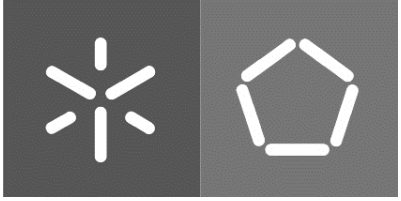

Universidade do Minho
Escola de Engenharia

António José de Freitas Maio

Bluetooth Low Energy para Monitorização da Postura no Ciclismo

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes
ao Grau de Mestre em Engenharia de Comunicações

Trabalho efetuado sob a orientação do
Professor Doutor José Augusto Afonso

outubro de 2014

DECLARAÇÃO

Nome: António José de Freitas Maio

Endereço eletrónico: antonio.fmaio@gmail.com

Telemóvel: 916111549

Número do Bilhete de Identidade: 13741796

Título dissertação**:** Bluetooth Low Energy para Monitorização da Postura no Ciclismo

Orientador: Professor Doutor José Augusto Afonso

Ano de conclusão: 2014

Designação do Mestrado: Ciclo de Estudos Integrados conducentes ao Grau de Mestre em Engenharia de Comunicações

Área de Especialização: Engenharia de Comunicações

Escola: Escola de Engenharia da Universidade do Minho

Universidade do Minho, ___/___/_____

Assinatura: _____

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao Professor Doutor José Augusto Afonso pela exímia orientação, dedicação, paciência e total disponibilidade, ao longo do desenvolvimento da dissertação.

A todos os professores, colegas, e amigos com quem me cruzei durante o meu percurso académico no Instituto Superior Técnico e na Universidade do Minho, o meu muito obrigado pelos conhecimentos transmitidos.

Ao Hélder Silva, Pedro Macedo, Diogo Francisco e colegas do laboratório pelo apoio prestado que contribuiu para o ultrapassar das dificuldades encontradas no conceção do projeto.

Uma menção especial para os meus amigos João Vilela, Joel Rodrigues e Paulo Alves, com quem partilhei de perto os melhores momentos ao longo destes 5 anos e que sempre me apoiaram e ajudaram nos momentos menos bons. Os meus sinceros agradecimentos pela vossa amizade.

Aos meus amigos do Judo, em especial ao meu treinador Mário Emídio, o meu muito obrigado pela compreensão em certos treinos, onde a minha concentração e dedicação não foram as melhores.

À Ana, um agradecimento especial, pela paciência, compreensão e motivação transmitida nos bons e nos maus momentos.

Por último, mas não menos importante, um profundo e sincero obrigado à minha família, com enfase aos meus pais e irmã, pois sem a eles este trajeto não seria possível. Agradeço a confiança e apoio prestado por todos em todas a fases do meu percurso académico, em especial para o realizar desta dissertação.

Resumo

A presente dissertação tem como finalidade analisar e aplicar o protocolo de comunicações sem fios BLE (Bluetooth Low Energy) à monitorização da postura de um atleta, durante a atividade de ciclismo. Um dos principais objetivos deste trabalho incide na implementação de uma infraestrutura de rede, baseada em BLE, que possibilite a integração de vários sensores. O sistema sem fios implementado garante maior simplicidade e flexibilidade de utilização no que diz respeito a uma mobilidade acrescida por parte de quem está a ser monitorizado, ao contrário do que acontece em sistemas cablados. Estes fatores fazem com que uma solução sem fios seja cada vez mais procurada no seio das áreas da saúde, desporto e entretenimento.

O sistema de monitorização de postura desenvolvido tem como finalidade obter as posições relativas a cada segmento corporal em que os nós estão colocados. O ângulo de inclinação do tronco do atleta, o ângulo de inclinação da bicicleta e o ângulo do joelho são os segmentos corporais alvos de análise. Os dados posicionais são obtidos periodicamente por meio de acelerómetros, magnetómetros e giroscópios e posteriormente enviados (via BLE) para um smartphone, que desempenha o papel de estação central. Na estação central existe uma aplicação, desenvolvida sob a plataforma Android, que recebe e interpreta os dados vindos dos diversos nós sensoriais e procede ao cálculo da postura nos segmentos corporais assinalados. Para a implementação do sistema, desenvolveu-se firmware para a comunicação periódica entre módulos BLE CC2540 da Texas Instruments; obteve-se os dados dos módulos sensoriais via protocolo SPI; desenvolveu-se uma aplicação Android capaz de receber a informação; e integrou-se o smartphone no sistema de monitorização de postura. Foi também realizada uma análise intensiva ao protocolo BLE e discutidos os resultados obtidos por via teórica e experimental.

Abstract

This thesis aims to analyze and apply the BLE (Bluetooth Low Energy) wireless protocol to monitor the posture of an athlete during the cycling activity. One of the main objectives of this work concentrates on the implementation of a network infrastructure based on BLE, which enables the integration of multiple sensors. The wireless system implemented ensures greater simplicity and flexibility of use with regard to increased mobility by the person who is being monitored, unlike what happens in wired systems. These factors make wireless solutions to be increasingly sought after within the areas of health, sport and entertainment.

The developed posture monitoring system aims to obtain the relative positions of each body segment in which we are placed. The tilt angle of the trunk of the athlete, the tilt angle of the bike and the knee angle are the body segments that are the target of analysis. Positional data are obtained periodically through accelerometers, magnetometers and gyroscopes and then sent (through BLE) for a smartphone, which plays the role of central station. At the central station there is an application, developed under the Android platform, which receives and interprets the data from the various sensor nodes in order to determine the posture in the monitored body segments. To implement the system, we developed firmware for periodic communication between CC2540 BLE modules, from Texas Instruments; we obtained the data of sensor modules via the SPI protocol; developed an Android application capable of receiving the information; and integrated the smartphone in the posture monitoring system. We also performed an intensive analysis of the BLE protocol and discuss the results obtained through theoretical and experimental evaluation.

Índice de conteúdos

Lista de figuras

Lista de tabelas

Lista de abreviaturas

8DPSK	8-ary Differential Phase Shift Keying
ADT	Android Development Tools
AMP	Alternate MAC/PHY
API	Application Programming Interface
ATT	Attribute Protocol
BAN	Body Area Network
BLE	Bluetooth Low Energy
BR	Basic Rate
CCC	Client Characteristic Configuration
COM	Communication Port
DK	Development Kit
DMP	Digital Motion Processor
EDR	Enhanced Data Rate
FDMA	Frequency Division Multiple Access
FHSS	Frequency Hopping Spread Spectrum
FSK	Frequency Shift Keying
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GFSK	Gaussian Frequency Shift Keying
I2C	Inter-Integrated Circuit

IEEE	Institute of Electrical and Electronics Engineers
LE	Low Energy
LOS	Line Of Sight
MAC	Medium Access Control
P2P	Peer To Peer
PAL	Protocol Adaptation Layer
PDU	Protocol Data Unit
PHY	Physical Layer
PSK	Phase Shift Keying
PUID	Personal User Interface Devices
QoS	Quality Of Service
RF	Radio Frequency
RFID	Radio Frequency Identification
SDK	Software Development Kit
SIG	Special Interest Group
SoC	System On Chip
SPI	Serial Protocol Interface
TDD	Time Division Multiplexing
TDMA	Time Division Multiple Access
TG	Task Group
UART	Universal Asynchronous Receiver/Transmitter
ULP	Ultra Low Power
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

UUID	Universal Unique Identifier
UWB	Ultra Wide Band
VLC	Visible Light Communication
VM	Virtual Machine
WBAN	Wireless Body Area Network
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WSN	Wireless Sensor Network
WPAN	Wireless Personal Area Network
WWAN	Wireless Wide Area Network

1. Introdução

1.1 Enquadramento e Motivação

A monitorização da postura de um indivíduo, aplicada ao desporto e à saúde, tem sido uma área de grande estudo e desenvolvimento no âmbito das redes de área corporal sem fios (WBAN – Wireless Body Area Network) [1]. Este tipo de redes tem a particularidade de requerer sensores de tamanhos reduzidos que possam ser convenientemente colocados no corpo de um paciente/atleta, e além disso, dada a aplicabilidade, requerem também baterias eficientes, que aliado ao baixo consumo por parte dos componentes possam contribuir para o aumento da autonomia e assim obter um melhor desempenho do sistema como um todo. A precisão e exatidão dos dados adquiridos são desafios importantes neste tipo de redes, na medida em que estes são de carácter sensível (e.g. sinais vitais de um individuo).

Numa WBAN (Figura 1.1) normalmente os dados recolhidos pelos sensores são enviados para uma estação central com capacidade para os coletar e realizar o respetivo tratamento. Os PCs começaram por ser a estação ideal, pois tinham grande poder de processamento e memória suficiente para tratar e armazenar os dados recolhidos. Com o passar do tempo e o avanço tecnológico, dispositivos pessoais de menor dimensão começaram também a assumir o papel de estação central, pois tinham a vantagem de apresentar uma maior portabilidade e continuar com o mesmo nível de processamento, como foi o caso dos computadores portáteis e PDAs. Atualmente os smartphones são dispositivos equipados com hardware que permitem elevado processamento de dados, memória de armazenamento e compatibilidade com protocolos de comunicação de curta distancia, como é o caso do Bluetooth Low Energy, apresentando assim uma grande vantagem para o objetivo final do sistema implementado nesta dissertação.

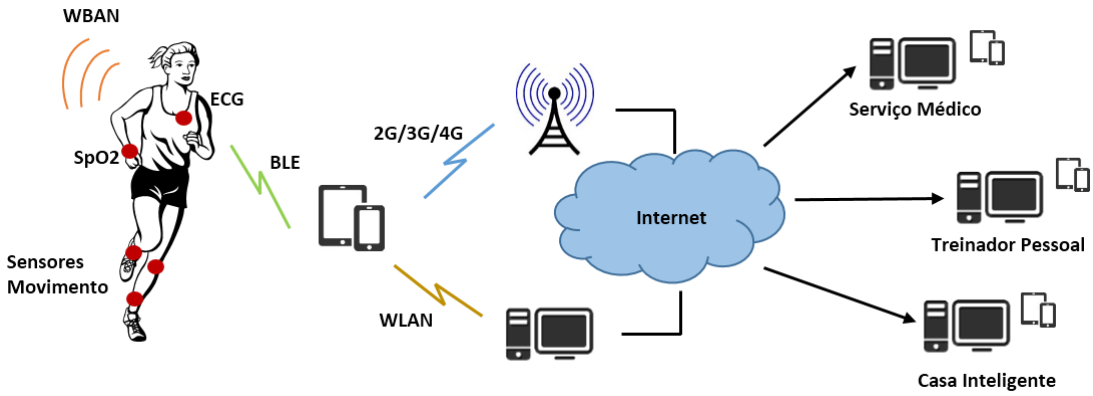

Figura 1.1 - Esquema geral de interação entre componentes de uma WBAN e de outras redes (adaptado de [2]).

A incorporação da plataforma Android no âmbito da dissertação vem contribuir para a flexibilidade do sistema, já que a estação central passa a ser um smartphone Android que recebe os dados lidos pelos nós da rede. A plataforma Android, estando em constante desenvolvimento e crescimento, aliado ao elevado poder de processamento de um smartphone onde está inserida, fornece ao programador uma panóplia de ferramentas para serem usadas em diversas aplicações para diversas áreas. Já um smartphone, sendo um equipamento de pequenas dimensões pode ser facilmente colocado em qualquer superfície, tornando fácil e cómoda a monitorização de uma WSN, por parte do utilizador.

Esta dissertação tem como finalidade contribuir para a monitorização e retificação da postura de atletas de ciclismo, já que o seu desempenho é fortemente influenciado pela mesma [3], ajudando-o assim a aumentar o controlo sobre a sua atividade física. Para o efeito, o ciclista terá sensores espalhados pelo corpo (tronco, membros superiores e/ou membros inferiores) e a informação irá ser monitorizada em tempo real e de forma eficaz recorrendo a um sistema de monitorização da postura, orientação e movimento do corpo baseado em módulos sensoriais desenvolvidos no Departamento de Eletrónica Industrial da Universidade do Minho [4]. Estes módulos são compostos por acelerómetros, magnetómetros e giroscópios, todos de três eixos.

Relativamente ao estabelecimento das comunicações sem fios, estas são baseadas em Bluetooth Low Energy (BLE). BLE é a versão mais recente da tecnologia

Bluetooth, que tem como principal característica ser uma tecnologia *wireless* de baixo consumo de energia (ULP – Ultra Low Power) na banda de frequência de 2,4 GHz. Está otimizada para ser usada em dispositivos que tenham que operar durante longos períodos de tempo sem que seja necessário substituir ou recarregar as suas baterias. Tempo de *standby* elevado (período em que o dispositivo está no estado de *sleep*), rápida conexão entre dispositivos e baixo consumo de energia na transmissão e receção de dados são características que tornam esta tecnologia de curto alcance numa ótima solução para ligações de proximidade entre dispositivos nas áreas da saúde, desporto e domótica, na medida em que fornece interoperabilidade com inúmeros dispositivos e oferece novas oportunidades a aplicações de comunicação de dados sem fios que haviam sido excluídas devido ao custo e gastos energéticos [5]. A análise do protocolo BLE foi assim efetuada tendo em conta duas implementações distintas, ou seja, a implementação do BLE em módulos da CC2540 da Texas Instruments e a implementação na plataforma Android.

Numa perspetiva mais técnica, como o hardware do sistema de monitorização já se encontra desenvolvido e devidamente testado, o âmbito desta dissertação será proceder à programação dos módulos BLE para aquisição dos dados recolhidos pelos sensores, e realizar a transmissão sem fios para um dispositivo móvel (smartphone), compatível também com a última versão do Bluetooth, onde estes serão processados.

1.2 Objetivos

Com esta dissertação pretende-se criar um sistema para avaliar a postura de um ciclista, recorrendo à recolha periódica dos dados lidos pelos sensores e enviando essa informação em tempo real via BLE para um smartphone. Neste dispositivo existirá uma aplicação Android que tratará os dados e mostrará os resultados. Deste modo, os objetivos são:

a) Implementação e testes de desempenho da rede de comunicação sem fios via BLE (Bluetooth Low Energy);

b) Implementação e testes do código de aquisição de dados dos módulos de monitorização da postura;
c) Integração de um smartphone Android no sistema para recolha e processamento de dados;
d) Testes finais e aprimoramento do sistema;

1.3 Estrutura da Dissertação

O presente documento está dividido em 6 capítulos.

No capítulo 1 é feita uma breve introdução ao tema desta dissertação. É abordada a motivação para a realização deste projeto bem como os objetivos propostos a alcançar. Por último está especificada a estrutura do documento.

O capítulo 2 apresenta uma visão geral dos conceitos e tecnologias usadas no âmbito desta dissertação. É feito o estado da arte das redes de área pessoal e das redes de sensores sem fios. É abordada a tecnologia de curto alcance Bluetooth, bem como a sua última vertente, o Bluetooth Low Energy. Por fim apresenta-se a plataforma de desenvolvimento Android e algum trabalho relacionado de índole aplicacional e protocolar.

No capítulo 3 é descrita em detalhe a implementação do sistema de monitorização de postura. 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 a nível aplicacional é também descrito neste capítulo. Por fim, a última secção deste capítulo diz respeito exclusivamente à aplicação Android desenvolvida, bem como a API usada e todas as funcionalidades a que o utilizador pode ter acesso.

O capítulo 4 contém uma análise detalhada do protocolo Bluetooth Low Energy no que diz respeito nomeadamente aos eventos de anúncio, eventos de conexão, latência, débito, consumo energético e número de escravos suportados na rede. É

feita também uma breve análise ao protocolo LPRT, frisando neste a capacidade da rede no que diz respeito também ao número de escravos suportados.

No capítulo 5 são apresentados os principais resultados obtidos, tendo em conta o desempenho do protocolo BLE em implementações com hardware e *stacks* distintas, ou seja, módulos da Texas Instruments e smartphone Android. As notificações por evento de conexão, a utilização do canal, a taxa de perda de pacotes, o consumo energético, o número de escravos suportados (BLE vs. LPRT), as colisões na transmissão e a postura do atleta são os parâmetros avaliados, teórica e experimentalmente, neste capítulo. Os resultados dos testes efetuados são devidamente acompanhados das respetivas discussões.

Por fim, o capítulo 6 contém as conclusões obtidas com o desenvolver do projeto e a indicação de possível trabalho futuro.

2. Estado da Arte

O presente capítulo aborda o estado atual dos temas principais que servem de base para esta dissertação. Em primeiro lugar, é feita uma introdução às redes de área pessoal, às redes de sensores sem fios e às redes de área corporal, no que diz respeito à sua arquitetura, características, requisitos e aplicações alvo. De seguida, é dado enfase ao Bluetooth como tecnologia de comunicação, onde são apresentadas as versões 3.0, 4.0, e a versão para baixo consumo, o Bluetooth Low Energy. Nesta última versão é apresentado em detalhe a constituição da pilha protocolar através da descrição das respetivas camadas. Posteriormente é introduzido o sistema operativo móvel Android ao nível arquitetural, aplicacional e o seu posicionamento no mercado. 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 Bluetooth Low Energy e outras tecnologias idênticas.

2.1 Redes de Área Pessoal

Um sistema de comunicação sem fios interliga, como o próprio nome indica, diversos equipamentos sem recorrer ao uso a cabos. As redes celulares móveis, bem como as redes de área local sem fios (WLAN - Wireless Local Area Networks) e de área pessoal sem fios (WPAN - Wireless Personal Area Networks) têm tido um forte impacto na sociedade, pois apresentam maior mobilidade e flexibilidade, uma rápida instalação e reestruturação das redes e sobretudo um menor custo quando comparadas com as tecnologias que recorrem ao uso de cabos [6].

As redes sem fios podem ser classificadas de acordo com a sua área de abrangência: redes pessoais ou de curta distância (WPAN - Wireless Personal Area Networks), redes locais (WLAN – Wireless Local Area Networks), redes metropolitanas (WMAN – Wireless Metropolitan Area Networks) e redes

geograficamente distribuídas ou de longa distância (WWAN – Wireless Wide Area Networks).

Entre as tecnologias de redes de comunicações sem fios, as WPAN são as que mais têm tido destaque nos últimos anos devido à grande revolução tecnológica de que se tem assistido. As redes sem fios de área pessoal (WPAN) são utilizadas para troca de informação entre dispositivos portáteis a distâncias relativamente pequenas e que requerem normalmente um gasto de energia reduzido. Ao contrário das redes de área local (WLAN), as WPAN não envolvem infraestruturas de rede muito complexas, sendo que o alcance das comunicações é claramente inferior ao das WLAN, pois o objetivo é cobrir a área em torno do utilizador e não servir múltiplos utilizadores. Cada vez existem mais dispositivos móveis de carácter pessoal que assumem um papel forte no dia-a-dia do utilizador, tornando a interligação entre dispositivos mais cómoda e menos dispendiosa, face aos dispositivos com fios, tornando assim possível desenvolver soluções mais económicas para um maior espectro de aparelhos.

Desde a última década que existe um grupo de trabalho do IEEE (Institute of Electrical and Electronics Engineers), denominado 802.15, a trabalhar na especificação e desenvolvimento de tecnologias de redes de área pessoal sem fios. Atualmente existem diversos subgrupos de trabalho (TG – Task Groups)[7], cada um deles focado em diferentes vertentes dentro do conceito geral das WPAN. O desenvolvimento contínuo de normas gerais para as tecnologias é deveras importante, pois garante a interoperabilidade de diversos dispositivos, reduzindo assim a complexidade e os custos para o utilizador final.

O TG 1 (WPAN/Bluetooth) foi formado com o intuito de desenvolver uma especificação para as redes de sensores sem fios, através do desenvolvimento de *standards* para as camadas física (PHY – Physical Layer) e de acesso ao meio (MAC – Medium Access Control), baseada na tecnologia inicial do Bluetooth (v1.1), tecnologia outrora desenvolvida por membros da indústria eletrónica e de telecomunicações.

O TG 2 (Coexistência) desenvolveu recomendações de forma a permitir a coexistência entre as redes de área pessoal sem fios e outras redes que operam em bandas de frequência na mesma área, tendo em vista a minimização da interferência mútua.

O TG 3 (WPAN de alto débito) visa adaptar as redes de área pessoal para aplicações que requerem altas taxas de débito (HR WPAN - High Rate WPAN), como processamento de imagem digital e multimédia em dispositivos portáteis, desenvolvendo um *standard* para as camadas PHY e MAC deste tipo de redes. Os requisitos de baixo custo e baixo consumo devem continuam a ser mantidos.

O TG 4 (WPAN de baixo débito) desenvolve *standards* para as camadas PHY e MAC para as redes WPAN de baixo débito (LR WPAN – Low Rate WPAN). O objetivo principal incide na redução da complexidade aumentando a longevidade da bateria dos dispositivos. As aplicações alvo incluem sensores, brinquedos interativos, comandos remotos e automação industrial. Este grupo foi dividido em subgrupos de forma a melhorar o *standard* original. Algumas melhorias observadas foram possibilidade de utilização de novas camadas físicas com suporte para frequências em conformidade com os regulamentos da China e Japão, o uso das tecnologias UWB (Ultra Wide Band) e Active RFID (Radio Frequency Identification), melhor suporte para os mercados industriais, melhoramentos na segurança e redução da complexidade do *standard* original.

O TG 5 (WPAN *mesh networking*) define as normas para a interoperabilidade, estabilidade e escalabilidade para as WPAN numa topologia em malha.

O TG 6 (Redes de área corporal) tem como objetivo desenvolver um *standard* para as redes de área corporal (BAN – Body Area Networks), garantindo um baixo consumo de energia para dispositivos que operam no interior, sobre ou em torno do corpo humano.

O TG 7 (VLC – Visible Light Comunication) está incumbido de desenvolver um *standard* para as camadas PHY e MAC em comunicações por meio de luz visível, de modo a suportar serviços multimédia de áudio e vídeo.

Como foi referido acima, o IEEE baseou-se na primeira versão do Bluetooth para desenvolver o *standard* IEEE 802.15.1 [8]. Esta tecnologia, especificada pelo Bluetooth SIG (Special Interest Group), mostrou enorme potencial sendo atualmente um dos exemplos mais comuns das WPAN. O Bluetooth SIG é formado por nomes sonantes do sector privado ligado às comunicações como Ericsson, Nokia, Motorola e após rutura com o IEEE continuou o desenvolvimento da tecnologia até chegar à versão 4.0, versão atual que inclui um novo sistema de comunicações com atenção acrescida para os gastos energéticos, o BLE. Este novo subconjunto da última versão do Bluetooth é útil para aplicações que requerem baixo débito de informação mas que possam atuar durante longos períodos de tempo sem necessidade de substituição de bateria, e apresenta a grande vantagem de estar incorporado na maioria dos smartphones atuais, o que não acontece com o ZigBee [9], que também é conhecido por ser uma tecnologia de baixo consumo energético. O aparecimento desta vertente tecnológica veio fomentar ainda mais o uso de sensores nas chamadas redes de sensores sem fios (WSN – Wireless Sensor Network). Nestas redes os sensores são normalmente usados em aplicações para monitorizar estados ou comportamentos e enviar essa pequena quantidade de informação para uma estação central onde irão ser tratados. Neste caso, a estação central pode ser perfeitamente um smartphone atual com suporte para BLE e com sistema operativo Android, pois este também é um sistema operativo móvel com grande abrangência no segmento dos dispositivos móveis e que disponibiliza a API para o desenvolvimento de aplicações BLE. Normalmente estas aplicações são desenhadas para atuar durante longos períodos de tempo, como por exemplo: estações meteorológicas, onde são usados sensores de temperatura; atividade física, onde são usados, por exemplo, sensores de ritmo cardíaco; e monitorização de postura e movimento, onde são usados sensores inerciais e magnéticos.

2.2 Redes de Sensores Sem Fios

Uma rede de sensores sem fios (WSN – Wireless Sensor Networks) é um tipo específico de rede sem fios formado por um conjunto de nós interligados entre si de

modo a medirem as gradezas físicas do meio em que estão inseridos (sensores) de forma distribuída ou até mesmo interferirem nesse mesmo meio (atuadores).

As principais características de uma rede de sensores sem fios são a baixa taxa de transmissão de dados, o baixo consumo de energia, a complexidade reduzida dos dispositivos, a escalabilidade e formação de redes *ad hoc*. Espera-se que uma rede de sensores sem fios seja concebida de forma a ser robusta, permitindo assim a tolerância a falhas; que a informação extraída seja fiável; que os dispositivos pertencentes à rede tenham autonomia suficiente para se manterem em funcionamento durante um período de tempo extenso; que seja escalável de modo a suportar um grande número de nós para garantir uma maior cobertura daquilo que se pretende monitorizar; e que seja flexível e facilmente adaptável a qualquer mudança que surja na rede.

A interação numa rede de sensores sem fios pode ser de diferentes tipos [10]. Através de medições periódicas, os dados podem ser lidos pelos sensores com uma certa periodicidade e reportados para a estação base ou outro nó. Um exemplo deste tipo de interação é um sistema de monitorização de postura. A deteção de eventos é outro tipo de interação possível dentro de uma WSN, onde um nó reporta informação para um outro nó sempre que um evento é despoletado. Por fim, através de pedidos de informação, um nó pode requisitar informação de outro nó acerca do estado da aplicação num determinado instante.

Uma das grandes preocupações e desafios aquando da conceção de uma rede de sensores sem fios é o consumo de energia e por isso várias medidas têm de ser tomadas de modo a encontrar uma solução viável. A maior parte da energia é gasta nas comunicações sem fio; por isso, além da escolha da tecnologia de transmissão a ser usada, deve-se aplicar uma potência de transmissão baixa de modo a conseguir uma poupança de energia. Por outro lado, quando um nó não necessitar de estar ativo quando não tem informação para recolher ou para enviar para outro nó, deve comutar para um estado de repouso, normalmente designado por modo "*sleep*", fazendo com que a eficiência a nível energético seja mais significativa. Devido a estes dois fatores, muitas vezes na conceção de uma WSN, opta-se por uma topologia

multihop, pois como o alcance das comunicações é reduzido devido à baixa potência de transmissão, usam-se nós intermediários para encaminhar os pacotes de informação entre o nó origem e o nó de destino.

Como referido anteriormente, a maior parte das interações numa WSN é através da recolha periódica ou esporádica dos valores lidos pelos sensores, pelo que o tráfego que circula numa rede deste tipo é muito menor do que numa rede de carácter convencional. Daí a taxa de transmissão dos dados (débito) ser também baixa, fazendo com que o principal objetivo de uma WSN, de modo geral, não seja obter altas taxas de transmissões de dados, mas sim grande eficiência ao nível energético.

Relativamente à sua arquitetura, um nó de uma rede de sensores sem fios pode ser visto como um conjunto de componentes simples e de tamanho extremamente reduzido de modo a satisfazer as necessidades e os princípios de uma WSN, como se pode observar na Figura 2.1. Existe uma unidade de processamento central, normalmente de baixa complexidade, com o objetivo de cumprir e suportar os requisitos mínimos da rede; uma memória limitada; um dispositivo capaz de efetuar a comunicação (rádio); sensores ou atuadores; e por fim uma bateria para alimentar todo o conjunto.

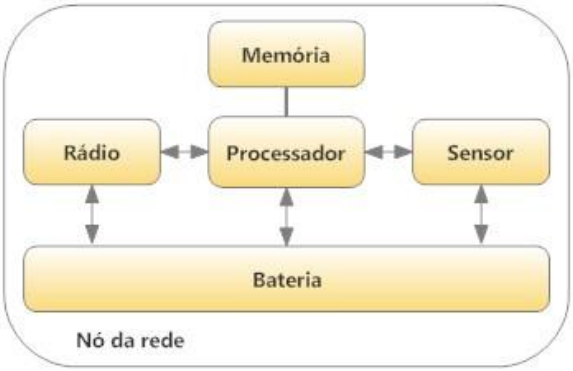

Figura 2.1 - Componentes de um nó de uma WSN (adaptado de [10]).

Atualmente os produtores deste tipo de dispositivos tendem a optar por uma abordagem SoC (SoC – System-on-chip), integrando múltiplos componentes num só tornando ainda mais reduzida a dimensão dos dispositivos das WSNs.

2.2.1 Redes de Área Corporal Sem Fios

Uma rede de área corporal sem fios (WBAN – Wireless Body Area Network) é um tipo específico de WSN onde diversos nós são colocados dentro, sobre ou em redor do ser humano com a finalidade de recolher informação acerca do mesmo. Os dados monitorizados são posteriormente enviados para uma estação central, e.g., um smartphone ou PC, onde são tratados e interpretados. Existem diversos sinais que podem ser monitorizados com recurso a uma WBAN como frequência cardíaca, nível de glicose no sangue, pressão sanguínea, movimentos corporais e postura, entre outros. A Figura 2.2 mostra um exemplo de uma WBAN, onde existem diversos sensores colocados num indivíduo a monitorizar diversos sinais vitais do mesmo. Os dados monitorizados estão a ser transferidos para um smartphone por meio de uma ligação sem fios.

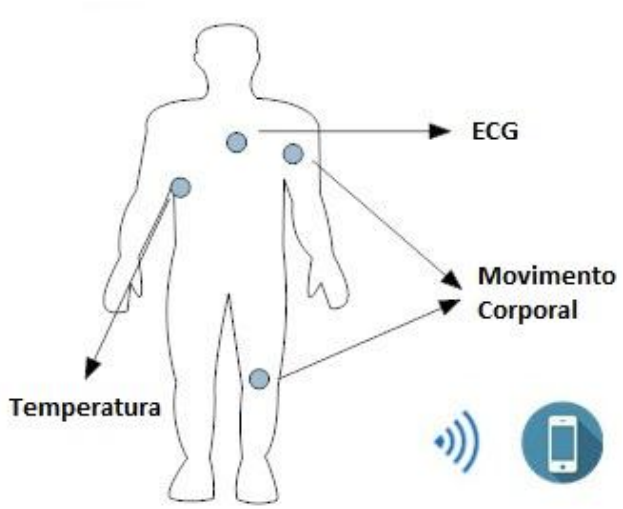

Figura 2.2 – Exemplo de uma WBAN.

Em comparação com os sistemas com fios, as WBANs proporcionam uma maior flexibilidade e conforto quando usadas no indivíduo, pois enquanto a monitorização está a ser feita não há praticamente interferência com outras atividades do dia-a-dia, contribuindo assim para o uso deste tipo de redes em áreas da saúde, reabilitação, desporto e entretenimento. Outro aspeto que também contribui para a facilidade de monitorização é a miniaturização dos nós da rede, exatamente para zelar pela mínima interferência possível no quotidiano do indivíduo.

Como foi referido, as WBANs são um tipo específico de WSN, derivando esta última do conceito WPAN já apresentado também. No entanto existem algumas diferenças aquando a conceção de WBANs relativamente às WSNs. As WSNs são geralmente implementadas num meio tipicamente estático e previsível enquanto as WBAN são implementadas num meio mais dinâmico, como é o caso do corpo humano, sujeito a movimentações mais repentinas e constantes. Apesar de uma WBAN ser desenhada para suportar comunicações a curtas distâncias, geralmente a área de corpo do ser humano, os fatores descritos acima podem dificultar a comunicação entre os dispositivos, acabando por afetar outros parâmetros importantes como a autonomia do sistema e a fiabilidade dos dados recolhidos, devido a um possível aumento da taxa de erros, e consequentemente provocar um decréscimo na eficiência da rede. A Tabela 2.1 apresenta resumidamente as principais diferenças entre uma WBAN e uma WSN.

Tabela 2.1 – Principais diferenças entre WSN e WBAN (adaptado de [11]).

	WSN	WBAN
Tamanho da rede	Metros/quilómetros (consoante a área a monitorizar)	Centímetros/milímetros (consoante o número de membros a monitorizar)
Nº de nós	Grande densidade de nós	Poucos nós mas de maior precisão
Função dos nós	Tarefa dedicada (homogéneos)	Tarefas diferentes (heterogéneos)
Transferência de dados	Perda de dados compensada pelo grande número de nós usados	Perda de dados crítica; necessidade de mecanismos de QoS

O sucesso crescente do uso de diversas tecnologias sem fios nesta área levou à criação, em 2007, do grupo de trabalhos IEEE 802.15.6, que como foi descrito anteriormente define um *standard* para a camada PHY e MAC das redes de sensores de área corporal. Este *standard* tem como principal foco proporcionar suporte de qualidade de serviço, baixo consumo de energia, alta fiabilidade de transmissão de dados e débitos até 10 Mbps [12]. Este *standard* tem ainda atenção ao efeito do corpo humano relativamente à proximidade das antenas dos dispositivos, a

modificação dos padrões de radiação para minimizar as taxas de absorção e a alteração das características de radiação devido aos movimentos do corpo.

2.3 Bluetooth

A tecnologia sem fios Bluetooth é um sistema de comunicação de curta distância pensado para substituir os cabos que ligam os diversos dispositivos fixos e móveis. As características da tecnologia Bluetooth são robustez, baixo consumo de energia e sobretudo de baixo custo para o utilizador final. A comunicação sem fios via Bluetooth tem a vantagem de não requerer uma linha de vista (LOS – Line of Sight) entre os dispositivos, o que não acontecia com o IrDA [13], tecnologia de comunicação infravermelhos mais usada antes do aparecimento do Bluetooth.

Tal como foi dito previamente, a versão 1.1 desta tecnologia serviu de base para o *standard* 802.15.1 devido ao seu grande sucesso. Seguiu-se a versão 2.0, em 2004, com a introdução de um mecanismo de melhoramento de débito, o EDR (Enhanced Data Rate). Em 2009 surgiu a versão 3.0 com um mecanismo HS (High Speed) que alegava taxas de transmissão de 24 Mbit/s, e mais recentemente a versão 4.0, que irá ser referida na secção 2.3.3. A versão utilizada para descrever a tecnologia Bluetooth nesta secção, é a versão 3.0 [14].

2.3.1 Camada Física

O Bluetooth opera na banda ISM licenciada dos 2.4 GHz possuindo um total de 79 frequências espaçadas de 1 MHz. De modo a combater as interferências utiliza um mecanismo de espalhamento espectral por saltos em frequência (FHSS – Frequency Hopping Spread Spectrum), alternando assim entre várias frequências definidas previamente de modo pseudoaleatório.

O *core* de um sistema Bluetooth consiste sempre numa entidade denominada por Host (anfitrião) e por um ou vários Controllers (controladores). O bloco Host é constituído por todas as camadas abaixo dos perfis de utilização e acima da interface

de HCI (Host Controller Interface). Já o bloco Controller compreende todas as camadas abaixo da camada HCI.

Na versão 3.0 do Bluetooth [14] existem dois tipos de controladores: BR/EDR (Basic Rate/ Enhanced Data Rate), que inclui as camadas rádio, *baseband*, *link manager* e HCI (opcionalmente); e AMP (Alternate MAC/PHY), que inclui AMP PAL (Protocol Adaptation Layer), AMP MAC, PAL (Protocol Adaptation Layer), e opcionalmente a HCI. O controlador BR/EDR já existia na versão anterior do Bluetooth, sendo que o controlador AMP foi a novidade introduzida nesta versão, acrescentando a função HS (High Speed). Este controlador é secundário, e permite que, após dois dispositivos efetuarem uma conexão via BR/EDR, se for encontrado o controlador AMP no outro dispositivo, o tráfego de dados possa ser movido do controlador BR/EDR para o controlador AMP e os dados serem transferidos via Wi-Fi. Basicamente, o controlador AMP utiliza a camada física e MAC do IEEE 802.11 para dar suporte à transmissão de grandes quantidades de tráfego.

A Figura 2.3 apresenta as possíveis arquiteturas da versão 3.0 do Bluetooth, combinando o bloco Host com o(s) bloco(s) Controller(s). Em a) existe apenas o controlador BR/EDR; em b) os controladores BR/EDR e AMP; e em c) o controlador BR/EDR com múltiplos controladores AMP.

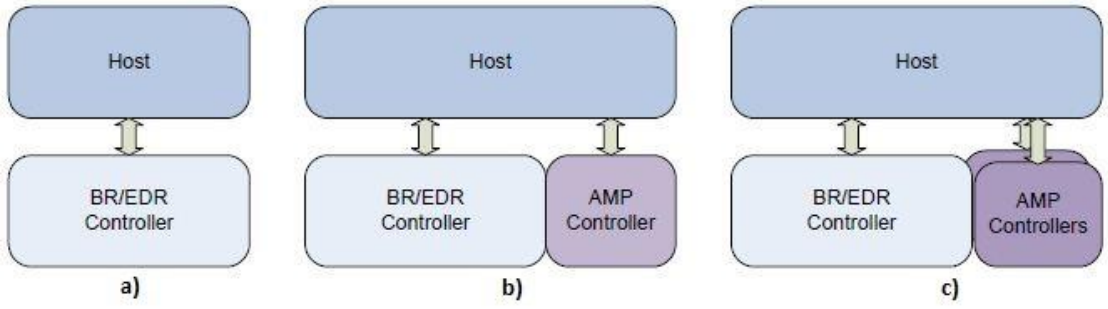

Figura 2.3 - Arquiteturas do Bluetooth 3.0 (adaptada de [14]).

Em termos de débito, o Bluetooth possui um débito de 1 Mbps para a versão Basic Rate (BR), e entre 2 Mbps a 3 Mbps para a versão Enhanced Data Rate (EDR), dependendo do tipo de modulação que é usada. Pode ainda alcançar um débito de 24 Mbps caso seja usada a transmissão do tráfego via ligação Wifi (IEEE 802.11). O

Bluetooth suporta 3 tipos de modulação que são GFSK (Gaussian Frequency Shift Keying), que oferece um débito de 1Mbps; PSK (Phase Shift Keying), que oferece um débito de 2 Mbps; e por último a 8DPSK (8-ary Differential Phase Shift Keying), com um débito de 3 Mbps.

2.3.2 Camada MAC

A uma rede Bluetooth dá-se o nome de *piconet*, e nesta existem dois tipos de dispositivos, mestre e escravo. Um mestre inicia sempre as conexões, sendo o responsável pelo acesso ao meio dentro da *piconet*. Um dispositivo pode pertencer a mais do que uma *piconet,* formando assim uma *scatternet*. Numa *piconet* os escravos associam-se ao mestre e a comunicação é feita sempre via mestre-escravo ou escravo-mestre através de um esquema de *polling*, não existindo por isso ligação direta entre escravos. A comunicação é gerida através de um esquema de divisão nos tempos (TDD – Time Division Multiplexing). A Figura 2.4 apresenta um exemplo de uma *scatternet* formada por três *piconets*.

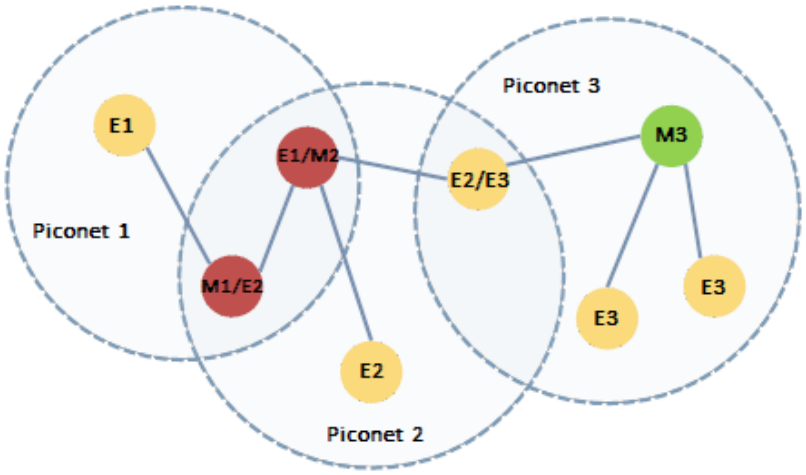

Figura 2.4 - Exemplo de um *scatternet* formada por 3 *piconets* (Mi – Mestre da piconet *i*; Ei – Escravo da piconet *i*) (adaptado de [6]).

A sequência de saltos numa piconet é determinada pelo endereço do dispositivo mestre, que é uma sequência de 48 bits única. Uma vez garantido que este endereço é único e sendo a sequência de saltos determinada a partir do mesmo, a própria sequência de saltos é também única para cada dispositivo que tome o papel de

mestre. Deste modo, torna-se teoricamente impossível que existam duas piconets com sequências de salto iguais, permitindo assim a coexistência de diversas *piconets*. Para que dois dispositivos possam comunicar entre si, necessitam de obter o endereço e sincronização com o relógio do mestre, para determinar a fase e a sequência de saltos da rede. Sem esta sincronização, os dispositivos não poderiam comunicar pois estariam desfasados entre si.

Quando um dispositivo de uma *piconet* se associa a outra *piconet* obtém o estatuto de *gateway*, pois passa a fazer a ponte entre as duas *piconets*. Este *gateway* apenas pode ser mestre de uma *piconet*, pois não poderiam coexistir duas piconets com a mesma sequência de saltos. Além disso, o facto de as piconets possuírem sequências de salto diferentes, diminui a interferência causada pela proximidade das redes.

Como foi referido, a interação entre mestre e escravo é feita através da divisão no tempo. Tal como sugere a Figura 2.5, o tempo está dividido em *slots* de 625 µs, e a duração da transmissão varia de acordo com o tamanho do pacote a enviar. Existem pacotes de ocupam 1,3 ou 5 *slots*, dependendo do seu tipo, definido neste *standard*.

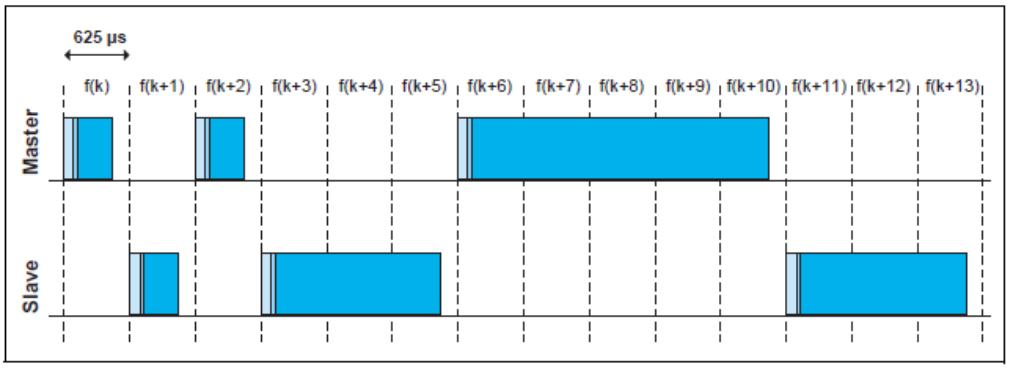

Figura 2.5 - Comunicação Mestre/Escravo [14].

Através da imagem podemos observar que o mestre inicia sempre as suas transmissões em frequências pares (k, k+2, k+6...),enquanto um escravo inicia em frequências ímpares (k+1, k+3, k+11...). Os tamanhos dos pacotes em *slots*, sendo números ímpares, fazem com que esta regra seja mantida entre mestre e escravo.

Quando um pacote que utiliza mais do que um *slot* é transmitido, a frequência usada no início desse *slot* é mantida ao longo de toda a transmissão. No final da transmissão é feito o salto para a frequência correspondente ao próximo *slot,* saltando as frequências ocupadas pelo pacote.

2.3.3 Bluetooth 4.0

A versão 4.0 do *standard* Bluetooth [15], além de possuir os controladores BR/EDR e AMP, possui também o BLE (Bluetooth Low Energy). O sistema BLE foi criado com o propósito de transmitir pacotes de informação muito pequenos (baixo débito) de cada vez, e ao mesmo tempo, aliado ao baixo custo, consumir uma quantia significativamente pequena de energia, quando comparada com os dispositivos que suportam BR/EDR [17]. Na Figura 2.6 é possível observar um exemplo de uma arquitetura possível na versão 4.0 do Bluetooth, combinando os diferentes controladores existentes. Além desta configuração é também possível definir apenas um controlador primário (BR/EDR ou BLE) com um host, contendo ou não um ou mais controladores secundários (AMP).

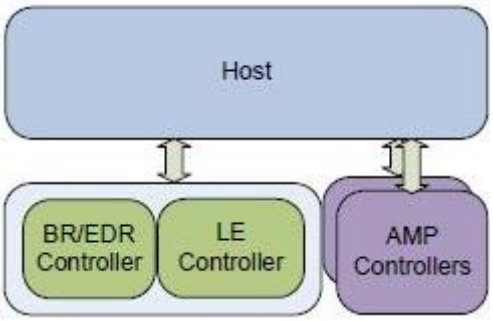

Figura 2.6 - Exemplo de arquitetura do Bluetooth 4.0 [15].

Dispositivos que suportam BR/EDR e BLE são referidos como dispositivos *dual-mode* e encaixam na categoria Bluetooth *Smart Ready.* Tipicamente, num sistema Bluetooth 4.0, um smartphone ou um computador portátil são dois exemplos de dispositivos *dual-mode*. Já dispositivos que suportem unicamente BLE são referidos como dispositivos *single-mode*, encaixando na categoria Bluetooth *Smart*. Estes dispositivos são geralmente usados para aplicações que requerem baixo consumo de

energia, são alimentados com recurso a baterias e apresentam tamanho reduzido. Exemplos deste tipo de dispositivos são sensores de monitorização vital ou ambiental. A Figura 2.7 representa a interação entre os diferentes sistemas Bluetooth, onde se observa que dispositivos *Smart Ready* são o elo central entre dispositivos que apenas suportam BLE (Bluetooth Smart) ou somente BR/EDR (Bluetooth).

Figura 2.7 - Interação entre diferentes sistemas Bluetooth (adaptado de [16]).

Relativamente às mudanças em relação à versão 3.0 do Bluetooth, pode-se verificar que a grande ênfase foi a introdução da componente de baixa energia (LE – Low Energy) acompanhada de novas funcionalidades como [15]:

- Camada física LE;
- Camada de ligação LE;
- Melhoramentos da camada HCI para LE;
- Modo teste direto LE;
- Encriptação AES;
- Melhoramentos da camada L2CAP para LE;
- Melhoramentos da camada GAP para LE;
- Camada ATT (Attribute Protocol);
- Camada GATT (Generic Attribute Profile);
- Camada de gestão de segurança (SM – Security Manager).

2.4 Bluetooth Low Energy

A especificação da versão 4.0 pelo Bluetooth SIG [15] define-a como sendo uma tecnologia de comunicação sem fios de curto alcance, muito baixo consumo energético (ULP – Ultra Low Power), com uma pilha protocolar leve e que permite a integração com a tecnologia Bluetooth existente. De resto, tem muito em comum com as versões anteriores, pelo facto de ambas serem de baixo custo, robustas e operarem na mesma banda de frequências.

A pilha protocolar BLE, representada na Figura 2.8, consiste em duas secções principais, que são o controlador (Controller) e o anfitrião (Host). A separação destas duas secções já é conhecida do padrão do Bluetooth BR/EDR. Quaisquer perfis ou aplicações usadas situam-se no topo da pilha protocolar.

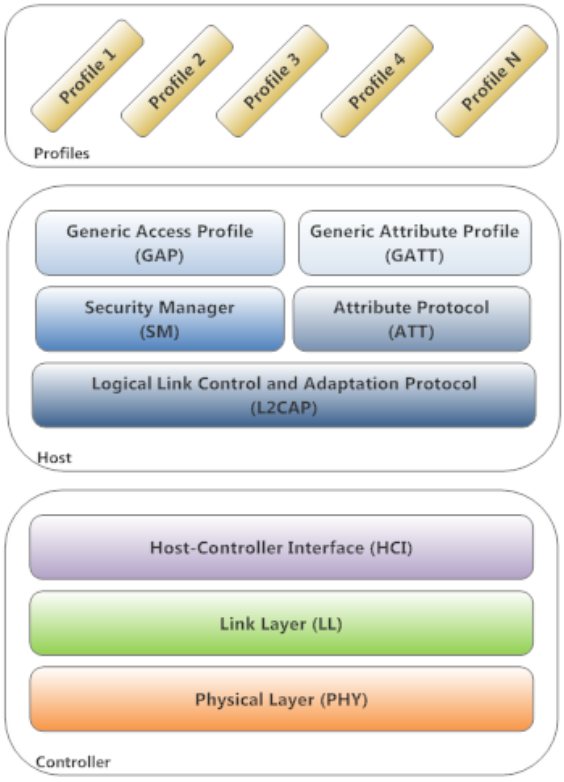

Figura 2.8 - Pilha Protocolar BLE (adaptado de [17]).

2.4.1 Camada Física

O BLE opera na banda de frequências de 2,4 GHz e define 40 canais de rádio frequência (RF – Radio Frequency). Existem dois tipos de canais RF no BLE: canais de anúncios e canais de dados. Os canais de anúncio são três e são usados para descobrir dispositivos, estabelecer conexões e para transmissões *broadcast*, enquanto os 37 canais restantes de dados são usados para comunicação bidirecional ente os dispositivos conectados.

É usado o mecanismo de espalhamento espectral por saltos em frequência (FHSS – Frequency Hopping Spread Spectrum) nos canais de dados para evitar interferências e problemas de propagação no meio sem fios como desvanecimento (*fading*) e múltiplos caminhos (*multipath*). Este mecanismo seleciona um dos 37 canais disponíveis para comunicação durante um dado intervalo de tempo [18].

O facto de existirem apenas três canais de anúncio torna mais provável a ocorrência da escolha do mesmo canal por parte de outra tecnologia de transmissão, resultando na degradação do sinal. Para combater esse facto, os canais de anúncio do BLE foram cuidadosamente escolhidos de modo a não coincidirem com os canais *standard* do Wi-Fi. A Figura 2.9 ilustra o mapeamento das frequências BLE para os canais de dados e de anúncio.

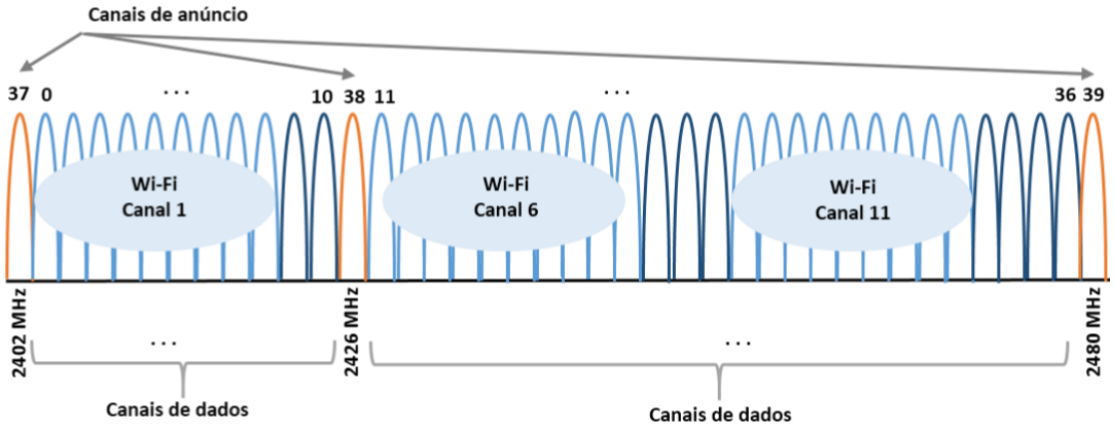

Figura 2.9 - Coexistência dos canais BLE e Wi-Fi (adaptado de [15]).

Todos os canais da camada física usam modulação Gaussiana (GFSK – Gaussian Frequency Shift Keying) que é baseada na modulação FSK, onde a diferença reside na

introdução de um filtro gaussiano para diminuir a largura espectral dos impulsos suavizando a transição entre os valores dos mesmos, permitindo também reduzir o pico de energia consumida. A taxa de transmissão na camada física é de 1 Mbps e o seu alcance é de aproximadamente 150 metros em zona ampla livre de obstruções [19]. A sensibilidade do recetor, segundo a especificação, é definida como o nível de sinal necessário para que a taxa de erro por bit (BER – Bit Error Rate) seja de 10^{-3}. Esse valor deve ser igual ou melhor que -70 dBm, o que garante vários metros de cobertura.

2.4.2 Camada Ligação

A operação da camada de ligação pode ser descrita em termos de uma máquina de estados que opera em 5 estados distintos, sendo eles: *standby*, anúncio (*advertising*), descoberta (*scanning*), iniciação (*initiating*) e conexão (*connection*), tal como sugere a Figura 2.10. Apenas um destes estados está ativo num certo período de tempo.

No estado de *standby* não são transmitidos ou recebidos quaisquer pacotes, e este estado pode ser acedido a partir de qualquer outro estado. No estado de anúncio é possível transmitir pacotes de anúncio para a rede, e há a possibilidade de ouvir e responder a respostas desencadeadas por esses canais de anúncio. Um dispositivo neste estado é chamado de anunciante (*advertiser*) e este estado pode ser acedido através do estado de *standby*.

A camada de ligação a operar no estado de descoberta escuta os canais de anúncio procurando por pacotes de anúncio vindos de dispositivos anunciantes. Um dispositivo neste estado é chamado de “scanner”, e o estado anterior a este é sempre o de *standby*. No estado de iniciação ocorre a escuta por pacotes de anúncio vindos de dispositivos específicos, e ocorre a resposta a esses canais de forma a iniciar-se uma ligação com esse dispositivo. Um dispositivo que inicia uma conexão é chamado de iniciador, e transita para este estado vindo do estado de *standby*.

Por fim, o estado de conexão pode ser acedido pelo estado de anúncio ou de iniciação, pois um dos dispositivos estava a anunciar e o outro é que descobriu e

iniciou a ligação. Assim, aquando num estado de conexão, dois papéis são definidos ao nível da camada de ligação: mestre e escravo. A um dispositivo que chega ao estado de conexão vindo de um estado de iniciação, é-lhe atribuído o papel de mestre; já o dispositivo que transitou do estado de anúncio para o estado de conexão fica com o papel de escravo.

A camada de ligação no papel de mestre comunica com um ou mais dispositivos escravos e define os tempos de transmissão dos dados. Já no papel de escravo, a camada de ligação, comunica apenas com um dispositivo mestre.

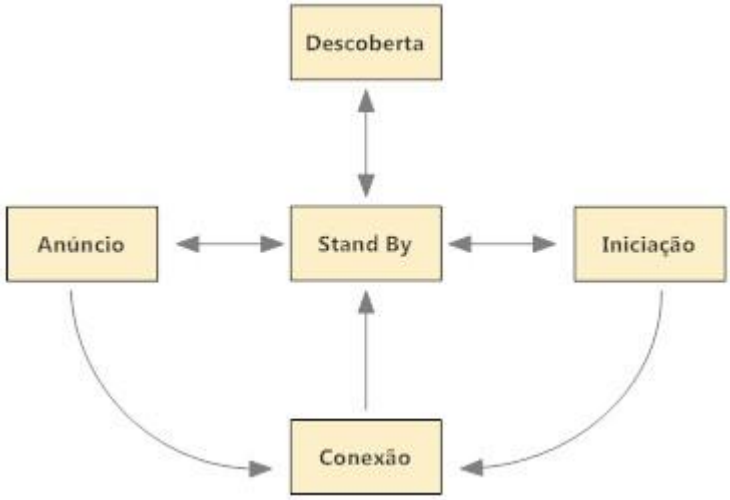

Figura 2.10 – Máquina de estados da camada de ligação (adaptado de [15]).

Existem 3 parâmetros essenciais ao funcionamento de todo o protocolo MAC do BLE, que são definidos aquando do estabelecimento de uma conexão: intervalo de conexão, latência do escravo e *timeout* de supervisão. Estes parâmetros podem ser descritos como:

• Intervalo de conexão (*connection interval*): Numa conexão BLE entre dois dispositivos é usado o esquema de saltos em frequência de modo a garantir que os dois troquem informação num dado canal num certo instante de tempo. Este momento é chamado de evento de conexão (*connection event*). Sendo assim o intervalo de conexão é o tempo que decorre entre dois eventos de conexão. O valor mínimo é de 7,5 milissegundos e o valor máximo é afixado em 4 segundos. Diferentes aplicações podem requerer diferentes intervalos de conexão;

• Latência do escravo (*slave latency*): Este parâmetro confere ao dispositivo periférico (escravo) a opção de saltar um certo número de eventos de conexão. Garante mais flexibilidade ao dispositivo periférico, dado que se não tiver dados para enviar pode escolher saltar os próximos eventos de conexão e ficar a dormir, sem ter de acordar para escutar o mestre, contribuindo assim para uma poupança ao nível energético. O valor atribuído a esta latência representa o número máximo de eventos que podem ser ultrapassados podendo ir de um mínimo de 0 até ao máximo de 499;

• *Timeout* de supervisão (*supervision timeout*): Uma ligação pode falhar devido a vários motivos, como o caso de um dispositivo se mover para fora da área de alcance, zonas de elevada interferência ou até falha de bateria por parte de um dos dispositivos. Posto isto, é importante que tanto o mestre como o escravo tenham controlo sobre o estado de ligação e, por isso, aquando do pedido de conexão é definido o valor de *timeout* de supervisão, sendo que este é reiniciado a cada vez que se envia um novo pacote. Desta forma, se após a transmissão este valor exceder, a ligação é considerada perdida.

2.4.3 Interface Controlador-Anfitrião (HCI)

A camada HCI fornece os meios de comunicação entre a secção controladora (Controller) e a secção anfitriã (Host) através de interfaces padronizadas. Esta camada pode ser implementada quer através de uma API, quer por interfaces de hardware como UART (Universal Asynchronous Receiver/Transmiter), SPI (Serial Peripheral Interface) ou USB (Universal Serial Bus).

2.4.4 Protocolo de Controlo e Adaptação da Ligação Lógica (L2CAP)

O L2CAP gere o encapsulamento dos dados para as camadas superiores, permitindo assim uma comunicação de dados lógica fim-a-fim entre dispositivos. Este processo é feito numa abordagem de melhor esforço (*best-effort*) sem usar mecanismos de retransmissões e controlo de fluxo, ao contrário do que acontece

noutras versões do Bluetooth. Os protocolos das camadas superiores fornecem estruturas de dados que se encaixam no tamanho máximo do payload do L2CAP, que é 23 bytes.

2.4.5 Camada de Gestão da Segurança

A camada de gestão de segurança define os métodos para emparelhamento e distribuição de chaves e fornece funções para que outras camadas da pilha conectem-se de modo seguro e troquem dados com outros dispositivos. Os dados são lidos ou escritos apenas quando a conexão é autenticada, e quando esta é formada, os dois dispositivos passam por um processo designado de emparelhamento. Quando este processo ocorre, as chaves são estabelecidas quer para encriptar quer para autenticar a ligação [17]. Nos casos normais, o dispositivo periférico irá requisitar uma chave, designada por "passkey", ao dispositivo central de modo a completar o processo de emparelhamento. Este pode ser um valor fixo, como "000000", ou outro fornecido pelo utilizador. Depois do dispositivo central enviar corretamente esta chave, os dois dispositivos trocam as chaves de segurança para encriptar e autenticar a ligação.

Em muitos casos, o dispositivo central e o periférico conectam-se e desconectam-se regularmente um do outro. O BLE tem uma função de segurança chamada de "bonding" que permite aos dois dispositivos, no processo de emparelhamento, fornecer um ao outro um conjunto de chaves de segurança de longo termo. Esta função permite que dois dispositivos restabeleçam rapidamente a encriptação e autenticação após voltarem a conectar-se sem terem de passar novamente pelo processo de emparelhamento.

A camada de ligação suporta encriptação e autenticação usando um algoritmos de cifras por blocos. Quando se aplica a encriptação e autenticação a uma conexão, são acrescentados 4 bytes ao *payload* do canal. Esses 4 bytes são chamados de Message Integrity Check (MIC). Também é possível transmitir dados autenticados através de uma ligação não encriptada. Neste caso, uma assinatura de 12 bytes é

colocada depois do *payload* na camada ATT. A assinatura é feita aplicando um algoritmo que usa AES de 128 bits como cifra de blocos.

2.4.6 ATT (Attribute Protocol)

Este protocolo permite expor a outros dispositivos certos componentes dos seus dados, chamados de atributos. Os atributos são estruturas de dados que armazenam informações fornecidas pela camada GATT, que será abordada mais à frente. Um atributo é composto por 4 elementos: 16 bits para a *handle* que define unicamente um atributo; um UUID (Universal Unique Identifier) de 16 ou 32 bits que define o tipo de atributo; a sua descrição e por fim o seu valor de tamanho variável.

No contexto do ATT, um dispositivo que expõe os seus atributos é referido como o servidor e o dispositivo vizinho é o cliente. Os papéis de mestre/escravo atribuído na camada de ligação são independentes dos papéis de cliente/servidor definidos neste protocolo. Por exemplo, um dispositivo mestre tanto pode ser um cliente ou um servidor, e o mesmo acontece com o dispositivo que toma o papel de escravo. É também possível um dispositivo ser cliente e servidor em simultâneo. O cliente pode aceder aos atributos do servidor através do envio de pedidos, e o servidor, de modo a garantir eficiência, pode enviar ao cliente dois tipos de mensagens não solicitadas contendo os atributos, que são: notificações, onde o servidor envia dados ao cliente e este não confirma a sua receção; ou indicações, que requerem uma confirmação por parte do cliente.

2.4.7 GATT (Generic Attribute Profile)

A camada GATT define uma *framework* para usar os procedimentos da camada ATT para descoberta de serviços e especifica a estrutura dos perfis. Todas as comunicações de dados que ocorrem entre dispositivos numa conexão BLE são processadas através de procedimentos desta camada. No BLE, todos os dados que estão a ser usados para um serviço são chamados de “características”. Uma característica é por sua vez um conjunto de dados que inclui um valor e certas

propriedades. Os dados relativos ao serviço e às suas características são guardados em atributos. De forma hierárquica podemos dizer que um perfil pode ter vários serviços e cada serviço pode ter várias características. A definição dos serviços e características está organizada sob a forma de atributos, mapeados numa tabela chamada de tabela de atributos.

Como já foi referido, toda esta informação está definida na camada GATT, mais propriamente no dispositivo servidor (GATT Server) pois é neste que estão definidos vários serviços, que como foi já dito são nada mais nada menos que uma coleção de dados para executar uma determinada função, como por exemplo informação obtida de um sensor de temperatura. O dispositivo cliente (GATT Client) tem a hipótese de aceder a estes atributos através de operações de leitura e escrita, consoante as permissões implementadas.

Segundo as normas definidas pelo Bluetooth SIG, a definição de um serviço tem como UUID o valor 0x2800 [16]. Todos os atributos seguintes pertencem a este serviço até ser encontrado novamente o mesmo UUID, significando assim o início de um novo serviço. Além disso, o valor deste atributo é outro UUID, mas que neste caso especifica o tipo de serviço que é, ou seja, o primeiro UUID referido diz respeito ao tipo de atributo e serve para delimitar o serviço, já o segundo refere-se ao tipo de serviço (e.g. sensor de temperatura, alarme de proximidade). Segue-se um exemplo na Tabela 2.2.

Tabela 2.2 – Exemplo de definição de dois serviços.

Handle	UUID	Descrição	Valor
0x0100	0x2800	Serviço A	UUID = 0x1802
…	…	Detalhes do Serviço A	…
0x0150	0x2800	Serviço B	UUID = 0x188F
…	…	Detalhes do Serviço B	…

As "características", fazendo parte dos serviços, têm também um UUID específico para as representar, que é 0x2803. Este valor delimita uma característica, sendo que cada uma delas tem no mínimo dois atributos: um para a identificar e outro para definir o seu valor. Além destes dois atributos, uma característica pode ter mais, caso necessário. Estes atributos extras são chamados de descritores e servem para dar mais informação acerca do valor da característica (Tabela 2.3).

Tabela 2.3 - Exemplo de definição de características para os dois serviços.

Handle	UUID	Descrição	Valor
0x0100	0x2800	Serviço A	UUID = 0x18xx
0x0101	0x2803	Característica 1	UUID = 0x2A2B Handle = 0x0102
0x0102	0x2A2B	Valor Car.1	Ex: 20
0x0104	0x2A2F	Descritor 1	Celsius
0x0110	0x2803	Característica 2	UUID = 0x2A2D Handle = 0x0111
0x0111	0x2A2D	Valor Car. 2	Ex: 13h:25

Tal como acontece nos serviços, sabe-se que este termina onde for encontrado um novo UUID de definição de serviço (0x2800). O mesmo acontece com as características, ou seja, uma característica acaba e inicia-se outra sempre que na tabela de atributos se encontrar um UUID igual a 0x2803. Este valor é também um valor padrão da norma definida pelo Bluetooth SIG.

Existem vários descritores já definidos pela norma, e um muito importante é o descritor de configuração da característica do cliente (CCC – Client Characteristic Configuration). Este descritor, cujo UUID é 0x2902, é um *bitmap* de 16 bits que podem ser lidos e escritos. É um atributo como outro qualquer do lado do servidor, sendo que os seus dois primeiros bits já estão tomados pela especificação do GATT, servindo para configurar se o valor da característica ao qual está associado será

enviado para o cliente sob a forma de notificação ou indicação. O bit “0” informa sobre notificações e o bit “1” sobre indicações. A incorporação deste descritor é apresentada na Tabela 2.4, na qual assume-se que as notificações estão ativadas.

Tabela 2.4 - Incorporação do descritor CCC no exemplo do serviço A.

Handle	UUID	Descrição	Valor
0x0100	0x2800	Serviço A	UUID = 0x18xx
0x0101	0x2803	Característica 1	UUID = 0x2A2B Handle = 0x0102
0x0102	0x2A2B	Valor Car.1	Ex: 20
0x0104	0x2A2F	Descritor 1	Celsius
0x0105	0x2902	Descritor CCC	0x0001
0x0110	0x2803	Característica 2	UUID = 0x2A2D Handle = 0x0111
0x0111	0x2A2D	Valor Car. 2	Ex: 13h:25

2.4.8 GAP (Generic Access Profile)

No nível mais alto da pilha protocolar do BLE situa-se a camada GAP. Esta camada é a interface direta com as aplicações e/ou perfis, sendo também responsável pela especificação dos papéis dos dispositivos, modos para a descoberta de dispositivos e serviços, gestão do estabelecimento e término de ligação e, por fim, a iniciação do processo de segurança. Esta camada opera sempre num destes 4 papéis: *broadcaster*, observador, periférico ou central.

Um dispositivo no papel de *broadcaster* apenas difunde informação através dos canais de anúncio e não suporta conexões com outros dispositivos. Um dispositivo que é observador é o complemento do *broadcaster*, ou seja, escuta os anúncios, não tendo também a capacidade de se conectar.

Um dispositivo central procura por anúncios e inicia conexões, opera como mestre e pode gerir múltiplas conexões. Por fim, um dispositivo periférico é um anunciador que tem a competência de se conectar e opera como escravo numa conexão simples, ou seja, está ligado a apenas um mestre.

Num sistema BLE típico, o dispositivo periférico anuncia que é um dispositivo pronto a estabelecer uma ligação, para dar conhecimento aos dispositivos centrais. Estes anúncios contêm o endereço único do dispositivo e podem conter dados adicionais como o nome do dispositivo. Na Figura 2.11, o dispositivo central, após receber o anúncio, envia ao dispositivo periférico um "*Scan request*" ao qual este lhe responde com um "*Scan response*". Este é o processo de descoberta de dispositivos, na medida em que o dispositivo central passa a saber que o dispositivo com quem comunicou é uma estação periférica e pode assim estabelecer uma ligação. O dispositivo central envia então um pedido de estabelecimento de conexão (*Link Request*) ao dispositivo periférico, e este responde com um "*Link Response*".

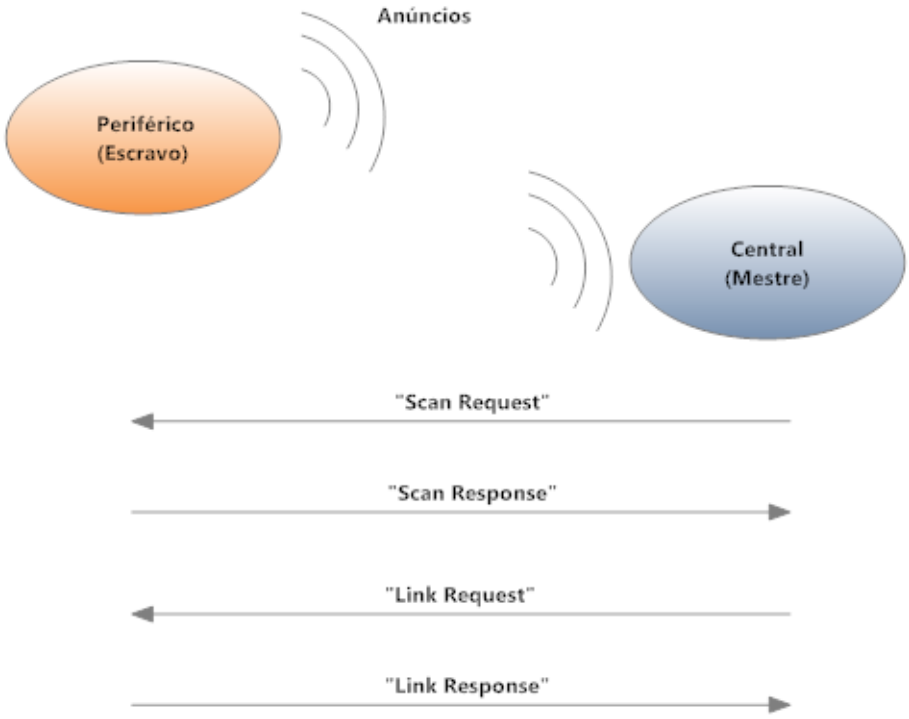

Figura 2.11 - Processo de estabelecimento de conexão entre mestre e escravo.

2.4.9 Formato dos Pacotes

O controlador LE (Low Energy) inserido nesta nova versão do Bluetooth apresenta apenas um formato de pacotes, quer para os que circulam nos canais de anúncios, quer para os que circulam nos canais de dados. O pacote consiste em 4 campos distintos: preâmbulo (1 byte), endereço de acesso (4 bytes), PDU (Protocol Data Unit) (2 a 39 bytes) e CRC (3 bytes), tal como se pode ver na Figura 2.12.

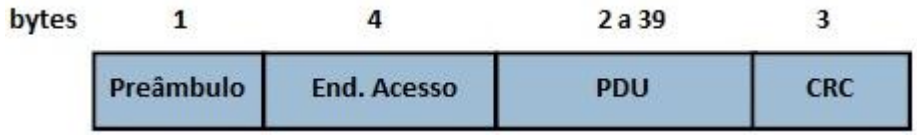

Figura 2.12 – Formato geral de um pacote BLE (adaptado de [15]).

O tamanho mínimo de um pacote BLE é então 10 bytes (80 bits) e o maior pacote possível pode ter até 47 bytes (376 bits).

O preâmbulo é usado no recetor para executar a sincronização da frequência, estimar o sincronismo de um símbolo e para o controlo do ganho automático. O endereço de acesso é um número de 32 bits, único e gerado aleatoriamente pelo dispositivo no estado de iniciação e enviado no pedido de conexão.

O CRC é um mecanismo de identificação de erros e é calculado sobre o PDU do pacote. É um CRC polinomial de 24 bits e todos os bits do PDU devem ser processados pela ordem de transmissão, ou seja, do menos para o mais significativo.

Relativamente ao PDU, existem dois tipos, dependendo se o pacote é transmitido nos canais de anúncios ou nos canais de dados. Estes dois tipos são descritos nas secções 2.4.9.1 e 2.4.9.2, respetivamente.

2.4.9.1 PDU dos Canais de Anúncio

O PDU que diz respeito aos canais de anúncio é constituído por um cabeçalho (*header*) de 2 bytes, visível na Figura 2.13, seguido do respetivo *payload*, que é específico para o tipo de PDU. Nesta categoria existem 7 subtipos de PDUs.

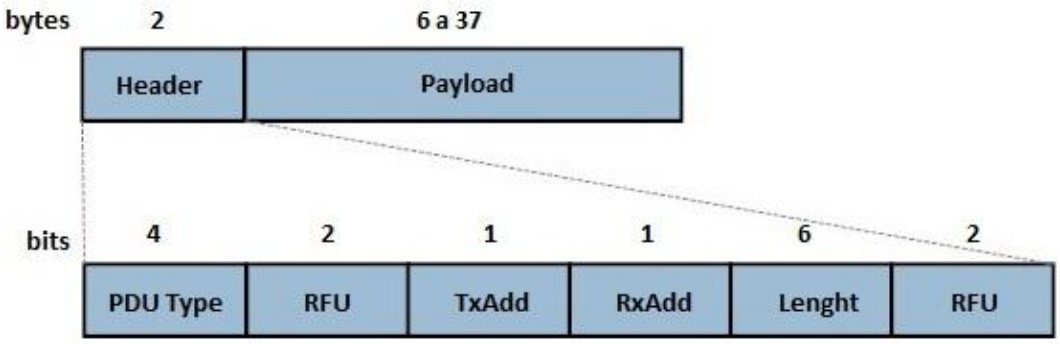

Figura 2.13 - Formato do PDU no canal de anúncio (adaptado de [15]).

O campo *Header* é composto por 6 subcampos. O *PDU Type* indica qual o tipo de PDU, pois dentro da categoria de PDUs de canais de anúncio existem 7 tipos possíveis. Estes tipos são descritos na Tabela 2.5. Os campos *TxAdd* e *RxAdd* tomam certos valores (zero ou um), dependo do tipo de PDU, e caso não sejam usados são reservados para uso futuro (RFU – Reserved Future Use). O campo *Lenght é* o tamanho do payload do PDU em bytes, que como está descrito na figura pode ir desde 6 a 37 bytes.

Tabela 2.5 - Subtipos de PDU do canal de anúncios (adaptado de [15]).

Tipo PDU	Nome do Pacote
0000	ADV_IND
0001	ADV_DIRECT_IND
0010	ADV_NONCONN_IND
0011	SCAN_REQ
0100	SCAN_RSP
0101	CONNECT_REQ
0110	ADV_SCAN_IND

2.4.9.2 PDU dos Canais de Dados

O PDU do canal de dados é constituído por um *header* (2 bytes), um *payload* de tamanho variável (0 a 27 bytes) e um campo *MIC* (Message Integrity Check) opcional (4 bytes). O *MIC* não está incluído nas situações em que se verifica uma conexão não encriptada ou uma conexão encriptada com o *payload* do PDU igual a zero. A Figura 2.14 ilustra o formato do PDU do canal de dados.

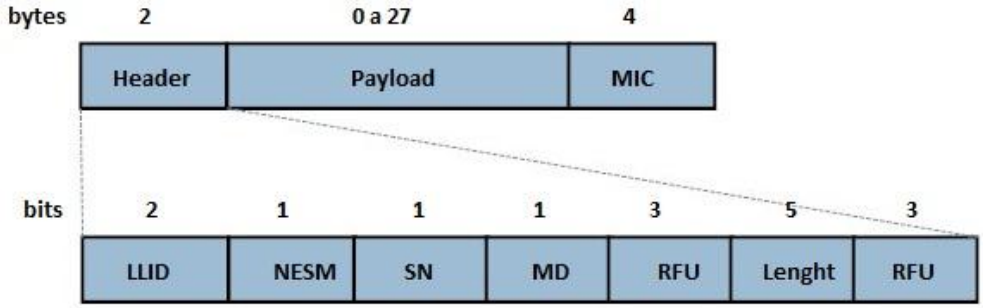

Figura 2.14 - Formato do PDU no canal de dados (adaptado de [15]).

O campo *LLID* define o tipo de *payload*. Se tiver o valor 01b (1) ou 10b (2), contém um *payload* de dados, do tipo "LL DATA PDU"; caso tenha o valor 11b (3), então o payload é de controlo, do tipo "LL CONTROL PDU". Um payload de dados significa que irá conter dados vindos da camada L2CAP, e se o valor for 01b significa que o payload é vazio. Este tipo de PDU é enviado pela camada de ligação no papel de mestre para permitir ao escravo responder com qualquer PDU de dados, incluindo um PDU vazio. Caso o valor seja 10b, o payload irá conter dados, ou seja, o seu tamanho é diferente de zero. Os PDUs de controlo têm como função garantir o controlo e manutenção da conexão ao nível da camada de ligação. O campo *NESM* indica o próximo número de sequência esperado, ou seja, é usado pelo recetor para reconhecer o último PDU enviado ou para requisitar um novo envio do mesmo. O campo *SN* contém o número de sequência usado para identificar os pacotes enviados pela camada de ligação.

O campo *MD* significa "more data", e permite indicar se o dipositivo tem mais dados para enviar (limitação do número de notificações descrito com mais detalhe na secção 5.1). Se nenhum dispositivo tem o bit MD ativo nos seus pacotes, o pacote

enviado a partir do escravo fecha a ligação. Caso um ou ambos os dispositivos tenham o bit MD ativo, o mestre continua com o evento de conexão enviando um novo pacote, devendo o escravo escutar depois de enviar o seu pacote. Se um pacote não for recebido pelo mestre vindo do escravo, o mestre termina o evento de conexão. Caso um pacote seja enviado pelo mestre e o escravo não receba, este último termina o evento de conexão.

Por fim, o campo *Lenght*, de 5 bits, indica o tamanho do payload e do MIC, sendo por isso o seu tamanho máximo limitado a 31 bytes. Como o MIC tem um tamanho fixo de 4 bytes, o *payload* deve ser igual ou menor que 27 bytes.

Como foi referido anteriormente, caso o *payload* do PDU não seja vazio, irá conter os dados vindo da camada L2CAP. O formato dos dados nesta camada reserva 4 bytes para o cabeçalho e os restantes 23 bytes para dados. Por sua vez o *payload* da camada L2CAP define um byte para o *opcode*, restando 22 bytes para dados, sendo que 2 desses bytes então destinados à indicação da "ATT_HANDLE" ou seja a posição de memória onde se encontra o atributo que contém os dados a serem enviados. Posto isto conclui-se que o tamanho máximo de dados de utilizador disponível para envio num pacote BLE é de 20 bytes. A Figura 2.15 mostra o formato do *payload* do PDU do canal de dados.

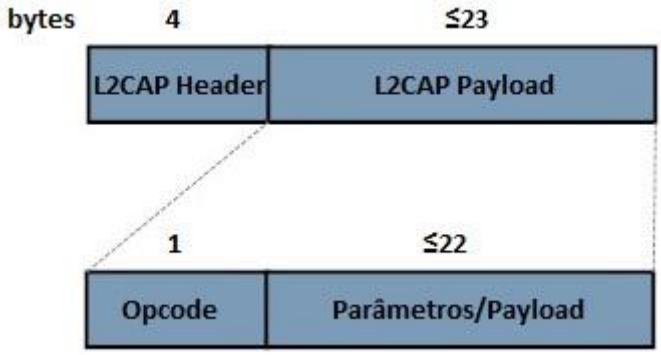

Figura 2.15 - Formato do *payload* do PDU de dados (adaptado de [15]).

2.5 Sistema Operativo Android

O Android [20] é um sistema operativo móvel baseado em Linux e desenvolvido pela Google. Desde o seu aparecimento, em 2005, o Android tem captado o

interesse de diversas companhias, criadores de aplicações e dos utilizadores em geral. Esta plataforma de software tem sido constantemente melhorada, quer em termos de novas características, quer em hardware suportado. O Android tem a vantagem de ser uma comunidade *open-source* e de estar cada vez mais disponível para um largo espectro de dispositivos.

A sua arquitetura consiste em 5 camadas: Aplicações (Applications), Framework de aplicações (Application Framework), Bibliotecas (Libraries), Android Runtime e Linux Kernel, como se observa na Figura 2.16. A camada de Aplicações situa-se no topo e oferece um conjunto de aplicações, visíveis ao utilizador, que fazem parte do conjunto essencial de qualquer sistema operativo móvel, como mensagem de texto e multimédia (SMS/MMS), relógio, calculadora, correio eletrónico, entre outros. A camada logo mais abaixo, Framework de Aplicações, fornece as APIs necessárias para serem usadas pelas aplicações da camada anteriormente descrita. Abaixo desta camada existe outra camada, Bibliotecas, que fornece as bibliotecas essenciais de suporte para as aplicações, como as de C/C++, multimédia, base de dados e desempenho gráfico. Por sua vez, esta camada engloba uma outra, Android Runtime, que fornece a sua própria máquina virtual (VM – Virtual Machine), denominada Dalvik, cuja pilha é baseada na Java VM, além de fornecer de igual modo outras bibliotecas de suporte às aplicações (Core Libraries). Por fim, a camada mais inferior diz respeito ao *kernel* Linux, que contém os controladores de Wi-Fi, Bluetooth, teclado, áudio, memória, câmara, entre outros.

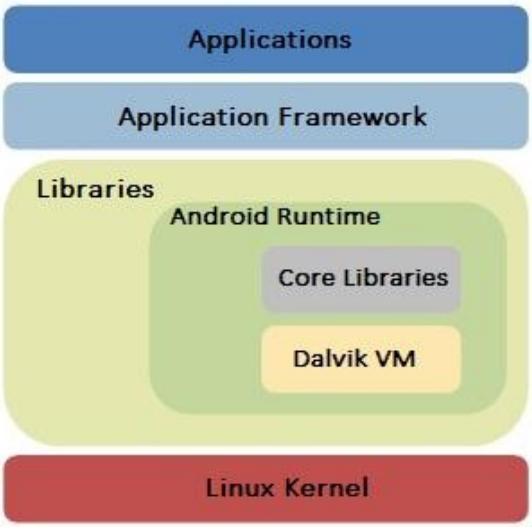

Figura 2.16 - Arquitetura do Android (adaptado de [21]).

Atualmente, o Android encontra-se na versão 4.4 KitKat. A partir da versão 4.3 Jelly Bean foi possível o suporte para dispositivos Bluetooth Smart Ready, ou seja, incorporação da tecnologia BLE. Toda a documentação e APIs acerca do BLE está disponível na página destinada aos criadores de aplicações Android [22].

O Android, dado à sua versatilidade e como plataforma open-source, tem captado cada vez mais a atração dos utilizadores comuns, afirmando-se assim no segmento dos dispositivos móveis. Segundo a IDC (International Data Corporation), entidade de referência para estudos de mercado na área das tecnologias, foram vendidas 255 milhões de unidades de dispositivos móveis com sistema operativo Android, no segundo trimestre de 2014. No mesmo período apresentaram uma quota de mercado na ordem dos 84,7 %. A Tabela 2.6 faz referência a estes números comparando os diferentes sistemas operativos móveis em diferentes períodos, relacionando o seu crescimento.

Tabela 2.6 – Venda de unidades (milhões) e quota de mercado dos sistemas operativos móveis, no segundo trimestre de 2013 e 2014 (adaptado de [23]).

Sistema Operativo	2T14 Vendas	2T14 Quota de Mercado	2T13 Vendas	2T13 Quota de Mercado	2T14/2T13 Crescimento
Android	255,3	84,7 %	191,5	79,6 %	33,3 %
iOS	35,2	11,7 %	31,2	13,0 %	12,7 %
Windows Phone	7,4	2,5 %	8,2	3,4 %	-9,4 %
BlackBerry	1,5	0,5 %	6,7	2,8 %	-78,0 %
Outros	1,9	0,6 %	2,9	1,2 %	-32,2 %
Total	301,3	100 %	240,5	100 %	25,3 %

2.5.1 Detalhes da Plataforma Android

Uma aplicação desenvolvida sob a plataforma Android tem 4 componentes essenciais: activities, services, content providers e broadcast receivers. Cada componente serve um determinado propósito e tem um ciclo de vida distinto que define como e quando o componente é criado e destruído.

As activities representam o ecrã de interface com utilizador para que este possa interagir com a aplicação, ou seja, basicamente é o que frequentemente se chama de janela. Uma aplicação pode ter várias activities, cada uma com tarefas distintas, sendo por isso independentes umas das outras. Por exemplo, uma aplicação de correio eletrónico pode ter uma activity para listar o correio recebido, outra para ler e outra para escrever. Todas trabalham de forma unida para fornecer a melhor experiência ao utilizador.

No Android, as activities são geridas como uma pilha de activities, em que quando uma nova activity é iniciada, é colocada no topo da pilha, tornando-se a activity em primeiro plano. A activity anterior permanece sempre em segundo plano

enquanto a activity atual existir. O ciclo de vida de uma activity pode ser explicado pela Figura 2.17.

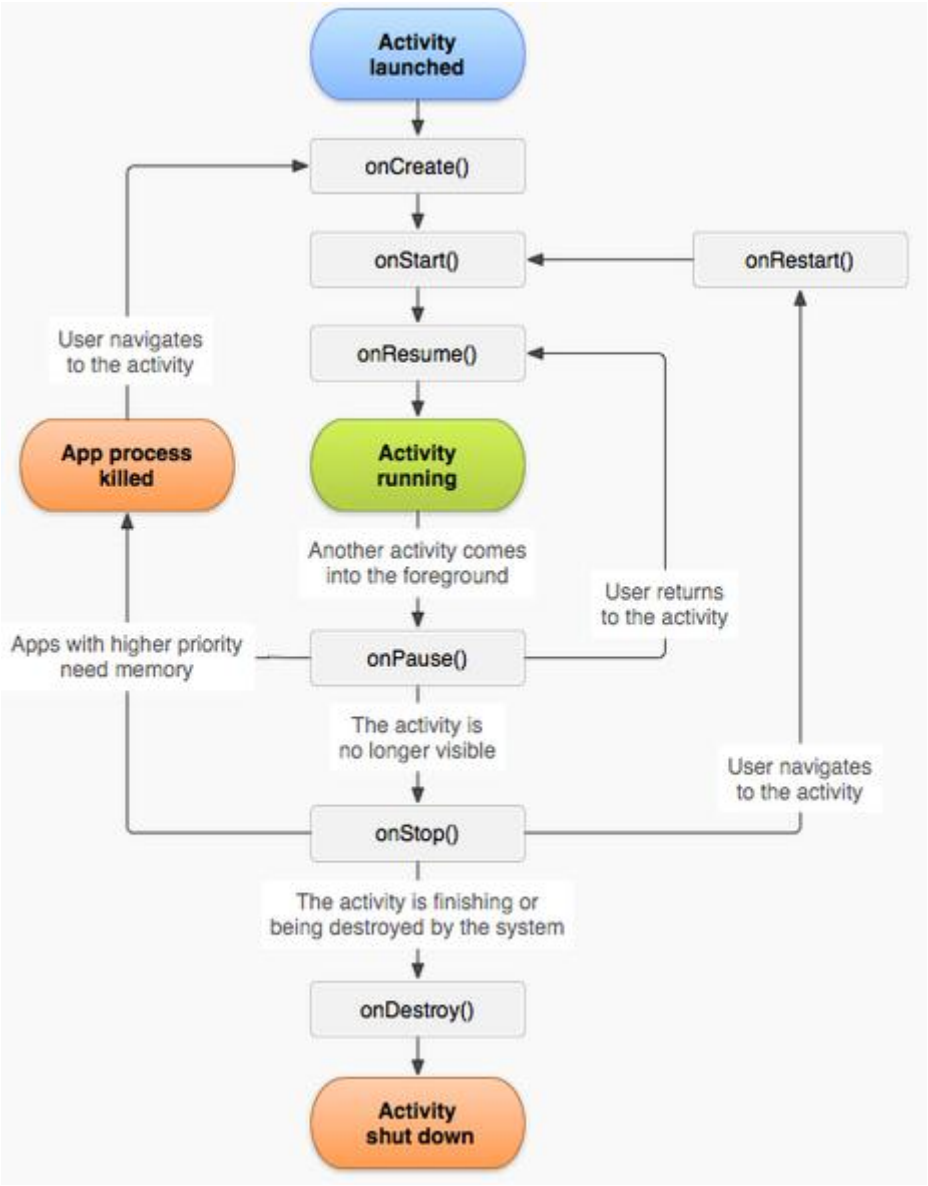

Figura 2.17 - Ciclo de vida de uma *activity* (adaptado de [22]).

Uma activity criada e que esteja em primeiro plano é considerada em execução (running). Caso esta activity perca protagonismo mas ainda assim seja visível, ou seja, se outra activity que não ocupe todo o ecrã apareça em primeiro plano, fica em pausa. Continua a ser uma activity presente, mas pode ser destruída (killed) a qualquer momento caso o sistema necessite de libertar memória.

Caso a activity seja completamente obstruída por outra, fica inativa. Continua com o estado atual e com as suas informações, mas já não é mais visível para o utilizador, sendo assim terminada, ou acaba destruída pelo sistema.

Existem métodos que podem ser implementados para desempenharem diversas operações quando a activity transita entre os diferentes estados, e que, de certa forma, auxiliam o programador a controlar o estado da aplicação. Os métodos são:

• onCreate() – é o primeiro método a ser executado aquando do arranque de uma activity. É responsável por carregar os layouts[1] XML, e é apenas executado uma vez durante o ciclo de vida de uma activity.

• onStart() – é o método invocado após o onCreate(), sendo também invocado quando uma activity que estava em segundo plano (invisível para o utilizador), volta a ter foco.

• onResume() – este método é chamado quando a activity retorna para primeiro plano, depois de estar parcialmente encoberta por outra activity. De notar que neste caso a activity nunca deixou de estar visível para o utilizador, apenas estava em pausa, não em paragem completa.

• onPause() – é o método que é executado quando a activity é parcialmente sobreposta por outra.

• onStop() – é o método executado quando uma activity perde completamente a visibilidade para o utilizador.

• onDestroy() – é o ultimo método a ser executado. Depois deste a activity é considerada destruída, não podendo ser relançada.

• onRestart() – é invocado imediatamente antes do onStart(), quando a activity passa novamente para primeiro plano depois de estar invisível ao utilizador.

[1] Um *layout* define a estrutura visual para a interface do utilizador, como os botões interativos, caixas de texto, caixa de validação, entre outros elementos.

Voltando aos componentes afetos a uma aplicação, os serviços (services) são um tipo de componente que corre em segundo plano, sendo desenhados para realizar operações durante longos períodos de tempo. Um serviço não fornece interface gráfica para diálogo com o utilizador, sendo a sua função executar sem interferir com a ação de outras activities.

Os content providers têm a capacidade de gerir um conjunto partilhado de dados das aplicações. É possível guardar os dados de uma aplicação na diretoria raiz do sistema Android, na base de dados do sistema (SQLite), na Internet ou em qualquer outro local que a aplicação tenha permissões para isso. Através dos content providers, aplicações podem requisitar ou até modificar dados de outra aplicação, ou seja, são responsáveis por fazer a ponte entre aplicações no que diz respeito à obtenção e troca de informação.

Por fim, existem os broadcast receivers, que são componentes que permitem a receção de mensagens por parte das outras aplicações ou pelo sistema. Estas mensagens são chamadas de eventos. Iniciar uma chamada, terminar uma chamada, ou receber um SMS, são exemplos de eventos que podem ocorrer e, através do broadcast receiver, são enviados avisos às aplicações para que estas, se desejarem, possam responder a estes eventos.

2.6 Trabalho Relacionado

As WPANs, WSNs e WBANs estão a tornar-se cada vez mais promissoras a cada dia que passa, tornando-se em alvos de estudo em diversas áreas. Esta secção tem o intuito de fornecer uma visão geral de alguns trabalhos e aplicações efetuados na área, e que de certa forma se encaixam e assemelham ao propósito do tema desta dissertação.

2.6.1 Aplicações baseadas em sensores

As áreas de saúde, desporto e entretenimento são usadas cada vez mais como aplicações das redes de sensores sem fios. A monitorização de sinais vitais do ser

humano é um dos principais motivos para que se use este tipo de redes nestas áreas da saúde e desporto.

Em [24] é apresentado o projeto MOHLL (Mobile Health Living Lab), que consiste na monitorização de sinais vitais de um paciente baseada numa WSN. Os sensores são utilizados para ECG (eletrocardiograma), SpO2 (saturação de oxigénio) e para temperatura corporal. A comunicação dos sensores é feita via ZigBee para uma estação central, que armazena os dados e posteriormente os envia para um PC ou PDA, via Internet para que possam ser consultados através de uma aplicação específica ou simplesmente através do navegador de Internet. A Figura 2.18 ilustra a arquitetura geral do sistema.

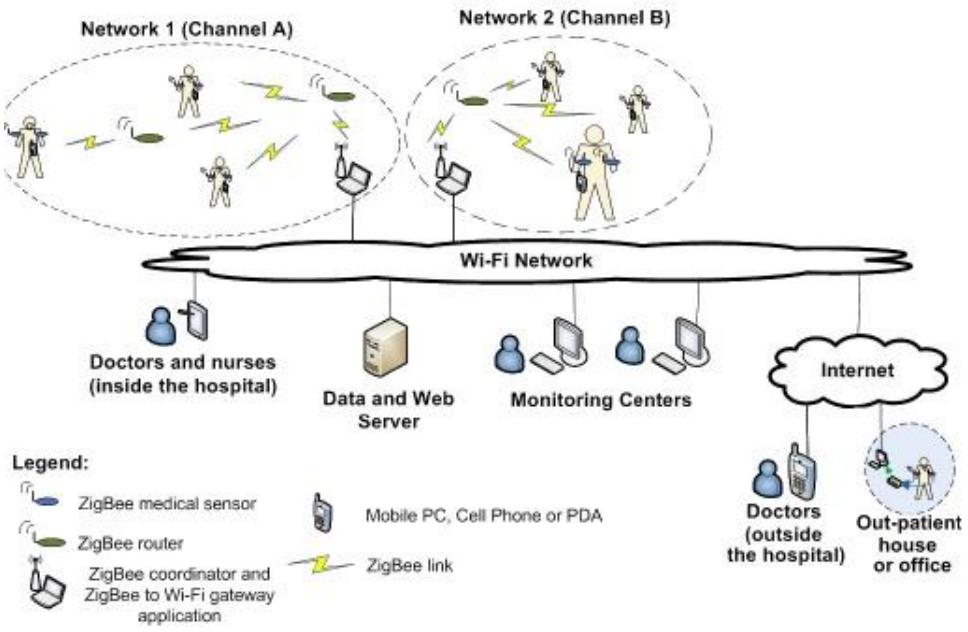

Figura 2.18 - Arquitetura geral do sistema MOHLL (adaptado de [24]).

Os autores reconhecem a necessidade da migração do sistema para smartphones com um sistema operativo mais universal e flexível, como o caso do Android ou Windows Phone, devido ao facto de estarem em constante crescimento e melhoramento.

A monitorização e avaliação de movimentos corporais ou postura do ser humano também é fortemente associada ao conceito de WSN, principalmente na área do desporto, *fitness, saúde* e entretenimento. Em [25], os autores apresentam um

protótipo de uma rede de sensores sem fios e a sua aplicação num sistema de captura de movimentos através de sensores inerciais e magnéticos. Este sistema apresenta uma arquitetura modular e permite monitorizar vários utilizadores em simultâneo usando apenas uma rede sem fios. Os requisitos são, por isso, que a rede suporte um elevado número de nós, uma operação em tempo real e fiabilidade na entrega dos dados. O sistema é flexível e versátil, permitindo um elevado leque de aplicações alvo. Os módulos sensoriais podem ser colocados em diversos segmentos do corpo humano, por exemplo, no caso de aplicações com fins fisioterapêuticos.

Como é descrito em [26], estes módulos sensoriais podem ser colocados na parte superior e inferior da perna, de modo a avaliar o ângulo formado entre esses segmentos. Em ambos os trabalhos são usados módulos sensoriais com acelerómetros e magnetómetros com fim de avaliar a inclinação e rotação do corpo humano em relação ao eixo da Terra.

Estes módulos sensoriais comunicam com a estação base através do protocolo eLPRT [27], que é um novo mecanismo de reserva proposto como alternativa à camada MAC do IEEE 802.15.4. O mecanismo original, GTS, que permite dar suporte a aplicações que requerem uma largura de banda dedicada e atraso limitado, revela alguns pontos negativos como a utilização ineficiente da largura de banda e suporte para um máximo de 7 dispositivos. O eLPRT introduz melhoramentos em certas características aumentando a taxa de transmissão de dados, melhora a utilização da largura de banda e aumenta o número de dispositivos suportados.

Em [28], os autores apresentam um sistema de controlo e aquisição de dados numa rede com vários nós, baseada em Bluetooth. Este sistema tem como objetivo permitir a navegação autónoma de um veículo aéreo (UAV – Unmanned Aerial Vehicle) através do processamento da informação recolhida pelos vários sensores e comunicação entre os mesmos. A rede é formada por uma estação mestre, conectada ao controlador de voo e até 7 estações escravas espalhadas ao longo da estrutura, conectadas claro a sensores e atuadores. A Figura 2.19 ilustra a distribuição dos diferentes nós ao longo da estrutura do UAV.

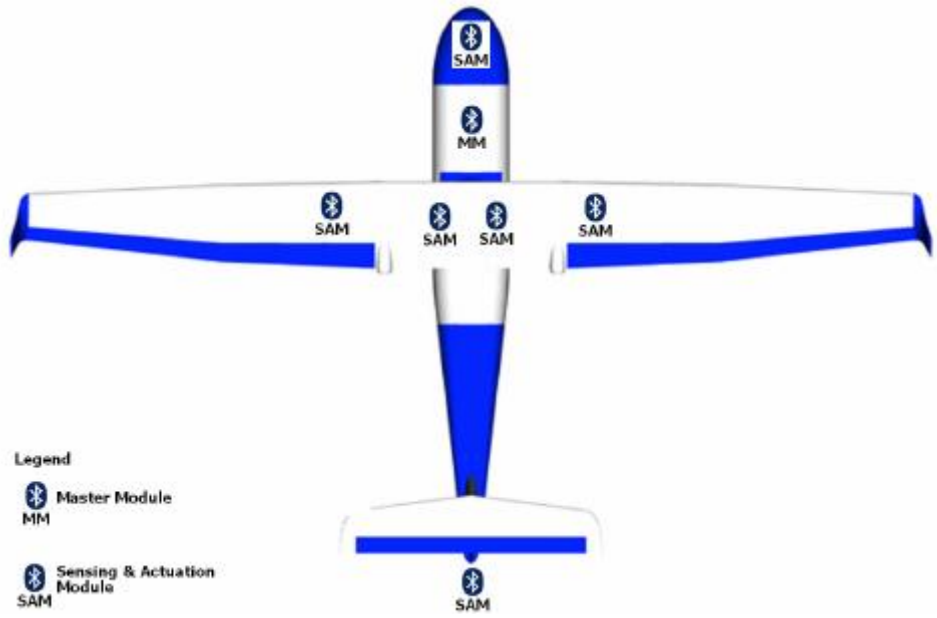

Figura 2.19 – Distribuição dos nós pela estrutura do UAV (adaptado de [28]).

O nó mestre (MM – Master Module) é colocado no corpo da fuselagem e atua como controlador de voo, efetua registo dos dados (*log*) e é responsável pela comunicação com a estação terrestre. Como escravos são usados 6 nós, SAM (Sensing and Actuation Modules), dos quais dois são colocados nas asas para controlo de propulsão e velocidade dos motores, controlo de superfícies como *ailerons* e monitorização da temperatura; dois colocados no corpo da fuselagem com GPS e giroscópio para assegurar informação acerca da navegação; um colocado na cauda do aparelho para controlo do ascensor/leme e informação do posicionamento; e finalmente um colocado na parte frontal que contém sensores de pressão para fornecer informação precisa acerca do controlo do voo. Além disso, este último nó fornece suporte para o mecanismo automático de descolagem e aterragem do aparelho.

2.6.2 Protocolos de comunicação

Independentemente da rede de sensores em causa, ou do sistema a implementar, é necessário optar por uma tecnologia de comunicação que apresente um protocolo que garanta uma comunicação de curto alcance com débito razoável,

minimize o consumo energético, garanta escalabilidade e que possa apresentar compatibilidade com outras plataformas de hardware e software para garantir interoperabilidade.

Em [1], os autores apresentam uma visão geral no que toca a aplicações, desafios e perspetivas no uso das WBANs. Referem os aspetos de QoS para conectividade em tempo real numa topologia de rede heterogénea, as questões de privacidade e segurança, bem como a intervenção de entidades reguladoras nas WBANs. O artigo faz uma comparação das características principais de diversas tecnologias sem fios que podem ser candidatas a serem usadas numa WBAN, como mostra a Tabela 2.7**.**

Tabela 2.7 – Características de tecnologias candidatas a uma WBAN (adaptado de [1]).

Tecnologia	Espectro	Modulação	Canais	Débito	Energia	Consumo	Topologia
Bluetooth	2,4 GHz	GFSK/PSK/8-DPSK	79	1-3 Mbps	~45mA/ 3,3 V	50 (nJ/b)	*Scatternet*
BLE	2,4 GHz	GFSK	40	1 Mbps	~28mA/ 3,3 V	92 (nJ/b)	*Piconet* (Estrela)
ZigBee	2,4 GHz	O-QPSK	16	250 kbps	~16,5mA/ 1,8 V	119 (nJ/b)	Estrela, Malha
ANT	2,4 GHz	GFSK	125	1 Mbps	~22mA/ 3,3 V	73 (nJ/b)	Estrela, Árvore ou Malha
Sensium	868 MHz 915 MHz	BFSK	16	50 kbps	~3mA/ 1,2 V	72 (nJ/b)	Estrela
Zarlink	402-405 MHz 433-434 MHz	2FSK/4FSK	10	200-800 Kbps	~5mA/ 3,3 V	21 (nJ/b)	P2P

Com o aparecimento da tecnologia sem fios BLE surgiram novos estudos e experiências de modo a medir o desempenho de WSNs ou WBANs baseadas nesta

tecnologia. Parâmetros como o débito, latência, tamanho da *piconet* e consumo energético são discutidos e avaliados em [29]. Os autores apresentam resultados teóricos e práticos em torno destes parâmetros que influenciam o desempenho do BLE. Em [30], alguns dos mesmos autores apresentam um estudo teórico intensivo acerca do débito máximo possível de alcançar numa ligação BLE, num ambiente de ausência de erros.

Em [31] é feita de igual modo uma avaliação a certos parâmetros do BLE (CC2540), IEEE 802.15.4 (CC2430) e SimpliciTI (CC2510) de modo a obter uma comparação entre os três protocolos estimando assim os seus desempenhos mediante as mesmas condições. Além dos testes de débito, neste trabalho os autores também conduzem testes de *turnaroud time* e gasto energético.

O consumo energético é uma das maiores preocupações na conceção de uma WSN, e o BLE foi criado com o propósito de contribuir para a diminuição dessa preocupação, tal como outras tecnologias como ZigBee/802.15.4. Para tal, em [32] os autores fazem uma análise profunda do consumo de energia do BLE através de medições em dispositivos reais e comparam com medições idênticas efetuadas em dispositivos ZigBee/802.15.4. Os autores concluem que de facto o BLE consome extremamente pouca energia, tendo um bom rácio de energia por bit transmitido, apesar de a *stack* usada apresentar algumas limitações. Os autores referem que a eficiência energética poderia ser melhorada caso no futuro o BLE permitisse o envio de um maior número de pacotes por evento de conexão e fosse implementando um mecanismo de saltos em frequência adaptativo (AFH – Adaptive Frequency Hopping) para combater a interferência. A versão da *stack* da Texas Instruments utilizada não implementa este mecanismo, fazendo com que os canais sob interferência não sejam detetados.

Em [33], é apresentado pela Texas Instruments um guião que descreve os procedimentos para medir o consumo energético num dispositivo a operar com escravo numa ligação BLE. É apresentado em detalhe a constituição de um evento de conexão bem como equações que traduzem a média da corrente gasta durante um

evento de conexão, todo o estado de conexão e a estimativa da duração de uma bateria.

2.6.3 Estudo da Postura do Atleta no Ciclismo

A postura de um atleta durante a atividade de ciclismo pode influenciar drasticamente o seu desempenho, se esta não for a mais correta. Existem diversos estudos nesta área de modo a prever e avaliar a postura durante o ciclismo em atletas de alta competição, ou simplesmente de ciclistas amadores.

Em [3], os autores compararam variáveis cardiovasculares e respiratórias de ciclistas não treinados e não fumadores com idades compreendidas entre os 17 e 25 anos, assumindo duas posições distintas aquando da prática do ciclismo: *upright* (posição mais elevada) e *aero* (posição aerodinâmica). Os resultados demonstram que na posição *upright* foi observado maior consumo de oxigénio, batimento cardíaco e carga máxima de trabalho.

Em [34], os autores optam por estimar as forças musculares e articulares sob a coluna cervical durante o ciclismo. Para tal, selecionaram 9 indivíduos, que se submeteram a testes que consistiam em pedalar nas posturas de descanso (esquerda), intermediária (centro) e de ataque (direita), tal como sugere a Figura 2.20.

Figura 2.20 - Diferentes posturas consideradas para os testes (adaptado de [34]).

As forças musculares e articulares foram calculadas utilizando um modelo biomecânico e equações de movimento. Uma análise da variância permitiu a comparação dos valores das forças nas diferentes posturas. Os resultados mostram que as magnitudes de ambas as forças aumentam significativamente à medida que a coluna cervical se torna mais estendida.

Além da postura do tronco, o desempenho dos ciclistas pode ser afetado se o ângulo do joelho não for o mais correto. Em [35], foram feitos testes anaeróbicos a ciclistas treinados e não treinados e concluiu-se que o assento deve estar regulado por forma a permitir um ângulo do joelho entre 25 a 35 graus para os ciclistas não treinados, contribuindo para a prevenção de lesões e aumento do desempenho.

Em [36], os autores afirmam que é essencial para qualquer ciclista otimizar o movimento do pedalar de modo a maximizar o desempenho e minimizar o risco de lesões, durante a atividade de ciclismo. Para isso propõem determinar em tempo real o movimento dos segmentos corporais inferiores (joelho e tornozelo) do atleta através de uma solução baseada numa rede de sensores sem fios. Os atletas têm acesso em tempo real ao *feedback* dado pelo sistema, podendo assim corrigir o movimento do seu pedalar. Neste estudo os autores comparam ainda o sistema desenvolvido com um já existente, baseado em capturas de imagem através de uma camara. Apesar de concluírem que estão bastante correlacionados, o sistema wireless apresenta vantagens nomeadamente de flexibilidade devido à não existência de cabos, facilidade de uso em ambientes *outdoor* e a capacidade de executar algoritmos, que garantem elevada precisão nas medidas, em micro processadores de baixa energia.

O Bike Fit [37] é um conceito que incide no facto de que todas as bicicletas devem ser ajustadas e adaptadas ao indivíduo, seja este um ciclista não treinado ou um ciclista perfeitamente apto para alta competição. Todos os ciclistas estão sujeitos a lesões e o Bike Fit contribui para o uso correto do equipamento como a altura do selim, altura dos pedais, altura do guiador dependendo da estatura do ciclista. O ajuste da postura não contribui apenas para prevenir lesões mas também para aperfeiçoar o desempenho do ciclista.

A TheBikeFit [38] é uma aplicação para smartphones compatíveis com iOS que contribui para o aperfeiçoamento da postura do atleta através da gravação de um vídeo com o atleta na postura que pretende avaliar, e posterior análise por parte de algoritmos apropriados para o efeito. Estes algoritmos calculam os ângulos de diversos segmentos corporais tais como joelho, braços, cotovelo, ombros, pés ou tronco. Após análise o utilizador pode partilhar as medições feitas nas redes sociais e até armazenar no serviço de *cloud* DropBox. A Figura 2.21 ilustra os diversos ângulos que podem ser medidos e uma medição ideal do ângulo do joelho.

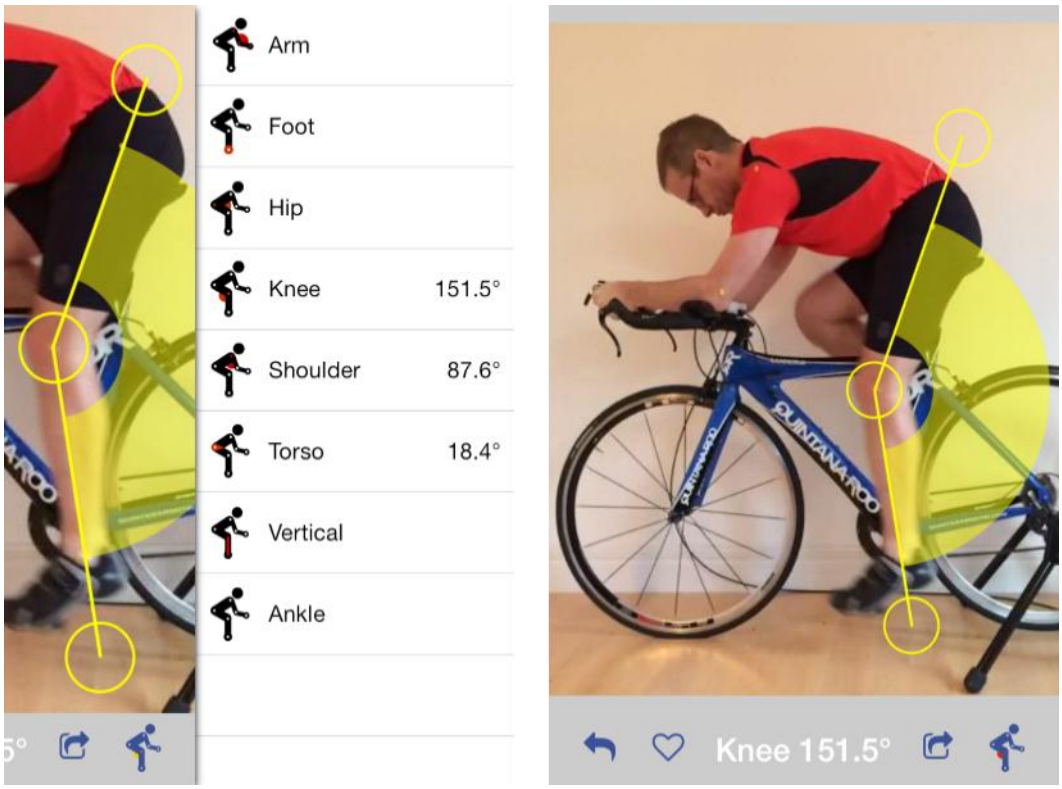

Figura 2.21 - Ângulos alvo de medição (esq.) e medição do ângulo ideal do joelho (dir) (adaptado de [38]).

3. Implementação do Sistema de Monitorização de Postura

Este capítulo tem como objetivo descrever uma proposta para a implementação do sistema de monitorização de postura. Primeiramente é feita uma descrição geral do sistema onde se refere todas as componentes afetas ao mesmo e descritas as plataformas de software e de hardware utilizadas. De seguida aborda-se os procedimentos que foram necessários para a implementação do sistema de monotorização, desde a descrição do firmware para comunicação entre as estações, aquisição dos dados sensoriais, formato da trama de dados e a migração para a plataforma Android.

3.1 Descrição Geral do Sistema

O sistema de monitorização de postura desenvolvido nesta dissertação é constituído por dois componentes físicos principais: smartphone (estação central) e os nós sensoriais, espalhados no corpo do atleta de ciclismo. A tecnologia usada para as comunicações é o BLE, constituindo o componente lógico do sistema. A Figura 3.1 ilustra a arquitetura do sistema de monitorização de postura.

Figura 3.1 - Arquitetura do sistema de monitorização de postura.

Cada nó sensorial tem a função de recolher os dados de movimento do segmento corporal em que está colocado. Os nós dividem-se em duas partes: módulo sensorial e módulo de interface rádio. O módulo sensorial [4], desenvolvido no Departamento de Eletrónica Industrial, é composto por um conjunto de 3 sensores, ou seja, acelerómetro, magnetómetro e giroscópio, que captam as variações em cada um dos seus 3 eixos *x, y, z*. As variações dos eixos dos sensores resultam das variações da força gravítica e do campo magnético, sendo que o acelerómetro permite detetar a inclinação através da diferença angular entre a posição do acelerómetro e o campo gravítico, o magnetómetro a rotação do corpo através da diferença angular entre a posição do magnetómetro e o campo magnético da Terra, e o giroscópio complementa o cálculo da inclinação e rotação, através da deteção de movimentos de rotação. Através do protocolo SPI (Serial Peripheral Interface), os dados lidos pelos sensores migram do módulo sensorial para o módulo de interface rádio, interface esta compatível com o BLE. A Figura 3.2 mostra a constituição do nó sensorial constituinte deste sistema.

Figura 3.2 - Constituição do nó sensorial (à esquerda o módulo sensorial e à direita o módulo de interface rádio).

Os dados dos nós sensoriais são então enviados via BLE para uma aplicação desenvolvida num smartphone, também este compatível com BLE (Bluetooth Smart

Ready). A aplicação efetua as funções básicas a uma conexão Bluetooth, como descobrir os nós sensoriais disponíveis e conectá-los ao smartphone. No que diz respeito às funções de monitorização, a aplicação é capaz de calibrar primeiramente cada um dos sensores, ativar a receção dos dados e, para cada um dos segmentos corporais onde os nós estão colocados, calcular os ângulos *Pitch*, *Roll* e *Yaw*.

Neste protótipo para monitorização de postura aplicado à monitorização de ciclistas, 3 nós sensoriais são distribuídos pelo corpo do indivíduo da seguinte forma: um no peito, de modo a medir a inclinação do tronco relativamente à bicicleta; um na parte superior da perna (coxa) e outro na parte inferior da perna (canela), de modo a obter o ângulo formado por ambos os segmentos. De modo a medir a inclinação da bicicleta (necessária para o cálculo da inclinação do tronco), optou-se por usar os sensores do smartphone, que é pousado no guiador da bicicleta, sendo uma posição ideal para esta medição, ao invés de colocar um outro módulo sensorial na bicicleta, usufruindo do sistema de sensores do smartphone e tirando assim mais partido do equipamento.

3.2 Plataforma BLE

A plataforma base de hardware para a implementação do módulo interface rádio BLE do nó sensorial é baseada no circuito integrado CC2540 da Texas Instruments [39]. Sendo este circuito integrado um SoC, tem embutido no mesmo chip um microcontrolador e um *transceiver*, apresentando assim uma grande vantagem em termos de dimensão no sistema final. O CC2540 opera na banda de frequências de 2,4 GHz e oferece uma taxa de transmissão de 1 Mbps.

Na Figura 3.3 está representado o CC2540DK (Development Kit) [40], que foi o *kit* de desenvolvimento usado para a implementação e testes do sistema. Este *kit* é composto por duas SmartRF05EB (a), duas antenas (b), dois módulos CC2540EM compatíveis com a tecnologia BLE (c), e um *dongle* USB CC2540 (d).

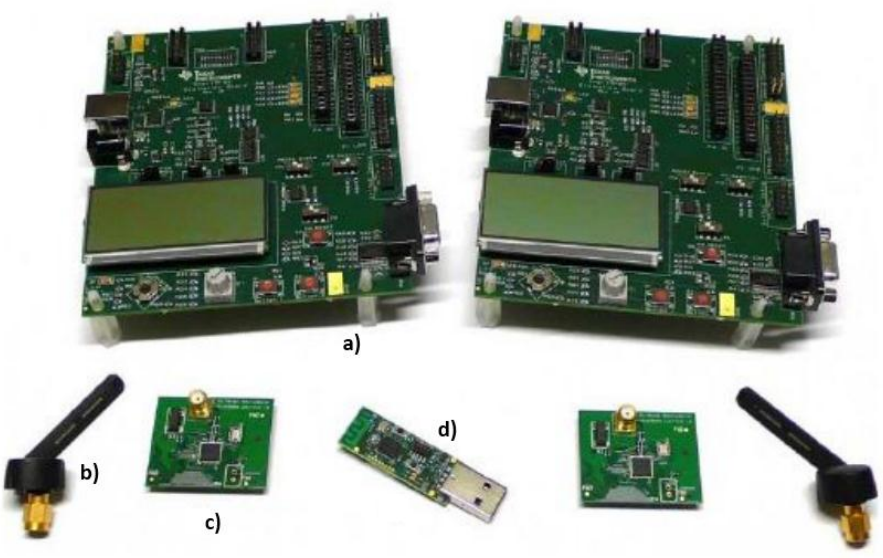

Figura 3.3 - Texas Instruments CC2540 Development Kit.

O CC2540, como foi referido, é um SoC de baixo custo e baixo consumo de energia desenhado para aplicações BLE. Possui um microcontrolador 8051 de alto desempenho e baixo consumo, 8 kB de memória RAM e 128 kB ou 256 kB de memoria *flash*, dependendo da versão. Neste caso em concreto o CC2540 utilizado tem 128 kB de memória *flash*.

Como periféricos principais possui um ADC de 12 bits com 8 canais e resolução configurável, temporizadores (*timers*) de uso geral de 16 e 8 bits, um temporizador de 32 kHz (*sleep timer*), duas USARTs com suporte para diversos protocolos série, um controlador DMA de 5 canais, um monitor de bateria e um sensor de temperatura.

A smartRF05EB [41] é a placa de desenvolvimento onde se conecta o módulo CC2540EM, permitindo o fácil acesso aos portos do mesmo e a coexistência de diversos componentes como LEDs, LCD, USB, UART, *joystick*, botões e potenciómetros.

O dongle USB CC2540 é usado como *sniffer*, capturando os pacotes que são trocados entre os diversos dispositivos. Este módulo permite uma fácil depuração, pois permite visualizar graficamente os dados presentes nos pacotes, a sua constituição e sequência temporal.

Relativamente aos módulos sensoriais usados neste sistema, estes foram desenvolvidos previamente no Departamento de Eletrónica Industrial da Universidade do Minho. Estes incorporam 3 sensores (acelerómetro, magnetómetro e giroscópio) de 3 eixos cada um. Cada amostra dos eixos tem uma resolução de 16 bits, ou seja, 2 bytes para cada eixo. Estes 3 sensores estão divididos em duas unidades distintas: MPU-6000 (Motion Processing Unit) da InvenSense [42], que contém o acelerómetro, o giroscópio e ainda um processador de movimento digital (DMP – Digital Motion Processor) com um barramento I2C auxiliar para servir de interface com um terceiro sensor. O intervalo de escala do acelerómetro é definido aquando da programação do mesmo, e pode ser de ±2g, ±4g, ±8g ou ±16g. A taxa de saída dos dados pode ser de 4 Hz a 1 KHz. Neste caso concreto usou-se intervalo de ±2g, que garante uma sensibilidade de 16,384 LSB/g. O barramento I2C auxiliar permite ao MPU-6000 aceder diretamente aos registos de um sensor externo, que no caso deste sistema é um magnetómetro IC HMC5883L, da Honeywell [43]. Este magnetómetro de alta resolução contém um ADC de 12 bits que consegue 5 milligauss de resolução em campos de ±8 gauss, o que garante um valor entre 1 e 2 graus de precisão de bússola.

Como está ilustrado na Figura 3.1, a entidade de hardware que atua como estação central é um smartphone. O dispositivo utilizado é um Google Nexus 5 com um processador de 2,26 GHz, 16 GB de memória interna e 2 GB de memória RAM. Vem equipado com a última versão do sistema operativo Android (v4.4) e, entre muitas outras características, com a versão 4.0 do Bluetooth, incluindo obviamente a vertente LE.

Para a edição do código C/C++, depuração e programação dos módulos CC2540 recorreu-se ao software IAR Embedded Workbench for 8051 [44]. Todo o código desenvolvido nesta fase baseou-se na implementação da pilha protocolar do BLE fornecida pela Texas Instruments, versão BLE-CC2540-1.3.2. A plataforma de desenvolvimento do código Android foi o Eclipse com o *plugin* ADT (Android Development Tools) [22], que permite adicionar ao Eclipse o SDK (Software Development Kit) do Android.

3.3 Desenvolvimento do Firmware

Numa fase primordial desta dissertação, o objetivo foi estabelecer a comunicação entre as duas estações. Para tal criou-se um perfil apropriado para o efeito, que disponibiliza um serviço de envio periódico de notificações da estação periférica para a estação central, ou seja, de escravo para mestre. O serviço está implementado na estação periférica tomando esta o papel de GATT Server (servidor). A estação central atua assim como GATT Client.

3.3.1 Criação do Serviço "Send Data"

O serviço criado, denominado "Send Data Service", contém apenas uma característica cujo valor é uma trama de 20 bytes e um descritor (CCC) com o intuito de ativar o modo de envio por notificações periodicamente (0x0001). As notificações, ao invés das indicações, são úteis para este caso, pois permitem o envio de dados do dispositivo periférico para o central sem que este os requisite e envie a confirmação de receção (ACK), pelo que o *overhead* é menor, tornando o processo de transmissão de dados mais eficiente. As permissões desta característica são apenas para notificações, e por isso o seu valor é afixado em 0x0010, de acordo com [16]. Na Tabela 3.1 é possível observar o serviço implementado.

Tabela 3.1 – Descrição do serviço "Send Data Service".

Handle	UUID	Descrição	Valor
0x0034	0x2800	Send Data Service	UUID = 0x2A2A
0x0035	0x2803	Car. 1: Info Sensores	UUID=0x2A2B Handle=0x0036 Perm = 0x0010
0x0036	0x2A2B	Valor	Trama 20 bytes
0x0037	0x2902	Descritor: CCC	0x0001

O desenvolvimento deste serviço foi baseado no serviço que a Texas Instruments fornece aquando da instalação da *stack* BLE, que está na aplicação "CC2540_SmartRF_SimpleBLEPeripheral". A aplicação para testar o dispositivo central, denominada "CC2540_SmartRF_SimpleBLECentral", também é fornecida. Com estas duas aplicações e o estudo do respetivo código em C/C++ foi possível observar o funcionamento da comunicação BLE e desenvolver o serviço para envio de dados de forma periódica.

Com recurso ao *software* BTool, o dispositivo central (GATT Client), ao conectar-se ao dispositivo periférico (GATT Server), descobre os serviços que este tem implementados. Através das operações que o BTool fornece é possível observar o serviço implementado, conforme mostra a Figura 3.4.

ConHnd	Handle	Uuid	Uuid Description	Value	Value Description	Properties
0x0000	0x0034			2A:2A		
0x0000	0x0035			10:36:00:2B:2A		
0x0000	0x0036			27:D7:FF:FF:FF:FF:FF:FF:FF:FF:F...		
0x0000	0x0037			01:00		

Figura 3.4 **- Visualização do serviço implementado através do BTool.**

3.3.2 Diagrama de Blocos do Firmware

Como foi referido, o *firmware* desenvolvido para o dispositivo periférico e central foi baseado em *firmwares* disponibilizados pela *stack* BLE-CC2540-1.3.2. Ambos já continham os procedimentos necessários para uma comunicação básica entre dois dispositivos BLE, e como tal o serviço "Send Data Service" foi criado no dispositivo periférico, ficando a receção e tratamento dos dados a cargo do dispositivo central. Na Figura 3.5 é possível observar o diagrama de blocos geral que descreve o processo de envio de informação, através de notificações, do dispositivo periférico para o dispositivo central.

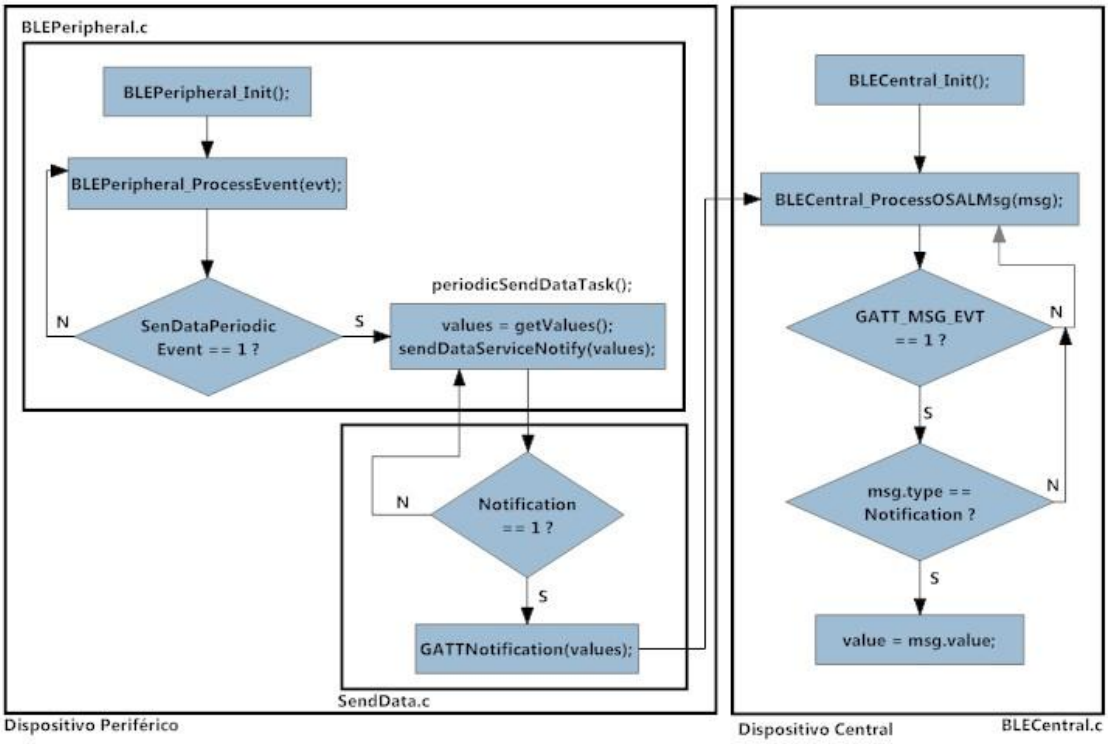

Figura 3.5 – Diagrama de blocos das aplicações do dispositivo periférico e central.

Relativamente ao dispositivo periférico, este inicia no bloco *BLEPeripheral_Init*. Este bloco é responsável pela inicialização de parâmetros da camada GAP no que diz respeito, por exemplo, ao papel que o dispositivo desempenha (neste caso periférico), definição do intervalo de anúncio e inicialização dos atributos da camada GATT, mais propriamente a inicialização dos serviços implementados. Na fase seguinte, *BLEPeripheral_ProcessEvent*, são processados todos os eventos que forem despoletados. Os primeiros são os eventos de inicialização e, no caso do *firmware* implementado, sempre que for detetado um evento de envio de dados periódico, é invocado o bloco *PeriodicSendDataTask*. Este bloco é responsável por obter os valores a enviar, que numa fase primordial são valores definidos pelo programador, e invocar assim o serviço criado, *SendDataServiceNotify*. Antes de proceder ao envio da informação da característica, o serviço avalia se as notificações estão ativadas e através da função *GATT_Notification,* envia a informação para o dispositivo central.

O dispositivo central efetua a sua inicialização através do bloco *BLECentral_Init*, mais propriamente a definição do papel de central no que diz respeito à camada GAP

e o papel de cliente no que diz respeito à camada GATT. É nesta fase que são ativadas as permissões para este dispositivo poder receber dados via notificações ou indicações. Tal como no dispositivo periférico, o dispositivo central processa também os eventos que são despoletados, mas também tem uma função que processas as mensagens resultantes da camada GATT, ou seja, através da função *BLECentral_ProcessOsalMsg*, caso a *flag GATT_MSG_EVT* seja ativada. Nesta fase são tratadas mensagens do tipo notificação, indicação ou escrita/leitura de atributos. No caso em concreto, se a mensagem for do tipo notificação, esta é tratada e recolhe-se o valor da mesma.

3.3.3 Software BTool

O software BTool mostrou-se uma ferramenta essencial nesta fase da dissertação pois é uma aplicação para PC fornecida pela Texas Instruments que permite estabelecer uma comunicação entre dois dispositivos BLE. Recorrendo a esta ferramenta é possível efetuar a procura de dispositivos, a sua conexão/término e observar toda a informação trocada entre os dispositivos. É também possível observar o conteúdo das características de um serviço implementado de forma a poder avaliá-lo com mais rigor. A leitura, escrita e receção de notificações das características é também possível caso existam permissões para tal. É útil para a deteção de possíveis erros e permite um grande controlo do dispositivo central [40]. A janela principal do *software* BTool está dividida em três principais secções: Informação do dispositivo, Centro de mensagens e Controlo do dispositivo. A secção "Informação do dispositivo" mostra a informação disponível acerca dos dispositivos conectados, como o endereço MAC, o identificador de conexão (*connection handle*) e o tipo de endereço (público ou privado). Como a ligação é feita via porta série para o PC, são também apresentadas todas as suas configurações, como número da COM, *baud rate*, mecanismo de controlo de fluxo, e bits paridade, paragem e de dados. Na secção "Centro de mensagens" é apresentado em tempo real a informação trocada entre mestre e escravo bem como as alterações dos parâmetros da conexão. Por último, na secção "Controlo do dispositivo" estão disponíveis todas as

funcionalidades de um dispositivo mestre, ou seja, descobrir e conectar vários escravos, definir os parâmetros de conexão, ler/escrever informação das/para as características e definições de segurança e emparelhamento.

O software BTool funciona recorrendo a um *firmware* fornecido pela Texas Instruments (CC2540_SmartRF_HostTestRelease_All.hex), que é inserido no dispositivo central (mestre), sendo este ligado via porta série ao PC, como se pode observar na Figura 3.6.

Figura 3.6 - Configuração do ambiente de testes com *software* BTool.

3.4 Aquisição dos Dados

Após concluída a implementação da rede BLE para transmissão de dados, transitou-se para a aquisição dos dados reais dos sensores, fornecidos pelo módulo sensorial. Esta aquisição foi feita através do protocolo SPI, que permitiu a comunicação entre a USART (Universal Synchronous/Asynchronous Receiver/Transmiter) do módulo de interface rádio e o módulo sensorial. Dentro do módulo sensorial, a comunicação entre o MPU-6000 (acelerómetro e giroscópio) e o HMC5883L (magnetómetro) foi feita através I2C (Inter-Integrated Circuit). Não foi necessário recorrer à implementação desta comunicação de raiz, visto que as configurações dos módulos sensoriais fornecidos já a englobavam. No entanto foi necessário implementar a comunicação via SPI, pois a troca dos dados realizou-se entre a USART do módulo de interface rádio BLE e o módulo sensorial. A Figura 3.7 ilustra o uso dos protocolos e comunicação entre periféricos para obtenção dos dados dos sensores.

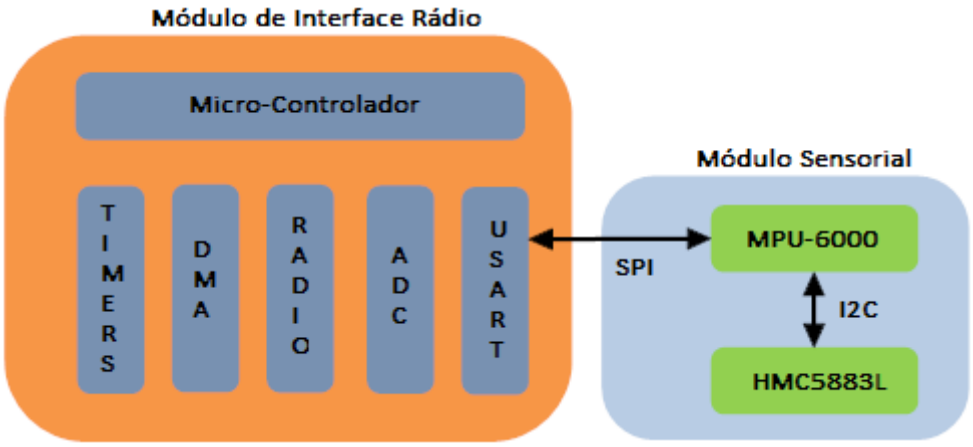

Figura 3.7 - Protocolos de comunicação entre periféricos.

O protocolo SPI é um protocolo de comunicação série síncrona entre microcontroladores e periféricos. Funciona no modo *full-duplex* (um canal para envio e outro para receção) e suporta velocidades superiores a 10 Mbps. Este protocolo especifica quatro sinais lógicos em que dois são de controlo e outros dois de dados. Os primeiros são o SS (Slave Select) e o CLK (Clock), e os segundos são o MOSI (Master Output Slave Input) e o MISO (Master Input Slave Output). A cada impulso de relógio, o mestre recebe um bit de dados na linha MISO e envia um bit na linha MOSI, sendo repetida esta comunicação (*full-duplex*) até finalizar a palavra a transmitir, tipicamente de 8 bits. Para se estabelecer uma comunicação SPI é necessário configurar os seguintes parâmetros: modo de operação, ou seja, mestre ou escravo; fase do relógio (normal ou invertida), polaridade do relógio (positiva ou negativa), ordem de transmissão (bit mais significativo em primeiro ou ultimo lugar) e a frequência do relógio. A Figura 3.8 ilustra o modelo geral para uma comunicação SPI.

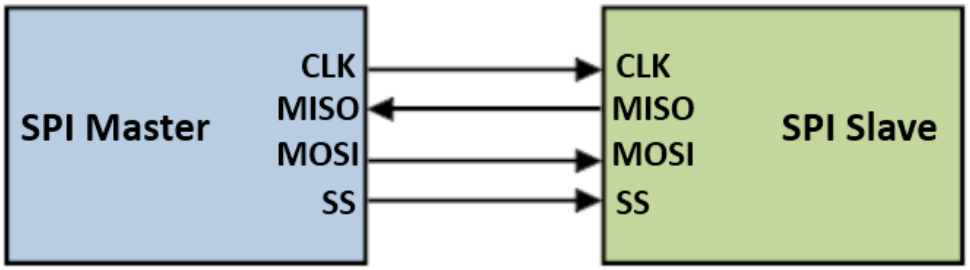

Figura 3.8 - Modelo de comunicação SPI.

Relativamente ao posicionamento do sinal de relógio em relação ao sinal de dados, existem 4 configurações possíveis, que combinam os parâmetros CPHA (fase do relógio) e CPOL (polaridade do relógio). Caso o CPHA seja igual a zero, significa que os dados são capturados na primeira transição do relógio; quando é igual a um, significa que são capturados na segunda transição do relógio. O CPOL indica a polaridade do sinal em repouso. Os dados são sempre recebidos pelo recetor na ordem contrária à qual foram transmitidos pelo emissor, fazendo com que haja estabilização dos dados durante meio período do relógio. A Figura 3.9 ilustra a combinação entre os parâmetros CPOL, CPHA e ordem do bit.

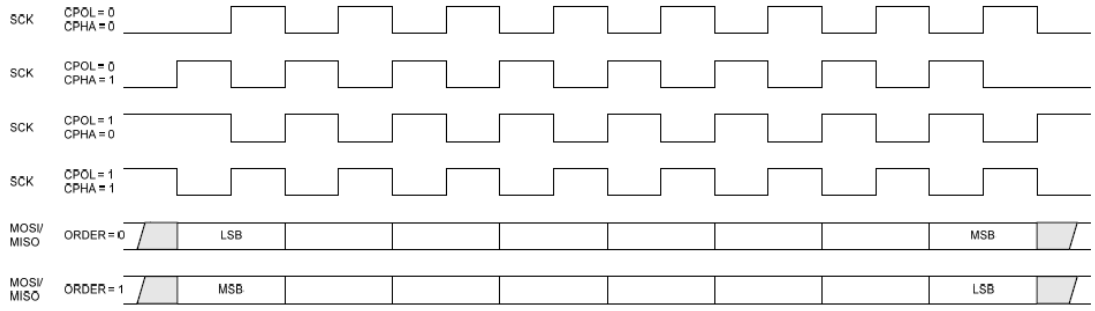

Figura 3.9 - Parâmetros CPOL, CPHA e ordem do bit (adaptado de [45]).

A comunicação SPI pode ser feita com base em três modos distintos: verificação por bits de estado, interrupção e DMA. Para a aquisição de dados reais optou-se pelo modo de verificação de bits de estado, que indicam quando os dados foram enviados e quando podem ser lidos. Neste modo os bits de estado são o UxCSR.Tx_BYTE e UxCSR.Rx_BYTE, podendo também recorrer-se ao bit UxCSR_ACTIVE. No dispositivo mestre, a ativação do bit UxCSR.Tx_BYTE pode ser usada para determinar quando foram enviados dados para o *buffer* UxDBUF. No escravo, a ativação do bit UxCSR.Rx_BYTE indica que os dados do *buffer* UxDBUF podem ser lidos.

De acordo com a especificação [42], o MPU-6000 atua sempre no modo escravo durante a comunicação mestre/escravo SPI, os bits são entregues na ordem do mais significativo para o menos significativo, e este suporta um frequência de relógio máxima de 1 MHz, polaridade de relógio negativa e fase do relógio invertida. As

operações de leitura concluem-se ao fim de dois ou mais bytes serem lidos ou escritos (no mínimo é necessário ler ou escrever dois bytes). O primeiro byte contém o endereço do registo e os restantes bytes contêm os dados. O primeiro bit do primeiro byte é responsável por indicar se é uma operação de leitura ou escrita, sendo "1" para leitura e "0" para escrita. Os restantes 7 bits servem para especificar o endereço do registo sob o qual se quer efetuar a operação.

Para se configurar a USART para usar a comunicação SPI, foi necessário escolher a localização dos pinos de I/O (Input/Output). O MPU foi configurado previamente para funcionar na alternativa 2 da USART0, e por isso, no módulo de interface rádio foram atribuídos os pinos P1 indicados na Tabela 3.2 aos sinais da comunicação SPI.

Tabela 3.2 - Localização dos pinos de I/O [45].

	USART0	
Alternativa 2	P1_2	SS
	P1_3	CLK
	P1_5	MOSI
	P1_4	MISO

As restantes configurações foram: modo mestre, polaridade do relógio negativa (U0GCR.CPOL = 0), fase do relógio invertida (U0GCR.CPHA=1), bit mais significativo transmitido em primeiro lugar (U0GCR.ORDER = 1) e frequência máxima de 1 MHz.

Como foi referido, a comunicação foi feita através da verificação de bits de estado, na qual se estabelece uma ligação *full-duplex* byte a byte consoante a mudança dos bits de estado. De notar que a transmissão se inicia quando o sinal SS se encontra a "0". Na Figura 3.10, é possível observar o diagrama do modo implementado para envio/receção dos dados.

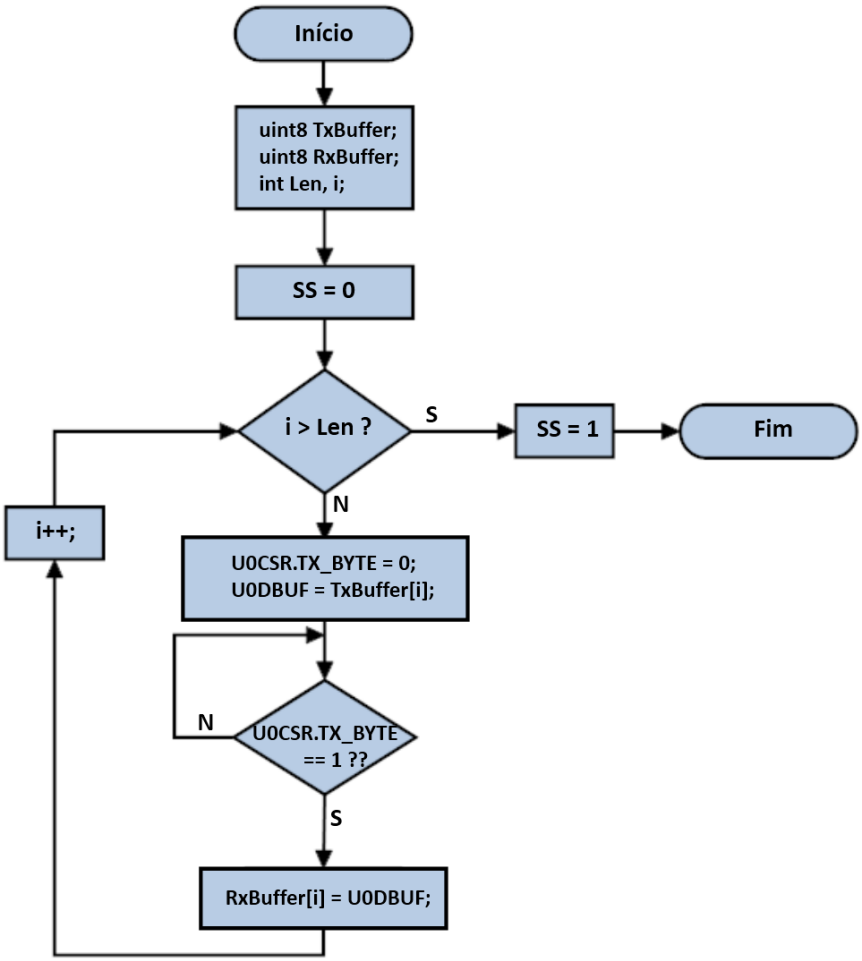

Figura 3.10 - Fluxograma de envio/receção via SPI (transferência por verificação de bits de estado).

No fluxograma, as variáveis *TxBuffer* e *RxBuffer* são *buffers* para transmitir e receber dados, respetivamente. A variável *Len* indica o número de bytes a serem lidos ou escritos. O início da comunicação para receção dos dados dá-se quando o sinal SS é colocado a zero coloca-se o bit U0CSR.TxByte a zero também antes de cada transmissão. No *buffer* U0DBUF coloca-se o primeiro byte a enviar e espera-se até que seja enviado. Quando o bit U0CSR.TxByte transitar para "1", significa que o byte foi transmitido e, como a ligação é *full-duplex*, guarda-se no *buffer* RxBuffer o conteúdo do U0DBUF. No caso da leitura dos sensores, o mestre envia primeiramente o endereço do registo, indicando a operação de leitura, e recebe uma resposta. De seguida, e até que tenha os dados de todos os sensores, envia bytes fictícios para que possa haver comunicação e, desta forma, possa receber byte a byte toda a informação dos sensores.

3.5 Formato da Trama de Dados

Após aquisição dos dados dos sensores por parte do módulo de interface rádio, via protocolo SPI, procede-se à definição e constituição da trama de dados a enviar para a estação central. Cada sensor é amostrado a 16 bits, ou seja, são necessários 2 bytes para representar os eixos *x*, *y* e *z* de cada sensor. Dado que os dados adquiridos via SPI dizem respeito a 3 sensores (acelerómetro, magnetómetro e giroscópio) são necessários 18 bytes no total para armazenar a informação dos mesmos. A cada amostra recolhida é adicionado um byte que contém um número de sequência (ID), de incremento automático, para possível validação no recetor, cálculo de débito e estimativa de perdas de pacotes. Por fim, de modo a distinguir a qual módulo sensorial a amostra pertence, à trama de dados é adicionado mais um byte contendo uma identificação *hard-coded*[2] do módulo sensorial (NS), ou seja, como existem 3 módulos sensoriais no sistema, esta identificação está compreendida entre 1 e 3. Posto isto, o tamanho total da trama é de 20 bytes. A Figura 3.11 ilustra a constituição da trama de dados com toda a informação referida anteriormente. Tal como foi frisado na secção 2.4.9.2, o payload máximo ao nível aplicacional, para transmissão de notificações via BLE é de 20 bytes, cabendo assim a trama de dados nesse tamanho.

Bytes	1	2	2	2	2	2	2	2	2	2	1
	ID	AccX	AccY	AccZ	MagX	MagY	MagZ	GyrX	GyrY	GyrZ	NS

Figura 3.11 - Trama de dados da interface rádio.

[2] Termo geralmente usado para referir que variáveis assumem valores pré-definidos pelo programador, em vez de tomarem valores obtidos por processamento.

3.6 Aplicação Android Desenvolvida

A aplicação desenvolvida, denominada de “Cycling Kinect”, tem como objetivo monitorizar a postura do atleta de ciclismo durante uma prova de treino ou de competição. A aplicação é executada no smartphone, que assume o papel de estação central e serve de interface com o utilizador, neste caso, o atleta. Recebe periodicamente a informação coletada pelos nós sensoriais, presentes em determinados segmentos no corpo do atleta, e com esses dados, após tratamento adequado, calcula a postura. Nesta fase de desenvolvimento, a aplicação não tem como finalidade indicar se o atleta em determinado momento apresenta ou não a postura correta, mas possui capacidade de indicar em tempo-real e armazenar num histórico, para futura análise, a inclinação da bicicleta, a inclinação do tronco e o ângulo do joelho do atleta, ao longo do percurso.

Ao ser iniciada, a aplicação interroga a rede à procura dos dispositivos (nós sensoriais) disponíveis, e procede ao estabelecimento da conexão. Após a conexão, é possível obter informação do nó sensorial e ativar a receção dos dados. Com os nós sensoriais devidamente colocados nos segmentos corporais corretos e calibrados, procede-se ao início de uma sessão de treino/competição, onde são calculados os valores dos parâmetros medidos, referidos no parágrafo anterior. Assim que o atleta quiser interromper a sessão, pode visualizar, através de um gráfico, o comportamento dos parâmetros medidos durante a sessão, incluindo a inclinação da bicicleta ao longo do percurso.

O diagrama presente na Figura 3.12 ilustra o fluxo da aplicação em termos de funcionalidades, através da representação dos blocos constituintes.

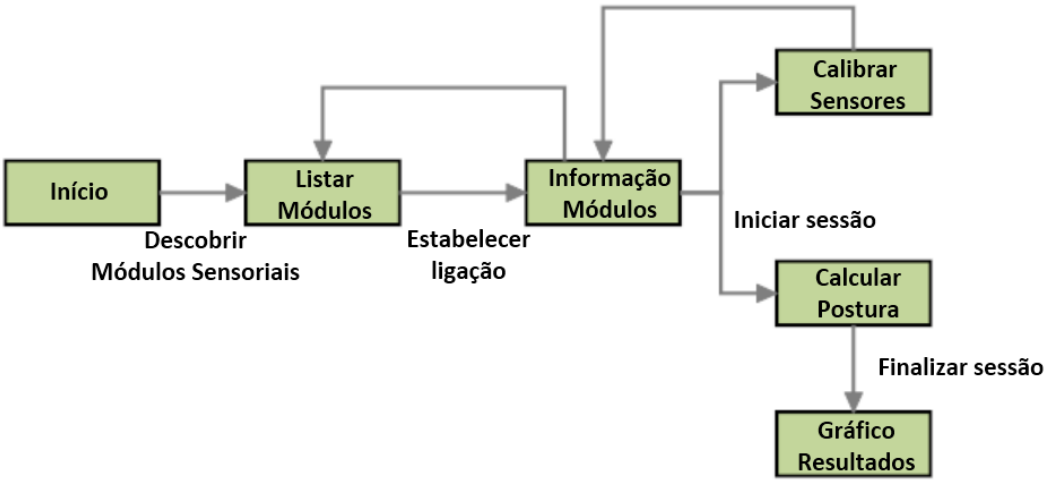

Figura 3.12 - Principais blocos da aplicação.

Os blocos apresentados na Figura 3.12 correspondem às principais activities constituintes do sistema. O bloco inicial, "Início", corresponde à activity principal "MainActivity" e serve de interface inicial da aplicação. O bloco intitulado de "Listar Módulos" corresponde à activity "DevicesList", onde são apresentados todos os módulos sensoriais encontrados via BLE. Após estabelecimento da ligação a activity que contém a informação do módulo é a "InfoSensor", que corresponde ao bloco "Informação Módulos". Os blocos "Calibrar Sensores" e "Calcular Postura" correspondem às activities "Calibrate" e "Posture", respetivamente. Por fim o bloco "Gráfico Resultados" engloba as activities "Results" e "Graph", tendo como objetivo ilustrar os resultados do cálculo da postura. Nas secções 3.6.2, 3.6.3 e 3.6.4 são descritas em mais detalhe as activities referidas, bem como o seu funcionamento.

3.6.1 Requisitos da Aplicação

A aplicação, sendo desenhada para ser executada na estação central, tem de obedecer a certos requisitos típicos de uma WSN e, claro, a requisitos ao nível do sistema de monitorização de postura em questão.

No que diz respeito a aspetos funcionais desejados para uma WSN, a aplicação dever: ser escalável, permitindo o suporte para vários nós sensoriais; suportar um débito necessário para entrega dos dados gerados pela aplicação; garantir fiabilidade

na transmissão dos dados num contexto de tempo-real; garantir desejavelmente um baixo consumo energético, que dependerá sempre da quantidade dos dados enviados, da frequência a que são enviados e também do número de nós conectados à estação central.

No que diz respeito aos requisitos da aplicação inerentes ao SMP, esta deve ser capaz de se conectar via BLE aos módulos sensoriais disponíveis e receber periodicamente a suas informações. Para o correto funcionamento e exatidão dos valores medidos pelos sensores de cada módulo, é necessário recorrer à sua calibração, que passa por registar os valores máximos e mínimos de cada eixo de cada sensor do módulo de forma independente. Estando todos os módulos sensoriais afetos ao sistema conectados e os respetivos sensores de cada um devidamente calibrados, a aplicação deve ser capaz de calcular a orientação dos diversos segmentos corporais monitorizados (ver em mais detalhe secção 3.6.4) e apresentar em tempo-real ao utilizador as variações nos ângulos medidos. O cálculo da postura é iniciado quando o atleta começa a sua sessão de treino livre ou competição, e, após a atividade terminar, o atleta deve ser capaz de visualizar o seu desempenho ao longo da sessão, através de diferentes traçados num gráfico.

3.6.2 Utilização da API BLE para Android

Conforme referido anteriormente, a tecnologia de comunicação utilizada na rede de sensores do sistema de monitorização de postura é o BLE. Esta tecnologia tem a vantagem de ser compatível com o hardware e software da maioria dos smartphones mais recentes. Apesar de ser uma tecnologia relativamente nova, verifica-se a tendência de estar a convergir no segmento das WSN para uso em diversas áreas.

Sendo a aplicação desenvolvida no sistema operativo Android e os nós da rede também compatíveis com esta tecnologia, a Google disponibiliza a API necessária para implementar os métodos e funcionalidades de descoberta de dispositivos, estabelecimento de conexão e ativação de notificações para receção dos dados.

A estação central, que executa a aplicação, tem o papel de mestre e os nós sensoriais desempenham o papel de escravos. Relativamente aos papeis definidos pela camada GATT, os nós sensoriais desempenham o papel de GATT Server (servidores), pois contêm a implementação do serviço e todos os dados a serem enviados para a estação central; enquanto o smartphone (estação central) desempenha o papel de GATT Client (cliente), pois recebe periodicamente os dados recolhidos pelos nós sensoriais, ou seja, usufrui do serviço implementado no GATT Server. A Figura 3.13 mostra a distinção entre os papéis desempenhados por ambas as entidades intervenientes do sistema.

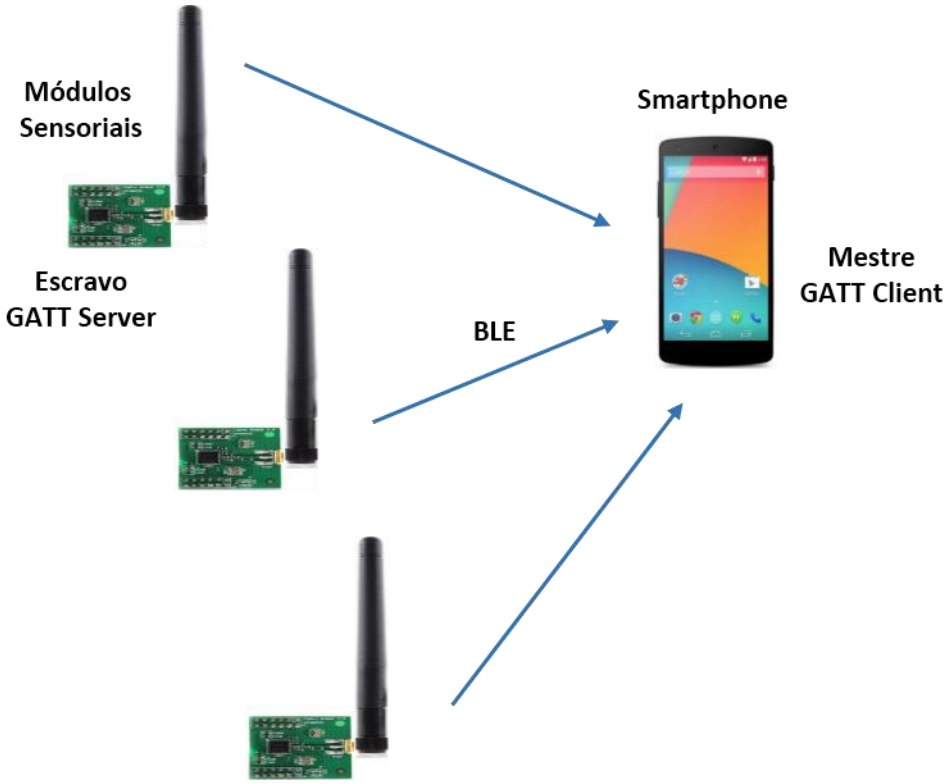

Figura 3.13 - Papeis das entidades do sistema.

Para a implementação da tecnologia BLE na aplicação, ou seja, GATT Client, recorreu-se a um conjunto de métodos já disponibilizados numa aplicação base, fornecida pela Google, através do portal Android Developers. Esta aplicação consiste em diversas activities para ativação do Bluetooth através de uma aplicação, descoberta de dispositivos compatíveis, estabelecimento da conexão a um GATT Server, e receção de dados sob a forma de notificações (GATT Notifications). Os

métodos inerentes a estas tarefas estão incluídos um serviço, denominado de *BluetoothLeService.*

Para usufruir das capacidades do Bluetooth na aplicação, foi necessário declarar as respetivas permissões no manifesto[3] da aplicação. Sem estas permissões não seria possível efetuar qualquer comunicação Bluetooth, como requisitar conexão, aceitar conexão e transferir dados. Foi declarada uma permissão específica para permitir que apenas fossem encontrados dispositivos compatíveis com o BLE. A Figura 3.14 representa as permissões implementadas no manifesto.

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>
```

Figura 3.14 - Permissões para o Bluetooth.

Antes de ser possível comunicar por BLE tem de se proceder à ativação do mesmo através da aplicação. Após esta ser iniciada, o menu principal apresenta o botão *"Find Devices"*, para proceder à descoberta dos nós sensoriais disponíveis para estabelecer conexão. É neste momento que a aplicação interroga ao utilizador se quer ativar o Bluetooth ou simplesmente abandonar a aplicação. A Figura 3.15 ilustra dois ecrãs que representam duas activities, sendo elas a activity principal "MainActivity" e a activity de início de descoberta de dispositivos, "DevicesList".

[3] O manifesto de uma aplicação Android é um ficheiro XML que contém toda a informação da aplicação, de modo a dar-se a conhecer ao sistema.

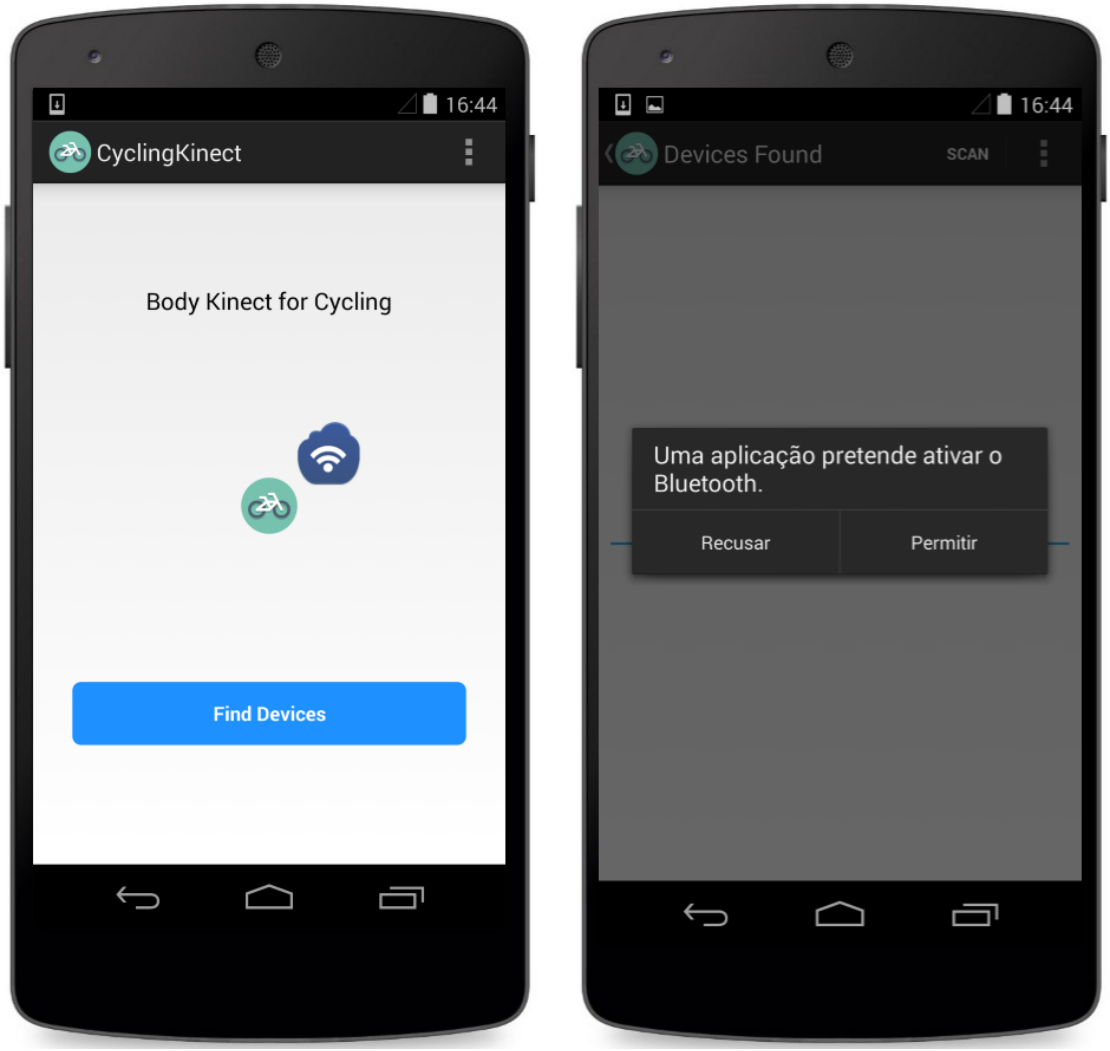

Figura 3.15 - Ecrã principal da aplicação (esq.) e pedido de ligação Bluetooth (dir.).

O objeto *BluetoothAdapter* é necessário em todas as *activities* que usufruam do Bluetooth. Representa o adaptador Bluetooth (interface rádio) do próprio dispositivo, e é através deste que a aplicação pode interagir com o Bluetooth.

Após ativação da interface Bluetooth, segue-se a descoberta de dispositivos. Este processo é feito recorrendo ao método *startLeScan()*, pertencente ao objeto *BluetoothAdapter.* Como o processo de descoberta é um processo exaustivo para a bateria, este fica automaticamente suspenso após 4 segundos, tempo suficiente para encontrar a quantidade de nós sensoriais constituintes do sistema. Caso se queira continuar a procurar dispositivos, aciona-se novamente o botão "SCAN" da activity "DevicesList". Esta activity apresenta, sob a forma de lista, o nome de cada

dispositivo compatível com a tecnologia BLE encontrado, seguido do seu endereço MAC. A Figura 3.16 ilustra em exemplo de descoberta de três dispositivos, neste caso, três nós sensoriais pertencentes ao sistema de monitorização de postura.

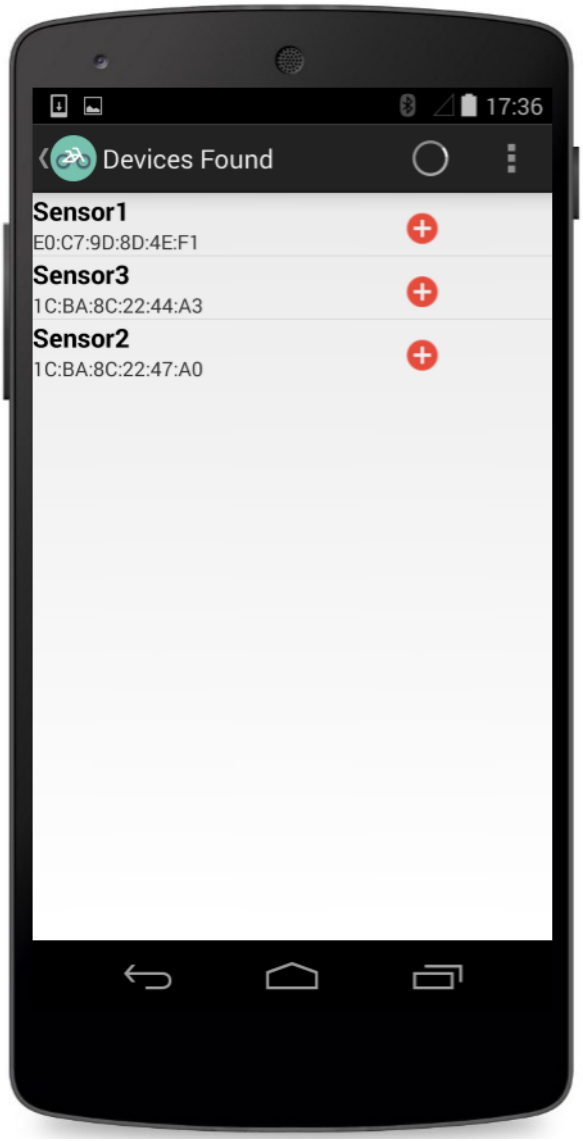

Figura 3.16 - Listagem de 3 sensores encontrados e respetivos endereços MAC, via BLE.

Após concluída a fase de descoberta dos dispositivos, passa-se para o estabelecimento de conexão, ou seja, a conexão entre o GATT Client e GATT Server. Este processo é feito através do método *connectGatt().* Este método retorna uma instância do objeto *BluetoothGatt*, que permite efetuar diversas operações do lado do cliente. A activity *"InfoSensor"* é onde se dá a conexão, se apresenta os dados recebidos e os serviços e respetivas características suportados pelo GATT Server.

Esta *activity* comunica com o serviço da aplicação, *BluetoothLeService,* que, como já foi referido, contém a API BLE necessária para interagir com o dispositivo.

O procedimento de receção dos dados na aplicação (GATT Client) é sob a forma de notificações periódicas. Na *activity*, ao surgir os serviços e características disponíveis no GATT Server, é possível ativar a receção de notificações para uma certa característica, que tem de conter as permissões no GATT Server. Este procedimento é feito através do método *setCharacteristicNotification().* No caso do serviço implementado para envio de dados, "Send Data Service", a única característica que contém a informação recolhida pelos sensores tem permissões apenas para notificações. O utilizador (atleta), ao efetuar a conexão com o sensor, tem então de seguida de ativar as notificações para essa característica para assim poder receber os dados. A Figura 3.17 ilustra a activity "InfoSensor", com as respetivas informações do GATT Server.

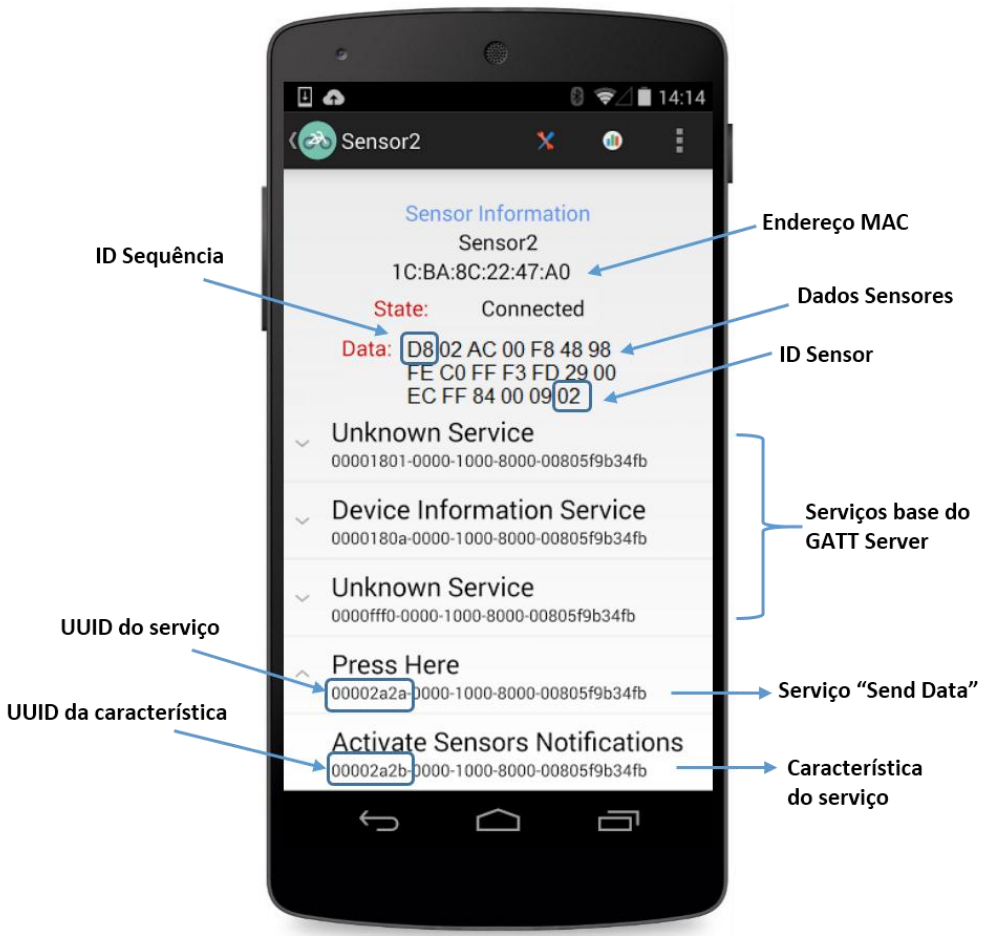

Figura 3.17 - Activity "InfoSensor" com a informação sobre o GATT Server.

A activity "InfoSensor" fornece indicações acerca do sensor conectado, do estado da conexão e dos dados recebidos. É possível observar que o número de bytes recebidos é igual a 20, em que o primeiro é o número de sequência, seguido dos dados recolhidos pelos sensores (18 bytes), e por último o identificador do sensor. Além do serviço implementado, a aplicação base para a implementação do GATT Server conta com serviços pré-definidos que são necessários para o funcionamento do mesmo, não podendo ser excluídos. O serviço implementado, "Send Data Service", está representado com o UUID igual a 0x2a2a, e apresenta o valor da característica, com UUID igual a 0x2a2b, tal como foi definido na Tabela 3.1.

De modo a permitir uma interface intuitiva para com o utilizador, a descrição do serviço e ativação da receção das notificações do mesmo foi alterada para uma descrição mais amigável.

3.6.3 Calibração dos Módulos Sensoriais

Antes do sistema e respetiva aplicação serem usados por um atleta, é necessário recorrer à calibração dos módulos sensoriais de forma independente. Este método consiste em obter os valores máximos e mínimos das leituras do acelerómetro, magnetómetro e giroscópio, ao longo dos eixos *x*, *y* e *z,* reportando-os para um ficheiro específico. Cada módulo, sendo a primeira vez conectado ao sistema, é submetido a este processo, sendo gerado um ficheiro de calibração para o mesmo. Após o ficheiro ser criado, há possibilidade de criar um novo, caso a calibração tenha sido feito de modo errado. Os cálculos da postura futuros irão ser sempre feitos com base no último ficheiro de calibração gerado para o sensor em questão. A Figura 3.18 ilustra a activity "Calibration" com o respetivo painel de calibração dos diferentes sensores.

Inicialmente o painel apresenta apenas o valor atual a alternar de forma constante. Por defeito, apresentam-se os valores máximos e mínimos estáticos para cada eixo e, à medida que se roda o módulo sensorial sobre o eixo pretendido, fixa-se o valor máximo e mínimo de cada um ficando estes a cor vermelha.

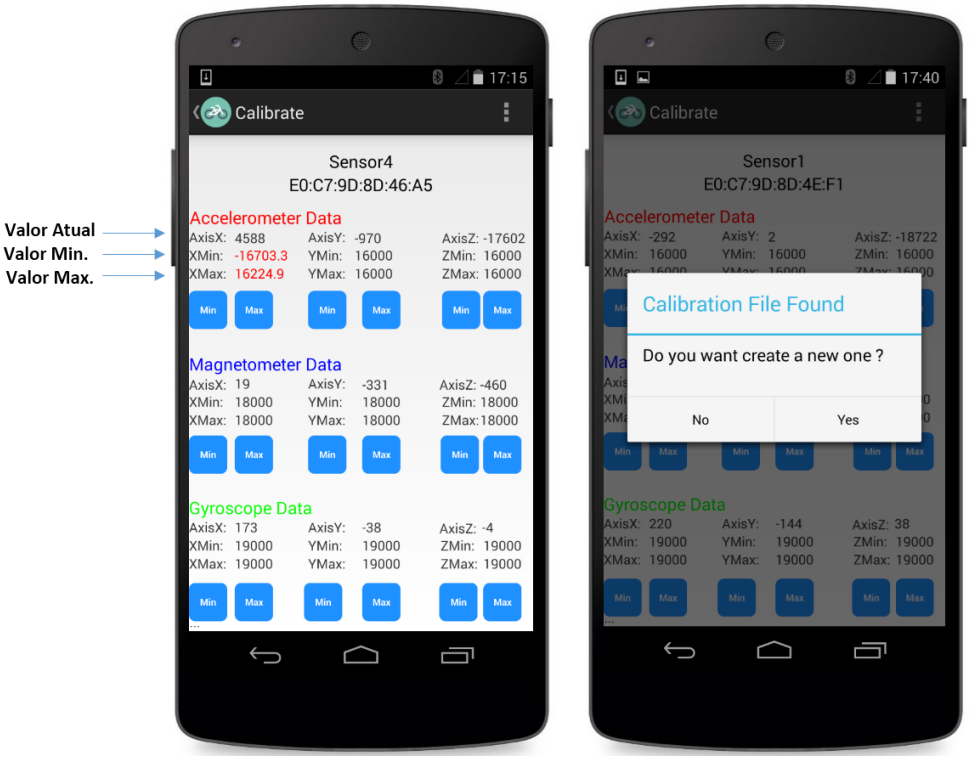

Figura 3.18 - Painel de calibração dos sensores (esq.) com a hipótese de alterar ficheiro criado previamente (dir.).

O exemplo da Figura 3.18 ilustra especificamente a calibração do eixo *x* do acelerómetro, onde é possível observar o valor atual compreendido entre o valor máximo e mínimo registado.

3.6.4 Cálculo da Postura

Após calibração de todos os módulos sensoriais pertencentes ao sistema, estão reunidas as condições para proceder ao cálculo da postura nos segmentos corporais pré-definidos. O atleta inicia uma sessão de treino/competição onde regista o seu nome e descrição da sessão, de modo a permitir uma futura construção de um histórico de sessões. A Figura 3.19 ilustra o início da activity “Posture”.

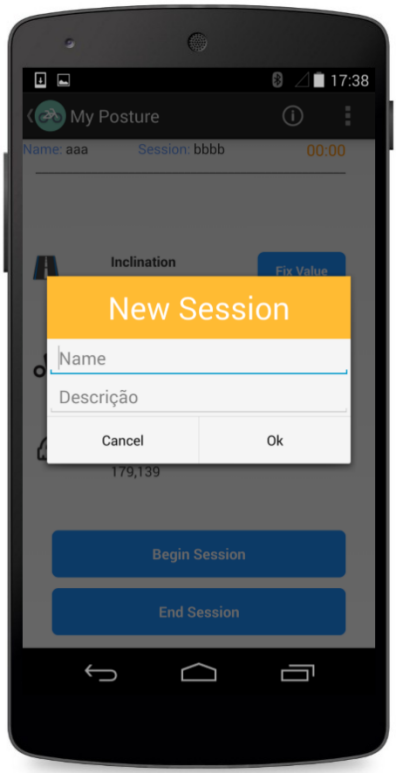

Figura 3.19 - Início da sessão para cálculo da postura.

Os três ângulos que descrevem a orientação dos nós sensoriais são o *Pitch*, o *Roll* e o *Yaw*. O *Pitch* e o *Roll* são calculados através das variações de um vetor normalizado do acelerómetro em cada eixo (a_x, a_y, a_z), enquanto o *Yaw* é obtido através das componentes horizontais (X_h e Y_h) do campo magnético terrestre. As equações que representam os ângulos *Pitch*, *Roll*, e *Yaw,* baseadas nas equações apresentadas em [26], são (1), (2), e (3), respetivamente.

$$Pitch = arctan\left(\frac{a_y}{\sqrt{a_x^2+a_z^2}}\right) \quad (1)$$

$$Roll = arctan\left(\frac{-a_x}{-a_z}\right) \quad (2)$$

$$Yaw = arctan\left(\frac{X_h}{Y_h}\right) \quad (3)$$

O sistema de coordenadas usado para calcular a orientação dos ângulos está representado na Figura 3.20. A parte frontal da orientação do módulo é sobre o eixo

do y, sendo que o *Pitch* define a rotação sobre o eixo *x*, o *Roll* sobre o eixo *y* e o *Yaw* sobre o eixo *z*.

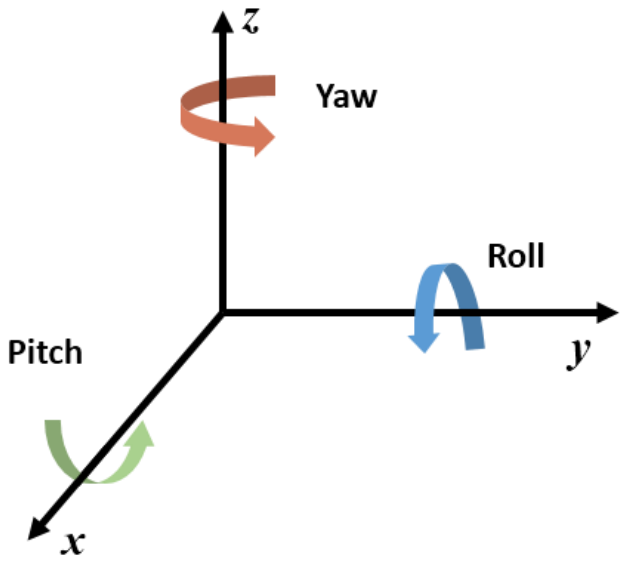

Figura 3.20 - Representação do sistema de eixos.

Um dos módulos sensoriais é colocado no tronco do atleta de modo a medir o ângulo de inclinação do mesmo. Nesta fase são definidas duas posições de referência: posição ereta (tronco perpendicular ao solo) e posição aerodinâmica (tronco inclinado para a frente). Para auxiliar no cálculo desta medida, recorreu-se ao cálculo do ângulo de inclinação da bicicleta através do sistema de sensores do smartphone. Este processo é necessário, pois permite indicar a inclinação real do tronco do atleta durante uma subida ou descida, ou seja, ao valor obtido em cada momento para a inclinação do tronco ($A_{ChestInc}$) é necessário subtrair a inclinação medida da bicicleta ($A_{BicicleInc}$). A equação (4) traduz o ângulo real da inclinação do tronco ($A_{RealChestInc}$).

$$A_{RealChestInc} = A_{ChestInc} - A_{BicicleInc} \quad (4)$$

De notar que, de modo a obter a inclinação correta do smartphone quando este é pousado no suporte do guiador, é necessário recorrer também à calibração do mesmo. Esta calibração consiste em subtrair constantemente o valor da inclinação ao valor registado aquando da colocação do equipamento no suporte, assumindo

que a bicicleta se encontra numa posição horizontal, onde o terreno é plano. Para recorrer a esta técnica existe um botão apropriado para o efeito acessível ao utilizador, na activity "Posture".

O ângulo de orientação necessário nesta fase, para determinação tanto da inclinação do tronco, como da bicicleta, é apenas o *Pitch*.

Os restantes dois módulos sensoriais constituintes do sistema de monitorização são colocados de modo a medir o ângulo do joelho. Para isso, um é colocado na parte superior da perna (coxa), enquanto o outro é colocado na parte inferior da perna (canela), simulando assim o efeito de um goniómetro.

O ângulo entre os segmentos corporais referidos é dado pela equação (5).

$$A_{Knee} = arcos\ \frac{\vec{V}_{upperLeg} \cdot \vec{V}_{lowerLeg}}{\|V_{upperLeg}\| \|V_{lowerLeg}\|} \tag{5}$$

Para o cálculo dos vetores $V_{upperLeg}$ e $V_{lowerLeg}$, são retirados os valores do *Pitch*, *Roll*, *Yaw* de cada de cada módulo sensorial em cada momento, e multiplicados pela matriz rotação em cada eixo. Desta forma obtém-se a orientação de cada módulo sensorial. As matrizes de rotação para cada eixo estão representadas por $R_X(\alpha)$, $R_Y(\theta)$ e $R_Z(\tau)$, em (6),(7) e (8) respetivamente. Os valores de α, θ e τ representam o ângulo de rotação sobre o eixo *x, y* e *z*, respetivamente.

$$R_X(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{6}$$

$$R_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{7}$$

$$R_Z(\tau) = \begin{bmatrix} \cos\theta & -\sin\tau & 0 \\ \sin\tau & \cos\tau & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

A Figura 3.21 ilustra a monitorização da postura de um atleta, referindo os ângulos medidos nos segmentos corporais, em graus.

A uma inclinação da bicicleta durante uma subida corresponde um ângulo positivo, enquanto uma descida apresenta valores angulares negativos. No que diz

respeito ao ângulo de inclinação do tronco, um reclinar para a frente significa valores positivos e um reclinar para trás valores negativos. De notar que o tronco totalmente esticado apresenta um ângulo de 0 graus e à medida que inclina para a frente ou para trás apresenta valores entre 0 e 90 graus ou 0 e -90 gaus, respetivamente.

O ângulo do joelho apresenta o valor máximo de 180 graus quando a flexão da perna é nula, ou seja, totalmente esticada, sendo que à medida que o atleta flete a perna, o valor do ângulo diminui.

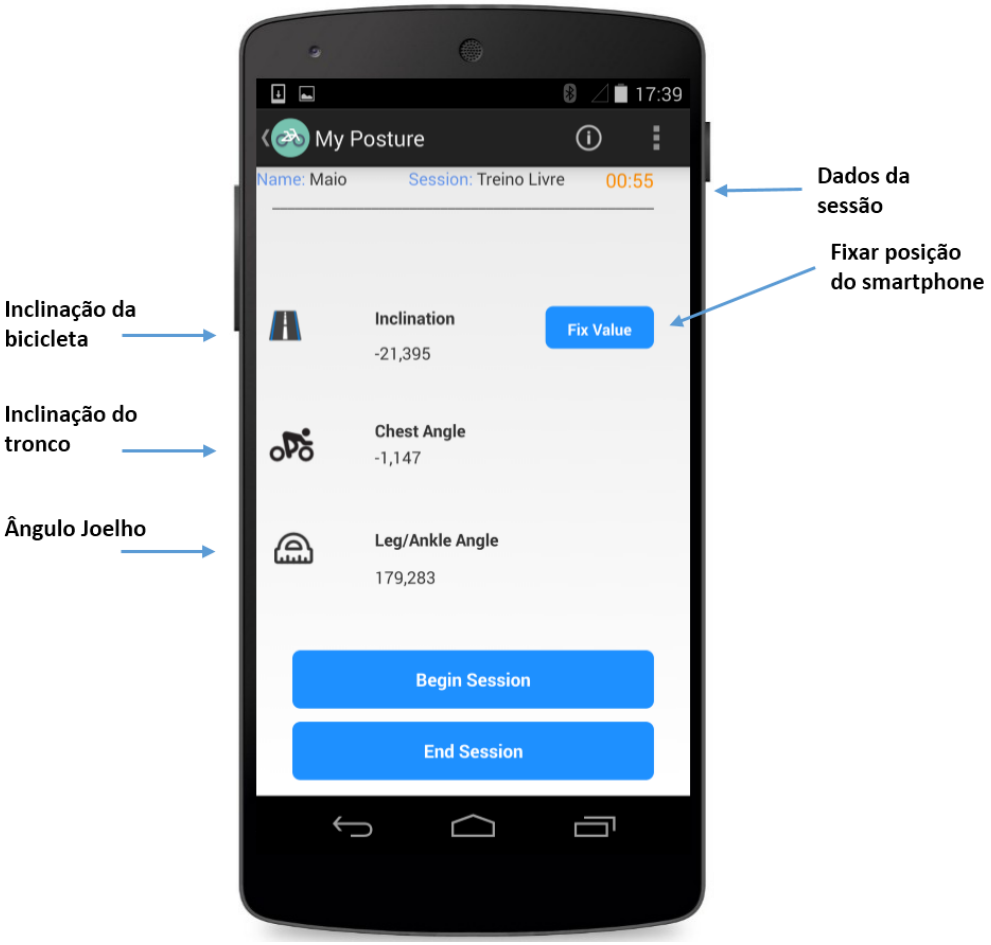

Figura 3.21 - Interface de monitorização da postura.

Os dados da postura patentes no exemplo da Figura 3.21 não correspondem a uma real medição efetuada numa atividade de ciclismo, apenas pretendem mostrar nesta fase o aspeto da interface para o utilizador e como os dados lhe chegam. Para este exemplo pousou-se apenas o módulo sensorial, numa posição de 0 graus, no selim da bicicleta e inclinou-se a mesma no sentido descendente, daí o valor

negativo. O ângulo da inclinação do tronco (Chest Angle) apresenta um valor negativo, porém muito próxima de zero pois apesar de ser colocado numa posição angular nula, a sensibilidade do acelerómetro definida e a taxa de amostragem dos dados causam uma constante variação, mesmo com movimentos e vibrações pouco significativas.

Após o término da sessão de treino/competição, o atleta pode visualizar os seus resultados sob a forma de gráfico. Para o efeito foi importada para o projeto a libraria *AChartEngine* [46], que permite a inclusão de gráficos nas aplicações Android. A Figura 3.22 ilustra um exemplo de variações da postura obtidas ao longo do tempo, onde a linha vermelha representa a inclinação da bicicleta, a linha verde a inclinação do tronco (peito) e a linha azul a ângulo do joelho, apesar desta última não estar visível neste exemplo.

Demonstrações reais do funcionamento dos sensores aplicadas ao ciclismo são apresentadas com mais detalhe na secção 5.7, no capítulo dos Resultados e Discussão.

Figura 3.22 - Exemplo de um resultado de uma sessão de treino/competição.

4. Análise do Protocolo BLE

Este capítulo pretende fornecer uma análise pormenorizada do protocolo de transmissão BLE. Grande parte da informação sobre esta tecnologia foi descrita na secção 2.4, como a descrição da pilha protocolar e formato dos pacotes. Desse modo, o objetivo principal deste capítulo é analisar os dois tipos de eventos existentes numa comunicação BLE, a latência da comunicação, o débito máximo entre dispositivos, o consumo energético durante o estado de conexão e o número de escravos suportados numa ligação BLE, comparando com o protocolo LPRT.

4.1 Eventos de Anúncio

Os eventos de anúncio são usados para transmitir dados nos canais de anúncio, pelo que, no início de cada evento de anúncio, o dispositivo que deseja transmitir dados (anunciante) envia um pacote de anúncio. Estes eventos são os definidos na secção 2.4.9.1. Cada evento é composto por um ou mais PDUs de anúncio enviados num canal de anúncio específico. Dependendo do tipo de evento de anúncio, certos PDUs vão ser permitidos como resposta, como ilustra a Tabela 4.1. Para cada tipo de evento de anúncio corresponde um PDU específico de canal de anúncio.

Tabela 4.1 - PDUs de anúncio e respetivos PDUs de resposta (adaptado de [15]).

PDU enviado	PDU de resposta	
	SCAN_REQ	CONNECT_REQ
ADV_IND	Sim	Sim
ADV_DIRECT_IND	Não	Sim
ADV_NONCONN_IND	Não	Não
ADV_SCAN_IND	Sim	Não

Nos casos do PDU ser do tipo ADV_IND ou ADV_SCAN_IND, o dispositivo que escute o canal de anúncios sem intenção de se conectar ao anunciante (*scanner*) pode requisitar mais dados do anunciante através do envio do SCAN_REQ PDU. No caso de os PDUs serem do tipo ADV_IND ou ADV_DIRECT_IND, o dispositivo que escuta o canal com a intenção de se conectar (iniciador) e trocar dados com o anunciante pode enviar o CONNECT_REQ PDU.

Para todos os eventos de anúncio indiretos (ADV_IND,ADV_NONCONN_IND ou ADV_SCAN_IND), o tempo entre dois eventos de anúncio consecutivos ($T_{advEvent}$) é expresso pela equação (9).

$$T_{advEvent} = T_{advInterval} + T_{advDelay} \quad (9)$$

A variável $T_{advInterval}$ deve ser no mínimo igual a 100 ms para os eventos do tipo ADV_SCAN_IND e ADV_NONCONN. Para o caso do ADV_IND ou ADV_DIRECT_IND, de ser no mínimo igual a 20 ms. De um modo geral, independentemente dos tipos de eventos, este valor tem de estar compreendido entre 20 ms e 10,24 segundos.

A variável $T_{advDelay}$ é um valor aleatório gerado pela camada de ligação, compreendido entre 0 e 10 ms. A Figura 4.1, ilustra o mecanismo descrito.

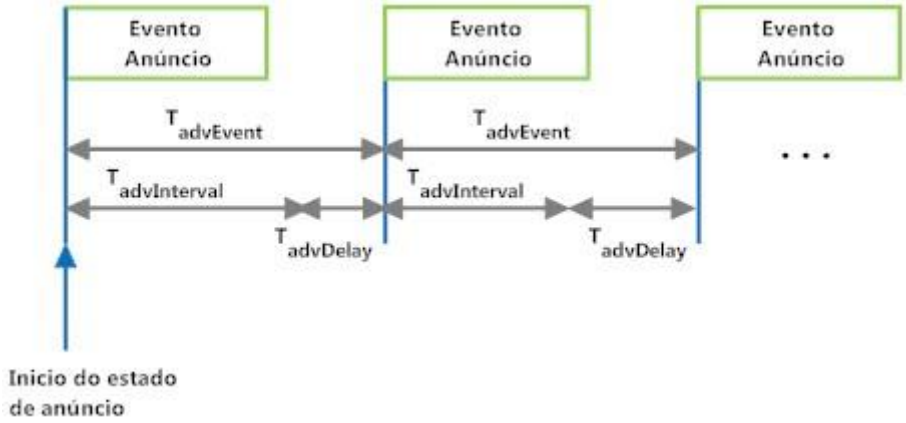

Figura 4.1 - Sequência de eventos de anúncio.

Se o anunciante receber um SCAN_REQ PDU por parte do dispositivo *scanner* contendo o seu endereço, deve-lhe responder enviando um SCAN_RSP PDU no mesmo canal de anúncio. Depois de este PDU ser enviado, ou se de acordo com as

políticas de filtragem o dispositivo anunciante está proibido de processar um PDU do tipo SCAN_REQ, este deve alternar para outro canal de anúncio para enviar um novo ADV_IND ou então finalizar o evento de anúncio.

O tempo entre o início de dois PDUs ADV_IND no mesmo evento de anúncio deve ser menor ou igual a 10 ms. Na Figura 4.2 observa-se e receção de um PDU do tipo SCAN_REQ e o respetivo envio de um PDU SCAN_RSP a meio do evento de anúncio, espaçados de T_IFS[4].

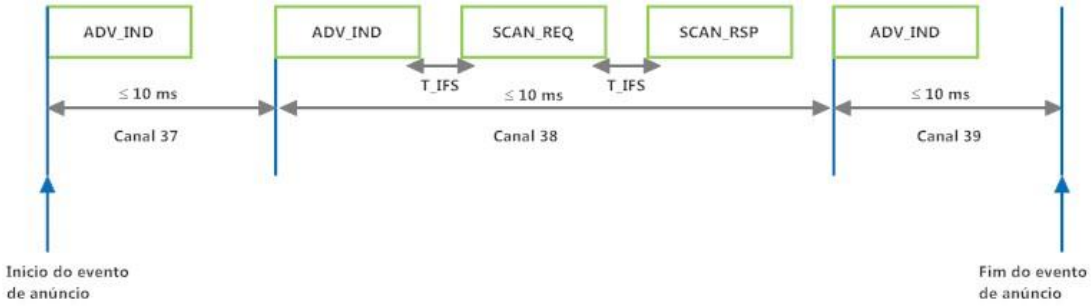

Figura 4.2 – PDUs de descoberta no evento de anúncio.

4.2 Eventos de Conexão

Caso o anunciante receba um PDU do tipo CONNECT_REQ por parte de um dispositivo iniciador de ligação, a camada de ligação deve transitar do estado de anúncio para o estado de conexão no papel de escravo. Por outro lado, se o dispositivo anunciante não estiver autorizado a processar PDUs do tipo CONNECT_REQ, deve-se mover para outro canal de anúncio e enviar um novo ADV_IND ou então terminar o evento de anúncio. A Figura 4.3 ilustra a receção de um PDU do tipo CONNECT_REQ, durante um evento de anúncio.

[4] T_IFS é o intervalo de tempo entre dois pacotes consecutivos no mesmo canal. Tem a duração mínima de 150 µs.

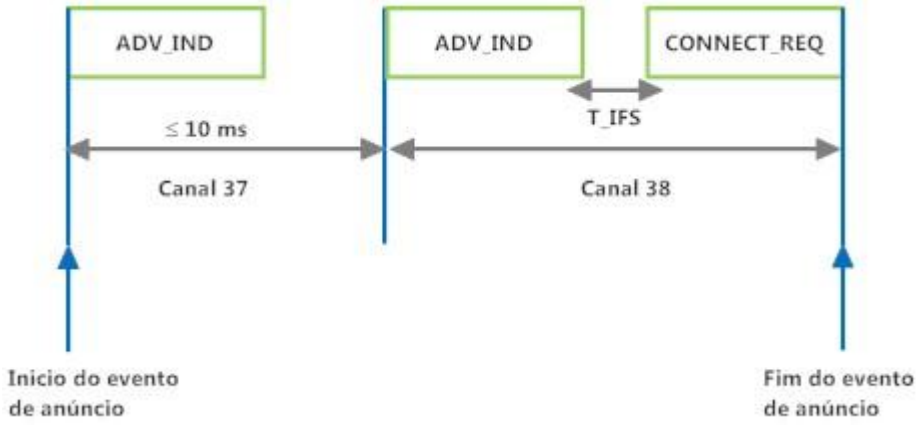

Figura 4.3 – Evento de anúncio com a receção de um PDU CONNECT_REQ.

Um evento de anúncio termina e um evento de conexão começa caso o dispositivo anunciante receba e aceite o pedido de conexão, ou seja, aceite e processe o PDU CONNECT_REQ vindo do dispositivo iniciador. Após entrar no estado de conexão, considera-se que a ligação está criada mas não estabelecida ainda. A ligação é apenas considerada estabelecida quando um pacote for recebido no canal de dados por um dos dispositivos.

Quando um dispositivo está numa conexão atua como mestre ou escravo, como foi já referido. O mestre controla o tempo de um evento de conexão, que é um ponto de sincronização entre o mestre e o escravo. A comunicação no evento de conexão é iniciada pelo mestre, que envia um pacote para o escravo. O mestre e o escravo alternam, enviando e recebendo pacotes no mesmo canal de dados, até que um dos dispositivos não tenha mais dados para transmitir (ver em mais detalhe na secção 5.1) ou até que o evento de conexão termine. O intervalo entre eventos de conexão ativos (*IEC*) é definido por dois parâmetros: o intervalo de conexão (*connInterval*) e a latência do escravo (*connSlaveLatency*), e pode ser calculado através da equação (10).

$$IEC = connInterval * (1 + connSlaveLatency) \qquad (10)$$

IEC também pode ser chamado de intervalo de conexão efetivo. De notar que caso a latência do escravo seja igual a zero, o intervalo entre eventos de conexão ativos é igual ao próprio intervalo de conexão.

O início de um evento de conexão tem o nome de ponto de âncora, e neste ponto o mestre tem de iniciar o envio de um pacote para o escravo através do canal de dados e o escravo deve estar a escutar. O início de cada evento de conexão é definido de acordo com o intervalo de conexão. O mestre tem ainda de garantir que o evento de conexão termina pelo menos T_IFS antes do novo ponto de âncora, ou seja, do início de um novo evento de conexão. Este intervalo de conexão, como foi já referido, pode ir de 7,5 ms até 4 segundos, e é definido pela camada de ligação no estado de iniciação, aquando do envio do PDU CONNECT_REQ. A Figura 4.4 ilustra a troca de pacotes entre mestre e escravo durante um evento de conexão.

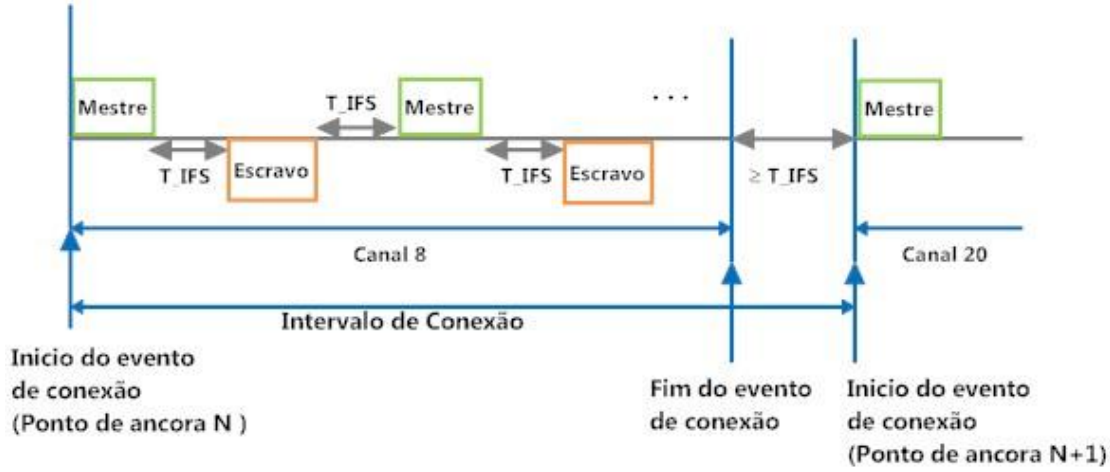

Figura 4.4 - Troca de dados entre mestre e escravo no canal de dados.

Como foi indicado na secção 2.4.9.2, o bit MD presente no cabeçalho do PDU de dados, indica se o dispositivo tem mais dados para enviar. Se nenhum dos dispositivos tiver ativado o bit MD nos seus pacotes, o pacote vindo do escravo finaliza o intervalo de conexão. Caso um ou ambos os dispositivos tiverem o bit MD ativo, o mestre deve continuar com o evento de conexão através do envio de outro pacote e o escravo deve escutar depois de enviar o seu pacote. Se um pacote não for recebido pelo mestre, este deve terminar o evento de conexão. O mesmo princípio se aplica ao escravo. De notar que, de mestre para escravo ou vice-versa, a receção de dois pacotes consecutivos com CRC inválido no mesmo evento de conexão, faz com que o evento seja logo terminado.

4.3 Latência

O método de transmissão de dados implementado na comunicação entre mestre e escravo foi através do envio de notificações. Como já foi referido, as notificações são enviadas do escravo para o mestre sem que este as requisite previamente através do envio de algum comando, e por outro lado este também não tem de confirmar a sua receção. A taxa de transmissão do BLE (R_{BLE}) é fixada em 1 Mbps, e o tempo total de transmissão ($T_{TotalTransm}$) de um pacote de mestre para escravo é dado pela equação (11), em que T_M e T_S são os tempos totais do envio de um pacote do mestre e escravo, respetivamente; e T_{IFS} é, como já referido anteriormente, o tempo de guarda entre qualquer pacote.

$$T_{TotalTransm} = T_M + T_{IFS} + T_S + T_{IFS} \quad (11)$$

O dispositivo mestre não envia qualquer informação para o dispositivo escravo, sendo que o seu payload é vazio. De acordo com o formato dos pacotes dos canais de dados apresentados na secção 2.4.9.2 é possível fazer uma análise detalhada do tempo de transmissão de cada pacote a circular no canal. O tempo de transmissão (T_T) de qualquer segmento de dados é dado pela equação (12), onde *L* significa o comprimento dos dados.

$$T_T = \frac{L}{R_{BLE}} \quad (12)$$

A equação (13) e (14) representam minuciosamente os tempos de transmissão do pacote do mestre (T_M) e do escravo (T_S), respetivamente.

$$T_M = T_P + T_{AA} + T_H + T_{CRC} \quad (13)$$

$$T_S = T_P + T_{AA} + T_H + T_{CRC} + T_{Not} + T_{AttH} + T_{Op} + T_{L2capH} \quad (14)$$

A Tabela 4.2 descreve em detalhe os tempos de transmissão de cada campo constituinte de cada pacote. Os campos TP, TAA, TH e TCRC são comuns para os dois tipos de pacotes.

Tabela 4.2 - Tempos de transmissão dos campos dos pacotes do mestre e escravo.

Sigla	Campo	Nº de bytes	Tempo Transmissão
T_P	Preambulo	1	8 µs
T_{AA}	Endereço Acesso	4	32 µs
T_H	Cabeçalho	2	16 µs
T_{CRC}	CRC	3	24 µs
T_{Not}	Notificação	0-20	0-160 µs
T_{AttH}	Handle do atributo	2	16 µs
T_{Op}	Opcode	1	8 µs
T_{L2capH}	Cabeçalho da camada L2CAP	4	32 µs

De acordo com as equações (13) e (14), e com os dados patentes na Tabela 4.2, T_M é igual a 80 µs e T_S igual a 296 µs. Como T_{IFS} é sempre, no mínimo, igual a 150 µs, aplicando a equação (11), o tempo total da comunicação ($T_{TotalTransm}$) entre mestre e escravo no envio de uma notificação é igual a 676 µs. A Figura 4.5 ilustra o diagrama de envio de uma notificação de escravo para mestre de acordo, com os tempos associados assinalados.

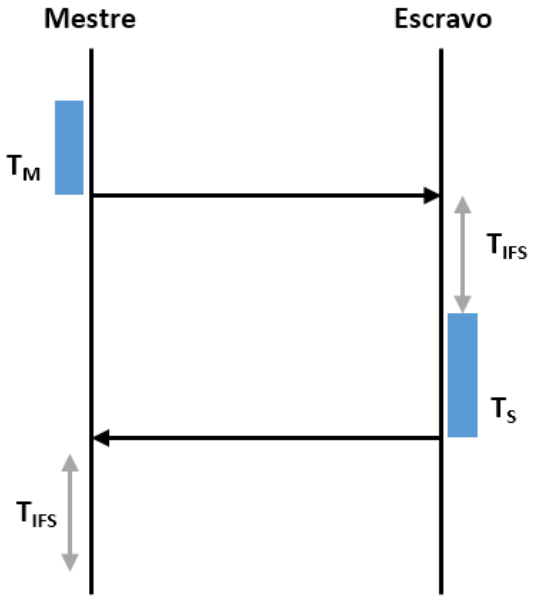

Figura 4.5 - Envio de uma notificação de escravo para mestre.

4.4 Débito

O cálculo do débito útil máximo ($R_{BleÚtilMax}$) entre dois módulos BLE baseado no envio de dados de escravo para mestre pode ser dado pela equação (15), assumindo que os dados podem ser transmitidos continuamente sem parar.

$$R_{BleÚtilMax} = \frac{PayloadApp}{T_{TotalTransm}} \quad (15)$$

O payload considerado ao nível da camada de aplicação (*PayloadApp*) é de 20 bytes pois apenas é contabilizado o tamanho máximo permitido para uma notificação. Ao nível da camada de ligação é 27 bytes (PayloadLL) pois além dos dados da notificação, têm incluído o cabeçalho da camada L2CAP, a *handle* do atributo e o *opcode*. Considerando estes campos obtém-se o débito bruto máximo ($R_{BleBrutoMax}$) para uma ligação BLE, equação (16), assumindo da mesma forma o envio contínuo de dados sem parar.

$$R_{BleBrutoMax} = \frac{PayloadLL}{T_{TotalTransm}} \quad (16)$$

Substituindo, nas equações (15) e (16), os valores já conhecidos do PayloadLL, PayloadApp e $T_{TotalTransm}$, obtém-se um $R_{BleÚtilMax}$ igual a 236,7 Kbps e um $R_{BleBrutoMax}$ igual a 319,5 Kbps.

Um dos parâmetros que influencia o débito numa comunicação BLE é o intervalo de conexão pois impedem que as transmissões sejam continuas e que se transmita dados sem parar. A transmissão de dados é feita nos eventos de conexão e num evento de conexão podem ser transmitidos mais que um pacote (notificação). Este fator, aliado a um intervalo de conexão reduzido, proporciona um aumento do débito. Este aumento do débito não é em relação aos débitos obtidos nas equações 15 e 16, mas sim relativamente ao uso de várias notificações num mesmo evento de conexão.

Devido à latência do escravo, que a par do intervalo de conexão, é também um parâmetro crítico para o cálculo do débito máximo, no cálculo do débito seguinte

(equação 17) considerou-se o intervalo entre eventos de conexão ativos (*IEC* – equação 10), e não apenas o intervalo de conexão (*connInterval*).

A equação (17) traduz o débito numa transmissão BLE ($R_{BleÚtil}$), ao nível da camada aplicacional, em função do *IEC* e do número de pacotes que é possível enviar no mesmo evento de conexão ($N_{Pacotes}$).

$$R_{BleÚtil} = \frac{N_{Pacotes} * PayloadApp}{IEC} \quad (17)$$

A Figura 4.6 ilustra graficamente diferentes curvas que significam diferentes configurações dos parâmetros usados na equação (17), exceto o tamanho do *PayloadApp*, que se mantém o máximo possível, ou seja, igual a 20 bytes.

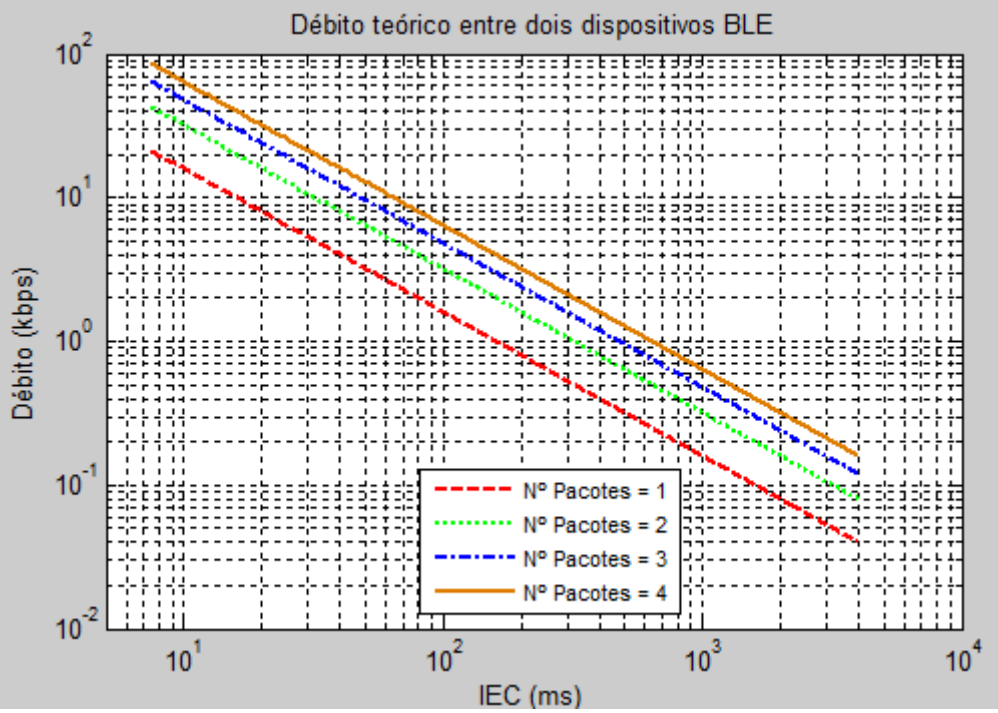

Figura 4.6 - Débito útil entre dois dispositivos BLE, em função de IEC e $N_{Pacotes}$.

Teoricamente, segundo esta análise, o maior débito útil ($R_{BleÚtil}$) possível entre dois dispositivos BLE é de 85,3 kbps (aproximadamente 11 kBps) assumindo um IEC igual a 7,5 ms (intervalo de conexão de 7,5 ms e latência do escravo igual a zero) e um envio de 4 notificações por evento de conexão.

4.5 Consumo Energético CC2540

O consumo energético é claramente um dos principais aspetos a ter em conta aquando da conceção de um sistema que albergue uma WSN, onde a autonomia das baterias dos sensores é crítica. No caso do BLE, existem certos parâmetros que devidamente ajustados contribuem para um gasto mínimo de energia, dependendo, claro, do objetivo final de cada aplicação.

A vantagem de ter um intervalo de conexão longo é a energia significativa que se poupa, pois o dispositivo pode dormir a maior parte do tempo entre os eventos de conexão. A desvantagem é que se o dispositivo tiver dados que necessita de enviar, terá de esperar até ao próximo evento de conexão, aumentando o atraso.

Por outro lado, a vantagem de ter um intervalo de conexão curto, é que existem mais oportunidade em enviar dados, dado que os dispositivos estão conectados com mais frequência. A desvantagem reside no facto de que mais energia vai ser gasta, pois o dispositivo vai estar frequentemente a acordar para os eventos de conexão.

A latência do escravo é também um parâmetro que influencia o consumo energético no dispositivo periférico. Quanto maior for o valor definido para este parâmetro, menor será o consumo energético, pois o dispositivo periférico não necessita de acordar durante esse número de eventos de conexão.

Um evento de conexão é composto por vários estados que se dividem em estados de pré-processamento, receção, transmissão e pós processamento. A Figura 4.7, retirada de [33], ilustra um evento de conexão durante a receção e transmissão de um único pacote, captado num dispositivo escravo. O eixo *y* representa a tensão necessária para cada fase do evento de conexão e o eixo *x* o tempo decorrido. No estudo, os autores medem a tensão sob uma resistência de 10 Ohms, sendo por isso fácil obter a corrente gasta em cada fase do evento de conexão, através da Lei de Ohm.

Figura 4.7 - Diferentes estados de um evento de conexão (adaptado de [33]).

O estado 1 é denominado por *wake-up* e ocorre quando o dispositivo acorda e o nível de corrente sobe suavemente. No estado 2, *pre-processing*, a pilha protocolar do BLE prepara a camada rádio para o envio e receção de dados. No estado 3, *pre-*R_x, dá-se a ativação do módulo rádio de modo a preparar a transmissão de dados. Reunidas as condições para iniciar a receção de dados, ocorre o estado 4, denominado por R_x. Neste estado o escravo escuta o canal, esperando por um pacote vindo do mestre. No estado 5, R_x*-to-*T_x *transaction*, o escravo para de escutar e prepara-se para transmitir um pacote para o mestre. Essa transmissão ocorre no estado 6, Tx. Após a comunicação entre mestre e escravo, a pilha do BLE processa o pacote recebido e prepara o temporizador para entrar no modo *sleep*. A este estado, 7, dá-se o nome de *post-processing*. Por fim, no estado 8, a corrente cai para valores mais baixos e o dispositivo fica num modo pre-sleep, esperando assim pelo próximo evento de conexão.

Através do tempo em que cada estado permanece ativo, da corrente gasta em cada um e do intervalo entre eventos de conexão ativos, é possível efetuar uma estimativa do consumo energético para um dispositivo escravo numa ligação BLE.

Através da equação (18), obtém-se uma média da corrente gasta num evento de conexão (I_{ConnEv}), durante o tempo em que o dispositivo permanece acordado (T_{awake})

através do tempo (T_{Si}) e corrente (C_{Si}) gasta em cada estado, em que *i* é o valor do estado atual.

$$\mathrm{I}_{ConnEv} = \frac{\sum_{i=1}^{totalStates}(T_{Si} * C_{Si})}{T_{awake}} \quad (18)$$

Para calcular a média da corrente gasta em todo o intervalo entre eventos de conexão ativos (I_{IEC}), equação (19), é necessário atender ao tempo em que o dispositivo está no modo *sleep* e respetiva corrente gasta (I_{Sleep}), após transmissão dos dados.

$$I_{IEC} = \frac{[(IEC - T_{awake}) * (I_{Sleep}) + (T_{awake}) * (I_{ConnEv})]}{IEC} \quad (19)$$

De modo a calcular uma estimativa da longevidade da bateria (*BatLife*), em horas, durante o estado de conexão, usa-se o valor obtido em (19). A equação (20) fornece este cálculo, onde *BatCapacity* é a capacidade da bateria em mAh.

$$BatLife = \frac{BatCapacity}{I_{IEC}} \quad (20)$$

4.6 Número de escravos suportados

A rede de comunicação BLE segue uma topologia em estrela, onde um mestre pode ter vários escravos associados, mas um escravo apenas pode comunicar com o seu mestre, pelo que a comunicação direta entre escravos não é permitida. Um evento de conexão, num dado canal físico, diz respeito apenas a uma ligação entre mestre e escravo, sendo que o mestre gere múltiplos eventos de conexão. A Figura 4.8 ilustra a topologia de rede possível na tecnologia BLE.

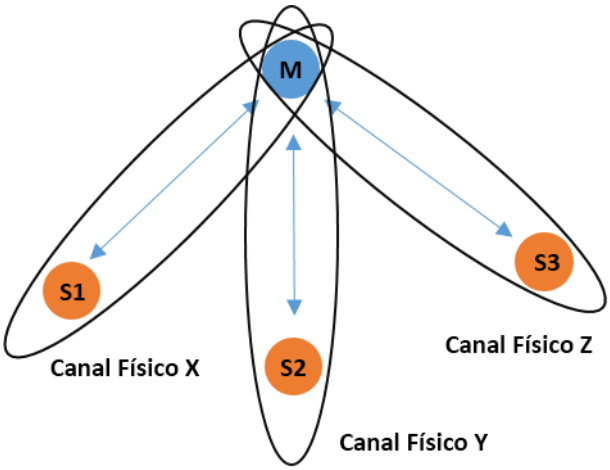

Figura 4.8 - Topologia em estrela de uma rede BLE.

De modo a obter uma estimativa do número de escravos suportados por uma rede BLE no cenário em que todos estão a transmitir informação para o mestre, é necessário efetuar uma análise da capacidade do canal de transmissão. Na secção 4.3, de acordo com o tamanho dos pacotes e respetivos tempos de transmissão, observou-se que o tempo necessário para a transmissão de um pacote com tamanho máximo era, no mínimo, 676 µs. Por outro lado, o número máximo de pacotes ($N_{MaxPacotes}$) que poderia ser transmitido num período igual ao intervalo entre eventos de conexão ativos (IEC) é expresso pela equação (21).

$$N_{MaxPacotes} = \frac{IEC}{T_{TotalTransm}} \quad (21)$$

Dado este número, é possível saber o número máximo de escravos permitidos a transmitir na mesma rede, desde que se saiba o número de pacotes[5] que cada escravo está habilitado a enviar por cada evento de conexão. A equação (22) mostra o número de escravos possíveis ($N_{MaxEscravos}$) durante uma ligação BLE simultânea, em função de $N_{MaxPacotes}$ e número de notificações por cada escravo (N_{Noti}).

$$N_{MaxEscravos} = \frac{N_{MaxPacotes}}{N_{Noti}} \quad (22)$$

[5] Os termos pacote e notificação têm o mesmo significado, já que é sob a forma do envio de notificações que o protocolo está a ser analisado.

A especificação do BLE não define o número de pacotes que podem ser trocados entre mestre e escravo durante um evento de conexão, e consequentemente o número de escravos suportados. O número de escravos e número de pacotes trocados entre mestre e escravo, pode estar limitado pelo *design* do protocolo ou pela implementação da stack nos diversos dispositivos.

4.7 Protocolo LPRT

O protocolo LPRT [47] é um protocolo orientado às redes de sensores sem fios, sendo concebido para permitir um baixo consumo energético para os dispositivos terminais, baixa latência na entrega dos dados, fiabilidade na transmissão dos dados e uma utilização eficiente da largura de banda. É um protocolo que utiliza CSMA/CA e TDMA como técnicas de acesso e não define uma camada física própria, usando a camada física do IEEE 802.15.4.

O acesso por contenção permite o envio de dados assíncronos por parte das estações, bem como a troca de mensagens de controlo. O principal problema é a probabilidade de colisões, mesmo sem o problema da estação oculta[6], e o consequente *overhead* necessário para evitá-las, que reduz o débito que se pode obter.

No acesso por TDMA, o tempo é dividido em pequenos intervalos, *mini-slots*, que são atribuídos às estações que pretendem transmitir dados. Para permitir o acesso por TDMA e respetiva sincronização temporal, primeiramente é enviada uma mensagem periódica (*beacon*), que contém a informação de sincronização. O tempo entre dois *beacons* é denominado por *superframe*. Cada *superframe* está dividida num número fixo de *mini-slots,* que são atribuídos às estações quando estas necessitam de alocação.

6 Ocorre quando uma estação (A) deteta o meio livre e tenta enviar uma mensagem para uma estação (B) que está a receber uma transmissão de uma terceira estação (C), cujo alcance não atinge a estação original (A). Diz-se que A está oculta da transmissão entre B e C.

O mecanismo GTS (Guaranteed Time Slot) do IEEE 802.15.4 permite apenas 16 *slots* para um máximo de 7 alocações de transmissões periódicas. Em contraste, o protocolo LPRT apresenta uma maior granularidade, com um maior número de *slots* disponíveis, permitindo aumentar a eficiência na utilização da largura de banda, bem como a possibilidade de alocação de um número muito maior de transmissões periódicas. A Figura 4.9 representa a estrutura de uma *superframe* usada no protocolo LPRT.

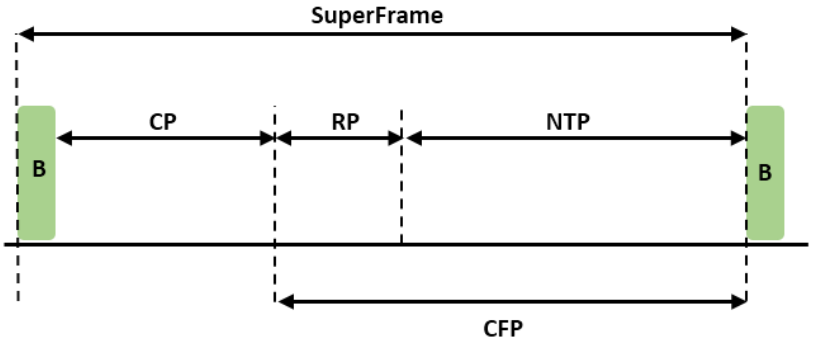

Figura 4.9 - Estrutura da superframe do protocolo LPRT (adaptado de [11]).

O início de uma *superframe* é marcado pela transmissão do beacon (B), iniciando-se de seguida o período de contenção (CP), durante o qual as estações se associam com o coordenador (estação base) e requisitam *mini-slots* para transmissão durante o período livre de contenção (CFP). As transmissões durante o CP são efetuadas sob as regras do CSMA/CA. No CFP, as transmissões de mensagens pelas estações são reguladas pela estação base. Este período é subdividido em dois períodos distintos: período de retransmissões (RP) e período de transmissão normal (NTP). O RP, de tamanho variável, surge como uma nova oportunidade para a estação base receber as mensagens que não foram recebidas com sucesso durante a superframe anterior. Caso não haja retransmissões pendentes, este período pode simplesmente não existir. De modo a manter a ordem cronológica das mensagens, o RP é colocado antes do NTP. Por fim, o NTP é o período onde são colocadas as alocações requisitadas pelas estações terminais durante o período de associação. De notar que todas as alocações do RP e do NTP são efetuadas de modo a impedir a

fragmentação da *superframe* e para que a duração do CAP possa ser maximizada, levando assim a um aumento da eficiência na utilização da largura de banda.

O acesso por CSMA/CA é utilizado para a transmissão de mensagens durante o CP. Um dispositivo que queira transmitir mensagens durante este período tem de esperar um tempo de *backoff* gerado aleatoriamente. Caso o meio esteja ocupado após este tempo, tem de esperar novamente outro tempo de *backoff* aleatório. Este processo é repetido até que o meio esteja desimpedido.

Existem outras melhorias do protocolo LPRT sendo elas o iLPRT (Improved LPRT) [48] e eLPRT (Enhanced LPRT) [27]. No que diz respeito ao iLPRT, os principais melhoramentos propostos baseiam-se na utilização de *beacons* mais pequenos, alocação de múltiplas retransmissões e também retransmissões de dados no período CP com a finalidade de aumentar a eficiência do protocolo. Ao contrário deste protocolo, que sofreu melhoramentos, mas sempre em vista a sua aplicação em redes homogéneas, o eLPRT surgiu como um protocolo que se poderia aplicar sem o condicionamento a um tipo de aplicação específica. Os melhoramentos introduzidos basearam-se num novo mecanismo de reserva proposto como alternativa à camada MAC do IEEE 802.15.4. O mecanismo original, GTS, que permite dar suporte a aplicações que requerem uma largura de banda dedicada e atraso limitado, revela alguns pontos negativos como a utilização ineficiente da largura de banda e suporte para um máximo de 7 dispositivos. O eLPRT introduz melhoramentos em certas características aumentando a taxa de transmissão de dados, melhora a utilização da largura de banda e aumenta o número de dispositivos suportados.

4.7.1 Número de escravos suportados

De modo a estabelecer um paralelismo com o BLE, no que diz respeito ao número de escravos suportados numa rede baseada em LPRT, apresentam-se de seguida uma análise, baseada em equações, que permitem obter o número de nós suportados ($N_{NosLPRT}$) em função da duração da *superframe* (T_{SF}) e do tamanho do PDU (P_{PDU}) a enviar.

A taxa de transmissão de dados do LPRT (R_{LPRT}) é de 250 kbps. O tempo de transmissão de um PDU é expresso pela equação (23).

$$T_{Transmissão} = \frac{P_{PDU}}{R_{LPRT}} \quad (23)$$

A uma determinada duração de *superframe* corresponde um número de mini-slots que podem ser usados em todos os períodos na mesma ($N_{MiniSlotsTotal}$). O número de mini-slots necessários para a transmissão de uma mensagem com tamanho P_{PDU} numa superframe é dado pela equação (24).

$$N_{MiniSlotsMenssagem} = \frac{T_{Transmissao} * N_{MiniSlotsTotal}}{T_{SF}} \quad (24)$$

Para determinar o número de nós que podem transmitir os seus dados durante o período destinado ao efeito (CFP) é necessário saber o número de mini-slots reservados para tal ($N_{MiniSlotsCFP}$). Deste modo o número máximo de transmissões que podem ocorrer durante o período CFP é dado pela equação (25).

$$N_{NosLPRT} = \frac{N_{MiniSlotsCFP}}{N_{MiniSlotsMenssagem}} \quad (25)$$

5. Resultados e Discussão

Neste capítulo são apresentados os testes efetuados e respetivos resultados obtidos, tendo em conta as decisões tomadas em cada momento e as limitações encontradas ao nível da implementação e das plataformas de hardware usadas. São apresentados os resultados da análise ao protocolo BLE, onde os parâmetros número máximo de notificações, utilização do canal, taxa de perda de pacotes, consumo energético, colisões na transmissão e número de escravos suportados irão ser avaliados. Para estes testes recorreu-se à comunicação entre dois ou mais módulos CC2540, e sempre que necessário recorreu-se ao software BTool e ao Packet Sniffer, também pertencentes à Texas Instruments, para complementar os resultados.

Paralelamente, sempre que possível e necessário, são apresentados testes efetuados ao sistema de monitorização de postura, seguindo os mesmos parâmetros acima descritos. Os testes relativos à postura do atleta são discutidos nesta fase. De notar que, nestes testes, o smartphone toma o papel de estação central.

5.1 Notificações por Evento de Conexão

Um escravo pode ser programado para enviar um certo número de notificações por evento de conexão. Os testes efetuados demonstram que, usando o payload máximo permitido para uma notificação BLE (20 bytes), o módulo CC2540 suporta apenas o envio de três notificações numa ligação entre um mestre e um escravo. No entanto, variando o tamanho payload é possível enviar mais do que três notificações por evento de conexão. A Tabela 5.1 mostra a relação entre o número de notificações e o tamanho das mesmas.

Tabela 5.1 - Correspondência entre tamanho do payload e número de notificações.

Tamanho do payload (bytes)	Nº de Notificações
17 a 20	3
13 a 16	4
10	5

Para o referido teste foi usado um intervalo de conexão igual a 10 ms, valor muito próximo do mínimo permitido pela especificação da stack do BLE da Texas Instruments, ou seja, 7,5 ms. No entanto, o hardware apresenta limitações quando aplicado este valor (7,5 ms), verificando-se perdas sucessivas de pacotes e eventos de conexão sem qualquer troca de informação entre mestre e escravo. De notar que apesar de se ter fixado o intervalo de conexão em 10 ms, este foi útil e apropriado para ilustrar um instante do envio de informação entre mestre e escravo, o que não reflete que possa ser usado para uma real comunicação durante longos períodos de tempo, devido às perdas inerentes ao protocolo aquando o uso de intervalos de conexão baixos. A Figura 5.1 ilustra uma captura de tráfego da comunicação entre mestre e escravo durante um evento de conexão. Primeiramente é feito o *polling* pelo dispositivo mestre, que envia um PDU vazio, seguindo-se a notificação por parte do dispositivo escravo, até um total de três notificações. Para o efeito, o tamanho do payload usado foi de 20 bytes, de modo a testar o máximo tamanho possível.

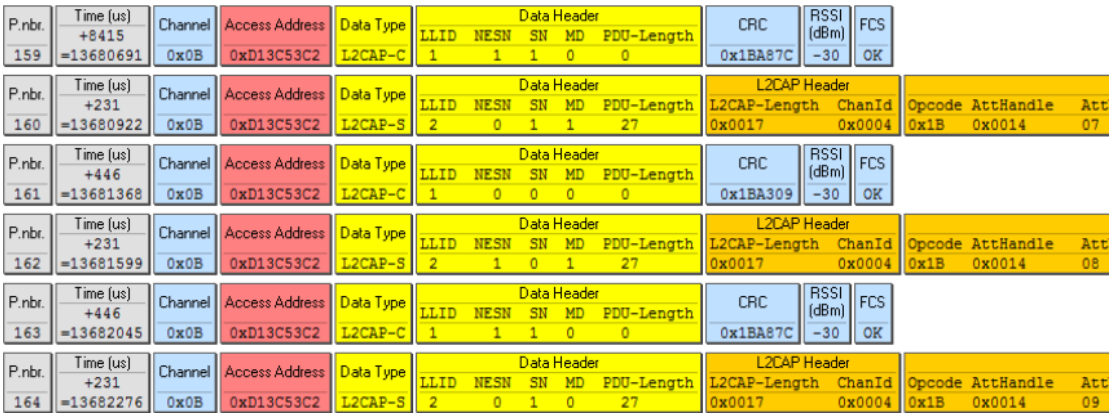

Figura 5.1 - Evento de Conexão no canal 11 (0x0B), com três notificações.

Independentemente do intervalo de conexão definido, o número máximo de notificações que são permitidas enviar em cada evento de conexão são três, caso se esteja a usar o tamanho máximo do payload.

No caso do sistema de monitorização de postura implementado, dado que é um sistema para captar movimentos corporais, optou-se por uma frequência de amostragem (f_s) dos sensores igual a 30 Hz. Para enviar o máximo de notificações, que contenham as amostras dos sensores, por evento de conexão (N_{Noti}), ou seja, três, o intervalo entre eventos de conexão ativos (com latência do escravo igual a zero) tem de ser igual a 100ms, como se pode observar pela equação (26).

$$N_{Noti} = f_s * IEC \tag{26}$$

Analisando com mais detalhe os cabeçalhos dos PDUs do mestre e do escravo durante o evento de conexão, na Figura 5.2, é possível observar o comportamento dos campos LLID, NESM, SN, MD e PDU-Lenght. O significado destes campos foi explicado na secção 2.4.9.2.

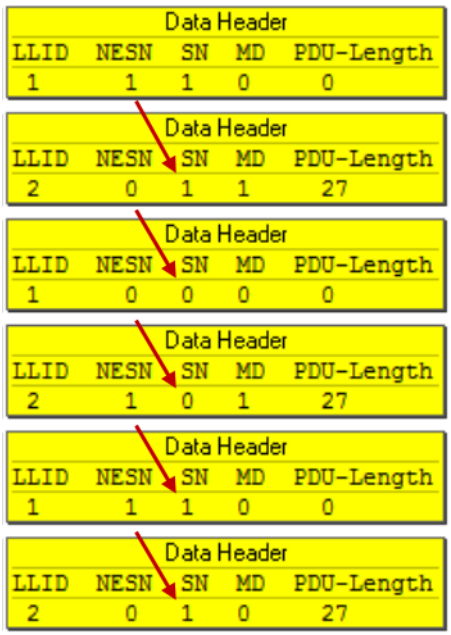

Figura 5.2 – Cabeçalho (Header) do PDU do mestre e escravo.

O primeiro campo, LLID, tem valor 1 nos pacotes pertencentes ao mestre, pois estes são PDUs de dados vazios, ou seja PDU-Lenght igual a zero. Já os PDUs do

escravo contêm dados (PDU-Lenght igual a 27), por isso o campo LLID toma o valor 2. Dado que o PDU enviado na direção mestre-escravo é um PDU vazio, o campo MD é sempre zero. Já nos pacotes do escravo, excetuando o último, este campo é sempre igual a 1, pois indica que ainda existem dados para enviar. No último pacote por parte do escravo, este campo é zero pois não são permitidas mais do que 3 notificações enviadas no mesmo evento de conexão. Os campos NESM e SN alternam de valor consoante a mudança dos pacotes, verificando-se o comportamento esperado, tal como indicam as setas a vermelho, representadas na Figura 5.2.

Através da Figura 5.3 e Figura 5.4 é possível observar, experimentalmente, o número de notificações durante um evento de conexão, nas situações em que o dispositivo mestre é um módulo CC2540 e o smartphone, respetivamente. Em ambos os casos, são rececionadas três notificações em 100 ms, tal como é esperado.

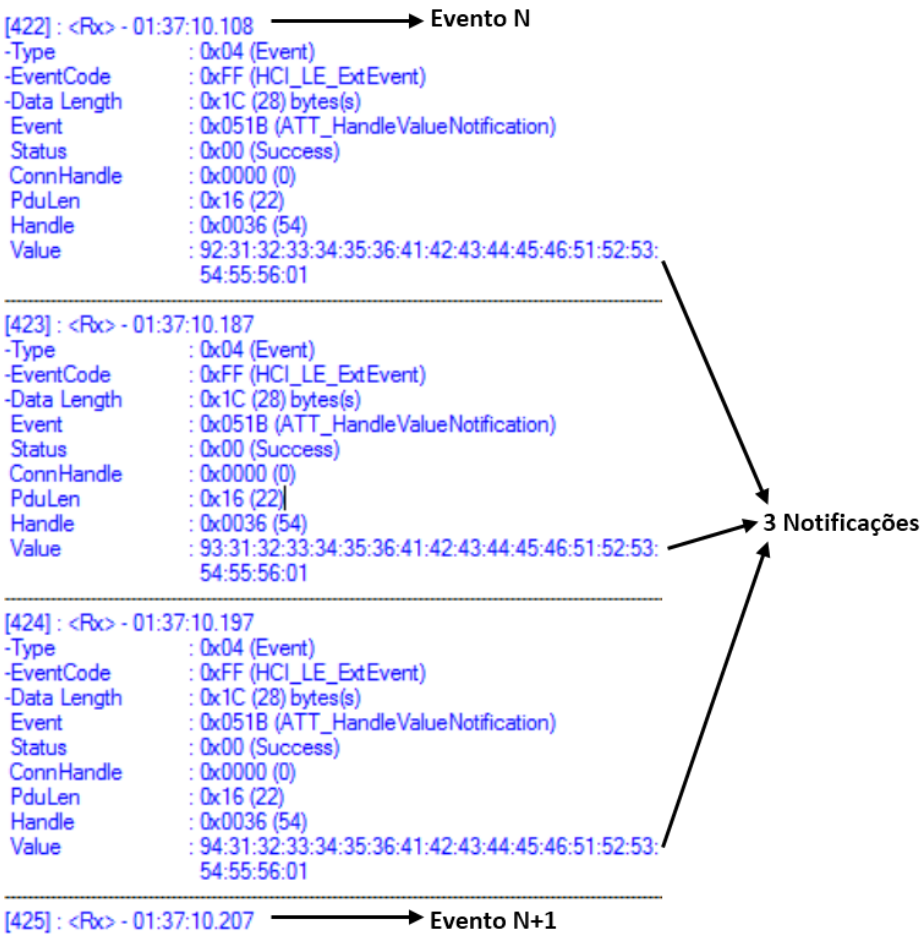

Figura 5.3 – Receção de 3 notificações com IEC de 100 ms (Mestre: CC2540).

Figura 5.4 - Receção de 3 notificações com IEC de 100 ms (Mestre: Smartphone Android).

5.2 Utilização do Canal

Para testar a utilização do canal de comunicação, optou-se pela definição do IEC igual a 100 ms de acordo com o sistema de monitorização de postura. O tempo total de transmissão de um pacote do mestre e do escravo, está previamente definido nas equações (13) e (14), respetivamente. De acordo com o número máximo de notificações permitidas por evento de conexão, o tempo total da transmissão de dados durante um evento de conexão é dado pela equação (27). As notificações têm um payload igual a 20 bytes.

$$T_{TotalTransmEv} = 3T_M + 3T_S + 6T_{IFS} \quad (27)$$

Sendo T_M igual a 80 µs, T_S igual a 296 µs e T_{IFS} igual a 150 µs, obteve-se, um tempo total de transmissão igual a 2028 µs. A captura de tráfego visível na Figura 5.5 ilustra a utilização do canal durante um evento de conexão. De modo a comprovar o resultado da equação (27), é também possível observar o fim do evento que precede o evento em questão, bem como o início do evento seguinte.

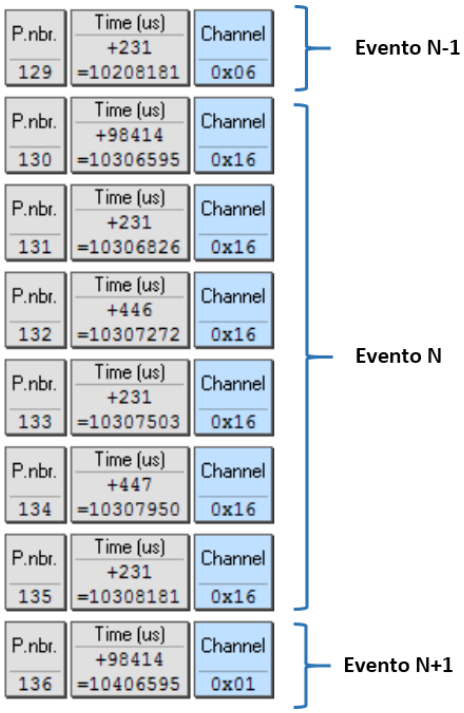

Figura 5.5 – Perspetiva da utilização do canal durante um evento de conexão no canal 0x16.

O evento de conexão que se inicia no canal 0x16, tem como ponto de início o instante 10306595 µs, e o evento de conexão seguinte, canal 0x01, inicia-se aos 10406595 µs. Comprovou-se assim os 100 ms de intervalo entre eventos de conexão ativos, que é o tempo que decorre entre o início de dois eventos de conexão ativos. O pacote cujo preâmbulo é o 135 é o último pacote enviado pelo escravo no evento de conexão N, e começa a ser enviado no instante 10308181 µs. Ao adicionar T_S e T_{IFS}, ao instante em que começa a ser enviado obteve-se um total de 10308627 µs. A este valor, foi subtraído o instante em que o evento se inicia e obteve-se a utilização do canal, com o valor 2032 µs. O valor resultante é muito próximo do valor obtido teoricamente pela equação (27).

5.3 Taxa de Perda de Pacotes

Dado que o protocolo BLE limita a transmissão contínua de dados devido à imposição de intervalos de conexão, torna-se mais interessante e pertinente efetuar avaliações da taxa da perda dos pacotes (PER – Packet Error Rate), do que

propriamente testes acerca do débito máximo possível. A avaliação do PER é dada pelo rácio entre número de pacotes perdidos ou inválidos (N_{Perdas}) e o número total de pacotes enviados (N_{Total}), tal como se observa na equação (28).

$$PER = \frac{N_{Perdas}}{N_{Total}} * 100\% \quad (28)$$

Para a realização destes testes experimentais à rede de comunicação BLE, utilizou-se o máximo payload possível para uma notificação, ou seja 20 bytes, e vários valores para o intervalo de conexão (connInterval), mantendo-se a latência do escravo sempre igual a zero. Os parâmetros de conexão, como intervalo de conexão e latência do escravo foram definidos no dispositivo mestre, de modo a surtirem efeito aquando do estabelecimento de conexão. Na realização dos testes considerou-se a comunicação BLE durante a transmissão de 1000 pacotes, onde o papel de estação central pode ser tomado por duas entidades distintas, sendo elas um módulo CC2540 e um smartphone Android. A Tabela 5.2 ilustra a configuração dos parâmetros para os diferentes testes.

Tabela 5.2 - Parâmetros usados nos testes de PER.

Testes	connInterval (ms)	slaveLatency	Payload (bytes)	N_{Noti}
Teste 1	7,5;10;15;20;25;33,75;50;60;100;500;1000	0	20	3
Teste 2	7,5;10;15;20;25;33,75;50;60;65	0	20	1,2,3

5.3.1 Teste 1

Durante este teste foram usados vários valores para o intervalo de conexão, sempre com a latência do escravo definida com o valor zero, de modo a maximizar o débito, já que deste modo não existem eventos de conexão inativos. A Figura 5.6 representa o gráfico que contém medições experimentais da taxa de perda de pacotes (PER) em função dos parâmetros definidos para o teste 1, ao fim da receção

de 1000 pacotes no dispositivo mestre. A linha vermelha corresponde aos resultados obtidos experimentalmente quando o dispositivo mestre é um módulo CC2540, enquanto a linha azul diz respeito aos resultados obtidos quando o mestre é o smartphone Nexus 5 do sistema. De notar que os valores de *connInterval* da tabela para os quais PER=0 não estão representados na Figura 5.6.

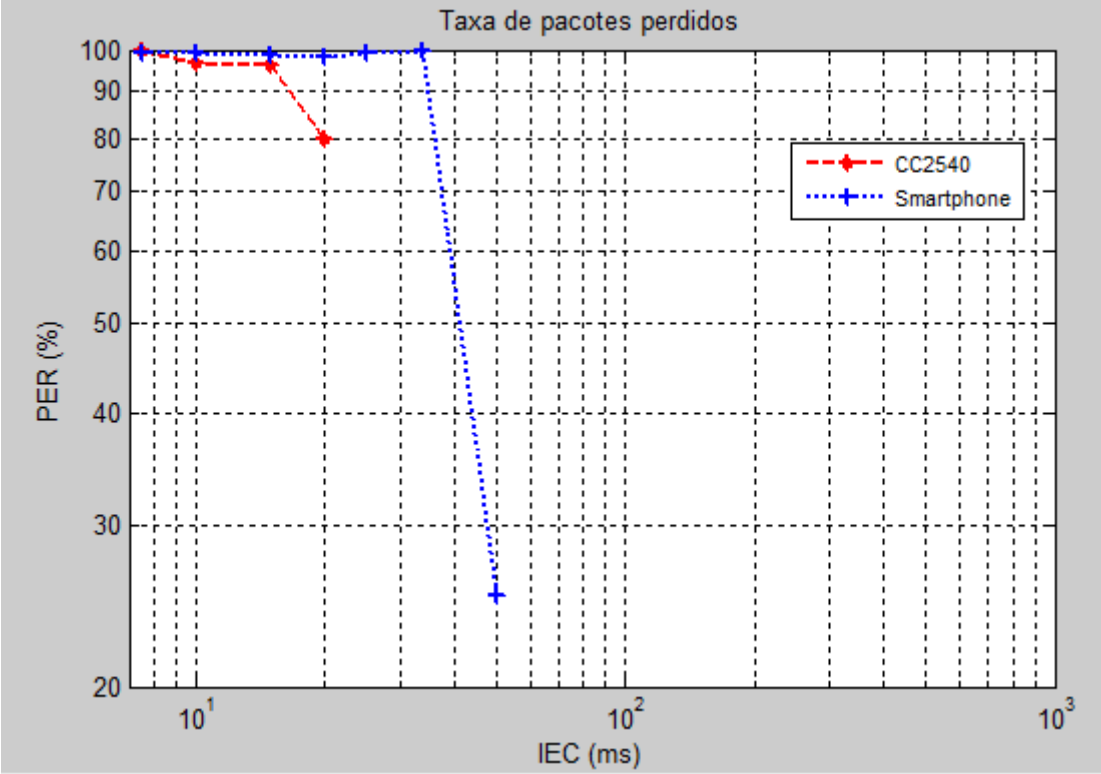

Figura 5.6 - Relação do PER quando o mestre é um módulo CC2540 e um smartphone.

Para valores muito baixos de IEC, tanto para o caso em que o dispositivo mestre é um módulo CC2540 ou para o caso de ser o smartphone Nexus 5, a taxa de perdas de pacotes é elevadíssima concentrando-se próxima dos 100 %. Estas perdas em rajada devem-se à limitação em termos de hardware e/ou software dos dispositivos usados, já que teoricamente a especificação do BLE assume que o intervalo de conexão pode ser no mínimo 7,5 ms. A partir dos 25 ms, já não se observam quaisquer perdas na comunicação quando o mestre é um módulo CC2540. Já para o caso do smartphone, a inexistência de qualquer perda só se verifica com intervalos a partir de 60 ms.

Teoricamente, com um IEC igual a 7,5 ms, aliado ao envio do máximo número de notificações permitido pelos módulos CC2530 (3) e com o máximo payload (20 bytes), obteríamos o débito útil máximo possível numa ligação BLE entre mestre e escravo próximo dos 64 kbps. Isto corresponde a um débito bruto igual a 118,4 kbps. Em comparação com o resultado obtido pelas equações 15 e 16, da secção 4.4, estes valores são mais baixos pois nas equações assume-se que a transmissão é contínua, mas na verdade o BLE impõe os intervalos de conexão impedindo esse comportamento (equação 17).

5.3.2 Teste 2

Neste segundo teste, efetuado na condição do dispositivo mestre ser o smartphone, escolheram-se instantes de IEC mais próximos no intervalo em que estes causam a perda de pacotes na comunicação BLE. Variou-se o número de notificações enviadas por evento de conexão de modo a averiguar qual o instante de IEC mínimo que não causa perda de pacotes. Através da Figura 5.7 é possível observar graficamente o traçado para as diferentes condições. Observou-se que à medida que o número de notificações enviadas por evento de conexão diminui, o instante de IEC necessário para garantir um PER igual a zero também diminui. Tal como já tinha sido averiguado, o último intervalo para o qual se verifica perda de pacotes é de 50 ms, considerando o envio de 3 notificações. Neste caso, o valor de PER é sensivelmente próximo dos 25%. Já no envio de duas notificações, para um intervalo de 33,75 ms o PER medido é de 51%, verificando-se a redução do PER para zero com intervalos superiores. Por fim, quando é enviada uma notificação por evento de conexão, o valor de IEC de 15 ms ainda corresponde a um PER próximo dos 100%, verificando-se inexistência de qualquer perda com intervalos superiores a este valor. De notar que os valores de *connInterval* da tabela para os quais PER=0 não estão representados na Figura 5.7.

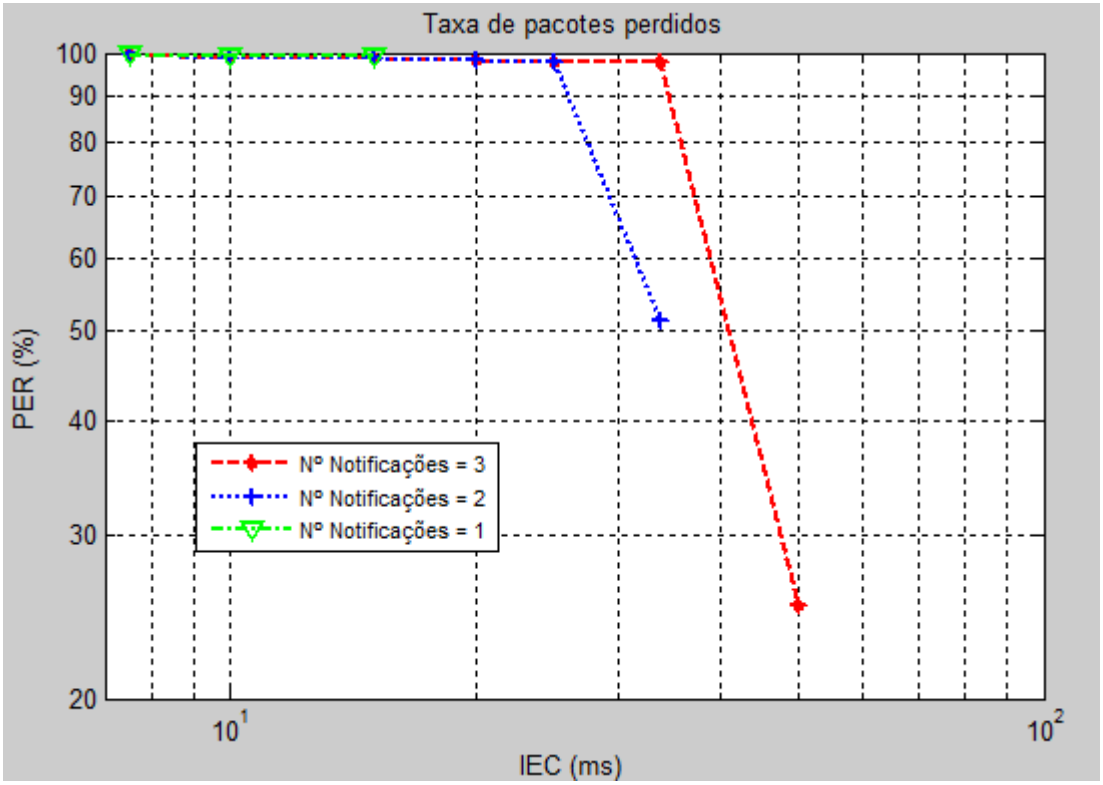

Figura 5.7 – Diferentes valores de PER em função da variação do número de notificações.

Para ambos os testes realizados, as perdas que se observam quando o IEC é baixo são causadas devido à limitação de cada dispositivo. Qualquer perda de informação que se verifique após os instantes onde o PER é zero será causada por erros no canal ou por fenómenos de *fading*[7] ou de ruído.

5.4 Consumo Energético

Na secção 4.5 foi apresentada uma série de equações que permitem estimar não só a corrente gasta num evento de conexão (I_{ConnEv}), mas também a corrente gasta em todo o estado de conexão (I_{IEC}). A realização da análise apresentada nesta secção permite uma aproximação teórica do gasto de energia durante um evento de conexão com a transmissão de vários pacotes entre mestre e escravo, variando o

[7] Atenuação (desvanecimento) que o sinal sofre num dado meio de propagação.

IEC. O tempo de duração da bateria (*BatLife*) também é estimado após cálculo dos valores anteriores.

Dado que não foram efetuadas medições experimentais da corrente gasta em cada estado durante a comunicação BLE entre dispositivos, nem contabilizado o tempo de permanência nos diferentes estados de um evento de conexão, a maioria destes valores vão ser substituídos pelos valores fornecidos no exemplo usado na secção 4.5, que foram obtidos experimentalmente em [33]. Os tempos de transmissão de um pacote do mestre e do escravo calculados previamente nesta dissertação (secção 4.3) são usados nesta análise, substituindo os valores do exemplo. A Tabela 5.3 ilustra a correspondência entre o tempo em cada estado e a respetiva corrente gasta. De notar que o evento de conexão, repartido nos estados apresentados, ilustra apenas o envio de uma notificação do escravo para o mestre.

Tabela 5.3 - Tempo e corrente dos diferentes estados de um evento de conexão.

Estado	Tempo (µs)	Corrente (mA)
1	400	6
2	340	7,4
3	80	11
4	80	17,5
5	150	7,4
6	296	17,5
7	1280	7,4
8	160	4,1

Assumindo um intervalo de conexão de 100 ms e a latência do escravo igual a zero, o IEC resultante é de 100 ms. De acordo com os valores da tabela obtém-se então um valor de T_{awake} de 2786 µs, I_{ConnEv} igual a 8,4760 mA e um I_{IEC} igual a 0,2371 mA. Este último corresponde ao consumo médio de energia estimado para o módulo CC2540. O valor de I_{Sleep} usado é o mesmo que o referido no exemplo, ou seja, de 1

µA. Sendo assim, assumindo uma bateria do tipo LIR2450 com capacidade igual a 120 mAh a alimentar o módulo, estima-se que este poderá transmitir continuamente (*BatLife*) durante sensivelmente 21 dias. De ressalvar que a análise efetuada não tem em consideração o consumo da placa de sensores, apenas do módulo CC2540.

A Figura 5.8 ilustra graficamente a variação do IEC e o seu efeito direto no período de longevidade da bateria, assumindo o envio de até três notificações por cada evento de conexão.

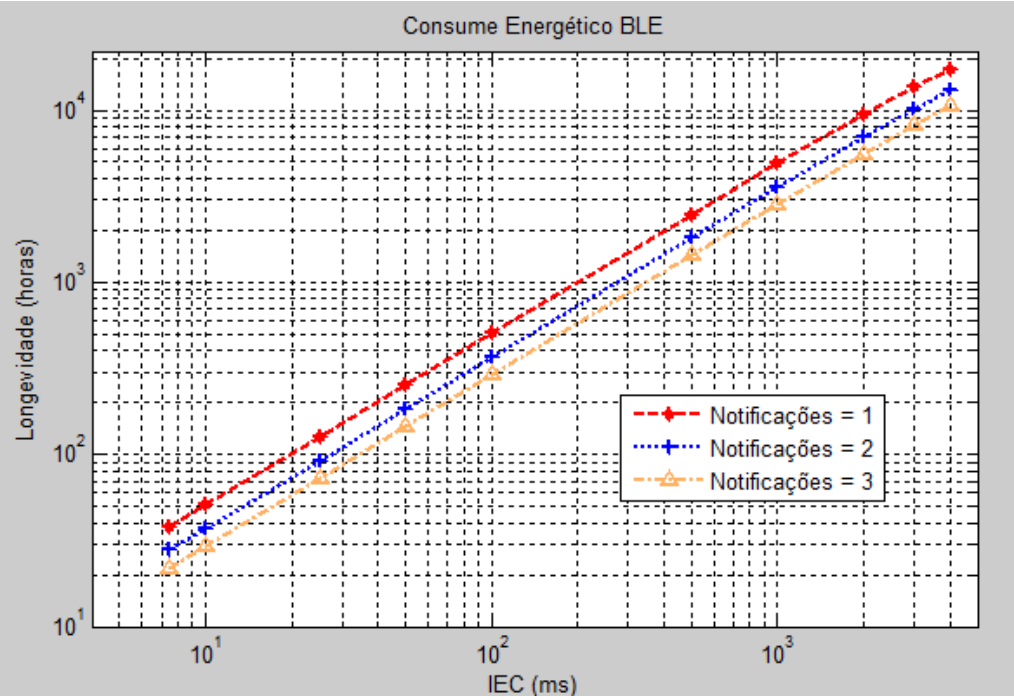

Figura 5.8 - Longevidade da bateria em função do IEC e nº de notificações.

Observa-se que à medida que o valor de IEC aumenta, a longevidade da bateria também aumenta devido ao facto do período de *sleep*, onde os dispositivos não estão a trocar dados, ser maior. Por outro lado, quando o IEC é reduzido, o período de *sleep* é menor, pois os dispositivos acordam frequentemente para proceder à troca de informação. O número de pacotes trocados por evento de conexão é também um fator decisivo na estimativa da longevidade da bateria, pois o período ativo por intervalo de conexão aumenta e consequentemente a corrente gasta é maior.

Os autores em [29] referiram uma longevidade máxima da bateria de até 14 anos. Em contraste, o máximo atingido neste estudo foi aproximadamente 750 dias (18000 horas). Esta diferença deve-se ao facto de se ter considerado a latência do escravo nula e intervalo de conexão de 4000 ms para obter o melhor resultado para o envio de uma notificação, enquanto os autores consideraram um intervalo de conexão de 86.25 ms e uma latência do escravo de 370 de modo a obterem um IEC igual a 32 s. Outro fator foi o uso de uma bateria com capacidade de 230 mAh por parte dos autores, enquanto este estudo foi baseado no uso de uma bateria com capacidade menor, ou seja, 120 mAh. Optou-se por usar este tipo de bateria pois foi a usada para alimentar os módulos sensoriais pertencentes ao sistema implementado, e assim obter uma aproximação o mais real possível. Refazendo os cálculos da secção 4.5 assumindo os valores de IEC e capacidade da bateria usados pelos autores em [29], obtém-se o mesmo valor de atuação do dispositivo escravo, ou seja, aproximadamente 14 anos.

5.5 Colisões na Transmissão

De modo a verificar se o protocolo BLE é capaz de evitar a ocorrência de colisões durante a comunicação entre um mestre e diversos escravos, foram efetuados testes experimentais onde se fixou o IEC com o valor de 100 ms e o número de notificações por evento de conexão igual a três, valor usado no sistema de monitorização de postura. Os testes foram feitos analisando os identificadores de sequência dos pacotes (notificações) e aplicando uma média móvel sobre o *delivery ratio* (em % corresponde a 100 – PER) utilizando uma janela com largura igual a 20 amostras. Nestes testes foi monitorizado um escravo apenas, verificando se durante a transmissão das suas notificações ocorrem perdas significativas à medida que se aumenta o número de escravos conectados.

Num primeiro teste, efetuou-se a comunicação entre um mestre e dois escravos, durante 10 horas e observou-se o comportamento apresentado na Figura 5.9.

Verificou-se uma taxa de entrega de pacotes por parte do módulo monitorizado de 100% neste caso.

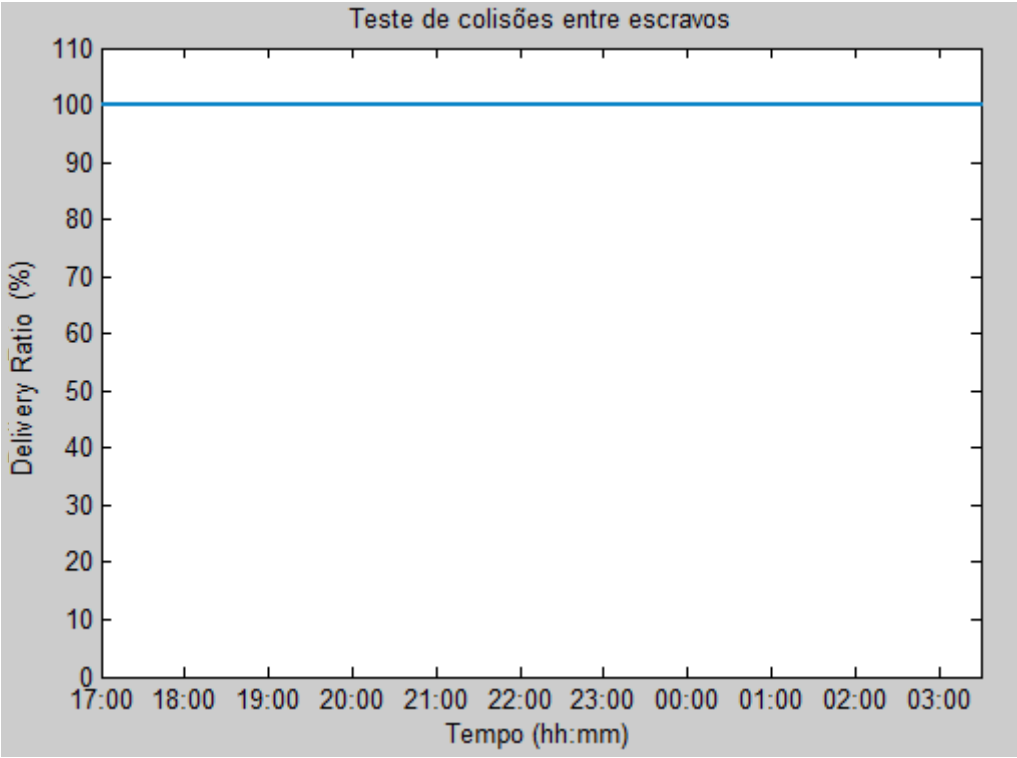

Figura 5.9 – Teste de colisões de pacotes numa ligação entre 1 mestre e 2 escravos.

De modo a confirmar a inexistência de sobreposição no envio da informação, averiguou-se a situação de uma ligação com 4 escravos durante um período de tempo ainda maior, 18 horas, usando assim todos os módulos disponíveis. O *IEC* continuou afixado em 100 ms e o número de notificações por evento de conexão igual a três. Os resultados observados estão representados graficamente na Figura 5.10.

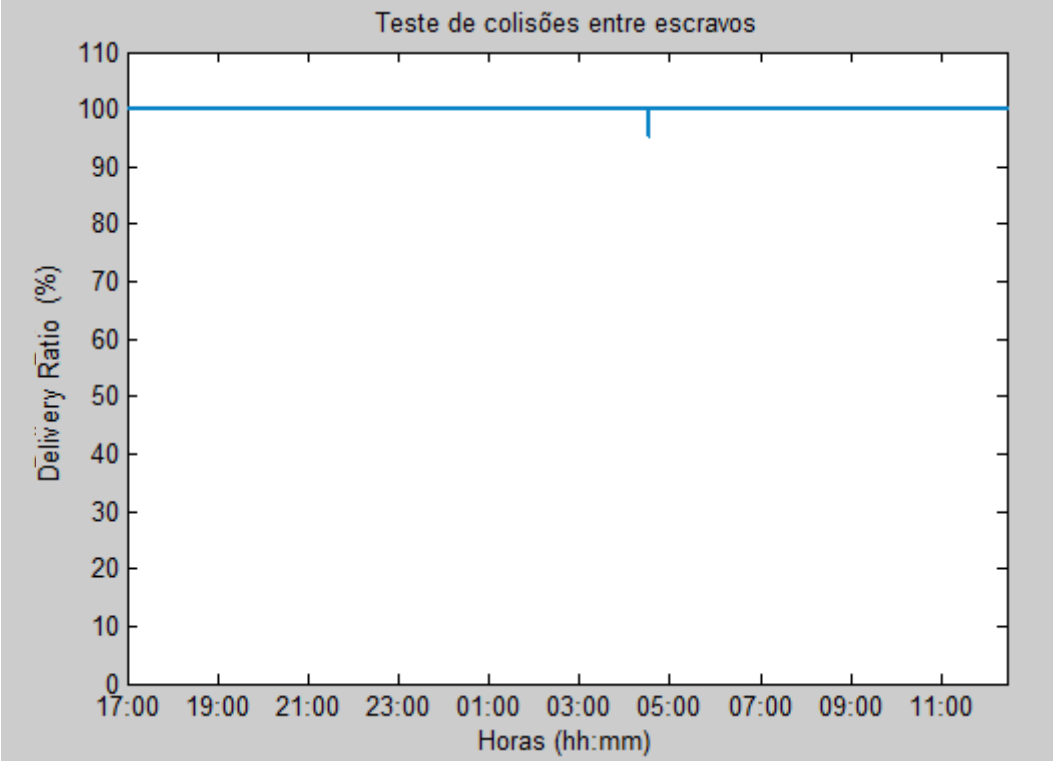

Figura 5.10 - Teste de colisões de pacotes numa ligação entre 1 mestre e 4 escravos.

De acordo com o gráfico, verificou-se uma taxa de entrega de pacotes de 95 %, mas apenas durante um instante entre as 03:00 e 05:00, ou seja, apenas um pacote se perdeu, pois 5% numa janela de 20 corresponde a 1. Excetuando este instante, durante a duração do teste (18 horas) não ocorrem perdas na transmissão dos dados por parte do módulo monitorizado, sendo a taxa de entrega de 100 %.

Os resultados obtidos mostram que todos os nós transmitem em períodos diferentes durante o tempo todo, havendo assim um mecanismo de sincronização no protocolo BLE que evita o *clock drift* e a consequente sobreposição dos períodos de transmissão, ou seja, dos eventos de conexão.

5.6 Número de Escravos Suportados

De modo a averiguar a capacidade da rede BLE implementada, no que diz respeito ao número de escravos suportados, foram realizadas análises teóricas e testes de índole e experimental.

5.6.1 Análise Teórica

No que diz respeito aos resultados da análise teórica, foi feita uma comparação entre o protocolo BLE e o protocolo LPRT, aplicada ao SMP [4] (secção 3.1), em condições idênticas de tráfego, de modo a averiguar as semelhanças e diferenças no que diz respeito ao número de escravos suportados em redes implementadas com estes protocolos. A Tabela 5.4 ilustra a definição dos parâmetros do SMP comuns às duas redes.

Tabela 5.4 - Parâmetros do SMP.

Parâmetro	Nome	Valor
Número de sensores	N_S	9 (3 sensores * 3 eixos)
Frequência de amostragem	f_S	30 Hz
Resolução dos sensores	Q_S	16 bits

A Tabela 5.5 e a Tabela 5.6 contêm a definição dos parâmetros usados em cada teste teórico para os dois protocolos. De notar que, para ambos, a informação sensorial é a definida para o sistema de monitorização de postura implementado no âmbito desta dissertação, ou seja 18 bytes.

Tabela 5.5 - Parâmetros do protocolo LPRT.

Parâmetro	Nome	Valor
Taxa de transmissão de dados	R_{LPRT}	250 kbps
Overhead camada PHY	OH_{PHY}	6 bytes
Overhead camada MAC	OH_{MAC}	9 bytes
Payload aplicacional	$Payload_{App}$	54 bytes
Duração da superframe	T_{SF}	100 ms
Mini-slots p/ superframe	M_{SF}	500

A duração da superframe (T_{SF}) foi definida em 100 ms. Com este valor e com uma frequência de amostragem (F_S) de 30 Hz, o número de amostras dos dados dos sensores é igual a 3 ($N_{SAMPLES}$). O tamanho do payload definido na aplicação ($Payload_{App}$) é igual a 54 bytes. Com 9 sensores amostrados a 16 bits, obtemos os 18 bytes de dados sensoriais. Para 3 amostras, o $Payload_{App}$ aumenta para 54 bytes. Este sistema adiciona ainda à mensagem um byte extra contendo a informação do nível de bateria, mas para se manter uma aproximação entre os dois protocolos, apenas vão ser considerados os dados dos sensores.

A duração mínima do período CP foi definida em 11 ms, o que é suficiente para suportar a troca de duas tramas com tamanho máximo, incluindo o *backoff* imposto pelo CSMA/CA e o período entre as tramas. A duração máxima do período CFP é a duração da superframe (100 ms), menos um período de reserva (15,26 ms), que é composto pelo CP mais o tempo necessário para transmitir um beacon de tamanho máximo (4,26 ms). Posto isto, o período CFP resulta em 84,74 ms, o que corresponde a um total de 423 *mini-slots*. De acordo com os valores definidos na Tabela 5.5, o tamanho do PDU a enviar, é de 69 bytes. Isto significa que o período de transmissão do mesmo é igual a 2,21 ms e o número de *mini-slots* necessários é de 11. É usado um *mini-slot* adicional para servir de tempo de guarda entre as transmissões dos nós sensoriais, resultando assim em 12 mini-slots para cada mensagem. A divisão no número de mini-slots disponíveis no CFP pelo número por mensagem é 35,25. Posto isto, o número máximo de transmissões que podem ocorrer durante o período CFP é igual a 35, ou seja, até 35 nós sensoriais podem estar conectados a enviar os seus dados.

De notar que existe um compromisso entre o atraso, energia consumida e a capacidade da rede (nós suportados). Para um aumento da duração da *superframe*, existe a tendência de a capacidade da rede aumentar e a energia consumida diminuir, pois o tamanho do $Payload_{App}$ aumenta, reduzindo o overhead do protocolo. Contudo o atraso tende a aumentar.

Tabela 5.6 - Parâmetros do protocolo BLE.

Parâmetro	Nome	Valor
Taxa de transmissão de dados	R_{BLE}	1 Mbps
Overhead camada PHY	OH_{PHY}	10 bytes
Overhead camada LL	OH_{LL}	4 bytes
Overhead camada L2CAP	OH_{L2CAP}	3 bytes
Payload aplicacional	$Payload_{App}$	18 bytes
Intervalo de conexão	*connInterval*	100 ms
Latência do escravo	*slaveLatency*	0
Intervalo entre eventos de conexão ativos	*IEC*	100 ms
Notificações por evento de conexão	N_{Noti}	3

No que diz respeito ao protocolo BLE, o IEC foi definido em 100 ms. Tal como no protocolo anterior, com uma frequência de 30 Hz, o número de amostras resultantes é igual a 3. Ao contrário do que acontece no protocolo LPRT, cujo tamanho máximo do PDU pode ir até aos 133 bytes, no BLE o tamanho máximo é de 37 bytes, dos quais apenas 20 destes podem ser usados para dados da aplicação. Para acomodar os dados de 3 amostras dos sensores, são necessárias enviar 3 notificações pois não cabem todas na mesma notificação. Do mesmo modo que foi feito para o protocolo anterior, apenas vai ser considerado os dados dos sensores como payload da aplicação, descartando os bytes extras de identificador de sequência e identificação do módulo sensorial. Dado que um pacote do mestre demora 80 µs e um pacote do escravo (contendo a notificação de 18 bytes) demora 280 µs, ao fim do envio das 3 amostras dos sensores e respetivos T_{IFS}, o tempo total é de 1,98 ms. Posto isto, se para a transmissão de 3 amostras dos sensores por parte de um nó, o tempo é de 1,98 ms, durante os 100 ms de IEC é possível acomodar 51 nós sensoriais.

A Figura 5.11 ilustra graficamente a comparação entre os protocolos LPRT e BLE, no que diz respeito à capacidade da rede (nós suportados), variando a duração da

superframe (no LPRT) e o correspondente intervalo entre eventos de conexão ativos (no BLE), mas mantendo os restantes parâmetros das Tabela 5.4 a 5.7.

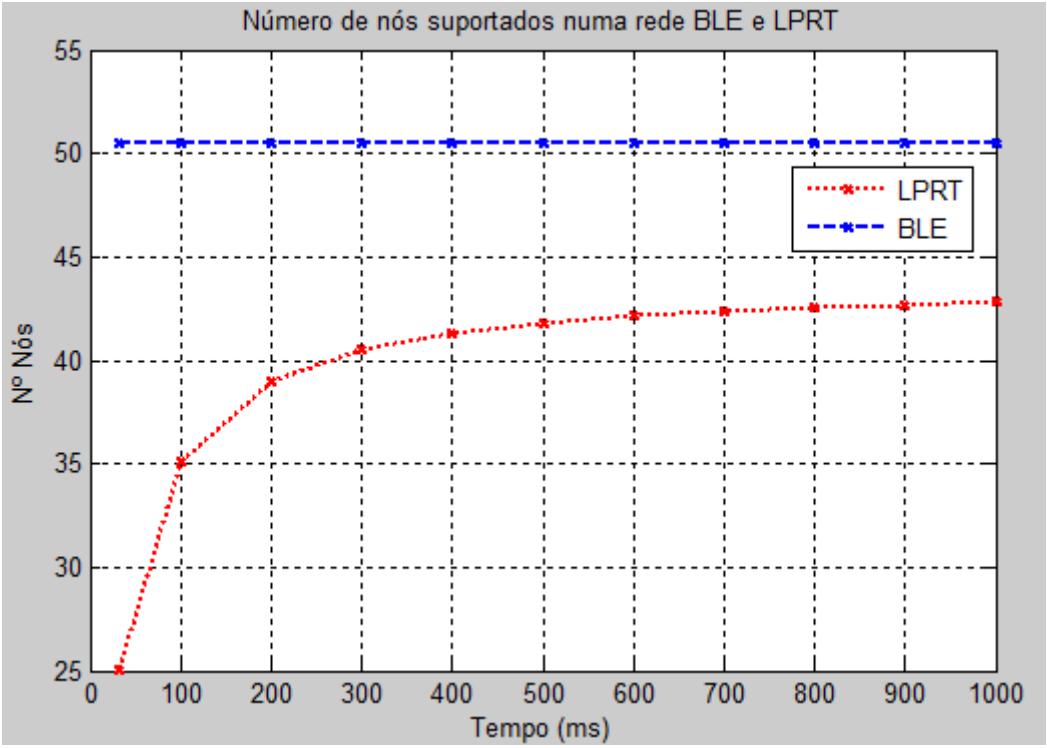

Figura 5.11 - Relação entre o número de nós suportados dos protocolos BLE e LPRT.

Como já foi referido, no caso do LPRT, à medida que o período da *superframe* aumenta, a capacidade da rede também aumenta, como mostra a Figura 5.10, pois o overhead inerente ao protocolo diminui face à quantidade de dados da aplicação. Por outro lado, no protocolo BLE, aumentando o intervalo entre eventos de conexão ativos (IEC), o número de nós suportados mantém-se em 51. Isto deve-se ao facto de cada amostra do conjunto de sensores ser enviada num pacote separado de tamanho fixo (notificações), existindo um overhead fixo associado a cada pacote.

De modo a obter uma estimativa do número de nós suportados pelo protocolo LPRT caso a camada física deste fosse a mesma do BLE no que diz respeito à taxa de transmissão de dados, efetuou-se os cálculos para a obtenção do número máximo de escravos, considerando, no cálculo do tempo de transmissão do PPDU, R_{LPRT} igual ao R_{BLE}, ou seja, 1 Mbps. O resultado pode ser observado na Figura 5.12, através da nova curva introduzida no gráfico anterior.

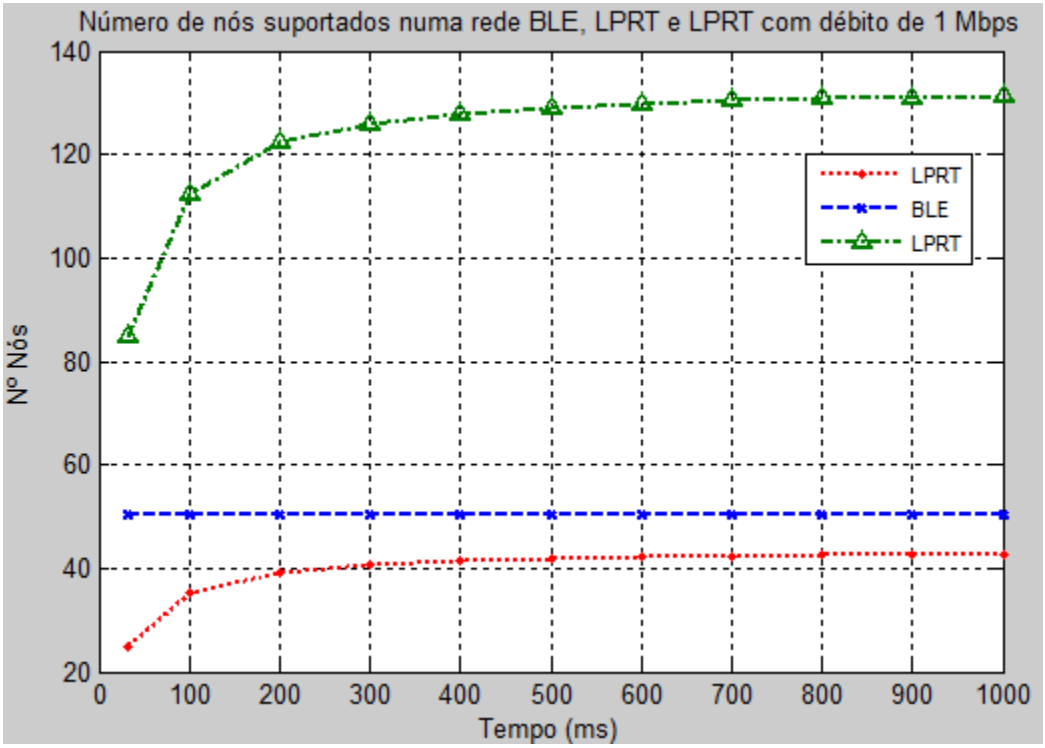

Figura 5.12 - Relação do número de escravos suportados nos protocolos: BLE, LPRT e LPRT com débito de 1 Mbps.

Os resultados ilustram que se o protocolo LPRT assentasse numa cada física idêntica à do BLE, relativamente à taxa de transmissão de dados, o número de escravos suportados seria superior ao dos protocolos BLE e LPRT, para todos os intervalos avaliados.

5.6.2 Avaliação Experimental

Seguidamente, apresentam-se resultados experimentais usando módulos CC2540, em que um deles desempenha o papel de mestre e outros três módulos o papel de escravos. Durante esta fase da dissertação, averiguou-se a possibilidade de adaptar o código fonte da aplicação central (mestre) para gerir múltiplas conexões pois não existe essa funcionalidade de origem. Chegou-se à conclusão que a versão da stack BLE da Texas Instruments não disponibiliza nenhuma aplicação de exemplo/teste para gerir múltiplas conexões, a não ser recorrendo ao software BTool, o único meio possível encontrado para tal. Já foi dito na secção 3.3.3 que,

para este software funcionar devidamente, é necessário programar o módulo CC2540 que desempenha o papel de mestre com um firmware apropriado, cujo código fonte não é disponibilizado. Recorreu-se, assim, ao software BTool para gerir múltiplas conexões, já que deste modo é possível assim conectar vários dispositivos escravos e observar os respetivos resultados dessa ação.

Com um IEC de 100 ms, testou-se o cenário de uma conexão entre um mestre e dois escravos, de modo a averiguar a possibilidade desta conexão e também se o número de notificações por evento de conexão se mantinha em três, ou seja, o mesmo comportamento de uma conexão entre um mestre e um único escravo. A Figura 5.13 representa um extrato do *log* de mensagens do BTool, onde é possível observar a conexão a dois dispositivos escravos.

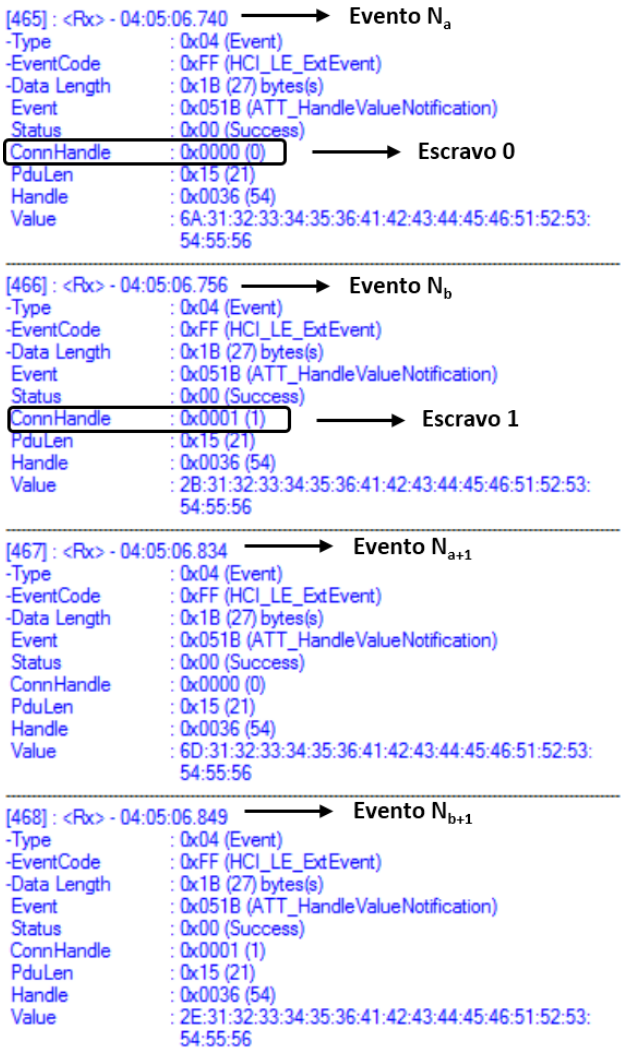

Figura 5.13 - Conexão entre mestre e dois escravos, com IEC igual a 100 ms.

Observou-se que os dois dispositivos escravos, quando conectados simultaneamente ao mestre, só enviam um pacote (notificação) por cada evento de conexão, ao invés de três, verificando-se a perda dos dois pacotes em falta. Os dispositivos escravos distinguem-se através do parâmetro "*ConnHandle*", sendo que este é basicamente o identificador da conexão, que incrementa sequencialmente à medida que o número de conexões também aumenta.

Podemos observar que durante o evento N_a, o dispositivo cujo identificador de ligação (*ConnHandle*) é o "0x00", envia uma notificação que tem como identificador de sequência o número "0x6A" (primeiro byte do campo "*Value*"). No evento seguinte (N_{a+1}), o mesmo dispositivo, volta a enviar uma notificação, sendo que esta tem como número de sequência o valor "0x6D". Nota-se assim que as notificações "0x6B" e "0x6C" foram perdidas.

O mesmo acontece para o escravo cujo identificador de conexão é o valor "0x01", que envia uma notificação, no evento N_b, com número de sequência igual a "0x2B", sendo que a próxima é "0x2E" (evento N_{b+1}), verificando-se assim de igual modo uma lacuna nos números de sequência recebidos.

Efetuando a conexão com um mestre e três escravos, verifica-se o mesmo comportamento, ou seja, cada escravo envia apenas uma notificação por cada evento de conexão, onde as restantes duas notificações de cada um se perdem.

Concluiu-se que, quando existem conexões simultâneas, apenas é possível enviar um pacote de cada escravo por cada evento de conexão. Este resultado está de acordo com a especificação da versão da *stack* da Texas Instruments usada na implementação do sistema (versão 1.3.2) [17]. A mesma especificação também refere que o máximo de ligações suportadas simultaneamente é igual a três.

De modo a contornar esta limitação do número de notificações por evento de conexão, aquando da conexão simultânea de mais do que um escravo, e manter uma frequência de amostragem de 30 Hz, como é desejável, diminuiu-se o IEC para 33,75 ms. O valor não pode ser fixado em 33,33 ms pois o valor do intervalo de conexão definido no firmware é expresso em unidades de 1.25 ms e para tal teve de ser igual a 27 para assim resultar em 33,75, de modo a ser próximo do valor pretendido

(33,33 ms). Deste modo, programou-se os escravos para enviarem apenas uma notificação a cada evento de conexão, de modo a que ao fim de três eventos de conexão (aproximadamente 100 ms), cada um tenha enviado um total de três notificações, não comprometendo assim os parâmetros definidos inicialmente para a conceção do sistema de monitorização de postura. A Figura 5.14 ilustra um extrato do software BTool, onde já se verifica o comportamento desejado, ou seja, a cada evento de conexão, cada escravo envia uma notificação, sendo que no evento seguinte torna a enviar, sem comprometer a sequência das mensagens.

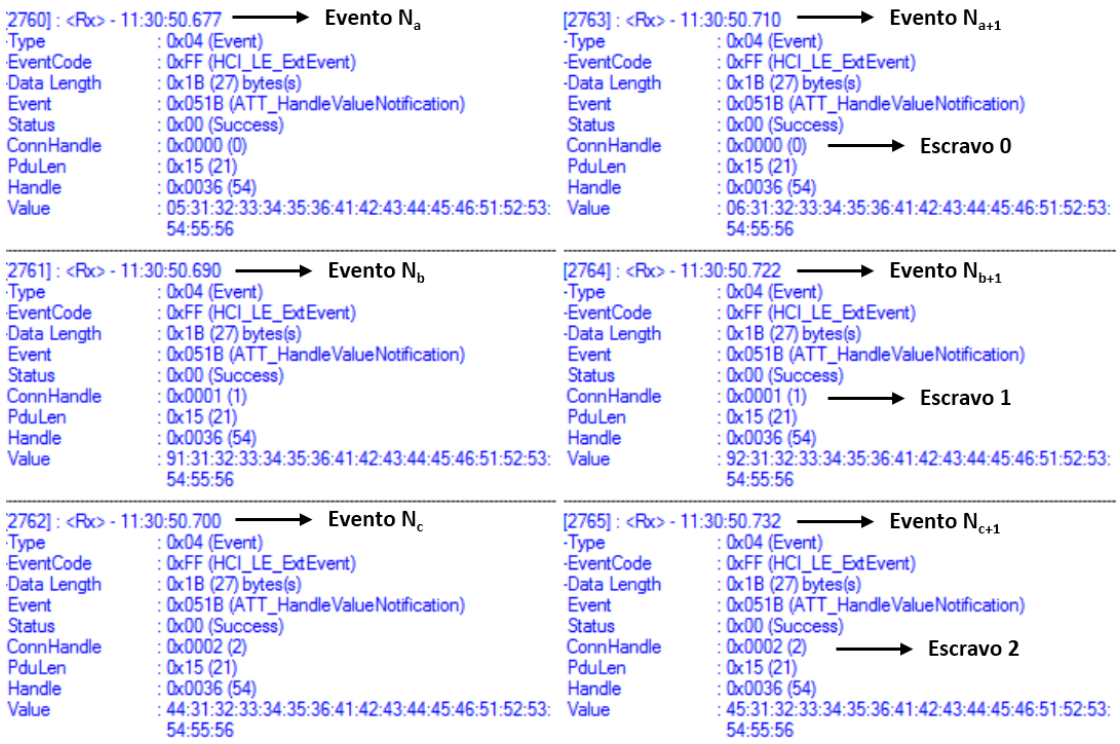

Figura 5.14 - Conexão entre mestre e três escravos, com IEC igual a 33,75 ms.

Migrando a estação central (dispositivo mestre) para o smartphone, observou-se que não seria necessário diminuir o IEC para 33,75 ms, pois mesmo com conexões simultâneas, o número de notificações de cada escravo por evento de conexão mantinha-se igual a três. A Figura 5.15 ilustra um extrato do log de mensagens do Android, via software Eclipse, onde se verifica a presença de 4 escravos a enviarem as suas notificações.

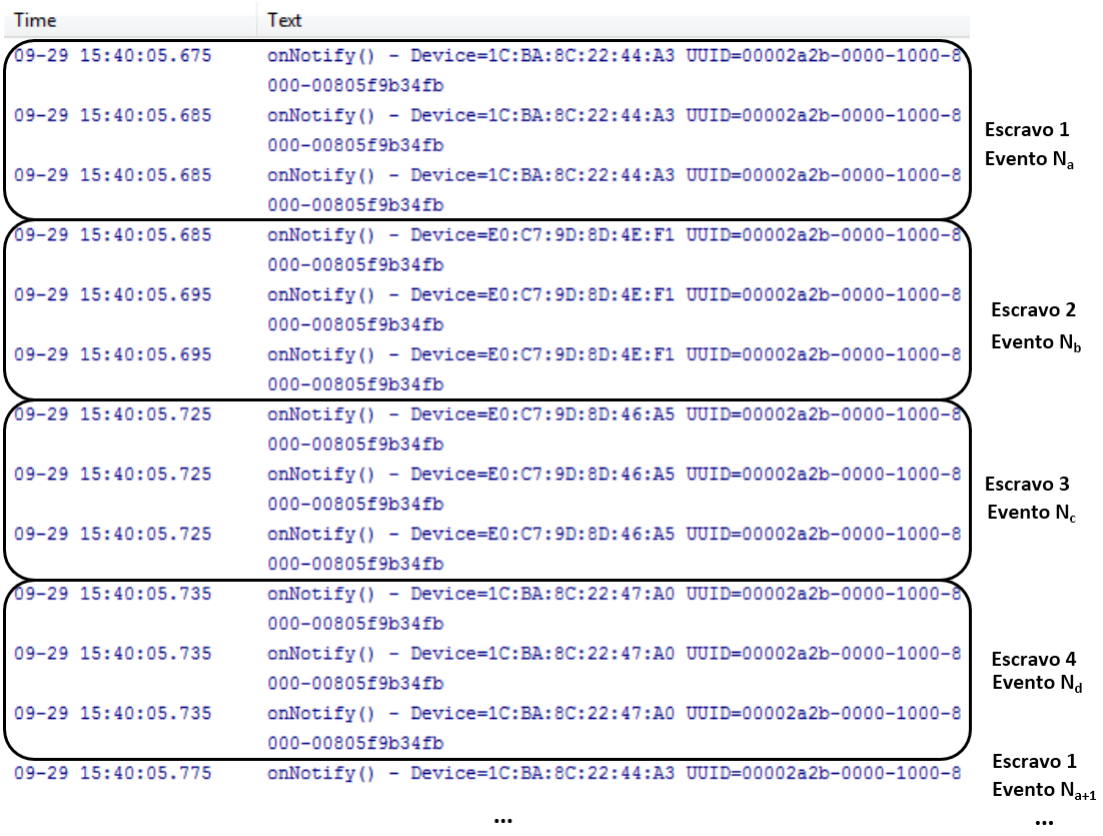

Figura 5.15 – Conexão entre mestre e 4 escravos, com IEC de 100 ms (Mestre – SmartPhone Android).

Entre a API BLE do Android e a stack BLE da Texas Instruments residem algumas diferenças no que toca à capacidade de acomodação de dispositivos escravos. A primeira diferença reside exatamente no número de escravos suportados, em que um módulo CC2540 no papel de mestre apenas suporta um máximo de três conexões (segundo a especificação [17]), enquanto com o smartphone usado com a API usada, pelo que foi averiguado, não se encontrou um limite concreto na especificação ([22]). Para o primeiro caso testou-se apenas com os 3 módulos disponíveis para desempenharem o papel de escravos, já que dos quatro módulos existentes, um teve de desempenhar o papel de mestre. No segundo caso, como o mestre passou a ser o smartphone, já foi possível utilizar os quatro módulos como escravos, concluindo assim que no mínimo, uma ligação BLE onde o mestre é um dispositivo Android, são suportado quatro escravos. Outra diferença está no número de notificações que é possível enviar quando se está perante um caso de conexões

múltiplas. No caso do CC2540, é apenas possível enviar uma de cada escravo por cada evento de conexão, enquanto no dispositivo Android, o número de notificações mantem-se igual, quer seja uma ligação de um mestre para um escravo ou, como neste caso, de um mestre para vários escravos.

5.7 Postura do Atleta

Para averiguar o resultado das medições efetuadas pelos nós sensoriais, foram efetuados testes com os sensores colocados num indivíduo, nos segmentos corporais definidos inicialmente. Foi colocado um sensor no peito de modo a calcular o ângulo de inclinação do tronco, entre duas posições de referência: posição ereta (tronco perpendicular ao solo) e posição aerodinâmica (tronco inclinado para a frente). Este cálculo foi realizado com o auxílio do cálculo da inclinação da bicicleta, pois só dessa forma é possível obter a real inclinação do tronco do atleta, em função do declive do terreno. Para a obtenção do valor da inclinação da bicicleta, recorreu-se ao sistema de sensores do smartphone do sistema. De forma a obter a variação do ângulo do joelho do atleta durante o pedalar, um sensor foi colocado na parte superior da perna (coxa) e outro na parte inferior da perna (canela). A Figura 5.16 ilustra a colocação dos sensores nos segmentos corporais assinalados.

Figura 5.16 - Colocação dos sensores: no tronco (esq.) e na perna (dir.).

De modo a avaliar a fiabilidade do sistema foram definidos 5 cenários de teste: cenário de referência (A), cenário de repouso (B), cenário de ataque (C), cenário de pedalar (D) e cenário de inclinação (E). Os três primeiros cenários estão representados na Figura 5.17. O cenário D pretende ilustrar o efeito de um goniómetro[8] através do cálculo do ângulo do joelho e o cenário E tem como objetivo ilustrar o comportamento do ângulo do tronco em função do declive do terreno. A postura do indivíduo durante os testes é de um atleta de ciclismo não treinado que apenas se limitou a recriar as posições do tronco de forma idêntica às da Figura 5.17. Quanto às posições do joelho, para cada caso, foram definidas de acordo com o propósito a testar.

Figura 5.17 - Cenário A - Referencia (esq.), cenário B - Repouso (centro) e cenário C- Ataque (dir.).

Os resultados para cada cenário apresentados as seguir são representados graficamente por três curvas: azul (ângulo do joelho), vermelha (inclinação da bicicleta) e verde (inclinação do tronco/peito).

5.7.1 Cenário A – Referência

No que diz respeito ao cenário A, onde o atleta apresenta uma posição de referência, o seu tronco está sensivelmente perpendicular ao solo, apresentando

[8] Um goniómetro é um instrumento de medida de forma circular ou semicircular, graduada em 360 ou 180 graus, respetivamente, utilizada para medir ou construir ângulos.

assim um ângulo de inclinação praticamente nulo. Nesta posição o ângulo do joelho apresentou-se em média em 106 graus e a inclinação da bicicleta é nula. Neste cenário o atleta não se encontra a pedalar e a bicicleta está imóvel. A Figura 5.18 ilustra graficamente valor dos ângulos obtidos pelos diferentes sensores, nas condições do cenário de referência.

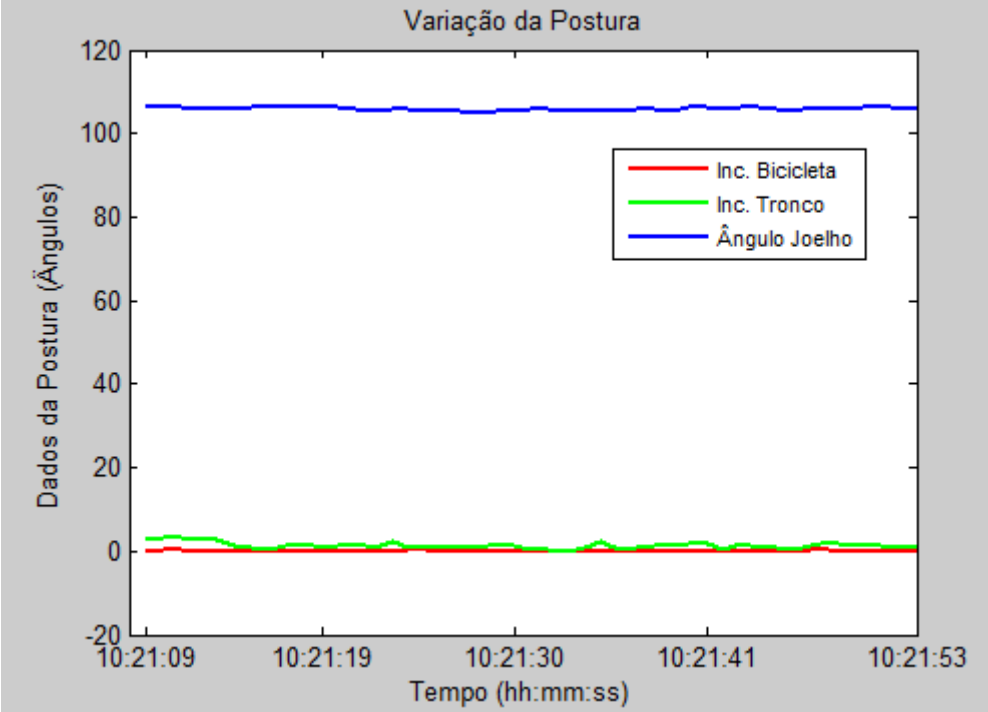

Figura 5.18 - **Representação gráfica do cenário A (Referência).**

5.7.2 Cenário B - Repouso

Relativamente ao cenário de repouso (B), foi testada a posição quando o ciclista está ligeiramente com o tronco inclinado para a frente, com valores próximos dos 16 graus. Num momento inicial (Figura 5.19) o ciclista encontra-se apenas em cima da bicicleta sem pedalar não se verificando alterações no valor do ângulo do joelho e com pouca variação da inclinação do tronco. Apesar do atleta estar numa posição sem pedalar e com o tronco numa posição fixa, este apresenta alguma variação, devido ao respirar, inerente ao facto do sensor estar colocado próximo da caixa torácica do atleta. A partir do instante 10:40:12, o atleta inicia a marcha, verificando-se variações do ângulo do joelho no intervalo de 60 a 110 graus, assim como,

obviamente, uma variação mais acentuada da inclinação do tronco, embora pouco significativa. A inclinação da bicicleta manteve-se nula durante as fases do teste. Neste cenário não foi considerado o facto do ângulo do joelho ser o mais correto ou não, apenas se pretendeu ilustrar o movimento do mesmo e as diferenças angulares.

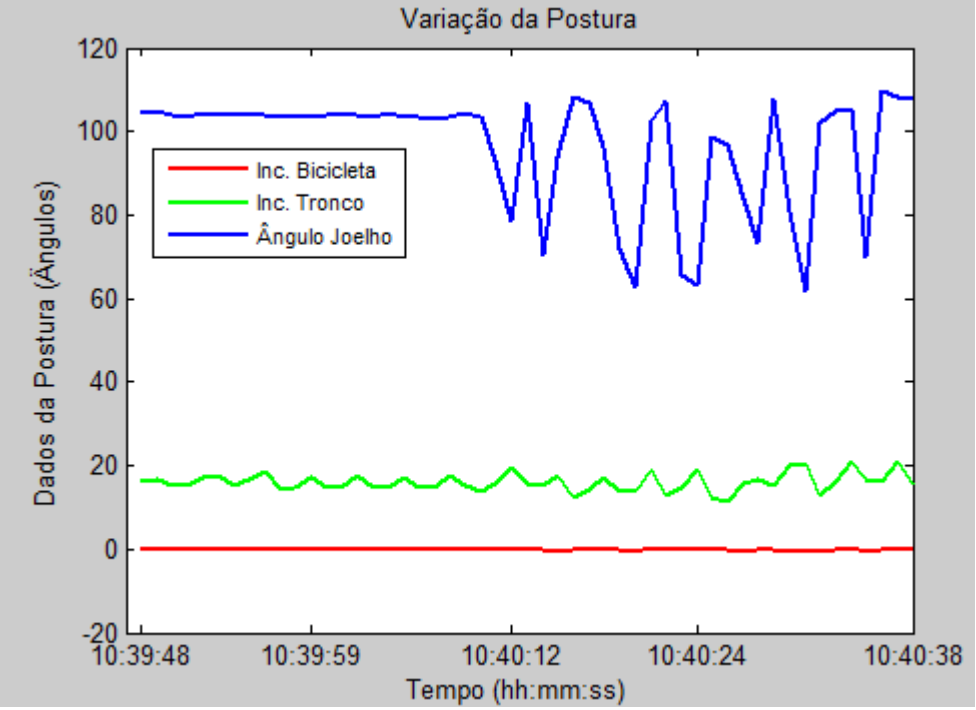

Figura 5.19 **- Representação gráfica do cenário B (Repouso).**

5.7.3 Cenário C - Ataque

Para descrever uma posição por parte do atleta mais aerodinâmica, recorreu-se a testes baseados no cenário de ataque (C), onde o atleta está com o tronco ainda mais inclinado para a frente e com uma das pernas, neste caso a que tem os sensores colocados, bem mais esticada. A altura do selim em relação à altura dos pedais é também um fator importante para garantir um máximo de desempenho neste cenário de teste. O objetivo não é o atleta esticar totalmente a perna ao pedalar, nem por outro lado ficar demasiadamente encolhida, mas sim de modo a dar o máximo rendimento a cada movimento de pedalada. A Figura 5.20 ilustra os resultados dos testes ao cenário C. Os valores registados para o ângulo de inclinação do peito rondam os 50 graus, ou seja, o tronco mais fletido para à frente em relação

ao cenário anterior. O ângulo do joelho numa primeira fase, sem pedalar e com a perna próxima da posição esticada, apresenta um valor próximo de 160 graus (seria 180 graus caso estivesse totalmente esticada, fazendo o joelho um ângulo raso). A partir do instante 10:47:30 o atleta inicia a marcha, verificando-se a variação do ângulo do joelho entre 80 e 150 graus.

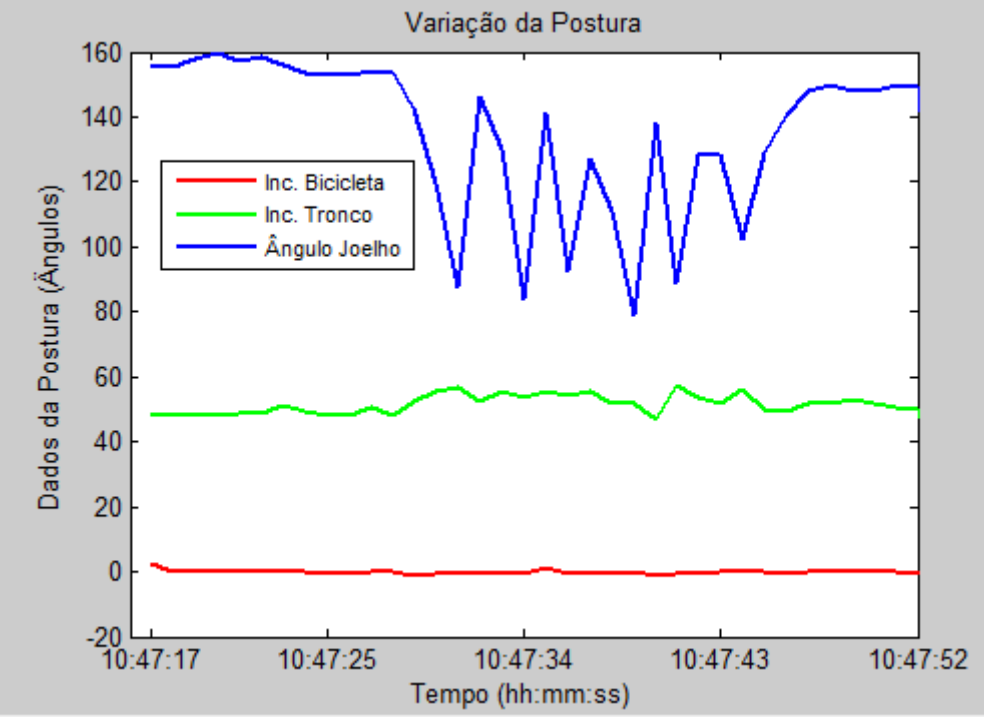

Figura 5.20 - Representação gráfica do cenário C (Ataque).

5.7.4 Cenário D - Pedalar

Para avaliar somente o ângulo do joelho de forma isolada, procedeu-se ao pedalar livre sem ter em atenção a inclinação da bicicleta e do tronco. Observa-se na Figura 5.21 que o ângulo inicial está fixado em 180 graus, representando a inexistência de flexão da perna. Quando o pedalar é iniciado lentamente, este ângulo vai oscilando entre 180 graus e 80 graus, de modo a refletir os valores conseguidos pelo atleta. Numa fase de pedalar com mais intensidade, entre os instantes 10:58:12 e 10:58:31, o ângulo varia menos, entre 150 e 80 graus.

Em [35] é referido que o assento deve estar regulado por forma a permitir um ângulo do joelho entre 25 a 35 graus para os ciclistas não treinados, contribuindo

para a prevenção de lesões e aumento do desempenho. Os autores medem o ângulo de forma inversa ao que é feito nesta dissertação, ou seja, consideram um ângulo de 0 graus quando a perna está totalmente esticada, e não 180 graus. O resultado final é o mesmo, ou seja, 150 graus neste teste correspondem ao mesmo nível flexão da perna que 30 graus no artigo referenciado. De recordar que esta medida é a considerada ideal quando o ciclista se coloca na bicicleta dependendo assim fortemente do comprimento das pernas dos atletas e da altura do selim, pelo que este deve ser ajustado para que o joelho apresente este ângulo independentemente da estatura do atleta em questão.

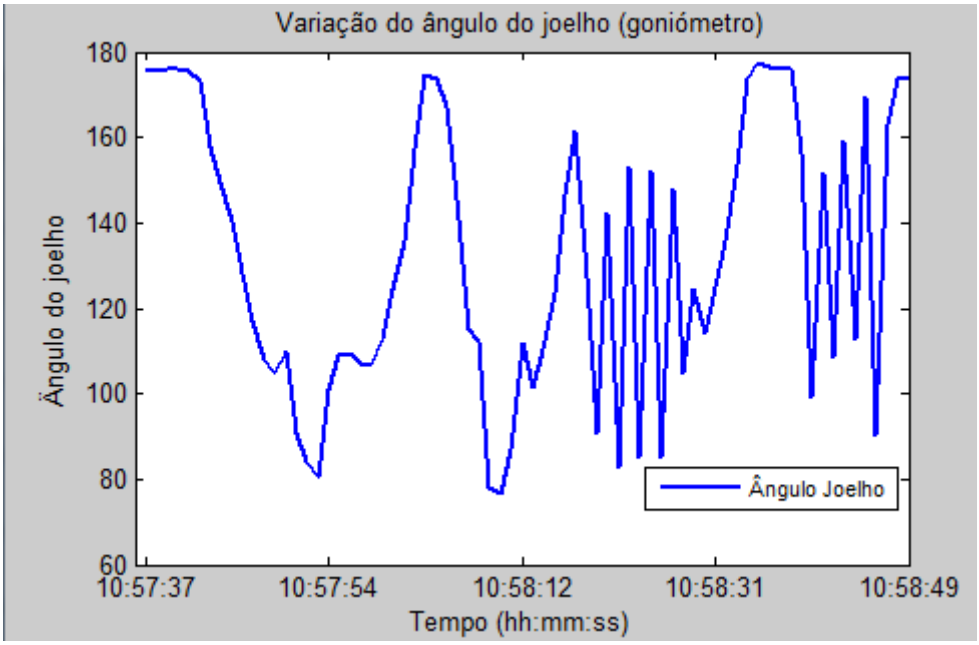

Figura 5.21 - Representação gráfica do cenário D (pedalar).

5.7.5 Cenário E – Inclinação

Por fim, para avaliar um cenário onde o valor medido para a inclinação do tronco do atleta tem de se manter fixo (se este não a alterar) independentemente da inclinação do terreno, fixou-se a inclinação do tronco de modo a formar um ângulo de sensivelmente 45 graus e observou-se se esta variou à medida que a inclinação do terreno se alterou. Nos instantes iniciais, a bicicleta não apresenta qualquer inclinação e o tronco do atleta está na posição vertical. Posteriormente este efetua uma inclinação do tronco para a frente e com o decorrer do andamento, primeiro

numa subida e depois numa descida, o ângulo de inclinação do tronco não sofre alterações bruscas, como ilustra a Figura 5.22. O facto de a inclinação do tronco do atleta ser calculada com o auxílio da inclinação da bicicleta tem como vantagem obter a inclinação real do tronco, o que de outra forma não seria possível, pois se esta fosse calculada isoladamente, aquando de variações na inclinação do terreno, mesmo que o atleta estivesse numa posição sempre constante, esta ia variar com o declive.

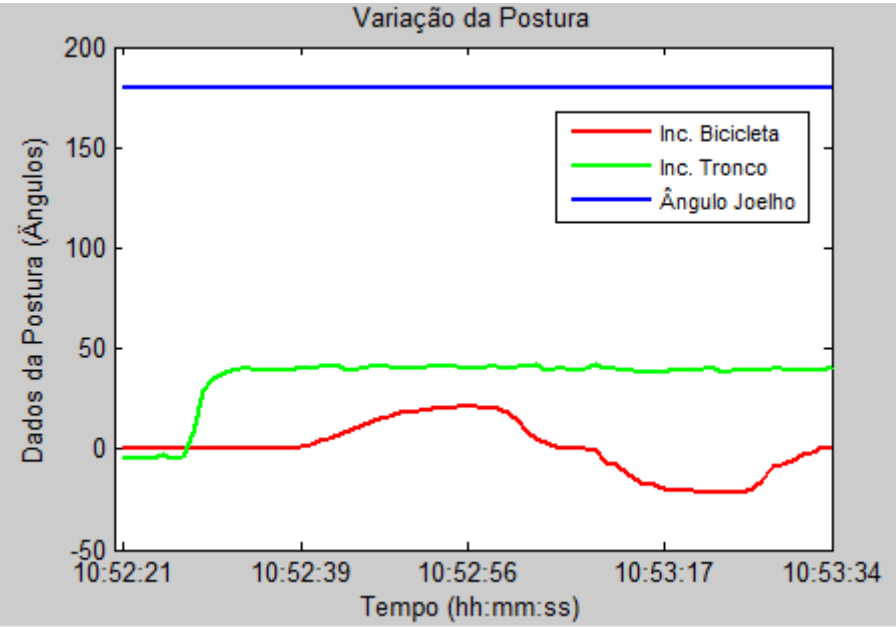

Figura 5.22 - Representação gráfica do cenário E (Inclinação).

6. Conclusões e Trabalho Futuro

O desenvolvimento de sistemas baseados em redes de sensores sem fios tem tido um comportamento promissor face às novas necessidades do mercado, nomeadamente na área do desporto, saúde e entretenimento. Cada vez mais se requerem sistemas que minimizem o consumo energético e que ao mesmo tempo consigam manter a fiabilidade dos dados enviados e recebidos. A miniaturização dos diversos componentes da rede contribui em grande parte para um amento da conceção das redes de área corporal e consequentemente a utilização destas em sistemas ligados às áreas acima mencionadas. A par do IEEE 802.15.4 e ZigBee, um dos principais *standards* para este tipo de redes é o Bluetooth, cuja sua versão tecnológica mais recente, o Bluetooth Low Energy, foi descrita em detalhe nesta dissertação.

Esta dissertação tinha como principal objetivo desenvolver um sistema de monitorização de postura (assente em sensores de captura de movimentos), baseado em BLE, que pudesse ser aplicado a um atleta de ciclismo durante as suas sessões de treino ou competição. Inicialmente foi proposto fazer um estudo do estado da arte do tema de modo a conhecer as diferentes tecnologias e áreas de interesse inerentes ao desenvolvimento do sistema. Foi estudado em detalhe o protocolo de comunicação BLE assim como a plataforma Android, onde assenta grande parte do trabalho. Por fim abordou-se o trabalho relacionado no que diz respeito a aplicações baseadas em sensores, protocolos de comunicação possíveis e também sobre a postura de um atleta de ciclismo. Posteriormente foi implementada uma infraestrutura de rede de comunicação BLE entre dois módulos CC2540 da Texas Instruments e efetuaram-se testes de desempenho à mesma. Estes testes incluíram parâmetros como o débito, consumo energético, latência da comunicação e número de escravos suportados, tendo em atenção os principais parâmetros de conexão do BLE, que são o intervalo de conexão e a latência do escravo. Nesta fase

de comunicação BLE foi criado um serviço que permitia o envio periódico de tramas entre mestre e escravo. Esta implementação serviu de base para a comunicação final entre mestre e diversos escravos. As principais dificuldades nesta fase do projeto centraram-se na compreensão do protocolo BLE e do seu funcionamento. A leitura e estudo intensivo da especificação do *standard* BLE e da especificação do CC2540 da Texas Instruments foram determinantes para ultrapassar esta fase inicial.

Ultrapassada a fase de estabelecimento da comunicação entre duas estações BLE, procedeu-se à aquisição real dos dados sensoriais por parte do módulo de interface rádio. Para o efeito foi usado o protocolo SPI, que permitiu a leitura/escrita dos dados através da USART0 do módulo de interface rádio. Inicialmente, o objetivo era usar o controlador DMA da stack do BLE para a aquisição de dados via SPI, mas tal não se concretizou, pois este controlador define um tipo específico de pacote, que não coincide com o formato aceite pelo MPU-6000, que é o componente de onde se extrai a informação dos dados sensoriais. Deste modo implementou-se uma função de leitura/escrita sem recorrer à stack do BLE para o efeito. Nesta fase, a maior dificuldade a ser superada foi a compreensão do protocolo SPI e a sua configuração para o funcionamento desejado. O facto de não usar os procedimentos SPI disponibilizados pela *stack* tornou de certa forma mais demorada a implementação da comunicação SPI.

A introdução da plataforma Android no trabalho de dissertação era um dos objetivos seguintes a ter em conta. A inserção deste conceito baseou-se na migração da estação central para um smartphone Android, que outrora era desempenhada por um módulo CC2540. A motivação que levou à integração de um smartphone Android neste sistema foi o facto de este incorporar hardware e software compatível com o BLE, e além disso o facto de ser uma plataforma de código aberto permite um fácil acesso a todas as ferramentas para o desenvolvimento de aplicações.

Na aplicação conectam-se os módulos sensoriais constituintes do sistema e recebem-se os dados dos acelerómetros, magnetómetros e giroscópios. O giroscópio de cada módulo sensorial é o único sensor que atualmente não está a ser usado. Os seus dados estão a ser adquiridos via SPI e são recebidos na aplicação, mas não estão

a ser usados como auxílio ao cálculo da postura. Para as medições dos segmentos corporais definidos, inicialmente apenas eram necessários os dados dos eixos do acelerómetro e magnético, sendo que o giroscópio iria contribuir para uma melhor precisão dos dados apresentados. Devido a uma questão de tempo optou-se então por implementar o sistema e colocá-lo apto para funcionar contendo apenas os dados do acelerómetro e magnetómetro, sem aplicar um algoritmo específico de compensação baseado no giroscópio.

Um dos grandes desafios desta fase de implementação foi a ambientação com a linguagem de programação Android e o conhecimento do seu paradigma. No entanto, o facto de esta ser baseada em Java ajudou imenso a compreensão e consequente desenvolvimento aplicacional. A par deste desafio, outra grande tarefa foi compreender o funcionamento dos sensores usados no sistema, bem como a interpretação dos dados para calibração e cálculo dos ângulos *Pitch*, *Roll* and *Yaw*, para determinar a orientação de cada módulo.

Posto isto, concluiu-se que o BLE é uma tecnologia promissora no ramo das redes de sensores sem fios, e com muitos argumentos a favor da sua proliferação nas redes de área corporal. Tal como outras tecnologias de baixo consumo e baixo débito, o BLE foi desenhado especialmente para enviar pequenas quantidades de informação em cada pacote, o que é apropriado para transmitir informação de diversos tipos de sensores. Além disso, possui uma topologia de rede escalável de modo a acomodar diversos módulos sensoriais. Com o desenvolvimento e atualização das *stacks* de cada equipamento, vai se verificar a tendência de se uma tecnologia concorrente às atuais existentes no mercado. O facto de o BLE estar integrado na maioria dos smartphones atuais faz com que a sua utilização se torne preferencial comparativamente, por exemplo, à tecnologia ZigBee, pois, embora esta possa ser integrada num dispositivo móvel recorrendo a um cartão SD, esta solução não é tão prática e estável como o uso de uma tecnologia que já venha embutida no próprio hardware do equipamento.

Como trabalho futuro propõe-se uma análise sobre todo o código desenvolvido de modo a melhorar certos aspetos, de forma a permitir a inclusão de novas

funcionalidades. No que diz respeito ao protocolo BLE, realizar mais testes experimentais de consumo de energia no equipamento usado neste projeto era uma mais-valia, para se ter noção da longevidade real de uma bateria em diversos sistemas baseados em BLE. A nível aplicacional, a melhoria da interface para com utilizador, a introdução de um algoritmo de compensação de erros baseado nos dados do giroscópio e a partilha dos resultados via web seriam certamente tarefas plausíveis de integrar no sistema até aqui desenvolvido.

Referências

[1] M. Patel, J. Wang, "Applications, challenges, and prospective in emerging body area networking technologies", IEEE Wireless Communications,2010, Vol. 17, No. 1, pp. 80-88.

[2] S. Saleem, S. Ullah, K. Kwak, “A study of IEEE 802.15.4 security framework for wireless body area networks”, *Sensors (Basel, Switzerland)*, 2011, *11*(2), 1383–95. doi:10.3390/s110201383.

[3] M. Ashe et al., "Body position affects performance in untrained cyclists", British Journal of Sports Medicine,2003, Vol. 37, No. 5, pp. 441-444.

[4] J. A. Afonso, J. H. Correia, H. R. Silva, L. A. Rocha, "Body Kinetics Monitoring System", International Patent WO/2008/018810A2 , February 2008.

[5] Nordic Semiconductor, “Bluetooth low energy wireless technology backgrounder”, March 2011.

[6] J. A. Afonso, “Integração de Sistemas de Comunicações”, Universidade do Minho, 2013.

[7] IEEE. (2010) 802.15 working group for WPAN. [Online]. Página disponível em: www.ieee802.org/15/

[8] IEEE Std 802.15.1, “Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)”, June 2005.

[9] ZigBee Alliance. (2014) [Online]. Página disponível em: www.zigbee.org.

[10] H. Karl, A. Willig, “Protocols and Architectures for Wireless Sensor Networks.” Wiley, 2005.

[11] H. Silva, “Protocolos de acesso ao meio em redes de sensores sem fio aplicadas

à captura de movimentos corporais", Dissertação de metrado em Engenharia de Comunicações, Universidade do Minho, 2010.

[12] IEEE. (2010). TG6 Body Area Networks (BAN) [Online]. Página disponível em: https://mentor.ieee.org/802.15/dcn/10/15-10-0245-06-0006-tg6-draft.doc.

[13] IrDA. (2011). Infrared Data Association [Online]. Página Disponível em: www.irdajp.info.

[14] Bluetooth SIG. (2009). Specification of Bluetooth System (Covered Core Package version: 3.0+HS) [Online]. Página disponível em: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[15] Bluetooth SIG. (2013). Specification of Bluetooth System (Covered Core Package version: 4.0) [Online]. Página disponível em: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[16] Bluetooth SIG. (2014). Special Interest Group [Online]. Página disponivel em: http://www.bluetooth.com/Pages/Bluetooth-Home.aspx.

[17] Texas Instruments, "CC2540 / 41 Bluetooth Low Energy Software Developer's Guide v1.3.2 (SWRU271F)", 2013.

[18] C. Gomez, J. Oller, J. Paradells, "Overview and evaluation of Bluetooth low energy: an emerging low-power wireless technology", Sensors (Basel, Switzerland), 2012, 12(9), 11734–53. Doi: 10.3390/s120911734.

[19] J. Decuir, "Bluetooth 4. 0 : Low Energy", Cambridge Silicon Radio (CSR), 2010.

[20] Android by Google. (2014). [Online]. Página disponível em: http://www.android.com.

[21] N. Gandhewar, R. Sheikh," Google Android: An Emerging Software Platform For Mobile Devices", International Journal on Computer Science and Engineering (IJCSE), 2010.

[22] Android Developers. (2014). [Online]. Página Disponível em: www. developer.android.com/develop/índex.html.

[23] IDC. (2014). International Data Corporation [Online]. Página Disponível em: http://www.idc.com/getdoc.jsp?containerId=prUS25037214.

[24] B. Fernandes, J. A. Afonso, R. Simões, "Vital Signs Monitoring and Management using Mobile Devices", Information Systems and Technologies (CISTI 2011),6th Iberian Conference, Chaves, Portugal, June 2011.

[25] H. Silva, P. Macedo, J. A. Afonso, "Design and Implementation of a Wireless Sensor Network applied to Motion Capture", 1st Portuguese Conference on Wireless Sensor Networks (CNRS 2011), University of Coimbra, March 2011.

[26] P. Macedo, J. A. Afonso, L. Rocha, R. Simões, "A Telerehabilitation System based on Wireless Motion Capture Sensors", International Conference on Physiological Computing Systems (PhyCS 2014), Lisbon, Portugal, 2014.

[27] J. A. Afonso, H. Silva, P. Macedo, L. Rocha, "An enhanced reservation-based MAC protocol for IEEE 802.15.4 networks", Sensors (Basel, Switzerland),2011, 11(4), 3852–73. doi:10.3390/s110403852.

[28] E. Coelho, P. Carvalhal, M. Ferreira, L. Silva, H. Almeida, C. Santos, J. A. Afonso, "A Bluetooth-based Wireless Distributed Data Acquisition and Control System", IEEE International Conference on Robotics and Biomimetics, 2006, 543–548. doi:10.1109/ROBIO.2006.340258.

[29] C. Gomez, J. Oller, J. Paradells, "Overview and evaluation of Bluetooth low energy: an emerging low-power wireless technology.", Sensors (Basel, Switzerland), 2012,12(9), 11734–53. Doi: 10.3390/s120911734.

[30] C. Gomez, I. Demirkol, J. Paradells, "Modeling the Maximum Throughput of Bluetooth Low Energy in an Error-Prone Link", IEEE Communications Letters, 2011, 15(11), 1187–1189.

[31] K. Mikhaylov, N. Plevritakis, J. Tervonen, "Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI", Journal of Sensor and Actuator Networks, 2013, 2(3), 589–613. doi:10.3390/jsan2030589.

[32] M. Siekkinen, M. Hiienkari, J. K. Nurminen,J. Nieminen, “How Low Energy is Bluetooth Low Energy ? Comparative Measurements with ZigBee/802.15. 4”, Wireless Communications and Networking Conference Workshops (WCNCW),2012, 232–237.

[33] Texas Instruments, “Measuring Bluetooth Low Energy Power Consumption”, Application Note ANO92, 2012.

[34] G. Delwing, M. Pasini, M. Torre, R. Fialho, F. Chaise, J. Loss, C. Candotti, “Modelo Para Quantificação Das Forças Musculares e articulares na coluna cervical durante o ciclismo”, Salão de Iniciação Cientifica, UFRGS, 2007.

[35] W. Peveler, J. Pouders, P. Bishop, “Effects of saddle height on anaerobic power production in cycling”, Journal of Strenght and Conditioning Research/ National Strenght and Conditioning Association, Colorado, 2007, v. 21, n. 4, p. 1023-1027.

[36] R. Marin-Perianu, M. Marin-Perianu, P. Havinga, S. Taylor, R. Begg, M. Palaniswami, D. Rouffet, “A performance analysis of a wireless body-area network monitoring system for professional cycling”, Personal and Ubiquitous Computing, 2011, 17(1), 197–209. doi:10.1007/s00779-011-0486-x.

[37] T. Alencar, K. Matias, “Importância da avaliação musculosquelética e biomecânica para o Bike Fit ”, Revista Movimenta,2009, Vol.2, N 3.

[38] TheBikeFit. (2013). [Online]. Página disponivel em: http://www.thebikefit.com/.

[39] Texas Instruments, “2.4-GHz Bluetooth ® low energy System-on-Chip”, June 2013.

[40] Texas Instruments, “Bluetooth Low Energy CC2540 Development Kit CC2541 Evaluation Module Kit - User’s Guide” (SWRU301A), 2013.

[41] Texas Instruments, “SmartRF05 Evaluation Board – User’s Guide” (SWRU210A), 2010.

[42] InvenSense, “MPU-6000 and MPU-6050 Product Specification Revision 3.2”, 1–

57, 2011.

[43] Honeywell,"3-Axis Digital Compass IC HMC5883L", 2010.

[44] IAR Systems. (2014). IAR Embedded Workbench for 8051 [Online]. Página disponível em: http://www.iar.com/ew8051/.

[45] S. Johnsrud, T. Sundet, "CC111xFx, CC243xFx, CC251xFx and CC253xFx SPI DN113 – swra223a", Texas Instruments, 2009.

[46] AChartEngine. (2013). Charting software library for Android applications [Online]. Página disponivel em: www.achartengine.org.

[47] J. A. Afonso, H. Silva, P. Oliveira, J. Correia, L. Rocha, "Design and implementation of a real-time wireless sensor network," Sensor Technologies and Applications, International Conference on, 2007,vol. 0, pp. 496–501.

[48] O. Gama, P. Carvalho, J. A. Afonso, P. Mendes, "An improved mac protocol with a reconfiguration scheme for wireless e-health systems requiring quality of service," Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE 2009), 2009.