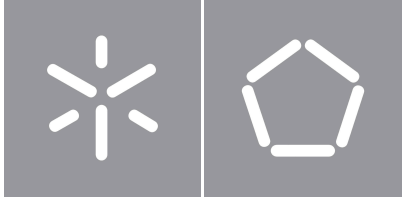
Universidade do Minho
Escola de Engenharia

Gonçalo Pinto Nogueira

Arquitetura Integradora com SNMP para Gestão de Edifícios

abril 2023



Universidade do Minho
Escola de Engenharia

Gonçalo Pinto Nogueira

Arquitetura Integradora com SNMP para Gestão de Edifícios

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação de
Bruno Dias (Departamento de Informática)
Nuno Vasco Lopes (Digital Transformation CoLAB)

abril 2023

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho:

CC BY

https://creativecommons.org/licenses/by/4.0/ *[Esta licença permite que outros distribuam, remixem, adaptem e criem a partir do seu trabalho, mesmo para fins comerciais, desde que lhe atribuam o devido crédito pela criação original. É a licença mais flexível de todas as licenças disponíveis. É recomendada para maximizar a disseminação e uso dos materiais licenciados.]*

Agradecimentos

Gostaria de começar por agradecer a todos aqueles que de alguma forma me apoiaram no desenvolvimento desta dissertação.

Agradeço primeiramente aos meus orientadores, o professor Bruno Dias e ao Professor Nuno Vasco Lopes pelo apoio prestado.

Ao Hélder Jordão, colaborador do DTx Colab que numa fase inicial da dissertação mostrou-se sempre disposto a ajudar e a fornecer todo o tipo de apoio necessário.

Aos meus pais e irmã, por estarem sempre lá para mim e por me sempre fornecerem todas as condições para chegar a este momento.

A toda a minha família, em especial ao meu avô, por todo o apoio dado ao longo dos anos e cujo sonho era ver-me tornar mestre. Apesar de longe sei que irá ficar contente.

A todos os que fizeram parte do meu percurso académico, em especial aos meus colegas de casa Carlos e Eduardo, muito obrigado por tudo.

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, Braga, abril 2023

Gonçalo Pinto Nogueira

Resumo

Ao longo das últimas duas décadas a utilização de sistemas domóticos, num mundo cada vez mais conectado e tecnológico, tem vindo a revelar-se cada vez mais atrativo e com maior aceitação do público em geral. Novos produtos suportados por novos protocolos e tecnologias estão constantemente a ser introduzidos no mercado. No entanto, este desenvolvimento foi quase sempre efetuado sem grande preocupação em definir regras e normas para que fosse possível a interoperacionalidade entre produtos de diferentes fabricantes, originando soluções pouco modulares, de elevado custo e forçando os clientes a escolher um ecossistema de um mesmo fabricante sem ser possível de forma rápida e facilitada integrar tecnologias de vários fabricantes num mesmo sistema.

O principal objetivo desta dissertação foi a definição de uma arquitetura integrada para sistemas domóticos baseada no protocolo de gestão **SNMP** e que permitisse ultrapassar algumas das mais importantes limitações das soluções atuais para este tipo de sistema. Nesse sentido foi criada uma nova **MIB** domótica para implementação num agente **SNMP** integrador. Além disso, foi desenvolvido um novo protocolo de gestão para dispositivos domóticos, mais simples que o **SNMP** e mais adequado para gestão de pequenos equipamentos sensores ou atuadores utilizados em sistemas domóticos. Este protocolo, designado por **Light SNMP** (**L-SNMP**), será utilizado na comunicação entre o agente **SNMP** e os dispositivos domóticos que implementam uma **Light MIB** (**L-MIB**) de domótica. No decorrer do projeto foi criado um sistema protótipo com dispositivos domóticos implementando a **L-MIB** de domótica, um agente **SNMPv2** implementando a **MIB** domótica e uma aplicação gestora **SNMP** que contém um simples interface com o utilizador. As experiências realizadas com este protótipo permitiram confirmar a correção funcional da solução e a sua viabilidade como alternativa tecnológica válida, potencialmente de baixo custo e com elevados níveis de interoperabilidade.

Palavras-chave : Domótica, **SNMP**, **SNMPv2**, Agente SNMP, **MIB**, **Light SNMP**,, **Light MIB**, dispositivos domóticos

Abstract

Over the past two decades the use of domotic systems in an increasingly connected and technological world, has been becoming increasingly attractive and generally with greater public acceptance. New products supported by new protocols and technologies are constantly being introduced into the market. However, this development was almost always carried out without major concerns to define rules and standards to allow inter-operationality between different manufacturers, resulting in unmodular, high-cost solutions and forcing customers to choose an ecosystem from the same manufacturer without being possible to integrate technologies from multiple manufacturers in the same system in a quickly and easily way.

The main objective of this dissertation was the definition of an integrated architecture based on the **SNMP** management protocol and which would allow the overcoming of some of the most important limitations of current solutions for this type of system. In this sense, a new **MIB** for home automation for implementation in an integrative **SNMP** agent. In addition, it has been developed a new management protocol for domotic devices, simpler than **SNMP** and more appropriate for management of small sensor or actuators devices used in domotic systems. This protocol, called **Light SNMP** (**L-SNMP**), will be used in communication between the **SNMP** agent and the domotic devices that implement a **Light MIB** (**L-MIB**). During the project, a prototype system was created with domotic devices implementing the **L-MIB** (**L-MIB**), an **SNMPv2** agent implementing the domotic **MIB** and an **SNMP** management application that contains a simple user interface. The experiments carried out with this prototype allowed to confirm the functional correction of the solution and its viability as a valid technological alternative, potentially low-cost and with high levels of interoperability.

Keywords : Home automation, **SNMP**, **SNMPv2**, SNMP Agent, **MIB**, **Light SNMP**, **Light MIB**, home automation devices

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

AI Artificial Intelligence.

API Application Programming Interface.

ASN.1 Abstract Syntax Notation One.

BER Basic Encoding Rules.

BLT Bluetooth.

CEBus Consumer Electronics Bus.

CoAP Constrained Application Protocol.

DTLS Datagram Transport Layer Security.

EA Estado da Arte.

EHS European Home System.

EIB European Installation Bus.

HBS Home Bus System.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

IA Inteligência Artificial.

IDEA Integrated Development Environment Application.

IID Instance Identifier.

IoT Internet of Things.

IP Internet Protocol.

IPv4 Internet Protocol version 4.

KNX Konnex.

L-MIB Light MIB.

L-SNMP Light SNMP.

MAC Media Access Control.

MD5 Message Digest Method 5.

MIB Management Information Bases.

MQTT Message Queuing Telemetry Transport.

OID Object Identifier.

PDF Portable Document Format.

PDU Protocol Data Unit.

PLC Power Line Communication.

REST Representational State Transfer.

RPD Relatório de Pré-Dissertação.

SHA Secure Hashing Algorithm.

SMI Structure of Management Information.

SMIv2 Structure of Management Information version 2.

SNMP Simple Networking Management Protocol.

SNMPv2 Simple Networking Management Protocol version 2.

SNMPv2c Simple Networking Management Protocol version 2 community.

SNMPv3 Simple Networking Management Protocol version 3.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

X10 X10.

XML Extensible Markup Language.

XSD XML Schema Definition.

ZGB ZigBee.

Parte I

Material Introdutório

Capítulo 1

Introdução

Neste capítulo é introduzido o conceito da domótica e apresentados os principais desafios da implementação e concretização destes sistemas que motivam os objetivos dos trabalhos realizados no contexto desta dissertação.

1.1 Enquadramento e Motivação

O tema da domótica apresenta-se atualmente como uma área importante na aplicação de tecnologias de informação procurando maximizar a eficiência de gestão dos recursos disponíveis nos edifícios através de sistemas computacionais distribuídos.

Além de preocupações económicas e ambientais, a preocupação com o bem estar dentro de um edifício é também relevante sobretudo quando se oferece a possibilidade de controlar sistemas complexos usando interfaces simplificados, com alto nível de abstração funcional. Para tornar estes sistemas uma realidade, é preciso que os vários equipamentos espalhados pelo edifício estejam interligados entre si de forma a possibilitar que a introdução de comandos no interface de utilizador possam ser transmitidos, interpretados e processados.

No mercado atual já existem diversas ofertas de produtos que tentam integrar a gestão de todos os equipamentos num único sistema. No entanto, caso o edifício contenha dispositivos que utilizam diferentes tecnologias de comunicação, não compatíveis entre si, a interligação entre os dispositivos domóticos revela-se bastante complexa e custosa, ou impossível de realizar. Isto deve-se ao facto dos fabricantes definirem e implementarem protocolos proprietários, não estabelecendo um modelo de comunicação que permita a interligação com dispositivos ou sistemas de gestão de outros fabricantes.

Sendo assim e com vista a criar condições para um controlo eficiente e inteligente de todos os dispositivos de hardware e software, serviços ou aplicações que controlam aspetos ambientais e de bem estar em edifícios, deve usar-se uma plataforma de integração com modelos de informação e tecnologias de

comunicação agnósticos, universais e normalizados.

A normalização e modularidade potenciaram o desenvolvimento independente e concorrente dos vários produtos e equipamentos. O mercado tornou-se mais dinâmico e competitivo, beneficiando o utilizador final.

A arquitetura desta plataforma poderá assim oferecer a capacidade de ligação com módulos de inteligência artificial capazes de tomarem decisões que permitam a configuração e manutenção autónoma e mais eficiente do estado operativo dos dispositivos, serviços e aplicações sem a necessidade de intervenção humana.

O trabalho de pesquisa e desenvolvimento desta dissertação devem responder a alguns aspetos conceptuais e tecnológicos deste tipo de arquitetura, sobretudo aqueles relacionados com o modelo de informação e com os protocolos de comunicação usados na interação entre todos os elementos e módulos, incluindo os modelos de dados necessários para conceptualizar, representar, codificar e transmitir toda a informação de gestão.

1.2 Objetivos

Conforme enunciado na secção anterior, a principal motivação para esta dissertação é a definição duma arquitetura que permita interligar todos os sistemas necessários à implementação de soluções dogmáticas utilizando uma única tecnologia de interligação baseado num modelo comunicacional e de informação com utilização já amplamente aceite em contexto de gestão de sistemas distribuídos. Mais ainda, a arquitetura proposta deve também permitir a inclusão de funcionalidades avançadas de automação e auto monitorização baseadas em mecanismos de **Inteligência Artificial** (**IA**).

Para atingir este objetivo geral os trabalhos da dissertação começaram por uma pesquisa das principais tecnologias já existentes na área da domótica, dando especial atenção aos protocolos comunicacionais e modelos de informação associados. Depois da realizada esta pesquisa, o objetivo foi efetuar uma análise comparativa dos prós e contras de cada tecnologia domótica e retirar-se ilações sobre os problemas mais relevantes na sua implementação.

Após uma reflexão sobre os resultados da análise efetuada o objetivo foi a definição duma proposta de uma arquitetura domótica agnóstica que potenciasse o desenvolvimento de sistemas autónomos e escaláveis de gestão ambiental e de bem estar em edifícios de quaisquer dimensões. Em específico, foi estabelecido como requisito que o modelo de informação e de comunicação fosse baseado num paradigma de gestão normalizado para a *Internet*. Ou seja, uma arquitetura com mecanismos de comunicação

assentes no **Simple Networking Management Protocol** (**SNMP**), um protocolo amplamente difundido para gestão de dispositivos, serviços e aplicações distribuídas.

Como derradeiro objetivo da dissertação foi realizado o desenvolvimento, teste e avaliação de um protótipo experimental de um sistema domótico simples que servisse como uma prova adequada do modelo proposto.

1.3 Organização da dissertação

Este documento está estruturado em duas partes. Na primeira parte é feita uma introdução ao tema da domótica, sendo enunciados os principais desafios à implementação e comercialização destes sistemas na atualidade que são a principal motivação para os objetivos deste trabalho, tal como indicado no primeiro capítulo.

No capítulo seguinte é apresentado o resultado duma revisão das tecnologias mais usadas nesta área das tecnologias da informação e também uma análise crítica de trabalhos relacionados.

No último capítulo da primeira parte são então discutidos os principais problemas e limitações dos sistemas atuais, sendo introduzidos os requisitos da solução a propor na segunda parte do documento.

Assim, no primeiro capítulo da segunda parte são especificados em detalhe os requisitos da arquitetura a definir, nomeadamente o modelo de informação e protocolos de comunicação. É depois apresentado a sua especificação sintática e semântica. No capítulo seguinte é descrita a implementação dum sistema protótipo seguindo estas esta especificação e possuindo um conjunto simples mas suficiente de funcionalidades por forma a validar os conceitos.

O penúltimo capítulo discute os resultados dos testes efetuados com o sistema protótipo. Por fim, no último capítulo são apresentadas as conclusões do trabalho efetuado e sugeridos caminhos de desenvolvimento futuro.

Capítulo 2

Estado da arte

"Desde que o Homem se tornou sedentário que as habitações foram usadas como meio de abrigo e de proteção. Os edifícios tornaram-se o cerne das atividades de negócio e constituem hoje a base da vida urbana." [3]

Ao longo da história do ser-humano, o conceito de habitação/edifício foi mudando de acordo com o progresso tecnológico. Com o decorrer do tempo, a proteção outrora procurada nestes locais deixou de ser o principal foco de atenção considerando-se atualmente os edifícios como um local de trabalho, colaboração e convivência. No inicio dos anos 60 surgiram os primeiros sistemas de controlo centralizado nos edifícios, devido ao aumento da complexidade das instalações técnicas. A principal área de intervenção foram os sistemas de climatização e iluminação. Nos anos 70, a introdução de microprocessadores permitiu um alargamento do domínio de sistemas de controlo, possibilitando a automação dos processos e uma supervisão de um maior número de equipamentos. Nas décadas seguintes, os dispositivos tornaram-se mais sofisticados, surgiram novos requisitos de conforto, segurança, flexibilidade dos locais de trabalho. Apareceram novas tecnologias de telecomunicações bem como de computação informática. Neste ambiente, caraterizado por uma constante evolução, é cada vez mais importante realçar os aspetos económicos e de consumo.

Por exemplo, o consumo no sector dos edifícios representa 22% da energia final consumida em Portugal. Apesar de longe dos 40% da média comunitária, este consumo tem aumentado de forma preocupante a uma taxa de 7.5% ao ano. Este número corresponde a um consumo de energia (e consequente emissão de CO2) equivalente a 3.5 milhões de toneladas de petróleo. [4]

Estes dados apontam para uma crescente necessidade de uma gestão eficiente do dispendioso património que os edifícios representam, tirando o máximo proveito dos dispositivos e dos recursos disponíveis dando relativa prioridade à eficiência energética, à economia de escala e à interoperabilidade.

A própria transição da sociedade industrial para a sociedade informática dos tempos atuais, a necessidade de flexibilidade e de adaptação a novas tecnologias e novos requisitos para a gestão dos edifícios,

que originaram o conceito de Edifício Inteligente na década de 80 do século passado.

Este conceito estava associado ao setor dos serviços, até pelos custos elevados da sua implementação. A motivação era uma gestão energética mais eficiente, com menores custos e que também fornecesse facilidades relacionadas com conforto, segurança e telecomunicações. A transposição desta filosofia para a área de edifícios levou ao aparecimento do termo domótica (originário do termo francês *Domotique*).

A confusão entre "domótica"e "edifício inteligente"é frequente e, hoje em dia, é praticamente impossível distinguir os dois conceitos ao nível dos objetivos globais, das tecnologias usadas e funcionalidades oferecidas.

2.1 Tecnologias de controlo e comunicação para domótica

Das tecnologias específicas para utilização na implementação de sistemas domóticas as mais representativas atualmente nos sistemas de controlo em domótica, são o *X10*, o **Consumer Electronics Bus** (**CEBus**), o **European Installation Bus** (**EIB**), o **Home Bus System** (**HBS**) e o **KNX**. Adicionalmente, existem tecnologias de comunicação genéricas que podem ser aproveitadas para a implementação de sistemas domóticos, como o **ZigBee** e **Bluetooth**.

2.1.1 X10

O protocolo **X10** é um dos protocolos mais antigos padrões de rede para sistemas de automação residencial, tendo sido desenvolvido na década de 70, com o propósito de transmitir dados através da linha elétrica a baixa velocidade (**Power Line Communication** - **PLC**). [4]

A patente original expirou em 1997 tornando este um protocolo aberto. A forma como se efetua a comunicação revela algumas vantagens quando comparado a outras tecnologias, permitindo ligar os dispositivos à corrente elétrica e estarem logo aptos para realizarem as suas funções, tendo assim um custo reduzido na sua tecnologia de interligação.

O protocolo implementa uma forma simples de endereçamento que permite endereçar unicamente 256 aparelhos, ainda que, numa rede *X10* possam estar mais aparelhos, devido à possibilidade de vários módulos usarem o mesmo endereço.

O processo de comunicação baseia-se na introdução de sinais de alta frequência na corrente elétrica, representando sinais binários. O sinal é inserido logo a seguir à passagem pela origem da onda sinusoidal de 50Hz, com um atraso máximo de 200 micro segundos.

Aquando da transmissão de um comando, duas mensagens são enviadas, uma primeira que seleciona

o dispositivo e a segunda que contém a ordem para o dispositivo. Além disso todos os comandos são transmitidos duas vezes para garantir redundância na transmissão.

Contudo existem alguns problemas no uso desta tecnologia. Um deles está relacionado com o ruído introduzido pelos equipamentos na rede elétrica e os comandos podem *passar para fora* da habitação a que se destinam. Com o intuito de minimizar este risco podem ser usados filtros para atenuar o ruído e impedir que comandos sejam captados fora do edifício. Além disso, este protocolo apresenta outras desvantagens relacionadas com um número limitado de dispositivos que podem ser usados e o baixo ritmo de comunicação, pois o envio de comandos demora cerca de 1 segundo, demasiado lento para as necessidades atuais. Finalmente, a necessidade de se usar numa rede elétrica com cabos da forma tradicional e o baixo nível funcional das interações protocolares são limitações importantes na integração desta tecnologia nos modernos sistemas domóticos.

2.1.2 CEBus

O **CEBus** é uma tecnologia funcionalmente mais completa do que o **X10** e é uma arquitetura aberta, suportando diferentes meios de comunicação: rede elétrica, par entrançado, cabo coaxial, infravermelhos, rádio frequência e fibra ótica. Este protocolo atua sobre redes ponto a ponto, não necessitando de um sistema de controlo. O ritmo de comunicação do protocolo é de aproximadamente 10 *kbps* e o formato de tramas é apresentado na figura 1.

Preâmbulo	Serviços do nível lógico	Endereço Destino	Endereço Origem	Serviços do nível de rede	Serviços do nível de aplicação	Mensagem CAL	FCS
1 byte	1 byte	2 bytes	2 bytes	1 a 8 bytes	1 a 2 bytes	n bytes	1 byte

Figura 1: Formato de tramas do protocolo **CEBus** [1]

Um endereço é uma grandeza de 32 *bits* em que os 16 *bits* mais significativos são usados para identificar um sistema e os 16 *bits* menos significativos identificam um dispositivo desse sistema. Cada sistema pode assim possuir até 65 536 endereços (2^{16}), dos quais alguns estão reservados. Por exemplo, o endereço 0 é usado em tramas de divulgação *(broadcast)*, que são recebidas por todos os nós do mesmo sistema. [1]

Esta tecnologia permite vários dispositivos usarem subconjuntos de funcionalidades, suporta distribuição de áudio e vídeo em formatos analógicos e digitais, utiliza uma estratégia distribuída e permite adição ou remoção de dispositivos sem interrupção. Como principais desvantagens temos o facto de funcionar

a um nível protocolar abaixo da camada de rede, tornando complexa a sua integração com sistemas domóticos modernos baseados num paradigma aplicacional **TCP**/**IP**.

2.1.3 LonWorks

É um sistema desenvolvido pela empresa norte-americana *Echelon Corp* para automação residencial. Pode operar até 32 000 dispositivos sendo a peça fundamental o *Neuron-Chip* que possui 3 processadores de 8 até 32 *bits*, 2 deles para comunicação (1 para **MAC** outro para o protocolo LONTalk) e o terceiro dedicado à aplicação. Como o protocolo fica no processador reduz custos, aumenta velocidade e o desenvolvimento de aplicações.

Cada nó na topologia possui computação local, fontes próprias e pode ser ligado a diversos dispositivos sensores e atuadores através de par entrançado (velocidade de 78 *kbit/s*), Ethernet (Velocidade de 100 até 1000 *Mbit/s*), Radiofrequência (distâncias até 3,2 km, velocidade de 9.6 a 128 *kbit/s*) e *Powerline* (PL-20 a 5 *kbit/s* e PL-30 a 2 *kbit/s*).

Com este sistema de automação é possível ter sensores e atuadores equipados com a sua própria inteligência trocando informações entre si, não se precisando sempre de uma unidade central de controlo. O processamento da informação pode ser feito localmente. Com a hipótese de utilização de tecnologias sem fios, permite minimizar a quantidade de cablagem necessária.

De notar que o custo dos dispositivos é considerável e a necessidade do uso do *Neuron Chip* bem como ferramentas específicas de configuração e teste é um fator negativo do uso destes sistemas. Além disso, a integração com outros sistemas, incluindo os baseados em aplicações *Internet*, é dificultado pois trata-se de uma tecnologia proprietária. [5]

2.1.4 Insteon

Este protocolo para automação residencial para controlo local e remoto de sistemas de iluminação, estores, climatização, deteção de movimento, abertura de portões, etc. A comunicação entre os dispositivos utiliza uma rede *peer-to-peer* com duplo canal de comunicação para aumentar a fiabilidade. Utiliza a rede elétrica existente e redes sem fios para todo o tipo de aplicações domésticas. Os dispositivos podem repetir mensagens ao longo da rede, eliminando o estrangulamento que pode ocorrer caso um dispositivo falhe. Quantos mais dispositivos a rede possuir maior a sua fiabilidade e para evitar uma saturação de mensagens na rede devido ás repetições, o protocolo limita a três o número de repetições de mensagens. [6]

As principais mais valias desta tecnologia são a sua fiabilidade, escalabilidade e baixo custo. No

entanto, tem como principais desvantagens ser uma tecnologia proprietária e que utiliza um protocolo não aplicacional tradicional o que dificulta a interligação com aplicações e dispositivos terceiros.

2.1.5 EIB/KNX

O **European Installation Bus** é um protocolo desenvolvido pela *European Installation Bus Association* com o objetivo de criar uma norma europeia que permitisse a comunicação entre todos os dispositivos de uma instalação domótica sem recorrer a tecnologias de rede local **TCP**/**IP**.

Utiliza uma arquitetura descentralizada, esta solução define uma relação elemento a elemento entre os dispositivos, o que permite distribuir a inteligência entre os sensores e atuadores instalados. [7] Mais tarde, baseado no **EHS** (**European Home System**) [8] e *Batibus* [9] nasce o protocolo **KNX**. [10] Este protocolo é aberto e normalizado, adaptando a pilha de comunicações do **EIB** e herdando a compatibilidade com os meios físicos do *Batibus* e **EHS**.

A rede distribuída utilizada pelo **KNX** pode ter até 65536 dispositivos e pode usar 5 meios físicos para interligação dos dispositivos: par entrançado (taxa de transferência de 9600 *bit/s*); rede elétrica (taxas de transmissão PL110 de 1200 *bps* e PL132 de 2400 *bps*; rádio e infravermelhos (sinais de rádio na banda de 868 MHz e taxa de transmissão até 16384 *kbps*, e, no caso dos infravermelhos, com taxa de transmissão de até 7 *kbps*; *Ethernet* (de forma a possibilitar a interligação entre redes **IP** e redes **KNX**, foi desenvolvido o protocolo *KNXnet/IP* que permite estabelecer uma ligação ponto-a-ponto entre um *router/gateway KNXnet/IP* e os *hosts* numa rede **IP**, o encaminhamento de segmentos entre redes é também suportado, sendo usado o endereço *multicast* reservado 224.0.23.12.).

A **KNX** é uma norma internacional aprovada por várias entidades mundiais e pode ser usado para todas as aplicações/funções possíveis em edifícios novos ou existentes, sendo estas instalações facilmente ampliadas e adaptadas a novas necessidades. Existe ainda um controlo de qualidade rigoroso na certificação dada aos fabricantes para produzirem dispositivos com esta tecnologia o que garante uma qualidade mínima dos produtos disponíveis no mercado que usam esta tecnologia. As principais desvantagens são o custo relativamente elevado dos equipamentos e a complexidade na interligação com dispositivos que implementem outras tecnologias e com aplicações desenvolvidas no paradigma **TCP**/**IP**.

2.1.6 Z-Wave

Esta é uma tecnologia de automação domótica sem fios criada e desenvolvida pela companhia *Zensys*. [7]

Utiliza um protocolo bidirecional, juntando todos os dispositivos numa rede *"mesh"*. O controlo pode

ser feito através de aplicações Internet. Destaca-se pelo seu baixo consumo e alcance de atuação. Atua numa estreita banda fora dos 2,4 GHz, o que faz com que não sofra interferências de redes sem fios e sistemas *Wi-Fi* ou *Bluetooth*.

Uma das grandes bandeiras do desenvolvimento desta tecnologia foi a criação da *Z-Wave Alliance* [11], uma aliança criada em 2005 por um grupo dos principais fabricantes de produtos de controlo e automação residencial, devido sobretudo ao descontentamento provocado pela forte fragmentação tecnológica na área. A interoperabilidade entre fabricantes é exigida pela entidade reguladora *Z-Wave*, permitindo aos produtores a especialização, garantido ao utilizador a compatibilidade entre marcas. Começou como um protocolo proprietário mas desde 2012 devido à grande recetividade, tornou-se uma norma standard aberta e pública, possibilitando ao utilizador deixar de estar preso a uma marca ou protocolo proprietário.

O modelo define 2 tipos de equipamentos principais: os controladores e os dispositivos "escravos". Apenas os controladores sabem construir e gerir uma rede. Existe um controlador primário onde são inseridos novos nós e os outros podem ser portáteis ou estáticos, mas apenas os estáticos ligados á corrente elétrica conseguem reproduzir a mensagem para os nós vizinhos.

Relativamente ás comunicações, o alcance ronda os 30 metros sem barreiras (se houver paredes cerca de 15 metros) a um ritmo máximo de 9600 *bps* (opera nas frequências de 866,42 MHz na Europa).

2.1.7 Thread

É um novo protocolo de comunicações sem fios desenvolvido pelo *Thread Group*, constituído por grandes empresas como *Samsung, Google, Apple, Qualcomm, Tyco*, entre outras. Usa pouca energia, é *IP-based*, e utiliza a topologia *"mesh"* permitindo que os dispositivos com bateria comuniquem entre si e com a *cloud*.

Opera usando a norma *IEEE 802.15.4*, na largura de banda de 2.4 GHz e todos os dispositivos estão interligados numa rede **IP**, significando que estão conectados entre si mas também podem estar ligados à Internet. Sendo desenhado especialmente para automação de residências/edifícios é similar ao *Wi-Fi* mas num modo de pouco consumo e pouca *bandwidth*, o alcance prático é cerca de 30 metros e transmissão de dados pode alcançar 250 *Mbps*. [12]

Apresenta as mesmas características que o *Z-Wave* e o *ZigBee* relativamente ao pouco uso de energia e uso duma rede *"mesh"*. No entanto, o principal beneficio está relacionado com a conectividade associada ao protocolo **IP**.

O grande problema do *Thread* é a sua introdução recente, ainda existindo um número limitados de equipamentos disponíveis compatíveis com a tecnologia.

2.1.8 MQTT

O protocolo **MQTT** (**Message Queuing Telemetry Transport**) é um protocolo de comunicação para dispositivos **IoT** que foi desenvolvido com o intuito de ser leve, e eficiente e escalável utilizando o modelo de publicação/assinatura para permitir que dispositivos enviem e recebam mensagens de forma assíncrona através da utilização de uma entidade intermediária e central designada de *broker*. [13]

Os dispositivos utilizam o modelo cliente/servidor, onde um dispositivo (cliente) se conecta a um *broker* **MQTT** para enviar e receber mensagens. O cliente pode publicar mensagens num tópico de destino, e outros clientes que tenham subscrito aquele tópico receberão as mensagens. Além da eficiência e baixo consumo de energia do protocolo, a tecnologia pode ser implementada por cima de redes **TCP**/**IP**, *Wi-Fi* ou **Bluetooth**. Por fim, o **MQTT** oferece recursos de segurança, como autenticação e encriptação de dados, para proteger as mensagens transmitidas entre as suas entidades.

2.1.9 CoAp

O **CoAP** (**Constrained Application Protocol**) é um protocolo de aplicação web criado especificamente para dispositivos **IoT** e redes com recursos limitados. Foi também concebido para constituir uma alternativa mais leve e eficiente ao **HTTP**. Esta tecnologia utiliza um modelo cliente/servidor para permitir aos dispositivos comunicarem com servidores na *cloud* ou outros dispositivos. Tal como o **HTTP** utiliza o formato de mensagem **REST**full (**Representational State Transfer**) para permitir a comunicação entre os dispositivos e servidores usando operações como *GET*, *PUT*, *POST* e *DELETE*. [14]

Uma das vantagens mais relevantes do **CoAP** deve-se à sua projeção para funcionar em redes com recurso bastante limitados, como, por exemplo, dispositivos com pouca memória ou pouca largura de banda. Além da utilização já referida do formato das suas mensagens suporta ainda várias opções de segurança, como criptografia de dados e autenticação de dispositivos.

Finalmente, o protocolo pode funcionar em vários tipos de redes, incluindo redes sem fios. Pode ainda ser implementado por cima de **UDP** ou **DTLS** (**Datagram Transport Layer Security**).

2.2 Tecnologias de comunicação genéricas

Existem várias tecnologias comuns para a interligação de dispositivos em redes locais e pessoais que podem ser usadas também em arquiteturas de sistemas domóticos. As vantagens da sua utilização são o baixo custo, a conveniência e a facilidade de interligação entre dispositivos de aplicação controladores.

2.2.1 Wi-Fi

Sendo o *Wi-Fi* muito comum nos edifícios faz sentido que sistemas domóticos tirem partido desta tecnologia. No entanto, não é o protocolo mais indicado para o todos os tipos de dispositivos.

As comunicações são feitas na largura de banda de 2.4 GHz ou 5 GHz e a distância teórica é relativamente longa, mas em termos práticos a distância deve ser no máximo de 60 metros.A taxa de transmissão vai desde 10 a 100 *Mbps*. Suporta teoricamente um limite de 256 dispositivos, mas, a congestão de rede pode ser um problema antes desse limite ser atingido, sobretudo se um dispositivo necessitar de maior largura de banda.

As vantagens associadas a esta tecnologia relacionam-se com a elevada taxa de transmissão e a facilidade da instalação de um ponto de acesso em comparação com a instalação de redes cabladas num edifício inteiro. Apresenta, no entanto, algumas limitações, que se prendem com as interferências que esta tecnologia gera, em espaços com muitos dispositivos a competir por largura de banda. Além disso os dispositivos *Wi-Fi* apresentam um grande consumo de energia para operarem. A distância e a segurança são também uma limitação a considerar.

O grupo responsável pela norma é o *Wi-Fi alliance* que reconhece as limitações do seu protocolo e por isso, introduziu o *Wi-fi Ha-Low*, direcionado ao mercado de edifícios inteligentes, desenvolvido para consumir pouca energia e aumentado o alcance de comunicações. No entanto, são necessários equipamentos específicos para tirar partido desta nova tecnologia.

2.2.2 Wi-Fi Ha-Low (IEEE 802.11ah)

Protocolo desenvolvido pela *Wi-Fi Alliance* totalmente orientado para o **Internet of Things** (**IoT**), com inúmeras vantagens quando comparado á maioria dos restantes protocolos comunicacionais de rede local, incluindo o *Wi-Fi 6* tradicional.

A grande diferença desta tecnologia deve-se ao facto de operar na largura de banda inferior a 1 GHz permitindo uma distância maior e menor consumo de energia.

2.2.3 Bluetooth

Apesar de atualmente a grande maioria de dispositivos **IoT** necessitar de conectividade *Wi-Fi*, existe já uma quantidade relevante de equipamentos que incluem conectividade *Bluetooth* para comunicações a curtas distâncias. Esta tecnologia providência transmissão de dados com maior ritmo do que o *Z-Wave* ou *ZigBee*, embora a distâncias máximas mais reduzidas. Como a maioria dos *smartphones* possui esta

tecnologia, facilita a conexão de dispositivos ao telemóvel ou *tablet* e é maioritariamente conhecido como tecnologia base para o *streaming* de áudio para auscultadores sem fios, colunas ou sistemas de som portáteis ou em automóveis. [15]

Este sistema funciona com um *setup Master/Slave*, podendo os dispositivos mudar de cargo, havendo sempre apenas um *master*. Consome pouca energia, significando que os dispositivos podem operar a bateria mas com a desvantagem de se ter pouco alcance.

Usa a banda de 2.4 *GHz* e as taxas de transmissão maiores que outros protocolos sem fios de baixo consumo, mas bastante menores do que o *Wi-Fi*. O *Bluetooth 3.0 e 4.0* pode atingir até a taxa de 24 *Mbps* e o alcance varia com base na classe de rádio do dispositivo (classe 3 cerca de 1 metro, os mais comuns classe 2 de 5 metros até 10 dependendo do ambiente e, os menos comuns, classe 1 com alcance de até 30 metros). [16]

Outra vantagem relativa desta tecnologia é o método de emparelhamento, que torna fácil para os dispositivos a descoberta e conexão entre si. Cada dispositivo tem um perfil com as suas características e os restantes, próximos o suficiente, conseguem ver esse perfil, prevendo-se a possibilidade de escolha na conexão a efetuar. Além disso, é uma tecnologia financeiramente atrativa.

As limitações prendem-se com operar na largura de banda muito concorrida (2.4 GHz) ficando suscetível a interferências e os ritmos máximos não são concorrênciais com o *Wi-Fi*, não permitindo, por exemplo, transmissão de vídeo.

2.2.4 Bluetooth Low Energy (LE)

Este protocolo foi desenhado para baixar ainda mais o consumo de energia do **Bluetooth** normal, transmitindo dados em mais de 40 canais na banda de frequência 2.4 GHz. Uma grande bandeira do protocolo é o facto de suportar vários tipos de topologia de comunicação, expandindo de *point-to-point* para *broadcast* e introduzindo a *mesh* tal como no *Z-Wave* e *ZigBee*, permitindo dispositivos trocarem mensagens entre si por saltos, o que leva inevitavelmente ao aumento do alcance. [17]

É frequentemente usado para o posicionamento de dispositivos com o propósito de responder à demanda cada vez maior de alta precisão em localizar serviços em espaços interiores. Inicialmente suportava capacidades de presenças simples e de proximidade, mas suporta agora a tecnologia *Bluetooth Direction Finding* e, medição de distancias com precisão elevada.

2.2.5 ZigBee

O *ZigBee* é um simples protocolo de redes sem fios que tem como objetivo ajudar na implementação de aplicações simples e autónomas (com a necessidade mínima de interação humana). As aplicações tendem a ter baixos requisitos de transmissão de dados e de baixo consumo energético. Tarefas simples de controlo de temperatura e luminosidade são bons exemplos de sistemas ideias para a utilização desta tecnologia.

Devido às suas características o **ZigBee** é normalmente comparado com outras tecnologias que utilizam redes sem fios como redes *Bluetooth*e *Wi-Fi*. Geralmente, uma rede destas é constituída por um coordenador, um ou mais dispositivos e, opcionalmente, um ou mais encaminhadores. O nó coordenador, sendo responsável por criar a rede *ZigBee*, começa por verificar o canal rádio com menos atividade para evitar interferências. Em seguida, a todos os nós é atribuído um endereço fixo **MAC** de 64 *bits* e um endereço dinâmico de 16 *bits*. Inicialmente atribui a si o endereço *0x0000* e aguarda por pedidos de nós para se juntarem à rede, podendo este pedido ser feito diretamente ao coordenador ou através de um encaminhador vizinho. [18]

As maiores desvantagens da tecnologia são relativas à interferência que é causada pela utilização com outros dispositivos *Bluetooth* e redes *Wi-Fi*, baixas taxas de transferência e dificuldade na interligação com sistemas aplicacionais tradicionais baseados no paradigma **TCP**/**IP**.

2.2.6 Comparação de protocolos

Protocolos	Frequência	Taxa de transmissão	Alcance	Tamanho da rede	Topologia	Encriptação
Wi-Fi 802.11n	2.4-5.8 GHz	450 Mbps	10-100 m	Milhares	Star, tree, P2P, mesh	Nenhuma
Bluetooth	2.402-2.48 GHz	0.7-2.1 Mbps	15-20 m	8	Star	Nenhuma
BLE	2.402-2.48 GHz	2 Mbps	10-15 m	N/A	Star	Nenhuma
ZigBee	869/915 MHz; 2.4 GHz	20/40 kbps, 250 kbps	10-100 m	65536	Star, mesh, cluster-tree	AES-256
Z-Wave	868/921 MHz	10-100kbps	30-50 m	232	Mesh	DES-56, AES-128
Ethernet	100-500 MHZ	1 Mbps-100Gbps	100 m	N/A	Bus, Star	Nenhuma
X10	120 kHz; 310-433.92 MHz	20-60 bps; 9.6 kbps	500-1000 m	256	Nenhuma; Stat	Nenhuma
INSTEON	131.65; 868-924 MHz	13.165 kbps; 38.4 kbps	40-500 m	256	P2P, mesh, dual mesh	AES-256
KNX	110/132 kHz; 868.3 MHz	1.2/2.4 Mbps	1000m;100m	15000	Tree, line star	Nenhuma, AES-128
LonWorks	60 kHz - 10 MHz	10 kbps	100 m	32000	Star, tree, bus	DES-56, AES-256
CEBus	902-928 MHz; 2.4-2.5 GHz	10 kbps	100 m	65536	P2P	Nenhuma

Figura 2: Comparação de Protocolos

2.3 Limitações tecnologias tradicionais dos sistemas domóticos

Com a existência de várias tecnologias incompatíveis entre si, a expansão da domótica tem sido limitada ao desenvolvimento de sistemas fechados proprietários. A falta de interoperabilidade entre dispositivos e entre dispositivos e sistemas controladores de diferentes fabricantes ou suportando tecnologias base diferentes, dificulta a separação no desenvolvimento específico dos dispositivos e das aplicações de controlo. Esta separação permitiria o desenvolvimento de aplicações de controlo por especialistas de *software*, sem preocupações tecnológicas de baixo nível, potenciando a integração de mecanismos e paradigmas avançados de gestão de sistemas distribuídos, incluindo tecnologias de inteligência artificial que aumentem o nível funcional, de automação e autonomização dos sistemas domóticos.

As soluções habituais consistem em sistemas desenvolvidos apenas para uma determinada tecnologia e com um comportamento já pré-definido incorporado nos equipamentos do sistema completo de *software* e *hardware*.

A médio e longo prazo devem ser definidas arquiteturas que incorporem tecnologias normalizadas e abertas permitindo o desenvolvimento de aplicações com vários níveis funcionais e de automação a baixo custo, sem necessidade de *hardware* especial domótico. Este objetivo pode ser atingido utilizando paradigmas de gestão já usados em larga escala na gestão de equipamentos, serviços e aplicações na Internet, como, por exemplo, o modelo **SNMP**. A sua adoção obrigaria a uma adaptação dos dispositivos dos sistema domóticos para suportarem o protocolo **SNMP** sobre uma rede **TCP**/**IP** (o que já ocorre hoje em dia em muitos tipos de equipamentos domóticos). O desenvolvimento e integração de aplicações de controlo seguindo o paradigma cliente/servidor da Internet seria, assim, a solução preferencial. A sua instalação num qualquer sistema computacional doméstico também seria facilitado, baixando custos e aumentando a conveniência. Este paradigma também facilitaria a ligação a sistemas remotos (na *cloud*, por exemplo) ou a sistemas autónomos inteligentes.

2.4 SNMP

O protocolo surge inicialmente com a necessidade de criar mecanismos que permitissem gerir a rede *TCP/IP*. Devido à sua simplicidade e ao grande poder de gestão de redes heterogéneas observado, levou a que fosse utilizado para a gestão de muitos tipos de sistemas. Normalmente o **SNMP** é associado à gestão de equipamentos **IP**, mas é importante mencionar que pode ser utilizado para gerir qualquer tipo

de dispositivo, serviço ou aplicação distribuída. [19]

A adoção massiva deste protocolo levantou desafios de segurança e novas funcionalidades de automação, levou à atualização do protocolo, surgindo assim o *SNMPv2* com novas primitivas protocolares e com a utilização da **Structure of Management Information version 2** (**SMIv2**) [20]. Mais tarde é lançado o **SNMPv3**, que não trouxe evoluções tecnológicas, focando-se sobretudo em aspetos relacionados com a imposição de mecanismos de segurança já incluídos no *SNMPv2* e que implementam privacidade, controlo de acesso e autenticação. De salientar, que a segunda versão que utiliza a autenticação frágil por *community strings* é conhecida por **Simple Networking Management Protocol version 2 community** e é atualmente a versão mais utilizada. [21]

O **SNMP** é um protocolo não orientado à conexão, assíncrono e assimétrico, que permite a troca de informações de gestão entre dispositivos de rede usando predominantemente o protocolo de transporte **UDP** (**User Datagram Protocol**) mas com a possibilidade de escolher outros menos convencionais. [22] Estas características facilitam a implementação do protocolo, não existindo a complexidade extra do *overhead* associado á confirmação de mensagens recebidas. [23]

Um problema relevante pode ser a dependência da utilização das redes **IP** que, em caso de falha, torna impossível qualquer tipo de comunicação entre os elementos do sistema, os gestores e os agentes. O gestor SNMP é responsável por enviar comandos ao agente, a pedir ou a alterar informação de gestão. O gestor pode também receber notificações não solicitadas do agente.[19] O agente é a entidade que está presente nos dispositivos a gerir, tendo como função principal receber e processar os comandos provenientes do gestor. Se for pedida uma informação o agente envia a resposta e se for pedido a alteração dum valor, o agente tenta alterar o valor e confirma ao gestor o resultado da operação. Estes procedimentos são possíveis atravées da manipulação de objetos de gestão implementados numa **Management Information Bases** (**MIB**). [24]

2.4.1 Formato das mensagens SNMP

As mensagens trocadas entre entidades **SNMP** estão divididas em duas partes: o cabeçalho e o corpo. No cabeçalho encontra-se a versão do protocolo e o nome da comunidade que identifica uma comunidade **SNMP**. O corpo da mensagem ou *PDU* é onde se situa a informação relevante . Na primeira versão existiam cinco tipos diferentes de **PDU**, *GetRequest*, *GetNextRequest*, *GetResponse*, *SetRequest* e *Trap*. [25] Na segunda versão introduziram-se dois novos tipos, o *GetBulkRequest* e o *InformRequest*. Finalmente, na terceira versão, não existiu mais nenhum novo tipo de **PDU**, focando-se sobretudo em campos do cabeçalho relativos à segurança.

2.4.2 Operações/primitivas SNMP

Após os diferentes elementos da rede estarem configurados, os gestores podem requisitar operações que servem na maioria das vezes para consultar ou alterar informações da **MIB**. Os agentes podem responder aos gestores ou reportar alguma situação relevante para a qual tenha sido configurado. As operações seguintes estão disponíveis em todas as versões:

- Operação de leitura (*Get*) - Usado pelo gestor para obter o valor duma instância de um objeto específico duma **MIB** implementada pelo agente;
- Operação de travessia (*Get-Next*) - Usado pelo gestor para obter o valor da próxima instância de um objeto duma **MIB** implementada pelo agente;
- Operação de configuração (*Set*) - Usado pelo gestor para definir o novo valor de uma instância de um objeto duma **MIB** implementada pelo agente;
- Operação de consulta múltipla (*Get-Bulk*)- Usado pelo gestor para obter de forma eficiente grandes quantidades de informação de objetos duma **MIB** implementada pelo agente. Esta primitiva foi introduzida no **SNMPv2**.

  Existe também operações de diferentes notificações para informar o gestor de algum evento importante. As operações seguintes foram definidas no SNMPv1:

- Operação de notificação (*Trap/Notification*) – Usada para enviar uma mensagem não solicitada de um agente para um gestor. O tipo de evento que pode ser reportado é limitado pelo protocolo de forma a manter a sua simplicidade. A operação de *Trap* foi atualizada para *Notification* no **SNMPv2**;

  A seguinte operação de notificação foi adicionada no *SNMPv2c*:

- Operação de informação (*Inform*) – Usada para enviar uma mensagem não solicitada de um agente para um gestor ou de gestor para outro gestor. Esta primitiva foi também introduzida no **SNMPv2**.

2.4.3 MIB

A **MIB** é um dos aspetos mais relevantes da arquitetura uma vez que especifica um *interface* para a manipulação dos objetos de gestão implementados por um agente **SNMP**. Essa manipulação permite a monitorização e configuração dos equipamentos, serviços e aplicações onde o agente reside.

Uma **MIB** é uma coleção de informação organizada hierarquicamente. Existem dois tipos de objetos numa **MIB**, escalares, que permitem a manipulação de objetos com uma única instância, e objetos compostos (ou tabulares) que permitem manipular vários objetos de instâncias relacionados e agrupados em tabelas.

A informação contida na **MIB** define os recursos que o sistema pode gerir. A especificação da **MIB** é realizada em **ASN.1** (**Abstract Syntax Notation One**) [ISO8824]. Esta é uma linguagem declarativa que serve de base para todas as declarações e definições na norma **SMI** e nas próprias **MIB**. Este fator simplifica o trabalho de codificação de informação pois permite a utilização da norma **BER** (**Basic Encoding Rules**) [ISO8825], como mecanismo de codificação e transmissão das mensagens **SNMP**.

A **Structure of Management Information** (**SMI**) define os tipos de objetos que uma **MIB** conter, ou seja, define a linguagem de especificação dos componentes de uma **MIB** (linguagem formal para descrever os objetos geridos). A estrutura específica com que todos os objetos presentes nas **MIB** são organizados é uma árvore onde cada nó possui um único identificador, **OID** (**Object Identifier**), representado por uma sequência hierárquica de números inteiros. [26]

Cada objeto definido numa **MIB** pode ter 3 tipos de acesso: *read-only* (o valor só pode ser consultado), *read-write* (o valor pode ser escrito ou lido) e *read-create*(o valor pode ser lido, escrito ou criado), este último tipo de acesso é especial e só pode ser usado na construção de tabelas dinâmicas.

Oficialmente, a **SMIv2** define os tipos de dados permitidos nas **MIB**, dividindo-se em tipos simples e tipos *application-wide* (ou escalares) e tipos simples construídos [27]. Como exemplos de tipos simples temos:

- *Integer* – Número inteiro positivo ou negativo;
- *OctetString* – Utilizado para especificar octetos de informação textual ou binária,
- *ObjectID* – Valor único usado para identificar um único objeto;
- *Bits* – Definido apenas na **SMIv2** para representar zero ou mais *bits* que especificam um valor.

Como exemplos dos tipos *application-wide* temos:

- *IP Address* – tipo de dados usado para endereços **IPv4**;
- *Network Address* – representa um endereço de uma família protocolar particular;
- *Counter* – valor inteiro não negativo que pode ser apenas incrementado, quando o valor máximo é atingido, o contador recomeça do zero;

- *Gauge* – valor inteiro não negativo que pode ser incrementado ou decrementado;
- *Timetick* – centésimos de segundo desde um evento;
- *Unsigned integer* – número inteiro não negativo.

Por fim, como exemplos dos tipos compostos, temos:

- *Row* – referencia uma linha numa tabela, cada elemento da linha pode ser do tipo simples ou *application-wide*;
- *Table* – referencia uma tabela com zero ou mais linhas, cada linha tem o mesmo número de colunas.

2.4.4 Ontologias

Em filosofia, ontologia é o estudo do tipo de coisas que existem. Com a emergência da *Semantic Web* o termo adquiriu um novo significado na ciência dos dados. Ao longo dos últimos anos o conceito foi alvo de especial atenção por diferentes áreas, que observaram um grande potencial nas ontologias para dotar os seus sistemas de inteligência artificial.

Ontologia é um vocabulário de representação, normalmente especializado num certo domínio. Mais precisamente, não é o vocabulário que qualifica a ontologia mas as conceptualizações que os termos no vocabulário pretendem capturar [28]. Ou seja, uma ontologia é uma forma de guardar, organizar e representar o conhecimento, é a definição de um conjunto de conceitos, a sua taxonomia, as relações entre si e as regras que gerem tais conceitos [29]. Além disso, uma ontologia deve ser especificada formalmente pois pode ser lida e interpretada tanto por humanos como por máquinas.

A grande funcionalidade que uma ontologia bem formulada pode apresentar é a capacidade de se implementar sistemas capazes de raciocinar sobre os dados armazenados gerando nova informação automaticamente [30].

Várias linguagens têm sido utilizadas para representar o conhecimento numa ontologia. O seu objetivo é especificar a um nível abstrato e conceptual um domínio de interesse. Estas linguagens devem cumprir os seguintes requisitos [31]:

- Possuir uma semântica bem definida;
- Ser intuitiva para o humano;
- Ter uma sintaxe;

- Incluir propriedades de raciocínio;
- Ser capaz de representar conhecimento humano;
- Ter o potencial para construir bases de conhecimento;
- Ser normalizada, com objetivo de assegurar interoperabilidade.

Atualmente existem várias linguagens para ontologias, tanto proprietárias como normalizadas, tais como a linguagem de especificação algébrica comum, lógica comum, *CycL*, *DOGMA*, *Gellish*, *IDEF5*, *KIF*, *RIF* e *OWL*. Estas linguagens podem ser classificadas como linguagens lógicas que incluem lógica de primeira ordem, lógica baseada em regras ou lógica de descrição, podem ser baseadas em *frames*, semelhantes a bases de dados relacionais ou também baseadas em grafos [32].

Na área de domótica, a utilização das ontologias pode ser importante para permitir o uso de técnicas de inteligência artificial com o intuito de implementar sistemas cada vez mais automáticos e autónomos. No entanto, o principal entrave à sua integração em sistemas domóticos tem haver com a falta de interfaces entre a informação das ontologias e a informação que é possível manipular nos sistemas de controlo atuais. Tal interface é muito complexo de implementar devido à grande disparidade no nível de abstração da informação. A introdução dum modelo de gestão aplicacional intermédio tal como um sistema baseado na tecnologia **SNMP**, pode ajudar a diminuir relevantemente as diferenças de abstração conceptual entre a informação disponível nos dispostivos domóticos e a informação dos sistemas ontológicos. Um sistema de gestão baseado na tecnologia de gestão **SNMP** poderia, assim, servir de *gateway* informativo e funcional.

2.4.5 Projetos Relacionados

A utilização do protocolo **SNMP** na área da domótica e na recolha de dados de rede pode possibilitar o controlo e introduzir as facilidades para um cientista de dados conseguir analisar e efetuar configurações, sendo agnóstico das tecnologias de rede usadas.

O estudo [33], analisa a capacidade do protocolo **SNMP** aplicado para controlo e monitorização de diferentes tipos de dispositivos. Comprovou-se, através de experiências em ambiente laboratorial na obtenção e alteração de informações sensoriais, a inexistência de um atraso significativo apenas para sistemas que requeiram repostas em tempo real longe dos requisitos dos sistemas domóticos.

A análise da aplicabilidade do **SNMP** no contexto de sistemas domóticos sobre tecnologias de comunicação sem fios foi estudada em [34]. Recolheram-se informações do ambiente através de sensores

(temperatura, luz e voltagem) que eram enviadas dum módulo sensor base (utilizando comunicação wireless *IEEE 802.15.4*) para um agente **SNMP**. Através da utilização das operações do **SNMP**, o gestor efetua depois consultas ao agente e apresenta as informações recolhidas. Os resultados observados permitiram concluir que a utilização do protocolo é bastante útil não só pela eficiência da transmissão e facilidade de armazenamento dos dados, mas sobretudo pela sua independência do tipo de sensores e protocolos de domótica utilizados.

Noutro estudo, [35], o autor investigou sobre a possibilidade de se utilizar o protocolo **SNMP** em conjunto com o uso de protocolos como o **MQTT** e o **CoAP** suportados pelos dispositivos domóticos. Dando especial atenção ao protocolo **MQTT** o autor propõe um sistema capaz de preencher a lacuna existente entre o **SNMP** e **MQTT**, através de elementos coletores, servidores e vários clientes **SNMP** tornando o sistema escalável. Analisando os resultados das experiências dum protótipo, do **SNMP** com as tecnologias **MQTT** e **CoAP**.

Tal como foi referido, as ontologias podem vir a ser usadas com o intuito de aumentar a interoperabilidade de sistemas, automatizar e autonomizar o controlo de tarefas de gestão de redes e, sobretudo, criar ambientes inteligentes capazes de inferir ações a tomar mediante uma base de conhecimento. Sendo estes objetivos importantes da domótica moderna, é fácil perceber que as ontologias têm vindo a ser introduzidas em projetos desta área.

No projeto de casa inteligente [36], os autores recorreram a quatro ontologias com vista a dotar o sistema domótico de inteligência, tornando-o capaz de se ajustar dinamicamente ao ambiente sem necessidade de o programar manualmente. Nas quatro ontologias contextualizam-se quatro domínios: os dispositivos, o ambiente, as funções e as preferências.

Capítulo 3

Problema e os seus desafios

Parte II

Core da Dissertação

Capítulo 4

Contribuição

Nesta capítulo é apresentada a arquitetura geral da solução proposta como solução para se atingirem os objetivos estabelecidos para os trabalhos desta dissertação.

Em particular, é definido um modelo baseado na arquitetura **SNMP**, definindo-se uma **MIB** específica para funcionalidades de sistemas domóticos. Este sistema tradicional de gestão aplicado a equipamentos, serviços e aplicações de rede, nomeadamente no contexto da Internet, também já foi testado no contexto dos sistemas domóticos [37] [38] [35], mas nunca foi adotado pela indústria que sempre tentou soluções funcionalmente mais próximas dos dispositivos domóticos do que dos sistemas de controlo, i.e, sempre deram primazia à facilidade de implementação das tecnologias nos dispositivos do que às capacidades de interoperabilidade e às funcionalidades avançadas dos sistemas de controlo e interface com o utilizador.

A forma encontrada para tentar resolver estes problemas foi a criação de um novo sistema complementar designado de **Light SNMP** (**L-SNMP**) que inclui uma nova definição de agente de gestão, ainda que baseado no paradigma dos agentes **SNMP**, implementando um tipo de **MIB** especial (**Light MIB** ou **L-MIB**) para objetos de gestão domótica.

O modelo **L-SNMP** é definido à custa dum novo tipo de **MIB**, permitindo assim implementações computacionalmente e de comunicações menos exigentes no lado dos fabricantes de dispositivos. Assim o **L-SNMP** permite a inclusão de um nível funcional intermédio entre os dispositivos domóticos e os agentes **SNMP** tradicionais. Esta nova solução também potencia uma definição funcional de mais alto nível para os objetos de gestão da **MIB** domótica dos agentes **SNMP**, facilitando a sua integração ou interação com sistemas de controlo baseados em tecnologias de inteligência artificial.

4.1 Arquitetura geral

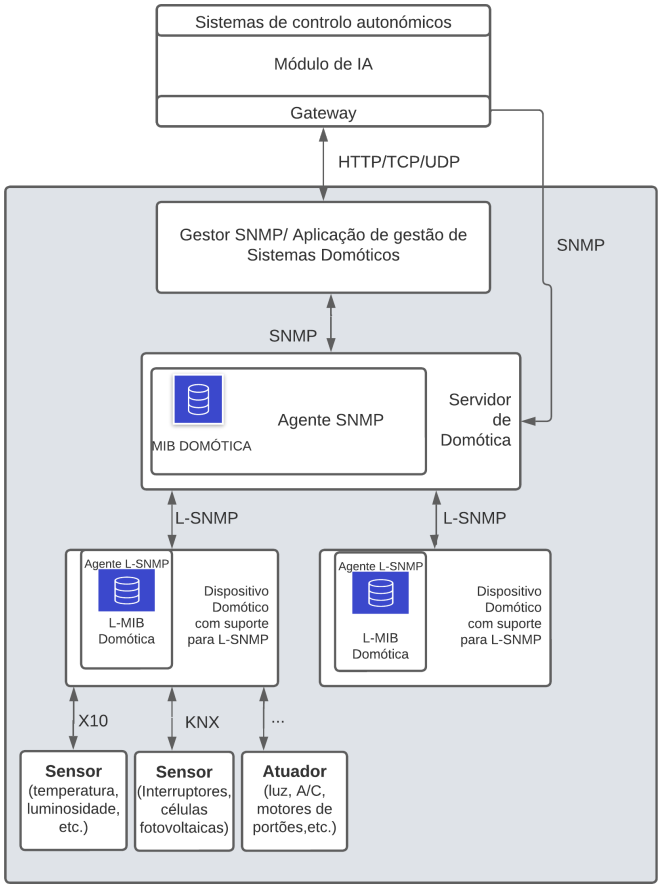


Figura 3: Arquitetura geral do sistema

Na figura 3 é apresentada a arquitetura geral do sistema definido, que é formada pelas entidades dispositivos, agentes, gestores e o módulo de **AI** que não faz parte do âmbito da dissertação mas encontra-se presente para demonstrar a possível interação dos gestores com o este tipo de módulo para dotar o sistema domótico de maior inteligência funcional.

Os dispositivos podem ser de qualquer tipo, como por exemplo uma televisão, ar condicionado, colunas de som, estores ou luzes. Cada dispositivo pode conter vários sensores responsáveis por capturar informações sobre o meio em seu redor e atuadores responsáveis por efetuar ações quando comunicadas pelo agente. Um dispositivo tipicamente conecta-se a um agente **SNMP**, ou gestor **L-SNMP**, através do **L-SNMP**. Os agentes **L-SNMP** dos dispositivos domóticos implementam um serviço de *middleware*

essencial na implementação da instrumentação dos objetos de gestão dos agentes **SNMP**.

4.2 Modelo de organização da informação

Um dos principais, senão o mais importante, objetivo deste sistema é abordar a domótica de um modo alargado e ágil, sendo por isso necessário desenhar um modelo de organização da informação independente da tecnologia utilizada pelos dispositivos e que seja robusta o suficiente de forma a responder às várias especificidades de cada um. Assim, foi definido o modelo representado na figura 4, com a organização de toda a informação agrupada em tabelas relacionais para uma subsequente especificação na **MIB** a implementar nos agentes **SNMP**.

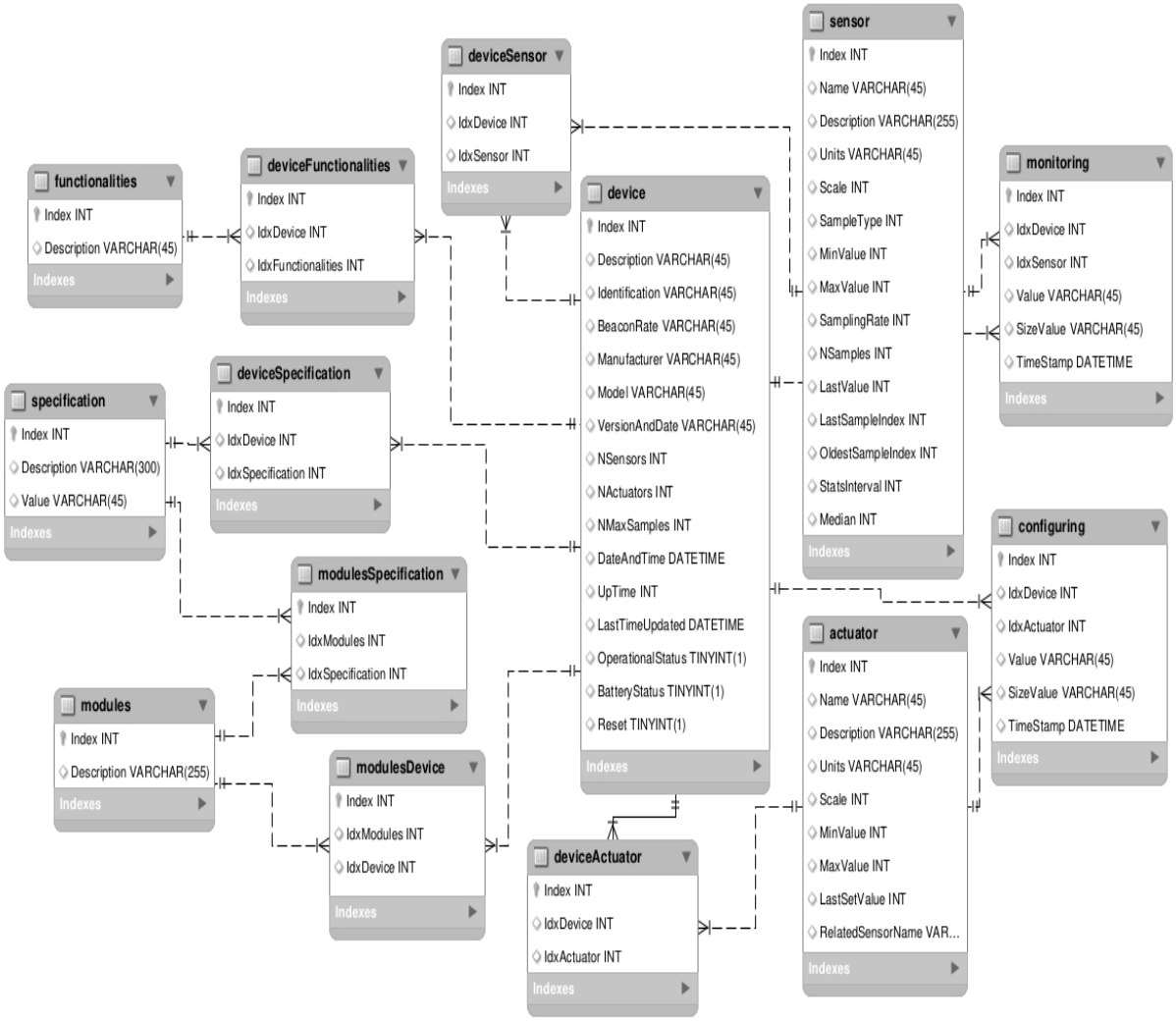


Figura 4: Esquema relacional das tabelas da MIB

A tabela *specification* permite guardar todas as especificações genéricas que podem ser associadas a cada dispositivo ou a cada módulo. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Description* - campo que contém a descrição da especificação;
- *Value* - campo que contém o valor associado à especificação.

A tabela *functionalities* permite identificar todas as funcionalidades que os dispositivos podem disponibilizar. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Description*, campo que contém a descrição das funcionalidades possíveis (por exemplo, medir temperatura ou ligar um dispositivo).

A tabela *modules* permite guardar a informação sobre todos os módulos geridos pelo sistema. Esta tabela é constituída pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Description* - campo que contém a descrição do módulo.

A tabela *modulesDevice* permite estabelecer uma relação entre dispositivos e módulos onde cada linha/elemento indica que um certo dispositivo pertence a um certo módulo. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *IdxModules* - campo que contém o índice do módulo sobre o qual se associa a um dispositivo;
- *IdxDevice* - campo que contém o índice do dispositivo que se está a associar a um módulo.

A tabela *modulesSpecification* permite agrupar a informação sobre as especificações de um módulo. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *IdxModules* - campo que contém o índice do módulo ao qual se associa uma especificação;
- *IdxSpecification* - campo que contém o índice da especificação a que se está a associar um módulo.

A tabela *deviceFunctionalities* contém a informação sobre uma relação entre um dispositivo e uma funcionalidade. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *IdxDevice* - campo que contém o índice do dispositivo a que se está a associar uma funcionalidade;
- *IdxFunctionality* - campo que contém o índice da funcionalidade ao qual se associa um dispositivo.

A tabela *deviceSpecification* contém a informação sobre uma relação entre um dispositivo e uma especificação. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *IdxDevice*, campo que contém o índice do dispositivo que se está a associar um especificação;
- *IdxSpecification*, campo que contém o índice da especificação sobre o qual se associa um dispositivo.

A tabela *device* contém a informação sobre os dispositivos geridos. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Description* - campo que contém uma descrição do dispositivo;
- *Identification* - campo que contém a identificação do dispositivo, podendo ser um nome ou um endereço associado ao dispositivo de forma a tornar possível identificar o diapositivo de forma unívoca no sistema;
- *BeaconRate* - campo que contém um valor que representa a taxa de frequência da transmissão de uma mensagem que funciona como *Beacon*;
- *Manufacturer* - campo que contém a informação do fabricante do dispositivo;
- *Model* - campo que contém a informação do modelo do dispositivo;
- *VersionAndDate* - campo que contém informação sobre a versão e a data de implementação/construção/fabrico do dispositivo;
- *NSensors* - campo que contém um valor que representa o número de sensores associados ao dispositivo;
- *NActuators* - campo que contém um valor que representa o número de atuadores associados ao dispositivo;

- *NMaxSamples* - campo que contém um valor que representa o número máximo de amostras que cada sensor associado ao dispositivo pode registar;
- *DateAndTime* - campo que contém a informação sobre a data e hora no dispositivo;
- *UpTime* - campo que contém um valor que representa a quantidade do tempo em que o dispositivo esteve disponível;
- *LastTimeUpdated* - campo que contém informação sobre a data e hora da última atualização de algum campo do dispositivo;
- *OperationalStatus* - campo que contém um valor que representa o estado operacional atual do dispositivo;
- *BatteryStatus* - campo que contém um valor que representa o estado atual da bateria do dispositivo caso o dispositivo seja alimentado dessa forma;
- *Reset* - campo que contém um valor que representa diversas formas de efetuar um *reset* ao dispositivo.

A tabela *sensor* contém a informação sobre um sensor presente num dos dispositivos geridos pelo sistema. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Name* - a identificação do sensor;
- *Description* - campo que contém uma descrição do sensor;
- *Units* - campo que contém a informação da unidade de medida do sensor;
- *Scale* - campo que contém um valor que corresponde à escala dos valores lidos pelo sensor;
- *SampleType* - campo que contém um valor que representa o tipo de amostras que o sensor lê;
- *MinValue* - campo que contém um valor que representa o valor mínimo capaz de ser lido pelo sensor;
- *MaxValue* - campo que contém um valor que representa o valor máximo capaz de ser lido pelo sensor;

- *SamplingRate* - campo que contém um valor que representa a taxa de frequência do registo das amostras pelo sensor;
- *NSamples* - campo que contém o valor do número de amostras realizadas pelo sensor desde o seu início até ao momento;
- *LastValue* - campo que contém o último valor lido pelo sensor;
- *LastSampleIndex* - campo que contém o valor que representa o índice da última amostra (a mais recente) registada;
- *OldestSampleIndex* - campo que contém o valor que representa o índice da amostra mais antiga ainda registada;
- *StatsInterval* - campo que contém o valor que representa o intervalo de tempo sobre o qual o sensor atualiza os cálculos de dados estatísticos sobre as medições efetuadas;
- *Median* - campo que contém o valor que representa um dado estatístico de mediana sobre as amostras registadas até ao momento.

A tabela *actuator* contém a informação sobre os atuadores presentes nos dispositivos geridos pelo sistema. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Name* - a identificação do atuador;
- *Description* - campo que contém uma descrição do sensor ou atuador;
- *Units* - campo que contém a informação da unidade de medida do sensor ou atuador;
- *Scale* - campo que contém o valor que corresponde à escala dos valores permitidos no atuador;
- *MinValue* - campo que contém o valor que representa o valor mínimo capaz de ser entendido pelo atuador;
- *MaxValue* - campo que contém o valor que representa o valor máximo capaz de ser entendido pelo atuador;
- *LastSetValue* - campo que contém o último valor introduzido no atuador;

- *RelatedSensorName* - campo que contém, caso exista, os nomes de sensores que estejam relacionados com o atuador.

A tabela *deviceSensor* contém as informações que definem relações entre dispositivos e sensores. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *IdxDevice* - campo que contém o índice do dispositivo que se está a associar a um sensor ou atuador;
- *IdxSensor* - campo que contém o índice do sensor ao qual se associa um dispositivo.

A tabela *deviceActuator* permite definir as relações entre dispositivos e um atuadores. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *IdxDevice* - campo que contém o índice do dispositivo que se está a associar a um atuador;
- *IdxActuator* - campo que contém o índice do atuador ao qual se associa um dispositivo.

A tabela *monitoring* contém a informação sobre os processos de monitorização em curso. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *IdxDevice* - campo que contém o índice do dispositivo onde o sensor que se está a monitorizar está implementado;
- *IdxSensor* - campo que contém o índice do sensor que se está a monitorizar;
- *Value* - campo que contém o valor da amostra registada;
- *SizeValue* - campo que contém o tamanho do valor lido;
- *TimeStamp* - campo que contém a data e a hora da criação de uma nova entrada de monitorização que ocorre quando é recebida informação de uma amostra efetuada por qualquer sensor.

A tabela *configuring* contém a informação sobre processos de configuração dos atuadores em curso. Os elementos desta tabela são constituídos pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *IdxDevice* - campo que contém o índice do dispositivo que dá acesso ao atuador que se pretende configurar;
- *IdxActuator* - campo que contém o índice do atuador que se está a a aceder;
- *Value* - campo que contém o valor inserido que se está a tentar configurar ou re-configurar;
- *SizeValue* - campo que contém o tamanho do valor a usar no processo de configuração;
- *TimeStamp* - campo que contém a data e a hora da criação do processo de configuração num atuador.

A instanciação deste modelo de informação foi feita numa base de objetos de gestão **SNMP**, a **Domotics MIB**, cuja especificação em **SMI** pode ser encontrada no Anexo A.2. A instrumentação desta **MIB** foi implementado no protótipo de agente domótico **SNMP** referido na secção 5.4 .

4.3 Light SNMP

A implementação da instrumentação da **DOMOTICS MIB** obriga à comunicação entre os dispositivos domóticos e o módulo do agente **SNMP**. Conforme já foi referido no capítulo anterior, existem variadas tecnologias de acesso a dispositivos domóticos, muitos deles, proprietários, o que dificulta imenso a implementação de instrumentação desta **MIB** no agente **SNMP**. A solução encontrada foi desenvolver um novo protocolo aberto, baseado no **SNMP** e designado **Light SNMP** (**L-SNMP**). Este protocolo é uma versão mais simples e com um nível funcional mais baixo, por isso com menor exigência computacional e comunicacional do que a versão mais recente do **SNMP**, a versão 3 (**SNMPv3**).

O **L-SNMP** pode assim ser integrado nos dispositivos domóticos como uma alternativa universal e aberta, facilitando a implementação de agentes **SNMP** tradicionais que implementem a **DOMOTICS MIB**. O protocolo pode ser encapsulado em **UDP**, ou diretamente sobre **IP** ou até sobre outros protocolos de rede local ou comunicação direta, como o **Bluetooth**.

4.3.1 Arquitetura L-SNMP

A arquitetura do modelo é relativamente simples, um dispositivo atua de forma semelhante a um servidor e o *Agente SNMP* como um cliente, devendo poder comunicar com vários dispositivos. Por sua vez os dis-

positivos podem comunicar com os seus sensores ou atuadores utilizando as tecnologias de comunicação atuais ou através da integração nativa dos sensores e atuadores nos módulos **L-SNMP**.

O protocolo desenvolvido utiliza o paradigma aplicacional de servidor/cliente e a comunicação é atómica, assíncrona, assimétrica e não orientada à conexão, tal como o **SNMP**.

Atendendo às características mencionadas pôde-se estabelecer que:

- Este protocolo deve ter os seus **PDU**s encapsulados em datagramas **UDP**, ainda que protocolos de rede ou de nível 2 possam ser usados;
- Se for usado o encapsulamento em **UDP**, cada dispositivo deverá implementar um agente **L-SNMP**, identificador por um endereço **IP** e uma porta **UDP**;
- Os sensores e os atuadores não podem ser acedidos diretamente por *Agentes SNMP* pelo que a sua identificação é apenas local, tendo como contexto apenas o próprio dispositivo;
- Cada dispositivo, para além de permitir o controlo dos seus sensores e atuadores, também deve permitir o seu próprio controlo, podendo assim ser manipulado como qualquer sensor ou atuador sem necessidade de comandos especiais;
- O protocolo é assíncrono não ficando estabelecidos quaisquer temporizadores de comunicação nem obrigações de interação/confirmação;
- Com o intuito de manter a simplicidade, não devem existir mensagens específicas de erro, apenas um campo de informação de erro em todas as mensagens de resposta e que deve referir-se apenas ao contexto da mensagem do pedido a que a resposta diz respeito;
- Pode usar-se a mesma estratégia de cada sensor, atuador e o próprio dispositivo serem monitorizados ou configurados através de valores de instâncias de objetos simples (uma simplificação do modelo de informação utilizado no **SNMP**);
- Apenas os dispositivos controlam quais os sensores e atuadores estão disponíveis aos gestores/-clientes/agentes **SNMP** para acesso pelo protocolo de comunicação;
- O protocolo é assimétrico com apenas duas primitivas (set-request e get-request) para uso específico dos gestores **L-SNMP** (normalamente um agente **SNMP**) e duas primitivas (notification e response) para uso específico dos agentes **L-SNMP**; a primitiva *response* só pode ser usada como resposta a uma primitiva **-request*, além disso, existe a primitiva *notification* que é enviada para

um conjunto de endereços **IP**:porta **UDP** pré-configurado (que pode incluir um endereço de *broadcast* na rede local), a uma frequência pré-configurada e que serve para anunciar as informações principais do dispositivo (para se dar a conhecer para permitir uma auto-configuração inicial do sistema) ou amostras recolhidas pelos sensores com o mínimo de intervenção humana (uma espécie de *beacon*);

- Cada **PDU** inclui um cabeçalho, um identificador de primitiva, uma primitiva e um *checksum*; o identificador de primitiva é um número aleatório gerado pelo gestor **L-SNMP**; o dispositivo deve usar esse identificador para associar a primitiva de resposta à correspondente primitiva **-request* (no caso da primitiva *notification* das informações iniciais do dispositivo, o identificador de primitiva é nulo); O cabeçalho do **PDU** inclui o tamanho total e o tamanho do cabeçalho, a versão do protocolo (um *byte*), um *timestamp* indicando o tempo no dispositivo ou no agente **L-SNMP** em que o **PDU** foi enviado, e, opcionalmente, campos que permitam implementar mecanismos simples de autenticação e confidencialidade; A primitiva inclui o tipo da primitiva, uma lista de identificadores de instâncias de objetos de gestão, uma lista de valores de instâncias de objetos de gestão e uma lista códigos de erro; Caso existam dados de autenticação e confidencialidade, estes serão enviados depois da primitiva. A informação mais detalhada da especificação dos **PDU** é feita na secção ??

4.3.2 Modelo de informação

Tal como foi necessário conceber um modelo de informação a ser implementado no agente **SNMP** sob forma de uma **MIB**, para o *Light SNMP* foi também essencial definir o seu modelo de informação e posterior codificação da **Light MIB** (**L-MIB**). Apresenta-se de seguida o esquema relacional (figura 5) constituído por tabelas que representam os diferentes grupos de objetos de gestão para dispositivos domóticos bem como as suas associações.

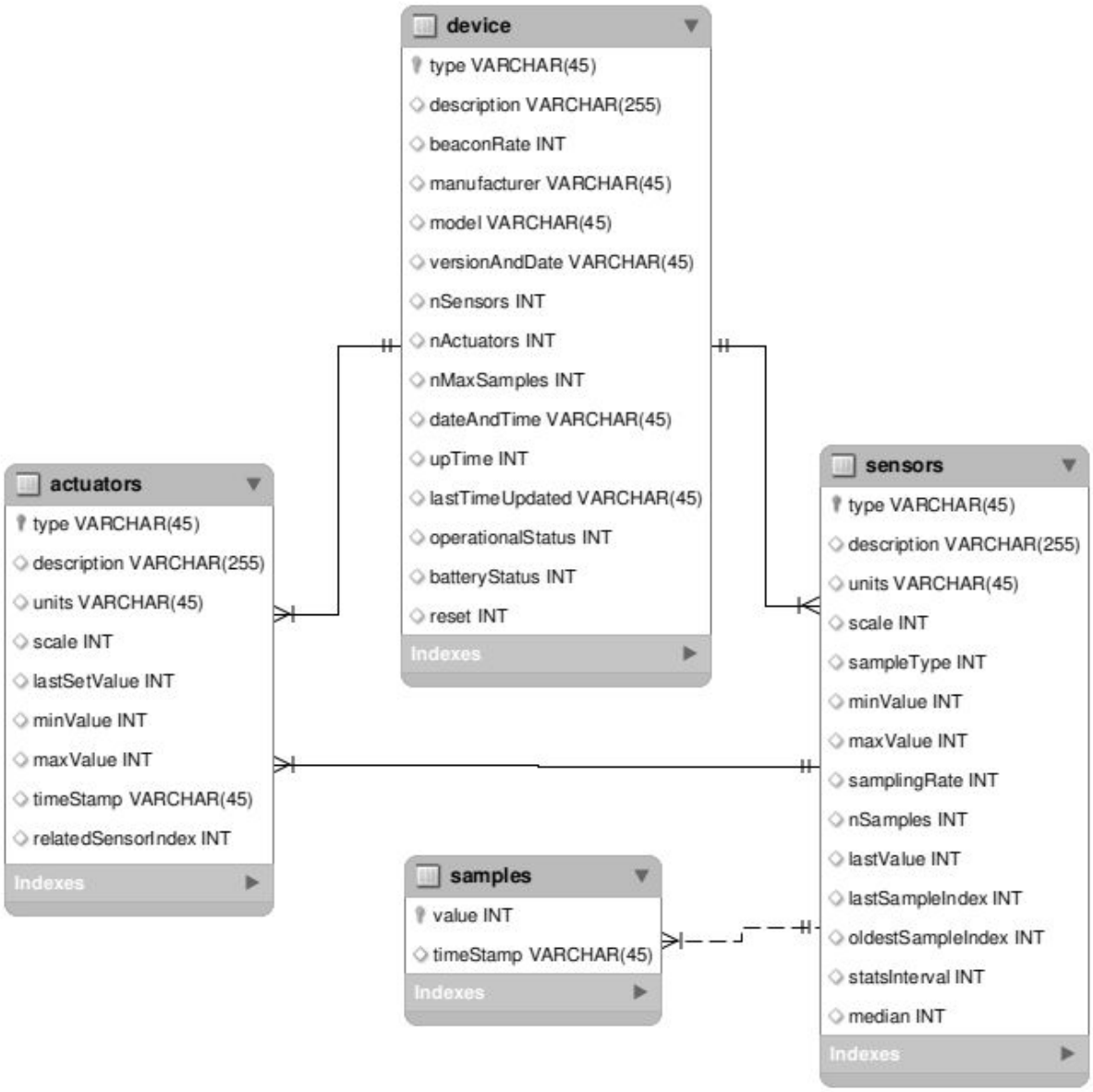


Figura 5: Esquema relacional das tabelas da **DOMOTICS L-MIB**

Antes da explicação individual de cada grupo e seus constituintes é importante realçar a grande semelhança no conteúdo deste esquema com aquele concebido para o **SNMP**, o que torna mais simples a implementação da **MIB** nos agentes **SNMP** e integração da informação recolhida e controlada me vários dispositivos domóticos com módulos **L-SNMP**.

A tabela *device* contém a informação específica do próprio dispositivo e inclui os seguintes campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Type* - tipo do dispositivo (isto permite, inclusive a utilização do **L-SNMP** em dispositivos em contextos diferentes de sistemas domóticos, como, por exemplo, em sistemas veiculares ou de redes

de sensores);

- *Description* - campo que contém uma descrição do dispositivo;
- *BeaconRate* - campo que contém um valor que representa a taxa de frequência da transmissão das mensagens de notificação que funcionam como *Beacon*;
- *Manufacturer* - campo que contém a informação do fabricante do dispositivo;
- *Model* - campo que contém a informação do modelo do dispositivo;
- *VersionAndDate* - campo que contém informação sobre a versão e a sua data de implementação/-fabrico do próprio dispositivo e/ou do seu *firmware* ou do módulo de *software* que implementa o agente **L-SNMP**;
- *NSensors* - campo que contém o valor que representa o número de sensores geridos no dispositivo;
- *NActuators* - campo que contém o valor que representa o número de atuadores geridos no dispositivo;
- *NMaxSamples* - campo que contém o valor que representa o número máximo de amostras que cada sensor associado ao dispositivo pode registar, quando é atingido este número, os valores das amostras mais antigas são substituídas pelas mais recentes;
- *DateAndTime* - campo que contém a informação sobre a data e hora no dispositivo;
- *UpTime* - campo que contém o valor que representa o período de tempo em que o dispositivo esteve disponível desde a sua inicialização;
- *LastTimeUpdated* - campo que contém informação sobre a data e hora da última atualização de alguma informação da **L-MIB**;
- *OperationalStatus* - campo que contém o valor que representa o estado operacional atual do dispositivo;
- *BatteryStatus* - campo que contém o valor que representa o estado atual da bateria do dispositivo caso o dispositivo seja alimentado dessa forma;
- *Reset* - campo que contém o valor que implementa as diversas formas de efetuar um *reset* ao dispositivo.

A tabela *sensors* contém informações sobre os sensores presentes no sistema e inclui os seguintes campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Type* - campo que indica a tipo do sensor (temperatura, luminosidade, etc.);
- *Description* - campo que contém uma descrição do sensor;
- *Units* - campo que contém a informação da unidade de medida dos valores das amostras registadas pelo sensor;
- *Scale* - campo que contém o valor que corresponde à escala dos valores lidos pelo sensor;
- *SampleType* - campo que contém o valor que representa o tipo dos valores das amostras que o sensor regista;
- *MinValue* - campo que contém o valor que representa o valor mínimo capaz de ser registado pelo sensor;
- *MaxValue* - campo que contém o valor que representa o valor máximo capaz de ser registado pelo sensor;
- *SamplingRate* - campo que contém o valor que representa a taxa de frequência da amostragem realizada pelo sensor;
- *NSamples* - campo que contém o valor do número de amostras realizadas pelo sensor desde o seu início até ao momento.
- *LastValue* - campo que contém o ultimo valor lido pelo sensor;
- *LastSampleIndex* - campo que contém o valor que representa o índice da última amostra da tabela *samples*;
- *OldestSampleIndex* - campo que contém o valor que representa o índice da amostra mais antiga da tabela *samples*;
- *StatsInterval* - campo que contém o valor que representa o intervalo de tempo em que o dispositivo atualiza os cálculos de dados estatísticos sobre as medições registadas pelo sensor;

- *Median* - campo que contém o valor que representa um dado estatístico de mediana sobre as amostras registadas até ao momento pelo sensor.

A tabela *actuators* contém a informação sobre os atuadores presentes no sistema. Esta informação permite o controlo de equipamentos domóticos (ligar luzes, aumentar temperatura do A/C, abrir um portão, etc.) e inclui os seguintes campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Type* - campo que indica o tipo do atuador;
- *Description* - campo que contém uma descrição do atuador;
- *Units* - campo que identifica a unidade de medida usada pelo atuador;
- *Scale* - campo que contém o valor que corresponde á escala dos valores reconhecidos pelo atuador;
- *LastSetValue* - campo que contém o último valor configurado no atuador;
- *MinValue* - campo que contém o valor que representa o valor mínimo capaz de ser entendido pelo atuador;
- *MaxValue* - campo que contém o valor que representa o valor máximo capaz de ser entendido pelo atuador;
- *TimeStamp* - campo que indica o tempo exato em que o último valor foi configurado no atuador. Indica ano, mês, dia, hora, minuto, segundo.
- *RelatedSensorIndex* - campo que contém, caso exista, o índex de um sensor que está relacionado com o atuador.

A tabela *samples* contém a informação sobre as amostras recolhidas pelos sensores do dispositivo. Esta tabela é constituída pelos campos:

- *Index* - o índice de cada elemento e que funciona como chave da tabela;
- *Value* - campo que contém o valor da amostra registada;
- *TimeStamp* - campo que indica o tempo exato em que o último valor foi inserido no atuador. Indica ano, mês, dia, hora, minuto, segundo.

O modelo especifica a forma como os dados são apresentados no *payload* do protocolo, inseridos no campo da primitiva e, por isso, foi necessário estabelecer algumas regras para a sua declaração formal tal como se sucede para as **MIB** do **SNMP**. A forma de declaração de objetos é baseada nos conceitos da **SMIv2**, sendo mais simplificada. Cada objeto é definido por um nome, pelo seu tipo e o tipo de acesso permitido (caso o objeto em questão seja um dos objetos transmitidos na mensagem de notificação existe um campo que contém a identificação do objeto que indica a taxa de transmissão das notificações), pela sua descrição funcional e o seu identificador.

- **Nome** - Indica o nome do objeto ou de um grupo de objetos (devem ser seguidas as mesmmas recomendações definidas na **SMIv2**).

- **Tipo** - Caso seja um objeto simples é apenas possível ter-se inteiros (valores entre 0 e 256^N, em que N é o número de *bytes* 1, 2, 4 ou 8) e sequências de inteiros. Apenas utilizando inteiros é possível ainda a definição de convenções de tipos funcionais adicionais. Por exemplo, um *short/integer/long/large* é um inteiro com N=1/2/4/8, uma *String* é uma sequência de *shorts* em que o primeiro elemento indica o tipo de codificação da *String*. De salientar ainda que apesar de atualmente serem definidos os tipos convencionais necessários para a elaboração da dissertação, o modelo permite que se vão acrescentando outras definições ao longo do tempo (tal como o **SMI** o permite no **SNMP**). Caso seja a definição de um grupo que esteja presente neste campo o tipo de estrutura utilizada para armazenar a informação é precedido da lista dos nomes dos objetos contidos neste grupo.

- **Acesso** - Indica o tipo de acesso permitido pelo grupo de objetos ou por um único objeto. O acesso indica a forma que é permitida a manipulação a informação do objeto, podendo ser do tipo *read-only* (permitindo apenas a leitura do valor do objeto) ou *read-write* (permitindo não só a leitura como também a alteração desse valor).

- **Notificação** - Lista dos nomes dos objetos cujos valores são incluídos nas mensagens de notificação.

- **Taxa de Notificação** - Campo onde se indica o nome do objeto que contém o valor da taxa de transmissão da mensagem de notificação.

- **Descrição Funcional** - Uma descrição o mais precisa e detalhada possível da funcionalidade (semântica) do objeto e do significado dos seus valores possíveis.

- **Identificador** - Representado por dois inteiros, implementando assim uma hierarquia de dois níveis onde o primeiro nível identifica o sensor, o atuador, o próprio dispositivo ou uma amostra. O segundo nível identifica uma propriedade do sensor, atuador ou dispositivo, como por exemplo, o tipo, o fabricante, as unidades usadas, a frequência de atualização, valor máximo/mínimo, etc.;

Apresenta-se em seguida o formato geral da descrição formal e sintática de um grupo de objetos seguido de um objeto pertencente a esse grupo neste modelo de informação de acordo com as regras mencionadas previamente.

```
# Definição dos grupos
nome_do_grupo_de_objetos OBJECT {
 TYPE tipo_de_estrutura_de_organização_da_informação LISTA DE OBJETOS
    PERTENCENTES AO GRUPO
 ACESS tipo_de_acesso_que_o_grupo_permite
 NOTIFICATION objetos_que_são_incluídos_nas_mensagens_de_notificação
 DESCRIPTION descrição_textual_do_grupo
 IID número_do_identificador_de_instância_do_grupo }

# Definição dos objetos dos grupos
nome_do_grupo_de_objetos.nome_do_objeto_pertencente_ao_grupo OBJECT {
 TYPE tipo_do_objeto TAMANHO_MAXIMO
 ACESS tipo_de_acesso_que_o_objeto_permite
 NOTIFICATION-RATE (Opcional) nome_do_objeto_que_indica_a_taxa_de_
      transmissão_da_mensagem_de_notificação
 DESCRIPTION descrição_do_objeto
 IID número_do_identificador_de_instância_do_grupo.número_do_identificador_de_
      instância_do_objeto_no_contexto_do_grupo}
```

Apresenta-se em seguida o início de especificação dos objetos dum grupo no modelo de informação **L-SNMP** estando a descrição completa presente no Anexo A.3:

```
device OBJECT {
 TYPE List type, description, beaconRate, manufacturer, model, versionAndDate,
 nSensors, nActuators, nMaxSamples, dateAndTime, upTime, lastTimeUpdated,
 operationalStatus, reset
 ACESS read-only
```

```
 NOTIFICATION type, description, nSensores, nAtuadores, upTime, lastTimeUpdated,
 operationalStatus, batteryStatus
 DESCRIPTION "Grupo com uma lista simples de objetos, onde cada objeto representa uma
 caracteristica de um dispositivo"
 IID 1 }

device.type OBJECT {
 TYPE String 45
 ACESS read-only
 NOTIFICATION-RATE beaconRate
 DESCRIPTION "Tipo do dispositivo"
 IID 1.1 }


device.description OBJECT {
 TYPE String 255
 ACESS read-only
 DESCRIPTION "Descricação do dispositivo"
 IID 1.2 }

[...]
```

Como se pode observar no exemplo apresentado, o objeto *device* contém os campos (ou cláusulas) *TYPE*, *ACESS*, *NOTIFICATION*, *DESCRIPTION* e *IID*. O tipo deste objeto mais genérico é uma lista de todas as propriedades dos dispositivos que podem ser representadas no modelo de informação. Tal como descrito anteriormente, o *ACESS* define a forma como é permitida a manipulação da informação do objeto sendo, no caso, apenas possível a sua leitura (ou monitorização). O campo *NOTIFICATION* indica os objetos do dispositivo que serão enviados na primitiva *notification* que serve de *beacon* para o dispositivo ser descoberto pelos clientes/gestores/controladores. A *DESCRIPTION* indica uma descrição do objeto e *IID* é a identificação do objeto (conceito simplificado dois **OID** do **SNMP**).

O segundo objeto *device.type* é um objeto do tipo *String* com um tamanho máximo de 45 caracteres e um *ACESS* apenas passível de leitura tal como o anterior. Relativamente ao campo *NOTIFICATION-RATE*, está presente o nome do objeto que define a frequência com que será realizado o envio de da primitiva *notification*. Os campos *DESCRIPTION* e *IID* indicam uma descrição do objeto e a identificação do mesmo.

Por fim, o terceiro objeto presente no excerto é semelhante ao anterior com exceção do tamanho máximo da

String permitido, que neste caso, é de 255 caracteres, e o seu valor nunca pode ser enviado nas notificações. De notar que o modelo permite o envio de notificações (mensagens não solicitadas) com o valor de qualquer objeto que esteja associado a uma cláusula de *NOTIFICATION*. Os valores dos objetos duma lista ou duma tabela, que sejam incluídos na mesma cláusula de *NOTIFICATION*, devem ser enviados na mesma mensagem de notificação (e devem ter o mesmo valor de *NOTIFICATION-RATE*).

Tendo em conta as características do protocolo proposto, foi necessário conceber um modelo de informação com um conjunto de objetos simples, mas capazes de abstrair as funcionalidades dos dispositivos domóticos que serão depois necessárias à implementação, nos agentes **Simple Networking Management Protocol** tradicionais, da instrumentação da *Domotics MIB*.

Enquanto a *Domotics L-MIB* permite uma gestão simplificada dum único dispositivo por agente, a *Domotics MIB*, permite a gestão integrada de todos os dispositivos dum domínio domótico, geralmente um edifício inteiro ou um conjunto de edifícios.

4.3.3 Organização e Identificação dos Objetos de gestão

Neste modelo de informação devem usar-se três tipos de estruturas de dados nas **L-MIB**: listas, tabelas simples e matrizes indexadas. Em termos organizativos, os objetos não estão integrados em grupos, como nas **MIB** do **SNMP**, simplificando a sua identificação e manipulação.

No caso específico da *Domotics L-MIB*, esta inclui uma lista de objetos (*device*), duas tabelas simples (sensores e atuadores) e uma matriz indexada (*samples*). Utilizar-se-ão estes objetos da *Domotics L-MIB* para explicações mais detalhadas sobre os tipos de objetos, a usa organização e formas de identificação que são possíveis nas **L-MIB**.

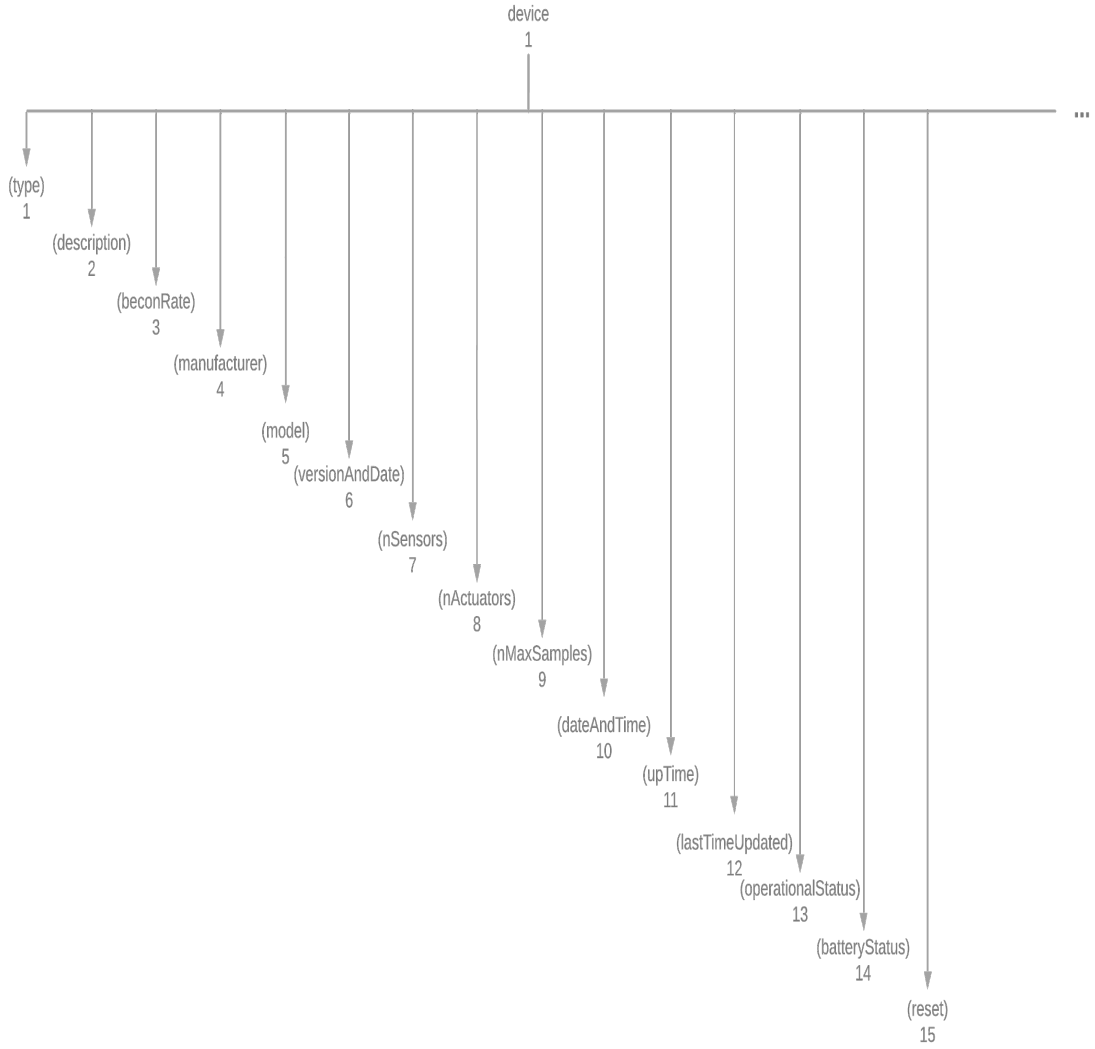


Figura 6: Definição da lista *device* na *Domotics L-MIB*

Uma lista, tal como o objeto (*device*) da *Domotics L-MIB* da Fig.6, é composta por 1 a N objetos simples; cada objeto é identificado por uma chave (inteiro maior que zero); um objeto simples contém uma ou mais instâncias (todas do mesmo tipo) (a), que são identificadas por um índice (inteiro de numeração contínua maior que zero) (b) ou um limite definido por dois índices (c):

- **LIST.OBJ** (a)
  - OBJ=0, refere-se ao número de objetos da lista.
- **LIST.OBJ.[INDEX]** (b)
  - OBJ > 0;
  - INDEX=0, número de instâncias do objeto identificado pela chave OBJ;

- INDEX>0, valor da instância (identificada por INDEX) do objeto identificado por OBJ.

- **LIST.OBJ.[INDEX1.INDEX2]** (c)

  - OBJ > 0;
  - 0<INDEX1<INDEX2, valores das instâncias com índices entre INDEX1 e INDEX2;
  - INDEX1=INDEX2=0, refere-se aos valores de todas as instâncias.

Em seguida enumeram-se alguns exemplos de possíveis do uso dos identificadores para a lista (*device*) da *Domotics L-MIB*:

- (device).0, indica o número de instâncias do objeto na lista identificada por (*device*);
- (device).(model) ou (device).(model).[1], indica a 1ª instância do objeto (*model*) da lista (*device*);
- (device).(model).[0], indica o número da instâncias incluídas no objeto (*model*) de (*device*);
- (device).(nSensors).[1], indica o número de sensores disponíveis/geríveis pelo dispositivo.
- (device).(versionAndDate).[1], indica a data de fabrico/instalação do *software* no dispositivo.

As tabelas simples permitem gerir o mesmo tipo de informação por cada elemento (sensores ou atuadores, no caso da *Domotics L-MIB*).

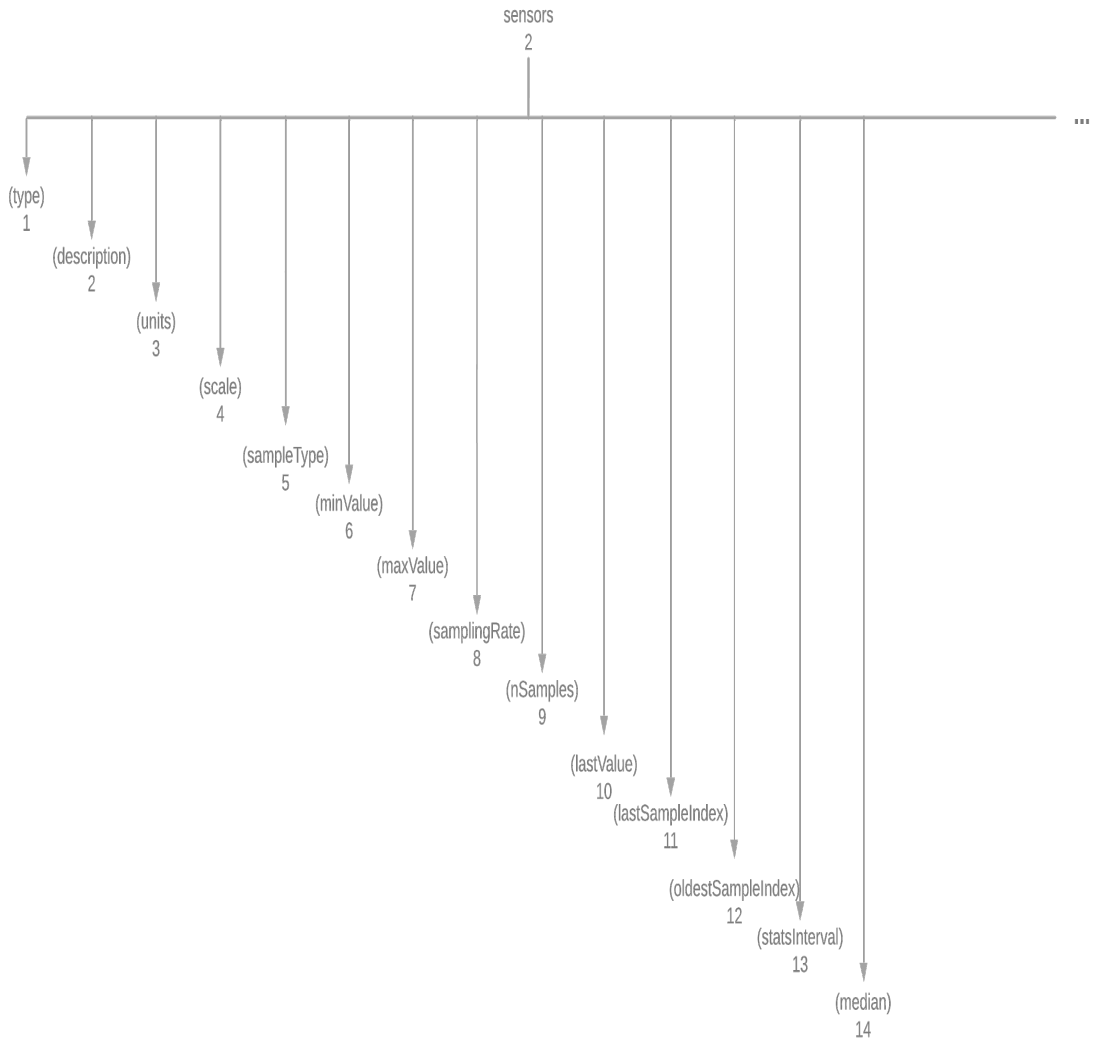


Figura 7: Tabela simples de sensores da *Domotics L-MIB*

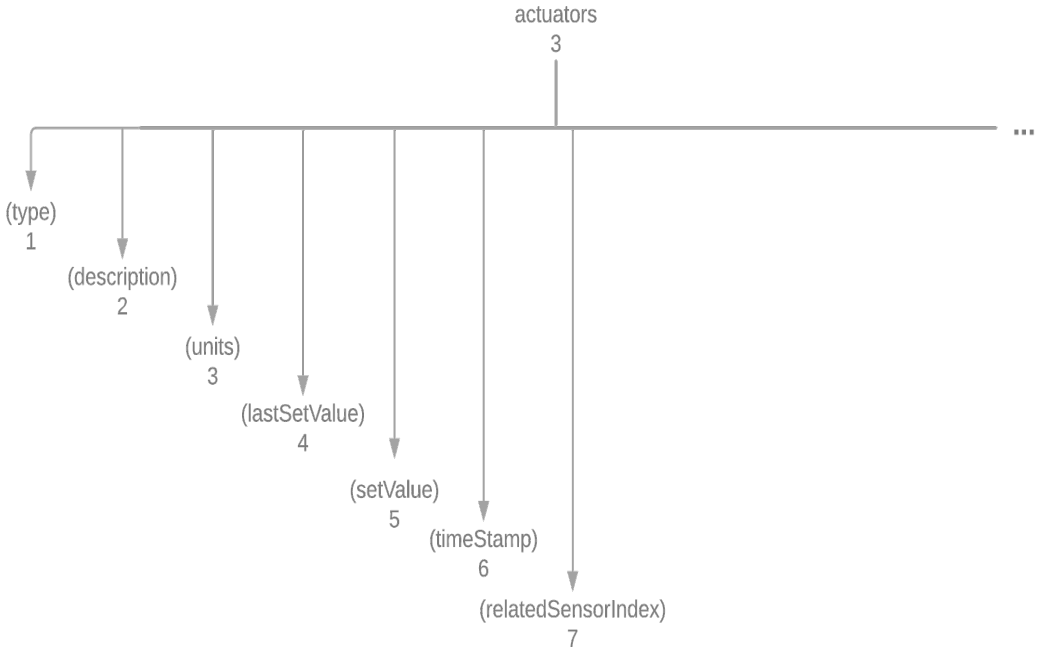


Figura 8: Tabela simples de atuadores *Domotics L-MIB*

Uma tabela, tal como os objetos sensores e atuadores da *Domotics L-MIB* (Fig.7 e Fig.9) tem N Colunas x M Linhas; cada linha equivale a um registo de N objetos; todas as linhas são iguais, i.e., são constituídas pelos mesmos tipos de objetos; todos os objetos têm o mesmo número de M instâncias; uma tabela simples é uma lista simples em que as colunas são os objetos da lista, todos os objetos têm o mesmo número de instâncias e todas as instâncias referidas pelo mesmo índice estão diretamente relacionadas. Os formatos permitidos para a identificação de objetos duma tabela são os seguintes:

- **TABLE.COLUMN**
  - COLUMN=0, refere-se ao número de colunas da tabela.
- **TABLE.COLUMN.[INDEX]**
  - COLUMN>0, indica uma determinada coluna da tabela.
  - INDEX=0, refere-se ao número de instâncias da coluna identificada por COLUMN e deve ser igual para todas as colunas da tabela;
  - INDEX>0, refere-se ao valor da instância identificada por INDEX (da coluna COLUMN).
- **TABLE.COLUMN.[INDEX1.INDEX2]**
  - COLUMN>0, indica uma determinada coluna identificado pelo número COLUMN;

- 0<INDEX1<INDEX2, referem-se aos valores das instâncias identificadas pelo limite definido por INDEX1 e INDEX2;
- INDEX1=INDEX2=0, referem-se aos valores de todas as instâncias da coluna COLUMN

Alguns exemplos do uso de identificadores válidos para as tabelas da *Domotics L-MIB*:

- (sensors).0, indica o número de campos/objetos por cada elemento/linha da tabela (sensores) da *Domotics L-MIB*;
- (sensors).(type) ou (sensors).(type).[1], refere-se à instância do objeto *type* na 1ª linha da tabela sensores da *Domotics L-MIB*;
- (sensors).(type).[0], indica o número de instâncias (linhas) da tabela sensores;
- (sensors).(nSamples).[1], identifica o valor da instância da primeira linha (ou 1ª instância) do objeto *nSamples* da tabela sensores;
- (actuators).0, indica o número de campos/objetos por cada elemento/linha da tabela (atuadores) da *Domotics L-MIB*;
- (actuators).(type) ou (actuators).(type).[1], refere-se à instância do objeto *type* na 1ª linha da tabela atuadores da *Domotics L-MIB*;
- (actuators).(type).[0], indica o número de instâncias (linhas) da tabela atuadores;
- (actuators).(lastSetValue).[1], identifica o valor da instância da primeira linha (ou 1ª instância) do objeto *lastSetValue* da tabela atuadores;

Finalmente, para além das listas e tabelas, o modelo de informação do **L-SNMP** inclui a definição de matrizes indexadas. Estas matrizes permitem associar indiretamente uma tabela por cada linha da tabela indexante. Ou seja, a matriz representa um conjunto de tabelas de dados em que cada tabela está associada (ou diz respeito) a uma linha/entrada da tabela indexante. De notar que as listas, tabelas e matrizes do modelo **L-SNMP** definem sempre um índice (ou chave) implícito que é um valor inteiro identificando uma instância duma lista numa linha duma tabela, ou uma tabela duma matriz indexada.

Enquanto a definição conceptual duma matriz indexada seja descrita como uma tabela de tabelas de dados, na prática a sua implementação é simples através duma única tabela em que todas as entradas/linhas referentes a uma tabela de dados têm o mesmo valor numa coluna/objeto definido implicitamente.

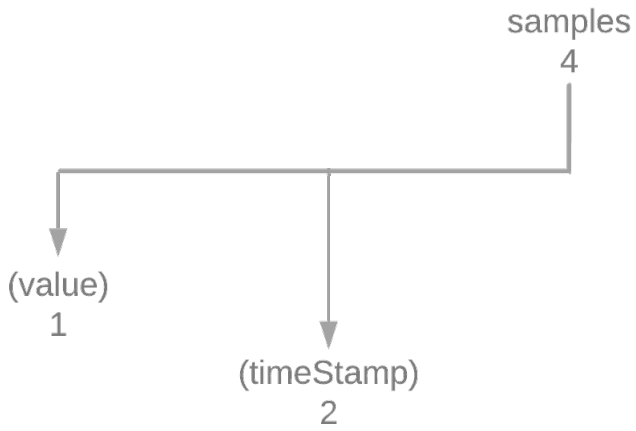


Figura 9: A matriz de tabelas de amostras é indexada à tabela de sensores na *Domotics L-MIB*

O formato possível para identificar objetos duma matriz indexada são apresentados em seguida:

- **ARRAY.<KEY>**
  - KEY=0, refere-se ao número de elementos do array, i.e., ao número de tabelas do array.
- **ARRAY.COLUMN.<KEY>**
  - COLUMN=0, refere-se ao número de colunas da tabela identificada por KEY (linhas/tabelas em que o valor implícito da chave/indíce é igual a KEY).
- **ARRAY.COLUMN.[INDEX].<KEY>**
  - COLUMN > 0;
  - INDEX=0, refere-se ao número de instâncias da coluna identificada por COLUMN e deve ser igual para todas as colunas da tabela identificada por KEY;
  - INDEX>0, refere-se ao valor da instância identificada por INDEX (da coluna COLUMN da tabela identificado por KEY).
- **ARRAY.COLUMN.[INDEX1.INDEX2].<KEY>**
  - COLUMN>0, refere-se ao objeto/coluna identificado pelo número COLUMN;
  - 0<INDEX1<INDEX2, refere-se aos valores das instâncias identificadas pelo limite definido por INDEX1 e INDEX2 (da coluna COLUMN da tabela identificado por KEY);

– INDEX1=INDEX2=0, refere-se aos valores de todas as instâncias da coluna COLUMN (da tabela identificado por KEY).

No caso da matriz indexada *samples* da *Domotics L-MIB*, a tabela indexante é a tabela *sensors*. Portanto a matriz *samples* pode ser implementada como uma tabela em que existe mais uma coluna implícita que identifica o sensor a que as amostras dizem respeito (ver definição completa no Anexo A.3). Por exemplo, para obter o valor da 1ª amostra registada na matriz *samples* obtida pelo sensor i, usamos o identificador (samples).(value).[1].<i> .

4.3.4 Regras de codificação e PDU das mensagens L-SNMP

Com o intuito de garantir uniformização e compactação na transmissão de mensagens entre as entidades que utilizam o protocolo, definiu-se um modelo de codificação da primitiva presente no *payload* das mensagens.

Este modelo de codificação tem regras inspiradas no modelo **Basic Encoding Rules** (**BER**) [39] já existente e define uma forma específica para a conversão de informação em forma binária.

Tal como no **BER**, este modelo segue a estrutura (*Type*, *Length* e *Value*) e permite a codificação dos seguintes tipos de valores: *Integer*, *Opaque*, *String*, *Date* e *IID** .

** Istance Identification* = Object [.Indexes][.Key]

Note-se que só é possível incluir nos **PDU** valores de instâncias de objetos simples, não sendo possível, tal como no **SNMP**, o envio de valores de instâncias de objetos não escalares, como tabelas ou listas.

Para o *Type* reservam-se três *bits* que seguem a seguinte codificação:

- *Integer* - 001
- *Opaque* - 010
- *String* - 011
- *Date* - 100
- *IID* - 101

No caso da *Length* reservam-se cinco *bits* e o seu valor representa o comprimento do valor podendo ser entre um e trinta e dois *bytes*.

Finalmente a codificação do *Value* depende do seu tipo:

- *Integer* - Ocupa entre um a oito *bytes* e no caso do *bit* mais significativo ser 1 indica um inteiro negativo;
- *Opaque* - *Array* de *bytes*, onde a semântica do valor é dependente do objeto representado.
- *String* - *Array* de *bytes* tal como o *Opaque*, mas o *array* é uma sequência de caracteres com intuito de serem legíveis por humanos. A codificação deve ser feita utilizando *UTF-8*;

- *Date* - Três *bytes* para data (ano + mês + dia), um *byte* para a identificação do fuso horário e mais quatro *bytes* para a hora (hora + minutos + segundos + milissegundos), sendo no total oito *bytes*.

- *IID* - *Size* (oito *bits*) + sequência identificadores de objetos (Um a oito *bytes*, onde cada *byte* representa um nível) + *Índices* (zero, um, dois ou quatro *bytes*) + *Key* (zero, um ou dois *bytes*). Os primeiros três *bits* do tamanho indicam o número de níveis no identificador de objetos (entre um e oito), os próximos dois *bits* indicam o número de índices, o sexto *bit* indica quantos *bytes* os indexes ocupam (0 - um *byte* cada, 1 - dois *bytes* cada), o sétimo *bit* indica se existe um valor para a chave e o oitavo *bit* indica quantos *bytes* a chave ocupa (0 - um *byte*, 1 - dois *bytes*).

Tendo em conta as regras de codificação definidas apresenta-se na figura 10 o esquema do formato do **PDU** criado que irá ser utilizado para transmitir informação entre dispositivos e agentes que deverá ser encapsulado em **UDP**.

Offset	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Total length																Header length															
4	32	Version																Payload length															
8	64	Timestamp																															
12	96																																
16	128	Security setup																Security data length															
20	160	Request Identifier																															
24	192																																
28	224	Checksum																															
32	256	Primitive																															
36	288																																
40	320																																
44	352																																
48	384																																
52	416																																
56	448	Security data (optional)																															

Figura 10: Esquema do formato do **PDU** das mensagens do **L-SNMP**

O protocolo define um tamanho máximo de 512 *bytes* para o **PDU**. Isto limita as probabilidades de fragmentação **IP** e facilita o encapsulamento seguro e uso de várias tecnologias de nível 2.

Os campos do **PDU** são:

- Total length - Campo que indica o tamanho total do **PDU** em *bytes*, ocupando 16 bits e é utilizada a conversão direta do valor em bits;

- Header length - Campo que representa o tamanho do cabeçalho do **PDU** em *bytes*, ocupando 16 bits e é utilizada a conversão direta do valor em bits;

- Version - Campo que representa a versão do protocolo, ocupando 16 bits e é utilizada a conversão direta do valor em bits. O único valor atualmente permitido é o valor 1;

- Payload length - Campo que representa o tamanho do *payload* do **PDU** em *bytes*, ocupando 16 bits e é utilizada a conversão direta do valor em bits;
- Timestamp - Campo que representa a data e hora do envio do **PDU**, ocupa 8 *bytes* e é utilizada a regra de codificação para o tipo *Date* mencionado previamente.
- Security setup - Campo que identifica o tipo de segurança do protocolo. Se não for utilizado qualquer mecanismo de segurança o valor deve ser zero;
- Security data length - Campo que representa o tamanho dos dados de confidencialidade em *bytes*, ocupando 16 bits e é utilizada a conversão direta do valor em bits. Se não for utilizado qualquer mecanismo de segurança o valor deve ser zero;
- Request identifier - Campo que representa um identificador gerado para identificar um pedido univocamente, ocupando 16 bits e é utilizada a conversão direta do valor em bits;
- Checksum - Campo que representa o valor do *checksum* gerado para a posterior deteção de erros na transmissão dos dados, ocupando 8 *bytes*;
- Primitive - Campo que representa os dados da primitiva **L-SNMP**, ocupando valor variável e utilizando as regras de codificação mencionadas na secção 4.3.1;
- Security data - Campo opcional que contém informações necessárias á implementação dos mecanismos de segurança. As regras de codificação dependem dos mecanismos usados.

Tal como mencionado na secção 4.3.1 a primitiva é constituída por quatro secções diferentes:o tipo da primitiva, a lista de identificadores, a lista de valores e a lista de erros. Assim apresenta-se a tabela de correspondência entre os valores dos quatro tipos de primitivas e o seu significado.

Value	*Bits*	*PrimitiveType*
0	00	notification
1	01	response
2	10	get-request
3	11	set-request

Tabela 1: Tabela de correspondência entre valores e significado de tipos de primitivas

Os restantes campos da primitiva são compostos por listas de valores que podem ser vazios ou compostos por um ou mais elementos. Com o objetivo de demonstrar como a primitiva é codificada apresenta-se um exemplo

na tabela 2 em que, na lista de identificadores, estão presentes os identificadores do tipo e da descrição de um dispositivo (**IID** 1.1 e 1.2 respetivamente) e na lista de valores incluem-se os respetivos valores associados a cada identificador (Tabela 3). De salientar que, neste caso, assume-se a ausência de qualquer erro na operação.

Type	*Length*	*Value*	*IID*
101	00000011	01000000 00000001 00000001	1.1
101	00000011	01000000 00000001 00000010	1.2

Tabela 2: Tabela exemplificativa duma codificação de lista de identificadores

4.3.5 Exemplos de interações

Nesta secção serão exemplificados os tipos de interações possíveis entre um dispositivo (agente **L-SNMP**) e um gestor (agente **SNMP**). Por hipótese considera-se que o dispositivo é um aparelho de **Ar Condicionado** com dois sensores(temperatura e humidade) e dois atuadores (temperatura e estado).

Setup inicial

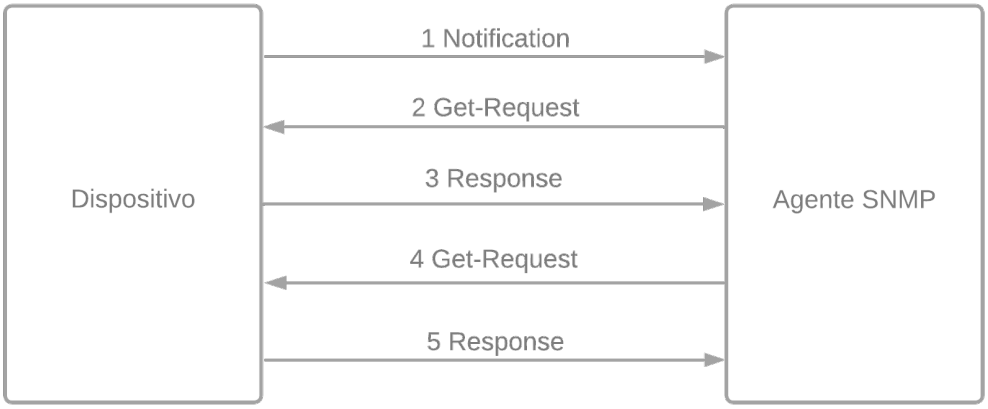


Figura 11: Esquema de comunicação entre Dispositivo e Agente SNMP do setup inicial

Inicialmente, após o dispositivo se tornar ativo, o próprio envia uma mensagem do tipo *notification* (com os objetos correspondentes da lista *device*) que funciona como uma espécie de *beacon* para se dar a conhecer ao agente (1).

Type	*Length*	*Value (hex)*	*Value (binary)*	*Valor*
011	00000011	6c 75 7a	01101100 01110101 01111010	”luz”
011	00010010	64 65 73 63 72 69 c3 a7 c3 a3 6f 20 64 61 20 6c 75 7a	01100100 01100101 01110011 01100011 01110010 01101001 11000011 10100111 11000011 10100011 01101111 00100000 01100100 01100001 00100000 01101100 01110101 01111010	”descrição da luz”

Tabela 3: Tabela demonstrativa da codificação de lista de valores

Na prática este ”*beacon*”deve ser enviado continuamente, mesmo depois do *setup* inicial.

1 Dispositivo -> Agente SNMP (mensagem não solicitada)

- *PrimitiveType -> Notification*
- *IdentifiersList ->* {(device).(type), (device).(description), (device).(nSensors), (device).(nActuators), (device).(upTime), (device).(lastTimeUpdated), (device).(operationalStatus), (device).(batteryStatus)}

- *ValuesList* -> {“arCondicionado”, “Ar condicionado do escritório A1”, 2, 2, 100, “10/10/2022 14:04:35”, 1, 0}
- *ErrorsList* -> { }

O agente **SNMP**, após receber esta mensagem com este tipo de primitiva efetua o processamento dos dadose faz o *setup* inicial na instrumentação da sua **MIB** e despoleta o envio de uma mensagem para o dispositivo com vista à obtenção de mais informações sobre o mesmo.

2 Agente SNMP -> Dispositivo

- *PrimitiveType* -> *Get-Request*
- *IdentifiersList* -> {(device).(manufacturer), (device).(model), (device).(beaconRate), (device).(versionAndDate), (device).(nMaxSamples)}
- *ValuesList* -> { }
- *ErrorsList* -> { }

O agente **SNMP**, efetua um pedido de *get-request* (2) para conhecer mais informações sobre o dispositivo. Envia ainda outro pedido *get-request* para a obtenção de informações de todos os sensores e atuadores associados ao dispositivo (4), a que o dispositivo responde com nova primitiva *response* (5).

3 Dispositivo -> Agente SNMP

- *PrimitiveType* -> *Response*
- *IdentifiersList* -> {(device).(manufacturer), (device).(model), (device).(beaconRate), (device).(versionAndDate), (device).(nMaxSamples)}
- *ValuesList* -> {“ Samsung”, “E56”, “v2 ago.2022”, 500}
- *ErrorsList* -> { }

Após a receção do primeiro *get-request* (2) o dispositivo envia uma mensagem com um tipo de primitiva *response* com todas as informações pedidas, incluindo as suas especificações.

4 Agente SNMP -> Dispositivo

- *PrimitiveType* -> *Get-Request*

- *IdentifiersList ->* {sensors).(type).[0.0], (sensors).(description).[0.0], (sensors).(units).[0.0], (sensors).(scale).[0.0], (sensors).(minValue).[0.0], (sensors).(maxValue).[0.0], (sensors).(samplingRate).[0.0], (sensors).(nSamples).[0.0], (sensors).(lastSampleIndex).[0.0], (sensors).(oldestSampleIndex).[0.0], (sensors).(statsInterval).[0.0], (sensors).(median).[0.0], (sensors).(sampleType).[0.0], (actuators).(type).[0.0], (actuators).(description).[0.0] , (actuators).(units).[0.0] , (actuators).(scale).[0.0], (actuators).(setValue).[0.0], (actuators).(timeStamp).[0.0], (actuators).(relatedSensorIndex).[0.0]}
- *ValuesList ->* { }
- *ErrorsList ->* { }

5 Dispositivo -> Agente SNMP

- *PrimitiveType -> Response*
- *IdentifiersList ->* {(sensors).(type).[0.0], (sensors).(description).[0.0], (sensors).(units).[0.0], (sensors).(scale).[0.0], (sensors).(minValue).[0.0], (sensors).(maxValue).[0.0], (sensors).(samplingRate).[0.0], (sensors).(nSamples).[0.0], (sensors).(lastSampleIndex).[0.0], (sensors).(oldestSampleIndex).[0.0], (sensors).(statsInterval).[0.0], (sensors).(median).[0.0], (sensors).(sampleType).[0.0], (actuators).(type).[0.0], (actuators).(description).[0.0] , (actuators).(units).[0.0] , (actuators).(scale).[0.0], (actuators).(setValue).[0.0], (actuators).(timeStamp).[0.0], (actuators).(relatedSensorIndex).[0.0]}
- *ValuesList ->* {{“temperatura”, “humidade”}, {“Sensor responsável pela medição da temperatura ambiente”, “Sensor responsável pela medição da humidade ambiente” }, {“°C”, “%”}, {-1,-1}, {-10,0}, {57,100}, {33,33}, {0,0}, {0,0}, {60,60}, {0,0}, {1,1}, {“atuadorEstadoAC”, “atuadorTemperaturaAC”}, {“ Atuador responsável por ligar ou desligar o ar condicionado”, “Atuador responsável por estabelecer um valor de temperatura que se pretende aclimatizar”}, {“ON/OFF”, “°C”}, {0,0}, {“”,””}, {0,1}}
- *ErrorsList ->* { }

Obtenção da última amostra lida por um sensor

Figura 12: Comunicação entre Dispositivo e Agente SNMP para obtenção da última amostra lida por um sensor

Para a obtenção de uma amostra do sensor de temperatura, o agente envia uma mensagem *get-response* para saber a chave da última amostra guardada pelo dispositivo (1).

1 Agente SNMP -> Dispositivo

- *PrimitiveType -> Get-Request*
- *IdentifiersList* -> {(sensors).(lastSampleIndex).[1]}
- *ValuesList* -> { }
- *ErrorsList* -> { }

O dispositivo responde com mensagem de *response* com o valor do último índice guardado, ou seja o índice com a data mais recente (2).

2 Dispositivo -> Agente SNMP

- *PrimitiveType -> Response*
- *IdentifiersList* -> {(sensors).(lastSampleIndex).[1]}

- *ValuesList* -> {14}
- *ErrorsList* -> { }

Depois de conhecida a chave (14 no caso), o agente pede ao dispositivo que envie a amostra com aquele índice (3).

3 Agente SNMP -> Dispositivo

- *PrimitiveType* -> Get-Request
- *IdentifiersList* -> {(samples).(value).[14], (samples).(timeStamp).[14]}
- *ValuesList* -> { }
- *ErrorsList* -> { }

Após a receção da mensagem *get-request* com os **IID**s de *(samples).(value).[14]e (samples).(timeStamp).[14]* o dispositivo responde com os valores correspondentes ao índice/chave 14 das amostras (4).

4 Dispositivo -> Agente SNMP

- *PrimitiveType -> Response*
- *IdentifiersList* -> {(samples).(value).[14], (samples).(timeStamp).[14]}
- *ValuesList* -> {192 , "10/10/2022 14:05:50"}
- *ErrorsList* -> { }

Atuação num dispositivo

Figura 13: Comunicação entre Dispositivo e Agente SNMP de atuação num dispositivo

Para configurar/atuar sobre o dispositivo (ligar ar condicionado e estabelecer temperatura a aclimatizar para 21 °C) o agente envia uma mensagem *set-request* onde indica o valor a inserir no atuador do dispositivo bem como o *timestamp* que representa a data e hora da atuação (1).

1 Agente SNMP -> Dispositivo

- *PrimitiveType -> Set-Request*
- *IdentifiersList* -> {(actuators).(setValue).[2], (actuators).(timeStamp).[2]}
- *ValuesList* -> { 21, "10/10/2022 14:06:40"}
- *ErrorsList* -> { }

Caso tudo corra bem e a atuação seja feita o dispositivo envia uma mensagem *response* que contém a mesma informação (lista de identificadores e valores) que recebeu. Esta mensagem serve como uma confirmação da introdução do valor no atuador e permite ao agente, após receber esta mensagem, guardar os valores na sua **MIB** (2).

2 Dispositivo -> Agente SNMP

- *PrimitiveType -> Response*

- *IdentifiersList* -> {(actuators).(setValue).[2], (actuators).(timeStamp).[2]}
- *ValuesList* -> {21, “10/10/2022 14:06:42”}
- *ErrorsList* -> { }

Capítulo 5

Aplicações

Um dos principais objetivos dos trabalhos da dissertação era o desenvolvimento dum sistema protótipo que pudesse validar a solução proposta. O protótipo terá que integrar implementação de módulos de *software* de todos os elementos da arquitetura, incluindo um módulo simulador de um dispositivo domótico.

5.1 Ambiente de desenvolvimento

A implementação do sistema foi efetuado utilizando uma máquina virtual *Linux Ubuntu* como sistema operativo, com o auxílio de várias ferramentas de desenvolvimento.

A linguagem *JAVA* foi adotada para desenvolver todos os componentes do protótipo devido a um conjunto de fatores, mas a vantagem principal é a compatibilidade com ferramentas de edição e implementação de **MIB**s para os agentes **SNMP**, nomeadamente a **API** *SNMP4J*, uma biblioteca *Open Source* que disponibiliza classes que implementam o protocolo **SNMP** e todo um conjunto de operações associadas à manipulação dos valores das instâncias dos objetos da **MIB**.

Relativamente à aplicação de controlo (gestor **SNMP**) onde é possível a visualização dos dados dos dispositivos bem como os seus sensores e atuadores recorreu-se novamente à linguagem *JAVA* e à biblioteca *SNMP4J* .

5.2 Ferramentas utilizadas

Com o intuito de facilitar e acelerar o desenvolvimento de todo o sistema referido, a utilização de um conjunto variado de ferramentas e bibliotecas tornou-se imprescindível e o seu uso permitiu a construção e evolução do sistema. Assim, nesta secção serão referidas as principais ferramentas e bibliotecas utilizadas.

5.2.1 SNMP4J

O *SNMP4J* [40] é uma **API** *open source* que permite o desenvolvimento de aplicações **SNMP** utilizando o *JAVA* como linguagem programática e suporta todas as três versões do protocolo.

A **API** apresenta como principais características:

- Implementação de todos os tipos de **PDU SNMP**;
- Suporte de protocolos **UDP** e **TCP** para encapsular as mensagens **SNMP**;
- Suporte a autenticação **MD5** e **SHA** com o **SNMPv3**;
- Suporte à utilização de programação com processos concorrentes;
- Suporte de várias funcionalidades de manipulação de valores de instâncias de objetos das **MIB**.

5.2.2 MIB Designer

O *MIB Designer* [41] é uma ferramenta que permite a conceção visual e edição de especificações de módulos **MIB** de acordo com a norma **SMI**. Com a utilização da ferramenta o utilizador não precisa de estar familiarizado com a sintaxe exata e rigorosa do **ASN.1** definida na **SMI** e que permitiram facilitar a especificação formal da *Domotics MIB*.

As funcionalidades fornecidas pela ferramenta são:

- Criação assistida de novos módulos **MIB** garantindo a sua correção sintática;
- Suporte de edição de todo o tipo de objetos **SMIv2**;
- Interface gráfico que facilita a criação e edição de tabelas em **MIB**;
- Suporte para exportar os módulos **MIB** em formatos *txt*, **HTML**, **XML**, **XSD** e **PDF**.

5.2.3 AgenPro

O *AgenPro* [42] é uma ferramenta que permite a geração automática e assistida de código para o desenvolvimento de aplicações **SNMP** (agentes e gestores **SNMP**).

Entre o conjunto variado de funcionalidades que a ferramenta oferece enumeram-se as principais:

- Disponibilizar um repositório de **MIB** normalizadas;
- Gerar código em *JAVA* ou *C++*;
- Garantir a implementação correta de múltiplos índices, ordenamento de tabelas, armazenar tipos e outras convenções textuais possíveis no **SNMP**.

No contexto do trabalho realizado, o software permitiu compilar a *Domotics MIB*. e obter os ficheiros que deram origem às classes *JAVA* para facilitar a implementação da instrumentação da **MIB** no código do agente **SNMP**.

5.3 Domotics MIB

A especificação formal da **MIB** foi criada seguindo a **SMIv2**, que, tal como referido em capítulos anteriores, representa a notação com a qual uma **MIB** deve ser escrita. Para uma melhor perceção da sua sintaxe, apresenta-se a especificação do grupo *device* (estando a restante **MIB** presente no Anexo A.2) seguida de uma pequena explicação da sua definiçao.

```
domoDevice OBJECT IDENTIFIER
-- 1.3.6.1.3.1.2
::= { domoticsMIB 2 }

domoDeviceNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Total number of existing devices in the system."
-- 1.3.6.1.3.1.2.1
::= { domoDevice 1 }


domoDeviceTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DeviceEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"A list with the set of devices that are part of the system, corresponding
an entry to each device."
-- 1.3.6.1.3.1.2.2
::= { domoDevice 2 }


domoDeviceEntry OBJECT-TYPE
SYNTAX  DomoDeviceEntry
MAX-ACCESS not-accessible
```

```
STATUS  current
DESCRIPTION
"An entry that contains management information applicable to each device
in particular."
INDEX {
domoDeviceIndex }
-- 1.3.6.1.3.1.2.2.1
::= { domoDeviceTable 1 }


DomoDeviceEntry ::= SEQUENCE {

domoDeviceIndex             Integer32,
domoDeviceDescription       OCTET STRING,
domoDeviceIdentification    OCTET STRING,
domoDeviceBeaconRate        Integer32,
domoDeviceManufacturer      OCTET STRING,
domoDeviceModel             OCTET STRING,
domoDeviceVersionAndDate    OCTET STRING,
domoDeviceNSensors          Integer32,
domoDeviceNActuators        Integer32,
domoDeviceNMaxSamples       Integer32,
domoDeviceDateAndTime       OCTET STRING,
domoDeviceUpTime            Integer32,
domoDeviceLastTimeUpdated   OCTET STRING,
domoDeviceOperationalStatus Integer32,
domoDeviceBatteryStatus     Integer32,
domoDeviceReset             Integer32}


domoDeviceIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
```

```
DESCRIPTION
"A single value greater than zero for each new device. The values
assigned must vary between 1 and N devices.
"
-- 1.3.6.1.3.1.2.2.1.1
::= { domoDeviceEntry 1 }
```

Inicialmente é definido o objeto **domoDevice** com o **OID** **1.3.6.1.3.1.2**, ou seja, defini-se um novo nodo na árvore de **OID**s da **MIB**. Esta declaração identifica portanto, um grupo onde todas as informações relativas aos dispositivos estarão agrupadas.

Em seguida é definido o objeto simples **domoDeviceNumber** sendo do tipo *Integer32*, indicando o número de dispositivos existentes no sistema domótico (ou seja, o número de linhas existentes na tabela de dispositivos). Como este valor não pode ser alterado por nenhum gestor está definido como *read-only*, o que significa que pode somente ser lido e não alterado.

A seguir encontra-se a especificação da tabela de dispositivos (**domoDeviceTable**). Esta tabela é constituída por uma sequência de entradas/linhas (**domoDeviceEntry**), que por sua vez, são constituídas por um conjunto de objetos que representam todas as informações possíveis de serem associadas a um dispositivo (apresentados em **DomoDeviceEntry ::= SEQUENCE**). Quer a tabela quer a entrada estão declarados como *not-acessible*, o que significa que não podem ser acedidos diretamente por nenhum gestor.

Por fim, inclui-se a especificação de um dos objetos das linhas da tabela, neste caso, o objeto **domoDeviceIndex**, que é a primeira coluna da tabela e representa o índice atribuído a cada dispositivo.

A implementação da **MIB** no agente foi feita com o uso da já mencionada ferramenta *AgenPro* e o resultado obtido dessa compilação está presenta na figura 14, que revela a árvore da **MIB** criada sobre o ramo experimental e denominada de **DomoticaSNMPMIB**.

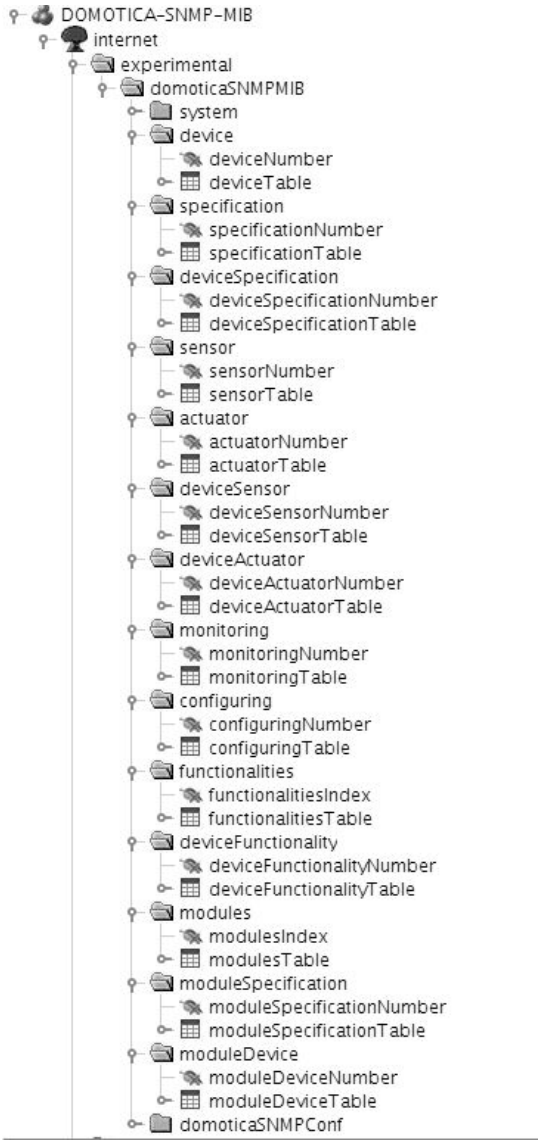


Figura 14: Esquema do processo de geração de código pelo AgenPro.

Esta **MIB** é composta por 15 grupos e é a instanciação do modelo de informação apresentado.

Finalmente é importante referir que na especificação da *Domotics MIB* foi acrescentado o grupo *system*, com intuito de armazenar algumas informações relevantes sobre a administração do sistema onde o agente é instalado.

5.4 AgenteSNMP

O agente **SNMP** é o constituinte mais importante do todo o sistema, devido ao facto de todos os dados e informação do sistema convergirem neste ponto, quer seja para a camada superior de gestão ou para a camada inferior dos dispositivos.

O agente **SNMP** do protótipo contruído inclui uma instrumentação da *Domotics MIB* e através da utilização do já referido *AgenPro* foi gerado um primeiro esqueleto do código de onde se partiu para a restante implementação

do agente.

Para uma compreensão mais detalhada sobre o processo de geração de código pelo *AgenPro*, apresenta-se na figura 15 um esquema onde se explica os principais passos executados pela ferramenta desde a importação da **MIB** até á criação dos ficheiros de código JAVA correspondentes.

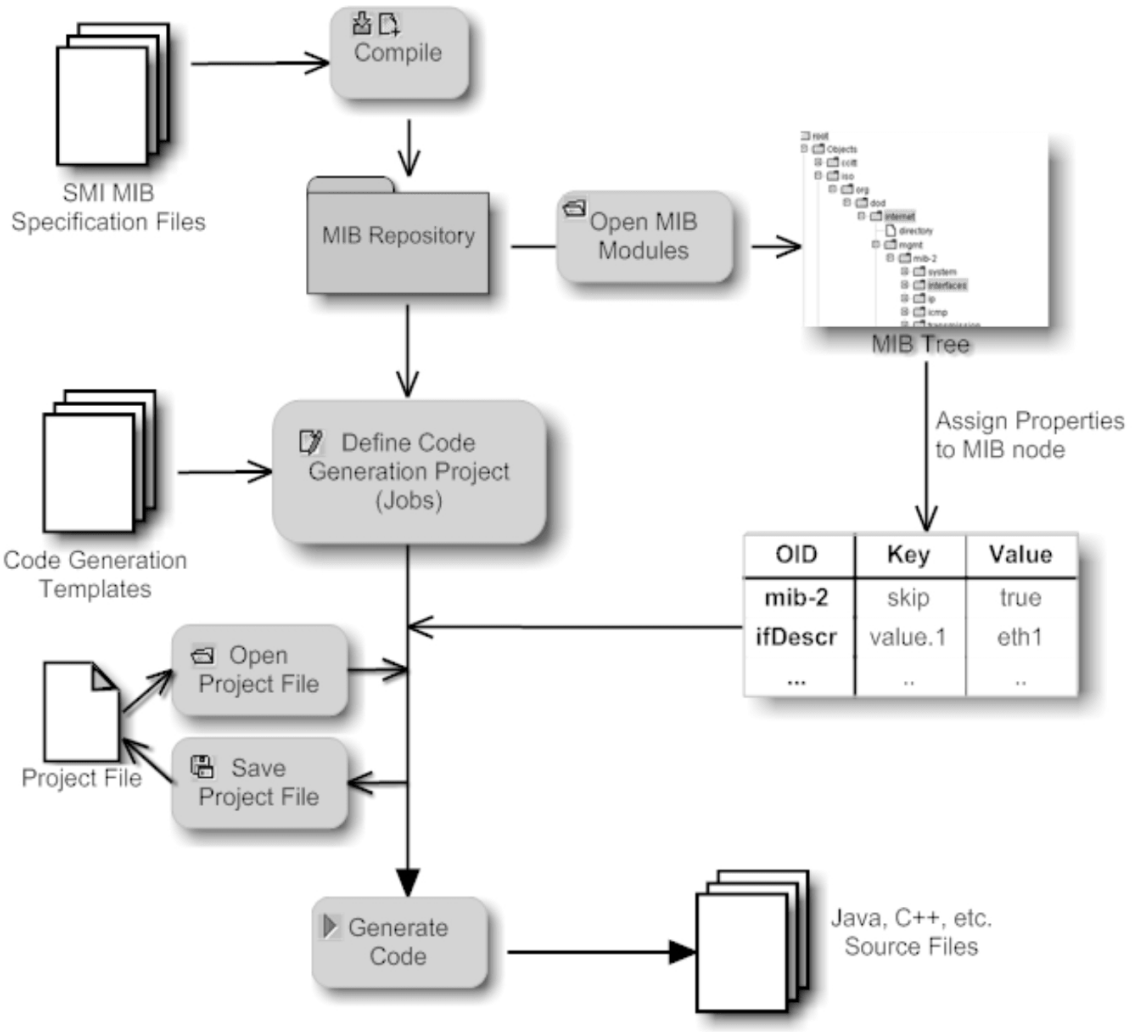


Figura 15: Esquema do processo de geração de código pelo AgenPro [2].

Os ficheiros contêm as classes sobre as quais se desenvolveu o agente. A figura 16 apresenta algumas informações da classe DOMOTICASNMPMIB (apenas alguns exemplos das suas variáveis e dos seus métodos).

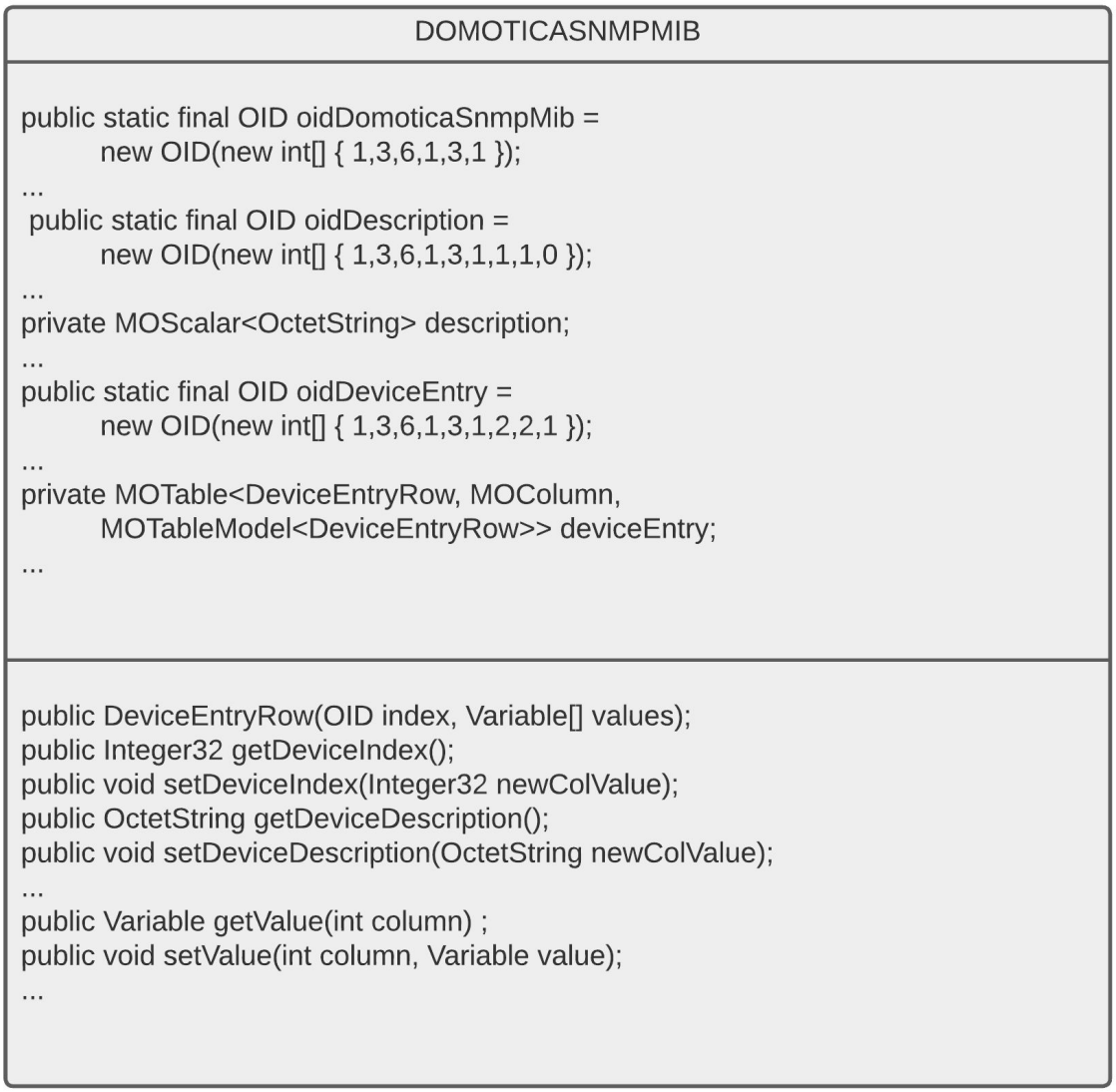


Figura 16: Diagrama da classe *DomoticaSNMPMIB*.

A classe *Agent*, também gerada inicialmente a partir do *AgenPro* contém alguns métodos e as variáveis do agente **SNMP** tal como se pode verificar na figura 19.

Figura 17: Diagrama da classe *Agent*.

Além dos métodos inicialmente gerados pela ferramenta, foram criados outros métodos para lidar com toda a lógica do agente **SNMP**. Estes métodos incluem funcionalidades da instrumentação (*msgHandler(receiverProtocol)*), armazenamento e recolha de informação da **MIB**. Além disso, também foi criado no agente um *lookupListener* que, aquando da configuração de um valor num atuador, ou seja, quando recebe um *SNMPSET* do gestor, o agente envia uma mensagem **L-SNMP** ao dispositivo para efetuar a alteração pretendida e caso a mesma seja bem sucedida, o dispositivo envia uma mensagem **L-SNMP** de resposta ao agente **SNMP** que, após a sua receção, cria uma nova entrada na tabela *configuring* da **MIB** com o índice do atuador, o índice do dispositivo que está associado a esse atuador e a data e hora atual.

5.4.1 Instrumentação da Domotics MIB e implementação do L-SNMP

A implementação da instrumentação da *Domotics MIB* é de grande relevância no sistema. O código gerado pelo agenPro não contempla este componente e foi, por isso, necessário proceder-se à sua implementação integral e posterior integração no agente, permitindo assim uma comunicação com os dispositivos na camada física. A instrumentação permite ao agente comunicar com os dispositivos utilizando o **L-SNMP** e para isso foi necessário a criação de dois módulos para enviar e receber mensagens **L-SNMP**.

Além desses dois módulos de comunicação foi também necessário a criação de uma classe para gerir as informações sobre os **Instance Identifier** *(***IID***)* . Estes identificadores são definidos e explicados no modelo de informação do *Light SNMP* em 4.3.2 e na figura 67 está detalhado o diagrama desta classe.

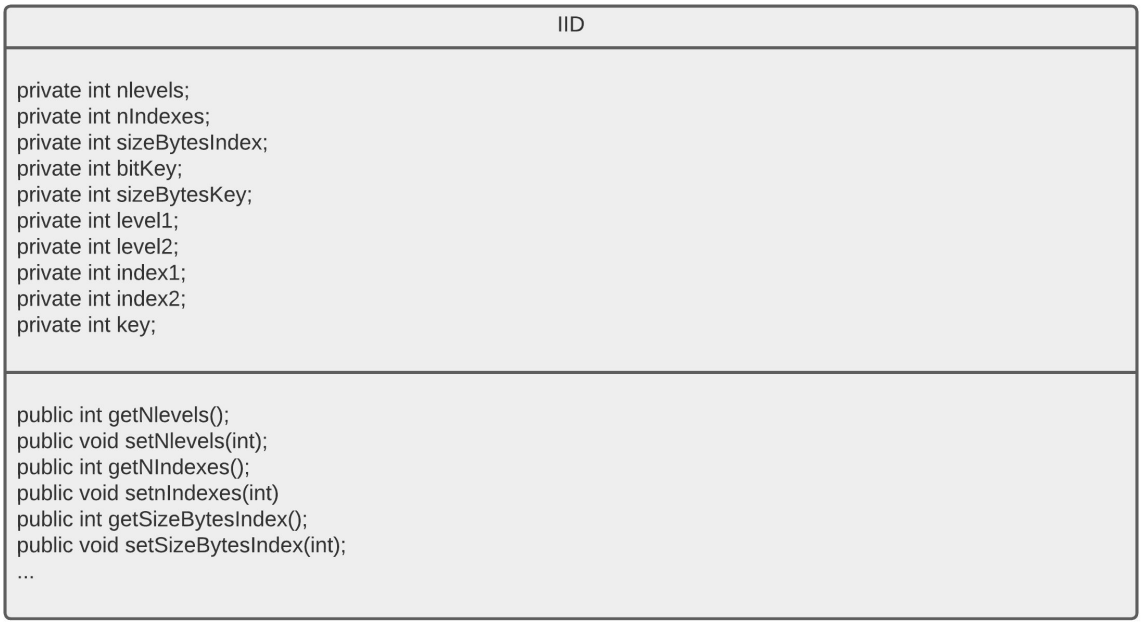


Figura 18: Diagrama da classe *IID*.

Nesta classe as variáveis de instância representam todo o tipo de informação que é necessário saber sobre o *IID*, o número de níveis e de índices, o tamanho que cada índice e chave ocupam e os valores dos níveis, índices e chave, caso existam.

Relativamente aos métodos desta classe, estão apenas presentes os tradicionais métodos *get* para obter o valor da variável e *set* para alterar o valor.

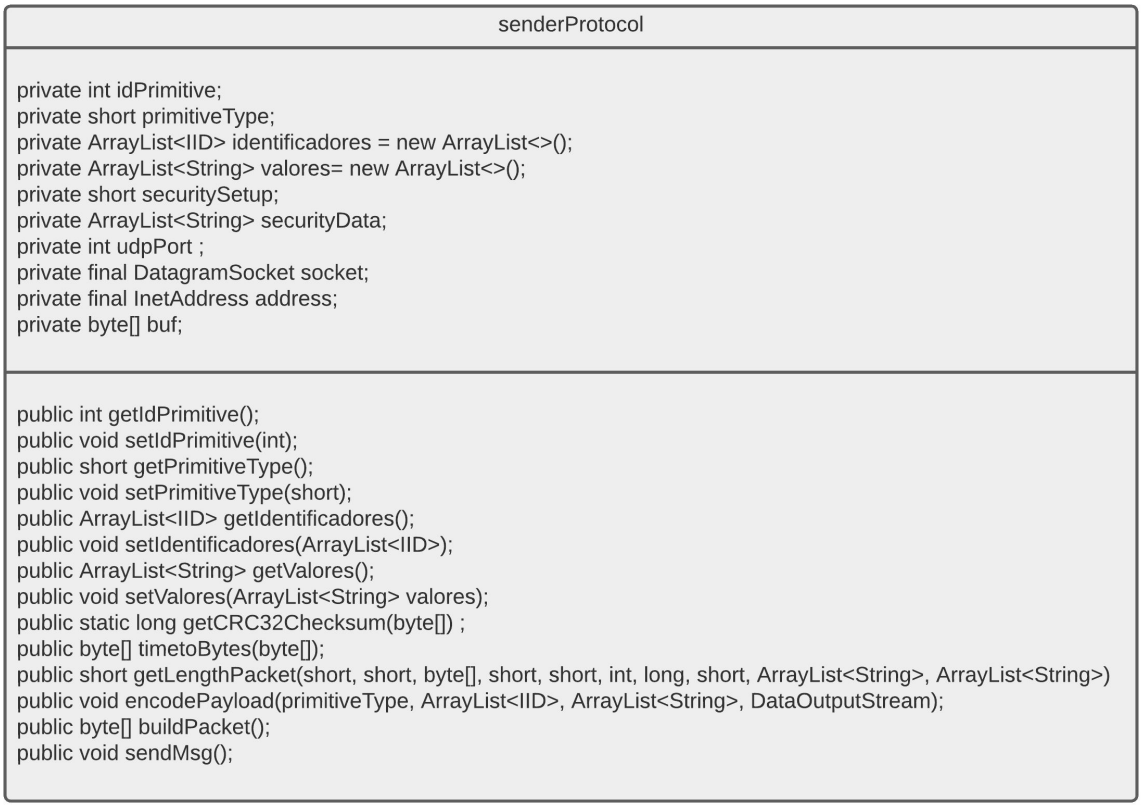


Figura 19: Diagrama da classe *senderProtocol*.

A classe *senderProtocol* 19 é responsável por formar uma mensagem de acordo com o formato do **L-SNMP** codificando os vários tipos de dados em sequências binárias de acordo com o modelo de codificação apresentado na secção 4.3.4 (diagrama de classe na figura 20).

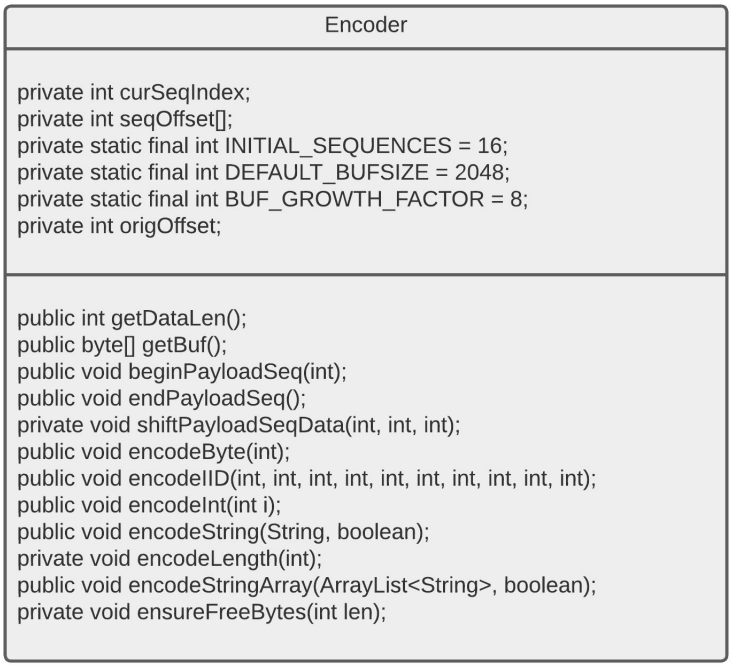


Figura 20: Diagrama da classe *Encoder*.

Também foi criada uma classe designada de *receiverProtocol* (diagrama de classe na figura 21) que é responsável por implementar a receção de mensagens do protocolo **L-SNMP**. Esta classe invoca também métodos da classe *Encoder* (figura 22) capazes de descodificar os *bytes* recebidos em informação útil e passível de ser utilizada pelo agente **SNMP**.

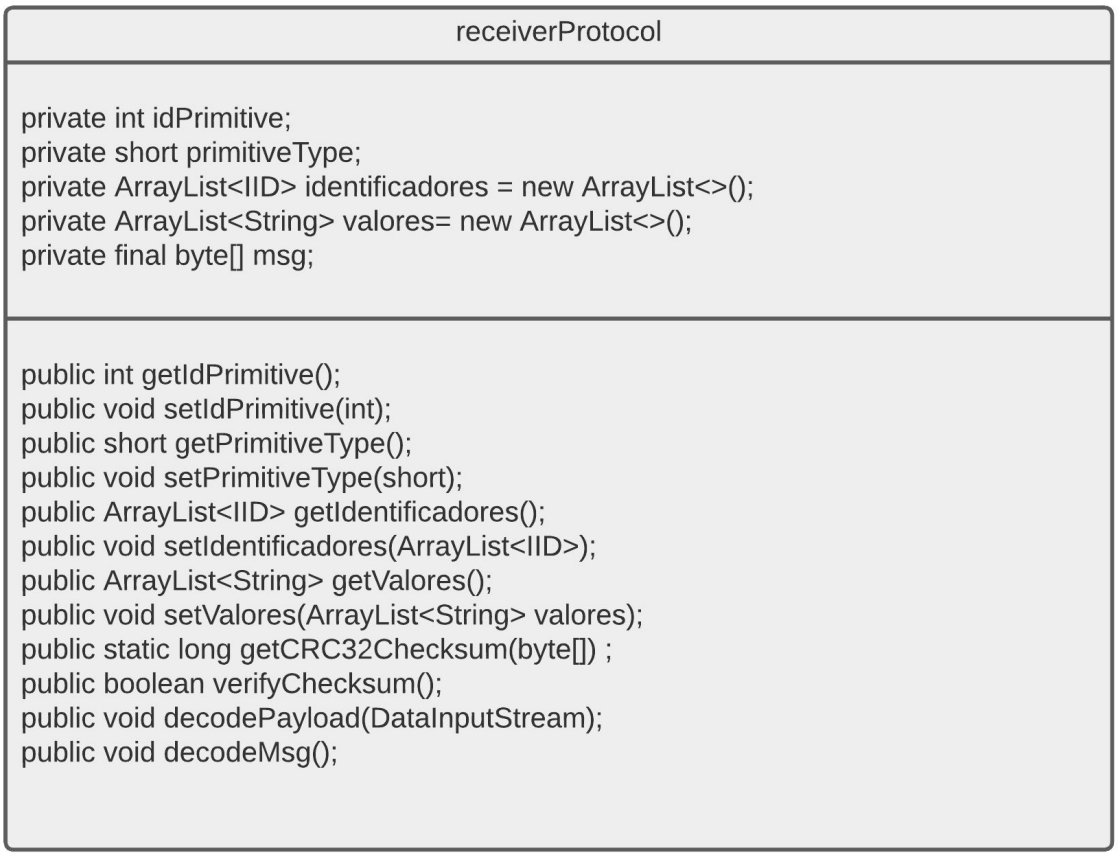


Figura 21: Diagrama da classe *receiverProtocol*.

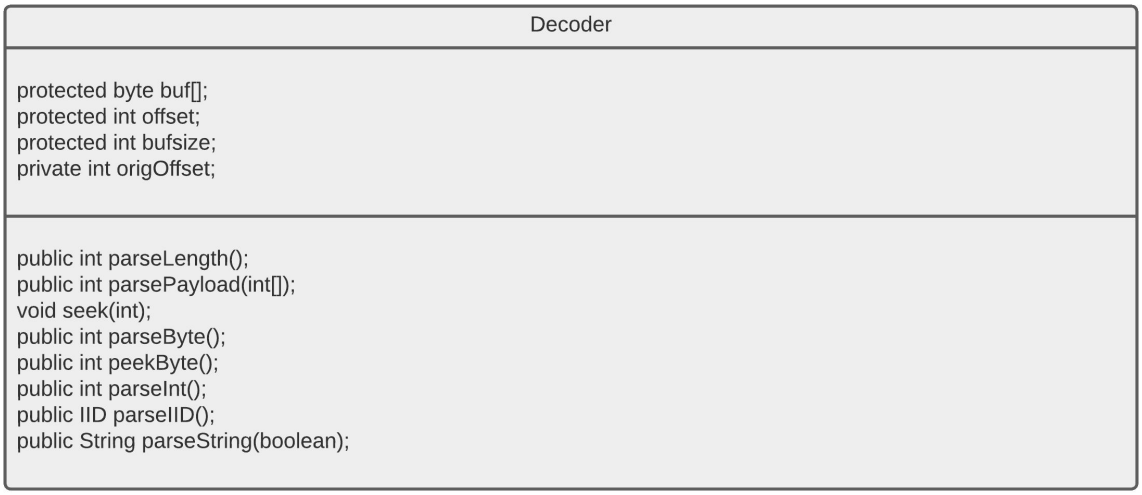


Figura 22: Diagrama da classe *Decoder*.

Na figura 23 é apresentado um diagrama de estados que descreve o comportamento do protocolo **L-SNMP** e que foi implementado no agente **SNMP**.

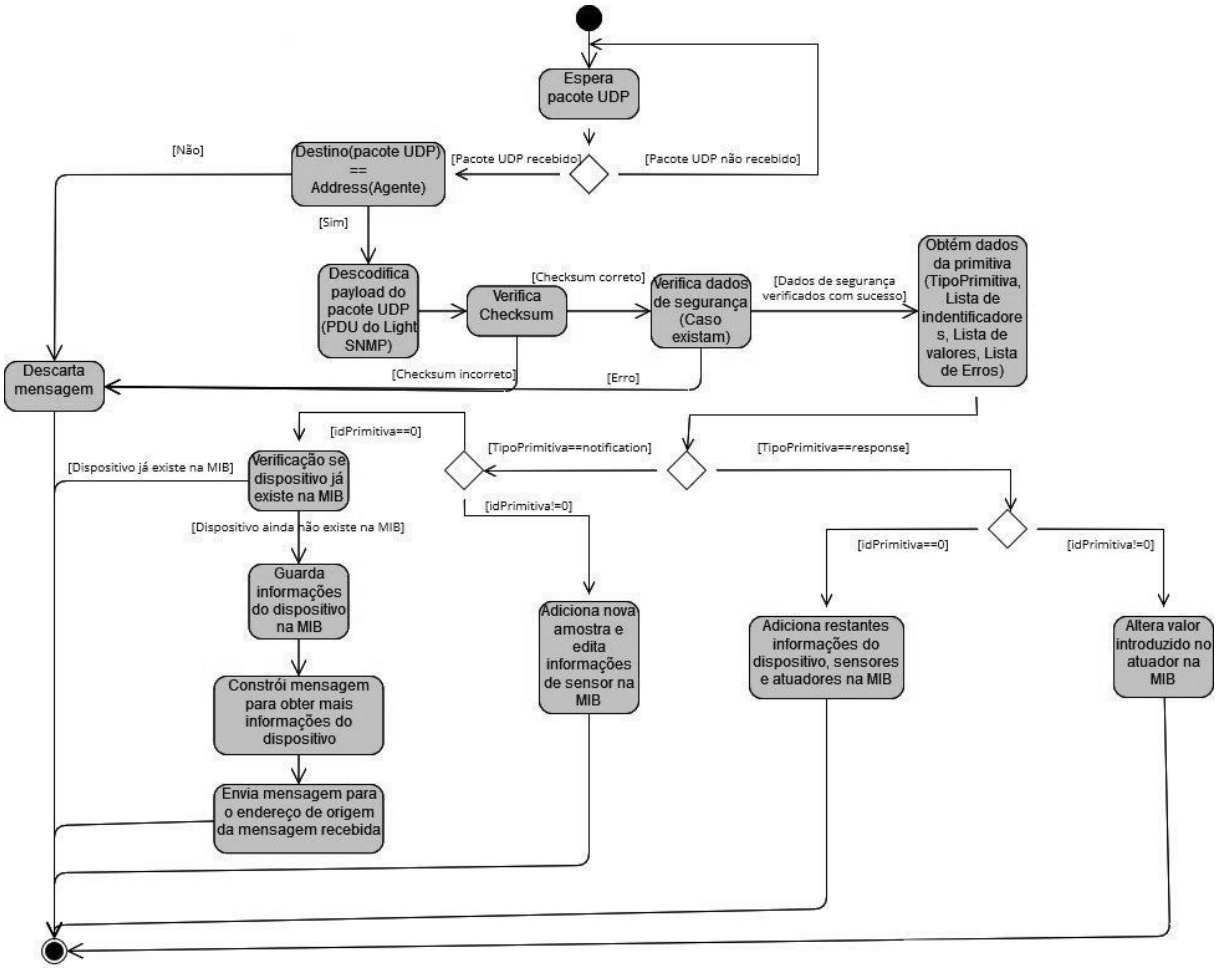


Figura 23: Diagrama de estados de comunicação Dispositivo-Agente SMNP no agente SNMP.

Quando é enviada uma mensagem do dispositivo para o agente **SNMP** é verificado o destino, o *checksum*, os dados de segurança e a primitiva. Primeiro é verificado se o destino do pacote UDP recebido é igual ao endereço do agente. Caso isso não se verifique a mensagem é descartada, senão é descodificada a informação presente no *payload* do pacote, que corresponde ao **PDU** do **L-SNMP** e é verificado se o *checksum* está correto. Caso se observe que esse valor não é o esperado a mensagem é descartada. O próximo passo é a verificação dos dados de confidencialidade caso estes estejam presentes na mensagem. Se não existirem dados de confidencialidade significa que não existe qualquer tipo de segurança e este passo é ignorado, caso existam e não sejam os esperados a mensagem é descartada. Após todas as verificações anteriores é feita a descodificação do *payload* do PDU que inclui as informações mais relevantes incluindo o tipo da primitiva, a lista de identificadores, a lista de valores e a lista de erros.

Se a primitiva recebida for uma *notification* e o *requestId* seja 0, significa que a mensagem corresponde a uma mensagem que funciona como espécie de *beacon* e indica que um dispositivo está a dar-se a conhecer ao agente que verifica se o mesmo já está presente na **MIB**. Caso isso se verifique o agente ignora a mensagem mas, caso o dispositivo ainda não esteja presente as suas informações são guardadas na tabela *device* e é construída uma

mensagem do tipo *get-request* com um requestId igual a 0 e com uma lista de identificadores que representam as restantes informações sobre o dispositivo bem como dos seus sensores e atuadores. A mensagem é enviada tendo como destino o endereço e porta da origem da mensagem previamente recebida. Se o *requestId* for diferente de 0, significa que o agente está a receber informações de amostras de sensores de dispositivos já presentes na **MIB** e assim sendo, essas informações são também guardadas nas tabelas *monitoring* e *sensor*. Quando o tipo da primitiva é *response*, se o *requestId* for igual a 0, significa que a mensagem contém informações adicionais do dispositivo, sensores e atuadores previamente requeridas pelo agente e essas informações são adicionadas na **MIB**. Se o *requestId* for diferente de 0, a mensagem é uma resposta de confirmação da introdução de um valor de um atuador do dispositivo o agente adiciona esse valor e o seu *timeStamp* na tabela *configuring* na **MIB**, bem como na tabela que contém informações sobre o atuador.

Na figura 24 apresenta-se o diagrama de estados que descreve o comportamento da instrumentação no dispositivo.

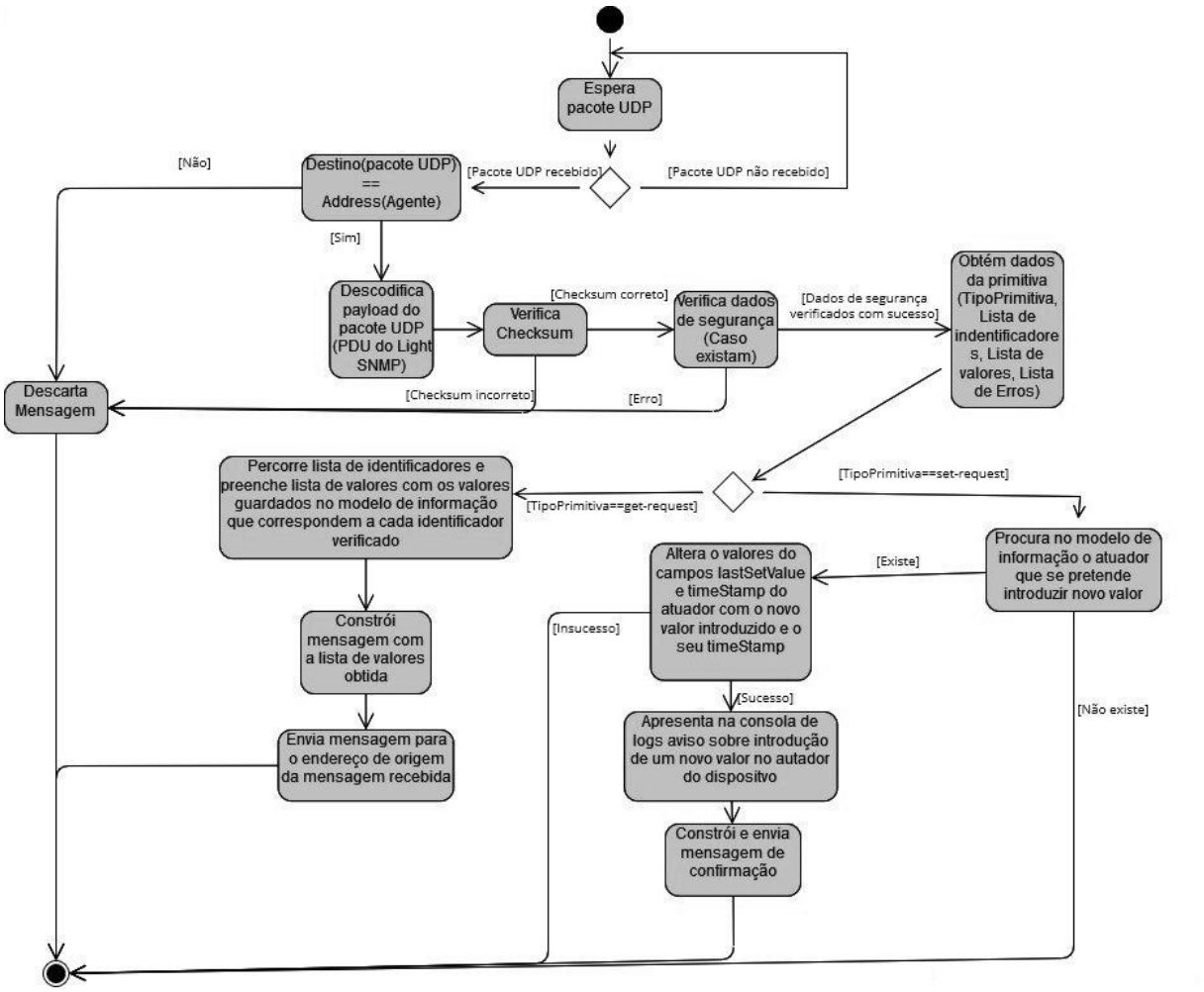


Figura 24: Diagrama de estados de comunicação Agente SNMP-Dispositivo no dispositivo.

No caso do envio de uma mensagem do agente para o dispositivo, a parte inicial de verificações de informações

de confidencialidade e garantia de uma correta transmissão de dados é semelhante ao explicado previamente na comunicação do dispositivo para o agente **SNMP**.

As diferenças começam na verificação do tipo de primitiva recebida que, neste caso, podem ser do tipo *get-request* ou *set-request*. Caso o tipo verificado seja o primeiro, o diapositivo entra num ciclo para percorrer a lista de identificadores, um a um, e verificar o que cada um representa e mediante o seu significado no contexto do modelo de informação definido para o protocolo é pesquisado o valor ou valores das instâncias respetivas. Após percorrida toda a lista de identificadores o agente **L-SNMP** do dispositivo cria uma nova mensagem com o mesmo *requestId* recebido, a mesma lista de identificadores e a lista de valores encontrados pela mesma ordem em que são requeridos e todos os outros campos conforme a estrutura definida no **PDU** do **L-SNMP**. A mensagem é enviada para o mesmo destino que a origem da mensagem recebida.

Caso o tipo da primitiva recebida seja *set-request* significa que é pretendido configurar um novo valor num dos atuadores do dispositivo. Sendo assim, é obtida a identificação do atuador ao qual se pretende efetuar uma alteração através da lista de identificadores e da lista de valores recebidos. O dispositivo verifica se esse atuador está associado a si e caso o encontre altera os campos *lastSetValue* e *timeStamp* desse atuador com os valores recebidos. Caso essa mudança seja bem sucedida, é gerada uma mensagem de resposta que é enviada de volta para o agente **SNMP**.

5.5 Gestor SNMP

Para ser possível gerir os dispositivos, consultando valores ou definindo valores de atuação, é necessário a construção de uma espécie de interface que permita ao utilizador interagir com o sistema desenvolvido. Tendo esse objetivo em mente, criou-se um pequeno programa em JAVA utilizando a biblioteca SNNMP4J, que permite comunicar com um ou mais agentes **SNMP**.

Este programa contém apenas três classes sendo uma delas apenas auxiliar e por isso não descriminada. A classe *main* (figura 25) responsável pelos métodos de representação gráfica e pela coordenação e lógica dos menus apresentados e a classe *SNMPManager* (figura 26) que contém os métodos de comunicação com um agente **SNMP**.

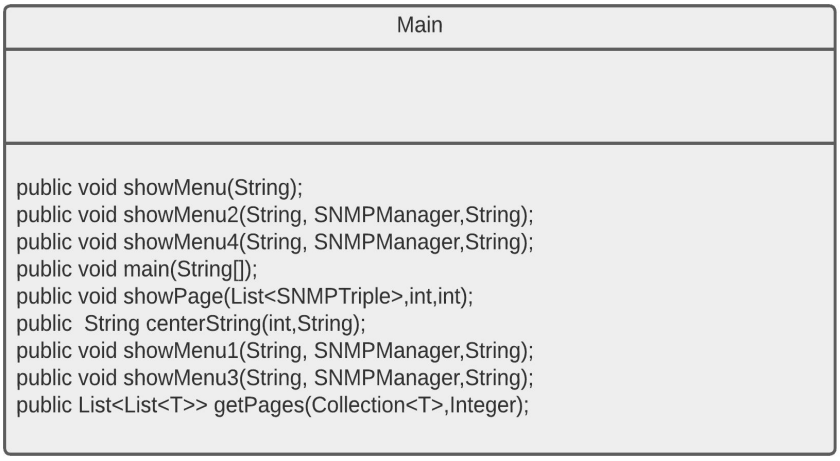


Figura 25: Diagrama da classe *main*.

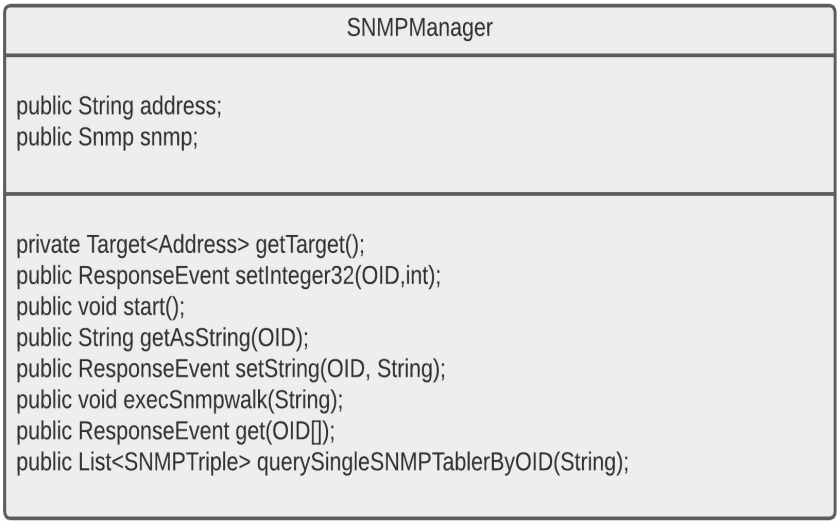


Figura 26: Diagrama da classe *SNMPManager*.

A aplicação gestora apresenta um simples menu no terminal que permite ao utilizador consultar todos os dispositivos geridos pela instrumentação dos agentes. Para efetuar algum tipo de consulta ou configuração o utilizador tem de primeiro introduzir o índice de um dos dispositivos existentes e em seguida selecionar uma das opções disponíveis: consultar informações do dispositivo, consultar valores dos sensores do dispositivo ou configurar atuadores.

5.6 Modelo do protótipo

A simulação dos equipamentos físicos domóticos apresenta uma importância de relevo no sistema visto que só com a utilização e integração de vários dispositivos é que será possível efetuar um conjunto válidos de testes da solução. A análise dos resultados obtidos servirão para concluir se a solução proposta cumpre os objetivos enunciados.

Os dispositivos devem ser capazes de recolher dados através de sensores simuladores ou produzir ações através de atuadores simuladores.

Na arquitetura proposta, os dispositivos são considerados entidades de *middleware*, podendo conter atuadores que executam algum tipo de função e/ou sensores que monitorizam informação do ambiente onde os dispositivos estão presentes, sendo estes sensores e atuadores os componentes de mais baixo nível funcional no sistema.

Tal como detalhado na secção 5.4.1, os dispositivos possuem capacidade de comunicar com o agente **SNMP** através da utilização do protocolo **L-SNMP**. No protótipo desenvolvido, os dispositivos utilizam as classes e os métodos desenvolvidos para o agente **SNMP**, nomeadamente as classes *senderProtocol* (com a utilização da classe *Encoder* para codificar os dados de acordo com as regras de codificação), *receiverProtocol* e **IID**, já apresentadas em secções anteriores.

Assim, no protótipo criado foram implementados três tipos de equipamentos domóticos: um estore elétrico, um ar condicionado e luzes ajustáveis (luzes onde se pode ajustar a luminosidade em vários níveis). O esquema completo do protótipo é apresentado na figura 27.

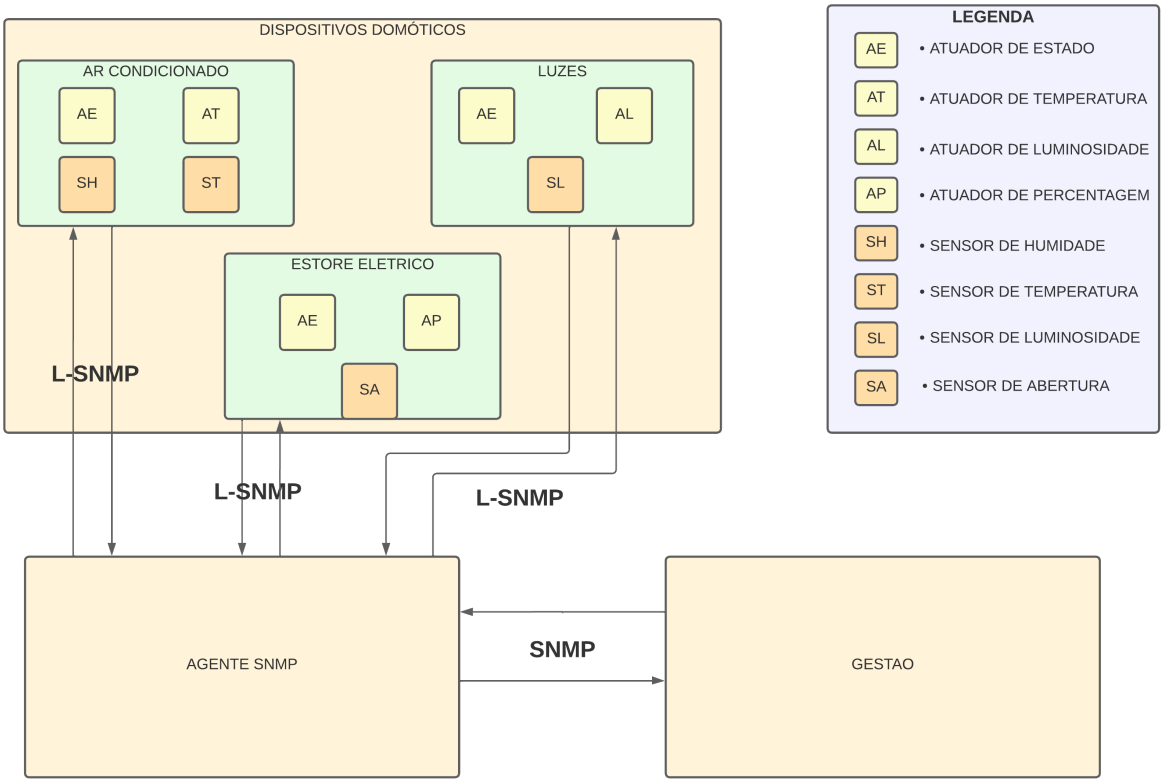


Figura 27: Esquema do protótipo criado com simulação do dispositivo domótico.

Cada dispositivo contém um número de sensores e atuadores diferente e com diferentes características. O estore elétrico utiliza um sensor e um atuador:

- Sensor de Abertura, responsável por medir o nível de abertura do estore;
- Atuador de Abertura, responsável por abrir ou fechar o estore.

O ar condicionado utiliza dois sensores e dois atuadores:

- Sensor de Temperatura, responsável por medir a temperatura ambiente;
- Sensor de Humidade, responsável por medir a humidade relativa ambiente;
- Atuador de Temperatura, responsável por definir a temperatura alvo;
- Atuador de Estado, responsável por ligar e desligar (*stand-by*) o equipamento.

As luzes são representadas por um sensor e dois atuadores:

- Sensor de Luminosidade, responsável por medir a luminosidade ambiente;
- Atuador de Luminosidade, responsável por ajustar a intensidade das luzes de acordo com o pretendido;
- Atuador de Estado, responsável por ligar e desligar as luzes.

Capítulo 6

Testes e Avaliação de Resultados

Os testes foram realizados executando os módulos de software do protótipo numa máquina virtual *Ubuntu Linux* com a versão 20.04 através da ferramenta **IDEA** *InteliJ Ultimate* versão 2021.3.2 .

A aplicação controladora permitiu introduzir alterações na configuração dos dispositivos e consultar informações, quer sobre os dispositivos em si, quer sobre os equipamentos simuladores (sensores e atuadores):

- No estore é possível alternar o seu estado entre ligado e desligado e definir um nível de abertura específica, caso se pretenda que o estore não fique totalmente aberto ou fechado.

- No caso do ar condicionado é possível alternar o seu estado entre ligado e desligado, definir uma temperatura alvo e consultar os dados de temperatura e humidade ambiente, medidos pelos sensores que efetuam essas medidas, independentemente do estado do dispositivo.

- No caso das luzes, para além de ser possível ligar ou desligar, também é possível definir um nível de intensidade luminosa com uma percentagem de intensidade máxima possível.

Ao iniciar a aplicação de gestão é apresentado ao utilizador todos os equipamentos que podem ser geridos (Fig. 28), ou seja, aqueles que são simulados nos dispositivos conectados.

```
Bem vindo ao SNMPManager (Press ENTER to continue)

                          Dispositivos disponíveis
*********************************************************************************
| INDEX  |                          DESCRIPTION                  |  IDENTIFICATION |
*********************************************************************************
| 1      | Luzes do escritorio do primeiro piso                  | Luzes           |
| 2      | Estore do escritorio do primeiro piso                 | Estore          |
| 3      | Ar condicionado da sala de estar do primeiro piso     | ArCondicionado  |
*********************************************************************************
Introduza o INDEX do dispositivo a gerir ou exit para sair
```

Figura 28: Menu inicial com os equipamentos controláveis.

Nesta aplicação controladora apenas é possível gerir um equipamento de cada vez, o que significa que o utilizador tem de indicar qual o dispositivo que pretende gerir e, quando pretender gerir outro dispositivo, necessita

de voltar a este menu inicial.

6.1 Controlo das luzes

Caso pretenda controlar as luzes o utilizador deverá ter à sua disposição o menu apresentado no figura 29, que resulta da escolha respetiva no menu inicial.

```
Introduza o INDEX do dispositivo a gerir ou exit para sair
1
Menu do dispositivo -> Luzes do escritorio do primeiro piso
--------------------------------
1 - Informações sobre o dispositivo
2 - Consultar valores atuais de sensores
3 - Consultar histórico de valores de monitorizacao (tabela)
4 - Consultar histórico de valores de monitorizacao (grafico)
5 - Introduzir valor em atuadores
6 - Voltar
--------------------------------
Introduza opção:
```

Figura 29: Menu que permite gerir as luzes.

Este menu apresenta todas as opções que o utilizador pode tomar sobre o dispositivo, como consultar informações sobre o próprio dispositivo, consultar valores atuais de sensores, consultar histórico de valores de monitorização, ou seja valores medidos previamente pelos sensores, podendo ser observada através de uma tabela ou de um gráfico e por último introduzir um valor num atuador do dispositivo.

Caso se pretenda visualizar as informações respeitantes ao equipamento (dispositivo) o utilizador introduz a opção 1 sendo apresentada uma tabela de informações como a presente na figura 30.

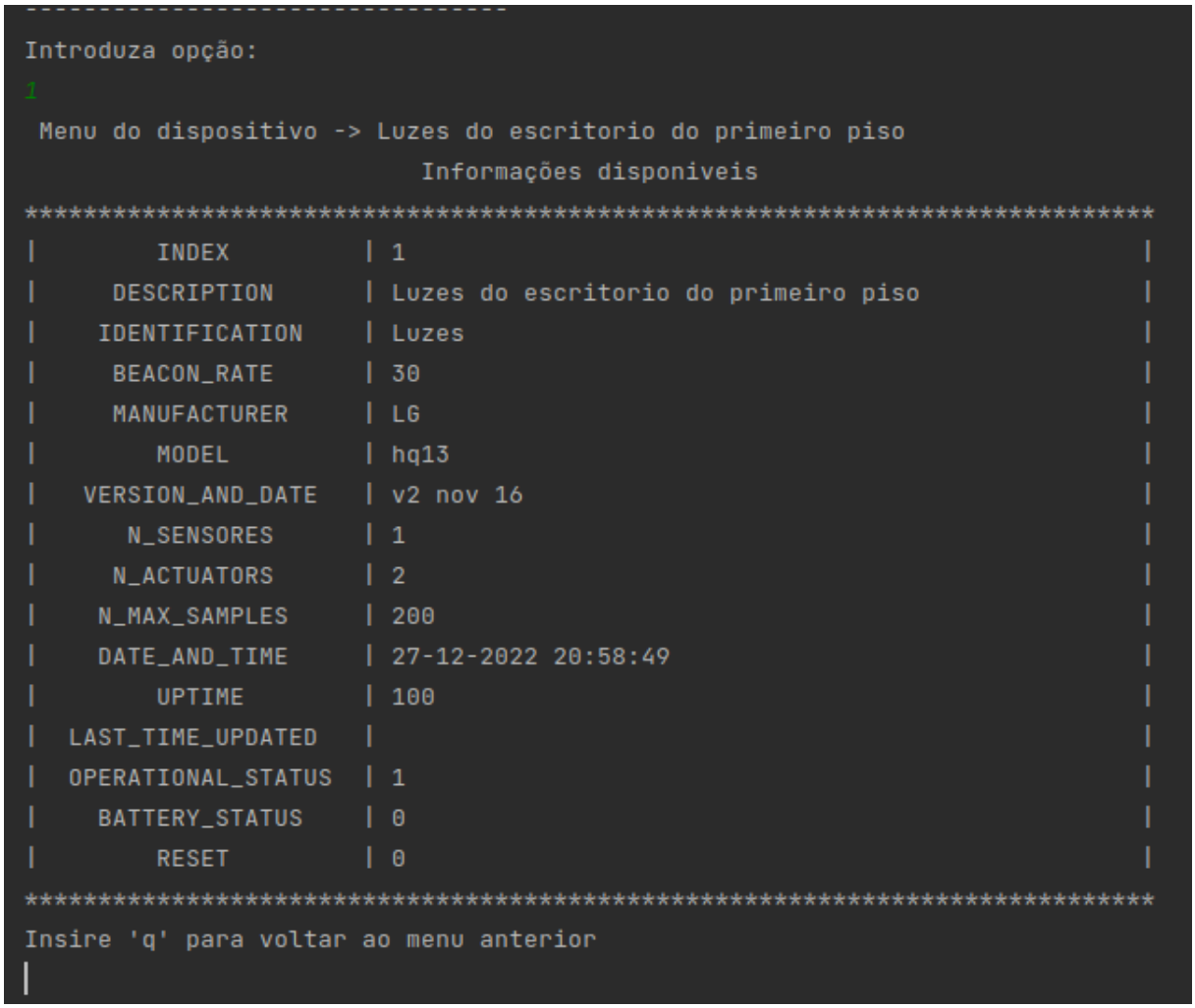


Figura 30: Tabela de informações das luzes

De salientar que neste exemplo o dispositivo está com as informações por defeito, ou seja, com as informações que a aplicação de simulação dos equipamentos introduz na **L-MIB** no arranque da sua execução.

Caso se pretenda obter o valor atual do sensor de luminosidade o utilizador deverá escolher a opção 2. A figura 31 apresenta um exemplo de teste da execução desta opção.

Figura 31: Informação sobre o último valor lido pelo sensor de luminosidade.

Caso se pretenda consultar o histórico de valores monitorizados o utilizador indica a usa 3 do menu anterior

(figura 29).

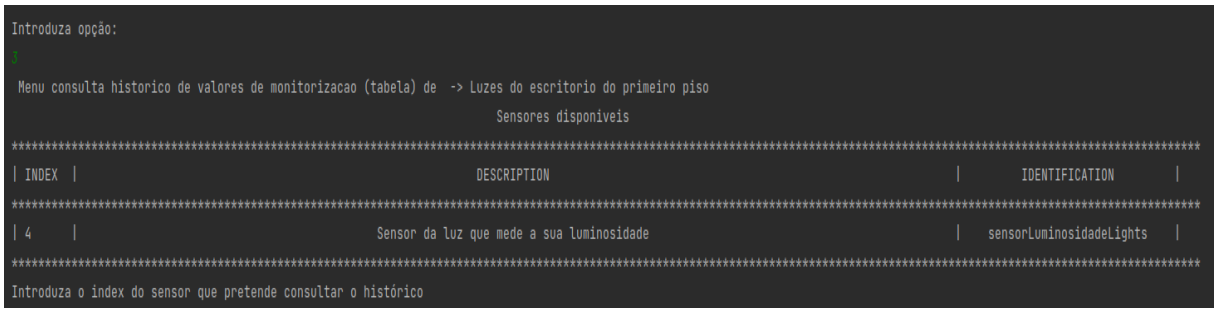


Figura 32: Lista de sensores das luzes disponíveis para consulta de histórico.

Como se pode observar na figura 32 após introduzida a opção 3, é apresentado ao utilizador uma tabela, neste caso, apenas com um sensor com vista a escolher qual sensor se pretende consultar o histórico de monitorização. Após selecionado o sensor pretendido no menu da figura 32 é apresentada uma tabela com as amostras medidas pelo sensor bem como a data e hora dessas amostras (figura 33).

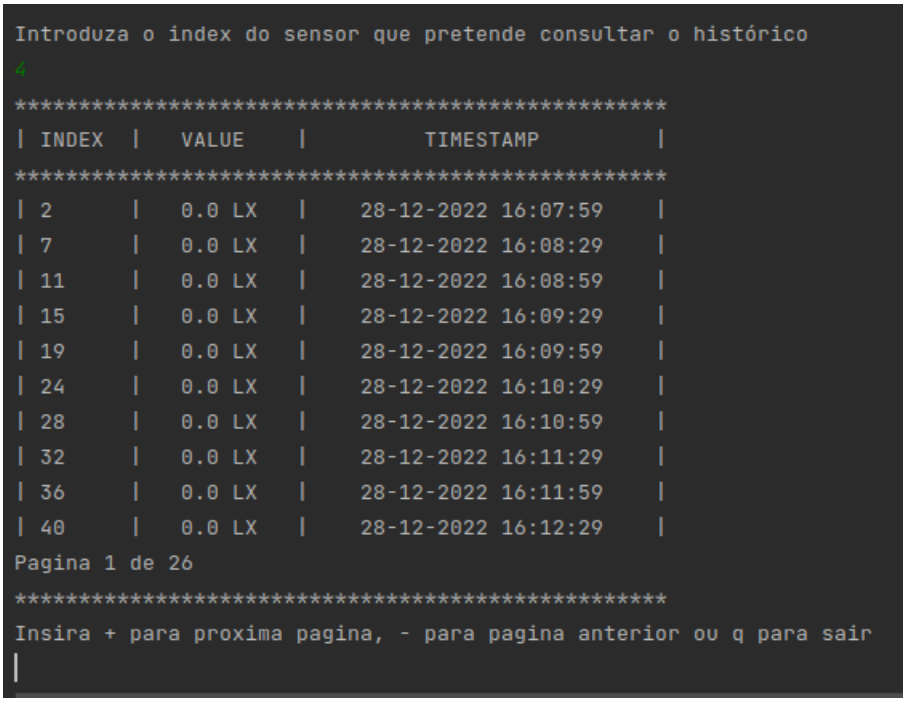


Figura 33: Histórico de amostras resultante da monitorização do sensor de luminosidade.

É possível observar que as amostras têm sempre o mesmo valor e que são geradas e efetuadas a cada 30 segundos.

Esta tabela pode ser dividida por várias páginas de informação, sendo possível avançar ou recuar.

Para ligar as luzes, ou seja, configurar um valor no atuador respetivo, o utilizador escolhe as opções desejadas dos menus respetivos (figuras 29 e 34).

No exemplo aqui apresentado, o utilizador liga apenas as luzes na sua intensidade máxima (figura 35)

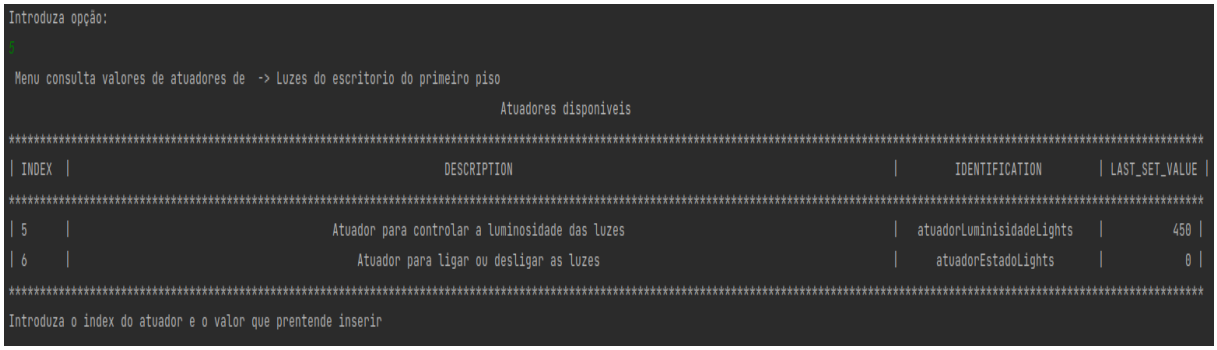


Figura 34: Lista de atuadores das luzes disponíveis

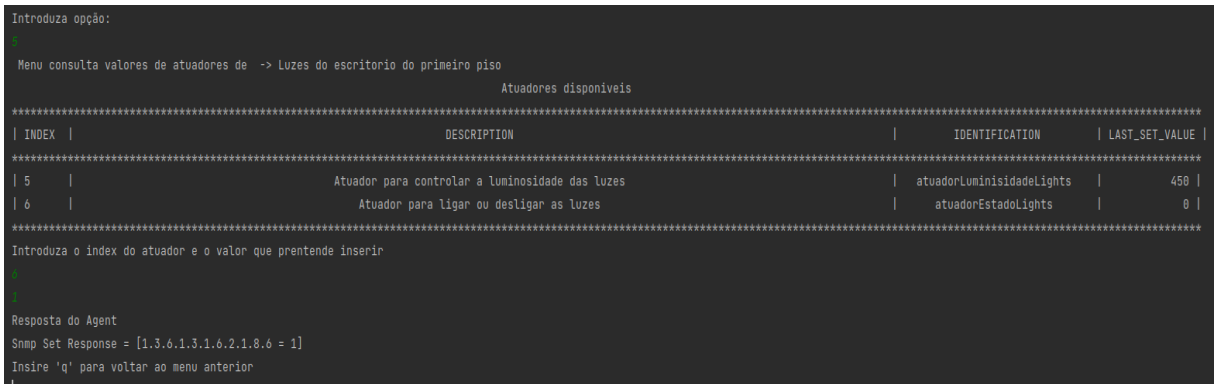


Figura 35: Escolha do atuador de estado das luzes e inserção do valor 1.

É possível verificar nos *logs* da aplicação que implementa os dispositivos a receção de uma mensagem do protocolo **L-SNMP** com uma primitiva do tipo *set-request* com a lista de identificadores e de valores respetivos (figura 36).

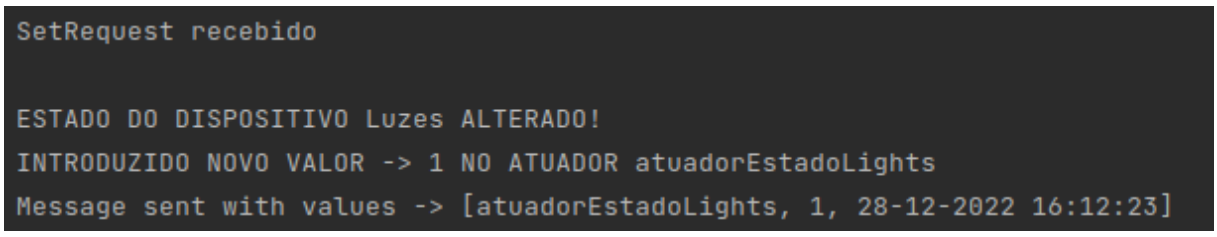


Figura 36: Informação de *log* da alteração do estado das luzes.

Se o pedido de configuração é bem sucedido é enviada uma mensagem **L-SNMP** tipo *response* com a identificação do atuador, o valor configurado e a data e hora da sua alteração. Esta mensagem, tal como explicado em secções anteriores, funciona como confirmação da configuração no dispositivo, permitindo assim ao agente **SNMP** alterar os valores na **MIB** respetiva e enviar uma mensagem de resposta **SNMP** para a aplicação controladora que, por sua vez, informa o utilizador da mudança efetuada.

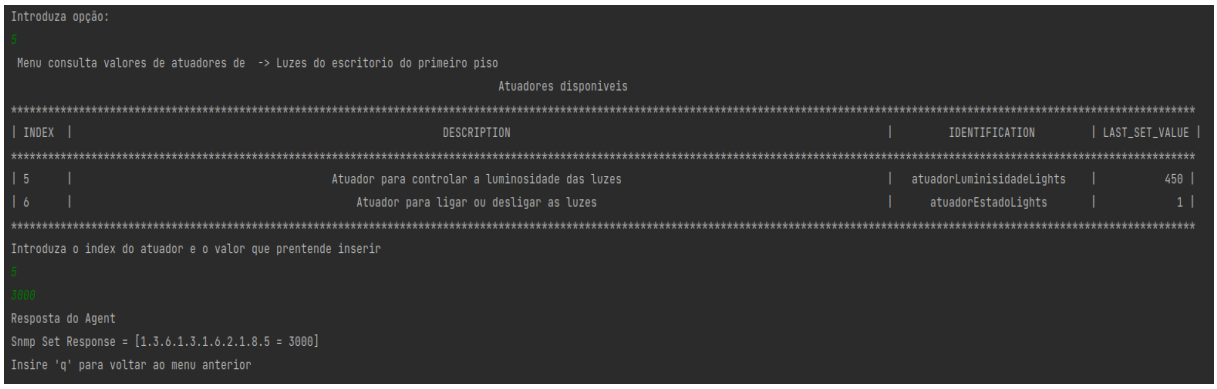


Figura 37: Configuração da intensidade luminosa para 300.0 lx

as figuras 37 a 42 mostram o resultado de efetuar a configuração da intensidade luminosa para 300.0 lx.

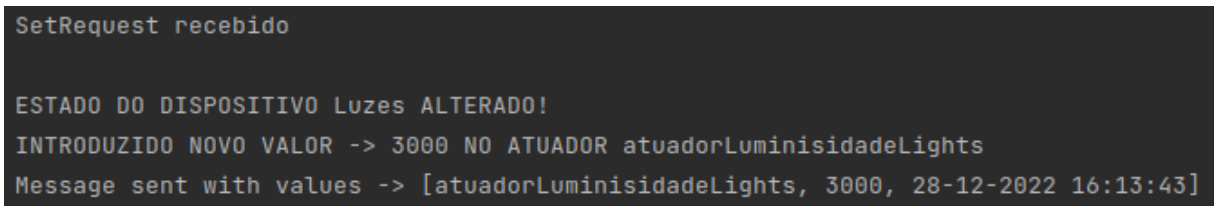


Figura 38: Informação da alteração da intensidade luminosa.

Em particular, a figura 42 mostra um gráfico do histórico do sensor de luminosidade, onde se pode ver claramente a modificação da luminosidade para o valor 300 lx.

Figura 39: Último valor lido pelo sensor de luminosidade.

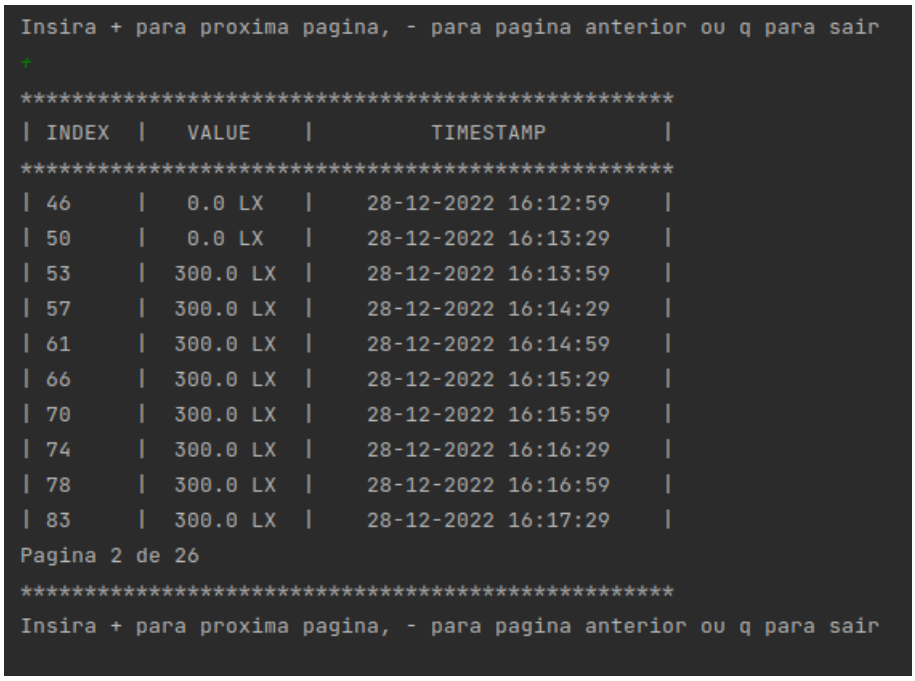


Figura 40: Tabela de histórico das amostras do sensor de luminosidade.

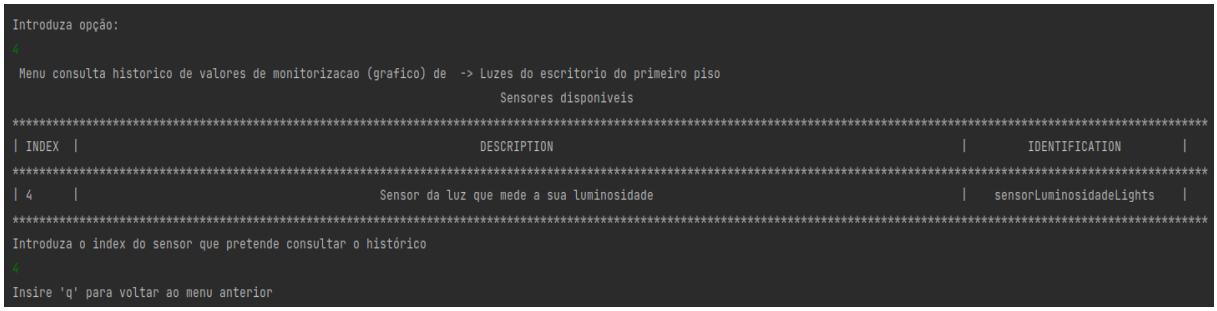


Figura 41: Lista de sensores disponíveis para consulta de histórico em gráfico.

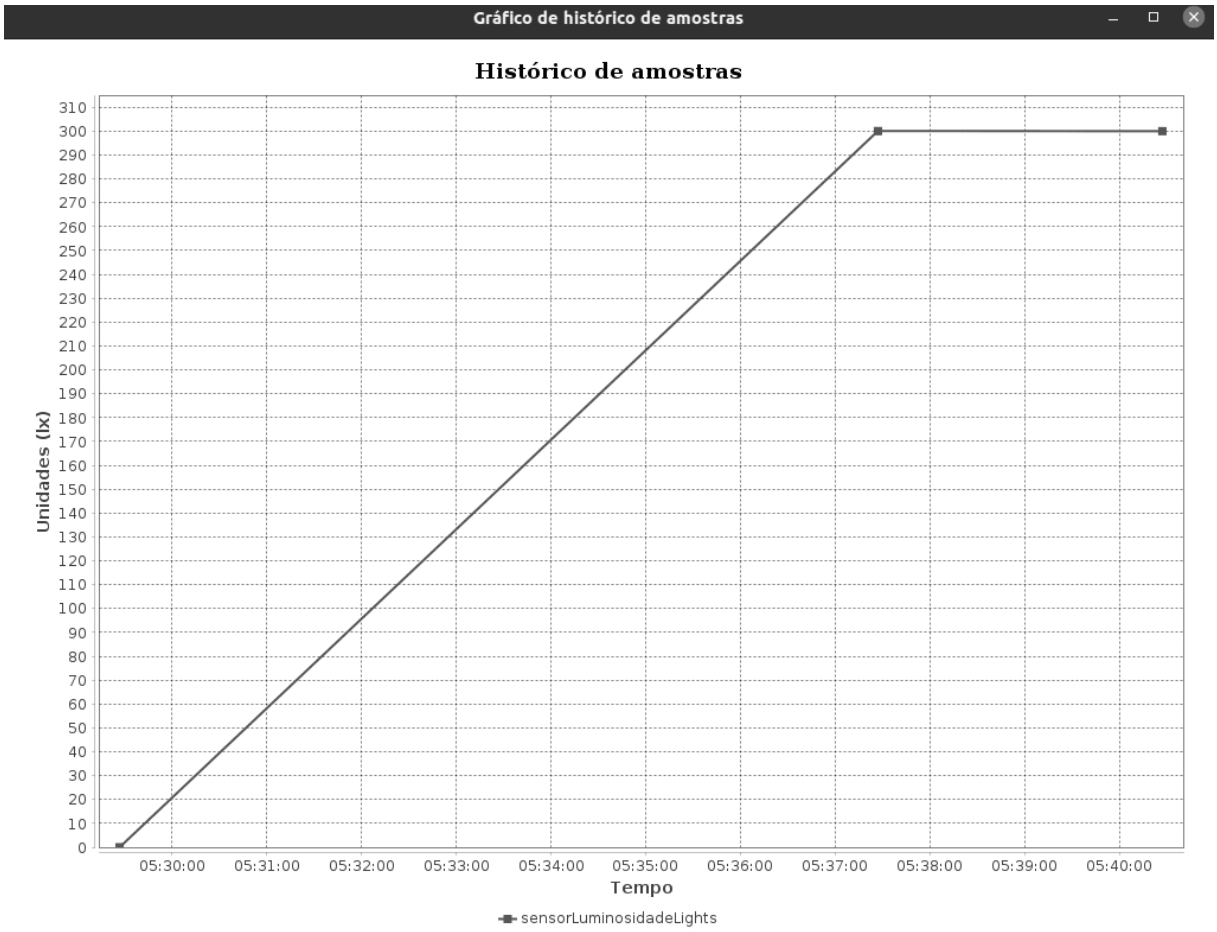


Figura 42: Histórico de amostras do sensor de luminosidade ao longo do tempo.

6.2 Controlo do estore

O estore funciona de forma muito semelhante às luzes havendo apenas uma modificação no tipo de sensores e atuadores. No entanto, a sequência de menus e as funcionalidades disponíveis são muito semelhantes. Fica apenas o conjunto de figuras (43 a 54) respetivas que ilustram o funcionamento do sistema nas mesmas situação de interação.

```
Introduza o INDEX do dispositivo a gerir ou exit para sair
2
Menu do dispositivo -> Estore do escritorio do primeiro piso
--------------------------------
1 - Informações sobre o dispositivo
2 - Consultar valores atuais de sensores
3 - Consultar histórico de valores de monitorizacao (tabela)
4 - Consultar histórico de valores de monitorizacao (grafico)
5 - Introduzir valor em atuadores
6 - Voltar
--------------------------------
Introduza opção:
```

Figura 43: Menu do equipamento Estore.

```
Introduza opção:
1
 Menu do dispositivo -> Estore do escritorio do primeiro piso
                        Informações disponiveis
*********************************************************************************
|        INDEX         | 2                                                      |
|     DESCRIPTION      | Estore do escritorio do primeiro piso                  |
|    IDENTIFICATION    | Estore                                                 |
|     BEACON_RATE      | 32                                                     |
|     MANUFACTURER     | Samsung                                                |
|        MODEL         | lp32                                                   |
|   VERSION_AND_DATE   | v4 dec 06                                              |
|      N_SENSORES      | 1                                                      |
|     N_ACTUATORS      | 2                                                      |
|    N_MAX_SAMPLES     | 200                                                    |
|    DATE_AND_TIME     | 27-12-2022 20:58:49                                    |
|        UPTIME        | 100                                                    |
|  LAST_TIME_UPDATED   |                                                        |
|  OPERATIONAL_STATUS  | 1                                                      |
|    BATTERY_STATUS    | 0                                                      |
|        RESET         | 0                                                      |
*********************************************************************************
Insire 'q' para voltar ao menu anterior
```

Figura 44: Tabela de informações do estore.

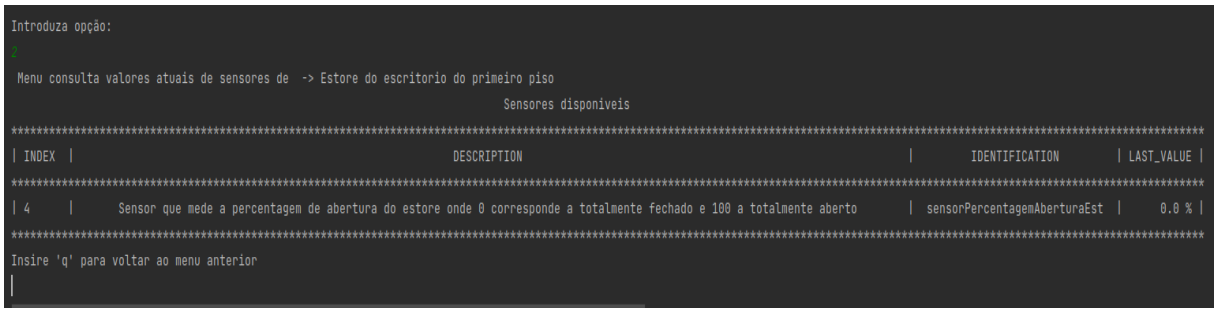


Figura 45: Informação sobre o último valor lido pelo sensor de abertura do estore.

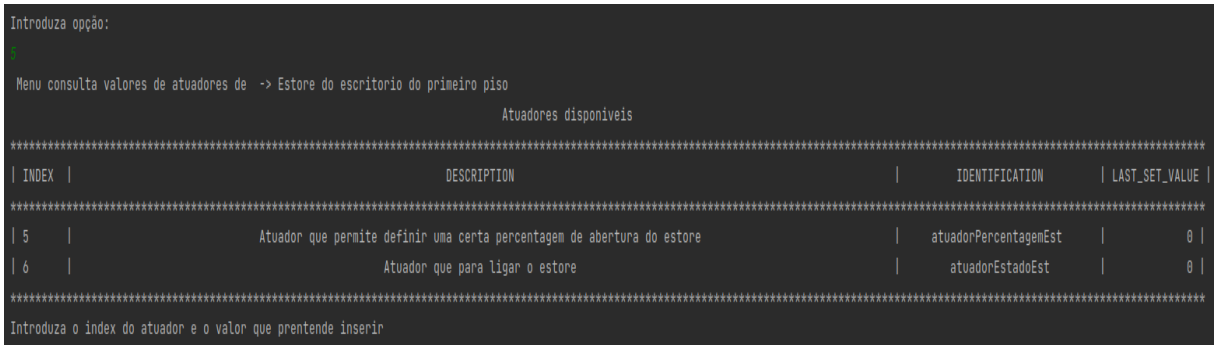


Figura 46: Lista de atuadores do estore.

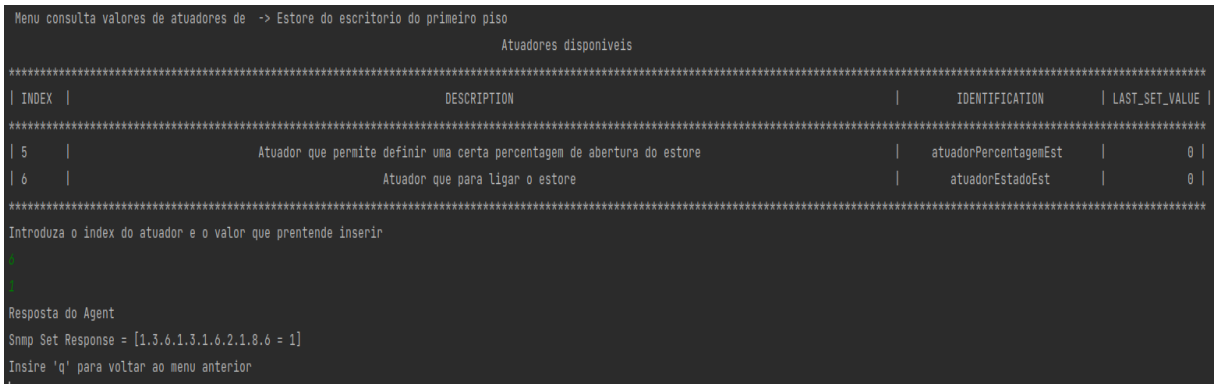


Figura 47: Abertura do estore.

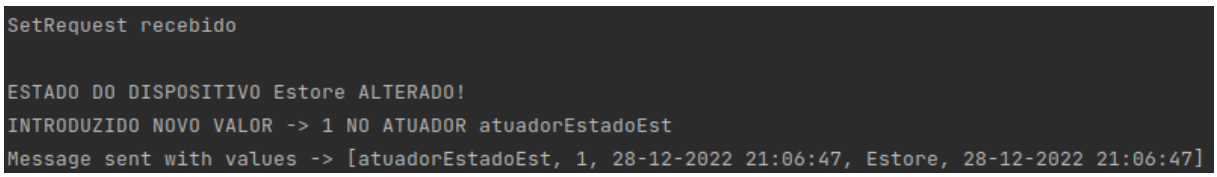


Figura 48: Informação nos *logs* do dispositivo.

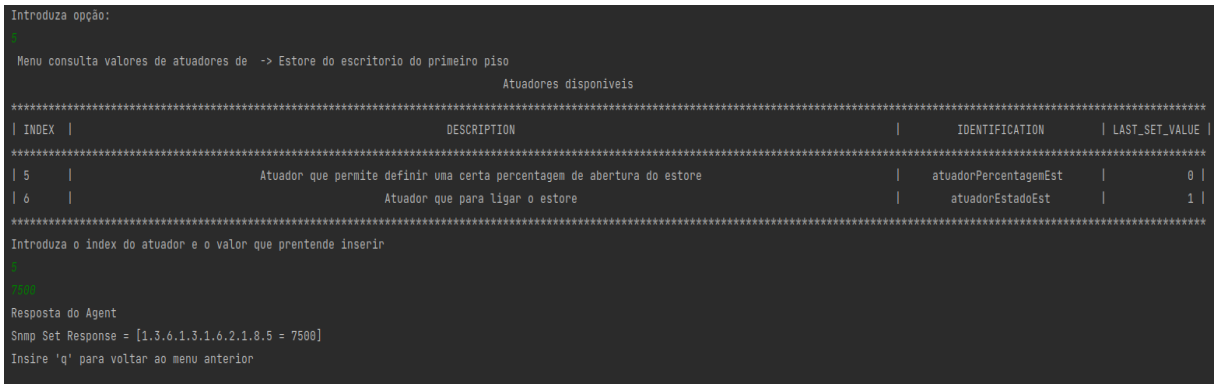


Figura 49: Abertura do estore a 75 % .

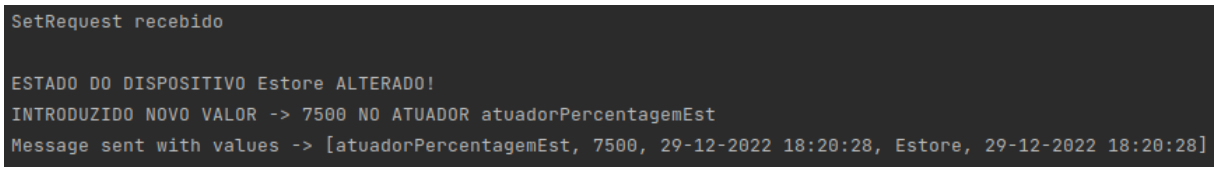


Figura 50: Informação nos *logs* do dispositivo.

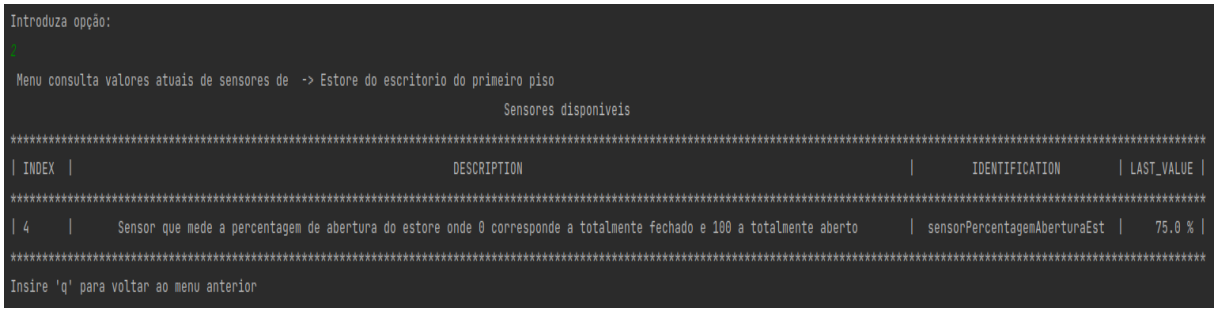


Figura 51: Último valor lido pelo sensor de abertura do estore.

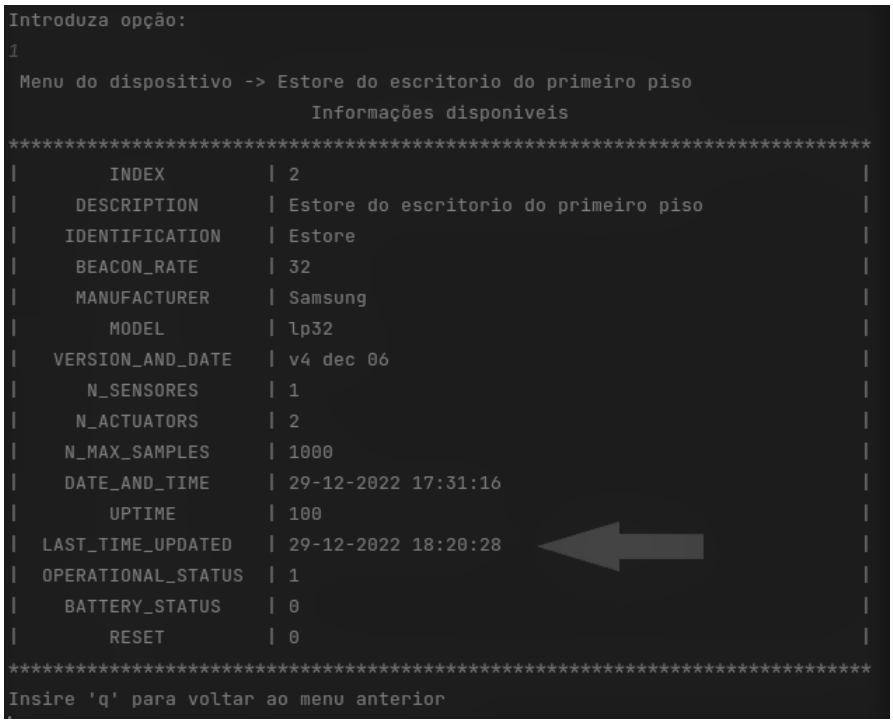


Figura 52: Tabela de informações do estore com atualização do campo *lastTimeUpdated*

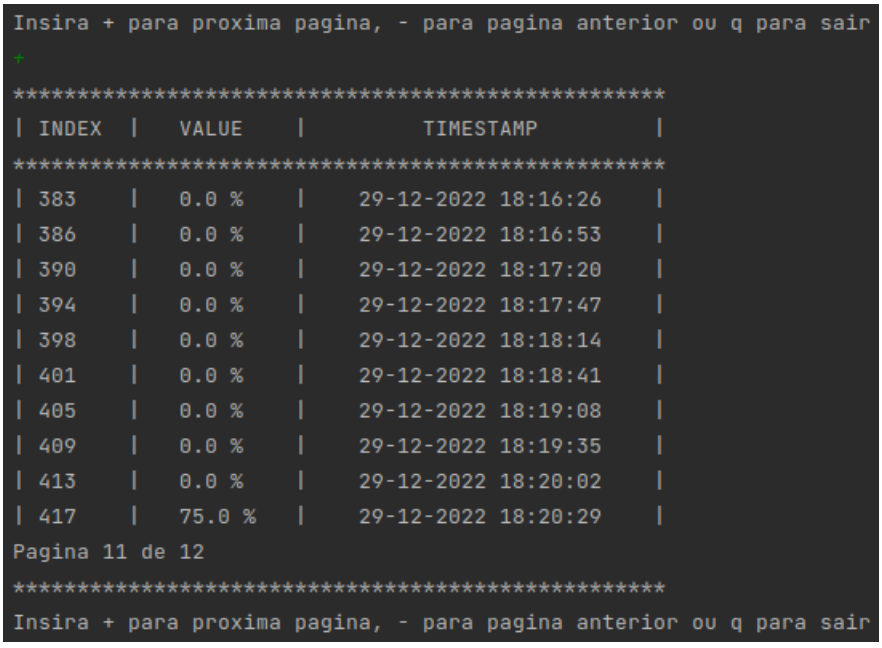


Figura 53: Tabela de monitorização do sensor do estore.

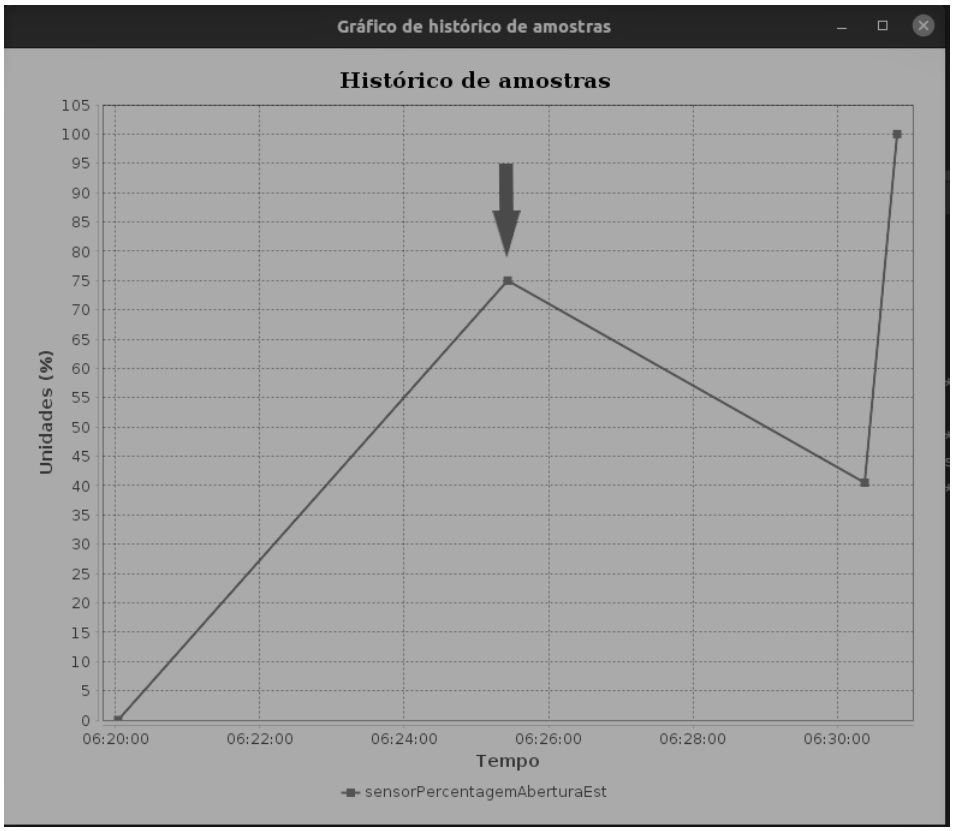


Figura 54: Gráfico do histórico de amostras do sensor de abertura do estore ao longo do tempo

6.3 Controlo do ar condicionado

O ar condicionado é o dispositivo mais complexo do protótipo desenvolvido devido ao número de sensores e atuadores associados ser maior e ao facto de que foi necessário simular a evolução dos valores lidos pelos sensores que são dinâmicos.

No entanto, o *interface* na aplicação gestora é em tudo semelhante ao dos outros dispositivos (ver sequência de figuras 55 a 67).

Para aceder ao menu do ar condicionado insere-se o índex 3 e é apresentado o menu tal com na figura 55.

```
Introduza o INDEX do dispositivo a gerir ou exit para sair
3
Menu do dispositivo -> Ar condicionado da sala de estar do primeiro piso
--------------------------------
1 - Informações sobre o dispositivo
2 - Consultar valores atuais de sensores
3 - Consultar histórico de valores de monitorizacao (tabela)
4 - Consultar histórico de valores de monitorizacao (grafico)
5 - Introduzir valor em atuadores
6 - Voltar
--------------------------------
Introduza opção:
```

Figura 55: Menu do equipamento ar condicionado.

Tal como nos outros dispositivos é possível consultar informações específicas do Ar Condicionado (figura 56) ou informações dos últimos valores lidos pelos seus sensores (figura 57) que apesar do dispositivo em si estar desligado, ou seja, não estar a climatizar, os seus sensores continuam a efetuar medições das condições climatéricas.

```
Introduza opção:
1
 Menu do dispositivo -> Ar condicionado da sala de estar do primeiro piso
                        Informações disponiveis
********************************************************************************
|        INDEX         | 3                                                     |
|     DESCRIPTION      | Ar condicionado da sala de estar do primeiro piso     |
|    IDENTIFICATION    | ArCondicionado                                        |
|     BEACON_RATE      | 25                                                    |
|     MANUFACTURER     | Bosch                                                 |
|        MODEL         | ac143                                                 |
|   VERSION_AND_DATE   | v1 oct 22                                             |
|      N_SENSORES      | 2                                                     |
|     N_ACTUATORS      | 2                                                     |
|    N_MAX_SAMPLES     | 1000                                                  |
|    DATE_AND_TIME     | 28-12-2022 19:59:35                                   |
|        UPTIME        | 100                                                   |
|  LAST_TIME_UPDATED   |                                                       |
|  OPERATIONAL_STATUS  | 1                                                     |
|    BATTERY_STATUS    | 0                                                     |
|        RESET         | 0                                                     |
********************************************************************************
Insire 'q' para voltar ao menu anterior
```

Figura 56: Tabela de informações do ar condicionado.

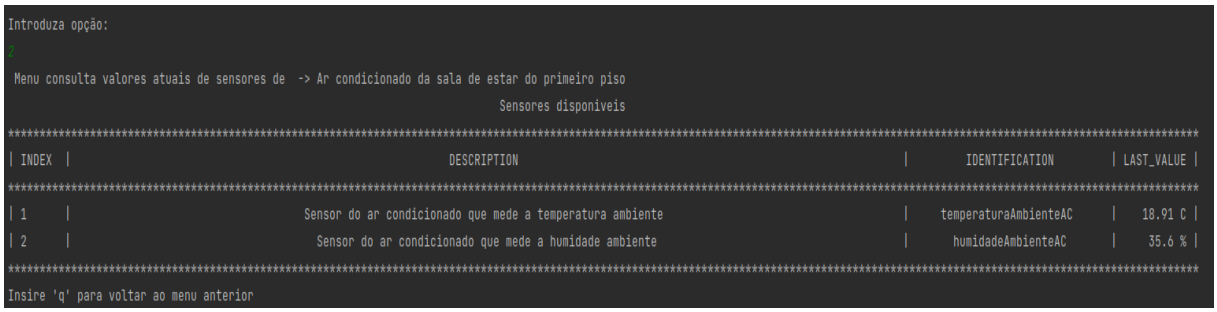


Figura 57: Informações sobre os últimos valores lidos pelos sensores do Ar condicionado

Na figura 58 é possível observar-se a primeira página da tabela e a variação dos valores monitorizados para o sensor de temperatura que vão variando de forma de ligeira.

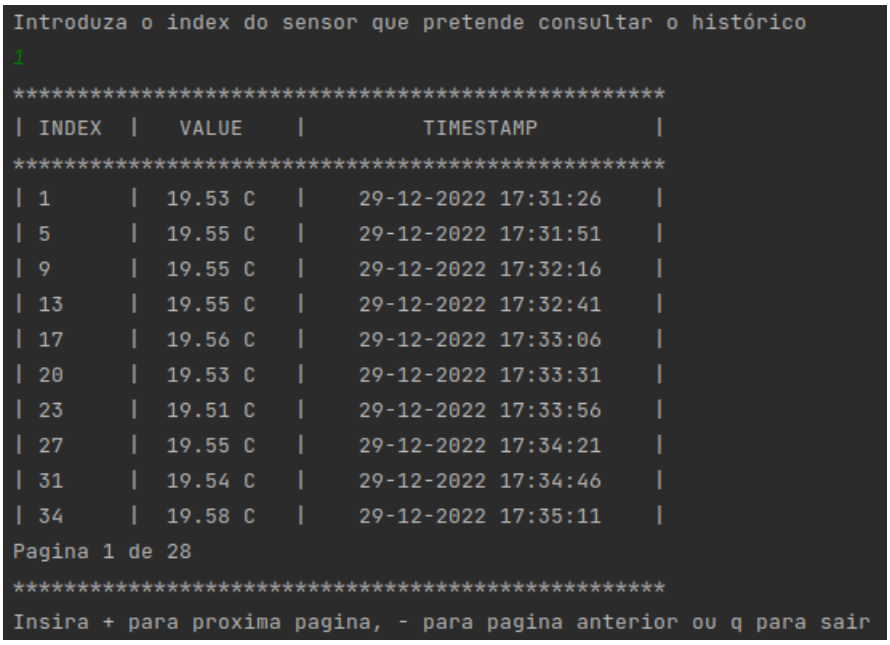


Figura 58: Tabela de informações de valores de monitorização do sensor de temperatura do A/C.

Para ser possível a inserção de um valor de temperatura para o qual o ar condicionado irá climatizar o espaço é necessário consultar primeiramente a sua lista de atuadores (figura 59). Em seguida identificar o atuador de estado e configurar o valor para ligar o dispositivo, tal como apresentado na figura 60.

Figura 59: Lista de atuadores do ar condicionado .

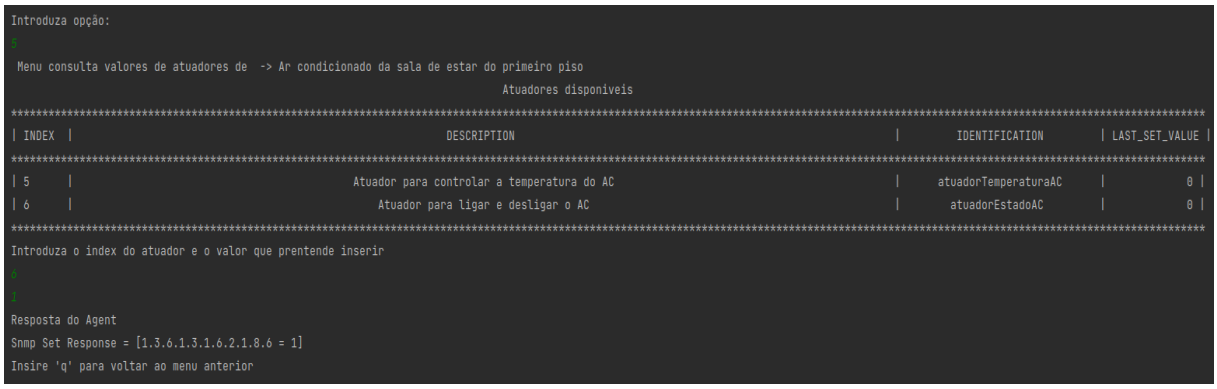


Figura 60: Ligar o A/C.

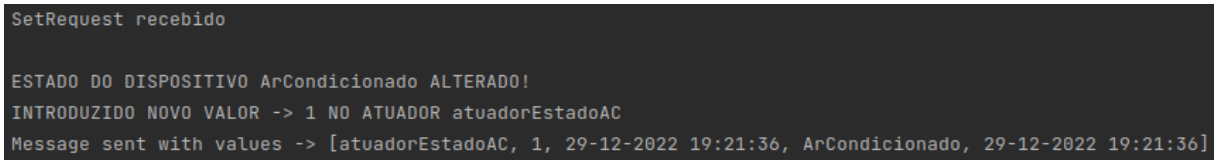


Figura 61: Informação dos *logs* no dispositivo.

Figura 62: Configuração da temperatura alvo do A/C a 22°C.

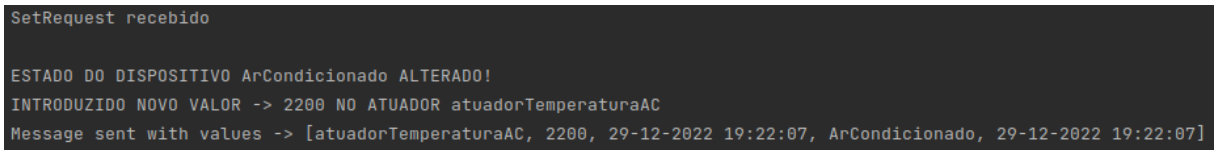


Figura 63: Informação dos *logs* do dispositivo da alteração da temperatura de climatização.

De salientar que o protótipo desenvolvido assume que a temperatura do local do ar condicionado muda instantaneamente para a temperatura introduzida e continua a simular a variação de valores assumindo como referência esse valor. O funcionamento explicado previamente é observável no último valor lido pelo sensor de temperatura ambiente presente na figura 64, onde se verifica que após a inserção do valor de **22.00 ºC**, a amostra seguinte que foi recolhida é de **22.05ºC**.

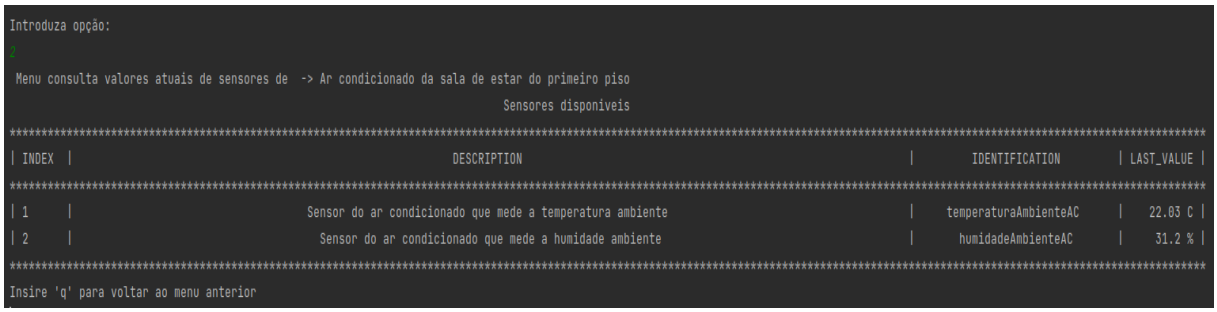


Figura 64: Últimos valores lidos pelos sensores do ar condicionado após inserção de nova temperatura.

Na figura 65 é possível observar o momento em que o valor foi configurado no atuador(campo *lastTimeUpdated*) e, na figura 66, onde se apresenta o histórico de valores de monitorização do sensor de temperatura sob a forma de gráfico, é também possível observar a mudança efetuada.

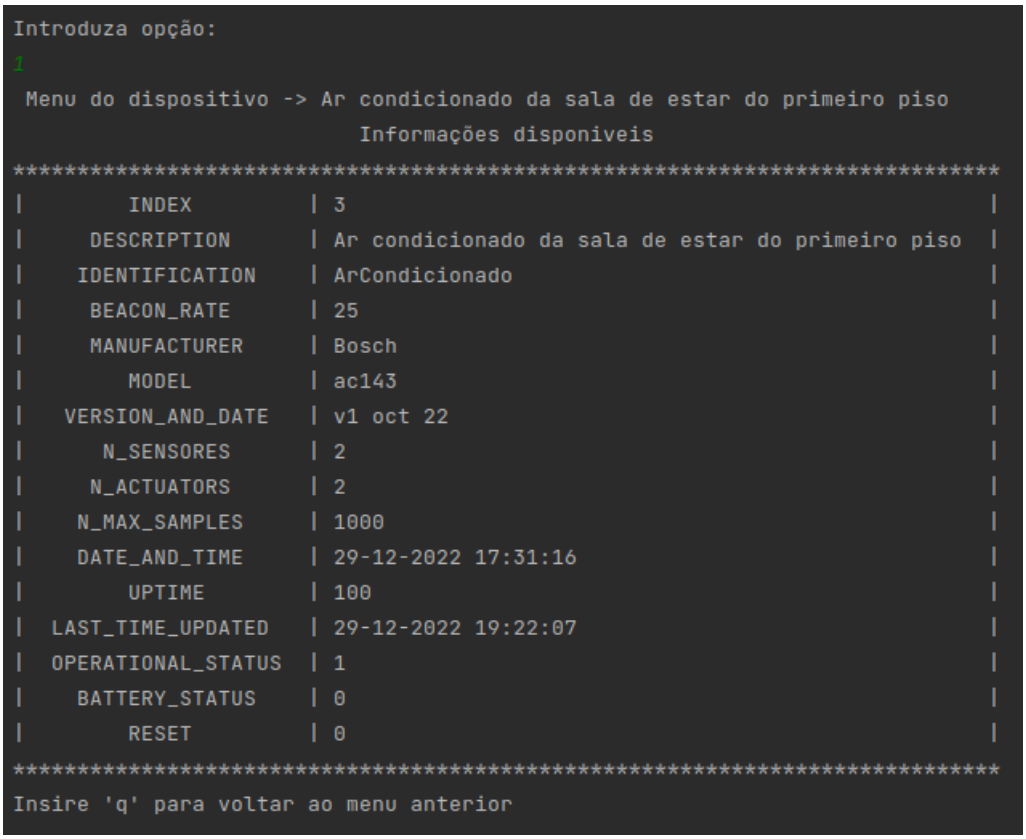


Figura 65: Tabela de informações do A/C com atualização do campo *lastTimeUpdated*

Figura 66: Histórico de amostras do sensor de temperatura ao longo do tempo.

Além da visualização do histórico do sensor de temperatura é também possível observar-se o histórico do outro sensor presente no ar condicionado, o sensor de humidade ambiente novamente sob a forma de gráfico, como apresentado na figura 67.

Figura 67: Histórico de amostras do sensor de humidade ao longo do tempo

6.4 Avaliação dos resultados obtidos

Após a realização dos testes com o protótipo desenvolvido, foi possível verificar que o agente **SNMP** manteve sempre a informação atualizada na sua **MIB** e em conformidade com os valores presentes na **L-MIB** dos dispositivos.

Relativamente ao tempo decorrido desde o instante em que uma alteração era introduzida até à sua atualização na **MIB**, pode-se considerar como negligenciável, quer nas comunicações por **SNMP** quer por **L-SNMP**. No entanto, deve ter-se em consideração que não foi testado um sistema em grande escala onde a possibilidade de algum tipo de latência pode não ser nula.

Sobre a aplicação de simulação de dispositivos, apesar da sua simplicidade, foi suficiente para a simulação efetiva e válida da funcionalidade deste tipo de sensores e atuadores, para o cálculo de estatísticas e para a validação da comunicação usando o **SNMP** e **L-SNMP**.

Relativamente à aplicação controladora pode-se referir que cumpriu os objetivos, tornando possível a monitorização e configuração dos equipamentos, permitindo a visualização dos dados da **MIB** do agente **SNMP** de forma imediata e transparente.

Capítulo 7

Conclusões e Trabalho Futuro

Nesta última secção é realizada uma análise ao trabalho efetuado durante a dissertação, explicando as principais dificuldades encontradas, as soluções adotadas e terminando com uma discussão sobre o trabalho futuro que poderá ser desenvolvido para melhorar a solução proposta.

7.1 Conclusões

A principal conclusão desta dissertação é que foi possível demonstrar a usabilidade do modelo de gestão **SNMP** no domínio dos sistemas domóticos. Além disso, para facilitar a sua integração e adoção pelos fabricantes de equipamentos domóticos foi desenvolvido um novo protocolo e um novo modelo de informação ainda mais simples de implementar que o **SNMP** tradicional. Assim, o **L-SNMP** acrescenta um nível intermédio de gestão, com requisitos tecnológicos menos exigentes, numa tentativa de disponibilizar uma arquitetura mais universal e aberta, baseada em normas bem estabelecidas para gestão de redes e serviços distribuídos na Internet.

Inicialmente os trabalhos da dissertação focaram-se no estudo das principais tecnologias domóticas existentes no mercado bem como na evolução dessas mesmas tecnologias ao longo do tempo. Desta forma foi possível analisar algumas das suas características diferenciadoras, o tipo de comunicação utilizada e as vantagens e desvantagens da sua utilização. Toda esta análise contribui para uma maior perceção do caminho percorrido nesta área e das necessidades ainda existentes para um crescimento ainda maior da área nos tempos atuais.

Pode ainda afirmar-se que, apesar da grande diversificação das tecnologias existentes, existe ainda um caminho a percorrer no que toca à interoperabilidade entre os dispositivos de tecnologias diferentes, o que serve de justificação principal para a solução aqui proposta.

Relativamente ao sistema desenvolvido, os resultados obtidos durante a fase de testes permitiram comprovar que a utilização dos protocolos **SNMP** e **L-SNMP** apresenta grandes potencialidades quando integrados num meio domótico. Apesar de se ter simulado um cenário completo em *software* sem recorrer a equipamentos reais, foi possível verificar que a arquitetura desenhada e implementada foi capaz de responder de forma positiva nos testes realizados. Foi possível confirmar que o **SNMP** e **L-SNMP** são independentes da tecnologia e dos equipamentos,

permitindo a sua aplicação em qualquer sistema domótico.

O protocolo **L-SNMP** e o modelo de informação correspondente, plasmado no caso concreto numa **L-MIB** para dispositivos domóticos, são a principal contribuição científica desta dissertação.

A utilização desta tecnologia de *middleware* potencia o desenvolvimento de sistemas de controlo mais universais e modulares, facilitando a conexão com sistemas inteligentes, com níveis de autonomia elevados. Outra influência que uma solução deste tipo pode trazer é a redução de custos no desenvolvimento de sistemas domóticos sobretudo nos módulos de controlo e nos meios de conexão dos equipamentos pois podem ser utilizadas tecnologias de meio físico mais comuns e universalmente já usadas para as a redes locais **IP**.

Como já foi previamente referido, o protótipo desenvolvido inclui um módulo que simulava os equipamentos físicos (luzes, estore, A/C). Este módulo implementa o protocolo **L-SNMP** e a instrumentação da *Domotics L-MIB*. Também o agente **SNMP** do protótipo implementa o protocolo **L-SNMP** para comunicação com o módulo anterior. De salientar que o âmbito da instrumentação da *Domotics L-MIB* é a gestão dos equipamentos geridos por um único dispositivo que, normalmente, serão produzidos e comercializados pelos fabricantes de equipamentos domóticos. Por outro lado, o âmbito da instrumentação da *Domotics MIB* é a gestão integrada de todos os dispositivos dum edifício inteligente. Neste caso, o desenvolvimento de sistemas de controlo domóticos poderá ser feito por empresas que não estão ligadas ao fabrico de equipamentos específicos dos sistemas domóticos. Este tipo de *software* pode ser desenvolvido e comercializado por empresas especialistas no desenvolvimento de *software* de controlo.

De notar que as próprias aplicações (*software* + *hardware*) de interface para os módulos de controlo que integram os agentes **SNMP** e **L-SNMP** podem ser modulares, de integração independente dos próprios módulos de controlo. Por exemplo, os próprios equipamentos de controlo domótico (comandos, interruptores físicos, etc.) podem ser geridos como recursos da *Domotics L-MIB* ou da *Domotics MIB*, desde que sejam abstraídos como grupos de sensores e atuadores.

É de salientar que a tecnologia **L-SNMP** pode ser muito útil noutros domínios fora da domótica, domínios em que o **SNMP** tradicional tem tido mais dificuldade em ser integrado. Exemplos relevantes são a gestão de sistemas de sensores e atuadores em veículos, a gestão de redes de sensores de larga escala, a gestão de componentes de fabrico industrial, a gestão de sistemas de geração de energia renovável, etc.

Por fim, deve indicar-se que o principal desejo que a adoção desta solução enfrenta é a dependência da vontade dos fabricantes de equipamentos domóticos aderirem a uma estratégia que os obriga a integrar o protocolo **L-SNMP** e a instrumentação da *Domotics L-MIB* nos seus dispositivos. Além disso, alguns dos principais fabricantes atuais podem não querer alterar o *status quo* do ramo e continuar a comercializar soluções fechadas completas, com tecnologias proprietárias em alguns dos módulos dos seus produtos.

A dominância do mercado por via do controlo das tecnologias usadas é uma estratégia antiga e que continua a ser usada na atualidade.

7.2 Trabalho Futuro

O sistema desenvolvido poderá ser complementado com novas funcionalidades que não foram consideradas, nesta altura, essenciais introduzir nos modelos de informação das **MIB** especificadas.

Seria também importante a realização dos testes integrando no protótipo dispositivos com equipamentos domóticos reais, aproximando ainda mais o protótipo das condições de usabilidade dos sistemas comerciais.

Por fim seria importante num trabalho futuro integrar tecnologias de inteligência artificial nos sistemas de controlo. Estes sistemas podiam interagir diretamente com os agentes através do **SNMP** ou através de tecnologias de *gateway* intermédio, tal como apresentado na figura 3.

Parte III

Bibliografia

Bibliografia

[1] Renato Nunes. Análise comparativa de tecnologias para domótica. *III Jornadas de Engenharia de Automação, Controlo e Instrumentação*, 2002.

[2] AGENTPP. . URL https://agentpp.com/agenpro4/AgenProManual.pdf. [Online], Acedido em Junho de 2022.

[3] Renato Nunes. Conceitos e temas associados aos edifícios inteligentes. 2002.

[4] Ricardo Lobão. Modelo simplificado de previsão do comportamento térmico de edifícios, 2007.

[5] RTA Real Time Automation. https://www.rtautomation.com/technologies/lonworks/ ,[Online], Acedido em novembro de 2021.

[6] Insteon. URL https://www.insteon.com/technology/ourtechnology. [Online], Acedido em Novembro 2021.

[7] João Gonçalves. Protocolos de automação doméstica – solução de automação residencial e vigilância baseada em protocolo z-wave, 2017. http://hdl.handle.net/10400.22/11724.

[8] Pierre GUILLEMIN. The european home systems protocol-concepts and product. *http://www.domotics.com/homesys/HSpapers/EHSproto.htm*, 1996. URL https://cir.nii.ac.jp/crid/1572543025131615744.

[9] P Loisel and S Quellec. Simplification of breeding management by the automatic transfer of data between piggery equipment and technical economic management. 1999.

[10] knx.org. URL https://www.knx.org/wAssets/docs/downloads/Marketing/Flyers/KNX-Basics/KNX-Basics_en.pdf. [Online], Acedido em novembro de 2021.

[11] Z-WAVE ALLIANCE. Our history. 2017. Disponível em https://z-wavealliance.org/z-wave_alliance_history.

[12] Thread Group. URL https://www.threadgroup.org/What-is-Thread/Thread-Benefits. [Online], Acedido em novembro de 2021.

[13] Raphael J Cohn, RJ Coppen, Andrew Banks, and Rahul Gupta. Mqtt version 3.1, 2014.

[14] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, June 2014. URL https://www.rfc-editor.org/info/rfc7252.

[15] Sherali Zeadally, Farhan Siddiqui, and Zubair Baig. 25 years of bluetooth technology. *Future Internet*, 11, 2019. ISSN 19995903. doi: 10.3390/fi11090194.

[16] THE IOT PAD. URL https://theiotpad.com/tips/home-automation-protocols. [Online], Acedido em novembro de 2021.

[17] Bluetooth. URL https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. [Online], Acedido em novembro de 2021.

[18] Khusvinder Gill, Shuang-Hua Yang, Fang Yao, and Xin Lu. A zigbee-based home automation system. *IEEE Transactions on Consumer Electronics*, 55(2):422–430, 2009. doi: 10.1109/TCE.2009.5174403.

[19] Douglas Mauro and Kevin Schmidt. *Essential SNMP: Help for System and Network Administrators*. "O'Reilly Media, Inc.", 2005.

[20] Dr. Jeff D. Case, Keith McCloghrie, Dr. Marshall T. Rose, and Steven Waldbusser. Introduction to Community-based SNMPv2. RFC 1901, January 1996. URL https://www.rfc-editor.org/info/rfc1901.

[21] Dr. Jeff D. Case, David Partain, Russ Mundy, and Bob Stewart. Introduction to Version 3 of the Internet-standard Network Management Framework. RFC 2570, April 1999. URL https://www.rfc-editor.org/info/rfc2570.

[22] Randy Presuhn. Transport Mappings for the Simple Network Management Protocol (SNMP). RFC 3417, December 2002. URL https://www.rfc-editor.org/info/rfc3417.

[23] Mark Fedor, Martin Lee Schoffstall, James R. Davin, and Dr. Jeff D. Case. Simple Network Management Protocol (SNMP). RFC 1157, May 1990. URL https://www.rfc-editor.org/info/rfc1157.

[24] Dr. Marshall T. Rose and Keith McCloghrie. Management Information Base for network management of TCP/IP-based internets. RFC 1156, May 1990. URL https://www.rfc-editor.org/info/rfc1156.

[25] Randy Presuhn. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). RFC 3416, December 2002. URL https://www.rfc-editor.org/info/rfc3416.

[26] Dr. Marshall T. Rose and Keith McCloghrie. Structure and identification of management information for TCP/IP-based internets. RFC 1155, May 1990. URL https://www.rfc-editor.org/info/rfc1155.

[27] Keith McCloghrie, Jürgen Schönwälder, David T. Perkins, and Keith McCloghrie. Structure of Management Information Version 2 (SMIv2). RFC 2578, April 1999. URL https://www.rfc-editor.org/info/rfc2578.

[28] B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems and their Applications*, 14(1):20–26, 1999. doi: 10.1109/5254.747902.

[29] JE López De Vergara, Víctor A Villagrá, Juan I Asensio, and Julio Berrocal. Ontologies: Giving semantics to network management models. *IEEE network*, 17(3):15–21, 2003.

[30] Christian Reinisch, Wolfgang Granzer, Fritz Praus, and Wolfgang Kastner. Integration of heterogeneous building automation systems using ontologies. In *2008 34th Annual Conference of IEEE Industrial Electronics*, pages 2736–2741. IEEE, 2008.

[31] Dieter Fensel, Frank Van Harmelen, Ian Horrocks, Deborah L McGuinness, and Peter F Patel-Schneider. Oil: An ontology infrastructure for the semantic web. *IEEE intelligent systems*, 16(2):38–45, 2001.

[32] V Maniraj and R Sivakumar. Ontology languages-a review. *International Journal of Computer Theory and Engineering*, 2(6):887, 2010.

[33] Choi Deokjai, Hongseok Jang, Kugsang Jeong, Punghyeok Kim, and S.H. Kim. Delivery and storage architecture for sensed information using snmp. pages 582–585, 09 2006. doi: 10.1007/11876601_71.

[34] Nikolay Kakanakov, Elena Dimitrova, Grisha Kostadinova, and Grisha Spasov. Using snmp for remote measurement and automation. 09 2007.

[35] Mihajlo Savić. Bridging the snmp gap: Simple network monitoring the internet of things. *Facta universitatis - series: Electronics and Energetics*, 29:475–487, 01 2016. doi: 10.2298/FUEE1603475S.

[36] Jingjing Xu, Yann-Hang Lee, Wei-Tek Tsai, Wu Li, Young-Sung Son, Jun-Hee Park, and Kyung-Duk Moon. Ontology-based smart home solution and service composition. In *2009 International Conference on Embedded Software and Systems*, pages 297–304. IEEE, 2009.

[37] André Silva. Gestão de dispositivos de domótica por snmp, 2010.

[38] Ran Giladi. Snmp for home automation. *International Journal of Network Management*, 14(4):231–239, 2004.

[39] Notation One. Information technology–asn. 1 encoding rules: Specification of basic encoding rules (ber), canonical encoding rules (cer) and distinguished encoding rules (der). *Interfaces*, 10(20-X):49, 1998.

[40] SNMP4J. URL https://www.snmp4j.org/. [Online], Acedido em Junho de 2022.

[41] AGENTPP. . URL https://agentpp.com/help/mds/5.0.1/index.html#t=MIBDesigner2%2FMIBDesigner2Title%2FMIBDesigner2Title.htm. [Online], Acedido em Junho de 2022.

[42] AGENTPP. . URL http://oosnmp.com/help/agp/5.0.0/index.html#t=AgenProManual%2FAgenProManualTitle%2FAgenProManualTitle.htm. [Online], Acedido em Junho de 2022.

Parte IV
Apêndices

Apêndice A
Anexos

A.1 Planeamento das atividades

O **Planeamento** do trabalho a alcançar foi, naturalmente, a primeira fase da realização da dissertação, onde se definiram os objetivos a alcançar e um esquema temporal das várias fases de desenvolvimento.

No trabalho de **Pesquisa** foi adquirido um conhecimento mais detalhado às tecnologias atuais mais usadas nos sistemas comerciais do ramo. Para finalizar esta fase, procedeu-se também a uma pesquisa de trabalhos académicos ou propostas científicas relacionados e foi feita uma análise crítica das limitações das soluções mais usadas atualmente por forma a que se pudesse definir uma nova solução baseada em tecnologia normalizada.

Em seguida, na fase de **Especificação**, foi definida a arquitetura da solução, tendo sido prestada especial atenção à definição dos modelos de organização de informação, as **MIB** (*Domotics MIB e L-MIB*). A especificação da arquitetura incluiu a definição do novo protocolo **L-SNMP**.

Na fase da **Implementação** o primeiro passo foi a geração do esqueleto do **SNMP** e a sua implementação da *Domotics MIB*. Posteriormente foram desenvolvidas as funcionalidades de instrumentação dos agente **SNMP**. Em simultâneo procedeu-se à criação de um módulo que simulasse um dispositivo domótico (incluindo sensores e atuadores). Este módulo inclui um agente **L-SNMP** que implementa a *Domotics MIB*. Por fim, foi desenvolvida uma pequena aplicação cliente (gestor **SNMP**) capaz de controlar o sistema.

Após concluídas todas as etapas anteriores procedeu-se a um conjunto de testes do sistema com o protótipo desenvolvido.

Tarefa	2021 Out	Nov	Dez	2022 Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Dez	2023 Jan	Fev	Mar	Abr
1. Background e **EA**	•	•	•															
2. Preparação do **RPD**		•	•	•														
3. Definição e especificação da solução				•	•	•	•	•	•	•	•	•	•					
4. Desenvolvimento do sistema protótipo										•	•	•	•					
5. Testes e análise de resultados												•	•	•	•			
6. Escrita								•	•	•	•	•	•	•	•	•	•	•

Tabela 4: Plano de atividades.

O trabalho desenvolvido, foi dividido em diversas fases com o objetivo de delinear o melhor caminho para atingir os objetivos propostos bem como uma gestão eficaz do tempo disponível. A figura 68 inclui um esquema com a descrição das várias atividades realizadas ao longo da realização da dissertação, agrupadas em fases temporais.

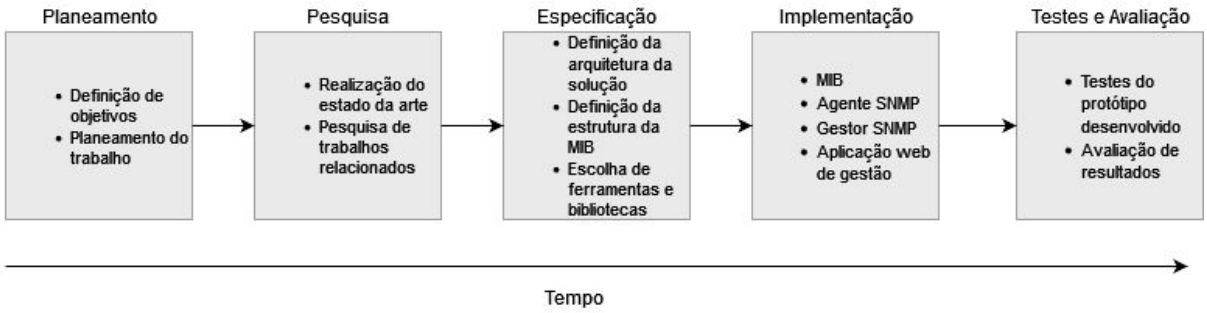


Figura 68: Fases de desenvolvimento dos trabalhos da dissertação.

A.2 DOMOTICS MIB

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

IMPORTS
enterprises,
experimental,
MODULE-IDENTITY,
OBJECT-TYPE,
Integer32
FROM SNMPv2-SMI
OBJECT-GROUP
FROM SNMPv2-CONF;

domoticsMIB MODULE-IDENTITY
LAST-UPDATED "202207112010Z" -- Jul 11, 2022, 8:10:00 PM
ORGANIZATION "DomoticaSNMP"
CONTACT-INFO
"Gonçalo Nogueira
email: goncalonogueira21@outlook.pt"
DESCRIPTION
"MIB for Domotic system"
-- 1.3.6.1.4.1.1 --
-- 1.3.6.1.3.1
::= { experimental 1 }


-- groups --

-- (1) This group contains objets for different informations about the  --
-- characteristics of the system itself. --

domoSystem OBJECT IDENTIFIER
-- 1.3.6.1.3.1.1
::= { domoticsMIB 1 }
```

```
-- (2) This group includes objets to define characteristics of a device in the --
-- domotic system. --

domoDevice OBJECT IDENTIFIER
-- 1.3.6.1.3.1.2
::= { domoticsMIB 2 }



-- (3) This group includes objets with specific specifications mainly about devices --
-- or modules.  --

domoSpecification OBJECT IDENTIFIER
-- 1.3.6.1.3.1.3
::= { domoticsMIB 3 }



-- (4) This group associates a specification to a given device. --

domoDeviceSpecification OBJECT IDENTIFIER
-- 1.3.6.1.3.1.4
::= { domoticsMIB 4 }



-- (5) This group includes objets to define characteristics of a given sensor. --

domoSensor OBJECT IDENTIFIER
-- 1.3.6.1.3.1.5
::= { domoticsMIB 5 }



 -- (6) This group includes objets to define characteristics of a given actuator. --

domoActuator OBJECT IDENTIFIER
-- 1.3.6.1.3.1.6
::= { domoticsMIB 6 }



-- (7) This group associates a sensor to a given device. --

domoDeviceSensor OBJECT IDENTIFIER
-- 1.3.6.1.3.1.7
::= { domoticsMIB 7 }
```

```
-- (8) This group associates an actuator to a given device. --

domoDeviceActuator OBJECT IDENTIFIER
-- 1.3.6.1.3.1.8
::= { domoticsMIB 8 }



-- (9) This group contains objets to characterize informations about a measured --
-- sensor's value that's been monitored . --

domoMonitoring OBJECT IDENTIFIER
-- 1.3.6.1.3.1.9
::= { domoticsMIB 9 }



 -- (10) This group contains objets to characterize informations about a value --
-- inserted to an actuator. --

domoConfiguring OBJECT IDENTIFIER
-- 1.3.6.1.3.1.10
::= { domoticsMIB 10 }



-- (11) This group includes objets for defining informations about an available --
-- functionality of a device. --

domoFunctionalities OBJECT IDENTIFIER
-- 1.3.6.1.3.1.11
::= { domoticsMIB 11 }



-- (12) This group associates a functionality to a given device. --

domoDeviceFunctionality OBJECT IDENTIFIER
-- 1.3.6.1.3.1.12
::= { domoticsMIB 12 }



-- (13) This group includes objets for defining characteristics about a module --
-- in the system.  --

domoModules OBJECT IDENTIFIER
-- 1.3.6.1.3.1.13
::= { domoticsMIB 13 }
```

```
-- (14) This group associates a specification to a given module. --

domoModuleSpecification OBJECT IDENTIFIER
-- 1.3.6.1.3.1.14
::= { domoticsMIB 14 }



-- (15) This group associates a device to a given module. --

domoModuleDevice OBJECT IDENTIFIER
-- 1.3.6.1.3.1.15
::= { domoticsMIB 15 }



-- (16) This group is required to ensure the correct declaration of the MIB. --
-- It contains all the objets defined. --

domoticaSNMPConf OBJECT IDENTIFIER
-- 1.3.6.1.3.1.16
::= { domoticsMIB 16 }



-- system (1)********************************************************* --
-- This group contains objets for different informations about the  --
-- characteristics of the system itself such as contact of the responsable, --
 -- location, etc. --


domoSystemDescription OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..500))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Textual description of the entity. Includes information such as name and
version of software used, operating system, date and time"
-- 1.3.6.1.4.1.1.1.5 --
-- 1.3.6.1.3.1.1.1
::= { domoSystem 1 }


domoSystemContact OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Text description of the person's contact for this management node. "
-- 1.3.6.1.3.1.1.2
```

```
::= { domoSystem 2 }


domoSystemName OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Textual description with the name of the system."
-- 1.3.6.1.3.1.1.3
::= { domoSystem 3 }


domoSystemLocation OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Physical location of this node. If location is unknown, field appears
empty."
-- 1.3.6.1.3.1.1.4
::= { domoSystem 4 }


domoSystemType OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A simple text description of the type of the system where the SNMP
agent is deployed. For example, 'Software Module on a dedicated server
for domotic services' or 'Software Module on a Gateway Device to an AI
System'"
-- 1.3.6.1.3.1.1.5
::= { domoSystem 5 }


-- domoDevice (2)******************************************************** --
-- This group includes objets to define characteristics of a device in the --
-- domotic system. --


domoDeviceNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Total number of existing devices in the system."
-- 1.3.6.1.3.1.2.1
::= { domoDevice 1 }
```

```
domoDeviceTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DeviceEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"A list with the set of devices that are part of the system, corresponding
an entry to each device."
-- 1.3.6.1.3.1.2.2
::= { domoDevice 2 }


domoDeviceEntry OBJECT-TYPE
SYNTAX  DomoDeviceEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"An entry that contains management information applicable to each device
in particular."
INDEX {
domoDeviceIndex }
-- 1.3.6.1.3.1.2.2.1
::= { domoDeviceTable 1 }


DomoDeviceEntry ::= SEQUENCE {

domoDeviceIndex             Integer32,
domoDeviceDescription       OCTET STRING,
domoDeviceIdentification    OCTET STRING,
domoDeviceBeaconRate        Integer32,
domoDeviceManufacturer      OCTET STRING,
domoDeviceModel             OCTET STRING,
domoDeviceVersionAndDate    OCTET STRING,
domoDeviceNSensors          Integer32,
domoDeviceNActuators        Integer32,
domoDeviceNMaxSamples       Integer32,
domoDeviceDateAndTime       OCTET STRING,
domoDeviceUpTime            Integer32,
domoDeviceLastTimeUpdated   OCTET STRING,
domoDeviceOperationalStatus Integer32,
domoDeviceBatteryStatus     Integer32,
domoDeviceReset             Integer32}


domoDeviceIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each new device. The values
```

```
assigned must vary between 1 and N devices.
"
-- 1.3.6.1.3.1.2.2.1.1
::= { domoDeviceEntry 1 }


domoDeviceDescription OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Text description of the device. It could indicate what's the use of the device and where it
-- 1.3.6.1.3.1.2.2.1.2
::= { domoDeviceEntry 2 }


domoDeviceIdentification OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Unique device identification. This identification may be a name or
 a code assigned to the device."
-- 1.3.6.1.3.1.2.2.1.3
::= { domoDeviceEntry 3 }


domoDeviceBeaconRate OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-write
STATUS  current
DESCRIPTION
"Objet that defines the frequency rate value of a notification message.
This message acts as a beacon message in the system and that's why the
name of the objet."
-- 1.3.6.1.3.1.2.2.1.4
::= { domoDeviceEntry 4 }


domoDeviceManufacturer OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the device manufacturer information. Most commonly just the
name of the manufacturer is represented."
-- 1.3.6.1.3.1.2.2.1.5
::= { domoDeviceEntry 5 }


domoDeviceModel OBJECT-TYPE
```

```
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the device model information. Like the manufacturer objet,
just the name of the model is usually represented."
-- 1.3.6.1.3.1.2.2.1.6
::= { domoDeviceEntry 6 }


domoDeviceVersionAndDate OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..55))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the device version information and the date of that version."
-- 1.3.6.1.3.1.2.2.1.7
::= { domoDeviceEntry 7 }


domoDeviceNSensors OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of sensors associated
with a device."
-- 1.3.6.1.3.1.2.2.1.8
::= { domoDeviceEntry 8 }


domoDeviceNActuators OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of actuators associated
with a device."
-- 1.3.6.1.3.1.2.2.1.9
::= { domoDeviceEntry 9 }


domoDeviceNMaxSamples OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the maximum number of samples that
each associated sensor to this device can store."
-- 1.3.6.1.3.1.2.2.1.10
::= { domoDeviceEntry 10 }
```

```
domoDeviceDateAndTime OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..55))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the date and time where the device was
stored in the MIB"
-- 1.3.6.1.3.1.2.2.1.11
::= { domoDeviceEntry 11 }


domoDeviceUptime OBJECT-TYPE
SYNTAX  Integer32 (0..100)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the percentage of time the device was
available."
-- 1.3.6.1.3.1.2.2.1.12
::= { domoDeviceEntry 12 }


domoDeviceLastTimeUpdated OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..55))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the date and time of the last update of
any information of the device."
-- 1.3.6.1.3.1.2.2.1.13
::= { domoDeviceEntry 13 }


domoDeviceOperationalStatus OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the operational state of the device,
where the value 0 corresponds to standby state, 1 corresponds to normal
state and 2 to eco-mode."
-- 1.3.6.1.3.1.2.2.1.14
::= { domoDeviceEntry 14 }


domoDeviceBatteryStatus OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
```

```
DESCRIPTION
"Objet defining the information about the device's battery status,
ranging from 1 to 100. If the device is not powered by battery,
the value must be 0."
-- 1.3.6.1.3.1.2.2.1.15
::= { domoDeviceEntry 15 }


domoDeviceReset OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Object defining the information to operate different types of device
reset. The value 0 and the default value, 1 corresponds to device shutdown,
2 to a hard reset and 3 to a soft reset."
-- 1.3.6.1.3.1.2.2.1.16
::= { domoDeviceEntry 16 }


-- domoSpecification (3)****************************************************** --
-- This group includes objets with specific specifications mainly about devices --
-- or modules. For example, the type of connection of a device. --


domoSpecificationNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the number of existing specifications in this system."
-- 1.3.6.1.3.1.3.1
::= { domoSpecification 1 }


domoSpecificationTable OBJECT-TYPE
SYNTAX  SEQUENCE OF SpecificationEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"A list with the set of specifications of devices or modules that are part
of the system, each entry corresponding to a specification."
-- 1.3.6.1.3.1.3.2
::= { domoSpecification 2 }


domoSpecificationEntry OBJECT-TYPE
SYNTAX  DomoSpecificationEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
```

```
"An entry that contains management information applicable to each
specification in particular."
INDEX {
specificationIndex }
-- 1.3.6.1.3.1.3.2.1
::= { domoSpecificationTable 1 }


DomoSpecificationEntry ::= SEQUENCE {

domoSpecificationIndex       Integer32,
domoSpecificationDescription OCTET STRING,
domoSpecificationValue       OCTET STRING }


domoSpecificationIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each new specification. The
values assigned must vary between 1 and N specifications."
-- 1.3.6.1.3.1.3.2.1.1
::= { domoSpecificationEntry 1 }


domoSpecificationDescription OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Text description of the specification."
-- 1.3.6.1.3.1.3.2.1.2
::= { domoSpecificationEntry 2 }


domoSpecificationValue OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Value that a specification can represent. If the specification doesn't
require a value to be defined, in this objet should appear NULL"
-- 1.3.6.1.3.1.3.2.1.3
::= { domoSpecificationEntry 3 }


-- domoDeviceSpecification (4)****************************************** --
-- This group associates a specification to a given device. --
```

```
domoDeviceSpecificationNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Number of existing relationships between devices and specifications.
These relationships allow you to associate certain specifications with devices."
-- 1.3.6.1.3.1.4.1
::= { domoDeviceSpecification 1 }


domoDeviceSpecificationTable OBJECT-TYPE
SYNTAX  SEQUENCE OF domoDeviceSpecificationEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Table with the various relationships between devices and specifications
existing."
-- 1.3.6.1.3.1.4.2
::= { domoDeviceSpecification 2 }


domoDeviceSpecificationEntry OBJECT-TYPE
SYNTAX  DomoDeviceSpecificationEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION ""
REFERENCE
"Each entry contains information about each relationship between the devices and
specifications."
INDEX {
domoDeviceSpecificationIndex }
-- 1.3.6.1.3.1.4.2.1
::= { domoDeviceSpecificationTable 1 }


DomoDeviceSpecificationEntry ::= SEQUENCE {

domoDeviceSpecificationIndex            Integer32,
domoDeviceSpecificationIdxDevice        Integer32,
domoDeviceSpecificationIdxSpecification Integer32 }


domoDeviceSpecificationIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each relationship between device and
specification. The assigned values must vary between 1 and N relations."
-- 1.3.6.1.3.1.4.2.1.1
```

```
::= { domoDeviceSpecificationEntry 1 }


domoDeviceSpecificationIdxDevice OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index value of a device."
-- 1.3.6.1.3.1.4.2.1.2
::= { domoDeviceSpecificationEntry 2 }


domoDeviceSpecificationIdxSpecification OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about an index of the SPECIFICATION
table where the specification referenced in this table is present."
-- 1.3.6.1.3.1.4.2.1.3
::= { domoDeviceSpecificationEntry 3 }


-- domoSensor (5)****************************************************** --
-- This group includes objets to define characteristics of a given sensor. --


domoSensorNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the total number of sensors
present in the system."
-- 1.3.6.1.3.1.5.1
::= { domoSensor 1 }


domoSensorTable OBJECT-TYPE
SYNTAX  SEQUENCE OF domoSensorEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"A list with the set of sensors that are part of the system,
corresponding an entry to each sensor."
-- 1.3.6.1.3.1.5.2
::= { domoSensor 2 }


domoSensorEntry OBJECT-TYPE
```

```
SYNTAX  DomoSensorEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"An entry that contains management information applicable to each sensor
in particular."
INDEX {
domoSensorIndex }
-- 1.3.6.1.3.1.5.2.1
::= { domoSensorTable 1 }


DomoSensorEntry ::= SEQUENCE {

domoSensorIndex           Integer32,
domoSensorName            OCTET STRING,
domoSensorDescripton      OCTET STRING,
domoSensorUnits           OCTET STRING,
domoSensorScale           Integer32,
domoSensorSampleType      Integer32,
domoSensorMinValue        OCTET STRING,
domoSensorMaxValue        OCTET STRING,
domoSensorSamplingRate    Integer32,
domoSensorNSamples        Integer32,
domoSensorLastValue       Integer32,
domoSensorLastSampleIndex Integer32,
domoSensorOldestSampIndex Integer32,
domoSensorStatsInterval   Integer32,
domoSensorMedian          Integer32 }


domoSensorIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each new sensor. The values
assigned must vary between 1 and N sensors."
-- 1.3.6.1.3.1.5.2.1.1
::= { domoSensorEntry 1 }


domoSensorName OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the name of the sensor."
-- 1.3.6.1.3.1.5.2.1.2
::= { domoSensorEntry 2 }
```

```
domoSensorDescripton OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Text description of the sensor. Informations such as location can be disclosed here. "
-- 1.3.6.1.3.1.5.2.1.3
::= { domoSensorEntry 3 }


domoSensorUnits OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the units measured by the
sensor."
-- 1.3.6.1.3.1.5.2.1.4
::= { domoSensorEntry 4 }


domoSensorScale OBJECT-TYPE
SYNTAX  Integer32 (-65535..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of decimal places
that the sensor value has. For instance, the value -1 corresponds
to multiplying the value by 0.1, the value 1 to multiply by 10."
-- 1.3.6.1.3.1.5.2.1.5
::= { domoSensorEntry 5 }


domoSensorSampleType OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about a number that identifies the
type of value that the sensor measures. the value 1 corresponds to an
integer."
-- 1.3.6.1.3.1.5.2.1.6
::= { domoSensorEntry 6 }


domoSensorMinValue OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
```

```
"Objet defining the information about the minimum value that the sensor
can read."
-- 1.3.6.1.3.1.5.2.1.7
::= { domoSensorEntry 7 }


domoSensorMaxValue OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the maximum value that the sensor
can read."
-- 1.3.6.1.3.1.5.2.1.8
::= { domoSensorEntry 8 }


domoSensorSamplingRate OBJECT-TYPE
SYNTAX  Integer32 (0..65535 | 0)
MAX-ACCESS read-write
STATUS  current
DESCRIPTION
"Objet defining the information about the frequency rate value in seconds
of sampling by the sensor."
-- 1.3.6.1.3.1.5.2.1.9
::= { domoSensorEntry 9 }


domoSensorNSamples OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of samples that
the sensor has collected so far."
-- 1.3.6.1.3.1.5.2.1.10
::= { domoSensorEntry 10 }


domoSensorLastValue OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the last value measured by the
sensor."
-- 1.3.6.1.3.1.5.2.1.11
::= { domoSensorEntry 11 }


domoSensorLastSampleIndex OBJECT-TYPE
```

```
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the identifier index of the
last sample registered by the sensor."
-- 1.3.6.1.3.1.5.2.1.12
::= { domoSensorEntry 12 }


domoSensorOldestSampleIndex OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the identifier index of the
oldest sample registered by the sensor."
-- 1.3.6.1.3.1.5.2.1.13
::= { domoSensorEntry 13 }


domoSensorStatsInterval OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the interval of time that it takes
between each calculation of statistics by the sensor. The unit of this value
is seconds."
-- 1.3.6.1.3.1.5.2.1.14
::= { domoSensorEntry 14 }


domoSensorMedian OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the median of the sensor samples."
-- 1.3.6.1.3.1.5.2.1.15
::= { domoSensorEntry 15 }


-- domoActuator (6)************************************************** --
-- This group includes objets to define characteristics of a given actuator. --


domoActuatorNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
```

```
DESCRIPTION
"Objet defining the information about the total number of actuators
present in the system."
-- 1.3.6.1.3.1.6.1
::= { domoActuator 1 }


domoActuatorTable OBJECT-TYPE
SYNTAX  SEQUENCE OF ActuatorEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"A list with the set of actuators that are part of the system,
corresponding an entry to each actuator."
-- 1.3.6.1.3.1.6.2
::= { domoActuator 2 }


domoActuatorEntry OBJECT-TYPE
SYNTAX  DomoActuatorEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"An entry that contains management information applicable to each actuator
in particular."
INDEX {
domoActuatorIndex }
-- 1.3.6.1.3.1.6.2.1
::= { domoActuatorTable 1 }


DomoActuatorEntry ::= SEQUENCE {

domoActuatorIndex              Integer32,
domoActuatorName               OCTET STRING,
domoActuatorDescripton         OCTET STRING,
domoActuatorUnits              OCTET STRING,
domoActuatorScale              Integer32,
domoActuatorMinValue           OCTET STRING,
domoActuatorMaxValue           OCTET STRING,
domoActuatorLastSetValue       Integer32,
domoActuatorRelatedSensorName  OCTET STRING }


domoActuatorIndex OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each new actuator. The values
assigned must vary between 1 and N actuators."
```

```
-- 1.3.6.1.3.1.6.2.1.1
::= { domoActuatorEntry 1 }


domoActuatorName OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the name of the actuator."
-- 1.3.6.1.3.1.6.2.1.2
::= { domoActuatorEntry 2 }


domoActuatorDescripton OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Text description of the actuator. Informations such as location can be disclosed here."
-- 1.3.6.1.3.1.6.2.1.3
::= { domoActuatorEntry 3 }


domoActuatorUnits OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the units measured by the
actuator."
-- 1.3.6.1.3.1.6.2.1.4
::= { domoActuatorEntry 4 }


domoActuatorScale OBJECT-TYPE
SYNTAX  Integer32 (-65535..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of decimal places
that the actuator value has. For instance, the value -1 corresponds
to multiplying the value by 0.1, the value 1 to multiply by 10."
-- 1.3.6.1.3.1.6.2.1.5
::= { domoActuatorEntry 5 }


domoActuatorMinValue OBJECT-TYPE
SYNTAX  OCTET STRING(SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
```

```
DESCRIPTION
"Objet defining the information about the minimum value that the actuator
can apply."
-- 1.3.6.1.3.1.6.2.1.6
::= { domoActuatorEntry 6 }


domoActuatorMaxValue OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..45))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the maximum value that the actuator
can apply."
-- 1.3.6.1.3.1.6.2.1.7
::= { domoActuatorEntry 7 }


domoActuatorLastSetValue OBJECT-TYPE
SYNTAX  Integer32 (-65535..65535)
MAX-ACCESS read-write
STATUS  current
DESCRIPTION
"Objet defining the information about the last value inserted in
the actuator."
-- 1.3.6.1.3.1.6.2.1.5
::= { domoActuatorEntry 8 }


domoActuatorRelatedSensorName OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the name of the sensor that is
related to the actuator. If there is no related sensor, the value of the
objet should be empty."
-- 1.3.6.1.3.1.6.2.1.9
::= { domoActuatorEntry 9 }


-- domoDeviceSensor (7)************************************************** --
-- This group associates a sensor to a given device. --


domoDeviceSensorNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of existing relationships
```

```
between devices and sensors. These relationships make it possible
to associate certain sensors with devices."
-- 1.3.6.1.3.1.7.1
::= { domoDeviceSensor 1 }


domoDeviceSensorTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DeviceSensorEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various
relationships between devices and sensors."
-- 1.3.6.1.3.1.7.2
::= { domoDeviceSensor 2 }


domoDeviceSensorEntry OBJECT-TYPE
SYNTAX  DomoDeviceSensorEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Each entry contains information about each relationship between
the devices and sensors."
INDEX {
domoDeviceSensorIndex }
-- 1.3.6.1.3.1.7.2.1
::= { domoDeviceSensorTable 1 }


DomoDeviceSensorEntry ::= SEQUENCE {

domoDeviceSensorIndex     INTEGER,
domoDeviceSensorIdxDevice INTEGER,
domoDeviceSensorIdxSensor INTEGER }


domoDeviceSensorIndex OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each relationship between device and
sensors. The assigned values must vary between 1 and N relations."
-- 1.3.6.1.3.1.7.2.1.1
::= { domoDeviceSensorEntry 1 }


domoDeviceSensorIdxDevice OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS read-only
```

```
STATUS  current
DESCRIPTION
"Objet defining the information about the index value of the device that will be associated
-- 1.3.6.1.3.1.7.2.1.2
::= { domoDeviceSensorEntry 2 }


domoDeviceSensorIdxSensor OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the sensor index value to be
associated with the device."
-- 1.3.6.1.3.1.7.2.1.3
::= { domoDeviceSensorEntry 3 }


-- domoDeviceActuator (8)********************************************** --
-- This group associates an actuator to a given device. --


domoDeviceActuatorNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Number of existing relationships between devices and actuators.
These relationships make it possible to associate certain actuators with
devices."
-- 1.3.6.1.3.1.8.1
::= { domoDeviceActuator 1 }


domoDeviceActuatorTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DeviceActuatorEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various
relationships between devices and actuators."
-- 1.3.6.1.3.1.8.2
::= { domoDeviceActuator 2 }


domoDeviceActuatorEntry OBJECT-TYPE
SYNTAX  DeviceActuatorEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Each entry contains information about each relationship between
```

```
the devices and actuators."
INDEX {
domoDeviceActuatorIndex }
-- 1.3.6.1.3.1.8.2.1
::= { domoDeviceActuatorTable 1 }


DeviceActuatorEntry ::= SEQUENCE {

domoDeviceActuatorIndex       INTEGER,
domoDeviceActuatorDevice      INTEGER,
domoDeviceActuatorIdxActuator INTEGER }


domoDeviceActuatorIndex OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each relationship between device and
actuators. The assigned values must vary between 1 and N relations."
-- 1.3.6.1.3.1.8.2.1.1
::= { domoDeviceActuatorEntry 1 }


domoDeviceActuatorDevice OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
Objet defining the information about the index value of the device that
 will be associated to the actuator."
-- 1.3.6.1.3.1.8.2.1.2
::= { domoDeviceActuatorEntry 2 }


domoDeviceActuatorIdxActuator OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the actuator index value to be
associated with the device."
-- 1.3.6.1.3.1.8.2.1.3
::= { domoDeviceActuatorEntry 3 }


-- domoMonitoring (9)********************************************** --
-- This group contains objets to characterize informations about a measured --
-- sensor's value that's been monitored in the system. --
```

```
domoMonitoringNumber OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of relationships
between devices and sensors being monitored."
-- 1.3.6.1.3.1.9.1
::= { domoMonitoring 1 }


domoMonitoringTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DomoMonitoringEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various
relationships between devices and sensors monitored."
-- 1.3.6.1.3.1.9.2
::= { domoMonitoring 2 }


domoMonitoringEntry OBJECT-TYPE
SYNTAX  DomoMonitoringEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Each entry contains information about each relationship between the devices
and monitored sensors or actuators."
INDEX {
domoMonitoringIndex }
-- 1.3.6.1.3.1.9.2.1
::= { domoMonitoringTable 1 }


DomoMonitoringEntry ::= SEQUENCE {

domoMonitoringIndex     Integer32,
domoMonitoringIdxSensor Integer32,
domoMonitoringValue     Integer32,
domoMonitoringSizeValue Integer32,
domoMonitoringTimeStamp OCTET STRING }


domoMonitoringIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each relationship between device
```

```
and monitored sensors. The assigned values should vary between
1 and N relations."
-- 1.3.6.1.3.1.9.2.1.1
::= { domoMonitoringEntry 1 }


domoMonitoringIdxSensor OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index value of a sensor."
-- 1.3.6.1.3.1.9.2.1.2
::= { domoMonitoringEntry 2 }


domoMonitoringValue OBJECT-TYPE
SYNTAX  Integer32 (-65535..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the value beeing monitored."
-- 1.3.6.1.3.1.9.2.1.3
::= { domoMonitoringEntry 3 }


domoMonitoringSizeValue OBJECT-TYPE
SYNTAX  Integer32 (-65535..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the size of the value beeing
monitored."
-- 1.3.6.1.3.1.9.2.1.4
::= { domoMonitoringEntry 4 }


domoMonitoringTimeStamp OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the date and time when the
value was monitored."
-- 1.3.6.1.3.1.9.2.1.5
::= { domoMonitoringEntry 5 }


-- domoConfiguring (10)********************************************** --
-- This group contains objets to characterize informations about a value --
-- inserted to an actuator. --
```

```
domoConfiguringNumber OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of relationships between
configured devices and actuators."
-- 1.3.6.1.3.1.10.1
::= { domoConfiguring 1 }


domoConfiguringTable OBJECT-TYPE
SYNTAX  SEQUENCE OF ConfiguringEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various relationships
between devices and actuators monitored."
-- 1.3.6.1.3.1.10.2
::= { domoConfiguring 2 }


domoConfiguringEntry OBJECT-TYPE
SYNTAX  DomoConfiguringEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Each entry contains information about each relationship between the devices
and monitored actuators."
INDEX {
domoConfiguringIndex }
-- 1.3.6.1.3.1.10.2.1
::= { domoConfiguringTable 1 }


DomoConfiguringEntry ::= SEQUENCE {

domoConfiguringIndex     Integer32,
domoConfiguringIdxSensor Integer32,
domoConfiguringValue     Integer32,
domoConfiguringSizeValue Integer32,
domoConfiguringTimeStamp OCTET STRING }


domoConfiguringIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
```

```
"A single value greater than zero for each relationship between device
and configured actuators. The assigned values should vary between
1 and N relations."
-- 1.3.6.1.3.1.10.2.1.1
::= { domoConfiguringEntry 1 }


domoConfiguringIdxSensor OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index value of an actuator."
-- 1.3.6.1.3.1.10.2.1.2
::= { domoConfiguringEntry 2 }


domoConfiguringValue OBJECT-TYPE
SYNTAX  Integer32 (-65535..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the value inserted in the actuator."
-- 1.3.6.1.3.1.10.2.1.3
::= { domoConfiguringEntry 3 }


domoConfiguringSizeValue OBJECT-TYPE
SYNTAX  Integer32 (-65535..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the size of the value inserted in
the actuator."
-- 1.3.6.1.3.1.10.2.1.4
::= { domoConfiguringEntry 4 }


domoConfiguringTimeStamp OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the date and time when the
value was inserted in the actuator."
-- 1.3.6.1.3.1.10.2.1.5
::= { domoConfiguringEntry 5 }


-- domoFunctionalities (11)****************************************** --
-- This group includes objets for defining informations about an available --
```

```
-- functionality of a device. Examples of functionalities include measuring --
-- temperature or turning on a device. --

domoFunctionalitiesNumber OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of device functionalities
present in the MIB."
-- 1.3.6.1.3.1.11.1
::= { domoFunctionalities 1 }


domoFunctionalitiesTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DomoFunctionalitiesEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various
functionalities present in the system."
-- 1.3.6.1.3.1.11.2
::= { domoFunctionalities 2 }


domoFunctionalitiesEntry OBJECT-TYPE
SYNTAX  DomoFunctionalitiesEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Each entry contains information about a functionality."
INDEX {
domoFunctionalitiesIndex }
-- 1.3.6.1.3.1.11.2.1
::= { domoFunctionalitiesTable 1 }


DomoFunctionalitiesEntry ::= SEQUENCE {

domoFunctionalitiesIndex       Integer32,
domoFunctionalitiesDescription OCTET STRING }


domoFunctionalitiesIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each new functionality. The values
assigned must vary between 1 and N functionalities."
-- 1.3.6.1.3.1.11.2.1.1
```

```
::= { domoFunctionalities 1 }


domoFunctionalitiesDescription OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about a text description of the
functionality."
-- 1.3.6.1.3.1.10.2.1.2
::= { domoFunctionalitiesEntry 2 }


-- domoDeviceFunctionality (12)****************************************** --
-- This group associates a functionality to a given device. --


domoDeviceFunctionalityNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of existing relationships
between the devices and the functionalities. These relationships make
it possible to associate certain functionalities with devices."
-- 1.3.6.1.3.1.12.1
::= { domoDeviceFunctionality 1 }


domoDeviceFunctionalityTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DomoDeviceFunctionalityEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various relationships
between devices and functionalities existing."
-- 1.3.6.1.3.1.12.2
::= { domoDeviceFunctionality 2 }


domoDeviceFunctionalityEntry OBJECT-TYPE
SYNTAX  DomoDeviceFunctionalityEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION ""
REFERENCE
"Each entry contains information about each relationship between the devices
and functionalities."
INDEX {
domoDeviceFunctionalityIndex }
```

```
-- 1.3.6.1.3.1.12.2.1
::= { domoDeviceFunctionalityTable 1 }


DomoDeviceFunctionalityEntry ::= SEQUENCE {

domoDeviceFunctionalityIndex              Integer32,
domoDeviceFunctionalityIdxDevice          Integer32,
domoDeviceFunctionalityIdxFunctionalities Integer32 }


domoDeviceFunctionalityIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each relationship between device
and functionality. The assigned values must vary between 1 and N relations."
-- 1.3.6.1.3.1.12.2.1.1
::= { domoDeviceFunctionalityEntry 1 }


domoDeviceFunctionalityIdxDevice OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index value of a device."
-- 1.3.6.1.3.1.12.2.1.2
::= { domoDeviceFunctionalityEntry 2 }


domoDeviceIdxFunctionalities OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index for the
DOMOFUNCTIONALITIES table where it contains the functionality referenced
in this table."
-- 1.3.6.1.3.1.12.2.1.3
::= { domoDeviceFunctionalityEntry 3 }


-- domoModule (13)****************************************************** --
-- This group includes objets for defining characteristics about a module --
-- in the system. A module is a representation of a gateway in the system --
-- acting as simple relaying device that redirects incoming messages --
-- to their destination. --
```

```
domoModuleNumber OBJECT-TYPE
SYNTAX  Integer32 (0..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of modules present in
the MIB."
-- 1.3.6.1.3.1.13.1
::= { domoModule 1 }


domoModuleTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DomoModulesEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various modules
present in the system."
-- 1.3.6.1.3.1.13.2
::= { domoModule 2 }


domoModuleEntry OBJECT-TYPE
SYNTAX  ModuleEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Each entry contains information about the module."
INDEX {
domoModuleIndex }
-- 1.3.6.1.3.1.13.2.1
::= { domoModuleTable 1 }


DomoModuleEntry ::= SEQUENCE {

domoModuleIndex       Integer32,
domoModuleDescription OCTET STRING }


domoModuleIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each new module. The values
assigned must vary between 1 and N modules."
-- 1.3.6.1.3.1.13.2.1.1
::= { domoModule 1 }
```

```
domoModulesDescription OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the module description
(for example, if it is a gateway and it's location)."
-- 1.3.6.1.3.1.13.2.1.2
::= { domoModulesEntry 2 }


-- domoModuleSpecification (14)****************************************** --
-- This group associates a specification to a given module. --

domoModuleSpecificationNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of existing relationships
between modules and specifications. These relationships make it possible
to associate certain modules with specifications."
-- 1.3.6.1.3.1.14.1
::= { domoModuleSpecification 1 }


domoModuleSpecificationTable OBJECT-TYPE
SYNTAX  SEQUENCE OF DomoModuleSpecificationEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various relationships
between modules and specifications existing."
-- 1.3.6.1.3.1.14.2
::= { domoModuleSpecification 2 }


domoModuleSpecificationEntry OBJECT-TYPE
SYNTAX  DomoModuleSpecificationEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION ""
REFERENCE
"Each entry contains information about each relationship between modules and
specifications."
INDEX {
domoModuleSpecificationIndex }
-- 1.3.6.1.3.1.14.2.1
::= { domoModuleSpecificationTable 1 }
```

```
DomoModuleSpecificationEntry ::= SEQUENCE {

domoModuleSpecificationIndex            Integer32,
domoModuleSpecificationIdxModule        Integer32,
domoModuleSpecificationIdxSpecification Integer32 }


domoModuleSpecificationIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each relationship between a module and a
specification. The assigned values must vary between 1 and N relations."
-- 1.3.6.1.3.1.14.2.1.1
::= { domoModuleSpecificationEntry 1 }


domoModuleSpecificationIdxModule OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index value of a module."
-- 1.3.6.1.3.1.14.2.1.2
::= { domoModuleSpecificationEntry 2 }


domoModuleSpecificationIdxSpecification OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index of the DOMOSPECIFICATION
table where the specification referenced in this table."
-- 1.3.6.1.3.1.14.2.1.3
::= { domoModuleSpecificationEntry 3 }


-- domoModuleDevice (15)********************************************* --
-- This group associates a device to a given module. That means that the device --
-- is connected to the associated gateway and every message is firstly sent to --
-- the gateway and only then is redirected to the intended destination. --


domoModuleDeviceNumber OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the number of existing relationships
```

```
between modules and devices. These relationships allow you to associate
certain modules with devices."
-- 1.3.6.1.3.1.15.1
::= { domoModuleDevice 1 }


domoModuleDeviceTable OBJECT-TYPE
SYNTAX  SEQUENCE OF domoModuleDeviceEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION
"Objet defining the information about the table with the various relationships
between modules and devices existing."
-- 1.3.6.1.3.1.15.2
::= { domoModuleDevice 2 }


domoModuleDeviceEntry OBJECT-TYPE
SYNTAX  DomoModuleDeviceEntry
MAX-ACCESS not-accessible
STATUS  current
DESCRIPTION ""
REFERENCE
"Each entry contains information about each relationship between modules and
devices."
INDEX {
domoModuleDeviceIndex }
-- 1.3.6.1.3.1.15.2.1
::= { domoModuleDeviceTable 1 }


DomoModuleDeviceEntry ::= SEQUENCE {

domoModuleDeviceIndex      Integer32,
domoModuleDeviceIdxModule  Integer32,
domoModuleDeviceIdxDevice  Integer32 }


domoModuleDeviceIndex OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"A single value greater than zero for each relationship between module and
device. The assigned values must vary between 1 and N relations."
-- 1.3.6.1.3.1.15.2.1.1
::= { domoModuleDeviceEntry 1 }


domoModuleDeviceIdxModule OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
```

```
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index value of a module."
-- 1.3.6.1.3.1.15.2.1.2
::= { domoModuleSpecificationEntry 2 }


domoModuleDeviceIdxDevice OBJECT-TYPE
SYNTAX  Integer32 (1..65535)
MAX-ACCESS read-only
STATUS  current
DESCRIPTION
"Objet defining the information about the index of the DOMODEVICE
table where the device referenced in this table."
-- 1.3.6.1.3.1.15.2.1.3
::= { domoModuleDeviceEntry 3 }


-- Conformance (16)****************************************** --
-- This group is required to ensure the correct declaration of the MIB.
It contains all the objets defined. --

domoticaSNMPGroups OBJECT IDENTIFIER
-- 1.3.6.1.3.1.16.1
::= { domoticaSNMPConf 1 }

domoticsMIBCompliances OBJECT IDENTIFIER
-- 1.3.6.1.3.1.16.2
::= { domoticaSNMPConf 2 }

basic OBJECT-GROUP
OBJECTS {
domoSpecificationNumber,
domoSpecificationIndex,
domoSpecificationDescription,
domoSpecificationValue,
domoSensorNumber,
domoSensorIndex,
domoSensorName,
domoSensorDescripton,
domoSensorUnits,
domoSensorScale,
domoSensorSampleType,
domoSensorMinValue,
domoSensorMaxValue,
domoSensorSamplingRate,
domoSensorNSamples,
domoSensorLastValue,
domoSensorLastSampleIndex,
domoSensorOldestSampIndex,
```

```
domoSensorStatsInterval,
domoSensorMedian,
domoActuatorNumber,
domoAactuatorIndex,
domoActuatorName,
domoActuatorDescripton,
domoActuatorUnits,
domoActuatorScale,
domoActuatorMinValue,
domoActuatorMaxValue,
domoActuatorLastSetValue,
domoActuatorRelatedSensorName,
domoDeviceActuatorNumber,
domoDeviceActuatorIndex,
domoDeviceActuatorDevice,
domoDeviceActuatorIdxActuator,
domoSystemDescription,
domoSystemContact,
domoSystemName,
domoSystemLocation,
domoSystemYype,
domoDeviceNumber,
domoDeviceIndex,
domoDeviceDescription,
domoDeviceIdentification,
domoDeviceBeaconRate,
domoDeviceManufacturer,
domoDeviceModel,
domoDeviceVersionAndDate,
domoDeviceNSensors,
domoDeviceNActuators,
domoDeviceNMaxSamples,
domoDeviceDateAndTime,
domoDeviceUpTime,
domoDeviceLastTimeUpdated,
domoDeviceOperationalStatus,
domoDeviceBatteryStatus,
domoDeviceReset,
domoDeviceSpecificationNumber,
domoDeviceSpecificationIndex,
domoDeviceSpecificationIdxDevice,
domoDeviceSpecificationIdxSpecification,
domoDeviceSensorNumber,
domoDeviceSensorIndex,
domoDeviceSensorIdxDevice,
domoDeviceSensorIdxSensor,
domoMonitoringNumber,
domoMonitoringIndex,
domoMonitoringIdxSensor,
domoMonitoringValue,
domoMonitoringSizeValue,
```

```
domoMonitoringTimeStamp,
domoConfigurationNumber,
domoConfigurationIndex,
domoConfigurationIdxActuator,
domoConfigurationValue,
domoConfigurationSizeValue,
domoConfigurationTimeStamp,
domoFunctionalitiesNumber,
domoFunctionalitiesIndex,
domoFunctionalitiesDescription,
domoDeviceFunctionalityNumber,
domoDeviceFunctionalityIndex,
domoDeviceFunctionalityIdxDevice,
domoDeviceFunctionalityIdxFunctionalities,
domoModulesNumber,
domoModulesIndex,
domoModulesDescription,
domoModuleSpecificationNumber,
domoModuleSpecificationIndex,
domoModuleSpecificationIdxModule,
domoModuleSpecificationIdxSpecification,
domoModuleDeviceNumber,
domoModuleDeviceIndex,
domoModuleDeviceIdxModule,
domoModuleDeviceIdxDevice }
STATUS  current
DESCRIPTION
"Group of all the objets defined in the MIB."
-- 1.3.6.1.3.1.16.1.1 --
-- 1.3.6.1.3.1.16.1.1
::= { domoticaSNMPGroups 1 }

END
```

A.3 DOMOTICS LIGHT MIB

```
informationModule
LAST-UPDATED "202210132010Z" -- Oct 13, 2022, 8:10:00 PM
ORGANIZATION "DomoticaSNMP"
CONTACT-INFO "Gonçalo Nogueira email: g1.nogueira.slb@hotmail.com"
DESCRIPTION "Information module for device informations"

-- groups --

-- (1) device --
-- This group includes objets to define characteristics of a device. --

-- (2) sensors --
-- This group includes objets to define characteristics of a sensor. --

-- (3) actuators --
-- This group includes objets to define characteristics of an actuator. --

-- (4) sample --
-- This group includes objets to define characteristics of a sample. -




-- device (1)********************************************************* --
-- This group includes objets to define characteristics of a device. --


device OBJECT {
 TYPE List type, description, beaconRate, manufacturer, model, versionAndDate,
 nSensors, nActuators, nMaxSamples, dateAndTime, upTime, lastTimeUpdated,
 operationalStatus, reset
 ACESS read-only
 NOTIFICATION type, description, nSensores, nAtuadores, upTime, lastTimeUpdated,
 operationalStatus, batteryStatus
 DESCRIPTION "Simple list of objects, where each object represents a characteristic
 from a device."
 IID 1 }

device.type OBJECT {
 TYPE String 45
 ACESS read-only
 NOTIFICATION-RATE beaconRate
 DESCRIPTION "Text description for the type of device."
 IID 1.1 }


device.description OBJECT {
 TYPE String 255
```

```
 ACESS read-only
 DESCRIPTION "Text description of the device. This description can include its
location and more information considered relevant."
 IID 1.2 }

device.beaconRate OBJECT {
 TYPE Integer
 ACESS read-write
 DESCRIPTION "Frequency rate in s for issuing a notifiction message that acts as a
message beacon to inform the agent that this device is on and ready to setup a
connection."
 IID 1.3 }


device.manufacturer OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Device manufacturer information. Most commonly just the
name of the manufacturer is needed."
 IID 1.4 }

device.model OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Device manufacturer model. Most commonly just the
name of the model is needed."
 IID 1.5 }

device.versionAndDate OBJECT {
 TYPE String 45
 ACESS read-only
 NOTIFICATION-RATE beaconRate
 DESCRIPTION "Device version information and the date of that version."
 IID 1.6 }

device.nSensors OBJECT {
 TYPE Integer
 ACESS read-only
 NOTIFICATION-RATE beaconRate
 DESCRIPTION "Number of sensors associated with a device."
 IID 1.7 }

device.nActuators OBJECT {
 TYPE Integer
 ACESS read-only
 NOTIFICATION-RATE beaconRate
 DESCRIPTION "Number of actuators associated with a device."
 IID 1.8 }

device.nMaxSamples OBJECT {
 TYPE Integer
```

```
 ACESS read-only
 DESCRIPTION "the maximum number of samples that each associated sensor
to this device can store."
 IID 1.9 }

device.nSamples OBJECT {
 TYPE Integer
 ACESS read-only
 DESCRIPTION "Total number of samples stored by all device sensors"
 IID 1.9 }


device.dateAndTime OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Date and time when device setup was performed."
 IID 1.10 }

device.upTime OBJECT {
 TYPE Integer
 ACESS read-only
 NOTIFICATION-RATE beaconRate
 DESCRIPTION "Information about the percentage of time the device was
available."
 IID 1.11 }

device.lastTimeUpdated OBJECT {
 TYPE String 45
 ACESS read-only
 NOTIFICATION-RATE beaconRate
 DESCRIPTION "Date and time of the last update of any information of
the device."
 IID 1.12 }

device.operationalStatus OBJECT {
 TYPE Integer
 ACESS read-only
 NOTIFICATION-RATE beaconRate
 DESCRIPTION "The operational state of the device, where the value 0
corresponds to standby state, 1 corresponds to normal state and 2 to
eco-mode."
 IID 1.13 }

device.batteryStatus OBJECT {
 TYPE Integer
 ACESS read-only
DESCRIPTION "The device's battery status, ranging from 1 to 100.
If the device is not powered by battery, the value must be 0."
 IID 1.14 }

device.reset OBJECT {
```

```
 TYPE Integer
 ACESS read-only
DESCRIPTION "Information to operate different types of device reset.
The value 0 and the default value, 1 corresponds to device shutdown,
2 to a hard reset and 3 to a soft reset."
 IID 1.15 }



-- sensors (2)****************************************************** --
-- This group includes objets to define characteristics of a sensor. --


sensors OBJECT {
 TYPE Table type, description, units, scale, sampleType, minValue, maxValue,
 samplingRate, nSamples, lastValue, lastSampleIndex, oldestSampleIndex, statsInterval,
 median
 ACESS read-only
 NOTIFICATION   nSamples, lastValue, lastSampleIndex, median
 DESCRIPTION "Simple table of the available device' sensors"
 IID 2 }

sensors.type OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Sensor type. For example, temperature, humidity, brightness,
 motion detection, etc."
 IID 2.1 }

sensors.description OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Text description of the sensor. Informations such as location
can be disclosed here."
 IID 2.2 }

sensors.units OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Information about the units measured by the sensor."
 IID 2.3 }

sensors.scale OBJECT {
 TYPE Integer
 ACESS read-only
 DESCRIPTION "The number of decimal places that the sensor value has.
For instance, the value -1 corresponds to multiplying the value by 0.1,
the value 1 to multiply by 10."
 IID 2.4 }

sensors.sampleType OBJECT {
```

```
 TYPE Integer
 ACESS read-only
 DESCRIPTION "A number that identifies the type of value the sensor measures.
The value 1 corresponds to integer."
 IID 2.5 }

sensors.minValue OBJECT {
 TYPE Opaque
 ACESS read-only
 DESCRIPTION "The minimum value that the sensor can read."
 IID 2.6 }

sensors.maxValue OBJECT {
 TYPE Opaque
 ACESS read-only
 DESCRIPTION "The maximum value that the sensor can read."
 IID 2.7 }

sensors.samplingRate OBJECT {
 TYPE Integer
 ACESS read-write
 DESCRIPTION "The frequency rate value in seconds of sampling by the sensor"
 IID 2.8 }

sensors.nSamples OBJECT {
 TYPE Integer
 ACESS read-only
 NOTIFICATION-RATE samplingRate
 DESCRIPTION "The number of samples that the sensor has collected so far."
 IID 2.9 }

sensors.lastValue OBJECT {
 TYPE Opaque
 ACESS read-only
 NOTIFICATION-RATE samplingRate
 DESCRIPTION "The last value read by the sensor"
 IID 2.10 }

sensors.lastSampleIndex OBJECT {
 TYPE Integer
 ACESS read-only
 DESCRIPTION "The identifier index of the last sample registered by the sensor."
 IID 2.11 }

sensors.oldestSampleIndex OBJECT {
 TYPE Integer
 ACESS read-only
 DESCRIPTION "The identifier index of the oldest sample registered by the sensor."
 IID 2.12 }

sensors.statsInterval OBJECT {
```

```
 TYPE Integer
 ACESS read-only
 DESCRIPTION "The interval of time that it takes between each calculation of
statistics by the sensor. The unit of this value is seconds."
 IID 2.13 }

sensors.median OBJECT {
 TYPE Opaque
 ACESS read-only
 DESCRIPTION "The median of the sensor samples."
 IID 2.14 }




-- actuators (3)********************************************************* --
-- This group includes objets to define characteristics of an actuator. --

actuators OBJECT {
 TYPE Table type, description, units, scale, lastSetValue, minValue, maxValue ,setValue, tim
 ACESS read-only
 DESCRIPTION "Simple table of the available device's actuators."
 IID 3 }


actuators.type OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Actuator type. For example, temperature, brightness, status, etc."
 IID 3.1 }

actuators.description OBJECT {
 TYPE String 255
 ACESS read-only
 DESCRIPTION "Text description of the actuator. Informations such as location
can be disclosed here."
 IID 3.2 }

actuators.units OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "The units measured by the actuator."
 IID 3.3 }

actuators.scale OBJECT {
 TYPE Integer
 ACESS read-only
 DESCRIPTION "the number of decimal placesthat the actuator value has.
For instance, the value -1 corresponds to multiplying the value by 0.1,
the value 1 to multiply by 10."
 IID 3.4 }
```

```
actuators.lastSetValue OBJECT {
 TYPE Opaque
 ACESS read-write
 NOTIFICATION
 DESCRIPTION "The last value inserted in the actuator."
 IID 3.5 }

actuators.minValue OBJECT {
 TYPE Opaque
 ACESS read-only
 DESCRIPTION "The minimum value that the actuator can apply."
 IID 3.6 }

actuators.maxValue OBJECT {
 TYPE Opaque
 ACESS read-only
 DESCRIPTION "The maximum value that the actuator can apply."
 IID 3.7 }


actuators.setValue OBJECT {
 TYPE Opaque
 ACESS read-write
 DESCRIPTION "Value inserted in the actuator."
 IID 3.8 }

actuators.timeStamp OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Indicates the exact time when the last value was entered into
the actuator. Indicates year, month, day, hour, minute, second and
optionally fractional seconds"
 IID 3.9 }

actuators.relatedSensorIndex OBJECT {
 TYPE Integer
 ACESS read-only
 DESCRIPTION "Sensor table index that identifies a sensor that is related to
the actuator."
 IID 3.10 }




-- samples (4)********************************************************* --
-- This group includes objets to define characteristics of a sample. --


samples OBJECT {
```

```
 TYPE IndexedTable valueType, value, timeStamp
 KEY sensors
 ACESS read-only
 DESCRIPTION "Indexed array of tables, each element contains a table and is
 identified by a key that is related to the sensor table row
 identified by an index of the same value."
 IID 4
}


samples.value OBJECT {
 TYPE Opaque
 ACESS read-only
 DESCRIPTION "Value of the sample"
 IID 4.1 }

samples.timeStamp OBJECT {
 TYPE String 45
 ACESS read-only
 DESCRIPTION "Indicates the exact time the sample was taken. Indicates year,
month, day, hour, minute, second and optionally fractional seconds"
 IID 4.2 }
```