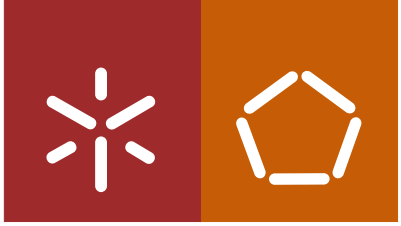
Universidade do Minho
Escola de Engenharia

Sónia Carolina de Carvalho Silva Antão

Monitorização de dados sensoriais biomédicos através de SNMP

julho de 2016

Monitorização de dados sensoriais biomédicos através de SNMP Sónia Carolina de Carvalho Silva Antão UMinho | 2016



Universidade do Minho
Escola de Engenharia

Sónia Carolina de Carvalho Silva Antão

Monitorização de dados sensoriais biomédicos através de SNMP

Dissertação de Mestrado
Mestrado Integrado em Engenharia de Telecomunicações e Informática

Trabalho efetuado sob a orientação do
Professor Doutor Bruno Alexandre Fernandes Dias

julho de 2016

DECLARAÇÃO

Nome: Sónia Carolina de Carvalho Silva Antão
Endereço eletrónico: sonia.antao@ymail.com Telefone: 916621407
Bilhete de Identidade/Cartão do Cidadão: 14159055
Título da dissertação: Monitorização de dados sensoriais biomédicos através de SNMP

Orientador:
Bruno Alexandre Fernandes Dias

Ano de conclusão: 2016
Mestrado Integrado em Engenharia de Telecomunicações e Informática

Universidade do Minho, _____/_____/_____

Assinatura:

AGRADECIMENTOS

Esta dissertação foi um culminar de cinco anos intensivos de estudos em Engenharia de Telecomunicações. Foi uma viagem dura, com muitos obstáculos mas que, felizmente, foram todos superados, não só devido ao esforço pessoal mas graças também a muito apoio de familiares e amigos queridos.

Quero começar por agradecer ao meu professor orientador Bruno Dias. Sempre se mostrou imensamente prestável e amigo durante todo este percurso.

Agradeço também ao professor Jorge Cabral pois, sem ele, não teria conseguido ultrapassar certos obstáculos que encontrei na minha dissertação relativamente à parte de eletrónica. Foi muito amigo quando realmente precisei de ajuda.

Os meus pais foram incansáveis comigo e, por isso mesmo, não podia deixar de retribuir a força que me deram. Ajudaram-me nos momentos mais difíceis onde a vontade de ir trabalhar já se tornava mais aliciante do que concluir o curso.

Ao meu amigo Portela que sempre me fez rir durante toda esta viagem e me ajudou nalgumas questões mais práticas.

Ao meu amigo João que me deu muito apoio sempre que precisei.

Às minhas amigas – Sara e Bruna – que também se preocuparam em me perguntar como ia o meu trabalho; deram-me força e foram elementos distratores fundamentais.

Por fim, e não menos importante, ao meu namorado, por toda a paciência, força e motivação que diariamente me deu para concluir o curso. Também ele foi uma enorme ajuda, pois esteve sempre comigo para o bem e para o menos bem.

A todos, um grande bem hajam.

RESUMO

A população mundial está a envelhecer, sendo que, na Europa, 5% da população tem mais de 80 anos e estima-se que este número venha a triplicar nos próximos 20 anos. Esta evolução traz consigo um novo conjunto de desafios sociais e económicos, nomeadamente no âmbito da prestação de cuidados médicos.

Uma das vertentes mais importantes da prestação de cuidados médicos é a monitorização de pacientes. Os sensores biomédicos atuais são dispendiosos e funcionam com sistemas computacionais protegidos pelos fabricantes, com tecnologias próprias não partilhadas. Assim, torna-se útil a definição dum novo paradigma capaz de diminuir relevantemente os custos de aquisição, instalação e manutenção desses sistemas de monitorização.

Nesta dissertação apresenta-se um modelo genérico de monitorização biomédica capaz de ser implementado em unidades hospitalares, visando a recolha de dados sensoriais biomédicos, o seu pré-processamento e eventual integração numa base de dados global (numa *cloud* local, por exemplo). A arquitetura permite a monitorização automática de pacientes de variados serviços de saúde com a menor intervenção humana possível, independentemente do tipo de sensor utilizado, com baixo custo de implementação e de aplicação universal.

O sistema desenvolvido utiliza mecanismos normalizados para a representação da informação a monitorizar assim como para a comunicação entre as entidades da arquitetura, e que são baseados nas tecnologias amplamente utilizadas para a gestão de redes Internet. Nomeadamente, foram criadas definições para novas bases de dados específicas para monitorização e configuração de sensores biomédicos utilizando o paradigma das *Management Information Bases*. Além disso, o protocolo de comunicação entre as entidades da arquitetura proposta é o *Simple Network Management Protocol* (SNMP).

Como prova de conceito foi implementado, com sucesso, um protótipo que ilustra a arquitetura proposta, incluindo o *hardware* dum sensor biomédico básico de baixo custo e o *software* dum agente SNMP e duma simples aplicação biomédica capaz gerar alertas em situações clinicamente pre-definidas por uma equipa médica.

Palavras-Chave: Sensores Biomédicos, Monitorização Biomédica, *Simple Network Management Protocol*, *Management Information Base*.

ABSTRACT

The world population is getting old, and, in Europe, 5% of the population is over 80 years and it is estimated that this number will triple over the next 20 years. This development brings with it a new set of social and economic challenges, in particular the provision of medical care.

One of the most important aspects of medical care is patient monitoring. Current biomedical sensors are expensive and work with computer systems protected by manufacturers to own technologies not shared. Thus, it is useful a new paradigm definition able to substantially reduce the costs of acquisition, installation and maintenance of these monitoring systems.

This dissertation presents a generic model of biomedical monitoring to be set in hospitals in order to gather biomedical sensor data, its preprocessing and eventual inclusion in a global database (a local cloud, for example). The architecture allows automatic monitoring of various health care patients with the least possible human intervention, no matter the type of sensor used, with low cost implementation and universal application.

The created system uses standard mechanisms for the representation of monitoring information, as well the communication between the entities of the architecture, which are based on widely used technologies for the Internet networks management. In particular, definitions were created for a new basis of specific data for monitoring and configuration of biomedical sensors using the paradigm of Management Information Bases. In addition, the communication protocol between the entities of the proposed architecture is the Simple Network Management Protocol (SNMP).

As proof of concept, a prototype that illustrates the proposed architecture, including the low-cost basic biomedical hardware sensor and software of an SNMP agent and a simple biomedical application that can generate alerts in clinically pre-defined situations by medical team, has been successfully implemented.

Keywords: Health Sensors, Health Monitoring, Simple Network Management Protocol, Management Information Base.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÓNIMOS

AES	Advanced Encryption Standard
APP	Application Programming Interface
API	Application Programming Interface
ASN.1	Asbtract Syntax Notation 1
CGMS	Sistema de Monitorização Contínua de Glucose/Glicose
CMIP	Common Management Information Protocol
DES	Data Encryption Standard
ECG	Eletrocardiograma
ECMO	Extracorporeal Membrane Oxygenation
HTML	HyperText Markup Language
ICMP	Internet Control Message Protocol
IETF	Engineering Task Force
IP	Internet Protocol
ISO	International Organization Standardization
JDK	JDK Java Development Kit
LED	Light Emitting Diode
MIB	Management Information Base
MO	Managed object
MOSFETs	Metal Oxide Semiconductor Field Effect Transistor
NETCONF	Network Configuration Protocol
OID	Object identifier
PC	Probabilidade de Coagulação
PIC	Pressão intracraniana
RSSF	Redes de sensores sem fios
Rx	Receiver
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SNMP4J	Simple Network Management Protocol for

	java
spO2	Oxímetro de pulso
SSH	Secure Shell
TCP	Transmission Control Protocol
Tx	Transmission
UDP	User Datagram Protocol

1. INTRODUÇÃO

Os gastos com as atividades do setor da saúde têm vindo a aumentar nas últimas décadas, sobretudo nas sociedades mais desenvolvidas [1]. Uma das vertentes mais importantes é o custo com sistemas de monitorização dos pacientes quando estes estão em regime de internamento em instituições de saúde. Na grande maioria dos casos, as soluções atuais incluem equipamento físico de monitorização sensorial e aplicações computacionais biomédicas embebidas implementadas à custa de tecnologias próprias dos fabricantes e que não permitem interligação entre equipamentos ou aplicações de fabricantes diferentes [2, 3].

O desenvolvimento destes tipo de produtos em ecossistemas tecnologicamente fechados e protegidos potencia custos elevados na sua comercialização, instalação, manuseamento e manutenção. Uma forma de contribuir para a contenção desses custos é apostar na automação da monitorização dos pacientes através da construção de equipamentos genéricos e normalizados para recolha, pré-processamento e armazenamento de dados sensoriais em sistemas integrados de bases de dados biomédicos e que podem ser então acedidos por qualquer aplicação médica que respeite as tecnologias escolhidas para comunicação e manipulação desses dados. Ou seja, a eficiência do modelo é reforçada, e os custos diminuídos, se forem utilizados mecanismos e protocolos normalizados nos processos de comunicação, representação e manipulação dos valores paramétricos recolhidos pelos sensores biomédicos.

O principal objetivo do projeto, do qual resulta esta dissertação, foi a definição duma arquitetura que pudesse servir de modelo para o desenvolvimento de soluções futuras baseadas em mecanismos e tecnologias normalizadas e já largamente testadas e utilizadas em contextos tecnológicos semelhantes.

1.1 Redes de sensores

Atendendo a que um dos principais componentes dos sistemas de monitorização biomédica é constituída pelos próprios sensores, seria de esperar que algumas das tecnologias associadas às redes de sensores em geral, e em especial às Redes de Sensores Sem Fios (RSSF), também se pudessem aplicar neste contexto. Uma RSSF é um conjunto de nós de

sensores sem fio interligados, onde cada nó deteta valores físicos e os envia para um supervisor (*manager*), ou utilizadores, através de nós específicos nomeados nós de agregação (*sink nodes*). A crescente necessidade das RSSF para amostrar, processar e comunicar uma grande quantidade de dados levou ao rápido desenvolvimento de tecnologias específicas ou à adaptação de tecnologias bem testadas e utilizadas em contextos relacionados. Estas redes podem monitorizar/medir parâmetros variados do mundo físico, e podem ser utilizadas em diversos domínios científicos [4]. A tipificação das suas aplicações no mundo real depende de restrições impostas pelos mecanismos de comunicação (alcance, largura de banda, atrasos, perdas, etc), pelo contexto de inserção (ambiente no exterior ou interior de edifícios, perto ou longe de meios urbanos, período útil de utilização, frequência de renovação energética, etc), e pelos mecanismos de medição (parâmetros medidos, frequência e precisão de amostragem, capacidades de calibração automática, capacidade de armazenamento intermédio, capacidade de validação, filtragem e pré-processamento dos dados, etc). Além disso, a utilização correta de um nó sensor pressupõe, obrigatoriamente, uma gestão inteligente dos seus principais recursos: a energia, a memória (capacidade de armazenamento) e a capacidade computacional (incluindo a implementação dos mecanismos/protocolos de comunicação). De salientar que as RSSF de larga escala processam ainda maiores quantidades de dados, tornando a sua gestão ainda mais complexa [5].

Neste sentido, a definição de soluções baseadas no Simple Network Management Protocol (SNMP) têm sido recorrentes [5, 6, 7]. Em geral, estas soluções preconizam a implementação de *Management Information Bases* (MIBs) específicas em agentes SNMP dos nós de agregação. A informação de gestão flui entre estes nós e as aplicações gestoras. Um dos principais problemas destas abordagens tem a haver com a quantidade de recursos necessário num nó de agregação para a implementação eficaz de um agente SNMP, incluindo a capacidade de agregação, filtragem e processamento de dados provenientes dum grupo extenso de nós normais. A largura de banda disponível nalguns ambientes de aplicação também pode ser um fator limitativo.

1.2 Sensores biomédicos

Em geral, o contexto de aplicação das redes de sensores biomédicos tem caraterísticas menos exigentes que as RSSF. Normalmente aplicam-se em ambientes de redes cabladas ou redes sem fios interiores sem problemas de limitação na largura de banda disponível, onde a

escala (número de sensores ligados a um nó de agregação) é sempre muito limitada. Por isso, parece uma estratégia interessante que na sua gestão se apliquem os mesmos princípios das RSSF, incluindo os modelos normalizados de gestão já amplamente utilizados na gestão de redes genéricas TCP/IP. Por outro lado, as redes de sensores biomédicos exigem, em geral, maiores capacidades computacionais (sobretudo nos nós de agregação) e exibem maiores necessidades energéticas, mas estas caraterísticas não costumam apresentar limitações relevantes nos contextos normais de utilização das redes de sensores biomédicos.

A tecnologia informática aplicada à área da saúde tem desempenhado um papel de extrema importância nos últimos anos. A Engenharia Biomédica tem estado em contínua e acelerada evolução. O desenvolvimento de novos instrumentos para uso médico e sua utilização adequada em ambiente médico-hospitalar levou à criação de sensores biomédicos – equipamentos eletrónicos destinados ao diagnóstico/monitorização de pacientes sob supervisão médica. Assim, os sensores biomédicos servem de interface entre o sistema biológico e o sistema eletrónico/informático de diagnóstico/monitorização médico. Estes sensores exigem, em geral, que se estabeleça contacto físico com os pacientes (ou que, pelo menos, haja uma proximidade grande). É importante, por isso, que a instrumentação física que está em contacto com o paciente (ou em grande proximidade dele) seja o menos intrusiva e o mais confortável possível. Logo, os sensores biomédicos devem conter apenas o hardware e software estritamente necessários para a leitura e envio imediato dos parâmetros biomédicos desejados a um sistema intermédio como um nó de agregação. Por outro lado, os nós de agregação das redes de sensores biomédicos não têm limitações computacionais nem restrições comunicacionais ou de consumo de energia que sejam relevantes, ou seja, que por si só sejam limitativos para o uso de tecnologias normalizadas na sua gestão.

1.3 Gestão de redes de sensores biomédicos com SNMP

Atualmente, os sensores biomédicos vendidos por empresas como a Mouser, Hetpro, Sterimed, Quirumed, etc., são acompanhados por *software* e protocolos que o cliente desconhece, utilizando apenas os sensores para o seu devido efeito. Estes produtos implementam um sistema fechado com uma pequena rede de sensores biomédicos ligados a um nó de agregação onde é implementada uma aplicação de diagnóstico/monitorização biomédica, normalmente com interfaces específicos e são, invariavelmente, equipamentos dispendiosos com contratos de manutenções caros. Um dos problemas associados ao contexto

fechado em que estes produtos são desenvolvidos tem a ver com a validação de diagnósticos automatizados uma vez que é difícil que a comunidade científica valide algoritmos proprietários desconhecidos no tratamento/processamento dos dados sensoriais. Logo, a sua utilização efetiva depende sempre da validação manual dum agente de saúde humano.

O consumidor final teria a ganhar se fosse possível baixar o custo de produção e de utilização destes sistemas de monitorização/diagnóstico. Um primeiro passo seria a adoção duma arquitetura comum e modular para estes sistemas, em que empresas duma mesma área de especialização pudessem concorrer livremente. Deveria ser possível adquirir um sistema de dignóstico/monitorização por módulos, substituindo apenas um módulo quando fosse desejado ou necessário. A única forma adequada de definir uma arquitetura universal e modular é utilizar tecnologias normalizadas já amplamente difundidas em contextos de aplicação semelhantes. Assim, dever-se-ia utilizar redes de comunicação universais, como as redes TCP/IP, para a interligação dos componentes principais do sistema – os nós de gregação dos dados biomédicos e os sistemas gestores de aplicações de diagnóstico/ monitorização biomédico.

Nesta dissertação propõe-se isso mesmo, a definição duma arquitetura genérica, modular e universal para sistemas de diganóstico/monitorização biomédico, em que são utilizados protocolos de comunicação universais entre os componentes do sistema. Mais ainda, preconiza-se que o armazenamento, filtragem e processamento inicial dos dados biomédicos amostrados pelos sensores seja efetuado em bases de dados normalizadas, MIBs, implementados nos nós de agregação e que o acesso e manipulação desses dados por aplicações de diganóstico/monitorização biomédico, a executar em sistemas computacionais gestores, seja feito através do protocolo de gestão SNMP.

Espera-se que esta estratégia possa despoletar o desenvolvimento e produção dos componentes do sistema em separado. O aumento da especificidade no desenvolvimento e produção costuma trazer um abaixamento dos custos e melhorias nas funcionalidades disponíveis. Por um lado, a adoção de normas internacionais na representação dos dados sensoriais biomédicos deve poder potenciar o desenvolvimento de algoritmos normalizados para diagnóstico/monitorização biomédico, levando a uma maior automação a todo o processo, diminuindo a necessidade de intervenção humana muito especializada.

Por fim, este paradigma torna possível o armazenamento dos dados sensoriais biomédicos recolhidos ao longo do tempo em sistemas de bases de dados alargados, como é o caso das *clouds* privadas dos centros hospitalares, ainda que também traga problemas relevantes de segurança, sobretudo no que concerne a garantias de confidencialidade.

1.4 Estrutura do documento

Esta dissertação está dividida em seis capítulos. O primeiro corresponde à introdução, onde se faz o enquadramento dos principais objetivos e motivações do trabalho apresentado na presente dissertação. O segundo capítulo dedica-se a uma breve exposição das tecnologias associadas e com maior relevo para a solução proposta. No terceiro capítulo é exposta a solução definida para a arquitetura proposta. No capítulo quarto são apresentados os aspetos mais relevantes do protótipo desenvolvido e no capítulo seguinte são discutidos os resultados obtidos com o protótipo. No último capítulo são feitas as conclusões desta dissertação e indicadas as principais dificuldades sentidas e possíveis melhorias a implementar em trabalho futuro. O documento fecha com a lista de referências e um anexo com a listagem completa da MIB desenvolvida para os sistemas de redes de sensores biomédicos.

2. TECNOLOGIAS ASSOCIADAS

A definição dos objetos conceptuais dos sistemas de bases de dados a monitorizar e a implementação da interligação entre os sensores de agregação e as aplicações médicas de diagnóstico, devem basear-se em metodologias abertas e já utilizadas em larga escala, não havendo dependência de tecnologias que usam *software* protegido pelos fabricantes.

Conforme já foi referido, esta abordagem permite a concentração dos esforços dos especialistas da área da saúde nos extremos aplicativos da implementação de algoritmos para funções de diagnóstico e na criação de sensores biomédicos mais económicos e eficientes. Por outro lado, todo o sistema de modelação de dados e comunicação pode ser desenvolvido por especialistas informáticos utilizando mecanismos e protocolos normalizados, como os baseados na tecnologia SNMP, que é o modelo mais utilizado em monitorização de equipamentos, serviços e aplicações distribuídas em redes IP.

2.1 Redes de sensores biomédicos

O desenvolvimento de equipamentos de diagnóstico para monitorização e análise de sinais biomédicos sofreu um forte crescimento nos últimos anos, e a procura de novos sistemas de aquisição de sinais biomédicos justifica a evolução de novas soluções que respondam às modernas exigências médicas.

Produtos de monitorização biomédica possuem *hardware* e *software* próprio, fornecidos como um sistema fechado e, em geral, sem mecanismos de integração direta dos dados numa base de dados biomédicos mais abrangente. Estes sistemas fechados são patenteados e inacessíveis a entidades externas para validação de algoritmos ou processos de implementação. Isto empurra as unidades hospitalares a utilizar produtos dum único fabricante, até porque, só assim é possível, algumas vezes, a integração em rede de vários sistemas iguais a funcionar na mesma instituição. Neste contexto, o custo destes produtos é, mais facilmente, mantido a níveis elevados, tal como a dependência tecnológica num único tipo de solução para *hardware* e *software*.

A maoria dos produtos de diagnóstico/monitorização biomédico requer uma intervenção humana frequente através tipos de interfaces aplicativos muito diferenciados de marca para

marca, ou até de produto para produto, com tomadas de decisão que podem influenciar determinantemente a adequabilidade do seu uso a um determinado paciente. Esta dependência de intervenções manuais, muitas vezes com necessidade de manipulação física de botões e manípulos, pode ser prejudicial em termos sanitários e está sujeita, não só à disponibilidade de operadores qualificados, como a ocorrência de erros de origem humana. A automação acrescida, potenciada pela normalização dos processos, é uma das formas de minimizar estes riscos.

Outro importante aspeto a considerar é o facto das unidades hospitalares ainda não manterem bases de dados biomédicos universais e seguras, com informação capaz de ser diretamente acedida e processada por sistemas automáticos e autónomos de diagnóstico/monitorização ou investigação médica. Também não costuma ser possível as unidades hospitalares partilharem diretamente entre si as suas bases de dados biomédicos. Hoje em dia, as bases de dados hospitalares mantêm apenas informação biográfica, o historial médico e o registo da prescrição medicamentosa dos seus pacientes. Além disso, a partilha de informação direta por meios informáticos entre essas unidades só é possível quando o sistema informático foi desenvolvido e instalado pelo mesmo fabricante.

Note-se que várias instituições de normalização têm estudado esta temática da aplicação de sistemas informáticos no diagnóstico/monitorização de pacientes em contextos institucionais (incluindo as componentes de representação dos dados biomédicos e das tecnologias de comunicação a utilizar num esforço de integração e interoperabilidade entre sistemas), mas sem publicações relevantes de normas internacionais. Talvez o esforço mais significativo de análise dos problemas desta área – identificada genericamente como eHealth – tem sido o da *European Telecommunications Standards Institute* (ETSI). Em [8] apresentam, inclusive, um conjunto de recomendações que definem uma arquitetura para os sistemas de informação que integrem todos os serviços de saúde informatizados de unidades hospitalares, incluindo os serviços de diagnóstico/monitorização de pacientes.

Tipos de sensores biomédicos

Existem inúmeros sensores biomédicos, permitindo o diganóstico/monitoriação de variados parâmetros sensoriais biomédicos. Em geral, os sensores biomédicos classificam-se em dois tipos – físico ou químico – podendo ser de monitorização contínua ou não. Os

sensores físicos registam quantidades. Como exemplo deste tipo de sensor biomédico temos o eletrocardiograma (ECG), que faz o estudo do batimento cardíaco através da amostragem dum sinal elétrico gerado pelo batimento do coração do paciente. Os sensores químicos registam e monitorizam concentrações de vários componentes químicos. Um exemplo são os sensores que amostram a quantidade de determinados gases presente num determinado contexto, sendo particularmente úteis em medidas biomédicas associadas ao sistema pulmonar.

Adicionalmente, os sensores biomédicos podem ser invasivos e não-invasivos. Sempre que possível, o segundo caso é a opção mais desejável, a não ser que seja realmente necessário a utilização de um sensor interior, o que normalmente reduz a interferência a fatores externos e aumenta a precisão das medições. Um caso típico de sensor invasivo são, por exemplo, os elétrodos que monitorizam continuamente os sinais elétricos gerados pelo batimento cardíaco dum paciente que utiliza um *pace-maker* cardíaco. Este procedimento invasivo decorre da necessidade da instrumentação sensorial ter de ficar próximo do coração.

Para além das caraterísticas biomédicas específicas destes sensores, existem também caraterísticas associadas de carácter geral e que são do mesmo tipo para a grande maioria dos sensores biomédicos, ainda que podendo apresentar valores bem diferenciados (parâmetros de alimentação energética, dimensões físicas, períodos de manutenção, etc.).

Redes de sensores biomédicos agregados

A escalabilidade em redes de sensores é um fator crítico, sobretudo em RSSF. Assim, para RSSF de larga escala, os mecanismos de hierarquização e agregação são críticos para assegurar a escalabilidade do sistema [9]. Existem, atualmente, técnicas de agregação de dados que podem ser usadas para reduzir o *overhead* gerado pelo *broadcast* [10,11]. Alternativamente, os sensores podem comunicar com os sistemas finais diretamente (possivelmente utilizando um protocolo de encaminhamento *multihop*) ou comunicar-se com um *cluster-head* usando *unicast*. Por outro lado, na estratégia *multicast* os sensores formam grupos previamente definidos por uma aplicação gestora e usam *multicast* para a comunicação entre membros do grupo.

Já tinha sido referido que, ainda que no caso específico das redes de sensores biomédicos para diagnóstico/monitorização em ambiente hospitar o fator da escala não ser relevante, torna-se essencial a utilização de mecanismos de agregação simples dos dados em nós de agregação. Neste tipo de redes de sensores, em vez da escalabilidade ser o fator determinante para a adoção de mecanismos de agregação, o ganho mais importante tem a ver com uma gestão mais eficiente dos recursos, tanto do ponto de vista tecnológico como do ponto de vista económico. Como a instrumentação dos sensores biomédicos estão em contacto com os pacientes (ou em grande proximidade) e é normal existirem vários sensores ligados simultaneamente a um paciente, faz todo o sentido que se faça a agregação dos dados sensoriais amostrados num único nó de agragação, ainda que os tipos das amostras recolhidas e as frequências de amostragem possam ser completamente diferentes entre si. Reduz-se assim o número de componentes com necessidades computacionais para embeber um sistema inteligente que implemente os processos de recolha, filtragem e processamento de dados, além dos mecanismos de comunicação para poder interagir com aplicações biomédicas ou sistemas de armazenamento de informação biomédica em bases de dados de larga escala. Outro problema que é mais fácil resolver se for utilizada uma estratégia de agregação de dados em nós de agregação é o da segurança, até porque se baixa o número de componentes do sistema com acesso a dados sujeitos a requisitos importantes de autenticação e confidencialidade.

A estratégia de agregação de vários tipos de dados sensoriais recolhidos num único nó de agregação pode ser implementada com um modelo abstrato e normalizado para a representação desses dados, como é o caso, por exemplo, dos objetos definidos nas MIBs, utilizando a norma *Struture of Management Information* (SMI) [12].

Por contra-ponto a uma estratégia de agregação que transfere inteligência da instrumentação sensorial para os nós de agregação, alguns projetos [13] defendem uma estratégia de aumento da inteligência da instrumentação sensorial como aproveitamento do aumento das suas capacidades computacionais e comunicacionais e da evolução do processo de miniaturização. No entanto, este processo acarreta sempre maior complexidade tecnológica (sincronização e relacionamento entre os dados recolhidos, algoritmos das aplicações biomédicas a implementar na instrumentação sensorial, energia consumida, protocolos de comunicação específicos, etc.) e custos adicionais na sua implementação.

2.2 SNMP e MIBs

O SNMP é uma arquitetura protocolar para gerir redes e que foi criada pelo *Internet Engineering Task Force* (IETF) em 1988. Está definido num conjunto alargado de normas e vai na sua terceira versão. O protocolo mais conhecido desta arquitetura é precisamente o SNMP [14]. Este protocolo aplicacional tem como principal objetivo permitir que as aplicações gestoras possam configurar e monitorizar o funcionamento dos recursos na rede (equipamentos, aplicações, etc.), acedendo e manipulando os valores das instâncias dos objetos de gestão implementados nas MIBs dos agentes [15, 16, 17, 18]. Atualmente, o SNMP é o protocolo mais utilizado para gerir aplicações distribuídas e redes de computadores, sobretudo em redes TCP/IP, que é a pilha comunicacional mais usada.

Um vantagem no uso do SNMP é o facto de ser encapsulado sobre o protocolo UDP, o que torna a sua implementação mais simples, face ao uso alternativo do TCP. Outra vantagem essencial do SNMP é a sua simplicidade e frugalidade em termos de consumo de recursos computacionais nos sistemas onde os agentes são implementados. Qualquer equipamento de rede, e imensos dispositivos informáticos, já trazem embebida uma implementação dum agente SNMP.

De notar que uma das principais debilidades desta arquitetura têm a ver com a transferência da eventual complexidade do sistema para a aplicação gestora, sobretudo quando se pretendem implementar funcionalidades de mais alto nível. Outra limitação que costuma ser referida é a falta de normas de segurança que garantam a imunidade a ataques em larga escala e o não repúdio, sendo que este tipo de requisito de segurança é muito complexo de se conseguir implementar, além de trazer ao sistema uma complexidade adicional muito significativa, incluisivé do lado do agente. Em termos de autenticação, confidencialidade e verificação de conteúdos, a arquitetura SNMP já disponibiliza um modelo de segurança e um mecanismo de controlo de acessos adequados e que estão definidos na norma *User-based Security Model* [19] e *View-Access Control Model* [20] respetivamente.

Também parecem evidentes os problemas de aplicabilidade do SNMP em contextos de larga escala, uma vez que se trata dum protocolo assíncrono e assimétrico e em que o principal paradigma para a implementação de processos de monitorização é o *polling*

intensivo dos agentes por parte das aplicações gestoras. No entanto, as evoluções ao nível da monitorização por notificações e na criação de mecanismos de hierarquização e filtragem da informação gerada pelos agentes [21], tem vindo a aumentar a imunidade da arquitetura a problemas de escalabilidade.

Ao longo das últimas décadas também se tem argumentado sobre a limitada performance do SNMP para lidar com grandes quantidades de informação gerada na instrumentação das MIBs dos agentes, uma vez que o o protoclo é assíncrono e os PDUs SNMP são encapsulados em UDP e são atómicos, isto é, uma mensagem SNMP não pode ser dividida por város PDUs. No entanto, uma cuidada definição do tipo de dados das MIBs, associada a uma adequada manipulação da informação, pode ajudar a resolver este problema.

Gestores, Agentes e MIBs

Como já foi referido, o SNMP é um protocolo aplicacional e tem como função permitir a troca normalizada de mensagens/dados de gestão entre os dispositivos geridos e as aplicações gestoras. Assim, os componentes principais desta arquitetura são: os agentes (ou servidores de gestão), os gestores (ou clientes de gestão) e as bases de dados de informações de gestão (MIB), conforme se pode verificar na figura 1.

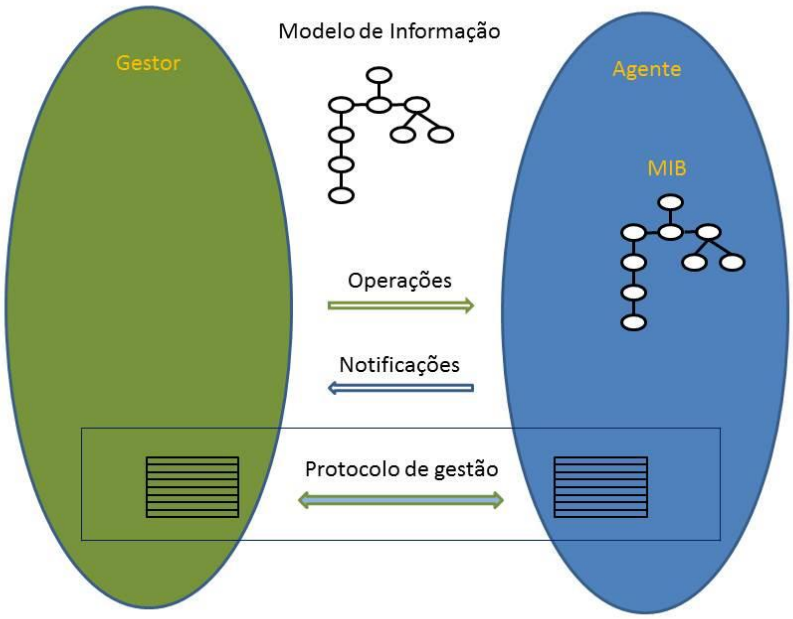


Figura 1 – Arquitetura SNMP

Os gestores, ou aplicações gestoras, são constituídos por *software* instalado em dispositivos capazes de realizar remotamente operações de monitorização e configuração dos recursos distribuídos pela rede [22]. Podem existir vários gestores, mas, por norma, os sistemas de gestão apenas usam um. Devem usar vários quando a quantidade expectável de informação de gestão a processar seja muito elevada e/ou o número de agentes seja muito grande ou quando os gestores implementam aplicativos diferentes.

Os módulos de *software* instalados nos dispositivos a serem geridos (por exemplo, encaminhadores de rede, servidores de aplicações web, etc.) são designados de agentes SNMP. Estes agentes executam os processos da instrumentação necessária à correta implementação dos objetos definidos em bases de dados normalizadas, as MIBs, e por executar os processos necessários à implementação dos mecanismos de segurança e de controlo de acesso. Os agentes são também responsáveis por responder aos pedidos dos gestores (todos os tipos de pedidos são respondidos com o mesmo tipo de PDU *Response*) ou enviar notificações definidas nas MIBs e configuradas previamente pelas aplicações gestoras.

Os gestores podem enviar pedidos de monitorização (comandos do tipo *Get*, *Get-Next* ou *Get-Bulk*) ou de configuração (comandos do tipo *Set*) aos agentes e esperam receber as respostas com os valores das instâncias dos objetos pedidos. Os gestores podem trocar informação entre si através de mensagens do tipo *Inform*. Os agentes, por seu turno, podem enviar alertas não solicitados por pedidos prévios dos gestores, sempre que determinados valores das instâncias dos objetos são atingidos (através de mensagens do tipo *Trap* ou *Notification*). Os agentes só podem enviar alertas previamente configurados pelos gestores. No protocolo SNMP não existem confirmações de receção de PDUs nem estão definidos temporizadores.

Os objetos de gestão são definidos na linguagem SMI e organizados hierarquicamente em grupos e tabelas nas MIBs [23]. Estes objetos são abstrações de parâmetros de funcionamento dos recursos que se querem gerir ou de valores estatísticos que se querem monitorizar. Para manter a simplicidade da arquitetura, são permitidos poucos tipos de objetos, um subconjunto limitado dos tipos possíveis em *Asbtract Syntax Notation 1* (ASN.1) [7]. A SMI tem como base a linguagem ASN.1, que permite definir um modelo universal de notação declarativa para tipos de objetos, independente da máquina ou sistema operativo.

Associado a cada implementação dum determinado objeto num determinado agente – ou seja, a uma instância dum objeto – está associado um tipo de acesso (leitura, escrita ou ambos) por defeito e outro por cada aplicação gestora que se queira definir em particular.

SNMP vs CMIP

O *Common Management Information Protocol* (CMIP) [24] é um protocolo para gestão de redes criado no contexto *Open System Interconncetion* (OSI). É um protocolo que é mais completo e complexo que o SNMP e que também pode lidar com MIBs. As capacidades funcionais dos agentes CMIP são menos limitadas das que os agentes SNMP e a arquitetura global, apesar de ser igualmente centralizada nas aplicações gestoras, permite mais níveis de hierarquização e mecanismos de notificação mais avançados. No entanto, a implementação do CMIP é muito mais exigente do ponto de vista dos recursos e da complexidade do software. Além disso, o CMIP foi originalmente criado dentro duma arquitetura global OSI, com a necessidade inerente de se ter de integrar nessa pilha protocolar. Com o desuso da utilização da pilha OSI completa, o IETF definiu uma possível adaptação do CMIP na pilha protocolar TCP/IP, no que ficou conhecido como CMOT (*CMIP over TCP*). Em qualquer dos casos, a utilização do CMOT nunca veio a ser afirmar-se na comunidade científica nem se integrou com sucesso em produtos comerciais.

SNMP vs NETCONF

O SNMP tem uma ampla utilização em contextos de monitorização, implementando um paradigma de *polling* baseado em interações atómicas de solicitação-resposta, com os PDUs quase sempre encapsulados em UDP [5], embora a possibilidade de se usar o TCP [25] esteja também prevista na arquitetura. Por outro lado, em contextos de configuração dos recursos a gerir, o SNMP é muito menos utilizado, muito por culpa da inexistência de mecanismos de segurança na sua primeira versão e de utilização opcional na sua segunda versão. Além disso, o facto do SNMP ser um protocolo assíncrono, não confirmado, aumenta as dificuldades na implementação de funcionalidades de configuração dos recursos a gerir (ou seja, dificuldades nos processos de modificação dos valores das instâncias dos objetos das MIBs).

Com o intuito de ultrapassar estas limitações do SNMP em contextos de configuração, foi criado o *Network Configuration Protocol* (NETCONF) [26]. Esta tecnologia emergente ainda tem uma base de implementação relativamente limitada. O NETCONF é baseado num protocolo de mensagens XML utilizando um transporte orientado à conexão, geralmente o

TCP. Além disso, o NETCONF é muitas vezes invocado dentro de uma sessão *Secure Shell* (SSH) [27].

O SNMP, contando nesta altura com três versões, sendo que nesta última versão os mecanismos de segurança são de implementação obrigatória, já não apresenta limitações relevantes nesse aspeto, tanto para monitorização como para configuração. Pode colocar-se a questão da performance do SNMP para contextos de configuração quando comparada com a performance de um protocolo desenvolvido especificamente para funções de configuração, como o NETCONF. Não existem estudos definitivos que atribuam vantagens significativas, em termos de performance, a nenhum dos protocolos, mas pode referir-se que num dos estudos comparativos mais citados pela comunicade científica da área de gestão de redes [28] foi realizada uma análise empírica que testou o SNMP e NETCONF em ambiente laboratorial, usando uma fonte aberta com implementação do servidor Yencap. Nas figuras 2 e 3 podemos observar os resultados mais relevantes dos testes efetuados.

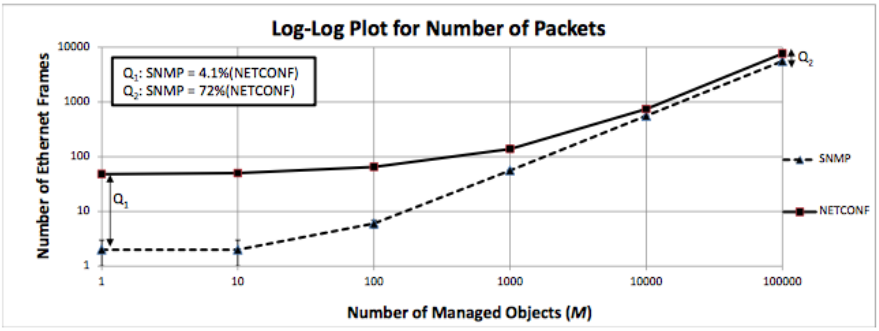


Figura 2 – Comparação de transmissão de pacotes de dados entre SNMP e NETCONF [28]

Operation on # of Managed Objects (*M*)	Number of Ethernet Frames		Standard Deviation	
	SNMP Protocol	NETCONF Protocol	SNMP Protocol	NETCONF Protocol
	SET	<edit-config>	SET	<edit-config>
1	2	48	0	0
10	2	50	0	0
100	6	65	0	0
1000	56	138.2	0	2.852
10000	557.8	740.3	0.616	23.295
100000	5558	7622.4	0	82.289

Figura 3 – Resultados obtidos no teste [28]

Repare-se que à medida que o número de objetos incluídos numa sessão de configuração aumenta, o SNMP e o NETCONF tendem a utilizar o mesmo número de *frames* Ethernet para transportar os dados de configuração, ainda que o SNMP utilize sempre um

número inferior. Para um número inferior de objetos a configurar, o NETCONF requer um número de *frames* Ethernet muito mais elevado do que o SNMP. Ou seja, o estudo concluiu que o NETCONF exige uma maior sobrecarga de processamento de PDUs nos agentes e nos gestores e que também gera mais tráfego na rede, requerendo uma maior largura de banda.

Por outro lado, concluiram que o tempo gasto na implementação da operação de configuração transmitida e o número de transações protocolares são menores para o NETCONF quando o número de objetos (não de operações de configuração) é grande. No entanto, este tipo de resultado é díficil de validar no primeiro caso – porque não se pode comparar diretamente a eficácia das implementações dos agentes utilizados, isto é, não se pode saber se a diferença de performance é devida às regras protocolares ou à qualidade da implementação – e o segundo resultado já era de esperar porque as mensagens atómicas SNMP são encapsuladas em UDP enquanto as sessões NETCONF utilizam o protocolo TCP.

Como conclusão final pode argumentar-se que o NETCONF é uma solução alternativa viável para as funções de configuração de redes complexas, em que o número de objetos por operação é muito elevado. Neste contexto pode ser vantajoso trocar a maior complexidade da implementação do NETCONF por um eventual ganho de perfomance no processamento dos objetos a configurar.

Assim, no contexto funcional de diganóstico/monitorização biomédica em ambiente hospitar, parece óbvio que a arquitetura mais indicada é a baseada no protocolo SNMP uma vez que não existem problemas de escabilidade (a quantidade de nós de agregação dos dados sensoriais que implementarão os agentes SNMP é relativamente baixa) e que a maior parte das interações operacionais serão de monitorização. Além disso, mesmo tendo em consideração que a quantidade de dados sensoriais recolhidos pelos agentes nos nós de agregação possa ser elevado, a sua transmissão através do SNMP nas redes locais das instituições hospitalares não será problemática devido às larguras de banda normalmente disponíveis.

3. SOLUÇÃO PROPOSTA

Num contexto de aplicação dum sistema genérico de diganóstico/monitorização biomédica em ambiente hospitalar, um ou mais pacientes são diagnosticados/monitorizados numa sala/quarto adequado para o efeito, sendo utilizados um ou mais sistemas sensoriais biomédicos por paciente. A instrumentação física costuma ser colocada diretamente no paciente (ou na sua proximidade) e está depois ligada a um dispositivo de diagnóstico/monitorização com interface integrado que será utilizado pelo pessoal especializado (enfermeiros e médicos). Em geral, por cada tipo e senor biomédico é necessário um sistema completo de diagnóstco/monitorização, incluindo a componente de interface. Nos produtos mais avançados, além dos dados analisados em tempo real pelo pessoal médico, a componente de interface permite recolher os dados sensoriais num suporte de armazenamento não volátil para posterior análise, ou até mesmo, recolher esses dados diretamente para uma base de dados desenvolvida pelo fabricante do sistema. Raramente é possível a integração desses dados numa base de dados global da instituição. Também não é possível que o resultado da monitorização de diferentes tipos de sensores, ou do mesmo tipo mas a serem utilizados em pacientes diferentes, possam ser integrados no mesmo sistema e analisados no mesmo interface. É raro, até, que os sistemas estejam interligados entre si através de tecnologias de rede local, muito menos através de aplicações de gestão distribuída.

Neste tipo de serviços, é necessária a presença física do pessoal médico junto do interface de cada sistema de monitorização para que um diagnóstico possa ser efetuado e eventuais decisões médicas possam ser tomadas.

Todas estas caraterísticas tornam este tipo de serviço médico muito caro (tanto em termos de equipamentos como em termos de recursos humanos) e com limitações na sua funcionalidade (por exemplo, o armazenamento não volátil dos dados recolhidos traz consigo obrigações/requisitos de segurança que, normalmente, as administrações das instituições preferem não enfrentar, pelo que, mesmo sendo tecnicamente possível, optam por não o implementar).

A solução que se propõe nesta dissertação parte duma arquitetura que está ilustrada na figura 4 e tem como base tecnológica a aplicação do modelo de gestão SNMP a este contexto de engenharia aplicada.

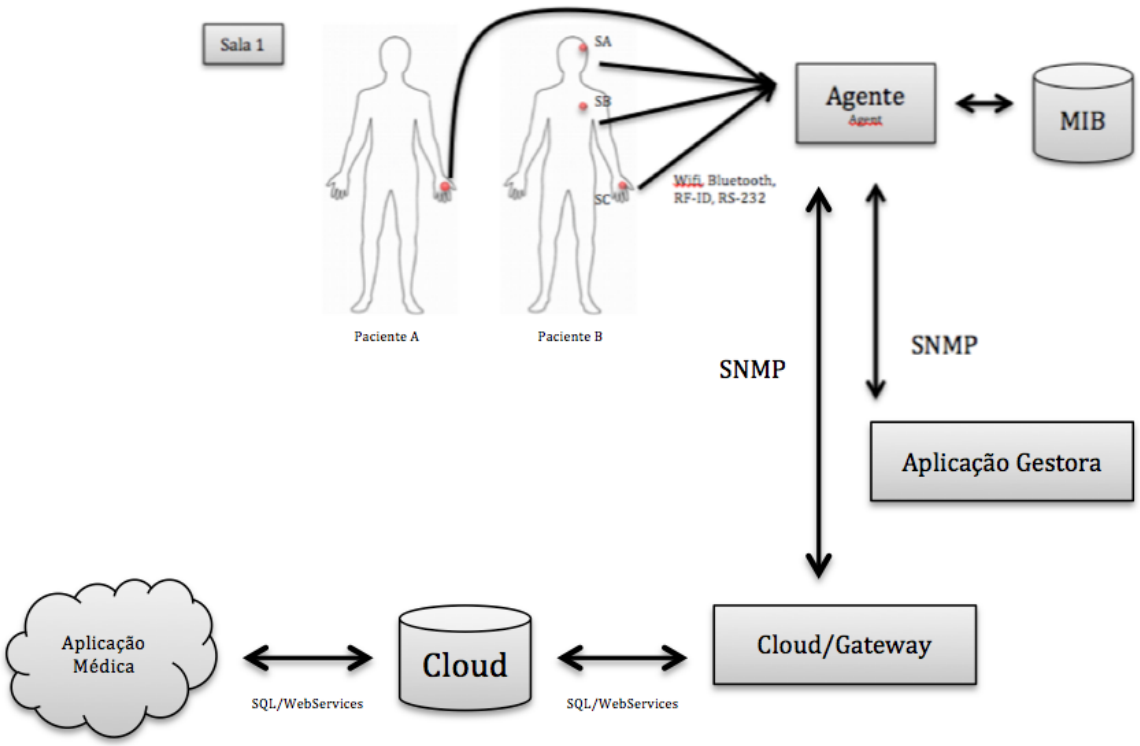


Figura 4 - Arquitetura geral da solução proposta

Neste modelo genérico, a unidade hospitalar será composta por várias salas onde um ou mais pacientes serão diagnosticados/monitorizados através dum sistema integrado de recolha e processamento de dados sensoriais. Os pacientes poderão ser monitorizados através da instrumentação física dum ou mais tipos de sensores (na figura 4 o paciente A tem colocado a instrumentação dum sensor genérico e o paciente B a instrumentação de três sensores específicos, SA, SB e SC). Todos estes sensores estão ligados a um sistema computacional com a função de nó agregador, utilizando tecnologias de comunicação de curto alcance, cabladas ou não (USB, RS-232C, Bluetooth, Wifi etc.). Em cada sala deverá existir um ou mais destes nós agregadores, dependendo do número de doentes e da quantidade e tipos de sensores usados (poderá até um nó agregador ser utilizado pelos sensores de mais do que uma sala), que estarão ligados ao sistema através da rede global da instituição e, através desta, à Internet. Em cada nó agregador estará embebido um módulo de *sofware* que executa a pilha protocolar TCP/IP e um agente SNMP com uma implementação duma MIB especial de agregação de dados biomédicos expressamente desenvolvida no contexto desta dissertação

Esta MIB permite gerir, duma forma normalizada e automática, o processo de recolha de informação bruta dos sensores biomédicos – desde as características independentes dos sensores (voltagem, amperagem, fabricante, unidades de medidas) até aos valores estatísticos relativos aos dados recolhidos (quantidade de amostras, tempos de recolha de dados, valores das amostras, etc.). Além disso, a MIB contém também objetos que são a abstração de parâmetros genéricos e universais representando a configuração do funcionamento dos sensores. Desta forma será possível configurar remotamente, duma forma automática e autónoma, com o mínimo de intervenção humana possível, todos parâmetros operacionais da rede de sensores.

A MIB desenvolvida é suficientemente genérica e universal por forma a permitir que se recolham dados de qualquer tipo atual ou futuro de sensor biomédico. Os dados em si podem também ser de variados tipos e podem sofrer logo um tratamento estatístico de filtragem e validação no próprio nó agregador, espelhando os requisitos enumerados para esses objetos na definição da MIB. Como a agregação dos dados pode ser feita para mais do que um paciente, esta MIB define também os objetos necessários para se poderem associar os pacientes aos dados recolhidos, ainda que não inclua uma base de dados de informação biográfica ou de historial médico dos pacientes. Até por uma questão de segurança, existe apenas a capacidade de relacionar os dados a uma identificação unívoca do paciente na instituição. Serão depois as aplicações biomédicas gestoras que terão, em caso de necessidade, de aceder à base de dados biográfica e de historial médico implementada algures no sistema informático da instituição e acedida através da rede local.

O primeiro nível aplicacional da arquitetura definida na figura 4 é implementado por aplicações gestoras embebidas em sistemas computacionais ligados à rede TCP/IP da instituição. Estas aplicações comunicam através do protocolo SNMP com os nós agregadores que têm embebidos os agentes SNMP e implementarão uma primeira linha de aplicações de diagnóstico/monitorização biomédica em tempo real, com interface próprio e implementando processos o mais automática e autonomamente possível, minimizando a intervenção humana especializada. Para aumentar a eficiência desta solução, além de existir em cada sala um sistema computacional implementando esta primeira linha de aplicações biomédicas de diagnóstico/monitorização, poderá existir um centro de diganóstico/monitorização para toda a instituição (ou um por cada tipo de serviço hospitalar), que funcionará como uma segunda instância da primeira linha de aplicações de diagnóstico/monitorização biomédica e também

como uma segunda linha de aplicações de diagnóstico médico mais complexo, em que se tenha em conta os dados recolhidos na monitorização dos pacientes e os dados do seu historial clínico. Qualquer uma destas linhas de intervenção médica pode gerar alertas automáticos quando os valores dos dados recolhidos permitem concluir que o estado do paciente requer a intervenção do pessoal médico. Estes alertas podem ser feitos através do interface normal dessas aplicações e/ou através de outros meios de comunicação por mensagens (SMS, e-mail, etc.). Os níveis de automação/autonomia e de complexidade da análise médica do estado dos pacientes depende apenas da inteligência dos sistemas aplicacionais gestores, ou seja, na capacidade de criação dos produtores deste tipo de *software*, e não por limitações técnicas da arquitetura. Pode até verificar-se num futuro muito próximo que os níveis de automação/autonomia permitam que uma primeira linha de intervenção médica seja realizada diretamente nos pacientes sem intervenção humana.

A arquitetura contempla também uma ligação a um componente de *gateway* entre os agentes SNMP nos nós de agregação de dados de diagnóstico/monitorização e outros sistemas de maior escala (por exemplo, um sistema de base de dados em *cloud* para o sistema nacional de saúde) para armazenamento e posterior processamento. Estes dados estariam, por exemplo, disponíveis para utilização em projetos de investigação biomédica para o desenvolvimento de novas terapias (ou aperfeiçoamento das já existentes) e/ou de novos medicamentos ou, mais simplesmente, permitir com facilidade a obtenção de dados estatísticos vários sobre os pacientes atendidos nos serviços de saúde a nível regional ou nacional.

De salientar também que o modelo apresentado permite facilmente a adoção dos requisitos de segurança expectáveis para este tipo de sistema: autenticação, confidencialidade e verificação de conteúdo. Estes requisitos são atingíveis através da utilização da segunda versão do SNMP com mecanismos de segurança ou através da utilização da terceira versão do SNMP, cujas implementações já incluem obrigatoriamente esses mecanismos de segurança. Adicionalmente, devem os próprios sistemas computacionais onde são executadas as aplicações biomédicas, prever sistemas de prevenção a ataques informáticos. Por seu turno, as próprias aplicações biomédicas podem implementar níveis de segurança adicionais, inclusivé com a ajuda de sistema externos baseados em tecnologias especializadas, como por exemplo, a identificação dos pacientes e do pessoal médico através de *tags* de rádio frequência (RF-ID).

Repare-se que a arquitetura permite o desenvolvimento modular e independente dos seus componentes: instrumentação sensorial, nós de agregação com agentes SNMP e sistemas computacionais com aplicações gestoras de diagnóstico/monitorização com interface incluído ou não (podem até desenvolver-se aplicativos que implementem apenas o interface entre as aplicações biomédicas e o pessoal médico especializado, aumentando ainda mais a modularização de todo o sistema). Todos os componentes da arquitetura utilizam redes de comunicação comuns, implementando protocolos normalizados. Isto potencia a especificidade de cada componente, tornando o mercado mais competitivo, tanto em termos de custo como em termos da quantidade e qualidade das funcionalidades desenvolvidas, incluindo as de automação e autonomia.

Por fim, referir que a arquitetura proposta está de acordo com as recomendações definidas pelo ETSI para a área *eHealth* [8], nomeadamente nos requisitos fundamentais de interoperabilidade e segurança.

4. PROTÓTIPO

Para validar a estratégia da arquitetura proposta planeou-se a implementação dum sistema protótipo, modular e completo de raíz, incluindo o *hardware* e *software* dum sensor biomédico de baixo custo e dum nó de agregação de dados biomédicos com um módulo de agente SNMP incorporado. O agente teria que implementar a instrumentação da MIB especificada. Por fim, o sistema ficaria completo com a implementação duma simples aplicação gestora para diagnóstico/monitorização biomédica.

Por limitações temporais não foi planeado o desenvolvimento de nenhum componente de *gateway* entre o agente SNMP/nó de agregação e um sistema externo de *cloud*.

Antes de iniciar a construção do hardware dum protótipo de sensor biomédico ou na programação do agente SNMP, foi preciso especificar a MIB para gestão de redes de sensores biomédicos. Espera-se que esta especificação possa ser adotada sempre que alguém pretender desenvolver um agente SNMP para um nó de agregação de dados sensoriais biomédicos.

4.1 A MIB

Uma forma de definir os objetos monitorizados e os seus comportamentos é através da SMI [29].

A SMI apresenta duas versões: a primeira que define precisamente como os objetos monitorizados são identificados e especifica o seu tipo de dados associados (SMIv1); a segunda versão apresenta melhorias para o SNMP (SMIv2).

Esta especificação dos objetos monitorizados pode ser dividida em três campos:

1. Nome: o nome ou o *object identifier* (OID) define um único objeto;
2. Tipo e sintaxe: um objeto monitorizado é definido usando um subconjunto do ANS.1. O ASN.1 é uma forma de especificar a maneira como os dados são representados e transmitidos entre os agentes e os gestores;
3. Codificação: através do *Basic Encoding Rules* (BER) é possível definir a codificação dos objetos monitorizados através de uma String de octetos.

Para além da especificação, é importante compreender como é que estes objetos estão organizados. Para tal a SMI desenvolveu uma hierarquia em árvore que permite agrupar os objetos tal como se pode observar na figura 5.

Figura 5 - Árvore de objeto SMI incluindo a MIB-II [30]

O OID de cada objeto é composto por uma série de inteiros baseada em nós da árvore e separados por pontos. Por exemplo, o OID para *mgmt* é 1.3.6.1.2.

A árvore começa no seu *Root-Node* e é possível verificar que a MIB já vai na sua versão II. Para o caso em concreto da concepção da minha MIB, a implementação teve de ser realizada no domínio privado. Caso fosse publicada, é a *Internet Assigned Numbers Authority* (IANA) que atualmente gere todas as atribuições de números de empresas privadas para instituições, organizações, empresas, etc.

É importante mencionar que a cada OID corresponde um tipo de dado, dado este fundamental para a gestão da rede e dos dispositivos da mesma.

Estes dados classificam-se da seguinte forma como se pode observar na tabela 1:

Tabela 1 - Tipos de representação de dados de um objeto [30]

Tipo de dado	Definição
Integer	Usado para representar números inteiros.
Octet String	Armazena uma sequência de bytes.
Counter	Representa um contador que unicamente pode incrementar seu valor e ao chegar em seu valor máximo, volta a zero.
Object Identifier	Usado para representar os identificadores dos objetos, isto é, a posição de um objeto dentro da árvore da MIB.
Null	Representa a ausência de um valor. Atualmente não é utilizado no SNMP.
Sequence	É uma estrutura de dados, ou seja, uma lista ordenada de tipos de dados diferentes.
Sequence of	É uma lista ordenada de tipos de dados iguais. É parecido com ao tipo "SEQUENCE", exceto que todos os tipos têm de ser iguais.
IpAddress	Representa um endereço Ipv4 de 32 bits.
NetworkAddress	O mesmo que "IpAddress", mas pode representar tipos de endereços de redes diferentes.
Gauge	Pode incrementar ou decrementar, mas nunca pode exceder seu valor máximo.
Timeticks	É um tipo de dados usado para medir tempos.
Opaque	Permite que qualquer outra codificação ASN.1 possa ser colocada em um Octect String.

É essencial esta atribuição a cada objeto pois só assim é que é possível a construção de uma MIB lógica de objetos monitorizados e corretamente definidos.

Um agente pode implementar muitas MIBs, mas todos eles implementam uma em particular – a MIB-II (RFC 1213). Esse padrão define as variáveis para coisas como estatísticas de interfaces assim como várias coisas relacionadas ao próprio sistema. O objetivo principal da MIB-II é fornecer informações gerais do TCP/IP [30].

4.2 MIB para gestão de dados sensoriais biomédicos

Do estudo das funcionalidades de sensores biomédicos já existentes no mercado e da consulta a enfermeiros, médicos e engenheiros biomédicos, concluiu-se que existem muitos sensores distintos e multifuncionais, isto é, chegam a "sobrepor-se" no tipo de amostras que medem e registam.

Como principais tipos de sensores temos o eletrocardiograma (ECG), o oxímetro de pulso, o sistema de monitorização contínua de glucose/glicose e medidores de pressão intracraniana. O ECG é o método complementar mais simples e mais importante no diagnóstico de doenças cardiovasculares, pois é um método não evasivo, barato e amplamente disponível. Este regista a atividade elétrica produzida pela ativação miocárdica, sendo o seu traçado constituído por várias deflexões causadas pela ativação, despolarização e repolarização das células [31]. A oximetria de pulso regista a quantidade de oxigénio existente no sangue e é um teste não invasivo, não doloroso, fácil de executar e com grande fiabilidade na avaliação da hipoxémia [32]. Em geral, um sistema de monitorização contínua de glucose/glicose mede os níveis de glucose/glicose a cada 10 segundos e armazena a média destas leituras em intervalos de cinco minutos [13, 33]. A PIC consiste num minúsculo sensor colocado logo abaixo da pele da cabeça do paciente e de um monitor externo especial para receção e análise das informações. O aparelho avalia o volume do líquido cefalorraquidiano, substância que reveste e protege o sistema nervoso central contra impactos, e também a concentração de sangue e a massa cerebral, entre outros fatores de risco para o aumento da pressão [34]. Um outro tipo importante de sensor biomédico é conhecido como sensor multiparâmetros e regista, em simultâneo, leituras de diferentes instrumentações sensoriais simples. Existem centenas de variações destes aparelhos. No entanto, todos captam alguns parâmetros em comum como o ECG, a frequência cardíaca, o débito cardíaco, a frequência de respiração, a temperatura e capnografia, etc. Nem sempre o uso de um aparelho multifunções é aconselhado. Quando, por exemplo, é necessária uma avaliação de oxigénio no sangue de um paciente cujo estado clínico não seja grave, basta utilizar um oxímetro de pulso.

Este tipo de informação permitiu definir as primeiras tabelas da MIB que podem ser verificadas nas três tabelas seguintes (tabelas 2, 3 e 4).

Tabela 2 - Tipo de sensor

SensorType	***Description***
1	ECG
2	Oxímetro de pulso
3	Pressão intracraniana
4	Multiparâmetros
5	Glucose/Glicose
6	fluxo sanguíneo
7	UNIVERSAL

Tabela 3 - Descrição das medições de cada sensor

SampleOrParametersType	*Description*
1	Freq cardíaca
2	onda P
3	intervalo PR;
4	complexo QRS
5	eixo elétrico
6	saturação arterial de oxigénio (SpO2)
7	Pressão intracraniana;
8	Temperatura;
9	tensão arterial
10	taxa de respiração
11	capnografia
12	débito cardíaco
13	temperatura de injeção
14	temperatura sanguínea
15	EEG
16	pressão invasiva
17	pressão não invasiva
18	pressão sanguínea não invasiva (NIBP)
19	glucose
20	Velocidade; fluxo sanguíneo (coagulação)
21	?
1001	Temperatura em Cº
1002	Temperatura em Fº
1003	Voltagem
1004	Corrente
1005	Freq. monitorização

Tabela 4 - Descrição das unidades medidas

UnitsType	*Description*
0	Cº
1	Fº
2	Volts
3	Amperes
4	Hz
5	Kg
6	Segundos
7	Horas
8	Milímetros
9	Percentagem
10	Pascal
11	Batimentos Por Minuto
12	Graus
13	Milímetros de mercúrio
14	LPM
15	Miligramas por decilitro

Relativamente aos números da tabela *Sample Or Parameters Type*, não são todos consecutivos devido à seguinte convenção adotada: de 1 a 21 temos as descrições relativas ao que é medido pelos sensores, de 1001 até 1005 temos as descrições que fazem parte do aparelho em si e que não interferem com as amostras recolhidas.

O tipo de tecnologia de ligação da instrumentação do sensor também é um parâmetro de funcionamento importante, ainda que independente dos valores das amostras recolhidas. Estes dados são geridos através da tabela *Interface Type* (tabela 5).

Tabela 5 - Descrição das diferentes interfaces

InterfaceType	*Description*
1	*Bluetooth*
2	*wifi*
3	*RS232*
4	*USB*
5	*LAN*
6	X

Os sensores possuem ainda mais algumas caraterísticas adicionais e universais e, mais uma vez, independentes do processo de amostragem. A informação dessas caraterísticas pode ser gerida através da tabela *sensors* (tabela 6).

Tabela 6 - Informação exclusiva sobre os diferentes tipos de sensores

SensorIndex	*Description*	*Doctor/Nurse*	*Location*	*Manufacturer*
1	O Eletrocardiograma é um exame que permite a avaliação elétrica da atividade cardíaca (potenciais elétricos) […]	*John Walker*	*Room 14 3rd Floor*	http://www.techplaza.pt/Punhos/Olympus-ECG-1-Grip-para-E-M10-P145246.html
2	Mede indiretamente a quantidade de oxigénio no sangue de um paciente […]	*Elise Mutter*	*Room 14 3rd Floor*	http://www.quirumed.com/pt/eletromedicina/oximetros-de-pulso?gclid=COO-jIX578gCFSMYwwodLkgH2g
3	Pressão intracraniana (PIC) é a pressão exercida pelo crânio sobre o tecido cerebral, […]	*Elise Mutter*	*Room 15 3rd Floor*	http://www.dabasons.com.br/index.php/produtos/neurocirurgia
…	…	…	…	…

Para saber que tipo de interface tem cada sensor, especificou-se a tabela relacional *Interface & Sensor* (tabela 7). Assim, para a primeira linha da tabela, por exemplo, o valor 1 referente à tabela *sensors* e o valor 1 referente à tabela *Interface Type* indicam que o sensor 1 (de que o John Walker é responsável no quarto 14, etc.) tem uma ligação por *Bluetooth* e que se encontra no estado *On*.

Tabela 7 - Relação entre sensor, o seu tipo de interface e ainda informação adicional

InterfacesSensorsIndex	*SensorIndexValue*	*InterfaceIndexValue*	*Error*	*Status*
1	1	1	-	*On*
2	1	2	1	*Off*
3	2	1	-	*On*

É necessário sabermos o que é que cada sensor recolhe/amostra. Assim, existiu a necessidade de criar outra tabela, apenas relativa às amostras: *Samples Table* (tabela 8).

Tabela 8 - Relação entre sensor e o tipo de amostra recolhida

SampleIndex	*PatientSensorsIndexValue*	*NValuesInSample*	*InitialTIme*	*FinalTime*	*FirstValueIndex*
1	1	5	2015-11-01 / 17:29:43	2015-11-01 / 17:31:43	2
2	2	2	2015-11-01 / 17:25:50	2015-11-01 / 17:31:50	3
3	3	2	2015-11-01 / 18:29:43	2015-11-01 / 17:31:50	6
...	...	...	...	...	...

Esta tabela é composta por seis colunas. A primeira coluna corresponde ao índex da tabela. A segunda relaciona a amostra recolhida com o paciente a que pertence. Esta coluna corresponde à chave primária de uma tabela auxiliar apresentada mais adiante. A terceira coluna devolve quantos valores da amostra foram lidos quando existem sensores que podem recolher e devolver mais do que um valor por amostra. A quarta e a quinta coluna são referentes ao tempo de medição. É essencial saber quando começou e terminou a recolha da amostra. A última coluna funciona como apontador para a tabela *Values Table* (tabela 10). Esta coluna aponta para o primeiro valor da amostra (ver tabela *Values Table* para uma melhor compreensão).

Assim como existem amostras recolhidas pelos sensores, também existem parâmetros de funcionamento. A gestão deste tipo de informação é conseguida através da manipulação dos objetos da tabela *Parameters Sensor Table* (tabela 9).

Tabela 9 - Relação entre sensor e o tipo de parâmetro recolhido

ParameterIndex	*SensorIndexValue*	*Description*	*NValuesInParameters*	*FirstValueIndex*
1	1		1	5
2	2		3	7
3	2		2	8
...	...	...	...	...

Esta tabela é idêntica à das amostras, mas, em vez dos tempos, temos uma coluna para a descrição do parâmetro. A próxima tabela *Values Table* (tabela 9) relaciona estas duas.

Tabela 10 - Tabela que nos fornece os valores recolhidos (quer sejam amostras ou parâmetros)

ValuesIndex	**NextSampleOrParameterIndexValue**	**SampleOrParametersTypeID**	**UnitsTypeID**	**Precision**	**Value**
1	3	1	1	0	130
2	1	2	5	1	128
3	4	3	12	-1	012
...	...	...	...	...	...

A primeira coluna corresponde ao *Index*. A segunda coluna, tal como o próprio nome indica, aponta para o número de ordem correspondente contido no *índex*, tal como se pode observar. Tem um funcionamento muito simples: os números das colunas *FirstValueIndex* das tabelas anteriores - *Samples Table* e *Values Table* – são colocados nesta coluna que apontam para o *Index* desta tabela. Assim, basta analisar as entradas duma linha para se perceber se se trata duma amostra ou de um parâmetro, que tipo de amostra ou parâmetro é, qual o seu valor, as unidades respetivas e qual a precisão dos valores recolhidos.

Como já foi referido anteriormente, é necessário saber-se a que paciente está associado um determinado sensor. Para tal foi criada uma tabela exclusiva dos dados básicos dum paciente: *Pacients* (tabela 11). De notar que o preenchimento dos valores desta tabela pode ser feito automaticamente por consulta direta à base de dados biográfica e de historial clínico dos pacientes da unidade hospitalar (caso esta esteja implementada e seja acessível por tecnologias normalizadas).

Tabela 11 - Informação relativa ao paciente

PacientIndex	*PacientLabel*	*Gender*	*Name*	*Situation*	*Age*
1	14124	M	Duarte	Doente diabético...	60
2	12414	F	Rita	Doente com cancro linfático	38

Na tabela *Sensors* (tabela 12) foi usada uma coluna para identificar qual o sensor ou os sensores que estão a monitorizar um paciente. Essa coluna é nada mais do que a chave primária entre a combinação de duas colunas: *SensorIndexValue* e *PacientIndexValue*. A primeira coluna faz parte da tabela *Sensors* e a segunda da tabela *Pacients.*

Tabela 12 - Relação entre paciente e sensor

SensorIndexValue	*PacientIndexValue*	*PacientSensorIndex*
1	14124	1
1	12414	2
2	64532	3

Por último, foi criada uma tabela que relaciona o sensor a ser usado com o seu tipo.

Tabela 13 - Classificação do tipo de sensor

SensorTypeIndex	*SensorIndexValue*	*SensorTypeID*
1	1	1
2	1	2
3	2	1

4.3 Implementação da MIB no MIBDesigner

A ferramenta escolhida para compilar a MIB especificada foi o MIBDesigner. Este programa é simples e eficaz na verificação sintática duma especificção duma nova MIB.

Na imagem da figura 6 apresenta-se a MIB desenvolvida. Como foi explicado previamente, a MIB é constituída por várias tabelas com várias entradas cada. Além disso, foi igualmente necessário incluir várias convenções textuais. As convenções textuais permitem redefinir tipos para lhes dar uma semântica mais específica [12].

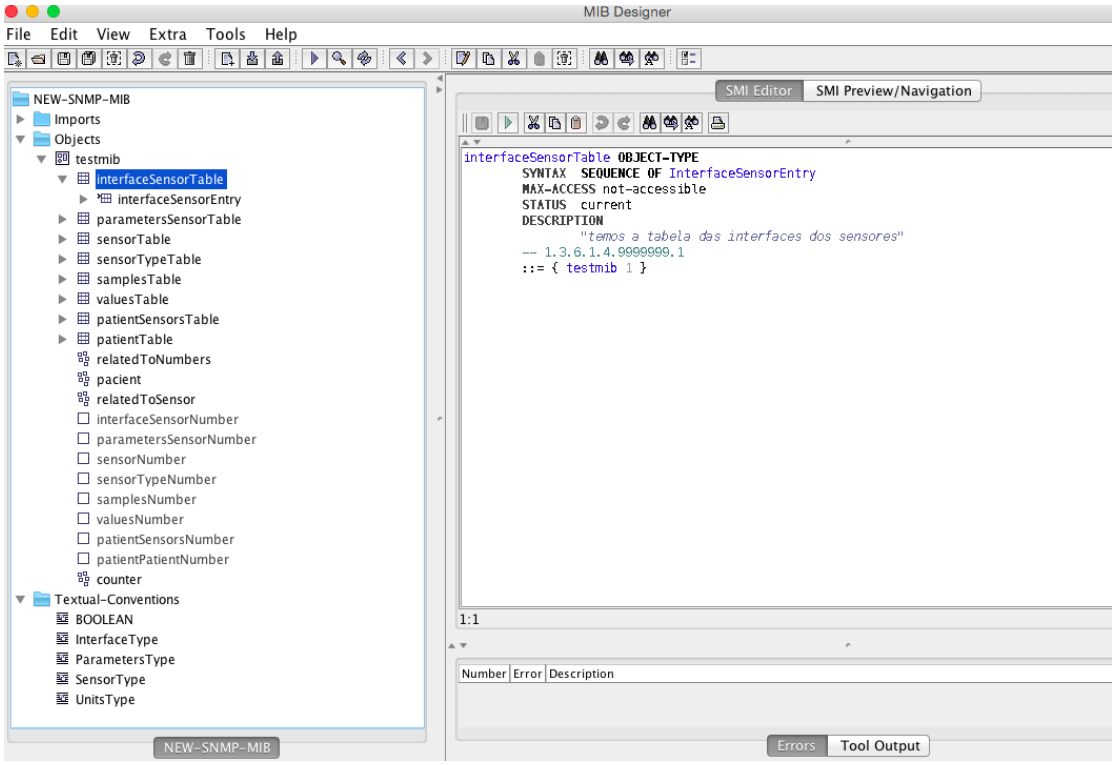


Figura 6 – Verificação sintática da MIB no programa MIBDesigner

Conforme se pode ver na especificação do lado direito da imagem da figura 6, a definição da tabela *interfaceSensorTable* possui várias entradas: o *MAX-ACCESS* nas tabelas é "not-accessible" [35] e o seu *STATUS* foi definido, nesta altura, como *current*. O valor *current* significa que a definição é atual e válida.

Como exemplo do processo de verificação da MIB nesta ferramenta, apresenta-se a imagem da figura 7, ilustrativa do preenchimento da informação das colunas da tabela *interfaceSensorTable*.

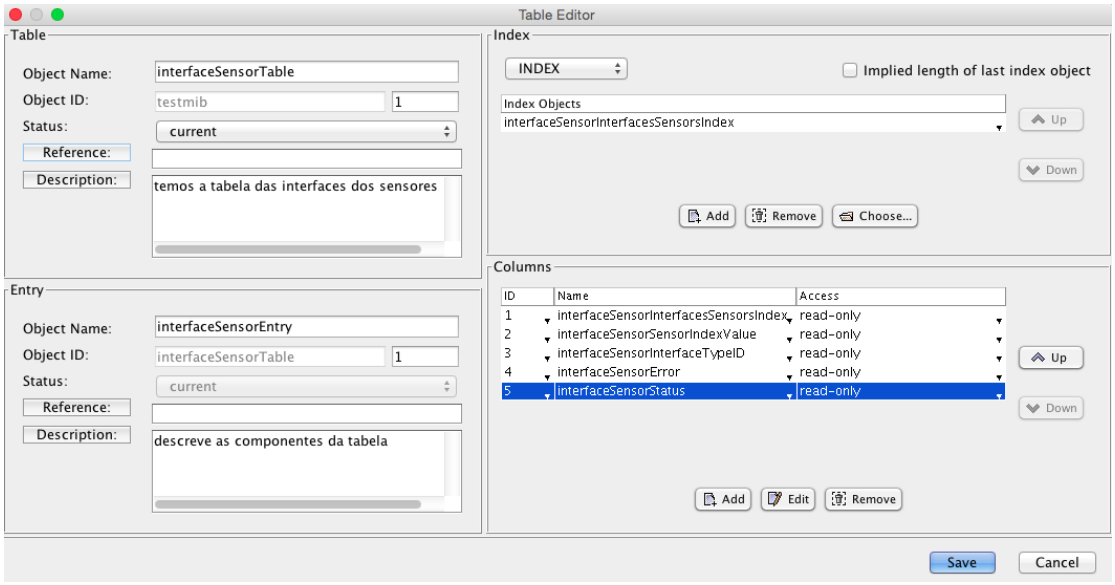


Figura 7 - Preenchimento das colunas constituintes da tabela interfaceSensorTable

Quando os campos forem preenchidos corretamente, observamos que o programa gera automaticamente a estrutura de dados correta. A operação é repetida para todas as outras entradas nas tabelas (ver figura 8).

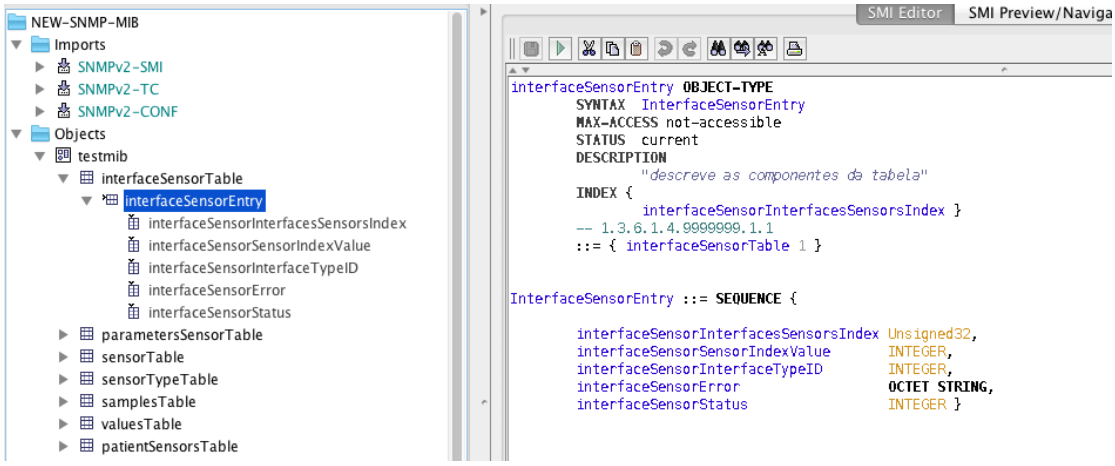


Figura 8 – Visualização da estrutura integral de cada tabela

Como é possível observar nas entradas da tabela, existem diversos tipos SMI para caracterizar cada objecto.

Após introduzirmos a especificação de todas as tabelas é necessário ainda criar grupos. As MIBs são organizadas em grupo hierárquicos de conjuntos de objetos (que pode ser tabelas). Os grupos servem, como o próprio nome indica, para agrupar os objetos que estejam de alguma forma relacionados, normalmente em termos funcionais.

Todos os objetos criados para esta MIB foram agrupados em 4 grupos, como se pode observar na figura 9.

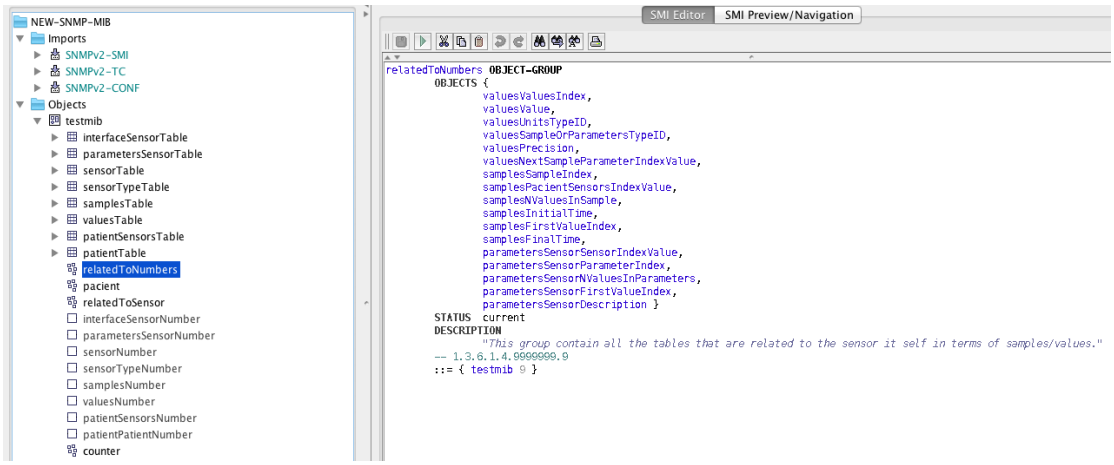


Figura 9 – Criação de grupos

Criou-se um grupo que contém apenas os objetos das tabelas relativas às caraterísticas universais dos sensores (*relatedToNumbers*), outro relativo aos dados dos pacientes (*pacient*), outro com os valores amostrados pelos sensores (*relatedToSensor*) e, por último, um grupo apenas com os índices de todas as tabelas (*counter*).

Como já foi referido, a especificação da MIB inclui a definição de algumas convenções textuais. Na figura 10 apresenta-se uma dessas convenções textuais – *InterfaceType* – que não é mais do que a enumeração dos possíveis tipos de tecnologia de interface que a instrumentação dos sensores podem usar.

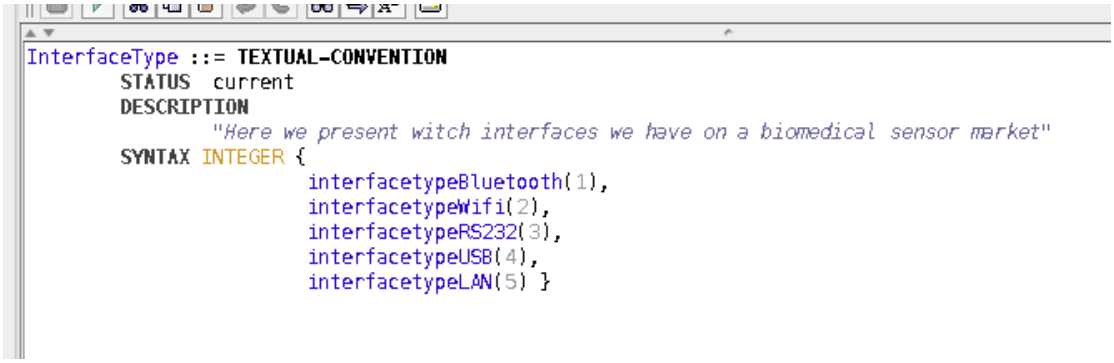


Figura 10 – Exemplo de uma convenção textual (InterfaceType)

O MIBDesigner verifica a sintaxe da MIB e gera automaticamente a estrutura correta de todos os *imports* necessários para a MIB ser compilada. Além disso, o programa gera, também, um módulo de identidade, isto é, apresenta, entre outros, os dados do autor da MIB.

Depois de verificada a correção sintática da MIB desenvolvida, foi preciso compilá-la de forma a gerar um "pseudo-código" em java para ser usado mais tarde na programação do módulo do agente SNMP. A ferramenta escolhida foi o Agenpro, versão 4.3.0.

A especificação completa e verificada da MIB pode ser consultada no Anexo I.

4.4 APIs SNMP

Para ajudar no desenvolvimento das componentes duma arquitetura de gestão SNMP existem algumas ferramentas disponíveis, sob a forma de *Application Programming Interface* (API), algumas sem custos associados. Destacam-se as três mais mais usadas pela comunidade científica de R&D da área de gestão de redes: *Simple Network Management Protocol for java* (SNMP4J), WebNMS SNMP API e iReasoning SNMP API.

SNMP4J

Esta API é a mais utilizada e a que possui mais suporte da comunidade. É ainda aquela que possui o pacote gratuito mais completo. É usada para todas as versões do SNMP, suporta todos os tipos de PDUs e fornece um modelo de *timeout* (*pluggin)*, tem suporte para aplicações *multi-threading* e para pedidos síncronos e assíncronos. Existe imensa informação online sobre esta API e o único requisito necessário é a instalação do Java 1.4.1 ou versão posterior [36].

WebNMS SNMP API

Tal como o SNMP4J, também esta API suporta todas as versões do SNMP. Suporta ainda IPv6, permite o envio de pacotes SNMP em *broadcast* para a descoberta automática de dispositivos SNMP na rede, permite ainda um carregamento das definições das MIBs através de um ficheiro pré-compilado, serializado ou até de uma base de dados para aumentar a performance das aplicações desenvolvidas. Esta API ainda tem como vantagem promover a

escalabilidade das aplicações guardando as definições das MIBs e as configurações SNMPv3 numa qualquer base de dados relacional. Como principal requisito tem a instalação do *Java Development Kit* (JDK) 1.2 ou versão posterior [37].

iReasoning SNMP API

Assim como as duas APIs anteriores, também esta suporta todas as versões do SNMP. Esta API permite o uso de algoritmos de encriptação *Data Encryption Standard* (DES) e *Advanced Encryption Standard* (AES) 128-bit. Suporta IPv6, TCP e UDP para a camada de transporte, fornece código de exemplos completos para todas as operações SNMP. Contudo, não permite construir um agente sem usar um pacote super iReasoning SNMP API, o iReasoning Agent Builder. O requisito para esta API é a instalação do JDK 1.3 ou versão posterior [38].

Da análise comparativa das diferentes APIs resultou a tabela 14 de avaliação das diversas características. Foi definida uma escala de pontuação de 1 a 3, com os valores mais altos a significar melhores performances.

Tabela 14 - Comparação de APIs

APIs	Características funcionais e compatibilidade	Segurança	Comunidade ativa	Requisitos	Pontuação Média
SNMP4J	3	3	3	3	**3**
WEB NMS	2	3	3	3	**2.75**
iReasoaning	2	2	2	3	**2.25**

Ainda que o estudo não tenha sido muito rigoroso, foi suficiente para se decidir utilizar as APIs do SNMP4J.

4.5 Arquitetura de *software* do protótipo

O *software* desenvolvido para o sistema diagnóstico/monitorização é composto por diversos componentes modulares que, em conjunto implementam o nó de gregação com o agente SNMP e a aplicação gestora para monitorização biomédica.

O esquema lógico de interligação/relação entre esses componentes está ilustrado na figura 11.

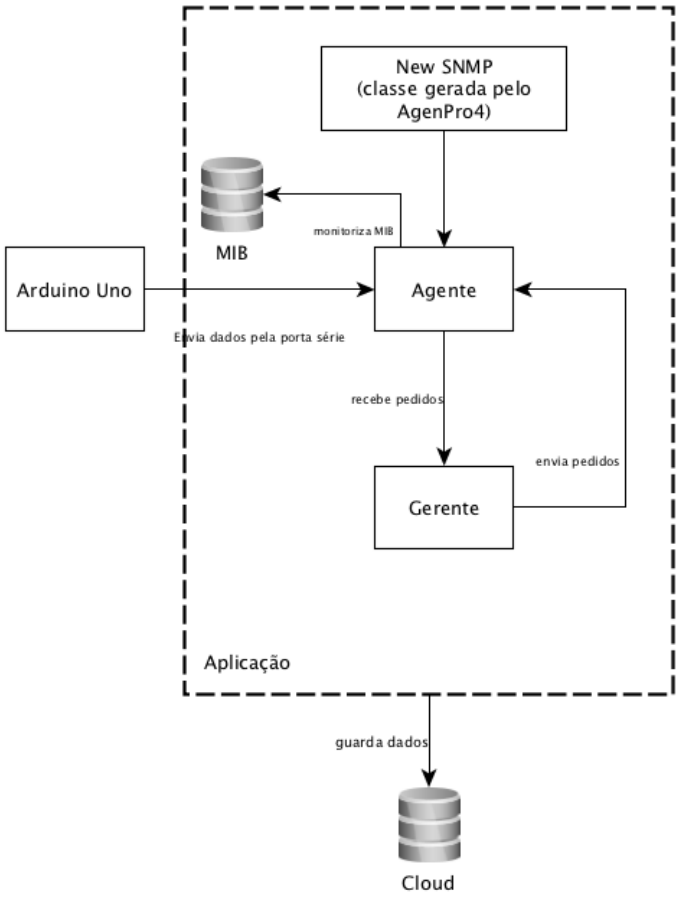


Figura 11 – Esquema lógico dos componentes de *software*

Adicionalmente, foi desenvolvida a instrumentação dum sensor biomédico recorrendo a criação dum circuito electrónico montado num sistema Arduino. Neste caso, tratou-se dum fluxímetro ótico que serve para medir valores de fluidez sanguínea (e, a partir desta, valores de probabilidade de hipercoagulação do sangue). No final deste capítulo pode consultar-se uma seção dedicada ao desenvolvimento deste sensor biomédico.

É necessário estabelecer-se uma ligação USB entre a instrumentação física do sensor e o sistema computacional onde residem todos os outros componentes de *software* do sistema protótipo. O *software* desenvolvido para o Arduino permite a recolha dos valores lidos no nó de gregação de dados biomédicos através duma conexão de porta série (ver fluxograma da figura 12).

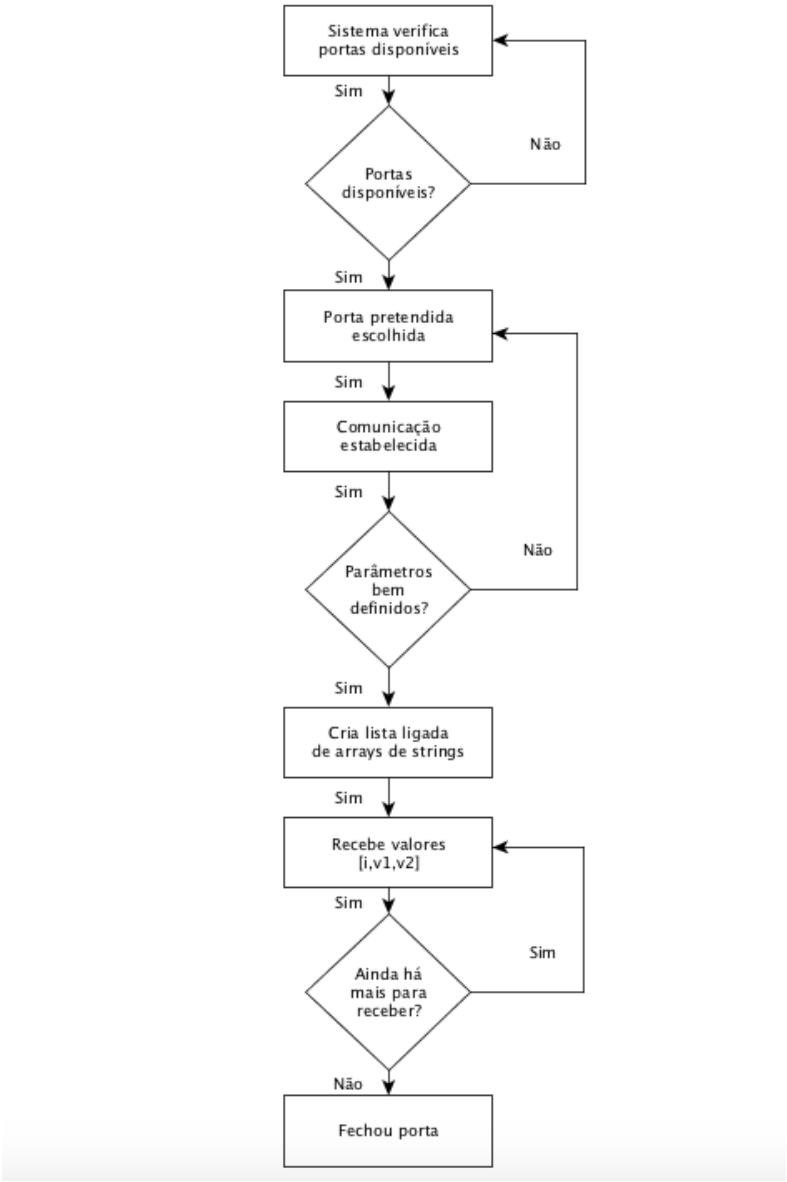


Figura 12 – Fluxograma da recolha de dados biomédicos

A máquina onde está inserido todo o *software* verifica todas as portas disponíveis às quais é possível fazer a ligação USB. Imediatamente, contecta-se automaticamente à entrada USB do arduino uno.

De seguida, da parte do arduino, estabelecem-se parâmetros importantes tais como a geração de sinais de uma onda quadrada a 10 KHz. Caso contrário o circuito não funciona.

Consecutivamente, o arduino envia o par de valores recolhidos pelo sensor, *v1*,*v2* e ainda um índex *i* para sabermos qual o número da amostra.

Caso não hajam mais amostras a serem lidas e recolhidas, a ligação é interrompida. Toda a explicação mais detalhada de como se processam/recolhem os dados no arduino está explicada no subcapítulo 4.8.

4.6 Implementação do agente SNMP

Para a criação de um agente SNMP utilizando a API SNMP4J é necessário estender a classe *BaseAgent*. Esta classe é uma classe abstrata que nos fornece uma espécie de esqueleto para os nossos agentes. Trata-se de uma classe abstrata e, para o protótipo, foi necessário implementar os métodos registerManagedObjects(); unregisterManagedObjects(); initTransportMappings(); addViews() e addCommunities(). Estes métodos tratam da inicialização do sistema com a configuração das comunidades SNMP e das vistas para controlo de acesso aos objetos da MIB.

Consulte-se o fluxograma da figura 13 para verificar o processo lógico implementado no agente SNMP.

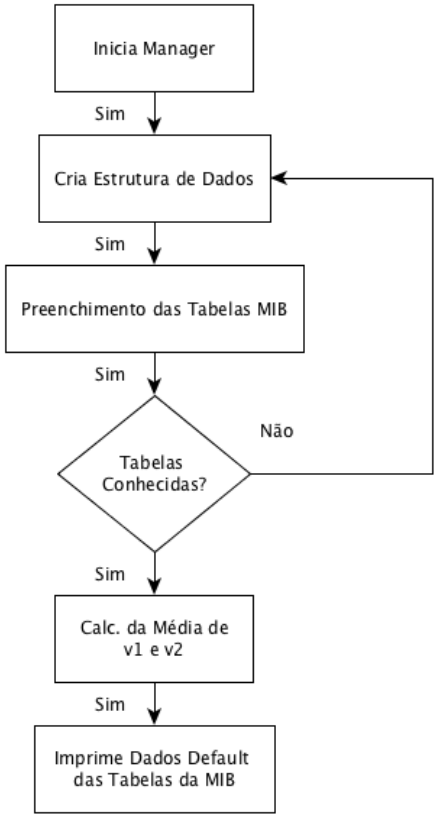


Figura 13 – Fluxograma do agente SNMP

No módulo do agente foi criada uma estrutura de dados com o intuito de ajudar a preencher todas as tabelas da MIB com pelo menos uma entrada. Para estes dados serem inseridos é necessário conhecer-se a estrutura das tabelas, isto é, quantas e quais os tipos de colunas de cada tabela. Assim, foi criado um array de arrays que representam as tabelas, as

suas entradas e os seus tipos. Desta forma consegui-se introduzir uma entrada para cada tabela, preenchendo todas as colunas, como se pode observar no exemplo incompleto da figura 14.

```
public void populate(){
    Variable[] rowsPreset[]={ //crio um array de arrays e cada array é um rowpreset
        //crio uma estrutuda de dados
        new Variable[]{new Integer32(1),new Integer32(1),new Integer32(1),new OctetString(),new Integer32(1)/
        new Variable[]{new Integer32(1),new Integer32(1),new OctetString("este parametro le a temperatura"),n
        new Variable[]{new Integer32(1),new OctetString("mede quantidade de O2,Ricardo Caetano"), new OctetSt
        new Variable[]{new Integer32(1),new Integer32(1),new Integer32(1)},//SensorTypeEntry -3
```

Figura 14 – Introdução dos campos de algumas tabelas da MIB

Apesar de se definir nesta estrutura quantos campos e quais os seus tipos, não se sabe de antemão a que tabela pertencem. O problema foi resolvido com a implementação de um *hashmap.*

Para além dos dados *default* introduzidos na MIB, é preciso também introduzir os valores resultantes do processo de amostragem na instrumentação do sensor biomédico (ver figura 15).

```
public void readCorrelacao(LinkedList bloodList){ //OID index.1 [index,v1,v2]
                                                   //OID index.2 [index,v1,v2]

    ListIterator<String[]> listIterator = bloodList.listIterator(); //pego na lista ligada e itero
    String[] aux;
    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd/HH:mm:ss.SSS"); //milesimo de segundo
    DefaultMOMutableTableModel modelValues = (DefaultMOMutableTableModel) this.agent.getMib().getVa
    DefaultMOMutableTableModel modelSamples = (DefaultMOMutableTableModel) this.agent.getMib().getS

    synchronized (this) {
        //model.clear();//remove Rows
        while(listIterator.hasNext()){ //para cada linha, adiciono uma row (enquanto tiver proxima)
            aux = listIterator.next();

            //FILTRO IMPORTANTE!
            if(aux!=null && aux.length==3 && aux[0]!=""){ //se houver campos, se forem 3, se o camp
            MOTableRow row = modelValues.createRow(new OID(aux[0]+".1"), new Variable[]{ //crio a l
                new UnsignedInteger32(Integer.parseInt(aux[0])), //o aux[0] é sempre o mesmo do OID
                new Integer32(),
                new Integer32(),
                new Integer32(),
                new Integer32(),
                new OctetString(aux[1]) //crio a tabela e digo só que tipos é que sao e o v1 é do t
            }
            );
            MOTableRow row2 = modelValues.createRow(new OID(aux[0]+".2"), new Variable[]{ //faço o
                new UnsignedInteger32(Integer.parseInt(aux[0])),
                new Integer32(),
                new Integer32(),
                new Integer32(),
                new Integer32(),
                new OctetString(aux[2])
```

Figura 15 – Armazenamento de valores amostrados na MIB

Um exemplo do resultado do preenchimento da tabela *Values* pode ser visto na tabela 15. Nesse exemplo, à amostra 1 estão associados os valores 1.1 e 1.2, 345 e 243, respetivamente.

Tabela 15 - Valores da porta série guardados na tabela *Values*

Index	**Valor**
1.1	345
1.2	243
2.1	321
2.2	382

Este preenchimento de dados provenientes do sensor não é feito diretamente do sensor para as tabelas. Como se pode observar no fluxograma da figura 13, há um calculo da média do par de valores *v1*,*v2*. Este cálculo foi necessário para conseguir obter uma melhor rentabilização de dados e ainda melhorar a performance do sistema total.

Assim, e observando a figura 16, vemos que os dados provenientes do sensor biomédico são recolhidos num formato String. Seguidamente, aplica-se o método *split()* e transformamos a String para um array. Após esse passo, cada par de amostras é guardado num *buffer*, também este com um índex *i*.

Neste ponto, o índex do *buffer* é importante pois ao atingir o valor 1000, é feito uma média aos 1000 pares de valores *v1*,*v2* e esta média é, posteriormente, armazenada na tabela *Values* da MIB. Este processo repete-se para as próximas 1000 e assim sucessivamente. Este calculo foi estudado para 100, 1000, 10000 amostras de forma a perceber-se qual o melhor tempo de rentabilização do processo e, consequentemente, da arquitetura no geral.

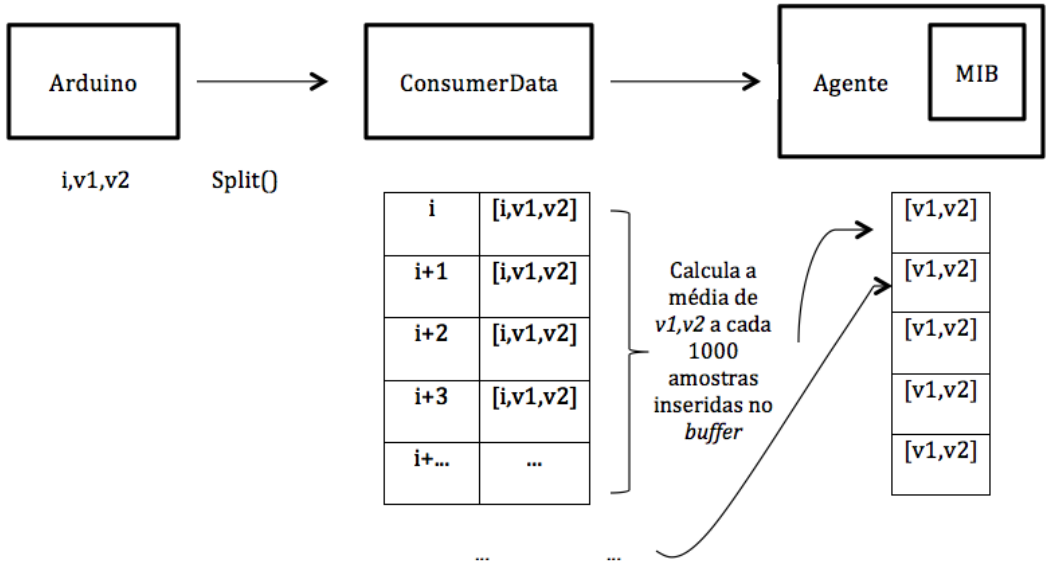


Figura 16 - Calculo de médias de *v1,v2*

4.7 Implementação da aplicação de monitorização biomédica

A aplicação de monitorização biomédica inclui também um módulo de gestor SNMP. É através desse módulo que a aplicação obtém, por *polling* intensivo, os valores das amostras recolhidas no sensor biomédico.

Em termos aplicacionais, além da implementação dum interface com os utilizadores, a funcionalidade mais importante desta componente é a geração/emissão de alertas quando o valor dos dados recolhidos pressupõe um estado de risco para o paciente ou um estado de funcionamento incorrecto do equipamento.

No protótipo desenvolvido, foi escolhido implementar um fluxímetro ótico de baixo custo que servirá para monitorizar a probabilidade de acontecer a hipercoagulação no paciente. Nos testes efetuados, o eventual estado de fluidez do sangue dum paciente foi simulado por manipulação direta dos sensores óticos do circuito eletrónico criado (consultar a última secção deste capítulo para uma discussão mais detalhada da construção deste sensor biomédico).

Quando um estado de fluidez pressupõe uma probabilidade de hipercoagulação de risco para o paciente, e se mantém por mais de um segundo, é enviado um alerta por email para a os responsável médicos. Além disso, o alerta e os dados biomédicos que o geraram, são

armazenados num ficheiro de *log* (ver figura 17). De salientar que, nesta aplicação, são detetados sintomas de *flapping* que podem gerar alertas desnecessários. Neste situação, os alertas não são enviados.

log.txt

```
540 Coagulado @2016-05-13 16:02:24.367 - 2016-05-13 16:02:24.907
270 Coagulado @2016-05-13 16:02:24.637 - 2016-05-13 16:02:24.907
797 Coagulado @2016-05-13 16:04:36.114 - 2016-05-13 16:04:36.911
543 Coagulado @2016-05-13 16:04:36.368 - 2016-05-13 16:04:36.911
270 Coagulado @2016-05-13 16:04:36.641 - 2016-05-13 16:04:36.911
811 Coagulado @2016-05-13 16:04:37.179 - 2016-05-13 16:04:37.990
541 Coagulado @2016-05-13 16:04:37.449 - 2016-05-13 16:04:37.990
270 Coagulado @2016-05-13 16:04:37.720 - 2016-05-13 16:04:37.990
810 Coagulado @2016-05-13 16:04:38.261 - 2016-05-13 16:04:39.071
540 Coagulado @2016-05-13 16:04:38.531 - 2016-05-13 16:04:39.071
270 Coagulado @2016-05-13 16:04:38.801 - 2016-05-13 16:04:39.071
810 Coagulado @2016-05-13 16:04:39.342 - 2016-05-13 16:04:40.152
540 Coagulado @2016-05-13 16:04:39.612 - 2016-05-13 16:04:40.152
270 Coagulado @2016-05-13 16:04:39.882 - 2016-05-13 16:04:40.152
815 Coagulado @2016-05-13 16:04:40.423 - 2016-05-13 16:04:41.238
541 Coagulado @2016-05-13 16:04:40.697 - 2016-05-13 16:04:41.238
270 Coagulado @2016-05-13 16:04:40.968 - 2016-05-13 16:04:41.238
811 Coagulado @2016-05-13 16:04:41.509 - 2016-05-13 16:04:42.320
542 Coagulado @2016-05-13 16:04:41.778 - 2016-05-13 16:04:42.320
271 Coagulado @2016-05-13 16:04:42.049 - 2016-05-13 16:04:42.320
811 Coagulado @2016-05-13 16:04:42.589 - 2016-05-13 16:04:43.400
540 Coagulado @2016-05-13 16:04:42.860 - 2016-05-13 16:04:43.400
270 Coagulado @2016-05-13 16:04:43.130 - 2016-05-13 16:04:43.400
811 Coagulado @2016-05-13 16:04:43.671 - 2016-05-13 16:04:44.482
536 Coagulado @2016-05-13 16:04:43.946 - 2016-05-13 16:04:44.482
271 Coagulado @2016-05-13 16:04:44.211 - 2016-05-13 16:04:44.482
815 Coagulado @2016-05-13 16:04:44.752 - 2016-05-13 16:04:45.567
540 Coagulado @2016-05-13 16:04:45.027 - 2016-05-13 16:04:45.567
264 Coagulado @2016-05-13 16:04:45.303 - 2016-05-13 16:04:45.567
```

Figura 17 – Ficheiro *log* dos alertas

4.8 Sensor universal de fluxo sanguíneo

Como sensor a implementar no protótipo do sistema desenvolvido foi escolhido o fluxímetro (medidor de fluxo sanguíneo).

Este tipo de sensor costuma ser implementado recorrendo a tecnologias baseadas em ultrassons. O fluxo é medido pela diferença temporal na propagação do sangue através dum tubo. Este método baseia-se em sinais bidirecionais. Para tal são utilizados transdutores eléctricos (dispositivos que transformam uma grandeza não eléctrica, física ou química, num sinal eléctrico e vice-versa) que são normalmente instalados no tubo com um ângulo compreendido entre os 30 e os 45 graus. Para medir o fluxo são necessários 2 transdutores: um a montante e outro a jusante (ver figura 18). Para determinar a velocidade do fluxo, usamos a seguinte equação:

$$V = \frac{MD}{\sin 2\theta} X \frac{\Delta T}{T_{up} X T_{down}}$$

com

M = Diferença temporal de propagação do sinal ultra-sónico;

D = Diâmetro do tubo;

θ= Ângulo entre o sinal ultra-sónico e a vazão/caudal;

T_{up} = Diferença temporal de propagação na direção do fluxo;

T_{down}= Diferença temporal de propagação na direção oposta;

ΔT= T_{up}–T_{down}.

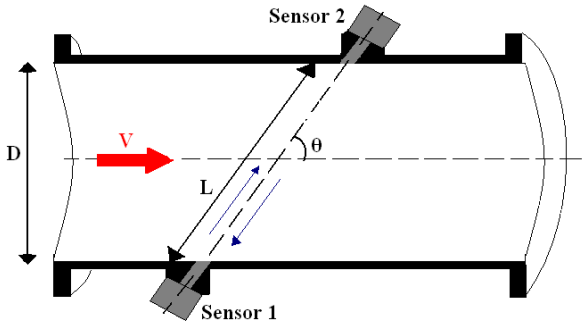


Figura 18 – Posicionamento de transdutores [39]

A tecnologia dos materiais necessários ao desenvolvimento deste tipo de fluxímetro é relativamente dispendioso e o processo de construção muito exigente.

Fluxímetro ótico – nova abordagem

Um técnica alternativa para a construção de fluxímetros baseia-se na medição do fluxo através da utilização de transdutores de infravermelhos. Uma das principais vantagens desta alternativa, para além do baixo custo, comparativamente com a medição baseada em ultrassons, é que a radiação IV é inofensiva para a potência de emissão utilizada. Existem outros métodos de deteção de partículas através de outros espectros, mas que, frequentemente, necessitam do recurso a radiação em comprimentos de onda que são menos seguros. Outro aspeto positivo é que não requer configurações no material, basta visualizar as amostras recolhidas no final e processá-las consoante o propósito pretendido. A desvantagem principal

é que requerem uma minuciosa calibração, pois são bastante sensíveis, sobretudo no que concerne a direccionalidade do feixe de radiação de infravermelhos.

Na figura 19 apresenta-se o esquema do fluxímetro ótico proposto.

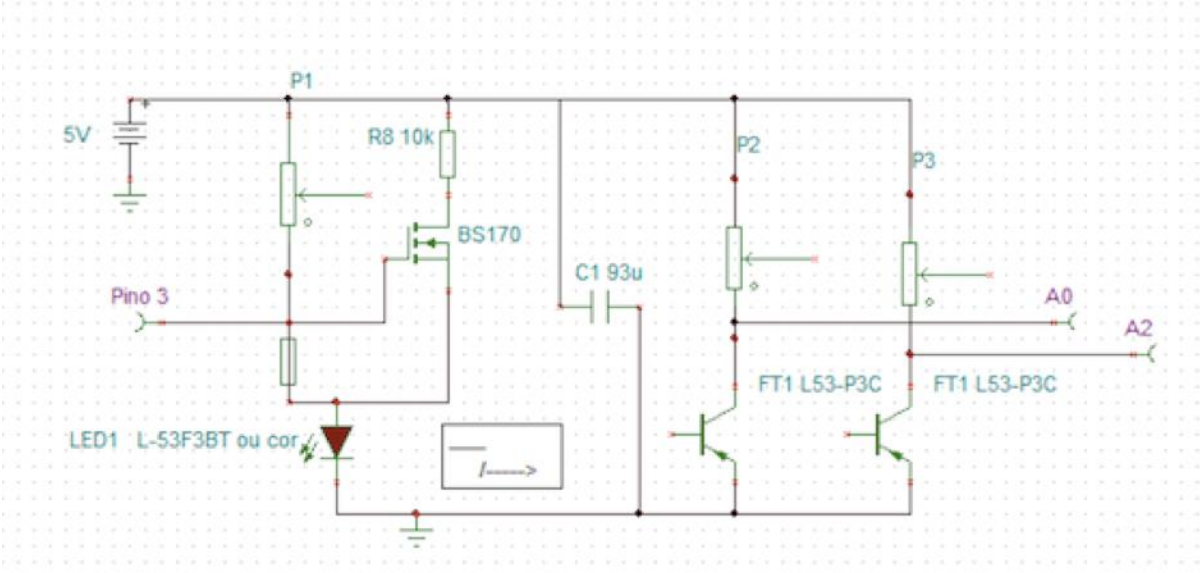


Figura 19 – Esquemático eletrónico

Na figura acima LED1 é um díodo emissor de infravermelhos (fotodíodo emissor). A resistência R2 é ajustada para aumentar ou diminuir a intensidade do feixe de infravermelhos gerado, o que pode ser útil para calibrar o sensor de fluxo. Na gate do MOSFET é aplicado uma onda quadrada com a frequência de aproximadamente 10 KHz, gerada recorrendo a um sinal de PWM proveniente da placa de desenvolvimento Arduino (divisor de relógio modificado através da programação direta dos registos de controlo do periférico PWM) (ver figura 20). O díodo emissor poderia ser colocado em paralelo com o MOSFET se fosse necessário diminuir o tempo que o díodo demora a desligar.

Dois fotodíodos receptores são conetados ao circuito de modo a amplificar o sinal recebido que será depois adquirido pelo conversor analógico/digital da placa do Arduino. O LED vermelho apenas nos indica que o circuito está em funcionamento em função da onda quadrada gerada.

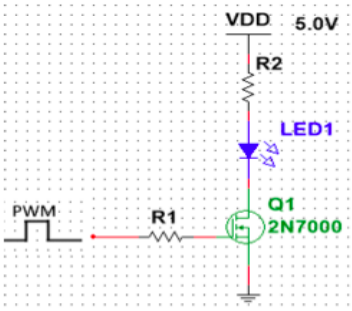


Figura 20 – Entrada PMW

Este circuito é fácil de ser adaptado para outras situações, na medida em que é possível a introdução de mais sinais de entrada, aumentando, para isso, o número de MOSFETs apenas ligados a um LED. Consequentemente os valores das resistências que os antecedem teriam de ser recalculados.

Testaram-se diversos fototransístores e fotodíodos e, de facto, os apresentados na figura são os melhores pelas suas características – sensibilidade, linearidade e comprimento de onda. Por fim, em vez de resistências fixas, utilizaram-se algumas variáveis de forma a maximizar o sinal de saída.

Como entrada no circuito, escolheram-se os portos P3.2 e P3.4 do microcontrolador 8051. Contudo, existem outros microcontroladores mais simples e eficazes para o mesmo fim. Depois de se reformular o esquema passou a utilizar-se o Arduino Uno. Para tal, usam-se os pinos A0 e A2 entre as saídas disponíveis, escolhendo como entrada o pino 3. Esta escolha não foi arbitrária. Este pino pode ser usado no modo PMW de 8 bits através da função *AnalogWrite()*.

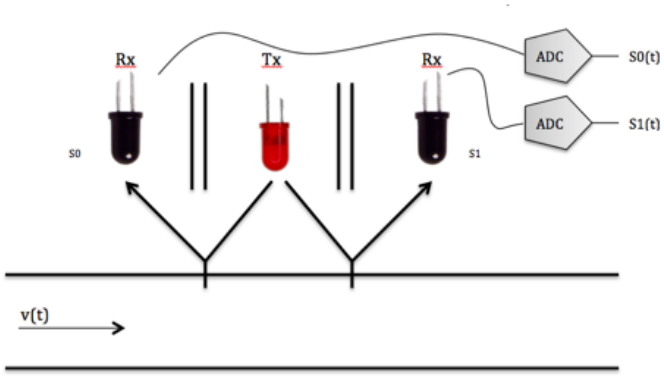


Figura 21 – Princípio da reflexão

O Tx é o LED vermelho emissor que irradia luz para o tubo que possui um líquido com velocidade *v(t)*, e esta é refletida para os fotodíodos – Rx. Este sinal captado constitui as saídas S0 e S1 (ver figura 21).

O código de controlo foi criado no programa *Arduino* e é bastate simples, como se pode observar na figura 22.

Figura 22 – Código arduino

Algum material, nomeadamente *datasheets*, utilizado para a construção do protótipo *hardware* pode ser consultado em [40, 41].

5. DISCUSSÃO DOS RESULTADOS OBTIDOS

Depois de todas as componentes estarem operacionais (tanto de *hardware* como de *software*) o protótipo foi testado e corrigido variadas vezes até se atingir uma versão estável e confiável. A verificação dos resultados de todo o processo, como foi mencionado no capítulo anterior, foi feita através dum interface de utilizador implementando um gráfico dinâmico com duas cores: uma para um estado benigno (em que o sistema não gera qualquer alerta) e outra para um estado problemático, em que a probabilidade de coagulação do sangue do paciente é muito elevada, requerendo a intervenção rápida do pessoal médico.

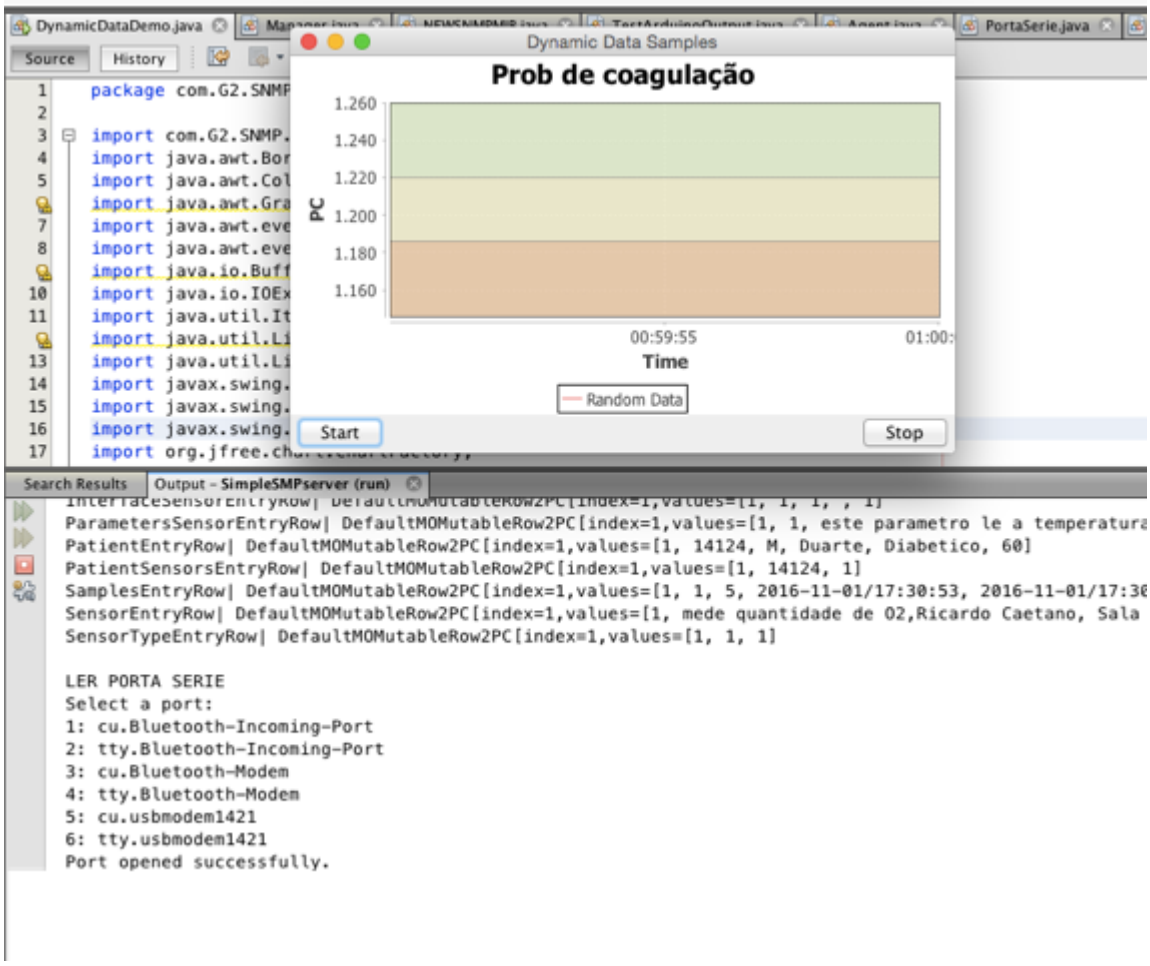


Figura 23 – Interface gráfico

Na imagem da figura 23 é possível observar o início da execução da aplicação gestora de monitorização da densidade sanguínea (na parte superior) e que gera o gráfico com uma gama de valores recebidos do agente SNMP que está implementado num nó de agregação de dados de sensores óticos de fluxo sanguíneo (na configuração em causa foi utilizado apenas

um fluxímetro ótico ao nó agregador). Na parte inferior da imagem é possível observar a configuração executada para o *software* do nó agregador, incluindo a inicialização da porta de comunicação série ligada à instrumentação do fluxímetro e módulo do agente SNMP.

Depois do utilizador iniciar o processo de monitorização o interface gráfico vai apresentando um valor das amostras em função do tempo, conforme se pode visualizar na figura 24. Note-se que o valor apresentado como Probabilidade de Coagulação (PC) não é um valor em percentagem. É antes um valor calculado pelo *software* desenvolvido para a instrumentação do sensor para este protótipo e que pode varia entre 1.16 e 1.26, sendo que valores menos indicam uma maior probabilidade maior de coagulação do sangue. Através da realização de diversos testes definiu-se, para a aplicação de monitorização em causa, o valor 1.19 como o valor limiar para se considerar a probabilidade de hipercoagulação um risco para o paciente. Numa futura versão melhorada do protótipo, os valores de PC deverão ser calculados de forma a ficarem num intervalo tradicional entre 0 e 1 e em que os valores mais altos representem probabilidades maiores de hipercoagulação.

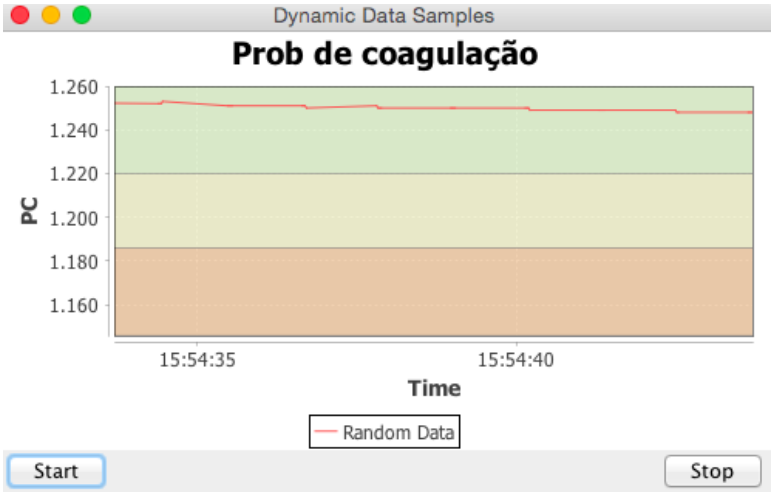


Figura 24 – Dados amostrados no visível (à luz)

Na imagem da figura 24 pode ver-se uma linha de valores de PC dum paciente típico que não apresenta qualquer tipo de risco de hipercoagulação sanguínea.

Na imagem da figura 25 podemos verificar um gráfico de monitorização dum paciente em que o risco médio de hipercoagulação começa a ser relevante.

.

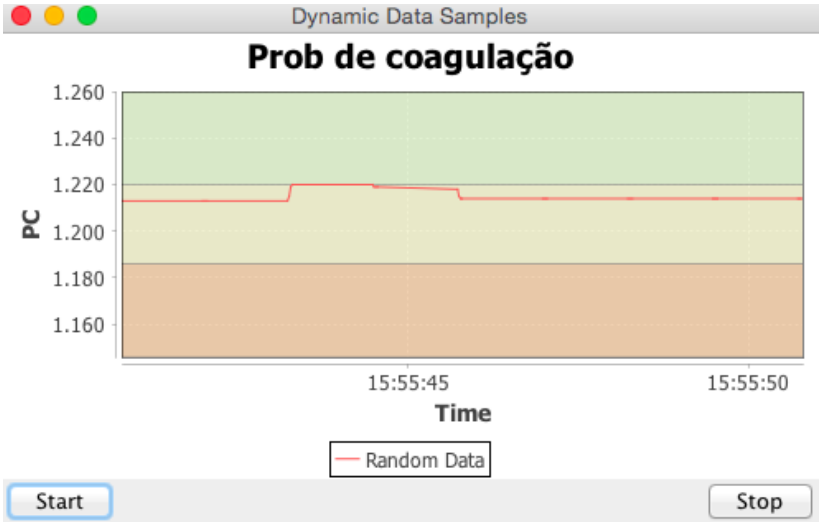


Figura 25 – Dados amostrados no visível (luz mediana)

Por fim, na imagem da figura 26 os valores corresponderiam já a uma situação de risco relevante para o paciente. Quando este estado se mantém por mais de um segundo é gerado e enviado um alerta para a equipa médica responsável através duma mensagem de email. O alerta, e os dados biomédicos, que levaram à sua emissão, são também armazenados num ficheiro de *log*.

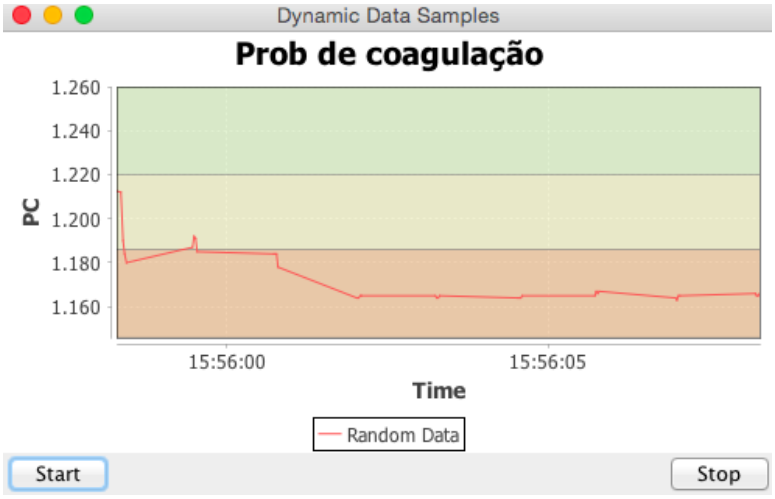


Figura 26 – Dados amostrados no visível (luz reduzida)

Na imagem da figura 27 pode verificar-se uma situação de variação rápida de valores de PC (fenómeno de *flapping*) que é detetada pela aplicação de diganóstico/monitorização. Nestes casos, a aplicação não gera os alertas desnecessários. No entanto, se a situação de

flapping se mantiver durante um período de tempo alargado, a aplicação gera um aviso específico, indicando que a situação precisará de atenção, ainda que não urgente. Eventualmente, pode indicar um funcionamento incorreto do *hardware* ou do *software*.

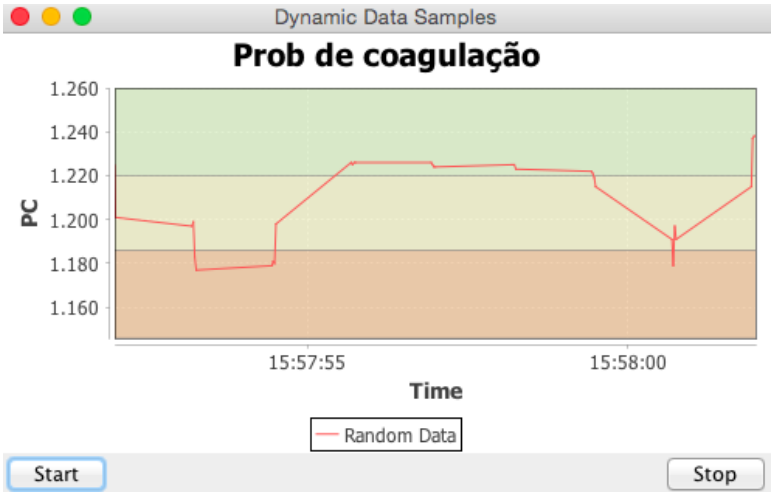


Figura 27 – Valores mínimos não relevantes

Apesar da relativa simplicidade do sistema protótipo desenvolvido, ele abarca todos os níveis funcionais da arquitetura, desde a implementação da instrumentação física de fluxímetros óticos até à implementação duma aplicação de diagnóstico/monitorização biomédica com interface gráfico, passando pela implementação do software dum nó de agregação de sensores, incluindo a implementação duma instanciação completa dum agente SNMP e da respetiva MIB específica, desenvolvida neste projeto para a gestão de redes de sensores de dados biomédicos. O funcionamento adequado e funcionalmente correto de todo o sistema implementado pode, assim, ser considerado uma prova de conceito válida da arquitetura proposta, incluindo a especificação da MIB.

5.1 Limitações do protótipo

Existem ainda questões relevantes que importa referir e que são limitações do protótipo desenvolvido. Uma delas é o facto de se verificar um atraso facilmente percetível entre o momento da amostragem dos valores pela instrumentação do sensor e a correspondente alteração do estado do gráfico do interface da aplicação do utilizador. Ora, dependendo do tipo de sensor em causa e dos requisitos de tempo real de algumas aplicações biomédicas, este atraso pode ser significativo, sendo que é sempre indesejável.

Podemos concluir com segurança que este atraso é sobretudo devido às limitações de *hardware* de muito baixo custo usado na implementação da instrumentação do sensor (por exemplo, o *baud rate* utilizado na comunicação entre a placa de aquisição de dados e o software de processamento da instrumentação do sensor é limitado a velocidades muito baixas). Além disso, o próprio circuito eletrónico desenvolvido para o fluxímetro não foi optimizado e utiliza componentes muito baratos e de baixa qualidade. Finalmente, o software do nó agregador utiliza APIs do pacote SNM4J para a implementação da instrumentação da MIB e dos protocolos comunicacionais do agente SNMP que é pouco eficiente.

Estas limitações de performance, que seriam um problema em situações de aplicação de sistemas biomédicos com requisitos de tempo real, não são devidos às estratégias inerentes à arquitetura proposta, mas antes resultam de limitações próprias dos recursos e do processo de implementação do protótipo.

Uma outra questão prende-se com o facto do protótipo também não endereçar quiaquer requisitos de segurança. Em ambiente de utilização isto era inaceitável, mas em ambiente laboratorial é perfeitamente admissível, sobretudo tendo em consideração que os requisitos normais de segurança para este tipo de sistema poderiam ser alcançáveis através da inclusão dos mecanismos de segurança básicos já normalizados e previstos para a arquitetura SNMP (sutenticação, confidencialidade e verificação de conteúdo), sendo que requisitos adicionais, mais complexos, devem ser implementados pelas próprias aplicações biomédicas.

6. CONCLUSÃO

O trabalho de engenharia aplicada apresentado nesta dissertação pretendeu aplicar um paradigma amplamente conhecido no contexto da gestão de redes, serviços e aplicações distribuídas ao contexto dos sistemas de diagnóstico/monitorização biomédico. A adaptação do paradigma do modelo SNMP a redes de sensores biomédicos no contexto hospitar reveste-se de algum caratér inovador.

Presentemente, o mercado oferece sensores biomédicos cujo software de utilização é protegido pelos fabricantes. Os aparelhos, nomeadamente os mais complexos, têm um custo elevado. Além disso funcionam com softwares não integráveis e sem capacidade de interagir entre si, nomeadamente na desejável troca de informação entre as bases de dados biomédicos.

Como contribuição principal deste trabalho de dissertação pretendia-se definir e validar uma arquitetura que fosse modular e universal, com utilização de protocolos e tecnologias de comunicação e de representação dos dados biomédicos duma forma normalizada. Espera-se que este novo modelo possa, assim, potenciar o desenvolvimento de sistemas de diagnóstico/monitorização biomédico por componentes modulares específicos, em que a interoperacionalidade é garantida pelas tecnologias normalizadas e baseadas em SNMP. A concorrência na produção destes componentes mais específicos, e o facto de se utilizarem redes vulgares para os interligar, potencia enormemente a baixa dos custos de investigação e desenvolvimento e, mais importante, dos custos de aquisição destes produtos. Além disso, em vez de termos um modelo de negócio fortemente influenciado por produtos fechados, passaríamos, aos poucos, a ter um modelo de negócio liderado por produtos abertos, facilmente integráveis e interconetáveis, facilitando a partilha de dados biomédicos, não só entre instituições hospitalares mas também entre instituições dedicadas à investigação biomédica ou farmacêutica.

Um dos aspetos mais importantes na definição da arquitetura proposta, e que é, por isso, outra das principais contribuições desta dissertação, foi a especificação duma MIB própria e específica para a gestão de redes de sensores biomédicos. A instrumentação destas MIBs deve ser implementada nos agentes SNMP que funcionam como nós de agregação dos dados recolhidos nos sensores.

Na implementação dum protótipo para validação do conceito apresentado foi decidido desenvolver um sistema modular completo de raíz, incluindo o *hardware* e *software* dum sensor biomédico de medição do fluxo sanguíneo (que pudesse medir a probabilidade de hipercoagulação do sangue) de baixo custo, o software dum nó de agregação de dados biomédicos com um módulo de agente SNMP integrado e que implementa a instrumentação da MIB desenvolvida. Foi também desenvolvida uma simples aplicação gestora para diagnóstico/monitorização biomédica da probabilidade de hipercoagulação do sangue dum paciente.

O sistema do protótipo foi completado, testado e revisto até se atingir uma versão estável e funcionalmente correta. A correção funcional do sistema, embora não otimizada, foi atingida com hardware de muito baixo custo e todas as tecnologias utilizadas são abertas e normalizadas. Todos os seus componentes interagem com protocolos comunicacionais normalizados e amplamente utilizados nas redes de comunicação mais comuns.

Ainda que o sistema protótipo não tenha tido em consideração os requisitos de segurança deste tipo de aplicações, a implementação futura dos necessários mecanismos de segurança não fica comprometido, uma vez que a sua utilização está já prevista nas tecnologias normalizadas que a arquitetura integrada.

Os testes efetuados tornaram evidentes algumas limitações de performance do sistema, sobretudo em relação ao atraso verificado entre a leitura das amostras nos sensores e a mudança respetiva do estado do interface da aplicação de diagnóstico/monitorização desenvolvida. A convicção, nesta altura, é a de que as dificuldades em garantir eventuais requisitos aplicacionais de tempo real se devem a limitações do *hardware* e do *software* desenvolvido e não ao modelo da solução proposta.

6.1 Trabalho futuro

O trabalho desenvolvido pode, num futuro próximo, ser plasmado num artigo científico que será submetido para revisão em revista ou conferência internacional na área da biomedicina ou na área de gestão de redes/alpicaçãoes distribuídas.

Espera-se também que, na próxima versão do protótipo, já se possam incluir mecanismos que garantam os requisitos de segurança essenciais a este tipo de aplicação.

Outro objetivo de médio prazo é a extensão do protótipo com a implementação duma gateway para um sistema de armazenamanento seguro dos dados biomédicos em base de dados na *cloud*.

Por fim, a otimização do módulo do agente SNMP pode ser conseguida no futuro passando a utilizar uma API em C++ em vez do SNMP4J.

REFERÊNCIAS

[1] OECD (2015), *Fiscal Sustainability of Health Systems: Bridging Health and Finance Perspectives*, OECD Publishing, Paris. DOI: http://dx.doi.org/10.1787/9789264233386-en

[2] *Monitores Cardiácos - Monitores Médicos - Eletromedicina*. *Quirumed.com*. Acedido em 7 janeiro 2016, de http://www.quirumed.com/pt/eletromedicina/monitores-medicos/monitores-cardiacos

[3] *Monitor Multiparâmetros | VITA 400*. *Alfamed.com*. Acedido em 7 janeiro 2016, de http://www.alfamed.com/monitores_multiparametro_vita_400_series.html

[4] Jabbar, H.; Lee, S.; Choi, S.; Baek, S.; Yu, S.; Jeong, T. A Novel Sensing Method and Sensing Algorithm Development for a Ubiquitous Network. Sensors 2010, 10, 8129-8142.

[5] Cuevas-Martinez, J.C.; Gadeo-Martos, M.A.; Fernandez-Prieto, J.A.; Canada-Bago, J.; Yuste-Delgado & A.J. Wireless Intelligent Sensors Management Application Protocol-WISMAP. Sensors, 10, 8827-8849, 2010.

[6] A. Sethi, D. Zhu, and P. Kalyanasundaram, "Shaman - an environment for distributed management applications," in *Integrated Network Man- agement Proceedings*, Seattle, USA, pp. 321–324, 2001.

[7] Steedman, D. *Abstract Syntax Notation.One (ASN.1)*. [online] Bgbm.org. Acedido em 20 fevereiro 2016, de http://www.bgbm.org/TDWG/acc/Documents/asn1gloss.htm

[8] ETSI TR 102 764: " eHEALTH; Architecture; Analysis of user service models, technologies and applications supporting eHealth". Technical Report. Fevereiro 2009.

[9] P. Marluce, A. Cláudio & C. Maria Clicia Stelling, "*Tutorial sobre Redes de Sensores*", 14, 2003.

[10] C. Intanagonwiwat, R. Govindan & D. Estrin, "Directed Diffusion: a scalable and robust communication paradigm for sensor networks", In Proceedings of the Fourth International Conference on Mobile Computing and Networking, ACM Press, 2000.

[11] Simple Network Management Protocol (SNMP. (2016). 1st ed. [ebook] pp.13-21. Acedido em 2 janeiro 2016, de http://www.dcc.fc.up.pt/~rprior/1415/AR/slides/09%20-%20SNMP.pdf

[12] *RFC 1155 - Structure and identification of management information for TCP/IP-based internets*. (1990). *Tools.ietf.org*. Acedido em 3 fevereiro 2016, de https://tools.ietf.org/html/rfc1155

[13] Mascarenhas S. Cérebro Vigiado. Sensor subcutâneo monitora pressão intracraniana em casos de acidentes e doenças. Engenharia Biomédica, 2009.

[14] W.R. Heinzelman , J. Kulik e H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks", In Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, ACM Press, pp 174-185, 1999.

[15] Gta.ufrj.br. (2016). *As Desvantagens do SNMP*. Acedido em 27 março 2016, de http://www.gta.ufrj.br/grad/00_1/joao/desv.htm

[16] Gta.ufrj.br. (2016). *As Vantagens do SNMP*. Acedido em 27 março 2016, de http://www.gta.ufrj.br/grad/04_1/snmp/vantagens.htm

[17] Pt.wikipedia.org. (2016). *Simple Network Management Protocol*. [online] Acedido em 13 janeiro 2016, de https://pt.wikipedia.org/wiki/Simple_Network_Management_Protocol.

[18] Technet.microsoft.com. *What Is SNMP?: Simple Network Management Protocol (SNMP); Services for Macintosh*. [online]. Acedido em 26 fevereiro 2016, de https://technet.microsoft.com/pt-pt/library/cc776379(v=ws.10).aspx

[19] *RFC 3414 - User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*. (2002). *Tools.ietf.org*. Acedido em 3 fevereiro 2016, de https://tools.ietf.org/html/rfc3414

[20] *RFC 3415 - View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*. (2002). *Tools.ietf.org*. Acedido em 3 fevereiro 2016, de https://tools.ietf.org/html/rfc3415

[21] Ju Min, G. & Geun Gwon, T. *An Implementation of Web-Based, Scalable SNMP Network Management System*, 2002.
DOI: 10.3745/KIPSTC.2002.9C.3.431

[22] ManageEngine, c. *Network Monitoring Software by ManageEngine OpManager*. [online] ManageEngine OpManager. Acedido em 28 maio 2016, de https://www.manageengine.com/network-monitoring/what-is-snmp.html

[23] Kb.paessler.com. *How do SNMP, MIBs and OIDs work? | Paessler Knowledge Base*. Acedido em 24 maio 2016, de https://kb.paessler.com/en/topic/653-how-do-snmp-mibs-and-oids-work

[24] *Gerência de Redes - Suzana Strauch*. *Penta2.ufrgs.br*. Acedido em 27 março 2016, de http://penta2.ufrgs.br/gere96/cmipXsnmp/cmip_stra.htm

[25] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol," vol. RFC 1157, 1990, pp. 1–36.

[26] R. Enns, "NETCONF Configuration Protocol", IETF RFC 4741, December 2006

[27] W. Chen, N. Jain, and S. Singh, "Anmp: Ad hoc network management protocol," in *Selected Areas in Communications, IEEE Journal on*, p. 26, 1999.

[28] Hedstrom, Watwe, Sakthidharan, B. *1 Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions* (1st ed., pp. 1-13). Interdisciplinary Telecommunications at the U niversity of Colorado, Boulder, 2011.

[29] Gta.ufrj.br. (2016). *Untitled Document*. [online] Available at: http://www.gta.ufrj.br/grad/04_1/snmp/mib.htm [Acedido a 13 Abril 2016].

[30] Anon, (2016). [online] Available at:

http://www.teleco.com.br/tutoriais/tutorialsnmpred1/pagina_4.asp [Acedido a 13 Abril 2016].

[31] *Simple Network Management Protocol* (1st ed., pp. 4-2). Acedido em 13 fevereiro 2016, de http://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucm/managed_services/8_6_1/cucm/managed_services/snmp.pdf

[32] Morais S, Mimoso G. Oximetria de pulso no diagnóstico de cardiopatia congénita. Sugestões para a implementação de uma estratégia de rastreio. Maternidade Bissaya Barreto, Centro Hospitalar e Universitário de Coimbra. 2014

[33] American Diabetes Association. Clinical Practice Recommendation 2006. Diabetes Care: supplement 1, 2006.

[34] Users.isr.ist.utl.pt. (n.d.). *2 - Electrocardiograma*. [online] Acedido em 2 dezembro 2015, de http://users.isr.ist.utl.pt/~jmrs/teaching/orientations/2005_6/ecg/web/Electrocardiograma.htm

[35] McCloghrie, Systems, Perkins, Schoenwaelder, Braunschweig, K. *Structure of Management Information Version 2 (SMIv2)*. 2nd ed. [ebook] Acedido em 30 maio 2016, de https://tools.ietf.org/pdf/rfc2578.pdf [Accessed 7 Jan. 2016].

[36] Fock, F. (2015). *SNMP4J-Agent. Agentpp.com*. Acedido em 22 março 2016, de http://www.agentpp.com/api/java/snmp4jagent.html

[37] *Java SNMP API/SNMP Stack/SNMP Library for SNMP Management - WebNMS SNMP API*. (2004). *Webnms.com*. Acedido em 18 maio 2016, de https://www.webnms.com/snmp/

[38] *JAVA SNMP API / SNMP Stack / SNMP Library / SNMP.*. (2002). *Ireasoning.com*. Acedido em 18 maio 2016, de http://www.ireasoning.com/snmpapi.shtml

[39] Lschafer. *Esquema de funcionamento de um medidor de fluxo ultra-sônico por Tempo de trânsito*. Acedido em 27 maio 2016, de https://pt.wikipedia.org/wiki/Medidor_de_vaz%C3%A3o#/media/File:Tempo_de_transito.PNG

[40] Solutions, M. *BS170 Transistor data sheet*. *Slideshare.net*. Acedido em 2 janeiro 2015, de http://www.slideshare.net/Microtechsolutions/bs170-transistor-data-sheet

[41] *OP906 datasheet, OP906 PDF, PIN Silicon Photodiode - OPTEK Technologies*. *Datasheetframe.com*. Acedido em 2 janeiro 2015, de http://www.datasheetframe.com/PDF/OP906-PDF/559636

ANEXO I – CÓDIGO DA MIB

```
NEW-SNMP-MIB DEFINITIONS ::= BEGIN

IMPORTS
        MODULE-IDENTITY,
        OBJECT-TYPE,
        Unsigned32
                FROM SNMPv2-SMI
        TEXTUAL-CONVENTION
                FROM SNMPv2-TC
        OBJECT-GROUP
                FROM SNMPv2-CONF;

testmib MODULE-IDENTITY
        LAST-UPDATED "201601111614Z"    -- Jan 11, 2016 4:14:00 PM
        ORGANIZATION "uminho"
        CONTACT-INFO
                "12345678"
        DESCRIPTION
                "this person is the MIB creator"
        REVISION "201601111614Z"        -- Jan 11, 2016 4:14:00 PM
        DESCRIPTION
                "Initial version."
        -- 1
        -- 1.3.6.1.4.9999999
::= { 1  3  6  1  4  9999999 }


Float32TC ::= TEXTUAL-CONVENTION
        STATUS  current
        DESCRIPTION
                "This type represents a 64-bit (8-octet) IEEE
                floating-point number in binary interchange format."
        REFERENCE
                "IEEE Standard for Floating-Point Arithmetic,
                Standard 754-2008"
        SYNTAX OCTET STRING
```

```
InterfaceType ::= TEXTUAL-CONVENTION
        STATUS  current
        DESCRIPTION
                "Here we present witch interfaces we have on a biomedical
sensor market"
        SYNTAX INTEGER {
                        interfacetypeBluetooth(1),
                        interfacetypeWifi(2),
                        interfacetypeRS232(3),
                        interfacetypeUSB(4),
                        interfacetypeLAN(5) }


ParametersType ::= TEXTUAL-CONVENTION
        STATUS  current
        DESCRIPTION
                "Here we present some parameters that concerns to the
sensor itself and also what can a sensor measure"
        SYNTAX INTEGER {
                        parameterTypeCardiacFrequency(1),
                        parameterTypePwave(2),
                        parameterTypePRRange(3),
                        parameterTypeQRSComplex(4),
                        parameterTypeEletricAxis(5),
                        parameterTypeBloodOxygenSaturation(6),
                        parameterTypeIntracranialPressure(7),
                        parameterTypeTemperature(8),
                        parameterTypeArterialPressure(9),
                        parameterTypeRespirationRate(10),
                        parameterTypeCapnography(11),
                        parameterTypeCardiacOutput(12),
                        parameterTypeInjectionTemperature(13),
                        parameterTypeBloodTemperature(14),
                        parameterTypeEEG(15),
                        parameterTypeInvasivePressure(16),
                        parameterTypeNonInvasivePressure(17),
                        parameterTypeNonBloodInvasivePressure(18),
                        parameterTypeGlucose(19),
                        parameterTypeBloodFlow(20),
                        parameterTypeTemperatureC(10001),
                        parameterTypeTemperatureF(10002),
```

```
                        parameterTypeVoltage(10003),
                        parameterTypeCurrent(10004),
                        parameterTypeMonitorizationFrequency(10005),
                        parameterTypeWeight(10006),
                        parameterTypeInitialization(10007),
                        parameterTypeBatery(10008),
                        parameterTypeDisplay(10009),
                        parameterTypeHumidity(10010),
                        parameterTypeAtmosfericPressure(10011) }


SensorType ::= TEXTUAL-CONVENTION
        STATUS  current
        DESCRIPTION
                "Here we present some sort of worldwide biomedical sensors"
        SYNTAX INTEGER {
                        sensortypeEGC(1),
                        sensortypePulseOximeter(2),
                        sensortypeIntracranialPressure(3),
                        sensortypeMultiparameter(4),
                        sensortypeGlucose(5),
                        sensortypeBloodFlow(6),
                        sensortypeUniversal(7) }


UnitsType ::= TEXTUAL-CONVENTION
        STATUS  current
        DESCRIPTION
                "Here we present witch units match with some parameters
(parameters related to samples ( from 1001 until 1011)"
        SYNTAX INTEGER {
                        unittypeC(1),
                        unittypeF(2),
                        unittypeVolts(3),
                        unittypeAmper(4),
                        unittypeHz(5),
                        unittypeWeight(6),
                        unittypeKg(7),
                        unittypeSeconds(8),
                        unittypeHours(9),
                        unittypeMilimiters(10),
```

```
                        unittypePercentage(11),
                        unittypePascal(12),
                        unittypeBeatsPerMinute(13),
                        unittypeDegrees(14),
                        unittypeMercuryMilimiters(15),
                        unittypeLPM(16),
                        unittypeMilligramsPerDeciliter(17) }


interfaceSensorTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF InterfaceSensorEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "This is the table that concerns about interfaces"
        -- 1.3.6.1.4.9999999.1
::= { testmib 1 }


interfaceSensorEntry OBJECT-TYPE
        SYNTAX  InterfaceSensorEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "Here we show how many entries/rows we have on this table”
        INDEX {
                interfaceSensorInterfacesSensorsIndex }
        -- 1.3.6.1.4.9999999.1.1
::= { interfaceSensorTable 1 }


InterfaceSensorEntry ::= SEQUENCE {

        interfaceSensorInterfacesSensorsIndex Unsigned32,
        interfaceSensorSensorIndexValue       INTEGER,
        interfaceSensorInterfaceTypeID        INTEGER,
        interfaceSensorError                  OCTET STRING,
        interfaceSensorStatus                 INTEGER }
```

```
interfaceSensorInterfacesSensorsIndex OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Table index"
        -- 1.3.6.1.4.9999999.1.1.1
::= { interfaceSensorEntry 1 }


interfaceSensorSensorIndexValue OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Give us the values”
        -- 1.3.6.1.4.9999999.1.1.2
::= { interfaceSensorEntry 2 }


interfaceSensorInterfaceTypeID OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Give us the type"
        -- 1.3.6.1.4.9999999.1.1.3
::= { interfaceSensorEntry 3 }


interfaceSensorError OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Give us the sensor error"
        -- 1.3.6.1.4.9999999.1.1.4
::= { interfaceSensorEntry 4 }
```

```
interfaceSensorStatus OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Give us the status: on or off"
        -- 1.3.6.1.4.9999999.1.1.5
::= { interfaceSensorEntry 5 }


parametersSensorTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF ParametersSensorEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "This table is related to all parameters sensors"
        -- 1.3.6.1.4.9999999.2
::= { testmib 2 }


parametersSensorEntry OBJECT-TYPE
        SYNTAX  ParametersSensorEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "Here we show how many entries/rows we have on this table"
        INDEX {
                parametersSensorParameterIndex }
        -- 1.3.6.1.4.9999999.2.1
::= { parametersSensorTable 1 }


ParametersSensorEntry ::= SEQUENCE {

        parametersSensorParameterIndex      INTEGER,
        parametersSensorSensorIndexValue    INTEGER,
        parametersSensorDescription         OCTET STRING,
        parametersSensorNValuesInParameters INTEGER,
        parametersSensorFirstValueIndex     INTEGER }
```

```
parametersSensorParameterIndex OBJECT-TYPE
        SYNTAX  INTEGER (1)
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Give us the index"
        -- 1.3.6.1.4.9999999.2.1.1
::= { parametersSensorEntry 1 }


parametersSensorSensorIndexValue OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "give us the values"
        -- 1.3.6.1.4.9999999.2.1.2
::= { parametersSensorEntry 2 }


parametersSensorDescription OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Sensor description"
        -- 1.3.6.1.4.9999999.2.1.3
::= { parametersSensorEntry 3 }


parametersSensorNValuesInParameters OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "parametros"
        -- 1.3.6.1.4.9999999.2.1.4
::= { parametersSensorEntry 4 }
```

```
parametersSensorFirstValueIndex OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "index of the first value"
        -- 1.3.6.1.4.9999999.2.1.5
::= { parametersSensorEntry 5 }


sensorTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF SensorEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "table related to sensors"
        -- 1.3.6.1.4.9999999.3
::= { testmib 3 }


sensorEntry OBJECT-TYPE
        SYNTAX  SensorEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "This is a table that provide us what function is related
to witch sensor, who is the manufacturer, where is the location related to
the sensor and also who is in charge for that sensor"
        INDEX {
                sensorSensorIndex }
        -- 1.3.6.1.4.9999999.3.1
::= { sensorTable 1 }


SensorEntry ::= SEQUENCE {

        sensorSensorIndex    INTEGER,
        sensorDescription    OCTET STRING,
        sensorLocalization   OCTET STRING,
        sensorPersonInCharge OCTET STRING,
        sensorManufacturer   OCTET STRING }
```

```
sensorSensorIndex OBJECT-TYPE
        SYNTAX  INTEGER (1)
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "index"
        -- 1.3.6.1.4.9999999.3.1.1
::= { sensorEntry 1 }


sensorDescription OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "description"
        -- 1.3.6.1.4.9999999.3.1.2
::= { sensorEntry 2 }


sensorLocalization OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Give us the room and floor where our sensor is"
        -- 1.3.6.1.4.9999999.3.1.3
::= { sensorEntry 3 }


sensorPersonInCharge OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Give us who is in charge for that sensor"
        -- 1.3.6.1.4.9999999.3.1.4
::= { sensorEntry 4 }
```

```
sensorManufacturer OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "give us the Manufacturer"
        -- 1.3.6.1.4.9999999.3.1.5
::= { sensorEntry 5 }


sensorTypeTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF SensorTypeEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION ""
        -- 1.3.6.1.4.9999999.4
::= { testmib 4 }


sensorTypeEntry OBJECT-TYPE
        SYNTAX  SensorTypeEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "We need a table that tell us witch name is related to
witch sensor. On the SensorTable we can not figure it out that. Therefore
we need this table to match the SensorTable Index with our Textual
Convention SensorType. Combined, we know what sensor we are talking about"
        INDEX {
                sensorTypeSensorTypeIndex }
        -- 1.3.6.1.4.9999999.4.1
::= { sensorTypeTable 1 }


SensorTypeEntry ::= SEQUENCE {

        sensorTypeSensorTypeIndex  INTEGER,
        sensorTypeSensorIndexValue INTEGER,
        sensorTypeSensorTypeID     INTEGER }
```

```
sensorTypeSensorTypeIndex OBJECT-TYPE
        SYNTAX  INTEGER (1)
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "index"
        -- 1.3.6.1.4.9999999.4.1.1
::= { sensorTypeEntry 1 }


sensorTypeSensorIndexValue OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "This column represents witch sensor we are talking about.
It is the Index from the Sensor table"
        -- 1.3.6.1.4.9999999.4.1.2
::= { sensorTypeEntry 2 }


sensorTypeSensorTypeID OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It   tell   us   witch   type   is   our   sensor:   ECG,
multiparameters..etc"
        -- 1.3.6.1.4.9999999.4.1.3
::= { sensorTypeEntry 3 }


samplesTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF SamplesEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION ""
        -- 1.3.6.1.4.9999999.5
::= { testmib 5 }
```

```
samplesEntry OBJECT-TYPE
        SYNTAX  SamplesEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "table that give us samples"
        INDEX {
                samplesSampleIndex }
        -- 1.3.6.1.4.9999999.5.1
::= { samplesTable 1 }


SamplesEntry ::= SEQUENCE {

        samplesSampleIndex              Unsigned32,
        samplesPacientSensorsIndexValue INTEGER,
        samplesNValuesInSample          INTEGER,
        samplesInitialTime              INTEGER,
        samplesFinalTime                INTEGER,
        samplesFirstValueIndex          INTEGER }


samplesSampleIndex OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "index"
        -- 1.3.6.1.4.9999999.5.1.1
::= { samplesEntry 1 }


samplesPacientSensorsIndexValue OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "With  this  column  we  can  check  witch  pacient  have  this
sample"
        -- 1.3.6.1.4.9999999.5.1.2
```

```
::= { samplesEntry 2 }


samplesNValuesInSample OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It give us the number of samples that a sensor can
provide. One sensor can measure more than 1 parameter, than we have more
than 1 sample"
        -- 1.3.6.1.4.9999999.5.1.3
::= { samplesEntry 3 }


samplesInitialTime OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It important to know when the sample was received. This
parameter is for the inicial time that was received"
        -- 1.3.6.1.4.9999999.5.1.4
::= { samplesEntry 4 }


samplesFinalTime OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It important to know when the sample was received. This
parameter is for the final time that was received. With both times, we can
discover witch sample are we looking for.'"
        -- 1.3.6.1.4.9999999.5.1.5
::= { samplesEntry 5 }


samplesFirstValueIndex OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
```

```
        STATUS  current
        DESCRIPTION
                "It is a pointer to the first value in the table values"
        -- 1.3.6.1.4.9999999.5.1.6
::= { samplesEntry 6 }


valuesTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF ValuesEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION ""
        -- 1.3.6.1.4.9999999.6
::= { testmib 6 }


valuesEntry OBJECT-TYPE
        SYNTAX  ValuesEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION ""
        INDEX {
                valuesValuesIndex }
        -- 1.3.6.1.4.9999999.6.1
::= { valuesTable 1 }


ValuesEntry ::= SEQUENCE {

        valuesValuesIndex                   Unsigned32,
        valuesNextSampleParameterIndexValue INTEGER,
        valuesSampleOrParametersTypeID      INTEGER,
        valuesUnitsTypeID                   INTEGER,
        valuesPrecision                     INTEGER,
        valuesValue                         Float32TC }


valuesValuesIndex OBJECT-TYPE
        SYNTAX  Unsigned32 (1)
        MAX-ACCESS read-only
        STATUS  current
```

```
        DESCRIPTION
                "index"
        -- 1.3.6.1.4.9999999.6.1.1
::= { valuesEntry 1 }


valuesNextSampleParameterIndexValue OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It is a pointer to the other number on the index column.
For example: the first value is 3. Then it goes to ValuesIndex find number
3. The match on this column is number 4, for example. It goes again to
ValuesIndex see match and search for more..'"
        -- 1.3.6.1.4.9999999.6.1.2
::= { valuesEntry 2 }


valuesSampleOrParametersTypeID OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "We can have a value that corresponds to a sample or
parameter"
        -- 1.3.6.1.4.9999999.6.1.3
::= { valuesEntry 3 }


valuesUnitsTypeID OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It presents the unit of the value measured"
        -- 1.3.6.1.4.9999999.6.1.4
::= { valuesEntry 4 }


valuesPrecision OBJECT-TYPE
```

```
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It show us what kind of measure is related to the value.
if we have a integer number, the precision is 0; if it is a decimal number,
in the tens, then is 1; if it is a decimal number, in the hundreds, then is
2; if it is a decimal number, in the decameters, then is minus 1 and so on"
        -- 1.3.6.1.4.9999999.6.1.5
::= { valuesEntry 5 }


valuesValue OBJECT-TYPE
        SYNTAX  Float32TC
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It presents the value measured"
        -- 1.3.6.1.4.9999999.6.1.6
::= { valuesEntry 6 }


patientSensorsTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF PatientSensorsEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION ""
        -- 1.3.6.1.4.9999999.7
::= { testmib 7 }


patientSensorsEntry OBJECT-TYPE
        SYNTAX  PatientSensorsEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION
                "It  is  important  to  know  what  and  how  many  sensors  a
pacient have"
        INDEX {
                patientSensorsPatientSensorsIndex }
        -- 1.3.6.1.4.9999999.7.1
```

```
::= { patientSensorsTable 1 }


PatientSensorsEntry ::= SEQUENCE {

        patientSensorsPatientSensorsIndex INTEGER,
        patientSensorsSensorIndexValue    INTEGER,
        patientSensorsPacientIndexValue   INTEGER,
        patientNumber                     INTEGER }


patientSensorsPatientSensorsIndex OBJECT-TYPE
        SYNTAX  INTEGER (1)
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Both combined (SensorPatientEntry 1 and 2) give us our
Index"
        -- 1.3.6.1.4.9999999.7.1.1
::= { patientSensorsEntry 1 }


patientSensorsSensorIndexValue OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Tell us witch sensor is being used"
        -- 1.3.6.1.4.9999999.7.1.2
::= { patientSensorsEntry 2 }


patientSensorsPacientIndexValue OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Tell us witch patient match the sensor"
        -- 1.3.6.1.4.9999999.7.1.3
::= { patientSensorsEntry 3 }
```

```
patientNumber OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "The number of network interfaces (regardless of their
current state) present on this system"
        -- 1.3.6.1.4.9999999.7.1.4
::= { patientSensorsEntry 4 }


patientTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF PatientEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION ""
        -- 1.3.6.1.4.9999999.8
::= { testmib 8 }


patientEntry OBJECT-TYPE
        SYNTAX  PatientEntry
        MAX-ACCESS not-accessible
        STATUS  current
        DESCRIPTION ""
        INDEX {
                patientPatientIndex }
        -- 1.3.6.1.4.9999999.8.1
::= { patientTable 1 }


PatientEntry ::= SEQUENCE {

        patientPatientIndex     INTEGER,
        patientPatientLabel     INTEGER,
        patientPatientGender    OCTET STRING,
        patientPatientName      OCTET STRING,
        patientPatientSituation OCTET STRING,
        patientPatientAge       INTEGER }
```

```
patientPatientIndex OBJECT-TYPE
        SYNTAX  INTEGER (1)
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "index"
        -- 1.3.6.1.4.9999999.8.1.1
::= { patientEntry 1 }


patientPatientLabel OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "It is the pacient number"
        -- 1.3.6.1.4.9999999.8.1.2
::= { patientEntry 2 }


patientPatientGender OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Gender"
        -- 1.3.6.1.4.9999999.8.1.3
::= { patientEntry 3 }


patientPatientName OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "the name"
        -- 1.3.6.1.4.9999999.8.1.4
::= { patientEntry 4 }
```

```
patientPatientSituation OBJECT-TYPE
        SYNTAX  OCTET STRING
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Tell us the patient condition"
        -- 1.3.6.1.4.9999999.8.1.5
::= { patientEntry 5 }


patientPatientAge OBJECT-TYPE
        SYNTAX  INTEGER
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "Tell us the patient age"
        -- 1.3.6.1.4.9999999.8.1.6
::= { patientEntry 6 }


interfaceSensorNumber OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "this object returns a number that tell us how many lines
we have in our table. As the name says, this object is related to the table
interfaceSensorTable"
        -- 1.3.6.1.4.9999999.12
::= { testmib 12 }


parametersSensorNumber OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "this object returns a number that tell us how many lines
we have in our table. As the name says, this object is related to the table
parametersSensorTable"
        -- 1.3.6.1.4.9999999.13
```

```
::= { testmib 13 }


sensorNumber OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "this object returns a number that tell us how many lines
we have in our table. As the name says, this object is related to the table
SensorTable"
        -- 1.3.6.1.4.9999999.14
::= { testmib 14 }


sensorTypeNumber OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "this object returns a number that tell us how many lines
we have in our table. As the name says, this object is related to the table
SensorType"
        -- 1.3.6.1.4.9999999.15
::= { testmib 15 }


samplesNumber OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "this object returns a number that tell us how many lines
we have in our table. As the name says, this object is related to the table
SamplesTable"
        -- 1.3.6.1.4.9999999.16
::= { testmib 16 }


valuesNumber OBJECT-TYPE
        SYNTAX  Unsigned32
```

```
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "this object returns a number that tell us how many lines
we have in our table. As the name says, this object is related to the table
ValuesTable"
        -- 1.3.6.1.4.9999999.17
::= { testmib 17 }


patientSensorsNumber OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "this object returns a number that tell us how many lines
we have in our table. As the name says, this object is related to the table
patientSensorsTable"
        -- 1.3.6.1.4.9999999.18
::= { testmib 18 }


patientPatientNumber OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-only
        STATUS  current
        DESCRIPTION
                "this object returns a number that tell us how many lines
we have in our table. As the name says, this object is related to the table
PacientTable. Also the name of this object is twice patient because we
already have a pacientNumber in the table patientSensorsTable that are
completely different from each other.one is for the number it self of the
patient and this one is to count how many lines we have on the table"
        -- 1.3.6.1.4.9999999.19
::= { testmib 19 }


relatedToNumbers OBJECT-GROUP
        OBJECTS {
                valuesValuesIndex,
                valuesValue,
```

```
                valuesUnitsTypeID,
                valuesSampleOrParametersTypeID,
                valuesPrecision,
                valuesNextSampleParameterIndexValue,
                samplesSampleIndex,
                samplesPacientSensorsIndexValue,
                samplesNValuesInSample,
                samplesInitialTime,
                samplesFirstValueIndex,
                samplesFinalTime,
                parametersSensorSensorIndexValue,
                parametersSensorParameterIndex,
                parametersSensorNValuesInParameters,
                parametersSensorFirstValueIndex,
                parametersSensorDescription }
        STATUS  current
        DESCRIPTION
                "This group contain all the tables that are related to the
sensor it self in terms of samples/values."
        -- 1.3.6.1.4.9999999.9
::= { testmib 9 }

pacient OBJECT-GROUP
        OBJECTS {
                patientSensorsPatientSensorsIndex,
                patientSensorsSensorIndexValue,
                patientSensorsPacientIndexValue,
                patientNumber,
                patientPatientIndex,
                patientPatientLabel,
                patientPatientGender,
                patientPatientName,
                patientPatientSituation,
                patientPatientAge }
        STATUS  current
        DESCRIPTION
                "This group is related to Hospital pacients"
        -- 1.3.6.1.4.9999999.10
::= { testmib 10 }

relatedToSensor OBJECT-GROUP
```

```
        OBJECTS {
                interfaceSensorInterfacesSensorsIndex,
                interfaceSensorSensorIndexValue,
                interfaceSensorInterfaceTypeID,
                interfaceSensorError,
                interfaceSensorStatus,
                sensorSensorIndex,
                sensorDescription,
                sensorLocalization,
                sensorPersonInCharge,
                sensorManufacturer,
                sensorTypeSensorTypeIndex,
                sensorTypeSensorIndexValue,
                sensorTypeSensorTypeID }
        STATUS  current
        DESCRIPTION
                "this group is related to objects that only concern about
the sensor and contain sensors characteristics"
        -- 1.3.6.1.4.9999999.11
::= { testmib 11 }

counter OBJECT-GROUP
        OBJECTS {
                interfaceSensorNumber,
                parametersSensorNumber,
                sensorNumber,
                sensorTypeNumber,
                samplesNumber,
                valuesNumber,
                patientSensorsNumber,
                patientPatientNumber }
        STATUS  current
        DESCRIPTION
                "this group contain all the object that only count how many
lines we have in each table"
        -- 1.3.6.1.4.9999999.20
::= { testmib 20 }

END
```