Universidade do Minho
Escola de Engenharia

Rui Miguel de Cima Dias Pereira

Infraestrutura de Comunicações para Suporte a Ambientes Inteligentes de Apoio à Saúde

outubro de 2013

Infraestrutura de Comunicações para Suporte a Ambientes Inteligentes de Apoio à Saúde
Rui Miguel de Cima Dias Pereira
UMinho | 2013

Universidade do Minho
Escola de Engenharia

Rui Miguel de Cima Dias Pereira

Infraestrutura de Comunicações para Suporte a Ambientes Inteligentes de Apoio à Saúde

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

Trabalho realizado sob a orientação do
Professor Doutor Carlos Alberto Silva

outubro de 2013

DECLARAÇÃO

Nome: Rui Miguel de Cima Dias Pereira

Endereço Eletrónico: ruicdpereira@gmail.com

Título da Dissertação: Infraestrutura de Comunicações para Suporte a Ambientes Inteligentes de Apoio à Saúde

Orientador: Professor Doutor Carlos Alberto Silva

Ano de Conclusão: 2013

Designação do Mestrado: Mestrado Integrado em Engenharia Biomédica – Ramo de Eletrónica Médica

Universidade do Minho, 31 / 10 / 2013

Assinatura: _____

Dedicatória

Mano,

és o meu companheiro de vida,
és o leme quando estou à deriva,
és o Sol e eu a Lua,
és o guardião da minha rua.

Ao meu irmão

Agradecimentos

Ao meu orientador, Professor Doutor Carlos Alberto Silva, por poder contar com o seu incentivo e reconhecimento a cada momento. O apoio, a disponibilidade manifestada e a confiança depositada contribuíram decisivamente para que este trabalho tenha chegado a bom termo. Agradeço também o seu interesse em auxiliar o projeto e ajudar com ideias e incentivos sempre úteis.

Ao Professor Doutor Paulo Mateus Mendes, fica o agradecimento pela disponibilidade e aconselhamento. Os conhecimentos transmitidos foram sem sombra de dúvida, uma mais-valia e uma enorme ajuda para a realização desta dissertação.

Ao meu irmão, Ricardo Pereira, pela confiança que sempre depositou no meu trabalho. Obrigado "mano mais velho", pela paciência, compreensão e pela constante ajuda em todos estes anos.

À minha namorada, Joana Ferreira, ouvinte atenta de algumas dúvidas, inquietações, desânimos e sucessos, pelo apoio, pela confiança e pela valorização sempre tão entusiasta do meu trabalho, dando-me desta forma, coragem para ultrapassar a culpa pelo tempo que a cada dia lhe subtraía.

Um especial agradecimento aos meus pais, por sempre me incentivarem perante os desafios, a fazer mais e melhor, quero partilhar convosco a alegria de os conseguir vencer continuamente. Uma palavra de reconhecimento muito especial para eles, pelo amor incondicional e pela forma como ao longo de todos estes anos, tão bem, souberam ajudar-me.

Finalmente, a toda a família e amigos, um sincero obrigado pelo apoio e ajuda disponibilizados.

Obrigado a todos.

Resumo

Devido ao envelhecimento da população, ao avanço tecnológico e à pressão para reduzir os custos de saúde, foi necessário o desenvolvimento de novas tecnologias em saúde para a promoção de uma saúde pró-ativa e de atendimento aos idosos. A recente e rápida proliferação de dispositivos inteligentes em ambientes residenciais conduziu a uma fraca interoperabilidade entre os mesmos. Paralelamente, a inexistência de mecanismos de autoconfiguração e de *interfaces* intuitivas nos sistemas atualmente disponíveis obriga os seus utilizadores a possuírem conhecimento altamente especializado para construir e gerir uma casa inteligente. Com o intuito de facilitar o acesso do utilizador comum aos sistemas, desenvolveu-se uma plataforma de *middleware* orientada a serviços, que permite o acesso às funcionalidades oferecidas por qualquer dispositivo ou aplicação existente num ambiente residencial, de forma transparente, intuitiva e abstraída das tecnologias de comunicação envolvidas.

Assim, no âmbito desta dissertação foi desenvolvido uma infraestrutura de comunicações para suporte a ambientes inteligentes de apoio à saúde com três características fundamentais: flexibilidade, adaptabilidade e integridade. Na posse destas características esta infraestrutura torna-se vantajosa uma vez que minimiza a necessidade de intervenção de pessoal auxiliar, com consequente redução de custos de operação, potencia uma melhor dinâmica em termos de execução de tarefas realmente importantes e ainda faculta a possibilidade de uma expansão progressiva. A infraestrutura de comunicação suporta três protocolos de comunicação (Ethernet, Bluetooth e RS-232), permitindo a partilha remota e transparente de dados fisiológicos dos pacientes e de eventos/alarmes através do uso do padrão *publisher/ susbcriber*.

Por fim, faz-se uma avaliação de toda a infraestrutura com base em dois cenários de teste de dois pacientes com diferentes necessidades e discutem-se os resultados.

Palavras-Chave: Ambiente Inteligente*,* *Middleware*, *Publish/ Subscribe*, Bluetooth, Ethernet, RS-232, *Interfaces*.

Abstract

Due to the aging population, technological development and the pressure to reduce healthcare costs, new healthcare technologies for proactive health and elder care were needed. The recent rapid proliferation of smart devices in residential environments led to poor interoperability between them. In parallel, the lack of mechanisms for auto configuration and intuitive interfaces on currently available systems requires its users to possess highly specialized knowledge to build and manage a smart home. With the intention to facilitate the systems access to the common user, a platform for service-oriented middleware has been developed, which provides access to the functionality offered by any existing device or application in a residential setting, in a transparent and intuitive form, and abstracted from the technologies communications involved.

Thus, within this dissertation was developed a communication infrastructure for healthcare intelligent environments support with three fundamental features: flexibility, integrity and adaptability. These features make the infrastructure advantageous since it minimizes the need of auxiliary personnel intervention, with consequent reduction in operating costs, favoring a better dynamic in terms of actually implementing important tasks and even provides the possibility of a gradual expansion. The communication infrastructure provides three communication protocols (Ethernet, Bluetooth and RS-232) using the publisher/subscriber standard, enabling remote and transparent sharing of the patients physiological data and events/alarms.

Finally, a review of the entire infrastructure is made based on two test scenarios and two patients with different needs and the results are discussed.

Keywords: Smart Environment, Middleware, Publisher/Subscriber, Bluetooth, Ethernet, RS-232, Interfaces.

Índice

Índice de Figuras

Índice de Tabelas

Nomenclatura

Acrónimos

AAL	Ambient Assisted Living
A/D	Conversor Analógico-Digital
AES	Advanced Encryption Standard
AMON	Advanced Telemedical Monitor
ASI	Adaptive Systems and Interaction
CPU	Unidade Central de Processamento
DS	Domain Server
ECG	Eletrocardiograma
EMG	Eletromiografia
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GSR	Galvanic Skin Response
IST	Information Society Technology
L2CAP	Logical Link Control and Adaptation
LED	Light Emitting Diode
LMP	Link Manager Protocol
LS	Local Server
MIT	Massachusetts Institute of Technology
NDA	Non-Disclosure Document
P2P	Peer to Peer
PDA	Personal Digital Assistant
PIB	Produto Interno Bruto
PPP	Point-to-Point Protocol
RF	Rádio Frequência
SDP	Service Discover Protocol
SPP	Serial Port Profile
TCP/IP	Transmission Control Protocol/Internet Protocol
TCS	Telephony Control protocol

UART	Universal Asynchronous Receiver-Transmitter
UE	União Europeia
UMTS	Universal Mobile Telecommunication System
USB	Universal Serial Bus
WAP	Wireless Application Protocol

Capítulo 1 - Introdução

O presente capítulo introduz os conceitos teóricos associados à integração de ambientes inteligentes na saúde. São ainda indicados a motivação e os objetivos da dissertação bem como a contribuição deste trabalho. Apresenta-se, no final do capítulo, a estrutura deste documento.

1.1 Soluções e-Health

Os avanços tecnológicos na área da saúde têm permitido que as pessoas vivam uma vida mais longa. As estatísticas demográficas mostram que a proporção dos idosos aumentou nas últimas décadas e a longevidade também está a aumentar continuamente [1]. Prolongar a vida é uma conquista importante, no entanto, o aumento da expetativa de vida está a tornar cada vez mais velha a população a nível mundial [2]. Por sua vez, este envelhecimento tem como consequência o aumento do número de idosos que necessitam de apoio e de acompanhamento quotidiano.

Do ponto de vista social, é importante que todas as pessoas que têm a necessidade de serem apoiadas no seu dia-a-dia permaneçam, sempre que possível, integradas no seu grupo familiar, independentemente da sua idade. Contudo, o envelhecimento tem implicações económicas enormes (sistema de saúde, por exemplo) [1]. Como é possível observar na Figura 1.1, ao longo dos últimos anos tem-se verificado um aumento generalizado da despesa do Estado em saúde, tendo até ultrapassado largamente o ritmo de crescimento do Produto Interno Bruto (PIB). Esta situação cria uma constante preocupação relativa à sustentabilidade do sistema de saúde atual em Portugal.

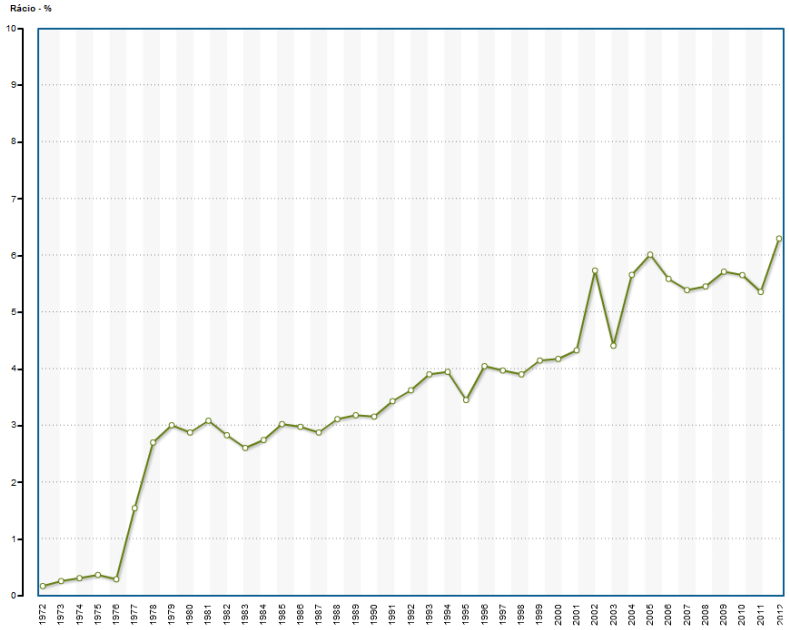

Figura 1.1 – Despesas do Estado em saúde em % do PIB, de 1972 a 2012
Fonte: [3]

Muito embora as despesas em saúde em Portugal sejam relativamente baixas, quando comparadas com outros países da União Europeia, não deixam de constituir um problema preocupante para a sustentabilidade do Serviço Nacional de Saúde. A resolução deste problema acarreta grandes desafios para a nossa sociedade, o que motiva a procura de soluções eficazes e de baixo custo [2].

Face a esta realidade, é necessário uma mudança de paradigma, substituindo-se o modelo de sistemas de saúde centralizados e reativos, centrados sobretudo na doença, para modelos de sistemas distribuídos (Figura 1.2) e pró-ativos focados essencialmente na gestão de bem-estar. Neste contexto, o controlo distribuído e contínuo da saúde constitui uma tecnologia chave na transição para sistemas de cuidados de saúde mais pró-ativos e acessíveis. Estes permitem que um indivíduo possa acompanhar de perto as alterações nos seus sinais vitais e forneçam *feedback* para o ajudar a manter um estado de saúde desejável. Além disso, quando integrados num sistema *e-Health*, estes permitem ainda alertar o pessoal médico sempre que ocorram mudanças que requeiram intervenção [4] [5].

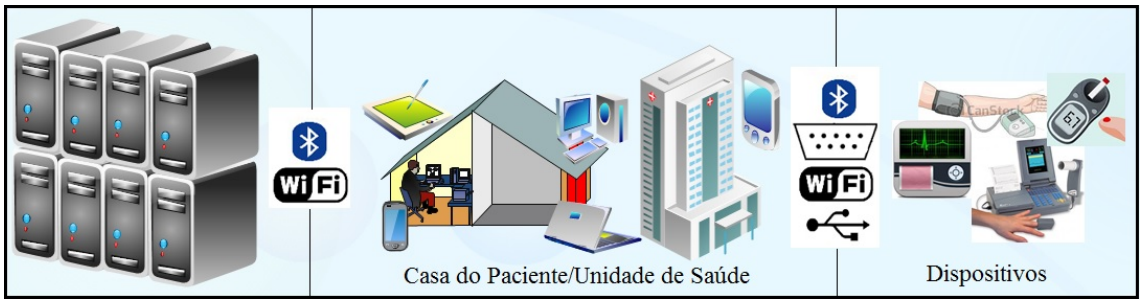

Figura 1.2 – Cenário de um sistema distribuído

Em termos gerais, um sistema distribuído é constituído por um grande número de dispositivos, que requerem a satisfação de alguns requisitos [6] [7]:

i. Interoperabilidade: é fundamental, uma vez que se pretende interligar um grande número de dispositivos com diferentes âmbitos de aplicação, tipos e marcas. Além disso, é desejável que ao longo do tempo seja possível integrar novos dispositivos, de diferentes fabricantes, com novas funcionalidades.

ii. Facilidade de Gestão: ponto-chave do sucesso deste tipo de sistemas, pois destinam-se a utilizadores comuns, com poucos ou nenhuns conhecimentos tecnológicos. Estes utilizadores só aderem, facilmente, a sistemas que possam ser geridos de forma intuitiva e segura.

iii. Nível funcional: a grande necessidade atual é a de criar soluções mais eficientes, que permitam maximizar a adaptação dos sistemas ao estilo de vida de cada utilizador.

iv. Segurança: caso a transmissão seja feita sem qualquer tipo de encriptação, é possível que uma terceira pessoa aceda mais facilmente à rede, nomeadamente ter acesso aos dados privados ou, hostilmente, alterar o funcionamento correto da rede.

Num sistema distribuído, cada processo tem responsabilidades bem definidas e interage com os outros para atingir uma finalidade comum. Um sistema distribuído pode ser organizado de várias maneiras, sendo as mais comuns a *Publisher/Subscriber* e a *Peer* to *Peer* (P2P) [6] [7].

Na arquitetura *Publisher/Subscriber*, os processos são divididos em dois grupos: *publishers* que implementam serviços específicos e *subscribers* que requisitam serviços aos

servidores por meio de requisições, aguardando uma resposta em seguida. Os *publishers* também podem ser *susbcribers* de outros *publishers*. A comunicação entre *publisher* e *subscriber* pode ser feita de maneira confiável, utilizando-se protocolos oferecidos pela rede (Ethernet, Bluetooth). A arquitetura *Publisher/Subscriber* facilita a centralização das ocorrências de eventos num único ponto e, consequentemente, facilita a composição de eventos [6] [7].

Figura 1.3 – Arquitetura *Publisher/Subscriber*

No caso da arquitetura P2P, os processos cooperam entre si para realizar uma tarefa em comum, interagindo como pares. Ao contrário da arquitetura Cliente/Servidor, neste padrão arquitetural não existe um elemento da estrutura responsável por organizar e distribuir recursos/atividades. Neste tipo de arquitetura são encontradas algumas dificuldades como, por exemplo, a instabilidade na ligação entre dois pontos. Esta dificuldade aumenta em sistemas distribuídos, pois, a entrada e saída na rede faz com que as ligações estejam constantemente em mudança [6] [7].

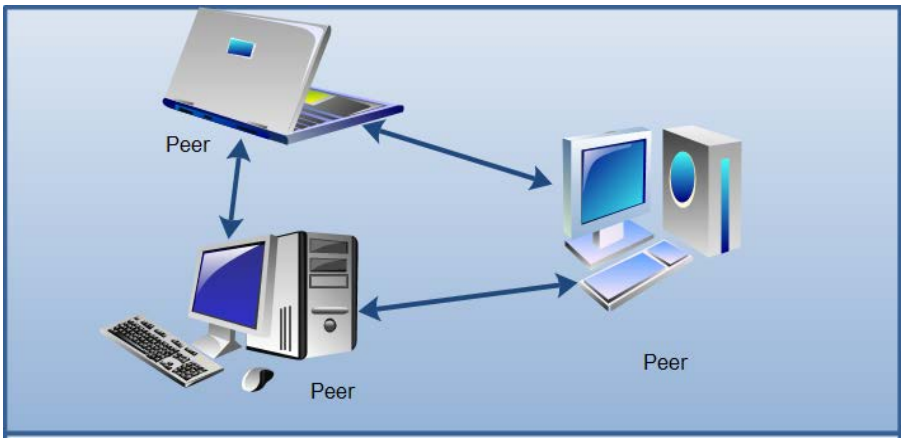

Figura 1.4 – Arquitetura P2P

Os avanços tecnológicos da eletrónica atual, particularmente ao nível da miniaturização e do baixo consumo, têm tornado possível desenvolver sistemas distribuídos capazes de longos

períodos de monitorização, fornecendo até monitorizações praticamente contínuas do estado de saúde dos pacientes. Ora, a monitorização contínua é muito útil para a medicina de prevenção, uma vez que fornece alertas para potenciais riscos de saúde do paciente, viabilizando uma intervenção médica mais rápida [8].

Assim sendo, a integração de novas tecnologias em sistemas *e-Health* na saúde tem revelado diversos e inquestionáveis benefícios nos sistemas de saúde europeus [9]:

i. Em primeiro lugar, minimiza os custos das atividades de saúde e melhora os processos. A necessidade de viagens quer dos pacientes quer dos profissionais de saúde é reduzida, e, consequentemente, reduzem-se os custos para os pacientes e para o sistema de saúde.

ii. Em segundo lugar, reduz o tempo necessário para executar tarefas e processos de saúde. A introdução da tecnologia reduz o tempo que os profissionais de saúde gastam em tarefas de tratamento de dados. Além disso, a maior eficiência de manuseio de dados incita a uma maior rapidez na tomada de decisão e na ação, o que pode salvar muitas vidas.

iii. Em terceiro lugar, o profissional de saúde pode lidar com um número maior de pacientes sem custos associados.

iv. Por último, a integração de soluções tecnológicas nas casas dos pacientes, que verifiquem continuamente o estado de saúde dos doentes crónicos e dos idosos, permite-lhes viver de forma autónoma e durante mais tempo nas suas casas. Esta *Ambient Assisted Living* (AAL) é vista como um protetor para os pacientes mais débeis uma vez que permite enviar um sinal de aviso ao prestador dos cuidados de saúde [10].

Assim, a integração das tecnologias de comunicação - Bluetooth, Ehternet e RS-232 - na saúde, em ambiente privado em rede com os profissionais de saúde, possibilita a coordenação dos esforços de modo a otimizar os interesses de todos os envolvidos. Ninguém pode lembrar-se de todos os medicamentos, todas as doenças ou até mesmo alergias, especialmente numa situação de emergência. A existência de uma infraestrutura de comunicações que reúna todos os registos médicos, computadorizados numa base de dados, de modo a fornecer todas as

informações necessárias aos profissionais de saúde, constitui uma excelente resposta na assistência à saúde e na monitorização das atividades diárias dos idosos e dos doentes crónicos nas suas próprias casas [9] [11].

1.2 Motivação e Objetivos

A infraestrutura de comunicações proposta nesta dissertação, vem de encontro à integração de um sistema distribuído na saúde. Esta infraestrutura, sendo um sistema inteligente de monitorização, resulta da necessidade sentida de desenvolver uma solução tecnológica que permita acompanhar os sinais vitais dos pacientes num contexto institucional, público ou privado, por um número alargado de utilizadores. A sua vertente automática e digital permite, por um lado, eliminar subjetividades inerentes ao paciente, uma vez que nos registos manuais podem ocorrer, por exemplo, erros na medição ou registo da data e hora e, por outro lado, resolver problemas a nível social e económico, gerados pelo envelhecimento da população.

Além disso, esta infraestrutura assegura também o controlo de todo o processo clínico, como o envio de alertas nos casos em que os parâmetros ultrapassem os limiares previamente estabelecidos pelos profissionais de saúde. Refira-se ainda que existe sempre a possibilidade de integração de novos módulos para recolha de outros sinais biológicos relevantes, permitindo assim a criação de uma rede flexível e adaptada a diferentes contextos.

Este projeto tem por objetivo desenvolver uma infraestrutura de comunicação que permita a integração de um conjunto de tecnologias já existentes, que possibilite a recolha e processamento de informação relacionada com os dados vitais dos seus utilizadores.

1.3 Contribuições

Esta infraestrutura de comunicações permite que os pacientes, sobretudo os idosos e os doentes crónicos, sejam monitorizados continuamente, tanto em casa como no ambiente hospitalar.

A maioria dos sistemas descritos no Capítulo 2 - Estado da Arte - demonstram um funcionamento localizado, dificultando a sua recolha remota. Dada a natureza do mercado, com muitos fabricantes, e a inexistências de normas as soluções atuais são em geral

independentes, e quando existem soluções distribuídas estas não permitem a integração de novos dispositivos que não foram pensados para serem interligados.

Para aumentar a adaptabilidade deste tipo de sistemas, é necessário ter em conta as tecnologias de comunicação existentes para os diferentes tipos de sensores e ainda os diversos ambientes em que podem ser integrados. Sendo assim, esta nova infraestrutura integra de forma transparente diversos tipos de comunicação - Bluetooth, Ethernet, WiFi, RS-232 -, que possibilita ajustar a plataforma ao ambiente e adequar-se a cada utilizador.

Esta infraestrutura de comunicações apresenta algumas características importantes:

i. É possível a expansão progressiva (integração de novos módulos) para aquisição de outros sinais biológicos relevantes.
ii. Passa a existir uma uniformização do sistema de leitura, apesar de cada um dos seus sub-sistemas poder variar de *Box* para *Box*, permitindo uma fácil criação de redes de registo e monitorização do estado de saúde de um conjunto de indivíduos;

1.4 Estrutura da Dissertação

O presente documento encontra-se dividido em seis capítulos. O primeiro capítulo contextualiza o estudo, indica os objetivos da dissertação e mostra a importância desta infraestrutura de comunicações. No segundo capítulo faz-se uma breve descrição teórica acerca das diversas infraestruturas existentes para a criação de ambientes inteligentes na saúde (Estado da Arte). No terceiro capítulo é apresentada uma pequena descrição teórica acerca dos conceitos relacionados com as tecnologias de comunicação utilizadas. O capítulo quatro descreve os detalhes relacionados com a implementação da infraestrutura. No capítulo cinco expõem-se os testes experimentais realizados e faz-se a discussão dos resultados obtidos. Finalmente, no capítulo seis apresentam-se as principais conclusões do trabalho realizado e apontam-se as perspetivas de trabalho futuro.

Capítulo 2 – Estado da Arte

Neste capítulo, faz-se um levantamento dos sistemas utilizados em *e-Health*. Na primeira secção, são apresentados diversos dispositivos médicos comerciais existentes no mercado que servem para monitorizar sinais fisiológicos; a seguir destacam-se os diferentes projetos de investigação e as suas ideias centrais.

2.1 Dipositivos Médicos Comerciais

Os seguintes sistemas para monitorizar dados fisiológicos estão entre os primeiros sistemas de AAL. Todavia, os aparelhos existentes são limitados pois adquirem os dados e disponibilizam-nos apenas ao próprio utilizador.

2.1.1 CardioNet

O sistema CardioNet permite monitorar o sinal cardíaco por telemetria. O sistema inclui um sensor, um aparelho de monitorização de saúde portátil, em forma de Personal Digital Assistant (PDA) ou telemóvel, e um serviço centralizado que arquiva os dados de vários utilizadores. Os elétrodos são colocados no peito e o dispositivo de aquisição é colocado em volta do pescoço ou no cinto. O sistema comunica por *wireless* com o monitor e depois o monitor faz o *upload* dos dados para o serviço central, recorrendo às infraestruturas de comunicação existentes. O utilizador pode introduzir os sintomas de modo a que o serviço centralizado possa correlacionar os dados. No entanto, o aparelho de monitorização também realiza algum processamento de sinal e é capaz de sinalizar anormalidades no ritmo cardíaco [12].

Figura 2.1 – CardioLab
Fonte: [12]

2.1.2 Polar Electro

A Polar Electro tem uma extensa linha de aplicações para monitorar o ritmo cardíaco, condição física, perda de peso e destina-se sobretudo a treinadores pessoais de atletas. O sistema típico inclui uma cinta torácica com elétrodos de electrocardiograma (ECG), um adaptador Bluetooth e um relógio ou *smartphone* como recetor sem fios. O recetor recebe informação do ritmo cardíaco e tem funções de cálculo de médias. A Polar Electro, para além da aplicação desportiva também é comercializada para monitorização do ritmo cardíaco de pacientes em reabilitação cardíaca. Apesar de ser importante para um doente em recuperação de um ataque cardíaco, o sistema é limitado, pois não está integrado num sistema centralizado nem permite a partilha dos dados com o médico. [13].

Figura 2.2 – Polar Electro
Fonte: [13]

2.1.3 BodyMedia

A BodyMedia oferece aparelhos portáveis para monitorização de dados fisiológicos. Estes aparelhos têm uma pequena seleção de bandas de braço com um acelerómetro de eixo simples para gravar o movimento corporal, dois sensores de temperatura para deteção da temperatura corporal e um sensor galvânico para medir alterações na condutividade da pele. A banda de braço armazena os dados localmente e tem capacidade para monitorizar os dados fisiológicos durante sete dias continuamente. Faz-se posteriormente o *upload* dos dados para o servidor *web* da BodyMedia. Os portais *web* permitem aos utilizadores o acesso aos dados de pesos e calorias despendidas e são utilizados em duas aplicações primárias: programas de investigação clínica e controlo do peso [14].

Figura 2.3 – BodyMedia
Fonte: [14]

2.2 Projetos de Investigação

As soluções com maior grau de semelhança com este trabalho surgem de alguns projetos de investigação científica nesta área. Seguidamente, faz-se uma breve apresentação de alguns desses projetos relacionados com este trabalho.

2.2.1 Codeblue

O projeto Codeblue (Figura 2.4) visa oferecer uma infraestrutura de comunicação sem fios para vários cenários de saúde, com aplicação num ambiente hospitalar. Esta tecnologia, desenvolvida na Universidade de Harvard, tem como principais funcionalidades a formação de redes *ad hoc*, o roteamento, o controlo de segurança e de autenticação bem como a filtragem e agregação dos dados dos sinais vitais. Os pacientes são equipados com sensores sem fios que adquirirem os sinais vitais e, em seguida, através da filtragem dos dados, são identificados os pacientes mais necessitados de atenção médica. O sistema inclui sensores de oximetria de pulso, sensores de ECG e sensores de movimento baseados num acelerómetro triaxial de vigilância médica. Esta tecnologia usa placas desenvolvidas e o sistema operativo TinyOS. A aplicação de triagem em tempo real e o *interface* de utilizador encontram-se num PDA [15] [16].

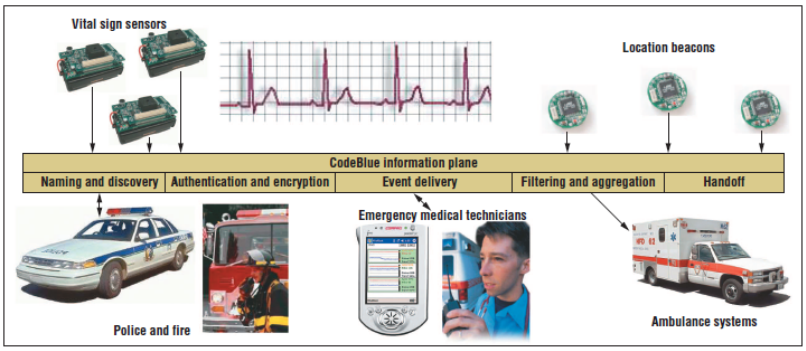

Figura 2.4 – Infraestrutura CodeBlue
Fonte: [16]

Refira-se ainda que a *10Blade* é uma pesquisa piloto e trabalha em colaboração com o Projeto Codeblue. Esta tecnologia promove três produtos que são componentes do projeto Codeblue: iRevive, iIncise e o sensor VitalDust. O iRevive representa a base de dados do paciente móvel e o iIncise a aplicação do PDA do utilizador [17].

2.2.2 MIThril

A plataforma MIThril (Figura 2.5) foi desenvolvida por investigadores do Massachusetts Institute of Technology **(**MIT). Esta tecnologia utiliza sensores portáteis para monitorizar o estado fisiológico de um utilizador e do ambiente circundante, a fim de descobrir novas técnicas para interação humano/computador. O sistema MIThril é centrado no "MIThril Real-Time Context Engine", cujos sensores, colocados no corpo, extraem as características relevantes e usa esses dados para monitorizar o utilizador. Os investigadores da MIThril usam, na sua rede, tanto sensores desenvolvidos como sensores comerciais. Em 2000, a característica definidora do sistema MIThril foi a integração de todos os sensores de forma modular e distribuída em roupas. Esta integração permitiu colocar os sensores onde eles eram mais úteis e adequados. Em 2003, foi acrescentado um PDA, de baixo custo sem fios, que permitiu redefinir o sistema MIThril como uma tecnologia sem fios, facultando-lhe uma melhor interação. O sistema MIThril inclui ECG, temperatura da pele, Galvanic Skin Response (GSR), acelerómetros de três eixos, giroscópios, sensores de pressão, câmaras e microfones [18] [19].

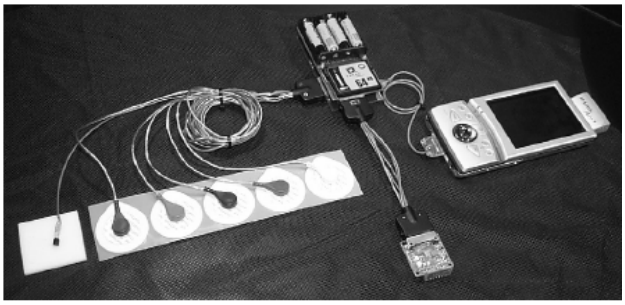

Figura 2.5 – Sistema MIThril
Fonte: [18]

2.2.3 HealthGear

Investigadores Adaptive Systems and Interaction (ASI), do grupo Microsoft, têm desenvolvido *o HealthGear* (Figura 2.6). Este é um sistema de rede de sensores sem fios, conectados via *Bluetooth* a um telemóvel, que armazena, transmite e analisa os dados fisiológicos e apresenta-os ao utilizador. A investigação tem sido usada sobretudo para o reconhecimento de padrões de comportamento dos indivíduos face a fatores externos, tais como: carga de trabalho, *stress*, exercício físico, dieta, sono, etc. [20].

Figura 2.6 – *Hardware* do *HealthGear*
Fonte: [20]

2.2.4 Advanced Telemedical Monitor

O projeto Advanced Telemedical Monitor (AMON) é um produto resultante da colaboração de empresas e de universidades europeias, financiado pela Information Society Technology (IST) da União Europeia (UE). O sistema (Figura 2.7) integra uma série de biossensores num dispositivo de monitorização de pulso, tais como: frequência cardíaca, ritmo cardíaco, ECG, pressão arterial, oximetria, respiração da pele e temperatura corporal. O sistema não só monitoriza os sinais vitais a partir de cada sensor, mas também faz uma análise preliminar para determinar, por exemplo, a frequência cardíaca e ainda, através de redes GSM/UMTS, transmite os dados para um sistema de telemedicina remoto para posterior análise e, se necessário, possível atendimento de emergência [21].

Figura 2.7 - Protótipo do AMON
Fonte: [21]

2.2.5 AlarmNet

Desenvolvido na Universidade de Virgínia, o AlarmNet (Figura 2.8) é uma rede de monitorização e de assistência a cuidados de saúde de residentes ou pacientes com necessidades diversas. Esse sistema unifica e acomoda dispositivos heterogéneos numa arquitetura comum. Integra sensores sem fios para dados fisiológicos e para dados do ambiente, *interfaces* com o utilizador, elementos de processamento e armazenamento de dados. Além disso, permite a integração de novos sensores quando elas se tornarem disponíveis [22].

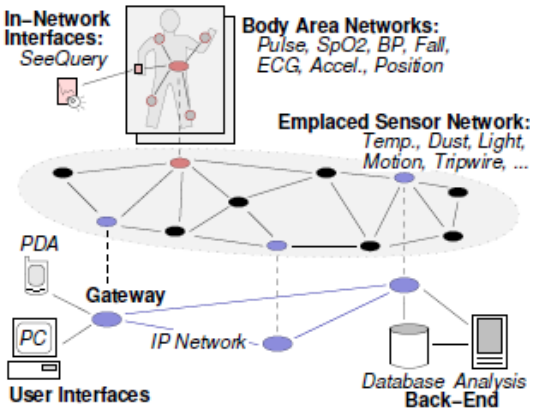

Figura 2.8 – Arquitectura do AlarmNet
Fonte: [22]

2.2.6 MobiHealth

O projeto MobiHealth (Figura 2.9), apoiado pela UE, foi desenvolvido como uma plataforma de serviços móveis de saúde para pacientes e profissionais da área. O sistema fornece uma plataforma completa de monitorização móvel de pacientes, através de redes Universal Mobile Telecommunication System (UMTS) e General Packet Radio Service (GPRS). Isso é possível equipando o paciente com diferentes sensores que monitorizam os seus sinais

vitais de forma contínua e que estão conectados entre si através de uma rede corporal de sensores sem fios (utilizando tecnologias como o Bluetooth ou ZigBee). Os dados recolhidos pelos sensores são concentrados num ponto central da rede. Esse ponto central reúne as informações da rede corporal e transmite-as, através das redes UMTS ou GPRS, para o sistema central remoto do projeto MobiHealth, onde uma equipa médica responsável faz a monitorização [23].

Figura 2.9 – Sistema MobiHealth
Fonte: [23]

2.2.7 OneCare

O OneCare, desenvolvido pela ISA Intellicare, consiste numa pequena consola que comunica através de uma rede sem fios, permitindo ao profissional de saúde acompanhar o estado de saúde do paciente (tensão arterial, peso e/ou glicemia, por exemplo) a partir de casa. Por sua vez, os dados das medições são disponibilizados aos profissionais de saúde através de um portal *web* [24] [25].

Figura 2.10 – OneCare
Fonte [24]

2.3 Sumário

Alguns sistemas tecnológicos, disponíveis no mercado, de monitorização de dados fisiológicos (CardioNet e Polar Eletro, por exemplo) apresentam um conjunto de limitações. No entanto, existem diversos projetos de investigação que permitiram desenvolver sistemas *e-Health,* com base em redes, com características análogas a este trabalho.

A Tabela 2.1 esquematiza os sistemas *e-Health* já desenvolvidos, referindo ainda o tipo de tecnologia de comunicação, o ambiente de inserção, a existência ou não de alertas, de processamento e de heterogeneidade de cada um deles.

Tabela 2.1 – Resumo dos sistemas e-Heatlh.

Sistema	GUI Design	Tecnologias	Ambiente	Processamento	Alertas	Heterogeneidade
CodeBlue	SIM	*Wireless*	Hospitalar	SIM	SIM	SIM
MIThtil	SIM	*Wireless*	Domiciliar	NÃO	NÃO	NÃO
HealthGear	SIM	Bluetooth	Domiciliar	SIM	NÃO	NÃO
AMON	NÃO	GSM/UMTS	Domiciliar	SIM	SIM	NÃO
AlarmNet	SIM	*Wireless*	Domiciliar	SIM	NÃO	SIM
MobiHealth	SIM	GPRS/UMTS Bluetooth/Zigbee	Domiciliar	NÃO	NÃO	NÃO
OneCare	SIM	Bluetooth/Ethernet	Domiciliar	NÃO	NÃO	NÃO

Capítulo 3 – Tecnologias de Comunicação

Neste capítulo, vamos apresentar diferentes tecnologias de comunicação que podem ser usadas num sistema distribuído para monitorização de pacientes. Nesta parte, fornece-se uma visão geral de três tecnologias de comunicação candidatas para a comunicação em sistemas distribuídos. Em primeiro lugar, apresenta-se a comunicação Transmission Control Protocol/Internet Protocol (TCP/IP) com uma visão geral de *WiFi*, em seguida, apresenta-se o Bluetooth e, por último, a comunicação RS-232.

3.1 TCP/IP

O Transmission Control Protocol/Internet Protocol (TCP/IP) refere-se ao conjunto dos dois protocolos de Internet que deram origem ao nome - Transmission Control Protocol (TCP) e Internet Protocol (IP). Esta é a tecnologia de base para a internet e é utilizada para a comunicação entre qualquer interligação de redes. [26] [27].

A comunicação TCP/IP é modelada em camadas. A divisão do protocolo em camadas permite uma maior facilidade na implementação, na validação dos códigos e na capacidade de desenvolver e implementar camadas alternativas. Cada camada comunica com as camadas que se encontram acima ou abaixo, através de *interfaces* concisas. Neste sentido, uma camada fornece um serviço para a camada imediatamente acima dele e faz uso dos serviços fornecidos pela camada imediatamente abaixo dela. Por exemplo, a camada IP fornece a capacidade de transferir dados a partir de um hospedeiro para outro, sem qualquer garantia de entrega. No entanto, a camada TCP faz uso deste serviço para fornecer uma aplicação confiável, ou seja, a entrega dos dados [26] [27].

A Figura 3.1 mostra que o protocolo TCP/IP é dividido em quatro camadas independentes [26] [27] [28]:

- Camada Física: inclui a *interface* física entre o dispositivo de transmissão de dados (por exemplo, estação de trabalho, computador) e um meio de transmissão ou de rede. Esta camada está relacionada com a especificação das características do meio de transmissão, a natureza dos sinais, a taxa de dados e

assuntos relacionados.

- Camada *Network*: refere-se à troca de dados entre um sistema e a rede à qual está ligado. O dispositivo transmissor deve fornecer à rede o endereço do dispositivo de destino, de modo que a rede possa encaminhar os dados para o destino apropriado. O IP, o protocolo mais importante nesta camada, fornece uma função de roteamento que permite a transmissão de dados entre múltiplas redes.

- Camada de Transporte: independentemente da natureza das aplicações que estão na troca de dados, geralmente há uma exigência de que os dados sejam trocados com fiabilidade. Isto é, os utilizadores querem ter a certeza de que todos os dados chegam ao destino pedido e chegam na mesma ordem em que foram enviados. Sendo assim, esta camada garante a confiabilidade da troca de dados e ainda fornece o controlo do congestionamento e do fluxo dos dados. O TCP é o protocolo mais comum para fornecer esta funcionalidade.

- Camada de Aplicação: contém a lógica necessária para suportar as diferentes aplicações do utilizador. A *interface* entre a aplicação e a camada de transporte é definido pelo número da porta e pelo *socket*.

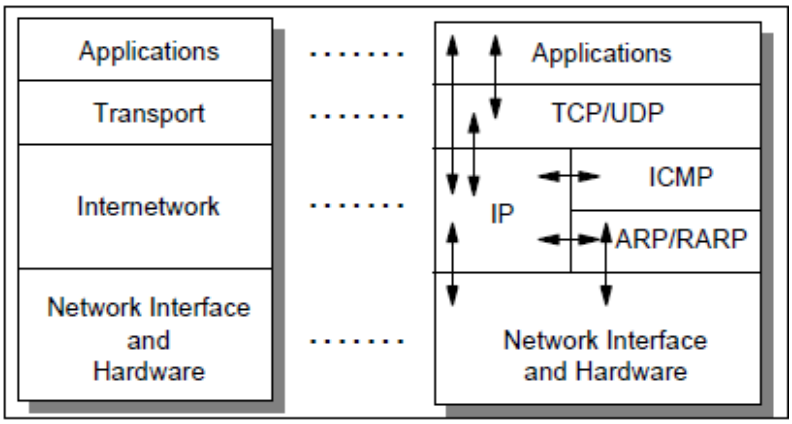

Figura 3.1 – Camadas do protocolo TCP/IP
Fonte: [27]

A Figura 3.2 indica como estes protocolos são configurados para comunicar. Alguns tipos de protocolos de acesso à rede são usados para conectar um computador a uma sub-rede. Este protocolo permite que o *host* envie os dados através da sub-rede para outra máquina, ou, no caso do *host* se encontrar noutra sub-rede, envie os dados a um *router*. O IP é implementado

em todos os sistemas de roteamento. Este atua como um retransmissor para mover um bloco de dados de um *host*, por meio de um ou mais *routers*, para outro *host*. O TCP é implementado para efetuar o controlo dos blocos de dados para garantir que todos sejam entregues de forma confiável [27] [28].

Para uma comunicação bem-sucedida, cada entidade no sistema global deve ter um endereço único. Na realidade, são necessários dois níveis de endereçamento. Cada *host* numa sub-rede deve ter um endereço global único, o que permite que os dados sejam entregues ao *host* adequado. Esse endereço é usado pelo IP para roteamento e entrega. Cada aplicação dentro de um *host* deve ter um endereço que é único dentro deste, o que permite o protocolo *host-to-host* (TCP) para fornecer dados para o processo adequado. Estes últimos endereços são conhecidos como portas [26] [28].

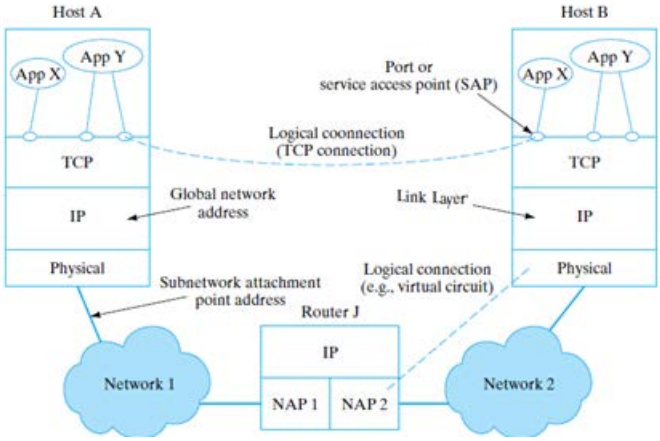

Figura 3.2 - Conceitos TCP/IP
Fonte: [28]

A interação das camadas entre dois dispositivos está ilustrada na Figura 3.3. Como se observa na Figura 3.3, a aplicação do dispositivo A deseja enviar uma mensagem ao dispositivo B. A aplicação A passa a mensagem até à camada de transporte com instruções para enviá-la para o *host* B. A camada de transporte entrega a mensagem à camada *network*. Note-se que esta camada não necessita da porta de destino. Tudo o que necessita de saber é o destinatário da mensagem. Em seguida, a camada *network* transmite os dados para a camada física - *Ethernet,* por exemplo - com instruções para enviar os dados ao *router* (a primeira passagem para o *host* B) [26] [28].

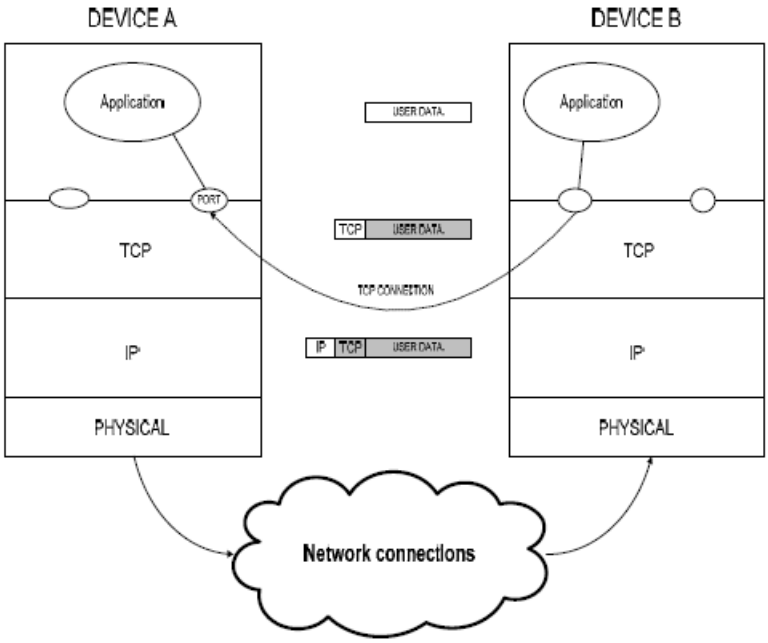

Figura 3.3 – Interação das camadas TCP/IP
Fonte: [26]

Cada camada acrescenta um cabeçalho com informação relevante. A informação relevante contida nos cabeçalhos é como se segue:

- Cabeçalho de transporte: Porta de destino, número de ordem e soma de verificação;
- Cabeçalho *network*: sub-endereço de rede do destino, instalações de pedidos.

No *router*, o cabeçalho de transporte é retirado e examinado o cabeçalho *network*. Com base na informação do endereço de destino a camada *network* do *router* encaminha os dados para o dispositivo B. Quando os dados são recebidos no B, ocorre o processo inverso. Em cada camada, o cabeçalho correspondente é removido e o restante é transferido para próxima camada, até que os dados originais do utilizador são entregues para a aplicação de destino [26] [28].

3.2 Comunicação Bluetooth

O Bluetooth é um padrão de comunicação sem fios de curto alcance, baixo custo e baixo consumo que utiliza tecnologia rádio. Usa a banda de 2,4 GHz, que é globalmente disponível para utilização sem licença e de baixa potência. O Bluetooth é destinado a prestar apoio a inúmeros dispositivos *wireless* entre diferentes perfis, que incluem dados, áudio, gráficos de vídeo, etc. [29].

Os aparelhos Bluetooth estão classificados em três diferentes classes dependendo da potência máxima que estão autorizados a transmitir e que também determinam a faixa máxima de transmissão rádio do Bluetooth, como demonstra a Tabela 3.1 [29].

Tabela 3.1 – Classes do Bluetooth

Classe	Máxima Potência	Alcance Rádio
Classe 1	100mW (20dDm)	~100m
Classe 2	2,5mW (4dBm)	~10m
Classe 3	1mW (0dBm)	~1m

A arquitetura do Bluetooth oferece serviços e funcionalidades básicas que tornam possível a conexão de dispositivos e a troca de dados entre estes dispositivos [29] [30] [31].

A tecnologia Bluetooth está dividida em três camadas lógicas: protocolos de transportes, protocolos *middleware* e a aplicação (Figura 3.4).

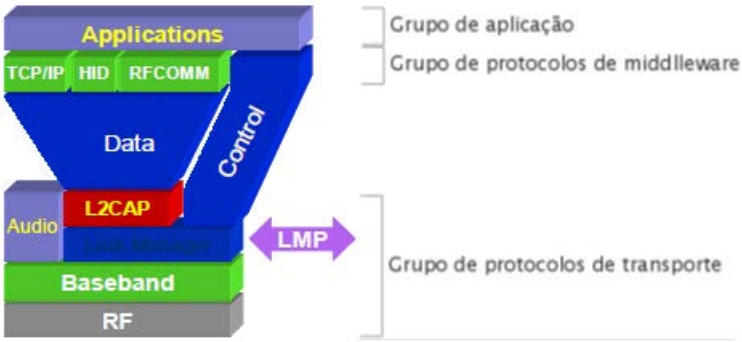

Figura 3.4 – Camadas do protocolo Bluetooth
Adaptado de: [30]

O grupo de protocolos de transporte permite aos dispositivos Bluetooth localizar outros dispositivos e gerir ligações físicas e lógicas para as camadas superiores. Neste contexto, protocolos de transporte não se equivalem aos protocolos da camada de transporte do modelo OSI (utilizado na especificação de protocolos de rede). Ao invés disso, estes protocolos correspondem às camadas físicas e de ligação lógica do modelo OSI. As camadas de rádio frequência (RF), Baseband, Link Manager, Logical Link Control and Adaptation (L2CAP) estão incluídas no grupo de protocolos de transporte. Estes protocolos suportam tanto comunicação síncrona como assíncrona e todos estes são indispensáveis para a comunicação entre dispositivos Bluetooth. O grupo de protocolos de *middleware* incluem protocolos de terceiros e padrões industriais. Estes protocolos permitem que aplicações já existentes e novas aplicações

operem sobre ligações Bluetooth. Protocolos de padrões industriais incluem Point-to-Point Protocol (PPP), TCP/IP, Wireless Application Protocol (WAP), etc.. Outros protocolos desenvolvidos, como o RFComm, permitem aplicações complexas que operaram sobre protocolos de transporte Bluetooth, protocolo de sinalização e controlo de telefonia baseada em pacotes (TCS), para a gestão de operações de telefonia e o Service Discover Protocol (SDP) que permite aos dispositivos obterem informações sobre serviços disponíveis de outros dispositivos. O grupo de aplicação consiste nas próprias aplicações que utilizam ligações Bluetooth. Estas podem incluir aplicações complexas ou aplicações orientadas a Bluetooth [30] [31].

Podem-se resumir as características das camadas do protocolo Bluetooth da seguinte maneira [29] [30] [31]:

- Camada de RF: corresponde essencialmente ao projeto dos *tranceivers*.
- Camada Baseband: define como os dispositivos *Bluetooth* se localizam e se conectam a outros dispositivos. Os papéis de servidor e cliente são definidos nesta camada. É nesta camada também que se definem os tipos de pacotes, processamento de pacotes, estratégias de deteção de erros, criptografia e transmissão de pacotes.
- Link Manager: implementa o Link Manager Protocol (LMP), que gere as propriedades do meio de transmissão (ar) entre os dispositivos. O protocolo LMP também gere a taxa de transferência de dados, autenticação, níveis de confiança entre dispositivos, criptografia de dados e controle do gasto de energia.
- Camada L2CAP: serve de *interface* entre os protocolos de camadas superiores e os protocolos de transporte de camadas inferiores. Esta camada também é responsável pela fragmentação e remontagem de pacotes.

Todos os dispositivos que implementam a especificação Bluetooth devem possuir minimamente seis componentes de *hardware*, como se mostra na Figura 3.5 [30] [31]:

- Host Controller: responsável pelo processamento de código de alto nível, tanto de aplicações como de algumas camadas inferiores das camadas de protocolos Bluetooth – controle de ligações lógicas, L2CAP, RFComm.
- Link Control Processor: um microprocessador responsável pelo processamento

das camadas mais baixas da pilha de protocolos como Link Manager e Link Controller.

- Baseband Controller: bloco lógico responsável pelo controlo do *transceiver* de RF.
- Transceiver RF: contém o sintetizador de rádio frequência e detetor de dados.
- RF Front-End: contém o filtro de banda passante da antena e é responsável pela troca de estados entre o emissor e o recetor.
- Antena.

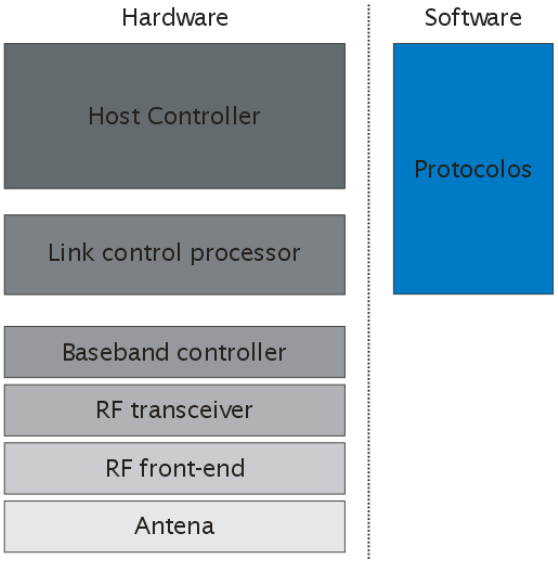

Figura 3.5 – Diagrama de blocos de *hardware* e *software* do Bluetooth
Adaptado de: [32]

3.3 Comunicação RS-232

A comunicação RS-232 é uma forma de diferentes equipamentos comunicarem com o outros dispositivos. Este tipo de comunicação é intitulado de série uma vez que os dados são enviados numa única linha, ou seja, é enviado um bit de cada vez. Este é um dos meios mais lentos de transferência de dados, porém, ainda é largamente utilizado devido à sua simplicidade e conveniência. Existem alguns protocolos de comunicação, mas esta dissertação abordará, principalmente, o RS-232 assíncrono [33].

Na transmissão assíncrona é permitida a transmissão dos dados sem ser necessário o envio do pulso do relógio para o recetor. Em vez disso, bits especiais são adicionados a cada

dado a fim de sincronizar o envio e receção dos dados [33] [34] [35].

Sendo assim, o protocolo de transmissão assíncrona é iniciado através de um bit de *start*, antes de cada byte de dados e encerrado com um *bit* de paridade (opcional) seguido de um bit de *stop*. O bit de *start* serve para iniciar o mecanismo de receção e o sinal de *stop* serve para parar o mecanismo de receção. O bit de paridade é usado para detetar erros em transmissões. [34] [35].

Na Figura 3.6, um bit de *start* é enviado, seguido por oito bits de dados, sem bit de paridade e um *stop* bit. O número de bits de dados, a formatação, a ordem dos bits de dados, bem como a velocidade de transmissão devem ser previamente acordados entre as partes que comunicam. O *stop* bit é na verdade um período de paragem, que pode ter um comprimento arbitrário. Ele não pode ser menor do que um valor especificado, normalmente este período tem 1 a 2 bits. No final de cada sequência, o recetor pára momentaneamente e fica a aguardar pelo próximo bit de *start*. É esta sequência que mantém o transmissor e o recetor em sincronismo [34] [35].

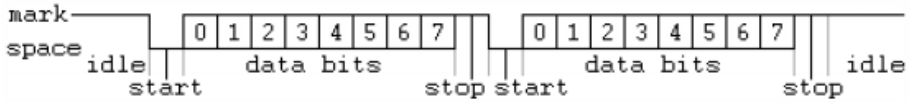

Figura 3.6 – Protocolo RS-232 Assíncrono
Fonte: [35]

O controlador da porta série é o Universal Asynchronous Receiver-Transmitter (UART). Quando se transmitem dados através da porta série, o UART converte os dados em formato paralelo, provenientes da CPU, em série para estes serem enviados um bit de cada vez. Ao receber os dados a partir da porta série, o UART converte os dados de volta em paralelo para a Unidade Central de Processamento (CPU) do dispositivo recetor utilizar. Para além da conversão de dados, o UART também indica o estado do meio de transmissão de modo a regular o fluxo de dados. Adicionalmente, o UART contém um *buffer* para guardar temporariamente os dados da transmissão até a CPU estiver disponível para receber novos dados [34] [35].

3.4 Sumário

As três tecnologias de comunicação consideradas nesta proposta para monitorização e assistência a pacientes são a TCP/IP, Bluetooth e comunicação RS-232. O TCP/IP é a tecnologia

de base para a internet e é utilizada para a comunicação entre qualquer interligação de redes. O Bluetooth é uma tecnologia de comunicação sem fios de curto alcance, baixo custo e baixo consumo. A comunicação série assíncrona é uma tecnologia bastante antiga, mas ainda muito usada na ligação de dispositivos a um computador pessoal.

Capítulo 4 - Infraestrutura de Comunicação

Neste capítulo são apresentados e especificados todos os componentes necessários para a implementação da infraestrutura de comunicação tendo em vista as características pretendidas. Para além disso, descrevem-se todas as etapas desde a arquitetura do sistema até à sua implementação.

4.1 Arquitetura da Infraestrutura de Comunicação

Uma infraestrutura que se presume ser muito interessante como sistema *e-Health* é o desenvolvimento de uma plataforma que contenha três características: flexibilidade, integridade e autenticação. Neste sentido, este sistema foi desenvolvido de modo a ser utilizado quer em ambiente privado (em que um ou um pequeno número de utentes utiliza o sistema para medir, registar e controlar algumas variáveis associadas à sua condição clínica) quer em ambiente clínico ou de *check up*, de modo a aumentar a vigilância e estado de profilaxia. Por último, existe a possibilidade da sua expansão progressiva (integração de novos sensores) para aquisição de outros sinais biológicos relevantes.

A arquitetura proposta para o sistema foi pensada tendo em conta a sua escalabilidade e adaptabilidade.

A escolha das tecnologias e a separação das várias camadas do sistema permite que os seus componentes possam ser testados isoladamente e também ser implementados noutras tecnologias, caso seja necessário.

4.1.1 Camadas da Infraestrutura de Comunicação

A infraestrutura de comunicação foi desenvolvida num *software* multicamadas com base nas características e funções de cada camada de rede.

A infraestrutura de comunicações utiliza o padrão de comunicação *publisher/subscriber* para a transmissão de dados, que permite aos dispositivos publicarem os sinais vitais, localizações e alertas para *smartphones* ou PCs remotamente por quem subscreva esses dados. O *publisher/subscriber* é um paradigma de comunicação que procura prover o desacoplamento

e flexibilidade necessários para aplicações distribuídas de larga escala. Nesta infraestrutura foi utilizada a variação *Type-Based publisher/subscriber*. Nesta variação os *subscribers* expressam o interesse subscrevendo o tipo de dados do evento que lhes interessa.

Nesta infraestrutura foi utilizada a estrutura de endereçamento definida no IPV6. Como cada tecnologia de comunicação tem o seu tipo de endereçamento decidiu-se arranjar uma forma de tornar esse processo de endereçamento único e fiável. Assim sendo, escolheu-se o endereçamento do IPV6 como forma de identificar todos os dispositivos da infraestrutura. Cada nó da rede tem que ser endereçado com o seu endereço e não é permitido haver dois nós diferentes com o mesmo endereço.

Seguidamente, descrevem-se, de forma breve e sucinta, os elementos básicos da arquitetura.

4.1.1.1 Domain Server

O *domain server* (DS) é o responsável pela coordenação de todas as conexões dos dispositivos no sistema, bem como a segurança, o registo de todos os nós e serviços e a verificação dos endereços. Além disso, faz a coordenação das ligações entre nós, nos casos de subscrição de dados, e também serve de ponte entre diferentes tecnologias de comunicação (Ethernet e Bluetooth, por exemplo).

Na infraestrutura de comunicação só existe um DS, que requer Ethernet, Wi-Fi ou Bluetooth como tecnologias de comunicação. Deste modo, o s*oftware* do DS pode ser executado numa variedade de plataformas: *smartphones*, PCs ou plataformas como a *pandaboard*, por exemplo.

É ainda importante salientar que o DS evita o congestionamento da rede e a sobrecarga de informação nesta infraestrutura. Por exemplo, se um *local server* (LS) quer subscrever um serviço que está a ser publicado noutro LS, o DS fornece as credenciais e autorizações necessárias para se ligaram mutuamente. Assim, a infraestrutura terá uma auto-organização, uma vez que uma nova ligação pode ser rapidamente criada e os dados reencaminhados em conformidade.

4.1.1.2 Local Server

Um *local server* (LS) é qualquer dispositivo que tenha uma das quatro tecnologias de

comunicação implementadas: Bluetooth, Wi-Fi, Ethernet ou RS-232. Sendo assim, o *software* do LS pode ser executado numa variedade de plataformas como, por exemplo, *smartphones*, PCs, sensores e microcontroladores.

Quando se pretende ligar um LS à infraestrutura de comunicação é necessário registá-lo no DS. Para o efeito, o LS necessita de se ligar ao DS ou a outro LS pertencente à infraestrutura. Quando é ligado diretamente ao DS, o LS é intitulado de *capable*; quando necessita de um LS *capable* para se ligar à infraestrutura e transmitir os seus dados é rotulado de *ineffective*. Um LS *ineffective* é um tipo de nó que necessita de um LS *capable* para comunicar com o DS. Ou seja, todo o processo inerente à comunicação (conexão, registo e subscrição de serviços) entre um LS *ineffective* e o DS é efetuado através deste LS *capable*.

Os sensores, os dispositivos série, bem como qualquer outro dispositivo (*smartphone*, PC) que não esteja ligado diretamente ao DS são exemplos de LS's *ineffectives*. A Figura 4.1 representa um exemplo da ligação de um LS *ineffective* à infraestrutura de comunicação.

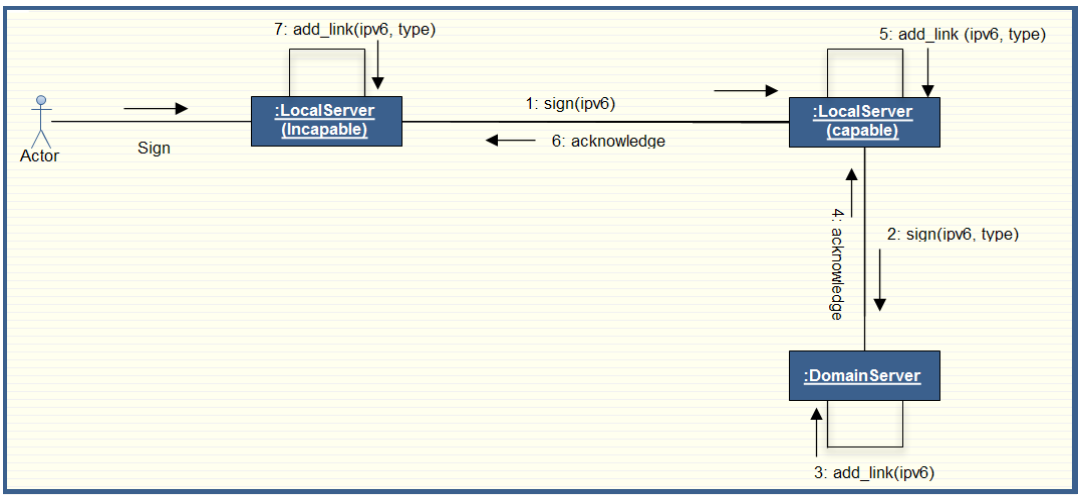

Figura 4.1 – Ligação de um LS *ineffective* à infraestrutura

Como o LS *ineffective* não está diretamente ligado ao DS ele propaga ou subscreve serviços por meio do seu LS capable. A Figura 4.2 representa um exemplo da propagação de um serviço do LS *ineffective*.

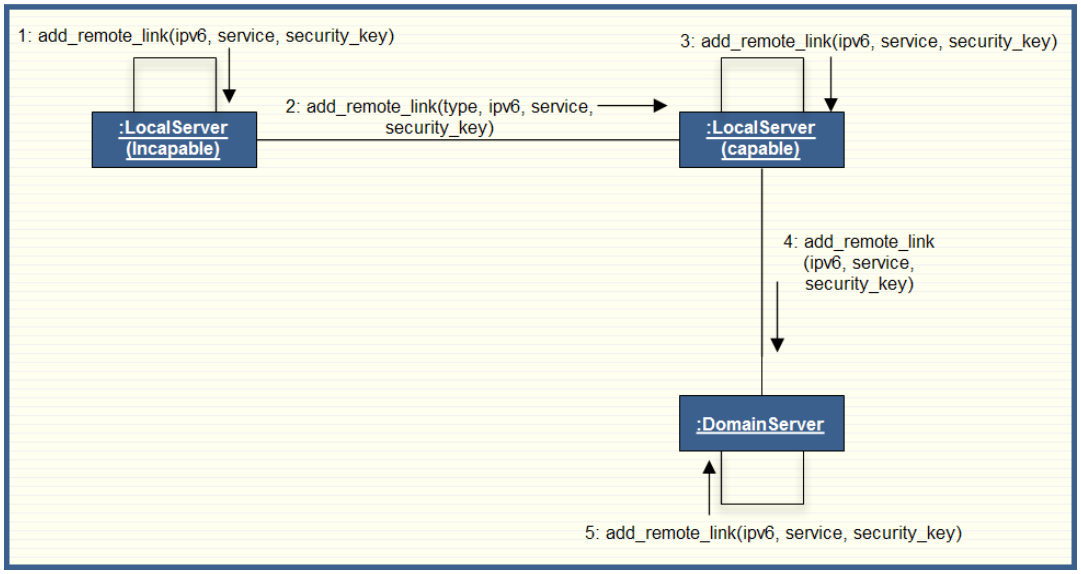

Figura 4.2 – Transmissão de serviços do LS *ineffective à* infraestrutura

Um LS só tem informação dos serviços que o rodeiam. Ou seja, enquanto que o DS tem informação de todos os serviços e ligações existentes na infraestrutura de comunicação, o LS só tem conhecimento dos serviços próximos (que o rodeiam) e dos LS *ineffectives* que estão ligados diretamente a ele.

Na Figura 4.3 encontram-se representadas algumas das formas de um LS se ligar à infraestrutura. A comunicação entre qualquer LS e o DS pode ser feita de um modo direto ou indireto ou via Ethernet/WiFi ou Bluetooth.

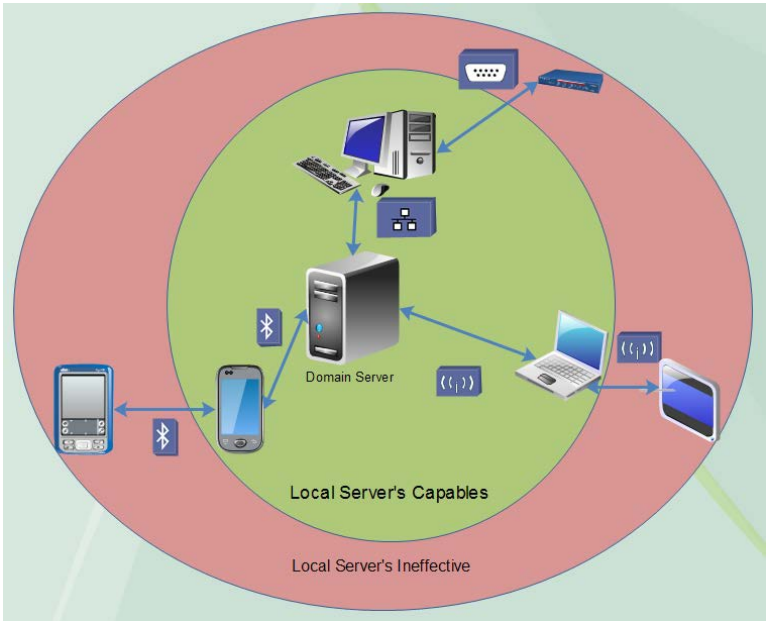

Figura 4.3 – Tipos de *Local Server*

O processo de desligar um LS é muito semelhante à sua ligação. Na Figura 4.4 está

representado o caso de uso da desconexão de um LS *ineffective*. Como é possível observar, o LS *ineffective* comunica ao seu LS que, por sua vez reencaminha a mensagem ao DS.

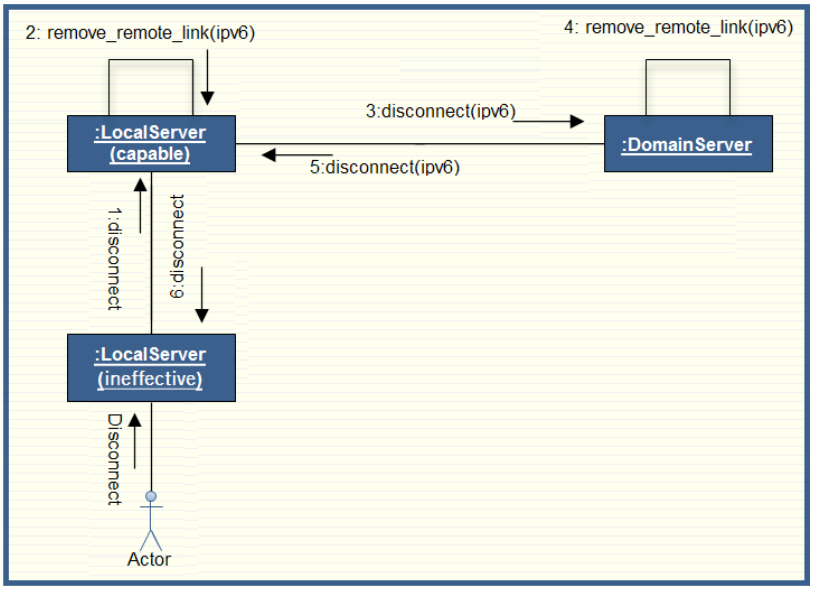

Figura 4.4 – Desconexão de um LS *ineffective*

Na Figura 4.5 é pode-se observar o processo de desconexão de um LS *capable*. Como o LS *capable* se vai desligar da infraestrutura, ele informa o LS *ineffective* e obriga-o a encerrar. Depois disto, o DS é informado da desconexão dos dois dispositivos.

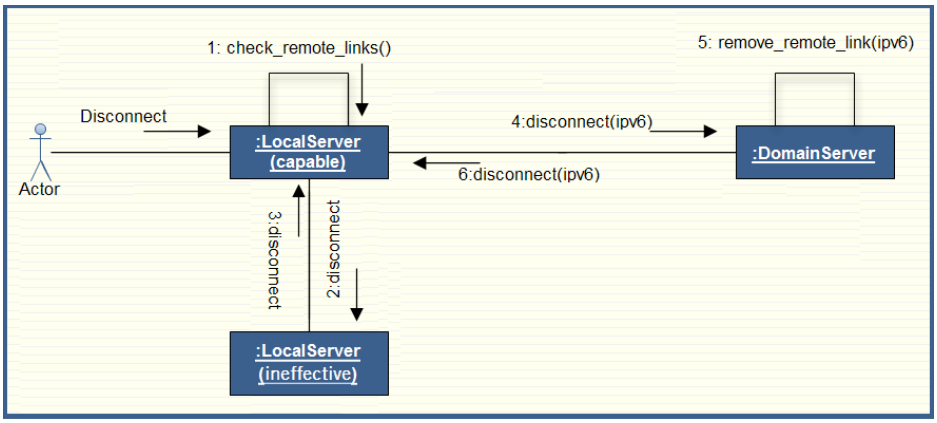

Figura 4.5 – Desconexão de um LS capable

No caso de um LS não ser um sensor, este fornece uma *interface* gráfica para os utilizadores, permitindo efetuar as seguintes operações:

- Autenticar utilizadores.
- Subscrever serviços.
- Visualizar dados fisiológicos.

- Gerar alertas de emergência para médicos e familiares quando são detetadas condições anormais.

4.1.2 Nós da Arquitetura

Cada elemento constituinte da infraestrutura pode desempenhar diferentes padrões de comunicação consoante a situação a que for sujeito.

4.1.2.1 Publisher

O *publisher* é o LS que tem a função de publicar dados para a infraestrutura de comunicação. O *publisher* compartilha os conteúdos da sua autoria com os *subscribers* de um determinado serviço. Nesta infraestrutura de comunicação foram implementados adaptadores para publicarem os dados dos seguintes dispositivos: sistema de aquisição de sinais (baseado no microcontrolador PIC32MX795F512L), monitor A&D Blood Pressure UA-767PBT-C40 e unidade Shimmer R2.

A Figura 4.6 mostra o caso de uso de um LS com a função de *publisher*. Quando um *publisher* deseja publicar um determinado serviço, verifica os seus *subscribers* e transmite os dados a cada um deles.

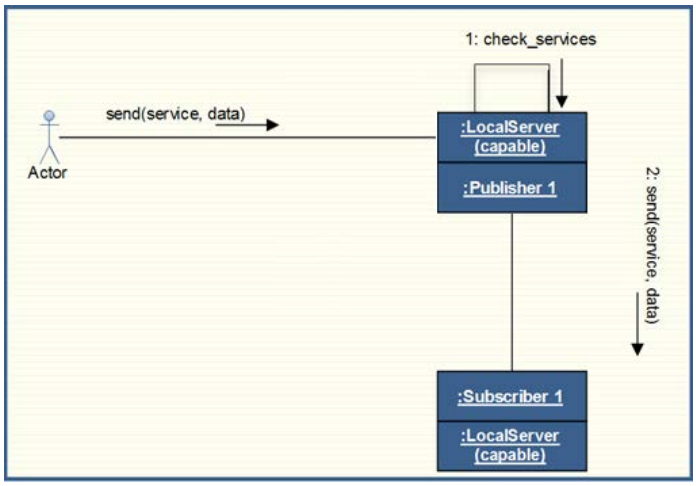

Figura 4.6 – LS a desempenhar papéis de *Publisher* e *Subscriber*

Outro ponto importante no padrão de comunicação *publisher/ subscriber* é a remoção de serviços. Quando é removido um serviço da infraestrutura os seus correspondentes *subscribers* têm que ser cancelados. Nas Figura 4.7 e Figura 4.8 é possível observar o caso de uso da remoção de um *publisher* e o consequente cancelamento do seu *subscriber*. Na Figura 4.7, quando o *publisher* é cancelado o LS transmite o sucedido ao DS que por sua vez cancela o

subscriber.

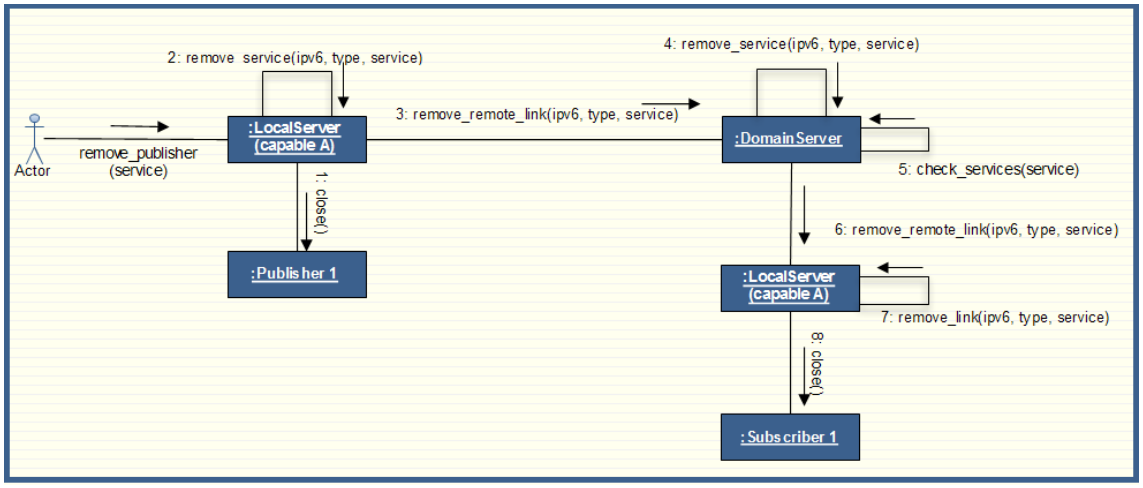

Figura 4.7 – Caso de uso de remoção de um *publisher* e do respetivo *subscriber* sem ligação entre LS's

No caso da Figura 4.8, em que os LS's se encontram ligados entre si, o LS *publisher* informa o LS *subscriber* da remoção do serviço e fecha a ligação no caso de não haver mais nenhum serviço em comum entre eles. O DS também é informado da remoção do serviço.

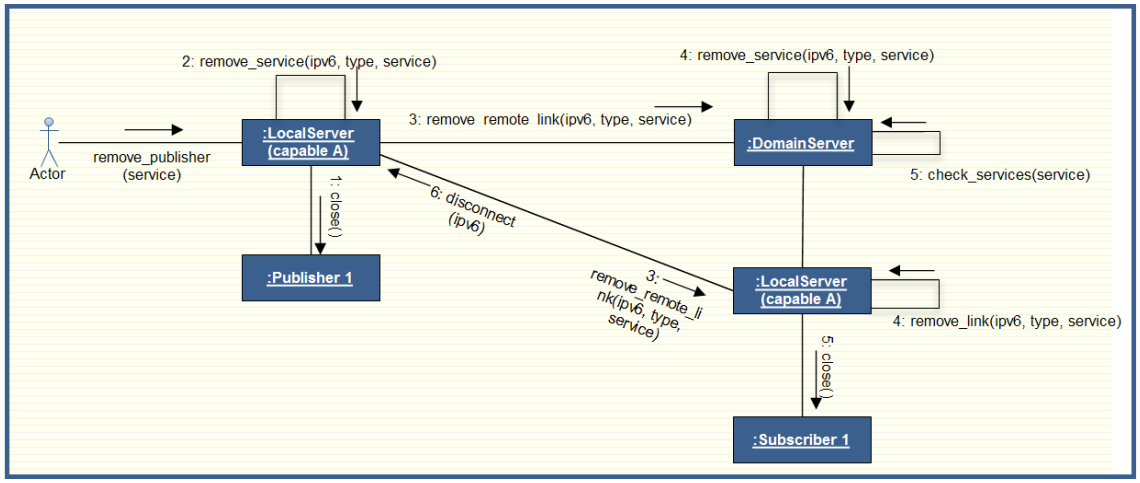

Figura 4.8 - Caso de uso de remoção de um *publisher* e do respetivo *subscriber* com ligação entre LS's

4.1.2.2 Subscriber

O *subscriber* é um LS que subscreveu um determinado serviço, sendo notificado pelo *publisher* com os dados desse serviço. O *subscriber* só pode ser criado após a autorização do DS.

A Figura 4.6 mostra também a subscrição de serviços de um *publisher*. Após a chegada de dados, o trabalho do *subscriber* é mostrar os dados ao utilizador.

No que diz respeito à remoção de *subscribers*, as Figura 4.9 e Figura 4.10 mostram o

processo de remoção. Tal com acontece na remoção do *publisher*, o DS é sempre informado pelo cancelamento do serviço. Note-se, porém, a diferença de neste caso o *publisher* continuar ativo após a remoção dos seus *subscribers*.

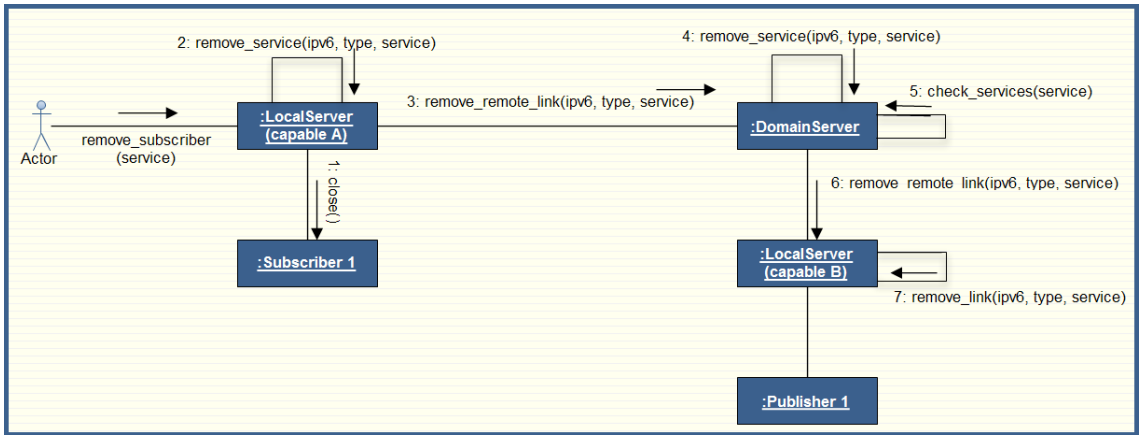

Figura 4.9 - Caso de uso de remoção de um *subscriber* sem ligação entre LS's

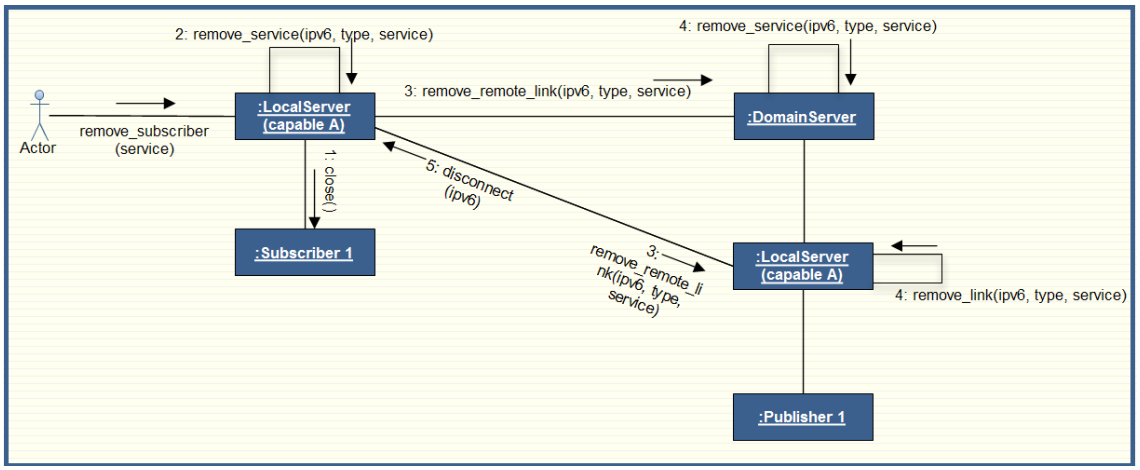

Figura 4.10 - Caso de uso de remoção de um *subscriber* com ligação entre LS's

4.1.2.3 Bridge

Como existem na infraestrutura de comunicação LS's com diferentes tecnologias de comunicação (Bluetooth e Ethernet) é necessário um nó que os interligue. Assim, a *bridge* é uma função desempenhada por um nó da infraestrutura, que permite a troca de mensagens entre LS's com diferentes tecnologias de comunicação.

A Figura 4.11 mostra um caso de uso em que o DS faz a ponte entre dois LS's incapazes de se ligarem diretamente.

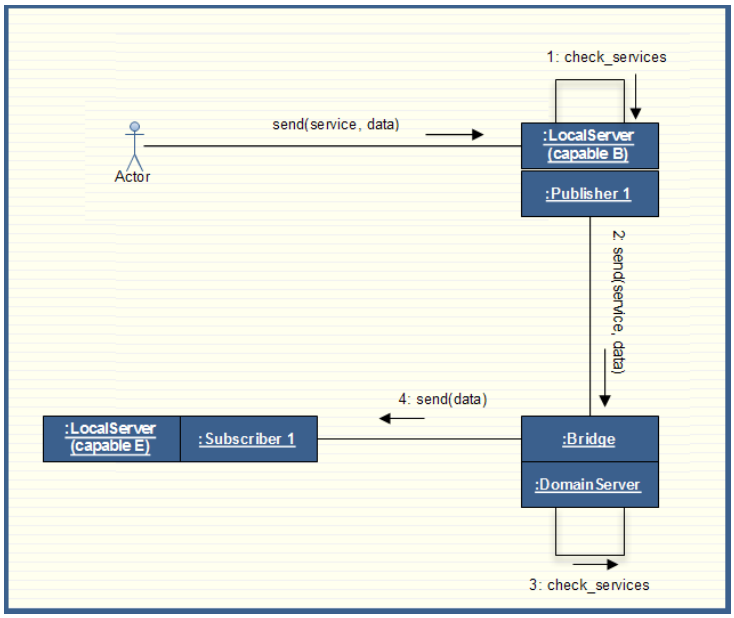

Figura 4.11 – DS desempenha papel de *Bridge*

4.1.3 Padrões da Arquitetura

O *Router* e o *Dealer* são os padrões de comunicação utilizados nesta infraestrutura. Cada um deles tem papéis distintos.

4.1.3.1 Router

O *router* tem o papel de encaminhar os dados provenientes de um LS e funciona como *frontend* de uma ligação. O *router* recolhe as mensagens de um conjunto de *publishers* e encaminha-as para o *dealer*.

4.1.3.2 Dealer

O *dealer* tem o papel de receção de mensagens de dados e respetivo envio para *N* nós. Ele funciona como *backend* de uma ligação. Quando o *router* lhe reencaminha as mensagens recebidas pelos *publishers*, o *dealer* verifica os *subscribers* existentes e, de seguida, transmite os dados.

A Figura 4.12 mostra um caso de uso em que um LS realiza simultaneamente o papel de *router* e de *dealer*. Quando forem publicados dados o *router* recebe os dados transmitidos e reencaminha-os para o *dealer*. Este verifica os *subscribers* existentes para aquele serviço e transmite os dados para cada um dos *subscribers*.

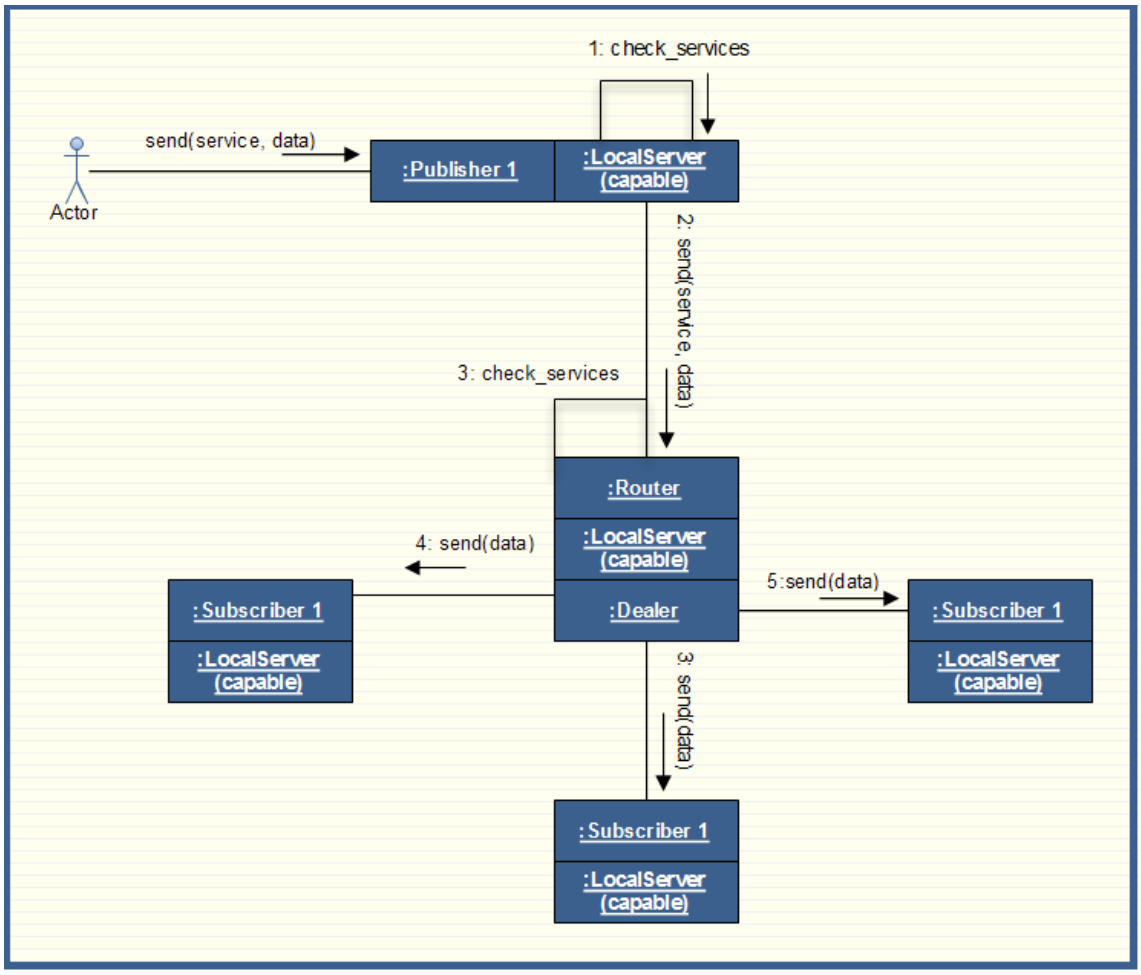

Figura 4.12 – LS desempenha papel de Router e Dealer

4.2 Implementação da Infraestrutura

A implementação dos protocolos de comunicação envolve a criação e manutenção das ligações, a transmissão de dados para o destino pretendido e ainda a interpretação dos comandos trocados por cada nó da infraestrutura de comunicação.

Nesta seção descreve-se a implementação dos protocolos de comunicação. Em primeiro lugar, apresenta-se o protocolo de comunicação entre todos os nós não sensores da infraestrutura. Em seguida, descrevem-se todos os protocolos de comunicação entre os sensores e os seus LS's. Por último, é discutido o sistema de segurança dos dados e a *interface* para o utilizador.

4.2.1 Implementação do protocolo da Infraestrutura

Todas as mensagens da infraestrutura de comunicação têm um identificador associado quando são enviadas quer pelo LS quer pelo DS. Estas mensagens contêm o identificador do

comando a executar e o seu conteúdo.

De seguida, descrevem-se as mensagens utilizadas pela infraestrutura de comunicação:

- *Sign* – comando enviado pelo LS para o DS com os dados de autenticação. Este comando pode também ser enviado por um LS *ineffective* a um LS *capable* que encaminha o comando para o DS.
- *ACK* – comando enviado pelo DS para o LS como resposta positiva à autenticação.
- *Disconnect* – comando utilizado para fechar a ligação, que pode ser utilizado quer pelo DS quer pelo LS. Este comando tem não só a funcionalidade de recusar ligações, mas também permitir que o utilizador possa desconectar-se da infraestrutura.
- *Add remote link* – comando enviado quando se pretende registar um novo serviço na infraestrutura (*publisher* ou *subscriber*). Este comando pode ser utilizado pelo DS e pelo LS.
- *Remove remote link* – comando enviado quando se pretende cancelar um serviço na infraestrutura (*publisher* ou *subscriber*). Este comando pode ser utilizado pelo DS e pelo LS.
- *Data* – comando utilizado para a publicação de dados. Este comando pode ser utilizado pelo DS e pelo LS.
- *Connect* – comando enviado pelo DS a um LS que pretende subscrever um serviço noutro LS com capacidades de comunicação compatíveis. Este comando também é enviado pelo LS *subscriber* ao LS *publisher* para efetuar a ligação.
- *Accept* – comando enviado pelo DS ao LS *publisher* para aceitar a ligação de um LS *subscriber*.

Na Figura 4.13 mostra a estrutura do pacote de dados do protocolo de comunicação para cada comando. Cada comando é iniciado pelo identificador seguido dos dados solicitados.

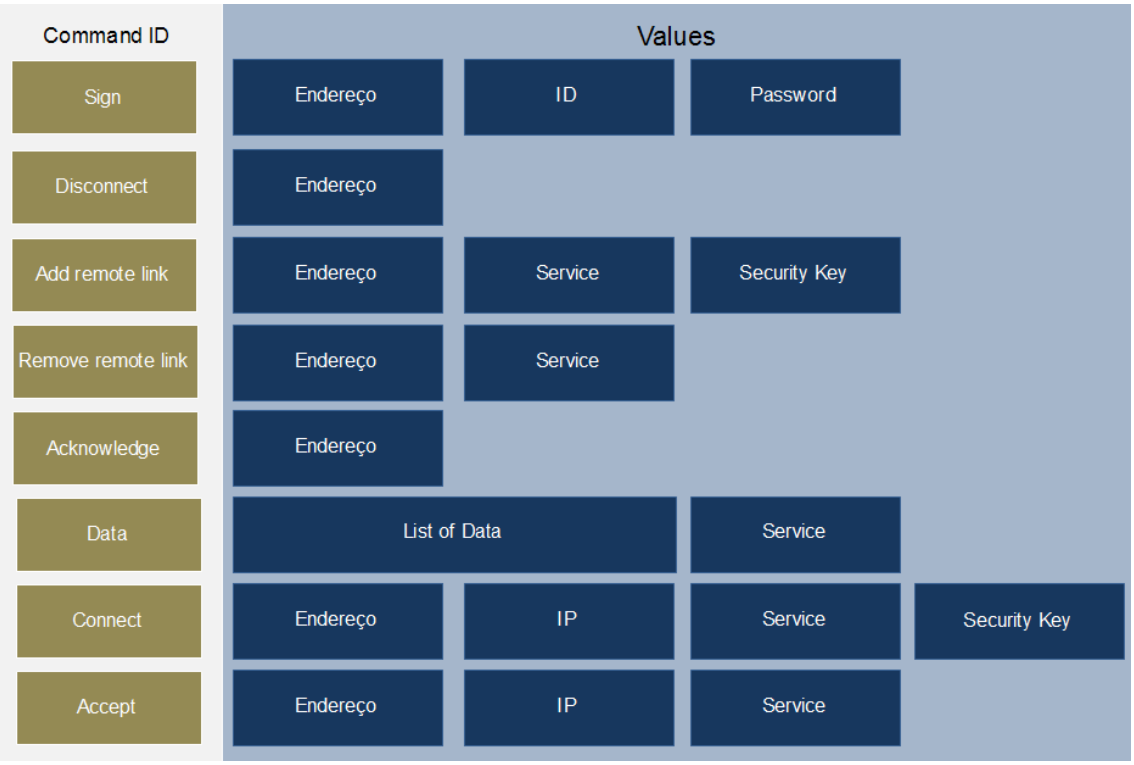

Figura 4.13 – Estrutura do pacote de dados do protocolo de comunicação

4.2.2 Sistema de aquisição com comunicação série

Na implementação do sistema de aquisição foi utilizado o microcontrolador PIC32MX795F512L [36], que envia os dados usando o protocolo série RS-232. Este sistema permite adquirir um sinal analógico, que depois é enviado para um LS.

Nesta dissertação definiram-se três períodos de amostragem diferentes para simular três serviços: S1, S2, S3. Para ser possível o envio dos três serviços em simultâneo para o LS, sem qualquer perda de dados, foram escolhidas as frequências de amostragem apresentadas na Tabela 4.1. A escolha destas frequências de amostragem foi baseada no *baudrate* (115200bps) escolhido para o protocolo RS-232.

Tabela 4.1 – Frequência de amostragem dos serviços do sistema de aquisição

Serviço	Frequência da interrupção (Hz)
S1	512
S2	256
S3	128

Na implementação deste sistema de aquisição foi desenvolvido um adaptador na

infraestrutura de comunicação responsável pela ligação série. Para iniciar o adaptador é necessário definir a porta à qual o dispositivo se encontra ligado. Após a inicialização do adaptador, o LS envia um comando de início (SIGN) para o dispositivo. No caso da receção correta do comando, o dispositivo responde também com o comando de início (SIGN) com o seu endereço. Na Figura 4.14 é representado o caso de uso da ligação entre o sistema de aquisição e o LS.

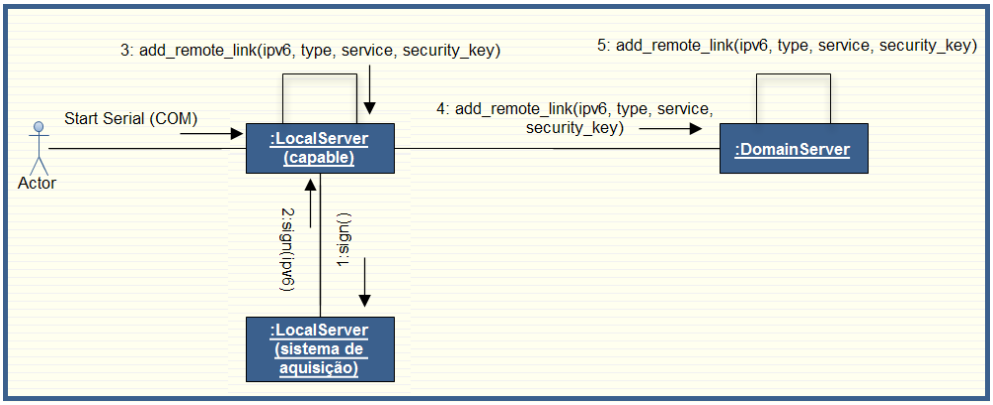

Figura 4.14 – Caso de uso da ligação do sistema de aquisição ao LS

Após efetuada a ligação, o sistema de aquisição pode encontrar-se em dois diferentes estados: ligado e *streaming*. No estado ligado, o sistema encontra-se preparado para receber qualquer pedido do LS. Quando o comando para adicionar um *subscriber* é recebido, o sistema entra no estado de s*treaming* e começa a amostragem de dados do serviço requerido. Quando o comando para remover a subscrição é recebido o sistema volta ao estado ligado. A Figura 4.15 mostra o diagrama de estados do sistema de aquisição.

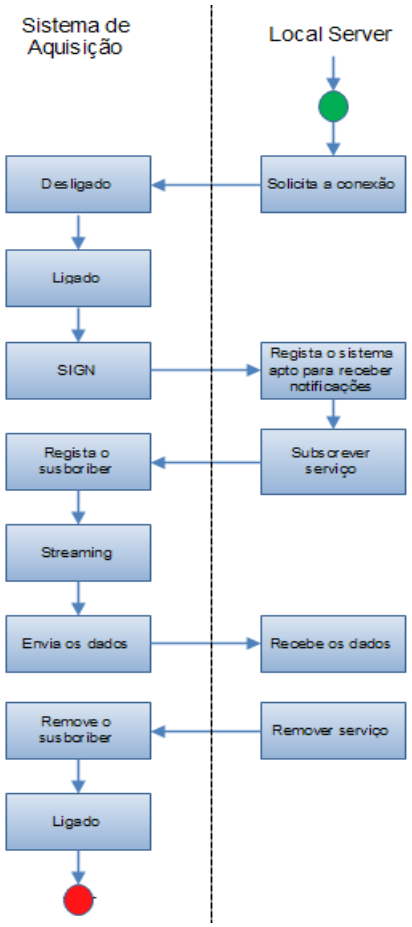

Figura 4.15 – Diagrama de sequência da implementação do sistema de aquisição

De seguida, exemplica-se a troca de mensagens entre o sistema de aquisição e o *local server*. Neste exemplo, pretende-se publicar fontes de dados cujos valores são representados por palavras de 16 bits. Como o A/D do microcontrolador é de 10 bits e de modo a não perder a sua máxima resolução utilizaram-se palavras de 16 bits. Devido ao protocolo de comunicação série RS-232, as tramas de dados e de controlo têm que ser partidas e enviadas byte a byte.

De forma a identificar os diferentes tipos de serviços, foi definido um conjunto de comandos cujo *header* é apresentado na Tabela 4.2.

Tabela 4.2 – Protocolo de Comunicação Série

Comando	**Bits**	**Bits**							
	Range	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
Disconnect	15:8	Endereço							
	7:0	1	0	0	0	0	0	0	0
Sign	15:8	Endereço							

	7:0	1	0	0	0	0	0	0	1
Add Remote Link	15:8	0	0	0	S2	S1	S0	0	0
	7:0	1	0	0	0	1	0	0	0
Remove Remote link	15:8	0	1	0	S2	S1	S0	0	0
	7:0	1	0	0	0	0	1	1	1
Data	15:8	0	DATA						
	7:0	1	DATA		S2	S1	S0	1	1

Na Tabela 4.2, os bits S1, S2 e S3 correspondem aos serviços. Ou seja, quando o dispositivo de aquisição envia um comando Data correspondente ao serviço S0 para o LS este coloca o S2 e o S1 (bits 4 e 3 correspondentes) a 0 e coloca o S0 (bit 2) a 1. Deste modo, o LS identifica a que serviço corresponde os dados recebidos. Os comandos *add remote link* e *remove remote link* funcionam da mesma forma.

4.2.3 Implementação do Sensor de Pressão Arterial

O monitor A&D Blood Pressure UA-767PBT-C40 (Figura 4.16) pode ser utilizado quer em Centros de Saúde quer em casa do paciente.

Figura 4.16 - Monitor A&D Blood Pressure UA-767PBT-C40

Este sensor conecta-se a um ponto de acesso, que contém o adaptador de ligação à infraestrutura de comunicação, via Bluetooth Serial Port Profile (SPP). O SPP Bluetooth cria uma porta de série para comunicações de entrada do monitor de pressão arterial.

A Figura 4.17 mostra as camadas da arquitetura do monitor. A camada RFCOMM oferece o protocolo de transporte para criação da porta série. O SDP é responsável pelo

protocolo de descoberta de dispositivos Bluetooth. A camada da porta série é responsável por emular a porta de ligação e fornecer uma API para a aplicação.

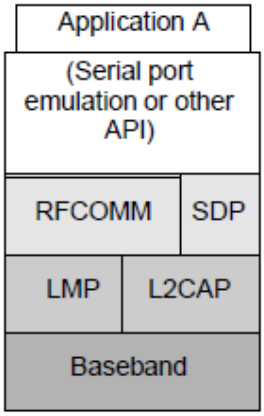

Figura 4.17 – Arquitetura em camadas do monitor A&D Blood Pressure UA-767PBT-C40

Como se mostra na Figura 4.18, a fim de ligar o monitor a um LS são necessárias as seguintes etapas:

- O LS esteja com o adaptador ativo para o monitor se ligar.
- Realizar a medição com o sensor de pressão arterial. No fim da medição o sensor tentará estabelecer comunicação com o último LS que estabeleceu ligação. Se esse LS não for encontrado o monitor de pressão arterial inicia a procura de um novo LS. Se nenhum LS for encontrado o monitor desliga-se e os dados são guardados na memória.
- Quando a comunicação é estabelecida pela primeira vez a palavra-passe será solicitada, que está disponível no Non-Disclosure Document (NDA) do sensor de pressão arterial.
- A instalação é feita e a comunicação é possível entre o monitor e o LS.
- Agora que a comunicação é iniciada, é necessário criar o protocolo de comunicação adequado de modo a receber os dados medidos pelo sensor de pressão arterial.

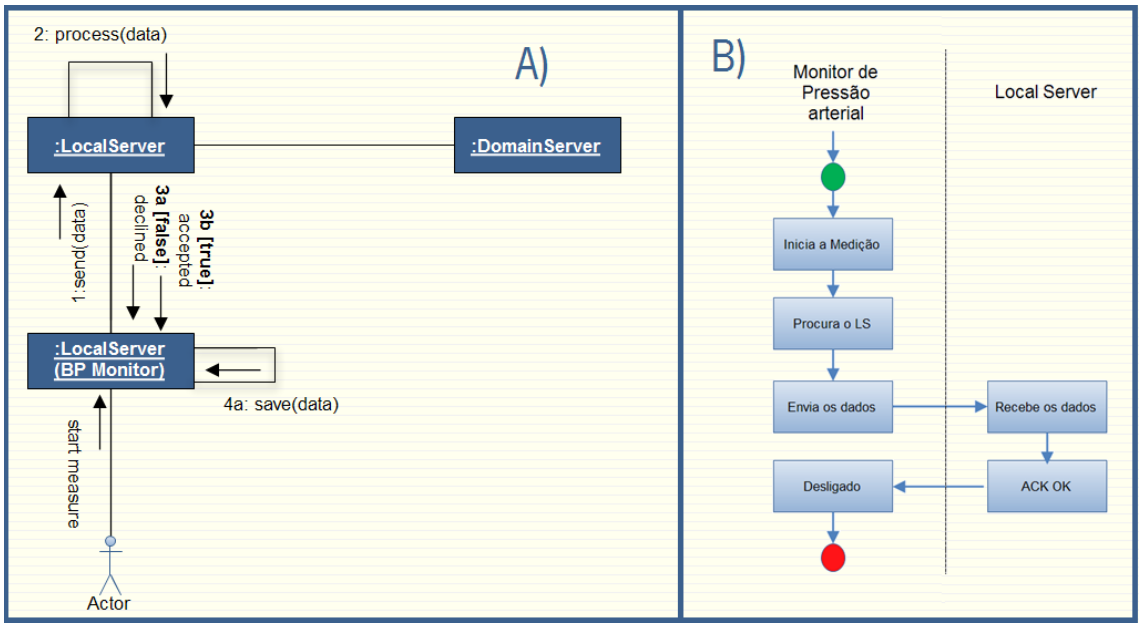

Figura 4.18 – A) Caso de Uso do monitor de pressão arterial B) Diagrama de sequência do monitor de pressão arterial

4.2.4 Implementação do Shimmer

A unidade Shimmer R2 consiste num dispositivo constituído por uma gama de sensores. Esta unidade incorpora: ECG, EMG, acelerómetro de 3 eixos, giroscópio, magnetómetro, sensor de resposta galvânica da pele (GSR) e strain gauge. A Shimmer é programada com *firmware* BtStream e pode estar num dos três estados: desligado, ligado ou s*treaming*. Quando a unidade Shimmer é ligada pela primeira vez, encontra-se no estado desligado e permanece nele até que uma ligação Bluetooth seja efetuada, ou seja, através da abertura de uma ligação série.

No estado ligado, a unidade Shimmer pode processar vários comandos para configurar os seus sensores e parâmetros de amostragem, calibração e iniciar a amostragem. Quando o comando para iniciar a amostragem é recebido, a unidade Shimmer entra no estado de s*treaming* e começa a amostragem de dados dos seus sensores e envia os dados através da ligação Bluetooth. Isto continua até que um comando para parar a subscrição é recebido, e assim a unidade Shimmer retorna ao estado ligado. Quando a ligação Bluetooth é fechada a unidade Shimmer desliga-se. A Figura 4.19 mostra o diagrama de estados da unidade Shimmer.

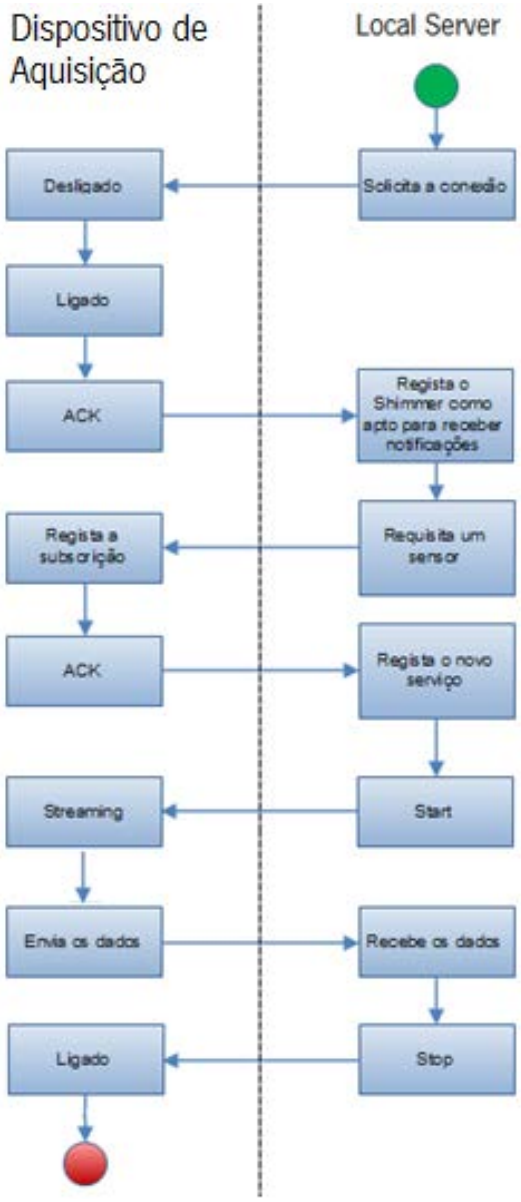

Figura 4.19 – Diagrama de estados da unidade Shimmer

Quando a unidade Shimmer está no estado ligado ou s*treaming*, não pode haver uma comunicação ativa entre a unidade Shimmer e o LS. Os pacotes de bytes são enviados em ambas as direções e estes podem consistir comandos, respostas ou dados.

A unidade Shimmer tem três grandes grupos de comandos: *set*, *get* e *action*.

Os comandos *set* são usados para definir os valores de todos os parâmetros configuráveis, tais como:

- Ativar os sensores.
- Escolher a taxa de amostragem.

- Escolher a gama do acelerómetro.
- Ligar ou desligar o regulador de 5V.
- Ligar ou desligar o *mutiplexer* de alimentação.
- Calibrar o acelerómetro, giroscópio, magnetómetro, EMG e ECG.
- *Auto-range* da gama do GSR.
- Controlar o Light Emitting Diode (LED).
- Escolher o tamanho do *buffer*.
- Controlar o ganho magnetómetro e a taxa de dados interno.

A troca de dados entre a unidade Shimmer e o *local server* (LS) para os comandos *set* são mostrados na Figura 4.20. O primeiro byte de cada pacote recebido pela unidade Shimmer ou o LS é um identificador, dizendo ao recetor qual a ação a realizar ou como interpretar os bytes seguintes. Para todos os pacotes que a unidade recebe, ela envia uma mensagem de confirmação (ACK) de volta para o LS, de modo a acusar a receção do comando.

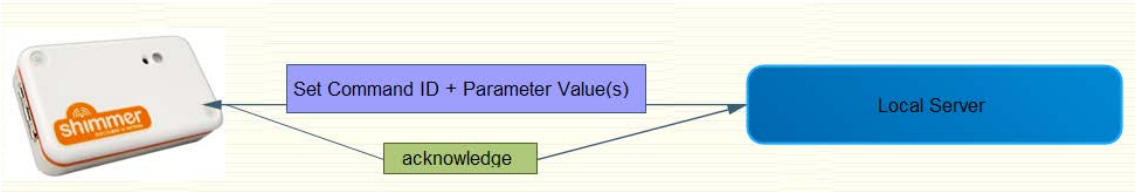

Figura 4.20 – Pacote para os comandos *set*

Os comandos *set* requerem que a unidade Shimmer receba dados após o byte de identificação. Por exemplo, quando se pretende alterar a taxa de amostragem, o pacote de dados vindo do LS deve conter um byte correspondente ao identificador seguido por um byte, que representa a taxa de amostragem a usar. Outro exemplo, é a calibração do acelerómetro, em que o pacote de dados deve ser iniciado com o identificador e seguido pelos parâmetros de calibração do acelerómetro.

Os comandos *get* são utilizados para pedir informação e exigem que a unidade Shimmer envie dados de volta para o LS. A troca de dados entre a unidade Shimmer e o LS para os comandos *get* são mostrados na Figura 4.21.

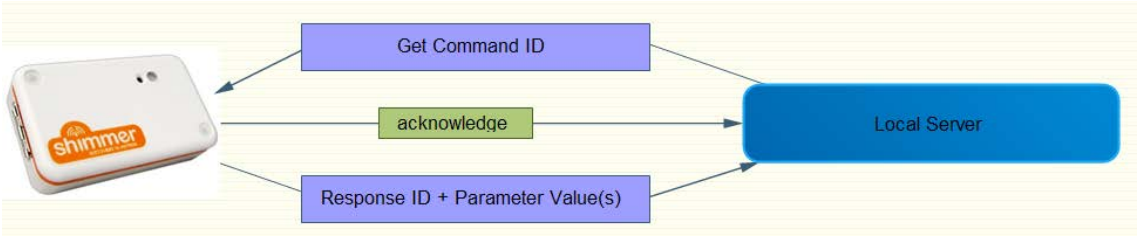

Figura 4.21 – Pacote para os comandos get

Após a receção de um comando *get*, a unidade Shimmer enviará uma mensagem de confirmação e, em seguida, irá preparar e enviar um pacote que contém o byte identificador, seguido dos dados que foram solicitados.

Por exemplo, a unidade Shimmer, quando recebe o comando para saber o valor da taxa de amostragem, envia um pacote de dados que contém dois bytes. O primeiro byte será o identificador e o segundo byte será o valor da taxa de amostragem. Do mesmo modo, se a unidade recebe o comando para saber os parâmetros de calibração do acelerómetro, ela irá enviar um pacote cujo primeiro byte é o identificador, seguido de 21 bytes que representam os parâmetros de calibração do acelerómetro.

Os comandos *action* são utilizados para o LS pedir à unidade Shimmer que leve a ação a cabo. Por exemplo, quando se pede a subscrição do sensor ECG, a unidade Shimmer prepara o sensor mas só envia os dados quando recebe o comando de *start*. Os dados do sensor são transmitidos até a unidade Shimmer receber um comando de *stop*.

Quando o *start* é recebido pela unidade Shimmer, esta irá enviar uma mensagem de reconhecimento de volta para o LS e inicia a transmissão dos dados do sensor. O tamanho do *Buffer* determina o número de amostras que são enviadas juntas num único pacote de dados. Na Tabela 4.3 mostra a estrutura do pacote de dados com o tamanho do *buffer* igual a dois, em que o TS indica a *timestamp* e o Ch indica o canal.

Tabela 4.3 – Estrutura do pacote de dados

Byte	0	1-2	3-4	5-6	...	(x-1)-x	(x+1)-(x+2)	(x+3)-(x+4)	(x+5)-(x+6)	...	(2x-1)-2x
Value	Packet Type	TS	Ch1	Ch2	...	ChX	TS	Ch1	Ch2	...	ChX
		Sample 1					Sample 2				

Se o tamanho do *buffer* for igual a um, então o pacote de dados contém apenas um *timestamp* e uma amostra de dados para cada um dos canais, ou seja, os bytes denotadas "Sample 1" na Tabela 5-2. Se o tamanho do *buffer* for superior a dois, então no início de cada pacote é adicionado o *timestamp* seguido da amostra de dados de cada canal até que o número de amostras seja igual ao tamanho do *buffe*r.

Os valores dos parâmetros de configuração são armazenados pela unidade Shimmer nos primeiros 90 bytes do Infomem, que é a parte da memória da unidade Shimmer que sobrevive a um *reset* ou ciclo de energia. O formato dos parâmetros de configuração da unidade Shimmer são os seguintes:

- Byte 0: Taxa de amostragem.
- Byte 1: Tamanho do *buffe*r.
- Bytes 2-11: Sensores selecionados.
- Byte 12: Gama acelerómetro.
- Byte 13: Configuração Magnetómetro.
- Bytes 14-17: Bytes de configuração.
- Bytes 18-38: valores de calibração acelerómetros.
- Bytes 39-59: valores de calibração giroscópios.
- Bytes 60-80: valores de calibração magnetómetro.
- Byte 81: gama GSR.
- Bytes 82-85: valores de calibração de EMG.
- Bytes 86-93: valores de calibração de ECG.

Os restantes dois bytes do Infomem são constituídos pela identificação de cada sensor contido na unidade Shimmer. Cada um destes sensores é identificado por um bit, como se observa na Tabela 4.4.

Tabela 4.4 – Bytes identificadores dos sensores da unidade Shimmer

Bits	Bits							
Range	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
15:8	Strain Gauge	Ritmo Cardíaco	Não Atribuído					
7:0	Acelerómetro	Giroscópio	Magnetómetro	ECG	EMG	GSR	ADC CH7	ADC CH0

É ainda importante salientar que determinadas combinações de sensores não podem ser utilizadas, em virtude da existência de conflitos entre eles. Na Tabela 4.5 são destacados os conflitos dos sensores: "OK" significa que ambos os sensores podem ser ativados simultaneamente, ao passo que "NA" significa que eles não podem. O adaptador implementado no LS, verifica a compatibilidade entre os sensores e em caso de incompatibilidade impede o utilizador de subscrever esse sensor.

Tabela 4.5 – Conflitos entre os sensores da unidade Shimmer

Sensor	Acel	Giro	Mag	ECG	EMG	GSR	ADCCH7	ADCCH0	Strain	Heart
Acel		OK	OK	OK	OK	OK	OK	OK	OK	OK
Giro	OK		OK	NA	NA	NA	OK	OK	NA	OK
Mag	OK	NA		NA	NA	NA	OK	OK	NA	OK
ECG	OK	NA	NA		NA	NA	OK	OK	NA	OK
EMG	OK	NA	NA	NA		NA	OK	OK	NA	OK
GSR	OK	NA	NA	NA	NA		OK	OK	NA	OK
ADCCH7	OK	OK	OK	OK	OK	OK		OK	OK	OK
ADCCH0	OK	OK	OK	OK	OK	OK	OK		OK	OK
Strain	OK	NA	NA	NA	NA	NA	OK	OK		OK
Heart	OK	OK	OK	OK	OK	OK	OK	OK	OK	

4.2.5 Segurança de dados

Uma vez que os dados recolhidos pelos sensores a partir do corpo de um utilizador serão enviados por uma rede "pública", a segurança é importante. Por isso, uma das características relevantes de uma infraestrutura de comunicação é fornecer segurança na

transmissão de dados, através da encriptação de mensagens dos dispositivos, antes de enviá-los através de Bluetooth / Ethernet. A autenticação também é requerida para que o *domain server* se assegure que comunica com o verdadeiro *local server* e vice-versa.

Para além da segurança, uma infraestrutura de comunicação deve ter também integridade e autenticação. Integridade significa que é essencial que os dados recebidos sejam os mesmos que os enviados e não tenham sido acidentalmente ou maliciosamente modificados, alterados ou destruídos. Por sua vez, a autenticação significa ter a certeza absoluta de que o remetente é realmente quem diz ser.

O modelo de segurança do sistema está baseado em três dimensões, a saber: controlo de acesso, chaves de segurança e encriptação de mensagens.

O controlo de acesso tem a funcionalidade de definir quem pode ligar-se à infraestrutura de comunicação. Esta aceita apenas ligações de utilizadores fidedignos. Entende-se por utilizador fidedigno um cliente que se liga ao sistema com uma conta válida. Cada conta possui um nome de utilizador e uma chave de acesso. O nome de utilizador é público, mas a chave de acesso é privada e do conhecimento do utilizador. Este controlo de acesso serve quer para os pacientes quer para os familiares quer ainda para os profissionais de saúde. Na Figura 4.22 está representado o caso de uso do controlo de acesso. Quando um utilizador se pretende ligar à rede, o seu dispositivo (LS) pede os dados da sua conta de acesso e depois do seu preenchimento, o LS envia o pedido de conexão ao DS. O DS, no caso da conta existir, como se mostra na Figura 4.22 A), adiciona a nova ligação à rede e envia um comando de confirmação de volta ao utilizador. No caso da conta não existir, como se mostra na Figura 4.22 B), o DS envia um comando para cancelar a ligação.

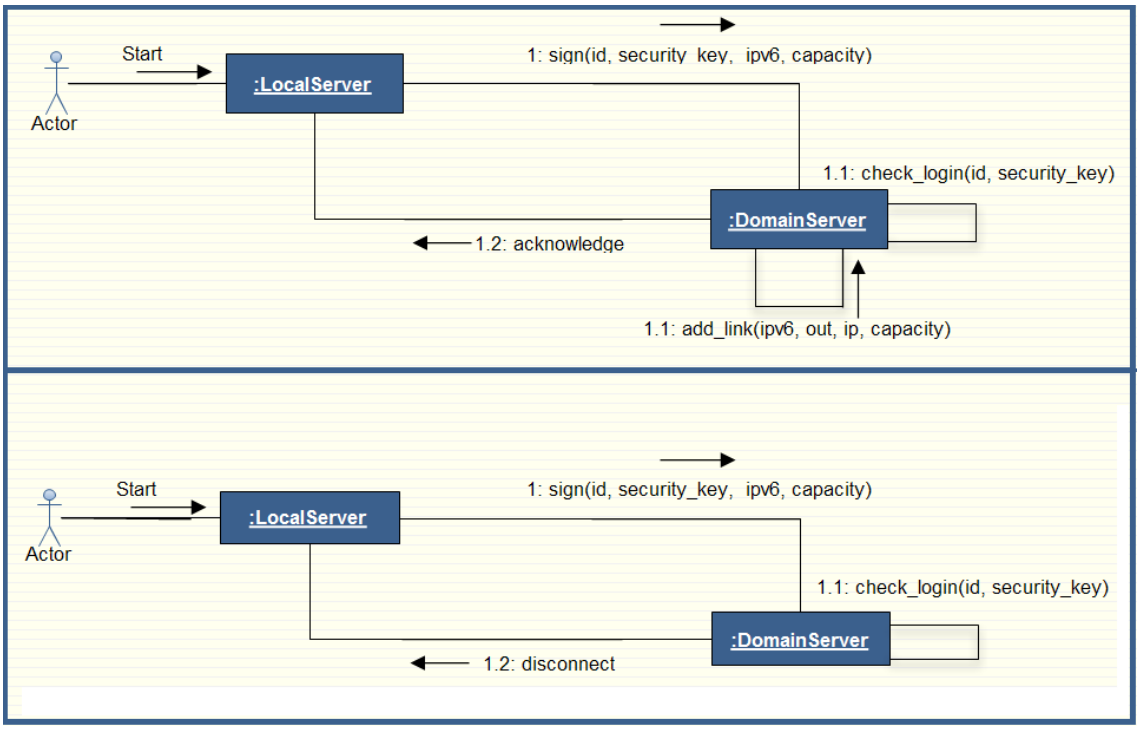

Figura 4.22 – Caso de uso do controlo de acesso à infraestrutura. A) *Login* correto B) *Login* incorreto

A ideia subjacente às chaves de segurança é garantir que somente pessoas devidamente autorizadas tenham acesso aos registos de saúde e isto é conseguido através do uso de chaves de segurança atribuídas a cada serviço. Quando um utilizador pretende subscrever um determinado serviço é necessário inserir o nome do serviço e a respetiva chave de segurança. Na Figura 4.23 está representado o caso de uso do controlo efetuado para a subscrição de serviços por parte de um utilizador. Quando um utilizador pretende subscrever um serviço ele necessita de inserir no seu dispositivo o nome do serviço que pretende e a sua chave de segurança. Depois, o seu dispositivo (LS) envia o pedido de subscrição ao DS. O DS, no caso da chave de segurança ser incorreta, responde com um comando de subscrição inválido, impedindo que o utilizador tenha acesso aos dados. No caso da chave ser a correta, o DS regista o novo *subscriber* na rede e responde com o comando de serviço criado.

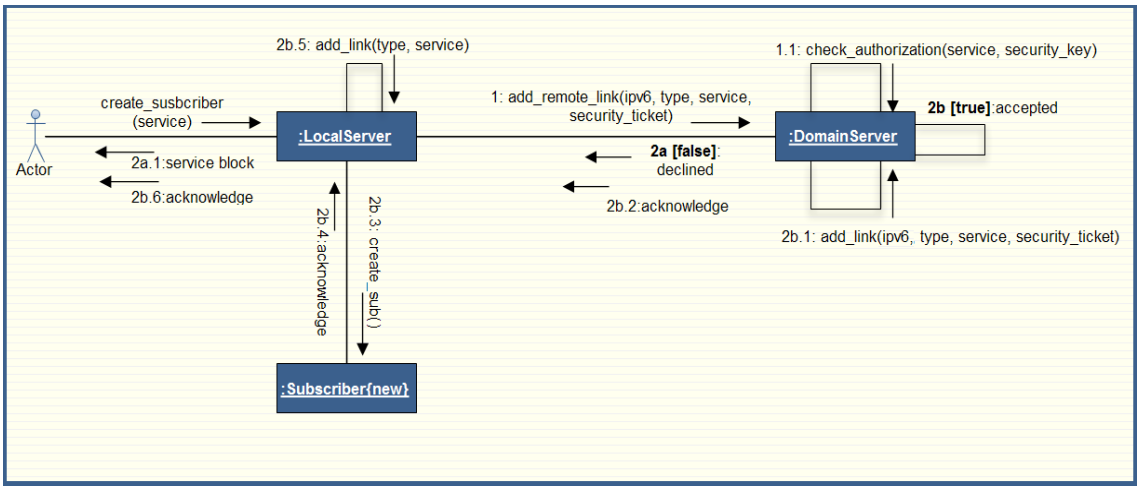

Figura 4.23 – Caso de uso do controlo na subscrição de serviços.

No que concerne à encriptação é importante referir que as trocas de mensagens entre os nós da infraestrutura de comunicação encontram-se encriptadas com um código de segurança. Para realizar a encriptação e a posterior desencriptação foi utilizado o *Advanced Encryption Standard* (AES) do Java. Na implementação do AES começa-se por definir um código de segurança de 128 bits. Neste padrão de encriptação os dados são convertidos em palavras de 16 bytes e colocados numa matriz 4x4. Em seguida, é utilizada a lógica XOR para realizar uma operação entre o código e os dados. Na etapa seguinte realiza-se a troca de bytes não linear de acordo com uma tabela de referência. Após esta troca não linear, procede-se à deslocação de cada fila um determinado número de posições. Por fim, os 4 bytes de cada coluna são combinados através de uma transformação linear.

Na Figura 4.24 é possível observar que antes do envio dos dados é feita a sua encriptação. Quando os dados chegam ao *subscriber* encontram-se alterados relativamente à mensagem enviada. No entanto, após a desencriptação, através da chave de segurança apropriada, a mensagem recebida é apresentada corretamente.

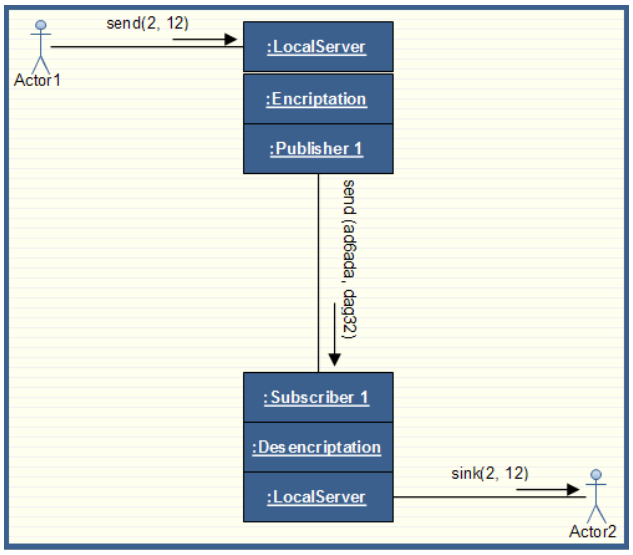

Figura 4.24 – Caso de uso da encriptação dos dados

Este sistema de autenticação está também baseado no endereço IPV6 implementado que identifica o dispositivo. Quando o tipo de tecnologia de comunicação do *subscriber* é compatível com a tecnologia de comunicação do dispositivo que está a publicar o serviço, o DS envia o endereço IPV6 do *subscriber* e o serviço pretendido ao *publisher* de modo que este apenas aceite a ligação que coincida com o endereço IPV6 do *subscriber*. Sendo assim, o *subscriber,* para estabelecer a ligação com o *publisher* do serviço desejado, necessita de confirmar a sua identidade. Depois de confirmada a identidade, o *publisher* começa a transmissão de dados para o *subscriber*. A Figura 4.25 representa o caso de uso deste sistema de autenticação.

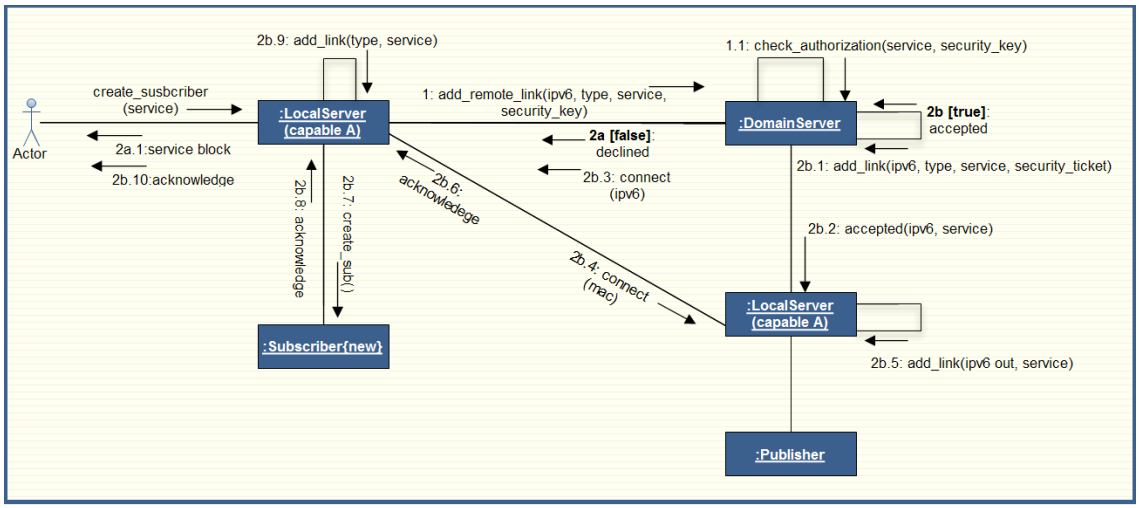

Figura 4.25 – Caso de uso na subscrição de serviços remotos

Este mecanismo de segurança serve para que esta infraestrutura possa restringir de forma segura o acesso a determinados serviços através de ligação direta ao LS, uma vez que se alguém tentar aceder de forma indevida a um serviço não conseguirá estabelecer uma ligação

válida ao sistema, pois o mecanismo de confirmação irá falhar e o utilizador não autenticado será desligado por ordem do *publisher*.

4.2.6 Interface Gráfica

Esta infraestrutura de comunicação fornece duas interfaces gráficas, baseadas em Java e em Android, que procuram ser de fácil utilização quer pelo pessoal médico quer pelos pacientes. Estas *interfaces* fornecem detalhes sobre o estado do paciente, permitem a subscrição de diferentes serviços e ainda a interação aos diversos sensores implementados na estrutura.

Como a utilização do *software* é restrita apenas a utilizadores autorizados, a janela inicial possui uma credencial de acesso ao *software*, como mostra a Figura 4.26. O utilizador terá de preencher os campos *login*, *password*, IPV6, IP/MAC e proceder ao *login*. No campo IPV6, o utilizador deve colocar o endereço IPV6 que pretende ser identificado pelo *domain server*. Este campo tem que ser único, uma vez que a repetição de IPV6 na mesma infraestrutura provoca a rejeição da ligação por parte do *domain*. No campo IP/MAC, o utilizador deve colocar o endereço do dispositivo a que se pretende ligar. Se algum campo for incorretamente preenchido ou as credenciais forem mal introduzidas uma mensagem de erro será dada ao utilizador.

Figura 4.26 – Janela Inicial. À esquerda Java. À direita Android

Após efetuada a autenticação, é aberta a janela principal da *interface*. Nesta janela pode-se executar uma série de tarefas, tais como:

- Visualizar graficamente os sinais adquiridos pelo paciente.
- Visualizar os valores da pressão arterial (pressão sistólica, pressão diastólica,

pressão arterial média e ritmo cardíaco).

- Ligar e desligar os sensores.
- Subscrever e cancelar serviços.
- Escolher o serviço a ser mostrado no gráfico.

Como é mostrado na Figura 4.27, o centro da janela é ocupado com o gráfico para se visualizarem os dados adquiridos pelos sensores. Na *interface* Java, no canto inferior esquerdo, é possível observar as caixas de texto onde são apresentados os valores provenientes da pressão arterial. Na *interface* Android, estes valores são representados no fim da janela. Em Java, os botões são mostrados na *interface* no canto inferior direito da janela. Em Android, os botões são mostrados quando se pressiona o botão menu. É importante referir que, quando um LS não tem capacidade para comunicar através da tecnologia Bluetooth, os botões dos sensores Bluetooth encontram-se desativados. Na *interface* Android, o botão Série não se encontra disponível uma vez que os *smartphones* não têm capacidade para utilizar esse protocolo de comunicação.

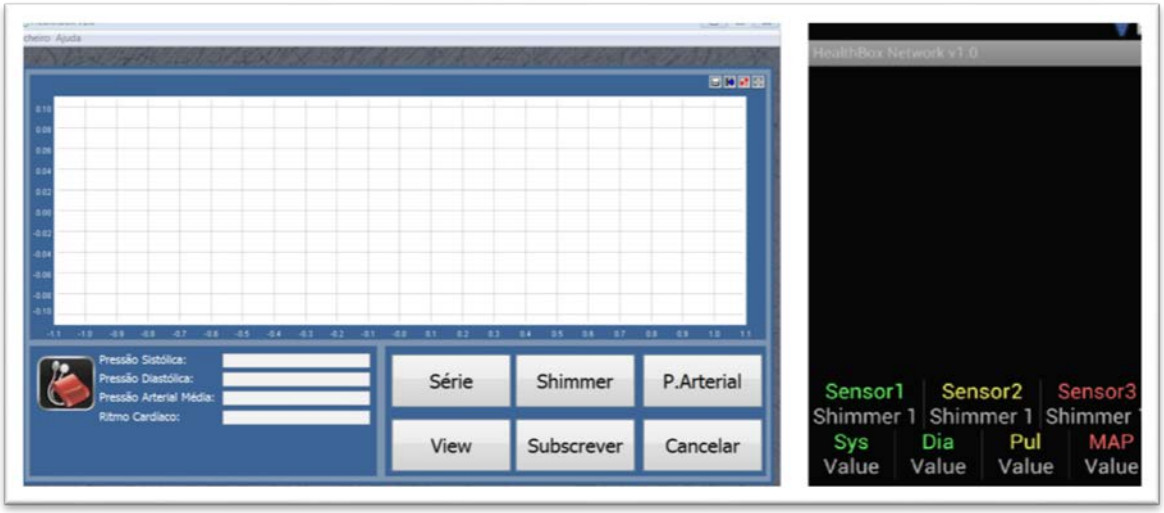

Figura 4.27 – Janela Principal. À esquerda Java. À direita Android

Uma janela adicional da *interface*, como se observa na Figura 4.28, funciona como *screensaver*. Esta janela é ativada após dez minutos de inatividade do utilizador.

Figura 4.28 – Screensaver da Interface Gráfica. À esquerda Java. À direita Android

É importante sublinhar que a aplicação em Android encontra-se *online* e pode ser transferida para qualquer *smartphone* e instalada [37].

4.3 Sumário

A arquitetura proposta para o sistema foi pensada tendo em conta a sua escalabilidade e adaptabilidade. Para o efeito, a infraestrutura de comunicação foi desenvolvida num sistema multicamadas, recorrendo-se ao padrão de comunicação *publisher/subscriber*

Nesta infraestrutura foi utilizada a variação *Type-Based publisher/subscriber* e foram implementados três diferentes sensores: sistema de aquisição, monitor A&D Blood Pressure UA-767PBT-C40 e a unidade Shimmer R2. Para a sua implementação foram desenvolvidos os protocolos de comunicação para a sua integração na infraestrutura.

O modelo de segurança desenvolvido no sistema contempla três dimensões: controlo de acesso, chaves de segurança e encriptação de mensagens.

A infraestrutura de comunicação fornece duas *interfaces* gráficas, implementadas em Java quer para um PC quer para um *tablet* (Android). Estas *interfaces* fornecem detalhes sobre o estado do paciente, permitem a subscrição de diferentes serviços e ainda a interação aos diversos sensores implementados na estrutura.

Capítulo 5 – Resultados

Neste capítulo são apresentados os cenários de testes efetuados à infraestrutura de comunicação desenvolvida, bem como a discussão dos mesmos tendo por base as características inicialmente pretendidas para o sistema.

5.1 Conceção dos cenários de teste

Normalmente, há o conceito de manter o foco linear, evidenciando a ideia de se manter um produto simples e funcional. O problema é que nem todas as pessoas encaixam realmente nos produtos desenvolvidos. Além disso, uma solução linear para todas as pessoas é insatisfatória, pois cada pessoa tem os seus problemas intrínsecos. Por isso, esta infraestrutura de comunicação tem como objetivo ser o mais flexível possível de modo a ser uma solução individual para cada utilizador e ainda ser adaptável a qualquer tipo de ambiente.

De modo a testar a flexibilidade do sistema foram criados dois cenários com pessoas virtuais que têm características transversais a um largo grupo de pessoas. Para o efeito, foram definidas duas pessoas com incidência nos seguintes parâmetros: historial, informação em saúde, conhecimentos tecnológicos, medos e frustrações e motivações. Estes parâmetros podem definir as necessidades atuais e futuras bem como as restrições que os utilizadores possam ter.

- José: 64 anos, casado, aposentado, dois filhos

Histórico: José é um professor aposentado e vive com a esposa, também aposentada, numa boa casa em Braga. Atualmente, vivem das suas reformas e não têm uma vida social ativa. Todas as vezes que precisa de ajuda chama um dos seus filhos, que também vivem em Braga.

Informação em Saúde: As principais preocupações são os problemas cardíacos e a hipertensão arterial que o obrigam a tomar medicamentos diariamente. Apresenta alguns problemas de memória e tem medo de se esquecer de tomar a medicação.

Conhecimento tecnológico: Tem um computador e um telemóvel que utiliza regularmente. Está disposto a aprender a usar a tecnologia, embora não confie inteiramente

nela.

Medos e frustrações: O seu principal temor é ter algum acidente vascular cerebral (AVC) ou enfarte do miocárdio que o impossibilite de pedir auxílio numa situação de emergência. Devido aos seus problemas cardiovasculares, receia que o sinal de emergência não seja ativado numa situação de ataque cardíaco. Acresce que a esposa, muitas vezes, tem ataques de pânico e numa situação de emergência receia que não o possa ajudar.

Motivações: Devido aos seus problemas cardíacos e de hipertensão, que o obrigam a deslocar-se à unidade de saúde com frequência e que lhe causam muito transtorno, desejaria ter um meio de contacto inteligente para suporte clínico capaz de usar em sua casa.

- Fátima: 59 anos de idade, viúva, aposentada, sem filhos

Histórico: Fátima vive sozinha desde que o seu marido morreu.

Informações de Saúde: foi-lhe diagnosticada um problema neuromuscular degenerativo. Ela tem dificuldade em mover os membros superiores e, por vezes, também os membros inferiores.

Conhecimento tecnológico: Nunca usou um computador na sua vida e só tem um telemóvel básico.

Medos e frustrações: Tem muito medo do futuro porque receia não saber lidar com o problema se a doença se agravar. Além disso, sente-se frustrada porque não pode desempenhar as tarefas domésticas.

Motivações: Quer estar preparada para o futuro. Sabe que as suas frustrações e fragilidades provavelmente vão piorar e ela quer ter a certeza de que pode lidar com isso da melhor maneira possível.

Estes cenários foram criados de modo a concretizar os seguintes objetivos:

- Cenário domiciliar: o paciente fica em casa e verifica continuamente o seu estado de saúde, garantindo-lhe a sua autonomia.
- Cenário Hospitalar: o paciente encontra-se na unidade de saúde e os profissionais de saúde fazem o diagnóstico médico ou monitorização no caso de internamento.

As pessoas criadas envolvem-se com os cenários da seguinte maneira:

- O José corresponde com o cenário domiciliar;
- A Fátima corresponde com o cenário hospitalar.

Na Figura 5.1 e na Figura 5.5 estão representados os cenários criados nesta dissertação. Cada cenário é especificamente construído sobre o conceito de produtos e serviços. A chave é ser capaz de fornecer recursos que permitem ter diferentes produtos para o utilizador, com a máxima compatibilidade entre eles.

5.1.1 Cenário Domiciliar

O cenário domiciliar consiste num sistema distribuído que permite a ligação entre a casa do paciente e a casa do familiar. A casa do paciente possui sensores com a tecnologia de comunicação Bluetooth (Shimmer R2 com ECG e o monitor de pressão arterial), dispositivos que subscrevem os serviços dos sensores e ainda um *domain server* (DS) que tem como principal função controlar todos os dispositivos na casa do paciente e encaminhar os dados para casa do familiar. A casa do familiar contém dispositivos que permitem a visualização dos serviços subscritos e ainda um *local server* (LS) que se encontra ligado ao DS na casa do paciente, via Ethernet.

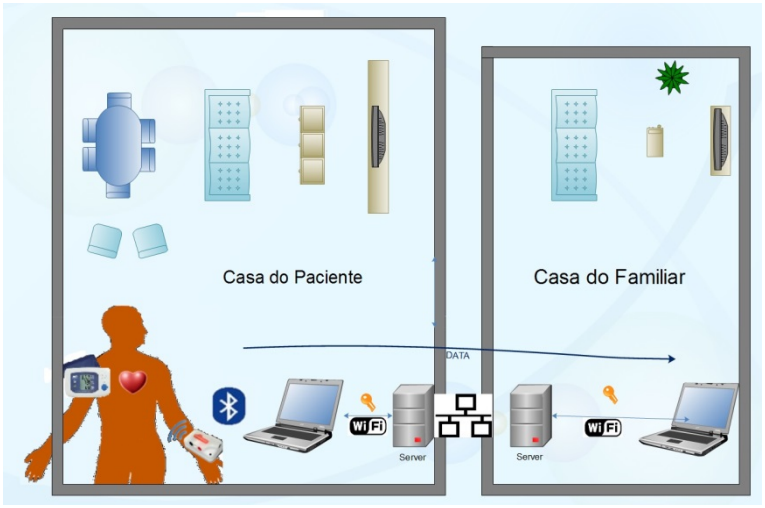

Figura 5.1 – Cenário Domiciliar

Neste cenário, o processo inicia-se com o registo do LS do paciente no seu próprio DS, através de uma ligação WiFi. Este LS é necessário para permitir a difusão dos dados enviados pelos sensores para o seu DS uma vez que os sensores são incapazes de se registarem diretamente num DS. Como os sensores são LS *ineffectives*, é responsabilidade do LS fazê-lo sempre que um sensor se conecte à infraestrutura. Assim sendo, o monitor de pressão arterial e

a Shimmer ligam-se ao LS do paciente através de uma ligação Bluetooth. A *interface* gráfica fornecida pelo LS permite a visualização dos dados recolhidos pelos sensores, o controlo do funcionamento dos sensores e ainda possibilita a sinalização de alertas.

Sempre que um familiar pretende subscrever algum serviço de acompanhamento (Figura 5.2), o dispositivo, que funciona como LS *ineffective* em sua casa, envia um pedido ao seu LS através da comunicação WiFi que, por sua vez, subscreve o serviço requerido ao DS em casa do paciente através de uma ligação Ethernet. Como já foi dito anteriormente, este serviço só pode ser requisitado através de autenticação, por razões de segurança de dados. Se a autentificação for incorrecta, o DS responde com uma mensagem de remoção do *subscriber*. Ao receber esta mensagem, o LS do familiar reencaminhada esta mensagem para o PC do familiar. Desta forma, os dados não serão enviados para o PC do familiar. Se a autentificação for aceite, o DS envia um comando de subscrição ao LS do paciente, permitindo-lhe o envio dos dados de pressão arterial.

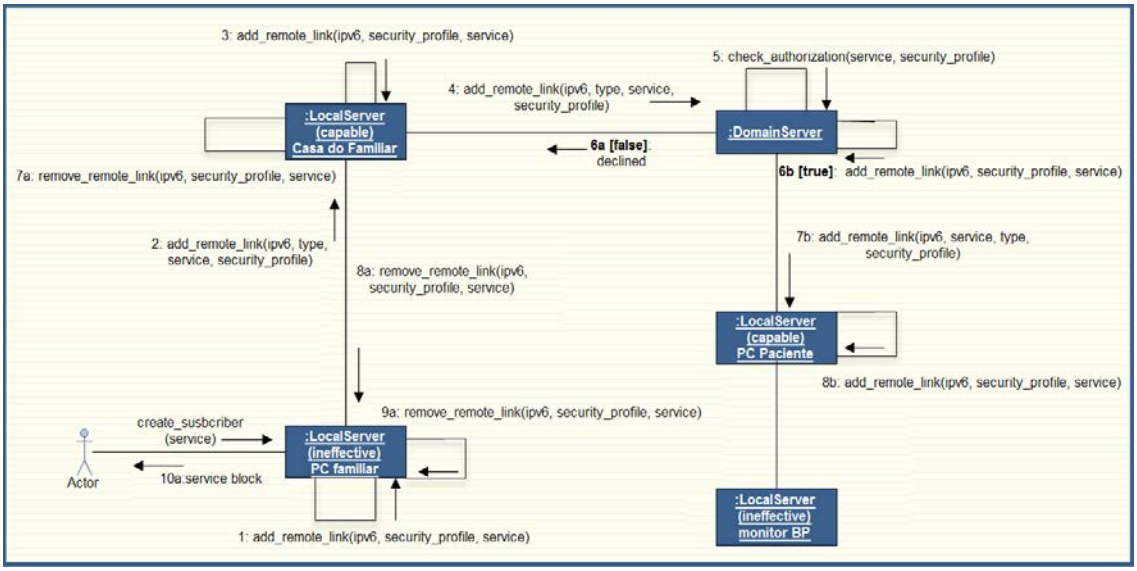

Figura 5.2 – Caso de uso para a subscrição do monitor de pressão arterial pelo familiar

Quando o José se monitoriza (Figura 5.3), o monitor de pressão arterial funciona como *publisher*, isto é, envia os dados para o LS de sua casa (PC do paciente). Consequentemente, o LS, desempenhando o padrão de *router* e *dealer*, envia para o seu DS, que neste caso funciona como *bridge*, e reencaminha os dados para o LS dos subscritores do serviço. Neste caso, o LS na casa do paciente funciona também como *bridge*, enviando os dados recolhidos para o dispositivo subscritor visualizar (PC do familiar).

No caso da unidade Shimmer, o processo de subscrição e posterior envio dos dados é praticamente idêntico ao monitor de pressão arterial. A única diferença reside no envio dos

dados. No caso da Shimmer, o envio dos dados é contínuo e com uma determinada frequência de amostragem. No monitor de pressão arterial são enviados somente quatro dados: pressão sistólica, pressão diastólica, ritmo cardíaco e pressão arterial média.

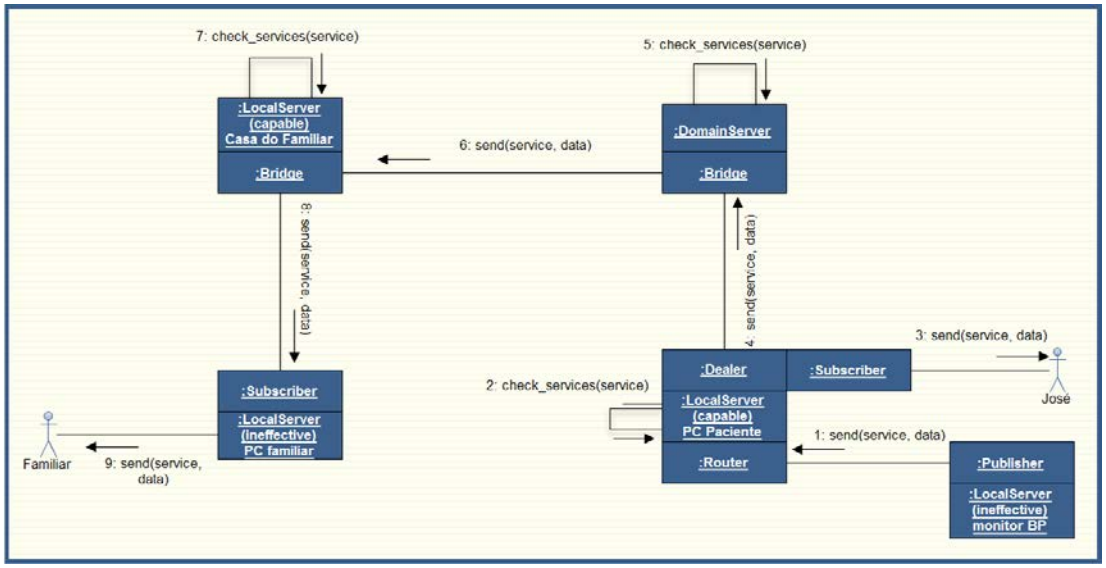

Figura 5.3 – Caso de uso no envio dos resultados dos sensores da casa do paciente para a casa do familiar

Este cenário é o ideal para o José uma vez que lhe permite controlar os seus próprios dados fisiológicos sem necessidade de se dirigir a uma unidade de saúde. Numa situação de emergência, o sistema emite um sinal de alerta para o seu filho que, através de aconselhamento médico, lhe presta rápida assistência ou o encaminha para uma unidade de saúde.

A Figura 5.4 mostra a visualização dos sinais fisiológicos do José - ECG e pressão arterial - no seu PC e no PC do seu filho.

Figura 5.4 – Apresentação dos sinais biológicos do José. À direita no seu dispositivo e à esquerda no dispositivo do seu filho

5.1.2 Cenário Hospitalar

O cenário hospitalar é constituído por um DS, pelos dispositivos LS's (que servem para visualizar/encaminhar os dados ou gerir sensores) e pelos sensores existentes no hospital, que são administrados pelos LS's, dado que eles são incapazes de se registarem autonomamente no DS.

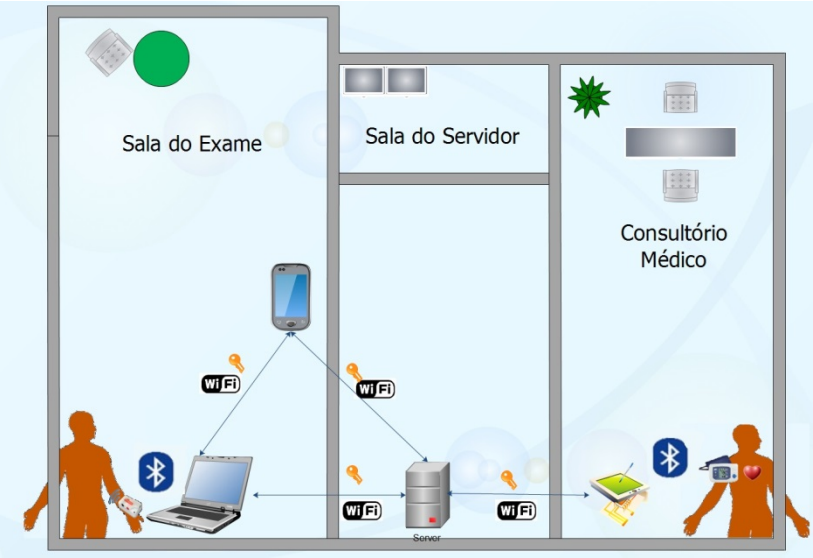

Figura 5.5 – Cenário Hospitalar

Quando é criado um novo nó na rede, este é registado no DS, via WiFi. Este nó pode ser um sensor médico ou um qualquer outro dispositivo. Tal como no cenário domiciliar, os sensores são geridos pelos LS's, através de uma ligação Bluetooth.

A partir do LS no consultório, o médico pode aceder diretamente aos sensores lá presentes assim como aos sensores presentes nas salas de exames, através do DS ou de outro LS. Esta ligação pode ser efetuada através de Ethernet ou Bluetooth dependendo da capacidade do LS. As salas de exames estão equipadas com dispositivos médicos ligados ao LS da respetiva sala por meio de Bluetooth. É também possível aceder a qualquer exame a partir de um dispositivo móvel (*smartphone*, PDA) em qualquer zona da unidade de saúde a partir de ligações WiFi e Bluetooth. Para o efeito, é necessário apenas a autenticação do serviço requerido. O DS é o responsável por fornecer o acesso ao serviço dando autorização para proceder à ligação entre os LS'*s*, no caso de tecnologias de comunicação idênticas (Ethernet/Ethernet ou Bluetooth/Bluetooth). No caso de tecnologias de comunicação diferentes (Ethernet/Bluetooth), o DS funciona como uma *bridge,* registando o serviço subscrito e fazendo o pedido ao LS, que publica os dados para os reencaminhar.

Este cenário é o ideal para a Fátima, uma vez que permite uma fácil manipulação dos dados por parte dos profissionais de saúde, proporcionando um diagnóstico precoce ou um ajuste da terapêutica de forma rápida e eficaz.

Neste cenário, a Fátima dirige-se ao consultório médico e realiza um exame de rotina para monitorizar a pressão arterial. O médico liga o monitor de pressão arterial e, após executar a medição, liga-se, via Bluetooth, ao seu dispositivo (PandaBoard ES) e envia os dados medidos. A Figura 5.6 mostra a apresentação do resultado da pressão arterial da Fátima no dispositivo médico que se encontra no seu consultório.

Figura 5.6 – Monitorização da Pressão Arterial. À direita apresentação do resultado no dispositivo do médico. À esquerda apresentação do resultado no sensor do médico

Quando o médico prescreve um novo exame de neurologia à Fátima esta desloca-se à sala de exames. O técnico ao executar o exame requerido coloca a unidade Shimmer na paciente e, através do seu PC, liga-se à unidade através de uma conexão Bluetooth que subscreve os dados do sensor EMG e do acelerómetro. Depois disto, a unidade Shimmer entra no estado *streaming* e inicia o envio de dados para o PC do técnico para futura visualização. A discussão dos resultados obtidos entre o técnico e o médico pode ser efetuada através do auxílio do seu *smartphone*. Isto é, o técnico, através do seu *smartphone*, pode subscrever os dados da unidade Shimmer por meio do DS. Quando o DS recebe o pedido de subscrição do *smartphone* este avalia a compatibilidade das tecnologias de comunicação dos dispositivos (WiFi-WiFi). Neste caso, em que existe compatibilidade, o DS envia uma mensagem para *smartphone* e outra para o PC com os endereços IPV6's de cada um de modo a realizar uma ligação segura. A Figura 5.7 representa o caso de uso da subscrição do *smartphone* do técnico ao exame de neurologia da Fátima.

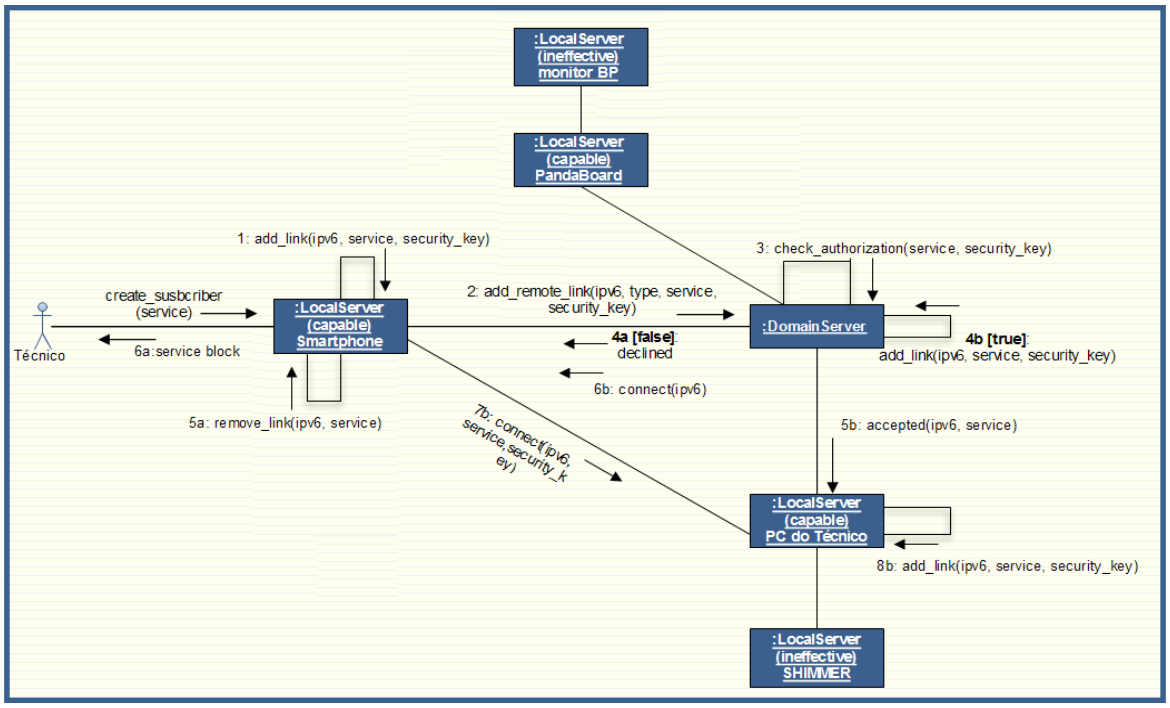

Figura 5.7 – Caso de uso para a subscrição remota do *smartphone* do técnico ao serviço EMG na unidade Shimmer

Após a conexão entre o PC e o *smartphone*, o PC reencaminha dos dados da unidade Shimmer para o *smartphone.* Deste modo, o técnico pode discutir os resultados obtidos com o médico em qualquer ponto do centro hospitalar. Neste caso, o PC desempenha o padrão de *router* e de *dealer*. A Figura 5.8 representa o caso de uso do envio dos dados da unidade Shimmer para os seus subscritores.

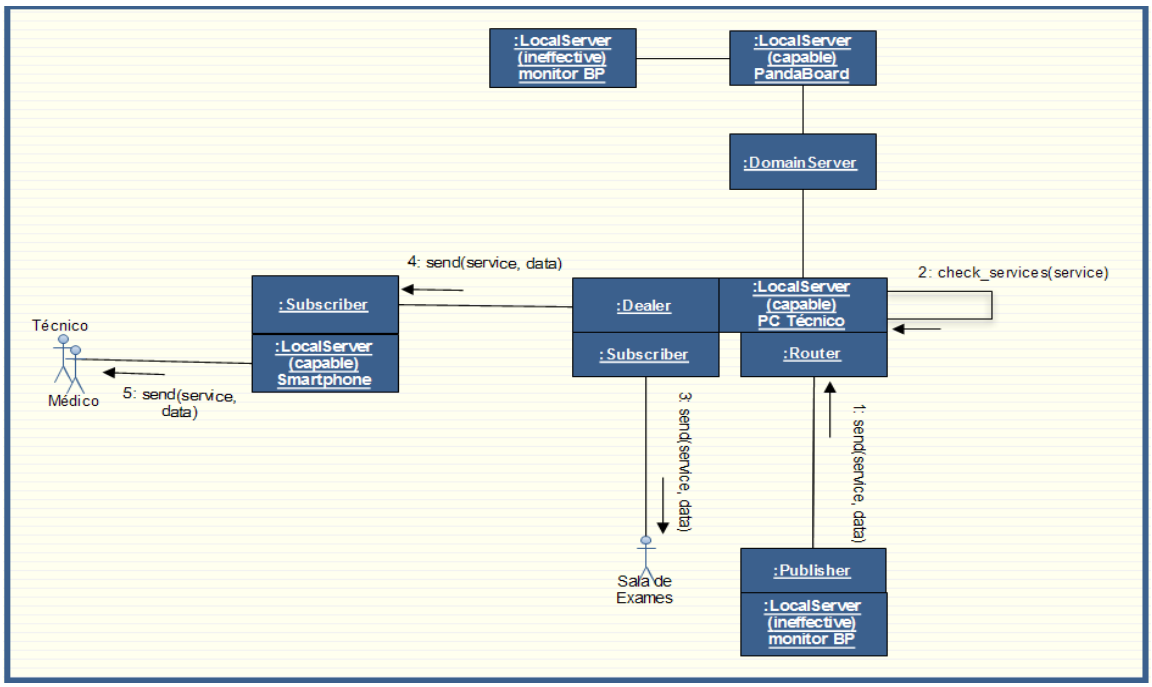

Figura 5.8 – Caso de uso para o envio dos dados da Shimmer para os *subscribers* da infraestrutura

O médico, no seu consultório, tem ainda a possibilidade de visualizar os resultados obtidos. Tal como no *smartphone* do técnico, o DS é o nó fundamental para orientar a ligação entre o dipositivo do médico e o PC na sala de exames.

A Figura 5.9 mostra o exame de EMG/acelerómetro realizado à Fátima. Na Figura 5.9 B) encontra-se representado o sinal do acelerómetro no PC do técnico, na sala de exames. Na Figura 5.9 C) é representado o sinal do acelerómetro no *smartphone* do técnico. Na Figura 5.9 D) é representado o sinal do acelerómetro no dispositivo do consultório médico.

Figura 5.9 – Apresentação do resultado do acelerómetro. A) montagem do sensor na Fátima B) apresentação do resultado no PC do técnico C) apresentação dos resultados no *smartphone* do técnico D) apresentação do resultado no dispositivo do médico

5.2 Discussão dos resultados

Com a validação destes cenários pode-se afirmar que esta infraestrutura de comunicação é flexível e adapta-se a diferentes circunstâncias. Todos os esforços de desenvolvimentos tecnológicos atuais em direção a uma arquitetura uniforme não foram bem-sucedidos, mas esta infraestrutura de comunicação parece ter potencial para adquirir esta uniformidade.

A infraestrutura de comunicação desenvolvida, devido à sua flexibilidade e adaptabilidade, é adequada para ser implementada como uma solução de e-Health que pode contribuir para proporcionar uma melhoria da qualidade de vida aos idosos e pessoas deficientes. Todavia, é importante salientar que, para uma melhor validação desta infraestrutura, deviam ter sido realizados testes relativos aos tempos de resposta e usabilidade.

Nos cenários AAL apresentados, os serviços e produtos para prestação de cuidados em casa dos pacientes e na unidade de saúde estão mais perto de satisfazer as necessidades dos

utilizadores. Todos os cenários criados para a validação da infraestrutura de comunicação tiveram respostas positivas como era esperado. No entanto, estes cenários não foram testados em ambientes reais, ou seja, os testes foram realizados em ambientes de teste controlado. Sendo assim, para uma melhor validação do sistema seria necessário integrá-lo em algumas casas de pacientes e em hospitais de modo a obter resultados mais fidedignos e consistentes.

Esta infraestrutura de comunicação contém procedimentos de segurança que protegem os dados sensíveis que fluem no sistema. Estes recursos são muito importantes porque a maioria dos dispositivos transmite informação muito sensível, que, de outra forma, qualquer pessoa podia ser capaz de capturá-los. As mensagens trocadas entre os diversos dispositivos estão encriptadas e só um dispositivo com o código de desencriptação é que recebe corretamente as mensagens.

No que diz respeito às tecnologias de comunicação desenvolvidas, verifica-se que todos os protocolos de comunicação foram bem implementados. Todos os comandos são executados corretamente e os sinais adquiridos pelos sensores são representados com boa qualidade. No entanto, o protocolo de comunicação com o sensor de pressão arterial pode vir a ser otimizado, uma vez que quando os dados enviados por ele não são corretamente recebidos, o sensor desliga-se e guarda o valor na sua memória. O ideal seria que este enviasse os dados novamente para o mesmo dispositivo até ter a indicação de que os dados foram recebidos corretamente.

Através de testes efetuados, acredita-se que a camada cliente/servidor é suficiente para a maioria dos casos de interesse. Geralmente, a aquisição de dados de vários pacientes não é útil (um médico não quer saber a pressão arterial de todos os pacientes na unidade de saúde e um paciente, em sua casa, não necessita de obter o sinal em todos os seus dispositivos). A estrutura permite a receção de dados de dois sensores e também o envio para três dispositivos sem qualquer problema no sistema.

Sublinhe-se que os diferentes projetos abordados no Capítulo 2 – Estado da Arte, até agora são encaminhados para resolver os problemas de grupos orientados para certos objetivos. Mas esta infraestrutura de comunicação foi desenhada para obter uma grande flexibilidade e adaptabilidade a diferentes ambientes e pacientes. Porém, não há ainda um protótipo que satisfaça inteiramente o conceito de um ambiente inteligente, seguro e flexível.

Há, porém, alguns aspetos que podem otimizar esta infraestrutura de comunicação. Nos

parágrafos seguintes referem-se alguns desenvolvimentos que poderão melhorá-la.

A implementação desta infraestrutura de comunicação é muito simplificada pelo uso da camada *publish/susbscribe.* No entanto, esta camada resulta em alguma ineficiência uma vez que com o aumento do número de clientes para um servidor poderá provocar um grande atraso na entrega dos dados e até, por vezes, o servidor entrar em *overflow* e desligar-se. Uma possível otimização seria o envio de dados através de *broadcasting*, que consiste na transmissão do fluxo de dados para muitos recetores ao mesmo tempo e que garantisse que a informação seria aceite unicamente pelo dispositivo pré-endereçado nessa informação.

A infraestrutura de comunicação desenvolvida tem o inconveniente de não ter um modo de procura de nós na rede. Ou seja, o utilizador necessita de saber o número de IP ou o endereço *MAC* (Media Access Control) do dispositivo que se pretende ligar. Ora, para pacientes idosos ou com problemas de memória ou até para os profissionais de saúde, isto é grave, porque estes podem-se esquecer do endereço do dispositivo a que se pretendem ligar e, assim sendo, a infraestrutura fica inutilizada. Uma forma de resolver este inconveniente seria a criação de um algoritmo de pesquisa de dispositivos na rede, que permitisse oferecer ao utilizador uma lista dos dispositivos disponíveis sem que este necessitasse de ter o conhecimento dos endereços dos nós da rede.

Muito embora todos os protocolos de comunicação tenham sido bem implementados, a integração da tecnologia de comunicação USB seria uma mais-valia para tornar esta infraestrutura ainda mais completa. Para o efeito, seria necessário criar um adaptador no *software* que implementasse o seu protocolo.

Um outro ponto importante a referir tem a ver com o código de autenticação do serviço. Para a subscrição de um serviço é necessário enviar para o *domain server* a *password* correta. Neste aspeto torna-se necessário aumentar a segurança na ligação entre o sensor e o seu *local server*. Para isto seria necessário desenvolver o protocolo de comunicação dos sensores de modo a encriptar as mensagens. Outra solução de segurança a ser implementada seria realizar um controlo baseado em funções (médicos, enfermeiras, farmacêutico, familiar). Assim, seria possível definir permissões de leitura e/ou escrita na rede.

Por último, a expetativa desta infraestrutura de comunicação é apresentar uma solução integrada duradoura, de baixo custo e viável. Com isto pretende-se dizer que a solução deve ser

facilmente adotada pelos utilizadores e qualquer empresa de desenvolvimento possa construir dispositivos e serviços que cumpram os padrões utilizados. Aqui apresentam-se os padrões e arquiteturas atuais que o projeto usa, no entanto, há espaço para a expansão deste projeto a uma utilização global, através de um serviço integrado numa plataforma tecnológica mais alargada tal como os serviços digitais de televisão, internet e telefone.

5.3 Sumário

Para testar a flexibilidade do sistema foram criados dois cenários com pessoas virtuais com incidência nos seguintes parâmetros: historial de saúde e conhecimentos tecnológicos. A validação destes cenários permite concluir que esta infraestrutura de comunicação é flexível e adapta-se a diferentes circunstâncias, podendo ser implementada como uma solução de *e-Health*. No entanto, para uma melhor validação desta infraestrutura, deviam ter sido realizados testes relativos aos tempos de resposta e usabilidade.

Esta infraestrutura de comunicação contém procedimentos de segurança que protegem os dados sensíveis que fluem no sistema, porque as mensagens trocadas entre os diversos dispositivos estão encriptadas. Os testes mostram também que todos os protocolos de comunicação foram bem implementados, uma vez que todos os comandos são executados corretamente e os sinais adquiridos pelos sensores são representados com boa qualidade.

Capítulo 6 – Conclusões

Este capítulo sintetiza as conclusões desta dissertação e simultaneamente apresenta o potencial futuro relacionado com os AAL.

6.1 Conclusão

A investigação e o desenvolvimento de AAL são importantes porque podem contribuir de forma decisiva para ajudar a aliviar as consequências socioeconómicas e os encargos financeiros, que resultam do crescente envelhecimento da população em Portugal e em países mais desenvolvidos.

Dada a importância de abordar maneiras de fornecer cuidados de saúde inteligentes para os idosos e doentes crónicos, os investigadores começaram a explorar soluções tecnológicas para melhorar a saúde e a prestação de cuidados de saúde de forma complementar aos serviços existentes. Atualmente, existe um vasto número de projetos de investigação nesta área. No entanto, estes apresentam algumas limitações que devem ser resolvidas, nomeadamente a falta de segurança e a limitação da sua adaptabilidade à heterogeneidade dos meios de transmissão, bem como incorporar dispositivos que não foram projetados para fazerem parte de um sistema distribuído integrado.

O objetivo desta dissertação foi o desenvolvimento de uma infraestrutura de comunicação que permitisse a integração de tecnologias já existentes ao nível da recolha e processamento de informação relacionada com os dados vitais dos seus utilizadores. Além disso, uma ideia que norteou este projeto foi o desenvolvimento de uma infraestrutura de comunicação que reunisse simultaneamente três características:

- Flexibilidade – poder ser integrada e ajustada a diferentes ambientes (institucional, público ou privado).

- Integridade – que os dados recebidos sejam os mesmos que os enviados e não sejam acidentalmente ou maliciosamente modificados, alterados ou destruídos.

- Autenticação – ter a certeza de que o remetente é realmente quem diz ser.

Relativamente à implementação da infraestrutura, esta envolveu o desenvolvimento do *software* orientado a objetos, implementação de uma *interface* gráfica para o utilizador, em Java para um computador pessoal e para o sistema operativo Android, tendo-se cumprido os objetivos inicialmente propostos.

Se é verdade que a prestação de serviços de saúde envolve a interação direta e íntima entre pacientes e os seus prestadores de serviço, que não pode ser substituído por tecnologia, é também verdade que os cuidados centrados no paciente são cada vez mais melhorados após a introdução de novas tecnologias na indústria dos cuidados de saúde. Além disso, as constantes inovações tecnologias nesta área parecem aumentar esta tendência.

6.2 Trabalhos Futuros

Nesta dissertação, desenvolveu-se uma infraestrutura de comunicação que poderá contribuir para melhorar a qualidade de vida das pessoas idosas e doentes crónicos, sem saírem das suas casas, através do recurso às tecnologias de comunicação (Bluetooth, Ethernet, RS-232), bem como a qualidade de serviço nos centros de saúde e hospitais. No entanto, ainda é possível a sua melhoria.

Uma insuficiência importante da atual infraestrutura de comunicação é a incapacidade da procura de dispositivos da rede. A integração de um sistema de procura de dispositivos seria uma mais-valia para esta infraestrutura, uma vez que o utilizador não necessitaria de saber o endereço do dispositivo a que se pretende ligar. Através deste sistema de procura seria dado ao utilizador uma lista dos dispositivos ao qual ele se podia ligar, facilitando ainda mais utilização do sistema.

Ainda na perspetiva de trabalho futuro, a integração de uma base de dados seria também uma mais-valia para o projeto, uma vez que permitiria guardar toda a informação do paciente, criando assim um histórico de saúde. Além disso, a base de dados permitiria também um controlo de acesso à rede baseado em funções (médicos, enfermeiras, farmacêutico, familiar). Assim, seria possível definir permissões de leitura e/ou escrita na rede.

Referências Bibliográficas

[1] T. Rehrl, R. Troncy, A. Bley, S. Ihsen, K. Scheibl, W. Schneider, S. Geldne, S. Goetze, J. Kessler, C. Hintermueller e F. Wallhoff, *The Ambient Adaptable Living Assistant is Meeting Its Users,* France: ALIAS, 2012.

[2] Eurostat, "Living conditions in Europe," Statistical booklet, Europe, 2008.

[3] DGO/MF, "Conta Geral do Estado," DGO/MF, Portugal, 2013.

[4] E. Jovanov, A. Milenkovic, C. Otto e P. de Groen, "A Wireless Body Area Network of Intelligent Motion Sensors for Computer Assisted Physical Rehabilitation," *Journal of NeuroEngineering and Rehabilitation,* vol. 2, n.º 6, pp. 2-6, 2005.

[5] E. Jovanov, A. Milenkovic, C. Otto, P. de Groen, B. Johnson, S. Warren e G. Taibi, "A WBAN System for Ambulatory Monitoring of Physical Activity and Health Status: Applications and Challenges," em *Proceedings of the 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Shanghai, China, 2005.

[6] A. S. Tanenbaum e M. Steen, Sistemas Distribuídos Princípios e Paradigmas, São Paulo: Pearson, 2007.

[7] G. Couloutis, J. Dollimore e T. Kindberg, Sistemas Distribuídos - Conceitos e Projeto, Porto Alegre: Bookman, 2007.

[8] P. Mendes, C. Figueiredo, M. Fernandes e Ó. Gama, "Electronics in Medicine," em *Handbook of Medical Technology*, Berlin Heidelberg, Springer, 2011, pp. 1337-1376.

[9] C. Ipektisidis, "e-health in the European Union: an assessment of its socio-political foundations," Maastricht University, Maastricht, 2005.

[10] W. Zagler, P. Panek e M. Rauhala, "Ambient Assisted Living Systems - The Conflicts between Technology, Acceptance, Ethics and Privacy," em *Dagstuhl Seminar Proceedings*, Austria, 2007.

[11] N. Noury, New Trends in Health Smart Homes, France: Healthcom '03, 2003, pp. 118-127.

[12] I. C. BioTelemetry, "CardioNet," [Online]. Available: https://www.cardionet.com/. [Acedido em 1 11 2013].

[13] P. Electro, "Polar Electro," [Online]. Available: http://www.polar.com/en. [Acedido em 1 Novembro 2013].

[14] I. BodyMedia. [Online]. Available: http://www.bodymedia.com. [Acedido em 2 Novembro 2013].

[15] V. Shnayder, B. Chen, L. K., T. Fulford-Jones e M. Welsh, "Sensor Networks for Medical Care," Harvard University Technical Report, Harvard, 2005.

[16] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, S. Moulton e M. Welsh, "Sensor Networks for Emergency Response: Challenges and Opportunities," IEEE CS, Harvard, 2004.

[17] H. S. N. Lab., "CodeBlue: Wireless Sensors for Medical Care," Harvard University, Cambrigde, 2011.

[18] R. De Vaul, M. Sung, J. Gips e J. Pentland, "MIThril 2003: Applications and Architecture," Massachusetts Institute of Technology, 2003.

[19] S. Pentland, "Healthwear: Medical Technology Becomes Wearable," IEEE Computer Society, 2004.

[20] N. Oliver e F. Flores-Mangas, "HealthGear: A Real-time Wearable System for Monitoring and Analyzing Physiological Signals," MSR Technical Report, Redmond - Washington, 2005.

[21] U. Anliker, J. Ward, P. Lukowicz, G. Troster, F. Dolveck, M. Baer, F. Keita, E. Schenker, F. Catarsi, L. Coluccini, A. Belardinelli, S. D., M. Alon, R. Schmid e M. Vuskovic, "AMON: A Wearable Multiparameter Medical Monitoring and Alert System," *IEEE Transactions on Information Technology in Biomedicine,* vol. 8, n.º 4, pp. 415-427, 2004.

[22] A. Wood, J. Stankovic, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang e R. Stoleru, "Context-Aware Wireless Sensor Networks for Assisted-Living and Residential Monitoring," Department of Computer Science, University of Virginia, Virginia, 2008.

[23] A. Van Halteren, R. Bults, K. Wac, D. Konstantas, I. Widya, N. Dokovsky, G. Koprinkov, V. Jones e R. Herzog, "Mobile Patient Monitoring: The MobiHealth System," *The Journal on Information Technology in Healthcare,* vol. 2, n.º 5, pp. 365-373, 2004.

[24] I. Intellicare, "A sua saúde sempre acompanhada: O seu cuidador a olhar por si, à distância," ISA, Coimbra, 2011.

[25] I. Intellicare, "ISA Intellicare apresenta na TV uma solução inovadora para acompanhar sinais vitais no domicílio.," ISA Intellicare, Coimbra, 2011.

[26] S. W., Wireless Communications and Networks, New Jersey: Pearson Education, Inc, 2005.

[27] L. Parziale, D. Britt, J. Forrester, W. Liu, C. Matthews e N. Rosselot, TCP/IP Tutorial and Techinal Overview, New York: Redbooks, 2006.

[28] W. Stallings, The TCP/IP Communications Architecture, New Jersey: Prentice Hall, 2000.

[29] R. Mettala, "Bluetooth Protocol Architecture," SIG, 1999.

[30] J. Kardach, M. C. Group e I. Corporation, "Bluetooth Architecture Overview," *Intel Technology*

Journal, n.º Q2, 2000.

[31] P. McDermott-Wells, "Bluetooth Overview," *Potentials, IEEE,* vol. 23, n.º 5, pp. 33-35, 2004.

[32] D. Djonin e J. Xhu, "The Bluetooth System," Victoria, 2001.

[33] L. C. M. inc., "RS-232: Serial Ports," 2002.

[34] P. P. e M. B., "Serial Communication by Using UART," Rourkela, 2000.

[35] N. Laddha e A. Thakare, "Implementation of serial communication using UART with configurable baud rate," *Internacional Journal on Recent and Innovation Trends in Computing and Communication,* vol. 1, n.º 4, pp. 263-268, 2013.

[36] M. T. Inc., "PIC32MX795F512L," Microchip Technology Inc., [Online]. Available: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en545660. [Acedido em 20 4 2013].

[37] R. Pereira. [Online]. Available: http://healthbox.store.aptoide.com/app/market/dynamic.example/1/4502868/Infraestrutura%20de%20Comunica%C3%A7%C3%B5es%20para%20Suporte%20a%20Ambientes%20Inteligentes%20de%20Apoio%20%C3%A0%20Sa%C3%BAde. [Acedido em 9 12 2013].

Anexos

Esta secção encontra-se dividida em três setores. No anexo A faz-se uma descrição de todas as funções e procedimentos contidos no *software* desenvolvido para esta infraestrutura de comunicações. O anexo B contém o diagrama de classe do *software* implementado. Por último, o anexo C apresenta a Pandaboard ES.

Anexo A. API do Software

Neste anexo são apresentadas e explicadas todas as classes do *software* da infraestrutura de comunicações bem como todos os seus métodos.

Classe BPManager – Classe que gera a ligação entre um *local server* (LS) *capable* e o monitor de pressão arterial. Implementa a classe IProcessCommand e estende a classe Thread, oferecida pelo Java.

Construtor	Descrição
BPManager (IAdapter iadapter)	Constrói um BPManager que recebe o adaptador da ligação.

Tipo	Método e Descrição
boolean	**process (Command command, String ip)** Processa os comandos recebidos.
synchronized void	**run()** Responsável por aceitar a ligação com o monitor de pressão arterial e receber os dados

Classe BTAdapter – Classe responsável por realizar a ligação Bluetooth com um LS ou um *domain server* (DS) em Android. Implementa a classe IAdapter.

Construtor	Descrição
BTAdapter (String endereço,	Constrói o adaptador Bluetooth. Recebe como

String uuid, String ipv6)	parâmetros do endereço e UUID do dispositivo que é necessário ligar-se e o próprio ipv6.

Tipo	Método e Descrição
String	**accept ()** Método Não Suportado
void	**close()** Encerra a ligação.
void	**destroy ()** Método Não Suportado
String	**getByInetAddress ()** Retorna o próprio ipv6.
InputStream	**getInputStream ()** Responsável por construir o fluxo de dados de entrada
String	**getLocalAddress ()** Retorna o seu próprio endereço.
OutputStream	**getOutputStream ()** Responsável por construir o fluxo de dados de saída
String	**getRemoteAddress ()** Retorna o endereço do dispositivo ao qual se encontra ligado.
boolean	**isClosed()** Verifica se a ligação se encontra fechada.

Classe Capable – Responsável pela construção dos adaptadores de cada nó da infraestrutura de comunicações.

Construtor	Descrição
Capable (String ip, int port,	Constrói os adaptadores de Ethernet e Bluetooth para

String uuid, String endereco, int port_bt, String ipv6)	Java.
Capable (String ip, int port, String uuid, String endereco, String ipv6)	Constrói os adaptadores de Ethernet e Bluetooth para Android.
Capable (String uuid, int port)	Constrói os adaptadores do DS para Java.
Capable (int port, String uuid)	Constrói os adaptadores do DS para Android.

Tipo	Método e Descrição
static int	**compare_capable (String sub, String pub)** Responsável por comparar a capacidade de diferentes LS's

Classe ClientManager – Classe responsável por gerir as ligações do LS na circunstância de cliente. Implementa a classe IProcessCommand e estende a classe Thread, oferecida pelo Java.

Construtor	Descrição
ClientManager (IAdapter iadapter, Command command, IDataSink idatasink)	Constrói um ClientManager que recebe o adaptador da ligação, o comando de início da ligação e a classe para a visualização dos resultados.

Tipo	Método e Descrição
static boolean	**create_publisher (String security_profile, String service)** Cria um novo serviço para publicação de dados do serviço escolhido.
static boolean	**create_subscriber (String security_profile, String service)**

	Cria um novo *subscriber* do serviço escolhido.
static boolean	**disconnect ()** Desliga o LS.
void	**insert_accept_links()** Guarda os IP's das ligações que pode aceitar para efetuar ligação.
boolean	**process (Command command, String ip)** Processa os comandos recebidos.
synchronized void	**Run ()** Constrói o fluxo de dados de saída e do LinkBuilder ().
static boolean	**Send (String service, List<Double> data)** Envia a lista de dados do determinado serviço para os *subscriber's* correspondentes.
static boolean	**unpublisher (String service)** Remove o *publisher* do serviço escolhido.
static boolean	**unsubscribe (String service)** Remove o *subscriber* do serviço escolhido.

Classe Command – Responsável pela construção de comandos para a troca de mensagens entre os diversos dispositivos.

Construtor	Descrição
Command (int type, List<byte[]> listdata, string service)	Envia dados para dipositivos série. Recebe o tipo, a lista de dados e o serviço como parâmetros.
Command (int type, List<Double>data, String service)	Comando para enviar os dados. Recebe o tipo, a lista de dados e o serviço como parâmetros.
Command (int type, String ipv6, List<Double> data, String service)	Construção do comando dos dados recebidos do monitor de pressão arterial. Recebe o tipo, o ipv6 do monitor, a lista de dados e o serviço.
Command (int type, byte[] ack)	Mensagem de *acknowledge* para o monitor de pressão

	arterial. Recebe o tipo e os bytes de *acknowledge*.
Command (int type, String service, String security_profile, String ipv6, int estrutura)	Construção do comando para adicionar ou remover ligações. Recebe o tipo, o serviço, a *password* do serviço, o ipv6 e o tipo de ligação (*subscribe* ou *publisher*).
Command (int type, String ipv6, String service, int estrutura)	Comando para remover o subscribe. Recebe o tipo, o serviço, o ipv6 e o tipo de ligação (*subscribe* ou *publisher*).
Command (int type, String ipv6)	Comando para desligar o dispositivo. Recebe o tipo e o ipv6.
Command (int type, String ipv6, int capable)	Comando de *acknowledge* do DS para os LS.
Command (int type, String security_key, String id, int capable, String ipv6)	Comando para iniciar a ligação. Recebe como parâmetros o tipo, o id do utilizador e a respetiva *password*, a capacidade do LS e o ipv6.
Command (int type, String ipv6, String ip, String service, String security_profile)	Comando para permitir a ligação entre LS. Recebe o tipo, o ipv6 remoto, o ip remoto, o serviço para subscrição e a respetiva *password*.

Classe DomainServer – Classe responsável por iniciar o DomainServerManager.

Construtor	Descrição
DomainServer (Capable adapteres)	Constrói um DomainServer que recebe os diversos adaptadores que suporta.

Tipo	Método e Descrição
static boolean	**disconnect ()** Requisita ao DomainServerManager para desligar o DS.

Classe DomainServerManager – Classe responsável por gerir as ligações do DS.

Implementa a classe IProcessCommand e estende a classe Thread oferecida pelo Java.

Construtor	Descrição
DomainServerManager (IAdapter iadapter)	Constrói um DomainServerManager que recebe o adaptador da ligação.

Tipo	Método e Descrição
void	**process_capable (String sub, String pub, String service, String security_profile, int compatibility)** Processa a compatibilidade de capacidades entre dois LS. Envia as mensagens a cada LS de acordo com a compatibilidade.
boolean	**process (Command command, String ip)** Processa os comandos recebidos
synchronized void	**run()** Aceita as ligações. Constrói o fluxo de dados de saída e o LinkBuilder().

Classe IAdapter – *Interface* de todos os adaptadores de comunicação.

Construtor	Descrição
IAdapter ()	*Interface* dos adaptadores.

Tipo	Método e Descrição
String	**accept ()** Aceita novas ligações e retorna o IP do dispositivo remoto.
void	**close()** Encerra a ligação.
void	**destroy ()**

	Destrói o adaptador.
String	**getByInetAddress ()** Retorna o próprio ipv6.
InputStream	**getInputStream ()** Responsável por construir o fluxo de dados de entrada
String	**getLocalAddress ()** Retorna o seu próprio endereço.
OutputStream	**getOutputStream ()** Responsável por construir o fluxo de dados de saída
String	**getRemoteAddress ()** Retorna o endereço do dispositivo ao qual se encontra ligado.
boolean	**isClosed ()** Verifica se a ligação se encontra fechada.

Classe IDataSink – *Interface* para a classe de visualização dos resultados.

Construtor	Descrição
IDataSink ()	*Interface* dos adaptadores

Tipo	Método e Descrição
void	**sink (List<Double> data, String service)** Recebe o serviço e a respetiva data.

Classe IProcessCommand – *Interface* para o processamento dos comandos

Construtor	Descrição
IProcessCommand ()	*Interface* para processar os comandos

Tipo	Método e Descrição
boolean	**sink (Command cmd, String ip)** Recebe o comando para processar e o IP do dispositivo que enviou o comando.

Classe Link – Classe responsável pela ligação entre dois dispositivos

Construtor	Descrição
Link(TOutput out, String ip, int capable)	Constrói um novo *Link* registando o fluxo de dados de saída, o IP e a capacidade do dispositivo remoto.

Tipo	Método e Descrição
void	**crashInput (String ipv6)** Desliga a ligação onde o fluxo de dados de entrada entrou em *overflow.*
static TOutput	**Create (IAdapter iadapter, Command cmd)** Cria o fluxo de dados de saída do adaptador colocado como parâmetro e envia o comando para iniciar a comunicação.
void	**downDomainServer (String ipv6)** Desliga o LS e todos os serviços.
double	**downLocalServer ()** Desliga a ligação com o LS que encerrou inesperadamente.
static String	**GetIPV6 ()** Retorna o endereço IPV6 do dispositivo.
static String	**getcapable_ipv6 (String ip)** Retorna o IPV6 do LS *capable* no caso de o LS ser *ineffective.*
boolean	**has_remote_address (String ip)**

	Verifica se o IP do parâmetro é o IP do Link.
static void	**insert_waiting_links (String ip, TOutput out)** Insere as ligações em espera até à receção do comando SIGN.
void	**receiver(IAdapter, iadapter, IProcessCommand iprocesscommand)** Classe responsável pela receção dos comandos. Tem como parâmetros o adaptador para criar o fluxo de dados de entrada e a classe onde os comandos têm que ser processados.
static void	**remove_remote_address (String ipv6)** Remove a ligação com o dispositivo com o IPV6 inserido como parâmetro.
void	**Send (Command cmd)** Responsável pelo envio do comando
static void	**Send (String ipv6, Command command)** Envia o comando para o dispositivo com o IPV6 inserido como parâmetro.

Classe LinkBuilder – Classe responsável pela construção da classe Link. Estende a classe Thread do Java.

Construtor	Descrição
LinkBuilder (TOutput out, String ip, IAdapter iadapter, int capable, IProcessCommand iprocesscommand)	Constrói um novo Link.

Tipo	Método e Descrição

void	**run ()** Constrói o Link e inicia a sua função de receber dados.

Classe Links – Classe responsável por guardar todos os Links do dispositivo. Implementa a *interface* Iterable do Java.

Tipo	Método e Descrição
void	**add_link (Link link)** Adiciona um novo link à lista.
Link	**get_link ()** Retorna o primeiro link da lista.
int	**n_links ()** Retorna o número de links da lista.
void	**remove_link (Link link)** Remove um link à lista.

Classe LocalServer – Classe que interage diretamente com a *interface* do utilizador. Responsável por iniciar o ClientManager e o ServerManager.

Construtor	Descrição
LocalServer (String id, String secury_key, Capable adapteres IDataSink idatasink)	Constrói um *Local Server* que recebe os dados da autenticação do utilizador, a classe para visualizar os resultados e os diversos adaptadores que suporta.

Tipo	Método e Descrição
static void	**Close ()** Fecha todas as classes do LS para o seu encerramento.
static boolean	**create_publisher (String security_profile, String service)**

	Requisita ao ClientManager um novo *publisher* do serviço escolhido.
static boolean	**create_subscriber (String security_profile, String service)** Requisita ao ClientManager um novo *subscriber* do serviço escolhido.
static boolean	**disconnect ()** Requisita ao ClientManager para desligar o LS.
static void	**Disconnectserial ()** Fechar a ligação Série
static void	**Disconnectshimmer ()** Fechar a ligação com a Shimmer
void	**iniatialize_BP (String uuid)** Estabelece a ligação com o monitor de pressão arterial. É recebido o uuid da ligação Bluetooth.
void	**iniatialize_BP (String uuid)** Estabelece a ligação com o monitor de pressão arterial. É recebido o uuid da ligação Bluetooth.
static void	**iniatialize_serial (int baudrate, String port)** Estabelece a ligação com o dipositivo de comunicação série. É recebido o baudrate e a porta da ligação.
void	**initialize_shimmer (String uuid, String address)** Estabelece a ligação com a unidade Shimmer. É inserido o uuid da ligação Bluetooth e o endereço da Shimmer.
static boolean	**send (String service, List<Double> data)** Requista ao ClientManager para enviar os novos dados inseridos pelo utilizador.

double	**shimmer_getSamplingRate()** Requisita à unidade Shimmer a frequência de amostragem.
static void	**start_shimmer()** Envia o comando de start à unidade Shimmer para iniciar a transmissão de dados.
static void	**stop_shimmer()** Envia o comando de *stop* à unidade Shimmer para parar a transmissão de dados.
static boolean	**unpublisher (String service)** Requisita ao ClientManager a remoção do *publisher* do serviço escolhido.
static boolean	**unsubscribe (String service)** Requisita ao ClientManager a remoção do *subscriber* do serviço escolhido.

Classe SCBTAdapter – Classe responsável por realizar a ligação Bluetooth com um LS ou um DS em Java. Implementa a classe IAdapter.

Construtor	Descrição
SCBTAdapter (String endereço, String uuid, String port, String ipv6)	Constrói o adaptador Bluetooth. Recebe como parâmetros o endereço, a porta e UUID do dispositivo que é necessário ligar-se e o próprio ipv6.

Tipo	Método e Descrição
String	**accept ()** Método Não Suportado
void	**close()** Encerra a ligação.
void	**destroy ()** Método Não Suportado

String	**getByInetAddress ()** Retorna o próprio ipv6.
InputStream	**getInputStream ()** Responsável por construir o fluxo de dados de entrada
String	**getLocalAddress ()** Retorna o seu próprio endereço.
OutputStream	**getOutputStream ()** Responsável por construir o fluxo de dados de saída
String	**getRemoteAddress ()** Retorna o endereço do dispositivo ao qual se encontra ligado.
boolean	**isClosed()** Verifica se a ligação se encontra fechada.

Classe Security – Classe responsável pela segurança dos dados.

Tipo	Método e Descrição
static byte[]	**encrypt (byte [] toEncrypt)** Responsável pela encriptação dos dados
static byte []	**decrypt (byte [] toDecrypt)** Responsável pela desencriptação dos dados

Classe SerialAdapter – Classe responsável por realizar a ligação Série. Implementa a classe IAdapter.

Construtor	Descrição
SerialAdapter (int baudrate, String port)	Constrói o adaptador Série. Recebe como parâmetros o baudrate e a porta.

Tipo	Método e Descrição
String	**accept ()** Método Não Suportado
void	**close()** Encerra a ligação.
void	**destroy ()** Método Não Suportado
String	**getByInetAddress ()** Retorna o próprio ipv6
InputStream	**getInputStream ()** Responsável por construir o fluxo de dados de entrada
String	**getLocalAddress ()** Retorna o seu próprio endereço.
OutputStream	**getOutputStream ()** Responsável por construir o fluxo de dados de saída
String	**getRemoteAddress ()** Retorna o endereço do dispositivo ao qual se encontra ligado.
boolean	**isClosed()** Verifica se a ligação se encontra fechada.

Classe SerialManager – Classe que gera a ligação entre um LS *capable* e o dispostivo de aquisição. Implementa a classe IProcessCommand e estende a classe Thread oferecida pelo Java.

Construtor	Descrição
SerialManager (IAdapter iadapter)	Constrói um SerialManager que recebe o adaptador da ligação.

Tipo	Método e Descrição
static void	**addservice (int sernsorview)** Regista um novo *subscriber*.
static void	**disconnect ()** Desliga o dispositivo de aquisição.
boolean	**process (Command command, String ip)** Processa os comandos recebidos.
synchronized void	**run()** Responsável por aceitar a ligação com o monitor de pressão arterial e receber os dados
static boolean	**verifyservice (int sernsorview)** Verifica se existe algum *subscriber* para o determinado serviço.

Classe ServerBT – Classe responsável por aceitar ligações Bluetooth em Android. Implementa a classe IAdapter.

Construtor	Descrição
ServerBT (String uuid, String name)	Constrói o adaptador Bluetooth. Recebe como parâmetros o uuid e o nome para os outros dispositivos se ligarem.

Tipo	Método e Descrição
String	**accept ()** Aceita novas ligações e retorna o IP do dispositivo remoto.
void	**close()** Encerra a ligação.
void	**destroy ()** Destrói o adaptador.
String	**getByInetAddress ()**

	Retorna o próprio ipv6.
InputStream	**getInputStream ()** Responsável por construir o fluxo de dados de entrada
String	**getLocalAddress ()** Retorna o seu próprio endereço.
OutputStream	**getOutputStream ()** Responsável por construir o fluxo de dados de saída
String	**getRemoteAddress ()** Retorna o endereço do dispositivo ao qual se encontra ligado.
boolean	**isClosed()** Verifica se a ligação se encontra fechada.

Classe ServerManager – Classe responsável por gerir as ligações do LS na função de *server*. Implementa a classe IProcessCommand e estende a classe Thread oferecida pelo Java.

Construtor	Descrição
ServerManager (IAdapter iadapter, int capable, IDataSink idatasink)	Constrói um ServerManager que recebe o adaptador da ligação, a capacidade do LS e a classe de visualização dos dados.

Tipo	Método e Descrição
boolean	**process (Command command, String ip)** Processa os comandos recebidos.
synchronized void	**run()** Aceita as ligações. Constrói o fluxo de dados de saída e o LinkBuilder().

Classe ServerSCBT – Classe responsável por aceitar ligações Bluetooth em Java.

Implementa a classe IAdapter.

Construtor	Descrição
ServerSCBT (String uuid, String name)	Constrói o adaptador Bluetooth. Recebe como parâmetros o uuid e o nome para os outros dispositivos se ligarem.

Tipo	Método e Descrição
String	**accept ()** Aceita novas ligações e retorna o IP do dispositivo remoto.
void	**close()** Encerra a ligação.
void	**destroy ()** Destrói o adaptador.
String	**getByInetAddress ()** Retorna o próprio ipv6.
InputStream	**getInputStream ()** Responsável por construir o fluxo de dados de entrada
String	**getLocalAddress ()** Retorna o seu próprio endereço.
OutputStream	**getOutputStream ()** Responsável por construir o fluxo de dados de saída
String	**getRemoteAddress ()** Retorna o endereço do dispositivo ao qual se encontra ligado.
boolean	**isClosed()** Verifica se a ligação se encontra fechada.

Classe ServerWifi – Classe responsável por aceitar ligações Ethernet. Implementa a classe IAdapter.

Construtor	Descrição
ServerWifi (int port, String ipv6)	Constrói o adaptador Ethernet. Recebe como parâmetros a porta para os dispositivos se ligarem e o ipv6 local.

Tipo	Método e Descrição
String	**accept ()** Aceita novas ligações e retorna o IP do dispositivo remoto.
void	**close()** Encerra a ligação.
void	**destroy ()** Destrói o adaptador.
String	**getByInetAddress ()** Retorna o próprio ipv6
InputStream	**getInputStream ()** Responsável por construir o fluxo de dados de entrada
String	**getLocalAddress ()** Retorna o seu próprio endereço.
OutputStream	**getOutputStream ()** Responsável por construir o fluxo de dados de saída
String	**getRemoteAddress ()** Retorna o endereço do dispositivo ao qual se encontra ligado.
boolean	**isClosed()** Verifica se a ligação se encontra fechada.

Classe Service – Classe responsável pela criação e controlo dos serviços

Construtor	Descrição
Service (String ipv6, int type, String security_key)	Cria um novo serviço associado ao IPV6 com a chave de segurança. O *type* do serviço pode ser *publisher* ou *subscriber.*

Tipo	Método e Descrição
boolean	**check_ipv6_services (ipv6)** Verifica a existência do serviço para aquele determinado IPV6.
List<String>	**listactiveservices()** Retorna os serviços ativos.
static void	**remove_remote_service(String ipv6 , string service, int type)** Remove o tipo do serviço indicado nos parâmetros.
List<String>	**verify_type (String service, int type)** Retorna todos os serviços do tipo escolhido no parâmetro.

Classe Services – Classe responsável por guardar todos os serviços do dispositivo. Implementa a *interface* Iterable do Java.

Tipo	Método e Descrição
void	**add_service (Service service)** Adiciona um novo serviço à lista.
int	**n_links ()** Retorna o número de serviços da lista.
void	**remove_link(Service service)** Remove um serviço à lista

Classe ShimmerManager – Classe responsável por gerir as ligações com a unidade Shimmer. Estende a classe Thread oferecida pelo Java.

Construtor	Descrição
ShimmerManager (IAdapter iadapter, IDataSink idatasink)	Constrói um ShimmerManager que recebe o adaptador da ligação e a classe de visualização dos dados.

Tipo	Método e Descrição
static void	**disconnectshimmer()** Desliga a comunicação com a Shimmer.
static void	**inscribe_shimmer()** Iniciação da comunicação com o Shimmer
static void	**removeservice(int sensorview)** Remove o serviço subscrito.
synchronized void	**run()** Responsável pela troca de mensagens com a unidade Shimmer
void	**sink()** Responsável pela formatação dos dados para os reencaminhar para a classe de visualização
static boolean	**verifyservice (int sernsorview)** Verifica se existe algum *subscriber* para o determinado serviço.

Classe TInput – Classe responsável pela receção e formatação das mensagens recebidas pelo dispositivo.

Construtor	Descrição
TInput (ObjectInputStream input)	Constrói a classe de receção de dados. Recebe como parâmetro o fluxo de entrada de dados.

Tipo	Método e Descrição
command	**readCommand ()** Recebe todas as mensagens e procede à formatação.

Classe TOutput – Classe responsável pela formatação e envio das mensagens recebidas pelo dispositivo.

Construtor	Descrição
TOutput (ObjectOutputStream output)	Constrói a classe de envio de dados. Recebe como parâmetro o fluxo de saída de dados.

Tipo	Método e Descrição
void	**sendCommand (Command cmd)** Formata o comando e procede ao seu envio.

Classe WifiAdapter – Classe responsável por realizar a ligação Ethernet. Implementa a classe IAdapter.

Construtor	Descrição
SerialAdapter (String ip, int port, String ipv6)	Constrói o adaptador Série. Recebe como parâmetros o ip e a porta do dispositivo remoto e o seu próprio ipv6.

Tipo	Método e Descrição
String	**accept ()** Método Não Suportado
void	**close()** Encerra a ligação.
void	**destroy ()** Método Não Suportado

String	**getByInetAddress ()** Retorna o próprio ipv6.
InputStream	**getInputStream ()** Responsável por construir o fluxo de dados de entrada
String	**getLocalAddress ()** Retorna o seu próprio endereço.
OutputStream	**getOutputStream ()** Responsável por construir o fluxo de dados de saída
String	**getRemoteAddress ()** Retorna o endereço do dispositivo ao qual se encontra ligado.
boolean	**isClosed()** Verifica se a ligação se encontra fechada.

Anexo B. Diagrama de Classe do Software

Na Figura B.1 mostra-se o diagrama UML de classes, que exibe os diferentes componentes que compõem a infraestrutura de comunicações.

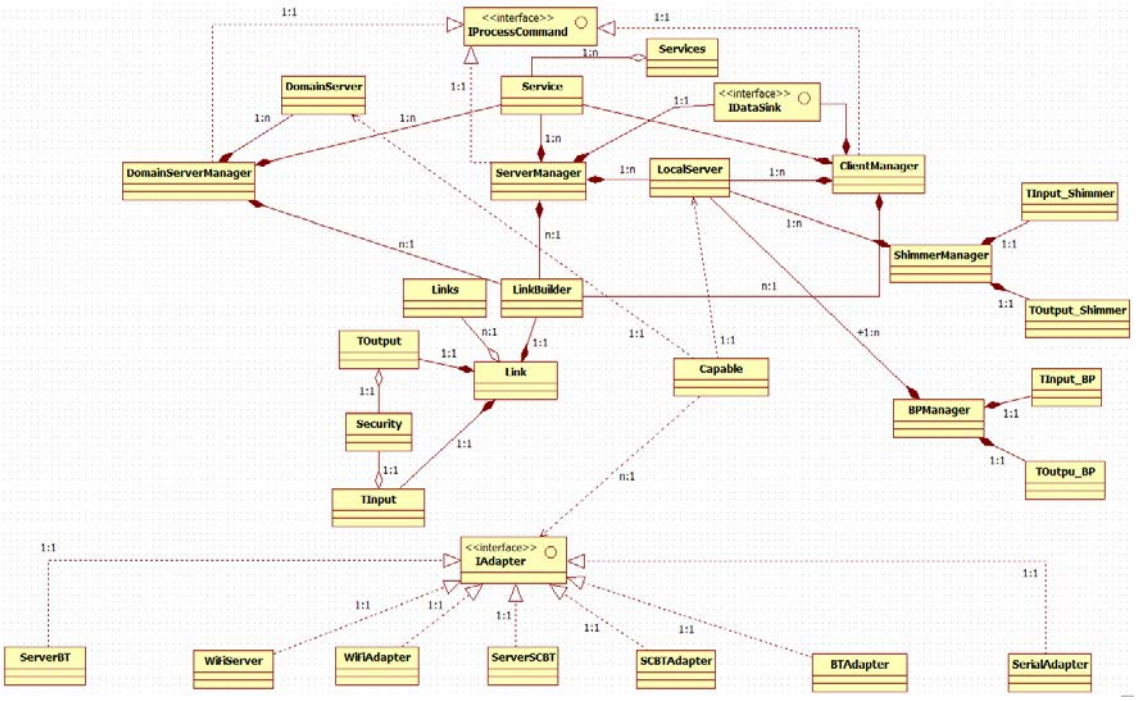

Figura B.1 – Diagrama de classe do *software*

Os principais componentes são: o *DomainServerManager*, responsável por gerir as ligações; o *Capable,* responsável por fornecer os adaptadores a cada dispositivo; o *IProcessCommand*, responsável por processar todas as mensagens da infraestrutura de comunicações; o *Service*, onde estão registados todos os serviços; o *Link*, onde estão registadas as várias ligações existentes e que são usadas pelo *ServerManager* e pelo *ClientManager* quando é criada uma nova ligação.

Anexo C. Pandaboard ES

Na Figura C.1 exibe a PandaBoard ES. Esta é uma plataforma de baixo custo e baixo consumo de energia, que se destina ao desenvolvimento de *software* móvel. É equipada com um chip Multimedia Application Platform Open de quarta geração (OMAP4). O nome da marca OMAP é da família da System on Chips (SoC) para aplicações multimédia móveis, desenvolvidas e fabricadas pela Texas Instruments (empresa norte-americana de semicondutores e informática).

A PandaBoard ES é equipada com o OMAP4430 SoC, que possui um processador dual-core ARM Cortex- A9 com cada núcleo a correr a 1 GHz, um SGX540 PowerVR Graphics Processing Unit (GPU) e um acelerador de imagem vídeo de alta definição (IVA- HD). A família OMAP4 é uma das primeiras arquiteturas, lançadas no mercado, baseadas em Cortex- A9 dual-core.

A PandaBoard ES suporta a reprodução de conteúdo de vídeo de alta definição em HDTV 1080p. O suporte de vídeo de alta performance da PandaBoard é possível graças ao seu *hardware* dedicado - o Sub-Sistema Ducati.

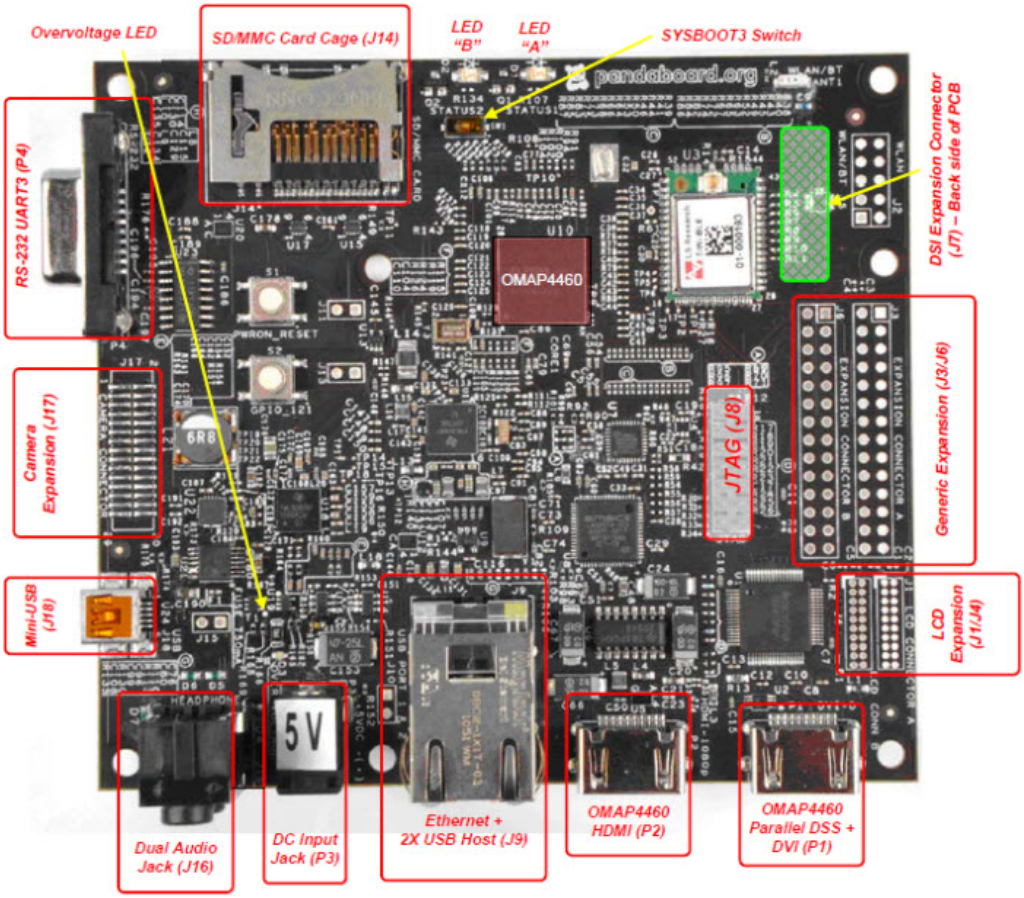

Figura C.1 - PandaBoard ES

Pandroid é um sistema operativo da Texas Instruments com base no sistema Android de suporte para a PandaBoard ES. O Android é um sistema operativo para dispositivos móveis, desenvolvido e mantido pelo Google. Trata-se de um sistema completo baseado no kernel do Linux de código aberto, ou seja, bibliotecas de *middleware*, que podem ser utilizadas por aplicações e pessoas que desenvolvem programas. As aplicações são os programas reais que o utilizador vê numa tela e com o qual interage. Para a implementação da infraestrutura de comunicações foi usado o Android 4.0.

A diferença entre Pandroid e Android é o número de fragmentos que permite o suporte para o *hardware* encontrado na PandaBoard ES (*hardware* específico da Texas Instruments ainda não incluiu o kernel do Linux). Como o Android é projetado para utilização em ecrã táctil e não tem suporte de rato USB, agregou-se um ecrã táctil à PandaBoard ES. Esta expansão, como se observa na Figura C.2, adiciona à PandaBoard ES novos periféricos, tais como: LCD de sete

polegadas, tela de toque capacitivo, bússola, giroscópio, cinco botões para o utilizador e suporte de cartão SIM.

Figura C.2 -Expansão da PandaBoard ES