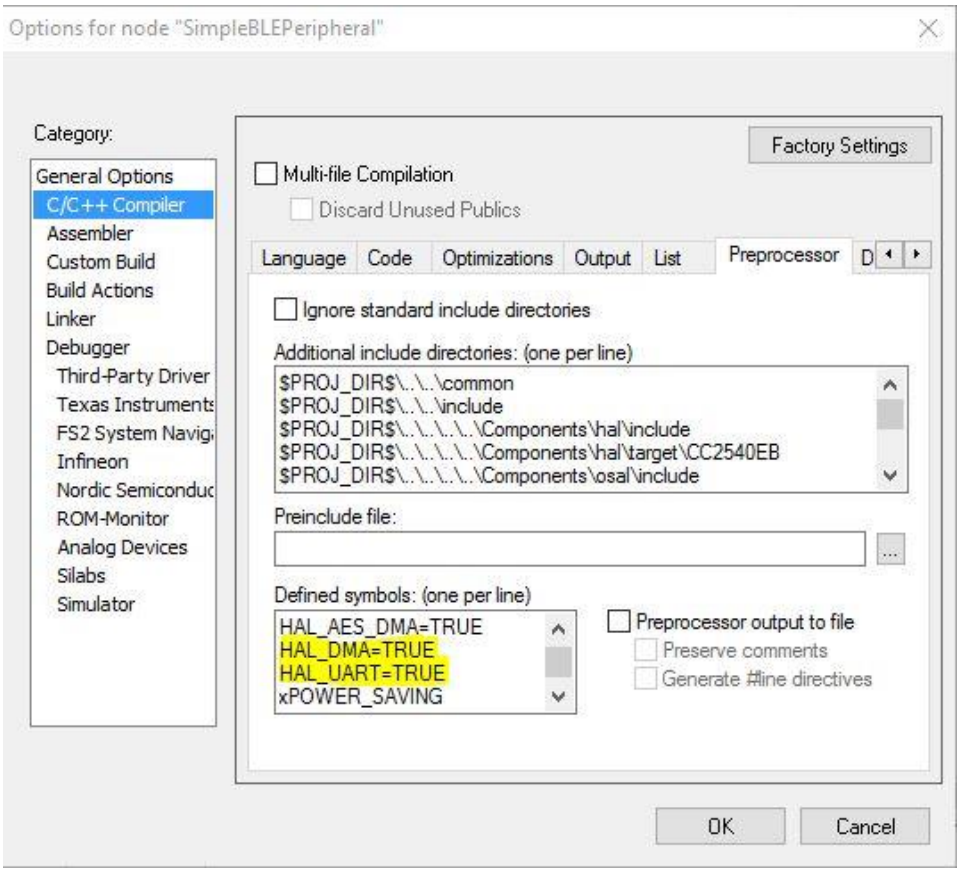


Figura 3.10 – Definições acrescentadas às opções do projeto.

Para enviar dados foi implementado um evento periódico que envia pedidos num intervalo de tempo de 500 ms. Quanto à receção dos dados a leitura é despoletada quando o *buffer* se encontra quase cheio, cheio ou ocorra um *timeout*.

3.3.1.5 Conversão de um valor ADC para um valor

De modo a ser possível a comunicação entre a placa periférica e o sensor temperatura foi necessário acrescentar a definição HAL_ADC=TRUE (Figura 3.10) às opções do projeto.

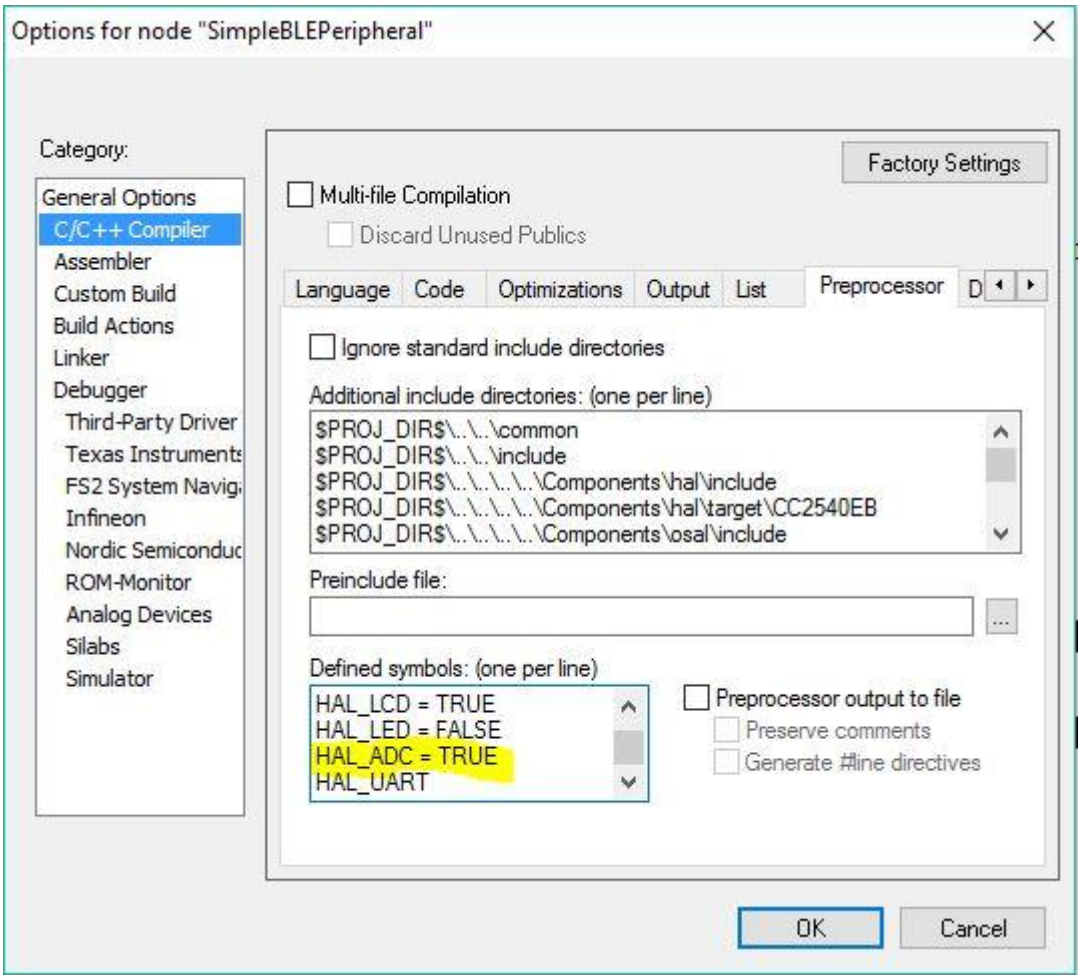


Figura 3.11 – Definição ADC acrescentada às opções do projeto.

A função HalAdcRead() permite a leitura do valor digital proveniente do sensor. Esta função aceita dois parâmetros, o canal e o número de bits de resolução de leitura. Existem 8 canais de leitura possíveis, do canal 0 ao canal 7, associado ao porto 0 por defeito. De modo a ser possível usar os pinos de conexão EM como entradas, o registo APCFG deve ser configurado com o valor "1". Os pinos definidos pelo código são diferentes dos pinos utilizados na placa SmartRF05EB; assim, após um estudo dos documentos [28], [30], foi construída a Tabela 3.1, que mostra a relação entre os portos utilizados no código e os pinos correspondentes nos conectores da placa (Figura 3.12).

Tabela 3.1 – Correspondência dos portos no código e os portos na placa SmartRF05EB.

Código	SmartRF05EB
P0.0	P20.12
P0.1	P18.07
P0.2	P18.09
P0.3	P18.11
P0.4	P18.13
P0.5	P18.15
P0.6	P20.17
P0.7	P18.17

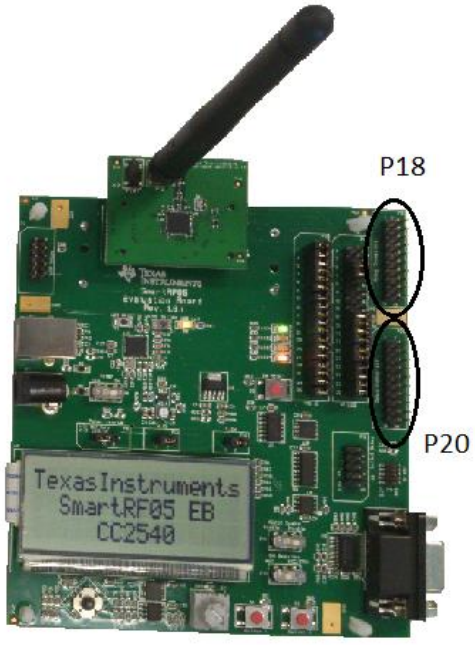


Figura 3.12 – Localização dos portos P18 e P20.

O parâmetro de resolução permite quatro opções de leitura: de 8, 10, 12 ou 14 bits. Dado que o valor retornado pela função previamente referida é um valor em formato *offset* binário do ADC, o valor máximo lido pelo ADC é dado pela seguinte equação:

$$N_{max} = 2^{R-1} - 1, \quad (3)$$

onde *R* toma um dos quatro valores de resolução. Para auxiliar o estudo da interface ADC recorreu-se ao potenciómetro incorporado na placa de desenvolvimento para gerar valores. Com a utilização de um multímetro foi possível medir o valor da tensão de alimentação do potenciómetro (V_{cc}), registando-se o valor de 3,3 V. Com isto foi possível escrever a equação (4) que permite obter uma relação entre o valor digital recebido através da interface (V_{adc}) e o valor real (V_{volts}) correspondente em Volts.

$$V_{volts} = \frac{V_{cc}}{N_{max}} \times V_{adc} \quad (4)$$

N_{max} resulta da equação (3) e V_{adc} é o valor de retorno da função HalAdcRead(). Em trabalhos futuros a leitura do potenciómetro deve ser substituída por sensores.

3.4 Protocolo de comunicação

Este protocolo tem como objetivo descrever a comunicação sobre UART entre o sistema de bateria ou o sistema de tração e o respetivo módulo BLE periférico CC2540EM. O protocolo consiste num mecanismo de *polling*, no qual os pedidos são efetuados de 500 em 500 ms. O módulo BLE envia periodicamente um pedido para receber os dados relacionados com os diferentes sensores e espera por uma resposta, como se pode verificar no diagrama de sequência da Figura 3.13.

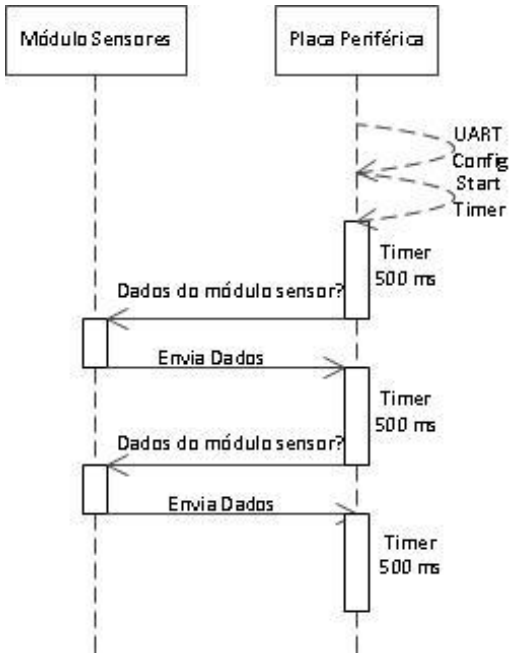


Figura 3.13 - Diagrama de sequência da comunicação.

O envio do pedido é feito através de uma trama de 1 byte com um único campo. Este byte serve para informar ao sistema que tipo de dados espera receber. Na Tabela 3.2 pode-se ver a identificação atribuída para o campo tipo da trama de pedido.

Tabela 3.2 - Correspondência do campo tipo de trama.

ID_Tipo	Tipo
0	Reservado
1	Dados de todos os sensores
2 a 255	Reservado

No desenvolvimento da dissertação apenas se usa um tipo de trama, os restantes 255 valores encontram-se reservados para uso futuro caso seja necessário.

3.4.1 Sistema da bateria

A Figura 3.14 representa um esquema geral do sistema da bateria, com a indicação dos 6 sensores que este contém, bem como a ligação à respetiva estação periférica.

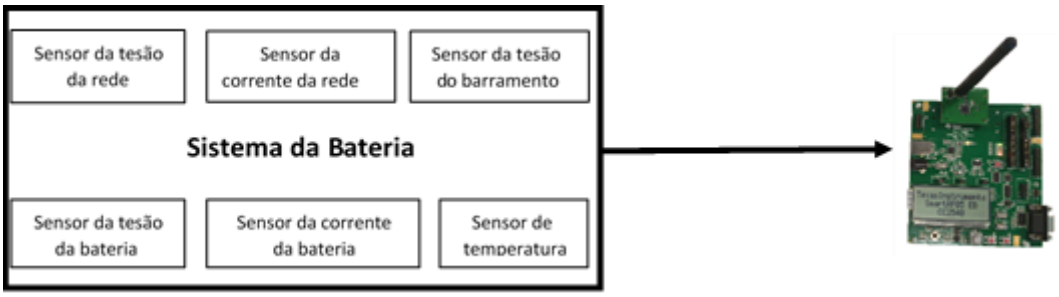


Figura 3.14 - Esquema geral do sistema da bateria.

O sistema da bateria envia um pacote de tamanho variável para a estação periférica. A trama enviada (Figura 3.15) é composta pelo endereço do sistema (1 byte), o tipo da trama (1 byte), o seu tamanho e as respetivas amostras dos 6 sensores: tensão da rede (Tr), corrente da rede (Cr), tensão do barramento (Tb), tensão da bateria (Tb), corrente da bateria (Cbat) e temperatura (T). Os valores encontram-se separados por um limitador ";", previamente definido. O sistema da bateria tem implementado um sistema de comunicação que apenas envia caracteres de "0" a "9" da tabela ASCII, pelo que não há a possibilidade de encontrar valores iguais ao carácter definido como limitador. Os valores lidos pelos sensores são processados e enviados pelo sistema da bateria como valores inteiros, não havendo necessidade de precaução de processar valores com casas decimais.

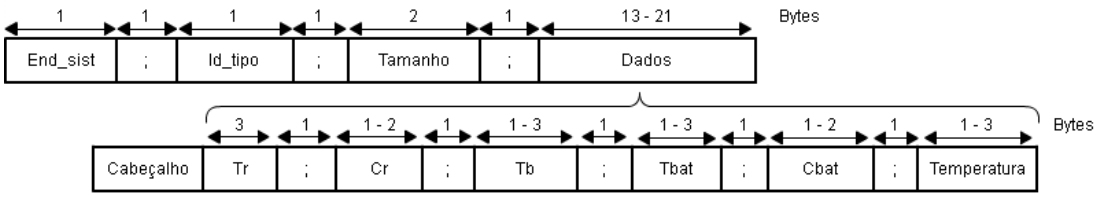


Figura 3.15 – Trama do sistema de bateria.

Na Tabela 3.3 pode-se ver a identificação atribuída para o campo endereço do sistema e na Tabela 3.4 a relação para o campo tipo de trama.

Tabela 3.3 - Correspondência do campo endereço do sistema.

End_sist	Sistema
0	Reservado
1	Sistema da bateria
2	Sistema de tração
3-255	Reservado

Tabela 3.4 - Correspondência do campo tipo de trama.

ID_Tipo	Tipo
0	Reservado
1	Mensagem de erro
2	Dados de todos os sensores
3 – 255	Reservado

As mensagens de erros encontram-se descritas na secção 3.4.3. Os restantes endereços encontram-se reservados caso seja necessário para um desenvolvimento futuro.

3.4.2 Sistema da tração

A Figura 3.16 apresenta o esquema geral do sistema de tração, com a indicação dos respetivos sensores.

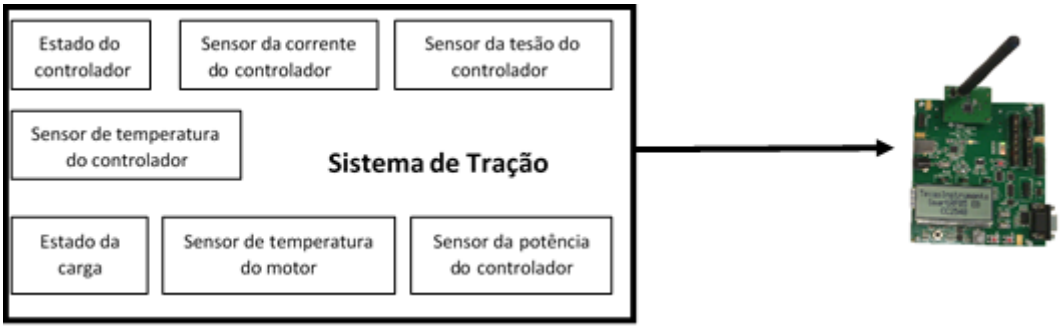


Figura 3.16 - Esquema geral do sistema de tração.

O sistema da tração envia um pacote de tamanho variável. A trama enviada (Figura 3.17) é composta pelo endereço do sistema (1 byte), o tipo da trama (1 byte), tamanho e as amostras dos seus 7 sensores: estado do controlador (Ec), corrente do controlador (Cc) e tensão do controlador (Tc), estado da carga (Ecarga), potência do controlador (Pc) e temperatura controlador (TempC). Como referido no sistema anterior, os valores encontram-se separados por um limitador ";", o sistema tem implementado uma comunicação por caracteres de "0" a "9" da tabela ASCII, e os valores são previamente processados e enviados como valores inteiros.

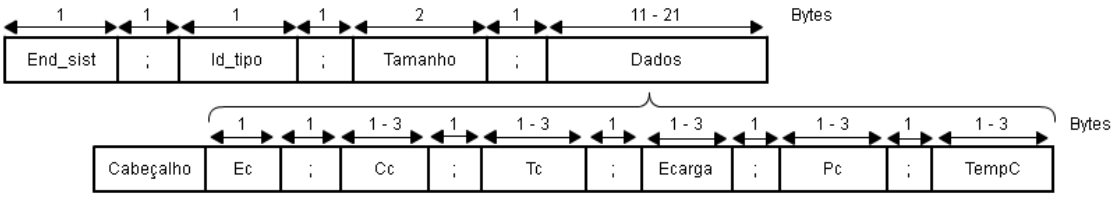


Figura 3.17 - Trama do sistema de tração.

A descrição do endereço do sistema e do tipo do pacote encontram-se na Tabela 3.3 e na Tabela 3.4 respetivamente.

3.4.3 Mensagens de erro

Em caso de algum sistema, seja o sistema de bateria seja o sistema de tração, não obter resposta de algum sensor, é comunicada à estação periférica uma mensagem de erro.

Os sistemas enviam um pacote de 7 bytes. A Figura 3.18 demonstra os seus diferentes campos. O campo do endereço do sistema, ocupa 1 byte, assim como os campos relacionados com o tipo e o tamanho. Os restantes campos contêm a mensagem "erro". Os campos encontram-se separados por um limitador ";", previamente definido.

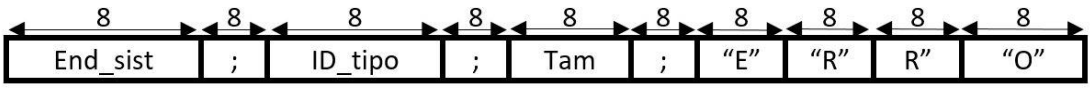


Figura 3.18 - Mensagem de erro.

A descrição do endereço do sistema e do tipo do pacote encontram-se na Tabela 3.3 e na Tabela 3.4 respetivamente.

3.4.4 Processamento das tramas nas estações periféricas

Como foi referido anteriormente, como os valores das tramas são enviados como caracteres, foi necessária a implementação de duas funções de otimização da trama na placa periférica antes de enviar a informação via BLE. As duas funções de otimização tiveram como objetivo diminuir o tamanho excessivo da trama relativamente ao tamanho máximo do *payload* permitido para o envio numa trama BLE, que é de 20 bytes. A primeira função retira o caracter limitador e a segunda função transforma os caracteres em valores numéricos do tipo inteiro.

3.5 *Software* Android

Uma vez que um dos objetivos propostos era a implementação de uma aplicação que fosse o mais eficiente possível, foi necessário ter em consideração alguns fatores importantes na sua estruturação.

Na construção da aplicação foi necessária a implementação de várias funções de comunicação com os módulos sensores, que envolvem receção de dados. A aplicação foi desenvolvida com o objetivo de proporcionar um acesso simples a esta informação, sendo este o aspeto mais importante no seu planeamento e implementação.

Nesta secção são discutidos o processo de desenvolvimento da aplicação final, a sua estrutura, as principais dificuldades enfrentadas e as soluções adotadas para os problemas encontrados.

3.5.1 Introdução

A aplicação desenvolvida, denominada "CEPIUM", é executada no *smartphone* e assume o papel de estação central, servindo de interface com o utilizador. O ambiente de *software* utilizado para o seu desenvolvimento foi o Android Studio 2.1 [33]. A aplicação recebe periodicamente a informação recolhida pelos nós sensoriais (estações periféricas) presentes em determinadas localizações do automóvel e apresenta-as de forma simplificada ao utilizador.

A aplicação foi projetada para que a sua interface com o utilizador fosse flexível e simples, facilitando o acesso de um modo organizado aos dados dos sensores. Com este aspeto em mente a aplicação foi criada utilizando maioritariamente *fragments*, permitindo a reutilização simplificada de certos componentes da interface com o utilizador. No total foram utilizadas quatro Activities. Uma foi utilizada na criação do menu inicial, a segunda foi utilizada para a descoberta de dispositivos e a terceira apresenta os dados do módulo sensor conectado.

Para ter acesso aos dados sensoriais, inicialmente deve-se conectar o *smartphone* com as estações periféricas. Deste modo, a aplicação interroga a rede à procura dos dispositivos disponíveis, apresentando-os numa lista. Assim, o utilizador tem a oportunidade de selecionar o dispositivo com que pretende estabelecer a conexão. Após efetuada a conexão, é possível obter informação do nó sensorial e ativar a receção dos dados através de notificações. Os dados são enviados periodicamente e são geridos por um serviço em background, dando a possibilidade ao utilizador de realizar outras operações dentro da aplicação sem perder a conexão e os dados transmitidos. Os dados são apresentados sob a forma de indicadores de tensão, corrente e temperatura. A Figura 3.19 ilustra a estrutura da aplicação em termos de funcionalidades.

Os blocos enumerados apresentados na figura acima correspondem às principais *activities* do sistema. O bloco assinalado com o número 1 corresponde à *activity* principal. Esta *activity* serve de interface inicial da aplicação e realiza a gestão dos *fragments*. A segunda *activity* desta aplicação encontra-se marcada com o número 2.

Os dispositivos encontrados via BLE são aqui apresentados. Após a seleção do dispositivo e o estabelecimento de conexão é apresentada *activity*, sinalizada com o número 3. Esta *activity* é responsável por criar serviços em background, que recebem as amostras via BLE e fazem a gestão das conexões com os dispositivos. Por fim, a *activity* exposta com o número 4 apresenta o estado do tempo. Nas próximas secções estas *activities* são explicadas com mais detalhe. Foi criada uma *navigation drawer* com vários menus, representada pela janela à esquerda da *activity* 1.

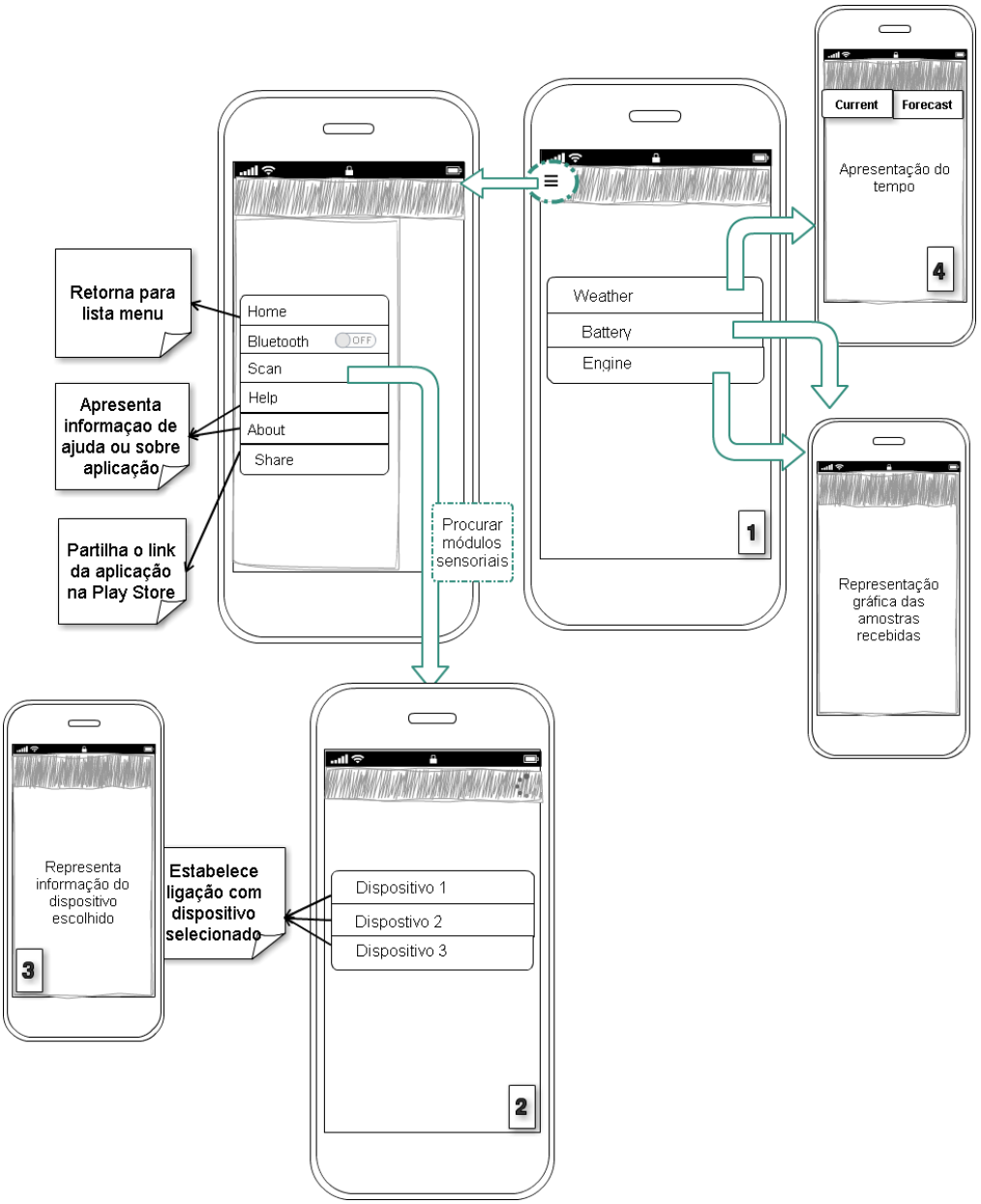


Figura 3.19 - Estrutura geral da aplicação desenvolvida.

3.5.2 Lista menu

Dado o propósito de oferecer um acesso simples à informação, a aplicação desenvolvida oferece a possibilidade de realizar diferentes ações com base na utilização de *fragments* que controlam a interface com o utilizador. As ações disponíveis permitem o acesso às seguintes funcionalidades (Figura 3.20):

- Ver informação do sistema da bateria
- Ver informação do sistema de tração;
- Verificar as condições do tempo.

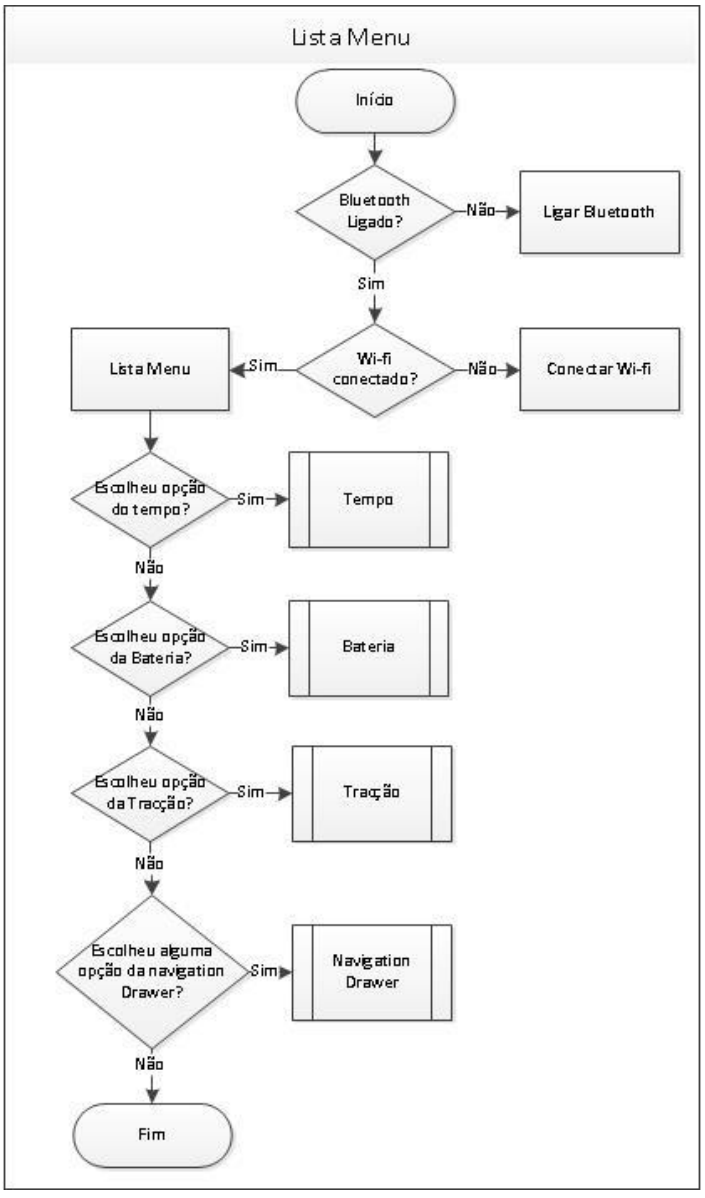


Figura 3.20 – Fluxograma da *activity* do menu principal.

As outras funcionalidades desta *activity* estão disponíveis num *navigation drawer* e encontram-se presentes em qualquer *fragment* (Figura 3.21):

- Ligar/desligar Bluetooth;
- Procurar módulos sensores;
- Ver informação da aplicação;
- Ver informação sobre o modo de utilização da aplicação;
- Partilhar o *link* da aplicação na Play Store.

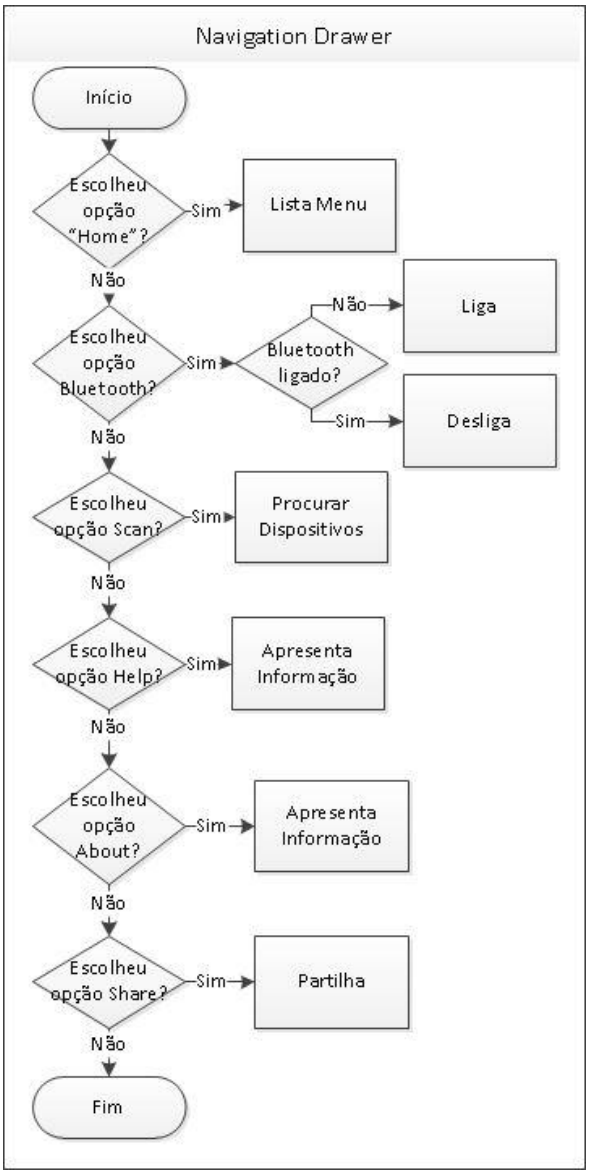


Figura 3.21 – Fluxograma do *navigation drawer*.

Após ligar o Bluetooth, para a aplicação funcionar corretamente, é apresentado o menu principal ao utilizador para escolher a ação que pretende efetuar.

Foi projetado e criado um *navigation drawer* (Figura 3.22) [34] que permite a navegação de menus numa aplicação. Este é implementado diretamente na *activity* que gere os diferentes *fragments* de modo a garantir a sua visibilidade, mesmo que um diferente menu seja escolhido. Este painel pode ser acedido com um clique no *icon* por defeito do Android, um Hamburger *icon*, ou efetuar-se um *swipe* da esquerda para a direita. O motivo de implementar esta funcionalidade foi o facto de não necessitar ocupar a interface em todos os momentos da aplicação. Com a seleção de um dos menus deste painel, a *fragment* correspondente a esse menu é inicializada. Na figura estão representados dois casos do *navigation drawer*, quando o Bluetooth se encontra ligado (figura à esquerda) e desligado (figura à direita).

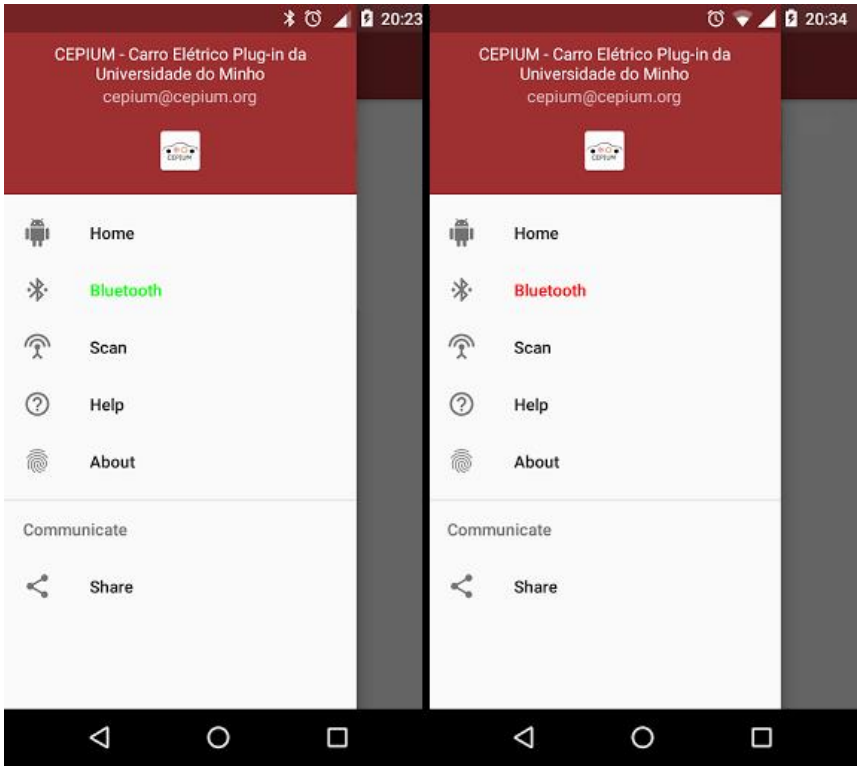


Figura 3.22 – *Navigation drawer* da aplicação.

3.5.2.1 Bateria e tração

O propósito desta *fragment* consiste em apresentar ao utilizador os dados recebidos dos módulos sensores. Os valores são representados com indicadores simples de tensão, corrente e temperatura. A Figura 3.23 e Figura 3.24 demonstram as *fragments* com os respetivos indicadores para os dados dos sensores. Inicialmente, o sistema da bateria apresenta os valores nominais indicados pelo GEPE e o sistema de tração os valores mínimos.

Os dados provenientes dos módulos sensores, são geridos e manipulados pela classe implementada, DataHandler.



Figura 3.23 – *Fragment* responsável por apresentar os dados do sistema da bateria.

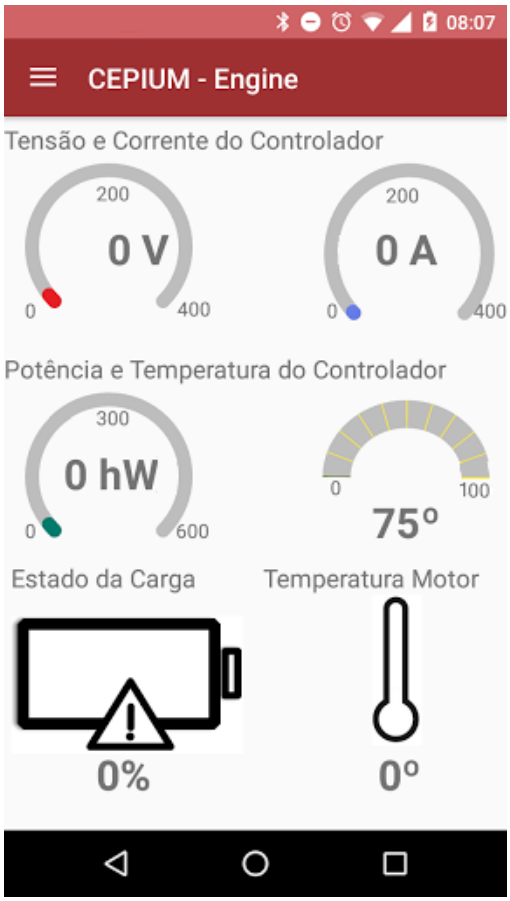


Figura 3.24 - *Fragment* responsável por apresentar os dados do sistema de tração.

3.5.2.2 Tempo

A *activity* identificada com o número 4 apresenta o estado do tempo. Dependendo da seleção do utilizador é possível consultar o tempo atual (Figura 3.25 à esquerda) ou a previsão do tempo dos próximos 3 dias (Figura 3.25 à direita).

O OpenWeatherMap foi o serviço *online* escolhido para fornecer os dados meteorológicos. A vantagem deste fornecedor *online* em relação as outras é a grande existência de documentação e o suporte de várias linguagens. O OpenWeatherMap fornece dados meteorológicos para mais de 200 mil cidades. Estes dados provêm de estações localizadas em aeroportos e estações meteorológicas oficiais, fornecendo dados atuais do estado do tempo e previsões. A API fornece os dados no formato JSON, XML e HTML. Para este trabalho de investigação escolheu-se o uso do formato JSON.

Esta *activity* é responsável por efetuar as *queries* corretas, com a devida localização do dispositivo, e processar os resultados. Tanto para enviar a *query* como obter a localização é necessário ter o Wi-Fi ou o serviço de dados móveis ativo.

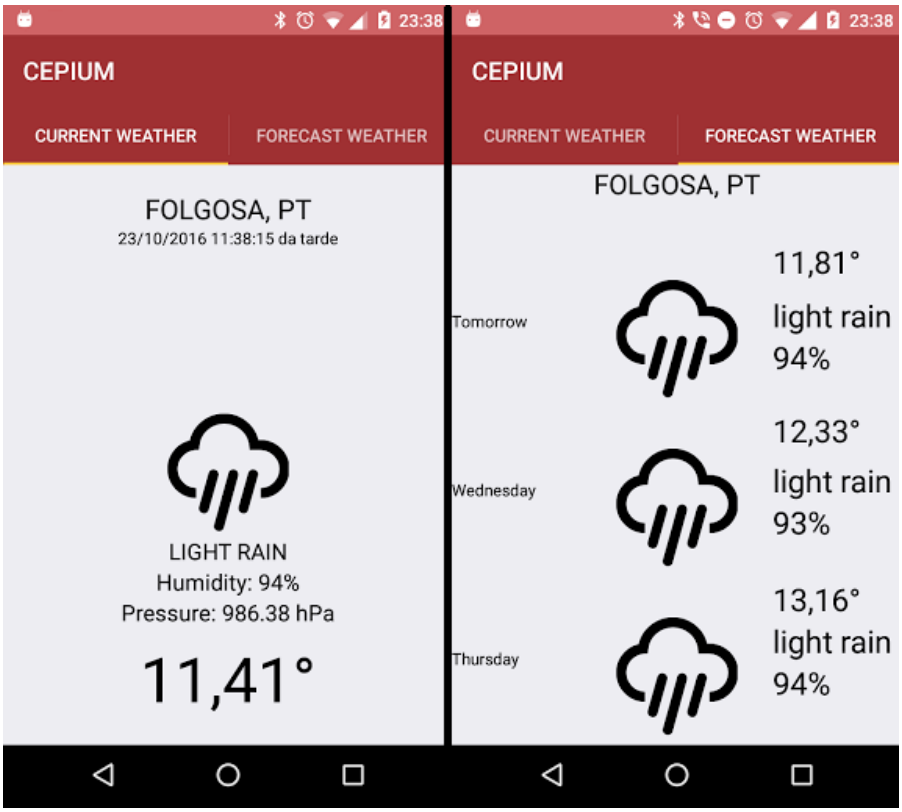


Figura 3.25 – *Activity* que mostra o estado do tempo.

3.5.3 Procura dispositivos

O Google disponibiliza a documentação, através do portal Android Developers, necessária para implementar os métodos de descoberta de dispositivos, estabelecimento de conexão e ativação da receção de dados através de notificações. O ficheiro *BluetoothLeService* é responsável por gerir estas funcionalidades.

Como já foi referido anteriormente, o *smartphone* desempenha o papel de uma estação central, como tal, o papel GATT que assume é o de cliente. Para utilizar o Bluetooth do próprio *smartphone* é necessário criar o objeto *BluetoothAdapter,* ficando possível executar as tarefas fundamentais. Para a descoberta de dispositivos

utiliza-se o método startLeScan() para iniciar a procurar de dispositivos com BLE ativo e stopLeScan() para parar esta procura. Por omissão na API do Google, este processo é interrompido ao fim de 4 segundos. Este intervalo de tempo é suficiente para encontrar os nós sensoriais. Os resultados da pesquisa são apresentados numa lista, com o nome do dispositivo e o seu endereço MAC (Figura 3.26).



Figura 3.26 – Exemplo de listagem de 3 dispositivos BLE encontrados.

Depois de encontrar os dispositivos BLE presentes no meio próximo, através de um toque do utilizador inicia-se o processo de estabelecer a comunicação entre a estação central e a estação periférica e dá-se início à terceira *activity* desta aplicação, explicada com mais detalhe na secção seguinte. A Figura 3.27 representa um fluxograma desta *activity*.

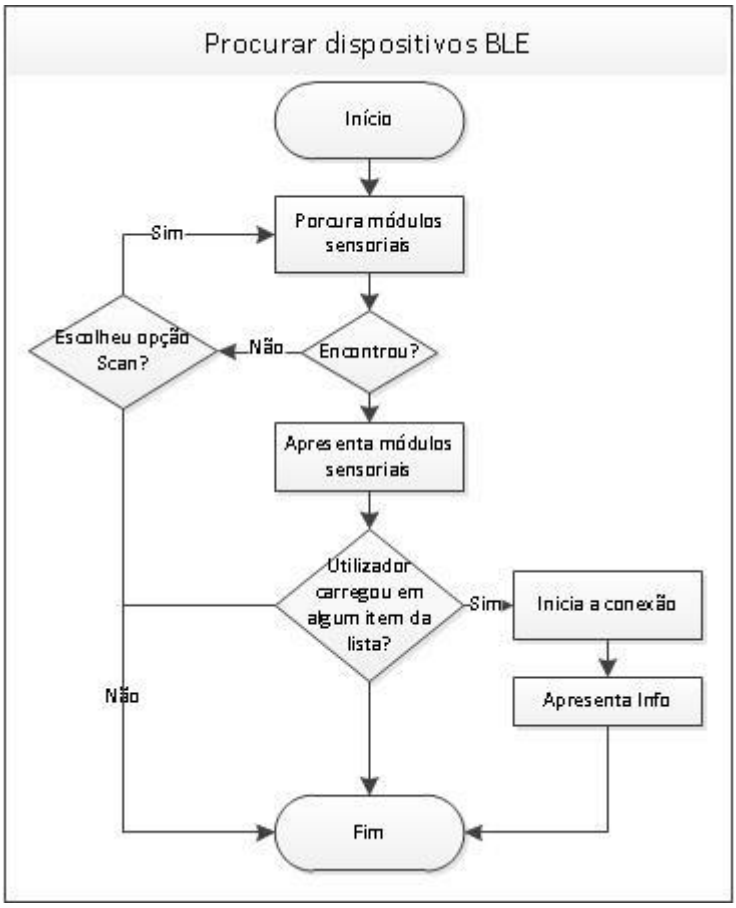


Figura 3.27 – Fluxograma da *activity* para procurar dispositivos BLE.

3.5.4 Informação dos sensores

Esta *activity* é responsável por mostrar os serviços e as respetivas características suportados pelo GATT servidor. Após uma conexão bem-sucedida, a informação dos serviços e características é apresentada ao utilizador. Dentre as características apresentadas existe uma específica em que é possível ativar a receção dos dados como notificações. Quando o utilizador seleciona esta característica, o método setCharacteristicNotification() é chamado. A API base, fornecida pelo Google, apresenta serviços já definidos que são essenciais para o funcionamento do Bluetooth.

Assim que as notificações são ativadas, os dados dos nós são enviados. Esta *activity* envia os dados para a classe DataHandler para desencapsular e guardar. É também nesta *activity* que se inicia o serviço para esta *activity* correr em background

e receber da aplicação os dados dos sensores em qualquer altura. A Figura 3.28 representa o fluxograma desta *activity*.

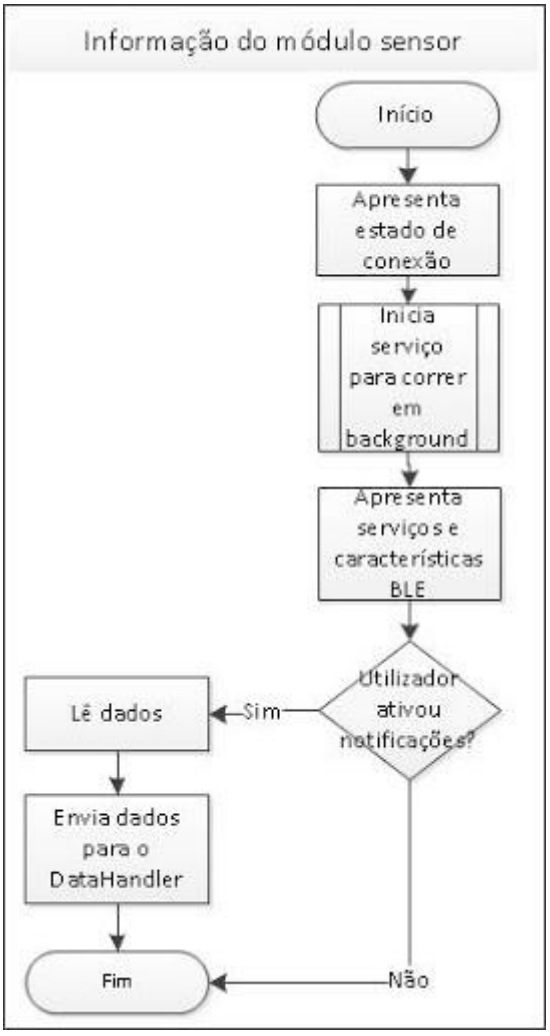


Figura 3.28 - Fluxograma da *activity* responsável por obter informação das estações periféricas.

4. Testes experimentais e resultados

Neste capítulo os testes e respetivos resultados obtidos ao longo desta dissertação são apresentados e discutidos. A implementação foi feita modularmente com o objetivo de facilitar a identificação de problemas e respetivas soluções. Desta forma são apresentados os principais testes, com a respetiva análise dos resultados. Primeiramente é apresentado o alcance da transmissão de dados com diferentes níveis de potência dos módulos CC2540. Os testes realizados destinam-se a ilustrar o comportamento das etapas apresentada no capítulo anterior. Sempre que foi necessário recorreu-se às ferramentas de *software* Packet Sniffer e Terminal para complementar os resultados.

4.1 Alcance e perdas em meio aberto

Estes testes foram realizados no campus da Universidade do Minho de Azurém, em campo aberto em linha de vista (Figura 4.1). O seu objetivo é determinar a relação entre a potência de transmissão e a distância entre as duas estações. A antena usada no teste foi a antena original do *kit* de desenvolvimento da Texas Instruments.

Para a realização deste teste a estação periférica enviava pacotes para a estação central, que media a intensidade do sinal recebido (RSS - Received Signal Strenght). O *firmware* desenvolvido foi carregado para ambas as estações. No entanto foi necessário fazer um recarregamento do *firmware* para a estação periférica sempre que foi alterado o nível da potência de sinal.

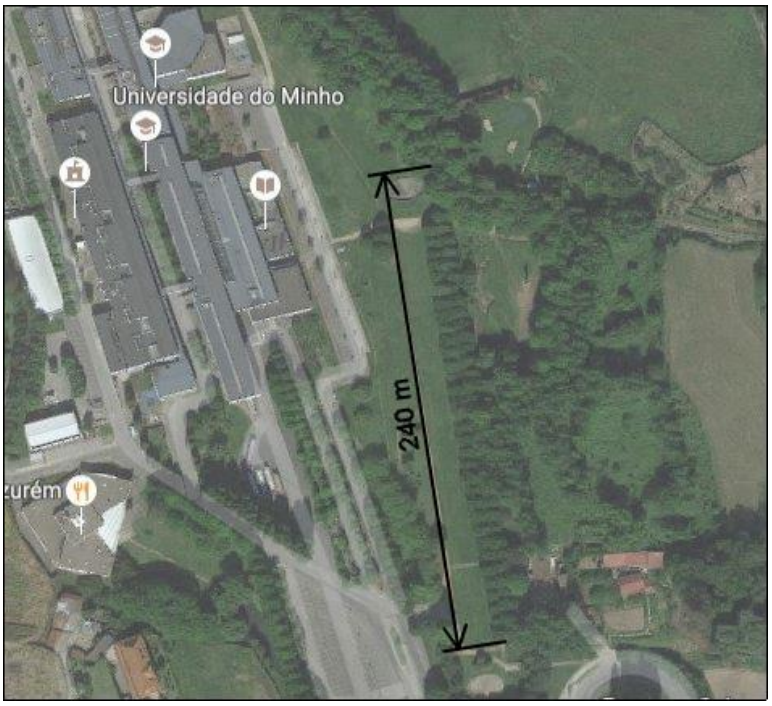


Figura 4.1 – Local onde se realizaram os testes.

Existem quatro níveis de potência que podem ser usados: -23 dBm, -6 dBm, 0 dBm e 4 dBm. Este último nível apenas se encontra disponível para os módulos CC2540, ao contrário dos módulos CC2541, que só aceitam os três primeiros. Para que o *firmware* altere corretamente o valor da sua potência foi necessário inserir a função HCI_EXT_SetT PowerCmd(power value) antes de enviar a notificação. Esta função aceita as quatro configurações mencionadas anteriormente como parâmetro. Assim que a conexão foi estabelecida, foram enviados 1000 pacotes, a cada 350 ms, ambas as placas se encontravam a 1 metro do chão. As tabelas abaixo e a Figura 4.2 apresentam os resultados obtidos durante os testes.

Tabela 4.1 – Intervalo de valores RSSI para uma potência de -23 dBm.

Distância (metros)	RSSI (dBm)
0	-41 a -39
5	-87 a -81
10	-94 a -85
15	-95 a -88

Tabela 4.2 – Intervalo de valores RSSI para uma potência de -6 dBm.

Distância (metros)	RSSI (dBm)
0	-34 a -32
5	-74 a -69
10	-76 a -71
15	-80 a -73

Tabela 4.3 – Intervalo de valores RSSI para uma potência de 0 dBm.

Distância (metros)	RSSI (dBm)
0	-34 a -32
5	-63 a -57
10	-70 a -63
15	-76 a -67

Tabela 4.4 – Intervalo de valores RSSI para uma potência de 4 dBm.

Distância (metros)	RSSI (dBm)
0	-34 a -30
5	-59 a -55
10	-72 a -58
15	-73 a -67

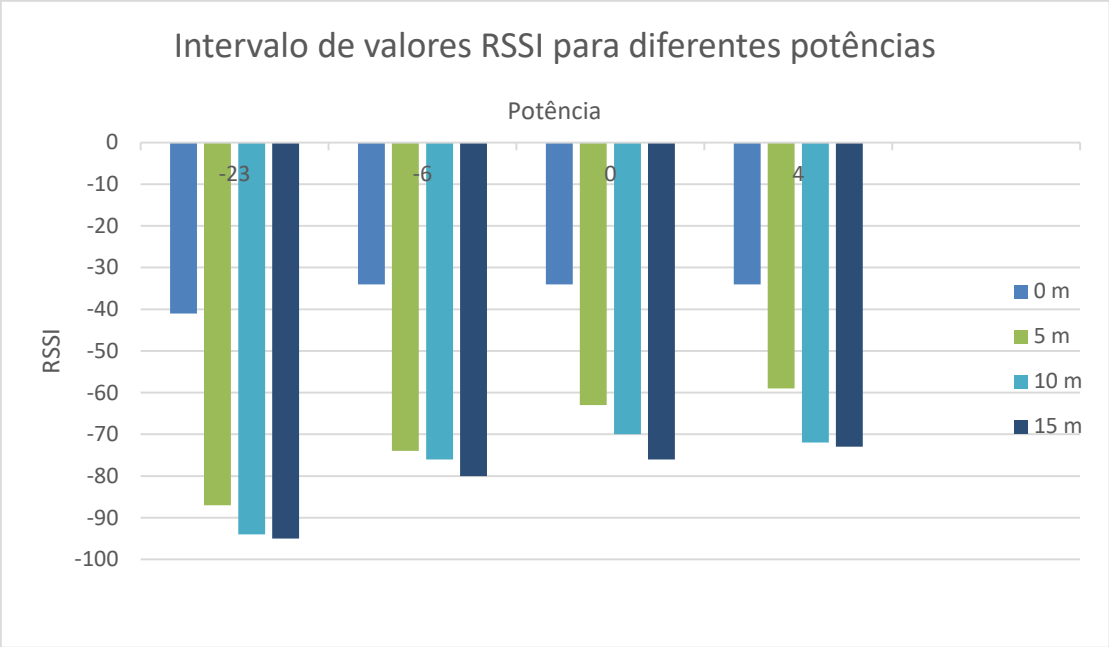


Figura 4.2 – Representação gráfica dos testes de alcance.

A distância máxima de transmissão até apresentar erros foi 33 m, com PLR de 5%, para -23 dBm, 158 m com 18% para -6 dBm, 170 m com 16% de perdas para 0 dBm e 235 m com 20% para 4 dBm.

Estes resultados estão sujeitos a vários fatores que podem causar alterações dos valores, como o meio em que se encontra, o estado do tempo e a carga das baterias.

4.2 Interferências e obstruções de sinal

Após a realização do teste anterior, testou-se a degradação da comunicação entre as estações quando colocadas no veículo. Uma vez que o metal é um material com grande potencial de obstrução devido ao efeito fotoelétrico e o motor de tração gera campos eletromagnéticos, foi necessário realizar este teste para verificar se estes fatores tinham algum efeito sobre o sistema. Realizaram-se alguns testes no veículo, com potência de transmissão de -23 dBm, colocando a placa periférica em diferentes localizações e mantendo a placa central num único local. A estação central foi pousada no interior do carro próximo do condutor, no *tablier* e começou-se por colocar a estação periférica junto ao motor do veículo, com o *capot* fechado. O teste decorreu

enquanto havia uma certa aceleração do motor, no entanto não se verificou perdas de pacotes na comunicação entre as duas placas. De seguida, alterou-se a localização da placa periférica para junto da bateria. Procedendo-se da mesma forma, voltou-se a verificar que não ocorreram perdas de pacotes. Repetiu-se este procedimento mais duas vezes: para a localização das placas nas luzes dianteiras e traseiras e para a localização das placas nos pneus do CEPIUM, sempre sem ocorrência de perdas de pacotes. Uma vez que não se verificou obstrução do sinal para este nível de potência, não foi necessário verificar os níveis mais altos.

Após a conclusão que o metal do veículo não desempenhava um papel de obstrução na comunicação, foi necessário verificar se as ondas eletromagnéticas existentes num ambiente interferem com a comunicação. Posto isto, utilizou-se um forno de micro-ondas como fonte de interferência para gerar ondas eletromagnéticas com na banda de frequências de 2,4 GHz. Este teste foi realizado num local fechado, demonstrado na Figura 4.3. As duas estações foram colocadas a uma distância aproximadamente de 10 m e entre elas existiam alguns obstáculos (Figura 4.3).

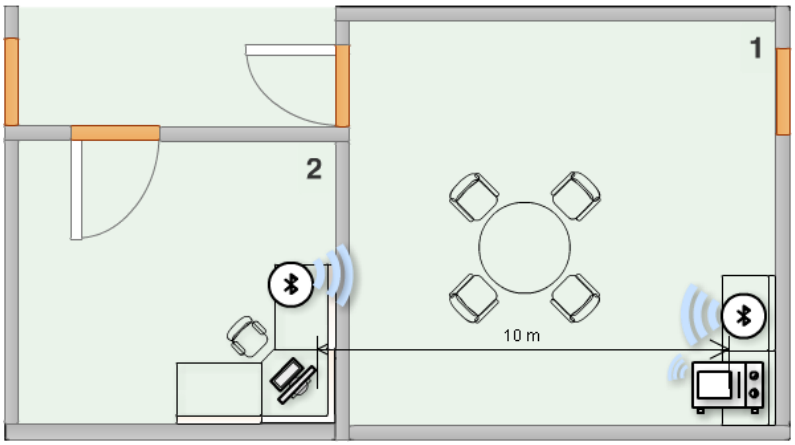


Figura 4.3 – Localização das estações para o teste de interferências.

A estação periférica foi colocada na sala designada como 1, junto à fonte emissora de ondas, que pode provocar interferência na comunicação, enquanto a estação central se encontrava na sala 2.

Este teste foi efetuado com o método de transmissão por notificações e a potência escolhida para este teste foi o nível menor permitido pelos módulos da TI, ou seja, -23 dBm. Neste cenário, obteve-se um PLR de 37%. Como se obteve um PLR com um valor muito alto, realizou-se de novo o teste. A potência de transmissão escolhida foi -6 dBm. Neste cenário, obteve-se um PLR de 0%. Uma vez que se obteve um resultado satisfatório e que provou ser robusto, não foi necessário verificar a interferência nas potências mais altas, uma vez que o baixo consumo energético é um requisito a ter em atenção.

4.3 Recolha de dados do sistema da bateria

Para estes testes recorreu-se à emulação dos dados porque na altura da entrega da dissertação, o envio de tramas, pelo sistema da bateria desenvolvido pelo GEPE, via interface RS-232, ainda não havia sido implementado. Sendo assim, a aplicação desenvolvida nesta dissertação foi submetida a testes de recolha de dados emulados do sistema da bateria através do *software* Terminal. Os testes foram realizados de acordo com a Figura 4.4.

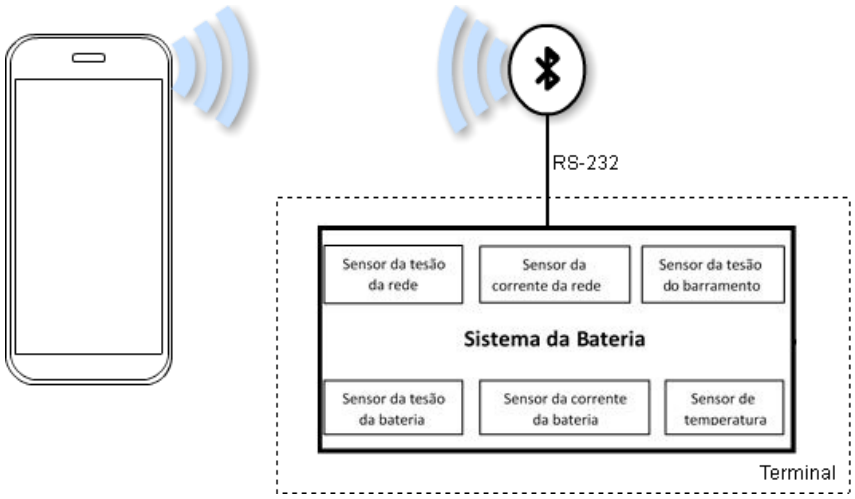


Figura 4.4 – Esquema utilizado para os testes com o sistema da bateria.

O sistema da bateria envia os dados para a placa periférica pela sua interface RS-232, que por sua vez os encaminha via BLE, após o processamento dos dados referidos na secção 3.4, para o *smartphone*.

O GEPE forneceu uma tabela (Tabela 4.5) que apresenta os valores máximos, mínimos e nominais dos sensores que fazem parte do sistema sensorial descrito. Estes valores foram utilizados para compor a trama apresentada na Figura 3.15 e emular através do Terminal a comunicação entre a placa periférica e o sistema da bateria.

Tabela 4.5 – Especificação dos valores dos sensores do sistema da bateria.

Sensor	Mínimo	Nominal	Máximo	Unidade
Tensão da Rede	210	230	240	V
Corrente da Rede	0	16	20	A
Tensão de Barramento	0	430	450	V
Tensão Baterias	0	300	360	V
Corrente Baterias	0	10	13	A
Temperatura	0		100	

A Figura 4.5 apresenta os diferentes dados recebidos pela aplicação desenvolvida.

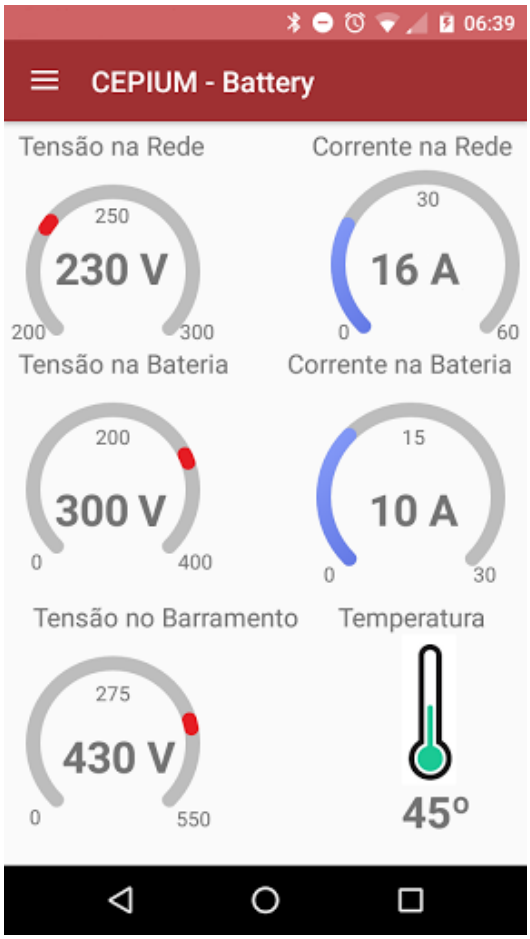


Figura 4.5 – Aplicação Android com dados emulados recolhidos do sistema da bateria.

4.4 Comunicação com múltiplos módulos sensoriais

O sistema desenvolvido tem em vista o uso de vários módulos sensoriais num veículo, em simultâneo, sendo esta, a motivação que levou a cabo a concretização destes testes. Conectaram-se duas placas periféricas via BLE a um *smartphone*, que desempenha o papel de estação central. Os dados transmitidos são recebidos, em ambas as estações, através da interface RS-232. A Figura 4.6 representa um esquema geral da realização dos testes.

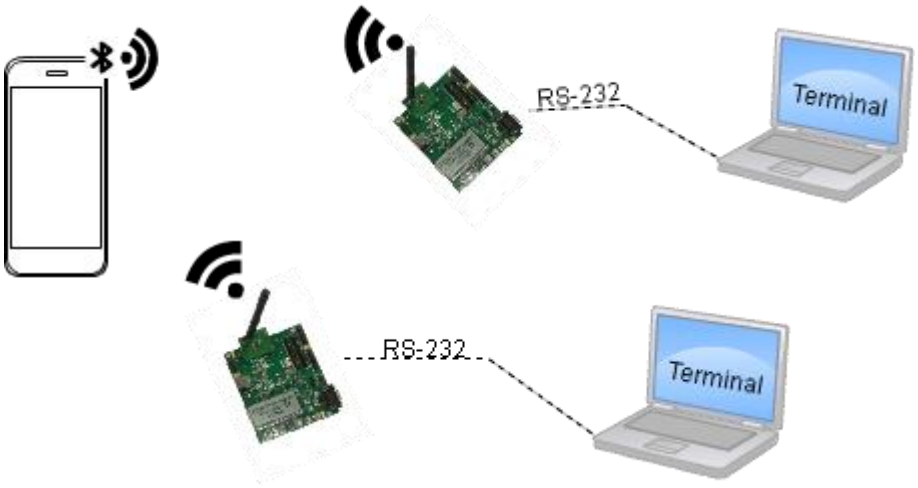


Figura 4.6 - Esquema como ocorreram os testes com múltiplos módulos sensoriais.

Para a realização destes testes voltaram-se a usar os valores fornecidos pelo GEPE, baseados na tabela anteriormente descrita (Tabela 4.5) e na Tabela 4.6, e o programa Terminal. Embora o sistema de tração não se encontre ainda implementado, usou-se assim os seus dados para se distinguir a comunicação com as duas estações periféricas.

Tabela 4.6 - Especificação dos valores dos sensores do sistema de tração.

Sensor	Mínimo	Máximo	Unidade
Estado Controlador	0	3	-
Corrente do Controlador	0	300	A
Tensão do Controlador	0	300	V
Estado de Carga	0	100	%
Potência do Controlador	0	500	hW
Temperatura Controlador	0	100	$°C$

A Figura 4.7 mostra o resultado da comunicação com dois módulos sensoriais. Label “I” identifica o envio de uma mensagem de informação e a label “C1” identifica a sua classe. Os bits assinalados por um círculo são os identificadores (endereços) dos diferentes sistemas, a sua interpretação pode ser vista na Tabela 3.3 da secção 3.4.

```
I/C1:: 01 9C 00 02 14 00 E6 00 10 00 AE 01 2C 01 0A 00 2D 00 00 00
I/C1:: 02 B6 00 02 14 00 03 00 2C 01 2C 01 64 00 F4 01 64 00 00 00
```

Figura 4.7 - Dados recebidos pela estação central.

A Figura abaixo demonstra a aparência gráfica da janela do sistema de tração da aplicação Android com os dados recebidos.

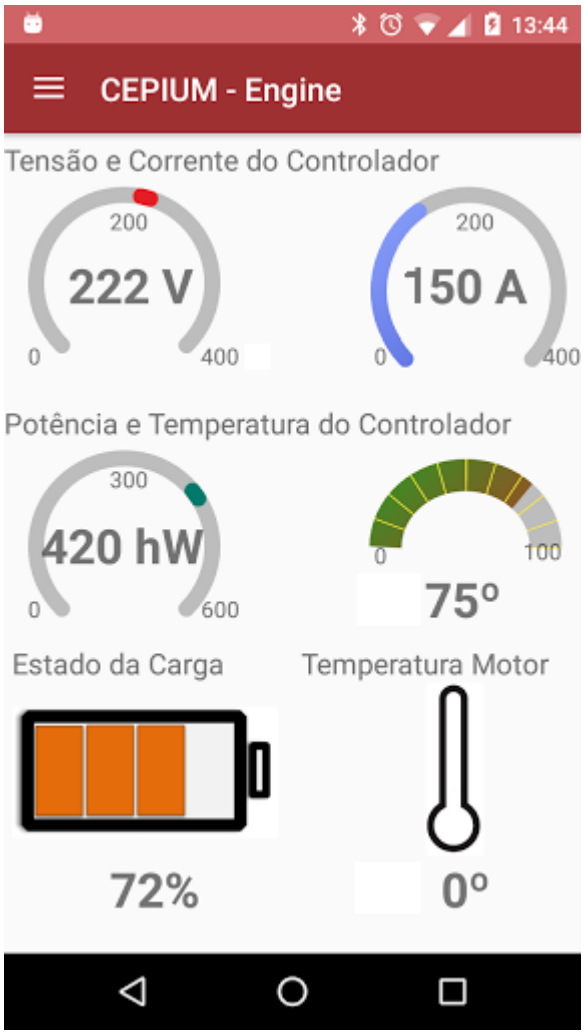


Figura 4.8 - Aplicação Android com dados emulados recolhidos do sistema de tração.

4.5 Comparação notificações/indicações

Como já foi referido no capítulo 2 existem dois métodos de comunicação no BLE: notificações e indicações. Estes testes foram feitos com o objetivo de determinar um período mínimo adequado para o evento periódico de envio de pacotes pela estação

periférica de modo a que não ocorram perdas de pacotes devido a erros na *stack* BLE dos módulos da TI. Foram realizados testes a duas distâncias diferentes: 1 m e 10 m, para diferentes valores da potência de transmissão e para os dois métodos de comunicação.

4.5.1 Teste a 1m

O primeiro teste, com o objetivo de quantificar as limitações causadas pela *stack* da Texas Instruments, foi executado num ambiente sem interferências, para não haver outros fatores que possam causar perdas de pacotes. No total, foram enviados 1000 pacotes com 20 bytes de *payload* (valor máximo permitido pelo BLE). As Tabelas de 4.7 a 4.10 apresentam os resultados obtidos enviando apenas uma notificação por intervalo de conexão (IC). A Tabela 4.7 com uma potência de -23 dBm, a Tabela 4.8 para -6 dBm, a Tabela 4.9 para a potência 0 dBm e, finalmente, a Tabela 4.10 para 4 dBm.

As tabelas apresentadas mostram os valores experimentais enviando apenas 1 notificação por conexão. Verifica-se que não ocorrem perdas a partir dos 10 ms, independente da potência de transmissão, mas em nenhum caso foi possível o envio de pacotes sem erros usando o IC mínimo especificado pelo Bluetooth [4]. Assim, estes testes permitiram concluir que existem limitações em termos de hardware/software dos módulos usados.

O PLR resulta da divisão entre o número de pacotes perdidos ($N_{perdidos}$) e o número total de pacotes enviados ($N_{pacotesTotal}$):

$$PLR = \frac{N_{perdidos}}{N_{pacotesTotal}} \times 100\% \quad (5)$$

Tabela 4.7 - PLR com notificações para uma potência de -23 dBm.

Intervalo de Conexão (ms)	Notificações (PLR)
7,5	78%
10	0%

Tabela 4.8 - PLR com notificações para uma potência de -6 dBm.

Intervalo de Conexão (ms)	Notificações (PLR)
7,5	78%
10	0%

Tabela 4.9 - PLR com notificações para uma potência de 0 dBm.

Intervalo de Conexão (ms)	Notificações (PLR)
7,5	78%
10	0%

Tabela 4.10 - PLR com notificações para uma potência de 4 dBm.

Intervalo de Conexão (ms)	Notificações (PLR)
7,5	78%
10	0%

Na segunda parte deste teste, repetiu-se o processo, mas usando o método das indicações. Neste teste, para além de se considerar o IC, foi necessário ter em consideração o período do evento periódico (PEP) com que se envia cada novo pacote, que pode ser maior que o IC. Por cada pacote enviado é necessário esperar pela receção do ACK, que só ocorre no evento de conexão seguinte. Caso este ACK não seja recebido, a aplicação da estação periférica bloqueia e não permite o envio de um novo pacote no intervalo de conexão seguinte, provocando a perda do novo pacote. Sendo assim, é necessário fazer PEP > 2 IC para possibilitar retransmissões antes de um novo

evento periódico. A Tabela 4.11 demonstra que o funcionamento com indicações não é correto quando PEP = IC.

Tabela 4.11 - PLR das indicações quando o PEP = IC.

Intervalo de Conexão (ms)	Evento Periódico (ms)	Indicações (PLR)
7,5	7,5	55%
10	10	53%
15	15	53%
20	20	53%
25	25	53%

As Tabelas 4.12 e 4.13 mostram os resultados obtidos para -23 dBm e -6 dBm, respetivamente, quando o PEP assume valores iguais ao dobro do valor IC. Por exemplo, caso o IC assuma um valor de 7,5 ms, o valor do evento periódico será de 15 ms. Para as potências de transmissão mais elevadas (0 dBm e 4 dBm) obtiveram-se os mesmos resultados.

Tabela 4.12 - PLR das indicações para uma potência de -23 dBm, quando o PEP = 2 IC.

Intervalo de Conexão (ms)	Evento Periódico (ms)	Indicações (PLR)
7,5	15	25%
10	20	17%
15	30	11%
20	40	7%
25	50	7%
50	100	3%
60	120	3%
100	200	0%

Tabela 4.13 - PLR das indicações para uma potência de -6 dBm, quando o PEP = 2 IC.

Intervalo de Conexão (ms)	Evento Periódico (ms)	Indicações (PLR)
7,5	15	25%
10	20	17%
15	30	11%
20	40	7%
25	50	7%
50	100	3%
60	120	3%
100	200	0%

Pelas tabelas anteriores pode-se verificar que, apesar de teoricamente o valor de EP = 2 IC seria suficiente, mesmo assim a *stack* ainda introduz erros com valores de IC baixos. Conclui-se que o valor do IC mínimo que é seguro utilizar é a partir de 100 ms, valor que vai de encontro ao recomendado nas especificações do Bluetooth. Consequentemente, verifica-se que o débito máximo possível com indicações é muito menor do que com notificações, pois o período mínimo recomendado para o evento periódico é muito maior, e além disso apenas é possível enviar uma indicação por IC, enquanto nas notificações é possível enviar 3, como foi verificado em [26].

A Figura 4.9 ilustra um exemplo de uma troca de pacotes entre um mestre e um escravo com PEP = 2 IC, caso ocorra uma perda. Após o escravo transmitir o pacote (D1), fica à espera da receção de uma confirmação, bloqueando todas as outras transmissões. No final de cada IC, efetua a retransmissão do pacote até receber a confirmação esperada. Com PEP = 2 IC há espaço para uma retransmissão, caso seja necessário fazer mais do que uma retransmissão por pacote é recomendável aumentar o PEP para que o envio do pacote seguinte (D2) não seja bloqueado.

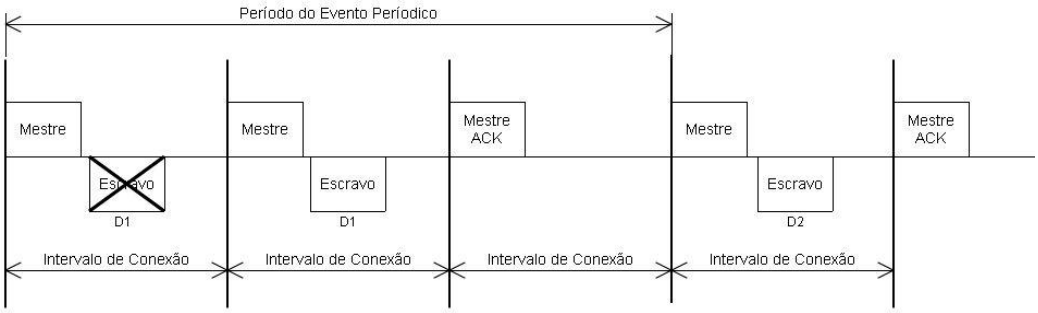


Figura 4.9 - Troca de dados entre o mestre e escravo com o método das indicações, caso ocorra uma perda.

A Figura 4.10 representa um exemplo de retransmissões que ocorreram durante os testes. Como se pode observar os dois pacotes captados pelo software SmartRF Packet Sniffer apresentam o mesmo número de sequência (assinalado com um círculo).

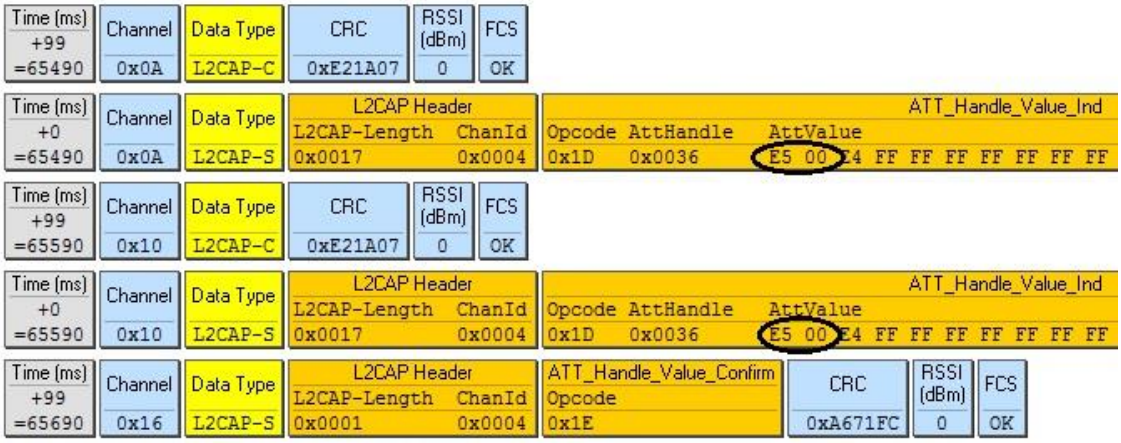


Figura 4.10 - Exemplo de retransmissões pelo método de comunicação por indicações.

Comparando os dois métodos de comunicação conclui-se que o intervalo de conexão mínimo recomendado para as notificações é de 10 ms, enquanto que para as indicações é de 100 ms. Para além disto, verifica-se que a potência de transmissão não tem qualquer influência em comunicações de curto alcance.

4.5.2 Teste a 10 m

O segundo teste tem como objetivo encontrar os valores mínimos da potência de transmissão e do PEP para ocorrer uma comunicação sem erros com o IC mínimo determinado no teste anterior num cenário em que os módulos foram colocados a

uma distância de 10 m e com obstruções. O local escolhido para a realização deste teste foi o mesmo que se encontra descrito na Figura 4.3, exceto pela não utilização do forno de micro-ondas. Assim como no teste anterior foram enviados 1000 pacotes com 20 bytes de *payload* e uma notificação/indicação por evento periódico.

Os testes com notificações foram feitos com PEP = IC. Como pode ser visto nas tabelas abaixo, o IC mínimo escolhido para este teste foi de 25 ms, devido ao facto de não ter sido possível estabelecer uma conexão estável para IC menores que 25 ms. Isso significa que o IC mínimo recomendado também varia com a distância.

Tabela 4.14 - PLR das notificações para uma potência de -23 dBm.

Evento Periódico (ms)	Notificações (PLR)
25	100%
33,75	88%
50	76%
60	50%
80	48%
100	43%

Tabela 4.15 - PLR das notificações para uma potência de -6 dBm.

Evento Periódico (ms)	Notificações (PLR)
25	30%
33,75	0%

Tabela 4.16 - PLR das notificações para uma potência de 0 dBm.

Evento Periódico (ms)	Notificações (PLR)
25	0%

Este teste permitiu concluir que o valor da potência de transmissão tem grande influência nos resultados obtidos e que a potência de -23 dBm torna a

comunicação pouco fiável para este método. Já para -6 dBm, conseguiu-se uma transmissão fiável com o PEP a partir de 33,75 ms, e com 0 dBm a partir de 25 ms.

Na segunda parte deste teste, repetiu-se o processo, mas usando o método das indicações. O IC utilizado foi de 100 ms como determinado no teste a 1m.Tabela 4.17 - PLR das indicações para uma potência de -23 dBm.

Evento Periódico (ms)	Indicações (PLR)
200	60%
350	50%
500	21%
800	11%
1000	0%

Tabela 4.18 - PLR das indicações para uma potência de -6 dBm.

Evento Periódico (ms)	Indicações (PLR)
200	9%
350	0%

Tabela 4.19 - PLR das indicações para uma potência de 0 dBm.

Evento Periódico (ms)	Indicações (PLR)
200	0%

Tal como para as notificações, a potência de transmissão teve grande influência nos resultados obtidos. Para -23 dBm, o PEP mínimo adequado foi de 1000 ms, enquanto para -6 dBm não se obtiveram erros a partir de 350 ms. Já para a potência de 0 dBm não se verificaram erros para um PEP igual ou superior a 200 ms (o dobro do IC).

5. Conclusões e trabalho futuro

Neste último capítulo da dissertação são apresentadas as conclusões da mesma, de acordo com o trabalho realizado. Além disso são apresentadas sugestões para melhorar e expandir o trabalho realizado no futuro.

5.1 Conclusões

Cada vez mais se incorporam dispositivos eletrónicos no nosso dia-a-dia para aumentar a nossa produtividade, organização ou mesmo melhorar a qualidade de vida, em áreas como o desporto, saúde ou domótica. Esta procura aumentou o desenvolvimento de sistemas baseado em redes de sensores sem fios, pois são uma solução que permite uma maior mobilidade. No âmbito desta dissertação foi realizado um estudo do estado da arte sobre as WSNs tendo em conta uma aplicação intra-veicular, nas redes IEEE 802.15.4/ZigBee e Bluetooth, pois são tecnologias de baixo consumo de energia. Esta última tecnologia foi descrita nesta dissertação, bem como a sua versão mais recente, Bluetooth Low Energy. Para além disto foram apresentados fundamentos teóricos sobre redes intra-veiculares e a plataforma Android.

O principal desafio deste trabalho de dissertação visava a integração, no carro elétrico CEPIUM do GEPE, de módulos sensoriais CC2540 e um *smartphone,* de modo a avaliar essa possível integração no contexto de uma rede intra-veicular, uma vez que esta requer fiabilidade na transmissão de dados. Mais especificamente, esta dissertação tinha como principais objetivos: implementar e avaliar o desempenho da rede de comunicações sem fios Bluetooth Low Energy em laboratório e no veículo; desenvolver e realizar testes do código para a recolha dos dados de sensores, obtidos através dos módulos BLE instalados no veículo; e desenvolver uma interface gráfica para um dispositivo móvel Android com a finalidade de apresentação dos dados. Apesar das dificuldades encontradas, todas os objetivos propostos foram alcançados.

Para o desenvolvimento, inicialmente, como já foi especificado, utilizou-se dois módulos CC2540 da Texas Instruments. Os testes efetuados nesta etapa contribuíram para o estudo do alcance da comunicação e a análise da fiabilidade em ambientes com interferências, não só com a presença de radiação eletromagnética da mesma gama de frequências, mas também com obstáculos físicos, como as partes metálicas do veículo. As principais dificuldades nesta fase do projeto centraram-se na compreensão do protocolo BLE e do seu funcionamento. O estudo da especificação do *standard* BLE e da documentação do *kit* de desenvolvimento CC2540, da Texas Instruments, contribuiu bastante para ultrapassar estas dificuldades.

A segunda fase do trabalho consistiu em adquirir dados sensoriais por interface RS-232. Para o efeito implementou-se um protocolo de comunicação de modo a permitir a comunicação de dados bidirecional entre os módulos sensoriais e as placas periféricas.

Ultrapassada esta fase, iniciou-se o desenvolvimento da aplicação Android. A aplicação desenvolvida teve como objetivo ser a interface homem-máquina para acesso aos dados sensoriais do veículo. Neste contexto, o *smartphone* que executa esta aplicação assumiu o papel da estação central. Nesta fase, uma vez que não foi possível aceder aos dados dos sistemas desenvolvidos pelo GEPE, realizaram-se testes com dados emulados. Através dos valores indicados pelo GEPE para os dados dos sensores, foi possível emular a sua transmissão através do software Terminal. Um desafio desta fase de implementação foi a adaptação da linguagem de programação, mas o facto de ser baseada em Java facilitou a sua compreensão.

Esta dissertação foi sem dúvida um projeto desafiante pela variedade e profundidade de conhecimentos adquiridos no domínio das comunicações via BLE e no desenvolvimento de uma aplicação Android. Começando pela compreensão da arquitetura BLE e do serviço que permite o envio periódico das tramas entre mestre e escravo, passando pela análise dos dois métodos de comunicação permitidos pelo BLE e da implementação do protocolo de comunicação sobre a interface UART até à implementação da aplicação Android, tudo foi um enorme desafio com grande complexidade de conceitos.

Posto isto, concluiu-se que o BLE é uma tecnologia promissora e viável para as IVWSNs. As principais características do BLE, comparada com tecnologias de rede como o Wi-Fi, são o baixo consumo, baixa taxa de transferência de dados e pilha de *software* pequena. O facto de o BLE estar integrado na maioria dos dispositivos móveis atuais faz com que a sua utilização se torne preferencial neste contexto quando comparada com tecnologias como o ZigBee.

5.2 Trabalho futuro

Apesar de cumpridos os objetivos desta dissertação, existem funcionalidades e desenvolvimentos que podem expandir e melhorar o trabalho desenvolvido. A primeira sugestão está relacionada uma análise sobre o código Android desenvolvido, de modo a melhorar o desempenho computacional, minimizar o consumo de energia e incluir novas funcionalidades, como guardar dados da aplicação numa base de dados na Internet, memória externa ou interna.

A segunda sugestão está relacionada com a otimização do protocolo de comunicação sobre a interface UART. O envio dos dados deveria ser modificado para enviar pacotes de tamanho fixo, enviando os valores como inteiros e não caracteres. Assim era possível reduzir drasticamente o tamanho dos pacotes, deixando de ser necessário efetuar o processamento dos dados nas estações periféricas, para o envio via BLE, bastando apenas o seu encaminhamento para a estação central.

A última sugestão seria a colocação de outros tipos de sensores, em diferentes pontos do veículo, por exemplo: sensores nas lâmpadas de sinalização e iluminação para detetar avarias, um sensor de temperatura (já se encontra definido as configurações necessárias para a introdução de um destes sensores, sendo apenas necessário manipular o resultado para um valor percentual), ou sensores nos pneus do carro para controlar a sua pressão. Esta última localização do sensor é a mais complicada de se realizar uma vez que se tem de ter em conta a rotação das rodas.

Referências

[1] J. Lin, T. Talty, and O. Tonguz, “On the potential of Bluetooth Low Energy technology for vehicular applications,” *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 267–275, Jan. 2015.

[2] “Bluetooth Low Energy | Bluetooth Technology Website.” [Online]. Available: http://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy. [Accessed: 23-Nov-2015].

[3] Nordic Semiconductor, “Bluetooth Low Energy wireless technology backgrounder,” March 2011.

[4] Bluetooth Special Interest Group, “Bluetooth Core_V4.0,” 2010.

[5] “GEPE - Group of Energy and Power Electronics.” [Online]. Available: https://www.gepe.dei.uminho.pt/includes/cepium_en.html.

[6] Bluetooth Special Interest Group, “Bluetooth Core Version 3.0 + HS,” 2009.

[7] E. Mackensen, M. Lai, and T. M. Wendt, “Performance analysis of an Bluetooth Low Energy sensor system,” September 2012, pp. 62–66.

[8] M. Di Natale, H. Zeng, and P. G. A. Ghosal, “Understanding and Using the Controller Area Network Communication Protocol,” *Clim. Chang. 2013 - Phys. Sci. Basis*, vol. 53, no. 9, p. 223, 2013.

[9] P. Venda, “Controller Area Network - Fundamentos,” *Area*, p. 10, 2003.

[10] M. Ahmed, C. U. Saraydar, T. Elbatt, J. Yin, T. Talty, and M. Ames, “Intra-vehicular wireless networks,” *GLOBECOM - IEEE Glob. Telecommun. Conf.*, pp. 1–9, 2007.

[11] A. Developers, “Android, the world’s most popular mobile platform,” *Android, the world’s most popular mobile platform | Android Developers*, 2014. [Online].

Available: https://developer.android.com/about/android.html. [Accessed: 05-Jul-2016].

[12] A. Rubin, “Where’s my Gphone?” [Online]. Available: https://googleblog.blogspot.pt/2007/11/wheres-my-gphone.html.

[13] “Android now has 1 billion active users per month,” 2014. [Online]. Available: http://thenextweb.com/google/2014/06/25/google-activations-io-2014/. [Accessed: 05-Jul-2016].

[14] N. Gandhewar and R. Sheikh, “Google Android : An Emerging Software Platform For Mobile Devices,” *Int. J. Comput. Sci. Eng.*, no. 12, pp. 12–17, 2010.

[15] P. Torino, P. Torino, G. Macario, M. Torchiano, and M. Violante, “An In-Vehicle Infotainment Software Architecture Based on Google Android An In-Vehicle Infotainment Software Architecture Based on Google Android,” no. June 2015, pp. 257–260, 2009.

[16] Google Inc., “App Components.” [Online]. Available: https://developer.android.com/guide/components/index.html. [Accessed: 11-Ago-2016].

[17] Google, “UI Overview,” 2012. [Online]. Available: https://developer.android.com/guide/topics/ui/overview.html#Layout [Accessed: 11-Ago-2016].

[18] “App Manifest.” [Online]. Available: https://developer.android.com/guide/topics/manifest/manifest-intro.html [Accessed: 11-Ago-2016].

[19] H. M. Tsai, C. Saraydar, T. Talty, M. Ames, A. Macdonald, and O. K. Tonguz, “ZigBee-based intra-car wireless sensor network,” *IEEE Int. Conf. Commun.*, pp. 3965–3971, 2007.

[20] S. Kamath and J. Lindh, “Measuring Bluetooth® Low Energy Power Consumption,” *Texas Instruments*, pp. 1–24, 2012.

[21] J. R. Lin, T. Talty, and O. K. Tonguz, “An empirical performance study of Intra-

vehicular Wireless Sensor Networks under WiFi and Bluetooth interference," *GLOBECOM - IEEE Glob. Telecommun. Conf.*, pp. 581–586, 2013.

[22] J. A. Afonso, A. J. F. Maio, and R. Simoes, "Performance Evaluation of Bluetooth Low Energy for High Data Rate Body Area Networks," *Wirel. Pers. Commun.*, vol. 90, no. 1, pp. 121–141, 2016.

[23] J. A. Afonso, H. D. Silva, P. Macedo, and L. A. Rocha, "An Enhanced Reservation-Based MAC Protocol for IEEE 802.15.4 Networks," *Sensors*, vol. 11, no. 4, pp. 3852–3873, 2011.

[24] B. Yu, L. Xu, Y. Li, M. Collotta, G. Pau, Z.-M. Lin, C.-H. Chang, N.-K. Chou, and Y.-H. Lin, "Bluetooth Low Energy (BLE) based mobile electrocardiogram monitoring system," *2014 Int. Conf. Intell. Green Build. Smart Grid*, vol. 44, no. December 2009, pp. 137–152, 2014.

[25] M. Collotta and G. Pau, "Bluetooth for Internet of Things : A fuzzy approach to improve power management in smart homes q," *Comput. Electr. Eng.*, vol. 44, pp. 137–152, 2015.

[26] A. J. Maio, "Bluetooth Low Energy para Monitorização da Postura no Ciclismo," Universidade do Minho, 2014.

[27] "VOLKSWAGEN Polo 3 Doors - 1994, 1995, 1996, 1997, 1998, 1999 - autoevolution." [Online]. Available: http://www.autoevolution.com/cars/volkswagen-polo-3-doors-1994.html#aeng_volkswagen-polo-iii-3-door-1994-10 [Accessed: 08-Out-2016].

[28] T. I. Incorporated, "2.4-GHz Bluetooth ® low energy System-on-Chip," 2013.

[29] "CC2540 Development Kit - CC2540DK - TI Tool Folder." [Online]. Available: http://www.ti.com/tool/cc2540dk. [Accessed: 19-Feb-2016].

[30] L. Belbin and T. Commonwealth, "USER ' s Guide," no. JANUARY 1995, 2015.

[31] "IAR Embedded Workbench." [Online]. Available: https://www.iar.com/iar-embedded-

workbench/#!?architecture=8051&device=CC2540F128&currentTab=Features . [Accessed: 19-Feb-2016].

[32] Texas Instruments, “Texas Instruments CC2540/41 Bluetooth Low Energy Software Developer’s Guide,” pp. 1–50, 2012.

[33] “Install Android Studio | Android Studio.” [Online]. Available: https://developer.android.com/studio/install.html [Accessed: 05-Jul-2016].

[34] “Creating a Navigation Drawer | Android Developers.” [Online]. Available: https://developer.android.com/training/implementing-navigation/nav-drawer.html [Accessed: 23-Set-2016].

[35] C. Gomez, J. Oller, and J. Paradells, “Overview and evaluation of Bluetooth Low Energy: An emerging low-power wireless technology,” Sensors (Switzerland), vol. 12, no. 9, pp. 11734–11753, 2012.