Universidade do Minho - Dep.to Informática
1º Semestre, 2024/2025
Mestrado em Engenharia de Telecomunicações e Informática
GVR - Módulo de Gestão de Redes

Ficha de Trabalho Prático
L-SNMPvS para Sistemas Domóticos

Objetivo Principal:

- Consolidação dos conhecimentos sobre os protocolos, mecanismos e filosofias da arquitetura de gestão *Internet-standard Network Management Framework* (INMF), dando especial relevo aos aspetos de interação protocolar, concetualização de objetos de gestão e especificação de MIB.

Observações:

- O trabalho deverá ser realizado ao longo de 30 a 40 horas efetivas de trabalho em grupos de até três alunos.

Requisitos Recomendáveis:

- Acesso a sistema com, pelo menos, um pacote *freeware* instalado com suporte a SNMP (versão 2, no mínimo): **Net-SNMP**, CMU-SNMP, SCOTTY, etc.
- Utilização de APIs de programação que facilitem a implementação de primitivas SNMPv2c ou SNMPv1.

AVISOS:

- Não serão tolerados atropelos aos direitos de autor de qualquer tipo de *software*...

Bibliografia genérica e material de apoio

Material de apoio:

- Manuais/Tutoriais do *net-snmp*;
- MIBs em `/usr/share/snmp/mibs` (ou diretoria equivalente da instalação);
- Recurso `http://net-snmp.sourceforge.net/wiki/index.php/Tutorials/`;
- Recurso `http://www.simpleweb.org/`;
- Recurso `http://www.snmplinks.org/`.

Bibliografia:

- G. Nogueira, *Arquitetura Integradora com SNMP para Gestão de Edifícios*, Universidade do Minho, 2023.
- M. Rose, *The Simple Book*, Second Edition, Prentice Hall, 1996.
- B. Dias, *Gestão de Redes*, PAPCC, Universidade do Minho, 1996.
- W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Addison-Wesley, 2000.
- D. Mauro, K. Schmidt, *Essential SNMP*, O'Reilly, 2001.
- Ver outros recursos na área de "Conteúdo" no BB da UC e no material fornecido no início do semestre.

Modelo L-SNMPvS para sistemas domóticos

No passado já foram definidos vários modelos de controlo domóticos baseados na arquitetura SNMP, definindo-se MIB específicas para as funcionalidades próprias destes sistemas. Ou seja, a estratégia 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, mas nunca foi realmente 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.

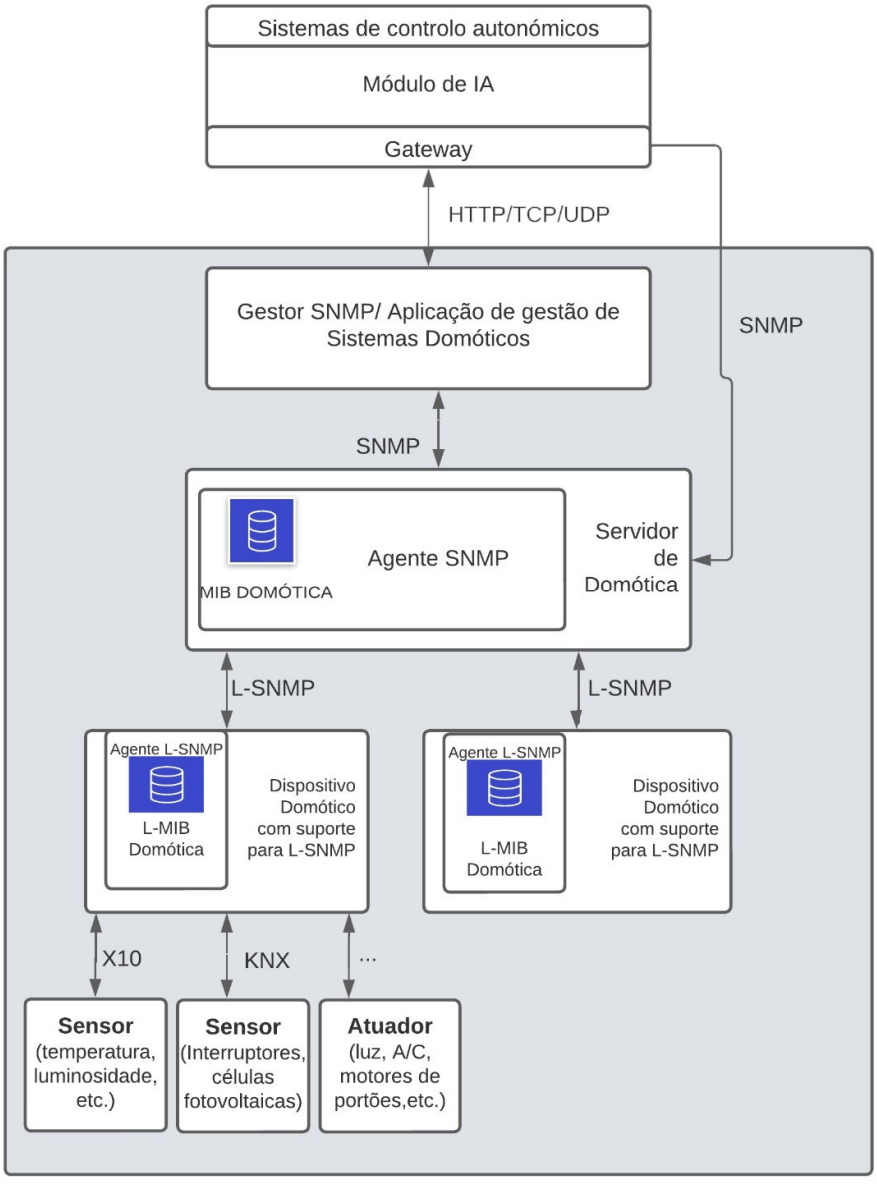

Figura 1: Arquitetura de gestão para sistemas domóticos usando SNMP e L-SNMP.

Recentemente, num projeto experimental criado na Universidade do Minho em cooperação com o laboratório DTX, foi desenvolvido um novo modelo para tentar resolver estes problemas. Esta abordagem ainda permite o uso de agentes SNMP mas acrescenta uma nova arquitetura complementar designada de *Light SNMP* (L-SNMP) que obriga à implementação adicional de um agente de gestão L-

SNMP, ainda que baseado no paradigma dos agentes SNMP, implementando um tipo de MIB especial (*Light MIB* ou L-MIB) diretamente nos dispositivos domóticos. Este novo tipo de MIB permite implementações computacionalmente menos exigentes do lado dos fabricantes de dispositivos. O L-SNMP permite a inclusão de um nível funcional intermédio entre os dispositivos domóticos (normalmente implementados em *hardware*) e os agentes SNMP tradicionais ou outro tipo de sistemas de controlo. 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 (ver Figura 1).

De notar que o modelo L-SNMP pode ser aplicado a outros sistemas de gestão para além dos sistemas domóticos, sempre que um nível adicional de abstração seja útil para definir as funcionalidades dos recursos a gerir e/ou sempre que estes tenham limitações computacionais e/ou de comunicações. Por exemplo, a tecnologia L-SNMP pode ser adequada na gestão de sistemas locais veiculares (dentro dum veículo), redes de sensores ou sistemas IOT.

Objetivos

O principal objetivo do trabalho é a implementação dum sistema de gestão domótico suportando alguns dispositivos virtuais abstraindo equipamentos de iluminação e de ar condicionado. O sistema deve ser simples, sem a necessidade de ligação a um sistema SNMP ou sistema de inteligência artificial para controlo. A comunicação entre o agente e o gestor deve ser feita utilizando uma versão simplificada do protocolo L-SNMP, o L-SNMPvS, conforme especificada neste enunciado.

Deve ser constituído pelos seguintes módulos de software:

- Equipamento virtual de monitorização que permita monitorizar a intensidade de luz e a temperatura ambiente em várias zonas duma habitação;
- Equipamento virtual de iluminação que permita ligar e desligar várias luzes, opcionalmente com vários níveis de intensidade, em várias zonas duma habitação;
- Equipamento virtual de ar condicionado que permita ligar e desligar um sistema de aquecimento ou arrefecimento, opcionalmente com vários níveis de intensidade, em várias zonas duma habitação;
- Agente L-SNMPvS que implemente uma L-MIBvS (versão simplificada da L-MIB) para sistemas domóticos e que realize a instrumentação de conexão funcional com os módulos dos equipamentos virtuais (pode até ser tudo implementado num único módulo); o agente deve implementar apenas os objetos necessários da L-MIBvS para sistemas domóticos conforme indicado em anexo deste enunciado.
- Interface aplicacional para o utilizador que permita verificar que luzes estão ligadas, qual a temperatura ambiente e que sistemas de ar condicionado estão ligados; a interface deve também permitir controlar manualmente o sistema de iluminação e de ar condicionado e, opcionalmente, fazer a configuração para um funcionamento automático; esta interface implementará um gestor L-SNMPvS para comunicação com o agente L-SNMPvS.

Protocolo L-SNMPvS

O protocolo L-SNMP é baseado no SNMP. É mais simples, mas permite algumas funcionalidades que o SNMP não permite diretamente (como, por exemplo, "*beacons*" e monitorização em "*broadcasting*"). Para o presente trabalho é requerida a implementação duma versão simplificada do L-SNMP, o L-SNMP-vS, com codificação das mensagens em sequências de carateres terminados por '\0' e encapsulamento direto em datagramas UDP.

O PDU das mensagens do protocolo do L-SNMPvS tem o seguinte formato:

```
Tag | Type | Time-Stamp | Message-Identifier | IID-List | Value-List | Error-List
```

Segue-se a especificação de cada um dos campos da mensagem L-SNMPvS:

- `Tag` – valor fixo igual a `kdk847ufh84jg87g` e que serve de identificador rápido do protocolo para simplificar *debug*; codificado como uma sequência de 16 carateres terminada por '\0';
- `Type` – indica o tipo de mensagem e pode ter um de quatro valores possíveis (`G`, `S`, `N` e `R`), que indicam, respetivamente, os tipos *get-request* e *set-request* para serem usados pelos gestores/clientes e os tipos *notification* e *response* para serem usados pelos agentes/servidores; codificado como um carater (código ASCII de um byte) com o valor respetivo;
- `Time-Stamp` – valor temporal, com precisão até aos milésimos de segundo, e que indica o momento (data, horas, minutos, segundos e milésimos de segundo) em que o pedido foi enviado pelo gestor/cliente (quando o se trata duma mensagem do tipo *get-request* ou *set-request*) ou quando os valores das instâncias dos objetos respetivos foram calculados pelo agente/servidor (neste caso, o *timestamp* refere-se ao tempo decorrido desde o início/*boot*/*reset* do agente/servidor e vai nas mensagens do tipo *notification* ou *response*); no primeiro caso o formato a usar é `day:month:year:hours:mins:secs:ms` (`ms` são os milésimos de segundo) e, no segundo caso, o formato a usar é `days:hours:mins:secs:ms`; codificado como o tipo *timestamp* que é utilizado na codificação de valores de instâncias (ver secções posteriores) no modelo L-SNMPvS;
- `Message-Identifier` – valor aleatório que deve identificar univocamente um pedido dum gestor por forma a que o agente identifique as respostas respetivas ou que deve identificar univocamente mensagens de notificação (*beacons*) do agente; codificado como uma sequência de 16 carateres terminada por '\0';
- `IID-List` – lista de identificadores de instâncias de objetos; cada elemento da lista é um *Instance Identifier* (IID); os IID são um conceito análogo aos OID do SNMP e a sua explicação mais detalhada poderá ser encontrada na bibliografia indicada e na secção própria deste enunciado; cada IID deve ser codificado usando as regras de codificação dos IID do modelo L-SNMPvS (ver secções posteriores);
- `Value-List` – lista de valores correspondentes à lista de IID anterior (o primeiro valor é o valor correspondente à instância referida pelo primeiro IID da lista anterior e assim por diante); cada valor deve ser codificado utilizando as regras de codificação para os tipos de dados possíveis no modelo L-SNMPvS;
- `Error-List` – lista de erros encontrados pelo agente/servidor; o gestor não usa este campo pelo que deve enviar uma lista vazia; no caso do agente deve incluir os códigos (lista de valores inteiros) dos erros definidos para o L-SNMPvS; no contexto deste trabalho devem ser os alunos a definir um conjunto pequeno de códigos de erro possíveis mas sugere-se a definição de, pelo menos, os seguintes: 0 (sem erros), 1 (erro na descodificação da mensagem), 2 (erro na Tag), 3 (tipo de mensagem desconhecido), 4 (mensagem duplicada), 5 (IID inválido ou desconhecido), 6 (tipo de valor desconhecido), 7 (valor não suportado) e 8 (lista de valores não corresponde à lista de IID); cada valor inteiro representando um erro deve ser codificado como uma sequência de dígitos decimais terminada por '\0'.

Os campos do PDU L-SNMPvS que são listas (IID, valores e erros) e devem ser codificados da seguinte forma (o tipo dos seus elementos é implícito):

```
N-Elements | 1st-Element | 2nd-Element | ... | Nth-Element
```

A primeira parte é um inteiro codificado como uma simples sequência de dígitos decimais terminada por '\0'. Na segunda parte, que só aparece quando o número de elementos da lista for maior do que zero, são codificados os elementos sequencialmente, do primeiro ao enésimo elemento. No caso de listas de IID, cada IID deve ser codificado utilizando o tipo IID do modelo L-SNMPvS. No caso de listas de valores, cada elemento será codificado com o seu tipo específico utilizando as regras definidas para o modelo L-SNMPvS e detalhado nas secções seguintes. No caso de listas de erros, cada erro é codificado como uma sequência de dígitos decimais terminada por '\0'.

As mensagens do tipo *get-request* e *set-request* são enviadas pela aplicação gestora para obter informação (valores de instâncias de objetos) implementada pela instrumentação do agente.

As mensagens do tipo *response* só podem ser usadas por um agente para enviar a informação referida no respetivo pedido *get-request* ou *set-request* dum gestor. Para além disso, no caso dum *set-request*, o agente deve, antes de responder ao gestor, tentar modificar as instâncias referidas no pedido com os valores respetivos também presentes no pedido (o que implicará a implementação de procedimentos de controlo/configuração dos recursos abstraídos pelos objetos em causa).

Tal como o SNMP, o protocolo L-SNMPvS é assíncrono, assimétrico, atómico e não confirmado (não existe obrigatoriedade de resposta, não existem respostas só de confirmação nem existem quaisquer valores de *timeouts* pré-definidos).

Por fim, as mensagens do tipo *notification* servem para os agentes enviarem, com uma frequência pré-determinada, informações de estado (valores de instâncias de objetos especiais que tenham sido pré-definidos para este tipo de notificações) a todos os gestores da rede local através de *broadcasting*. A implementação deste tipo de mensagens é opcional.

Identificadores de Instâncias de Objetos

Todas as instâncias dos objetos implementados na instrumentação duma MIB do L-SNMPvS (L-MIBvS) têm de estar organizadas numa estrutura de dados que pode ser de dois tipos: Grupo ou Tabela. As instâncias são identificadas/referenciadas por uma sequência de inteiros conhecido por Identificador de Instância de Objeto (IID) em que cada inteiro tem a sua função representativa:

```
Structure | Object | 1st-Index* | 2nd-Index*
```

** Zero ou mais elementos*

Nas L-MIBvS não existe exatamente o mesmo conceito de Grupo como nas MIB tradicionais, mas os objetos também não podem ser referenciados diretamente no primeiro nível de identificação. Os campos que identificam a estrutura de dados e o objeto dentro da estrutura de dados são obrigatórios, portanto, a sequência de inteiros que forma um IID tem, no mínimo, dois inteiros.

A codificação de sequências de inteiros dum IID segue as regras definidas para o L-SNMPvS e será detalhada em secções posteriores deste enunciado, mas serão explicados os significados de cada campo dum IID:

- O valor inteiro `Structure` identifica uma das estruturas de dados de mais alto nível da L-MIBvS e o seu valor tem de ser maior do que zero. O tipo da estrutura não é indicado explicitamente no IID mas é sabido implicitamente pela especificação da L-MIBvS.
- O valor inteiro `Object` identifica um dos objetos pertencentes à estrutura de dados referido por `Structure`. Os objetos podem ser de quatro tipos (*integer*, IID, *timestamp* ou *string*).
- Nos grupos não existe qualquer relação entre os objetos duma estrutura que não seja, eventualmente, um agrupamento funcional ou administrativo genérico, ou seja, um grupo deve conter objetos que se relacionem funcionalmente ou administrativamente duma forma genérica,

mas não deve existir qualquer relação operacional no valor das suas instâncias. Ao invés, nas tabelas existe uma relação operacional entre as instâncias dos objetos. Neste caso, as instâncias são identificadas por ordem de referência (índices) e as instâncias com o mesmo número de referência (índice) em todos os objetos estão relacionadas operacionalmente, ou seja, a ordem das instâncias identifica uma coleção de instâncias de objetos relacionados com o mesmo recurso a gerir duma forma análoga à duma MIB no SNMP tradicional onde os índices são designados de chaves. Os objetos dum grupo podem ter um número diferente de instâncias implementadas no mesmo agente, mas os objetos duma tabela têm de ter o mesmo número de instâncias implementadas no mesmo agente. É comum referir-se que uma chave dum OID do SNMP identifica, numa tabela duma MIB, uma linha dessa tabela. Também no L-SNMPvS podemos afirmar que os valores dos índices representam uma ou mais linhas duma tabela numa L-MIBvS.

- Se o valor de `Object` for igual a zero, então o IID representa o número de objetos no grupo ou tabela (neste último caso cada objeto pode ser visto como uma coluna da tabela, portanto o IID representa o número de colunas da tabela).
- Se o IID não contiver qualquer índice e o valor de `Object` for superior a zero, então o IID representa implicitamente a primeira instância do objeto indicado, isto é, é equivalente a `Structure.Object.1`
- De notar que as N instâncias dum objeto são identificadas pelos índices 1 a N, respetivamente. Se existir apenas o valor do primeiro índice e se este for igual a zero o IID representa o número de instâncias do objeto implementadas naquele momento pelo agente (no caso das tabelas representa o número total de instâncias da coluna respetiva, ou seja, o número de linhas da tabela). Se existir apenas o valor do primeiro índice e se este for maior que zero o IID identifica a instância respetiva (se o valor do índice for maior que o número N de instâncias do objeto implementadas naquele momento pelo agente, o IID não deve ser considerado válido pelo agente). Se existirem dois índices e forem os dois iguais a zero, o IID representa o conjunto de valores de todas as instâncias desse objeto (no caso das tabelas representa os valores de todas as instâncias duma coluna da tabela). Se existirem dois índices e forem ambos maior do que zero, o IID representa o grupo de valores das instâncias identificadas pelos índices entre o primeiro e o segundo índice do IID. Se existirem dois índices e apenas um deles for igual a zero, ou se o segundo for mais pequeno que o primeiro, ou se algum for maior do que N, então o IID deve ser considerado inválido.

Tipos de Valores e sua Codificação no modelo L-SNMPvS

Conforme já foi referido, existirão apenas quatro tipos de dados a ser usados no modelo L-SNMPvS: *integer*, *timestamp*, *string* e IID. Genericamente todos estes tipos são codificados através do seguinte triplo: **`Data-Type | Length | Value`**.

O campo `Data-Type` deve ser codificado como uma sequência com um único carater (código ASCII de um byte) terminada por '\0'. O carater identifica, respetivamente, cada um dos quatro tipos de dados: `I`, `T`, `S` e `D`.

O campo `Length` indica quantos elementos compõem o tipo e deve ser codificado como uma sequência respetiva de dígitos decimais terminada por '\0'. No caso do tipo *integer* e do tipo *string*, a sequência deste campo deve representar o valor 1. No caso do tipo *timestamp*, a sequência deste campo deve representar o número de componentes do *timestamp* (5 ou 7). No caso do tipo IID, a sequência deste campo deve representar o número de componentes do IID (2, 3 ou 4).

O campo `Value` é codificado de forma específica para cada tipo:

- Para o tipo *integer*, este campo é codificado como uma sequência de dígitos decimais, representando o valor decimal respetivo (com a eventual inclusão no início do sinal `+` ou `–` para indicar um inteiro negativo ou positivo), terminada por '\0'; o maior valor decimal absoluto que é possível codificar é 2^{64}-1.
- No caso do tipo *timestamp*, variante com 7 componentes, este campo indica uma data completa do tipo `day:month:year:hours:mins:secs:ms`; cada um dos 7 componentes da data é um número inteiro codificado como uma sequência de dígitos decimais (os dias vão de 1 a 31, os meses são numerados de 1 a 12 e os anos são indicados pelo número completo, por exemplo, 2024) terminada por '\0'; os separadores ':' não são incluídos na codificação e são apresentados aqui apenas para facilitar a compreensão da semântica associada ao valor codificado;
- No caso do tipo *timestamp*, variante com 5 componentes, este campo indica um valor temporal que representa um intervalo temporal com a duração definida por `days:hours:mins:secs:ms`; cada um dos 5 componentes é um número inteiro codificado como uma sequência de dígitos decimais terminada por '\0'; os separadores ':' não são incluídos na codificação e são apresentados aqui apenas para facilitar a compreensão da semântica associada ao valor codificado;
- Para o tipo *string*, este campo é a sequência de carateres (códigos ASCII de um byte cada) que a define, terminada por '\0';
- Para o tipo IID, este campo é uma sequência dos 2, 3 ou 4 valores inteiros (ou componentes) que compõem um IID, conforme definido na secção anterior deste enunciado; cada componente é codificada pela sequência de dígitos decimais respetiva, terminada por '\0'.

Domotics L-MIBvS

```
[…]
-- Structures:
-- (1) device Group
-- (2) sensors Table
-- (3) actuators Table

-- device (1)
-- This group includes objects that represent characteristics of a Domotics device.

device OBJECT {
TYPE Group
INCLUDE id, type, beaconRate, nSensors, nActuators, dateAndTime, upTime, lastTimeUpdated,
operationalStatus, reset
NOTIFICATION id, type, nSensores, nAtuadores, dateAndTime, upTime, lastTimeUpdated,
operationalStatus
NOTIFICATION-RATE beaconRate
DESCRIPTION "Simple list of objects, where each object represents a characteristic
from a domotics device."
IID 1 }

device.id OBJECT {
TYPE String
ACESS read-only
DESCRIPTION "Tag identifying the device (the MacAddress, for example)."
IID 1.1 }

device.type OBJECT {
TYPE String
ACESS read-only
DESCRIPTION "Text description for the type of device (“Lights & A/C Conditioning”, for
example)."
IID 1.2 }
```

```
device.beaconRate OBJECT {
TYPE Integer
ACESS read-write
DESCRIPTION "Frequency rate in seconds for issuing a notification message with information
from this group that acts as a beacon broadcasting message to all the managers in the LAN.
If value is set to zero the notifications for this group are halted."
IID 1.3 }

device.nSensors OBJECT {
TYPE Integer
ACESS read-only
DESCRIPTION "Number of sensors implemented in the device and present in the sensors
Table."
IID 1.4 }

device.nActuators OBJECT {
TYPE Integer
ACESS read-only
DESCRIPTION Number of actuators implemented in the device and present in the actuators
Table."
IID 1.5 }

device.dateAndTime OBJECT {
TYPE Timestamp
ACESS read-write
DESCRIPTION "System date and time setup in the device."
IID 1.6 }

device.upTime OBJECT {
TYPE Timestamp
ACESS read-only
DESCRIPTION "For how long the device is working since last boot/reset."
IID 1.7 }

device.lastTimeUpdated OBJECT {
TYPE Timestamp
ACESS read-only
DESCRIPTION "Date and time of the last update of any object in the device L-MIBvS."
IID 1.8 }

device.operationalStatus OBJECT {
TYPE Integer
ACESS read-only
DESCRIPTION "The operational state of the device, where the value 0
corresponds to a standby operational state, 1 corresponds to a normal operational state
and 2 or greater corresponds to an non-operational error state."
IID 1.9 }

device.reset OBJECT {
TYPE Integer
ACESS read-write
DESCRIPTION "Value 0 means no reset and value 1 means a reset procedure must be done."
IID 1.10 }
```

```
-- sensors (2)
-- This Table includes objects that permit access to all sampled values from all the
-- sensors in the device

sensors OBJECT {
TYPE Table
INCLUDE id, type, status, minValue, maxValue, lastSamplingTime
DESCRIPTION "Table with information for all types of sensors connected to the device."
IID 2 }

sensors.id OBJECT {
TYPE String
ACESS read-only
DESCRIPTION "Tag identifying the sensor (the MacAddress, for example)."
IID 2.1 }

sensors.type OBJECT {
TYPE String
ACESS read-only
DESCRIPTION "Text description for the type of sensor (“Light”, for example)."
IID 2.2 }

sensors.status OBJECT {
TYPE Integer
ACESS read-only
DESCRIPTION "Last value sampled by the sensor in percentage of the interval between
minValue and maxValue."
IID 2.3 }

sensors.minValue OBJECT {
TYPE Integer
ACESS read-only
DESCRIPTION "Minimum value possible for the sampling values of the sensor."
IID 2.4 }

sensors.maxValue OBJECT {
TYPE Integer
ACESS read-only
DESCRIPTION "Maximum value possible for the sampling values of the sensor."
IID 2.5 }

sensors.lastSamplingTime OBJECT {
TYPE Timestamp
ACESS read-only
DESCRIPTION "Time elapsed since the last sample was obtained by the sensor."
IID 2.6 }

-- actuators (3)
-- This Table includes objects that permit control over all the actuators
-- in the device

actuators OBJECT {
TYPE Table
INCLUDE id, type, status, minValue, maxValue, lastControlTime
DESCRIPTION "Table with objects to control all actuators connected to the device."
IID 3 }
```

```
actuators.id OBJECT {
TYPE String
ACESS read-only
DESCRIPTION "Tag identifying the actuator (the MacAddress, for example)."
IID 3.1 }

actuators.type OBJECT {
TYPE String
ACESS read-only
DESCRIPTION "Text description for the type of actuator (“Temperature”, for example)."
IID 3.2 }

actuators.status OBJECT {
TYPE Integer
ACESS read-write
DESCRIPTION "Configuration value set for the actuator (value must be between minValue and
maxValue)."
IID 3.3 }

actuators.minValue OBJECT {
TYPE Integer
ACESS read-only
DESCRIPTION "Minimum value possible for the configuration of the actuator."
IID 3.4 }

actuators.maxValue OBJECT {
TYPE Integer
ACESS read-only
DESCRIPTION "Maximum value possible for the configuration of the actuator."
IID 3.5 }

actuators.lastControlTime OBJECT {
TYPE Timestamp
ACESS read-only
DESCRIPTION "Date and time when the last configuration/control operation was executed."
IID 3.6 }
```

Relatório e outras recomendações

O código dos protótipos do agente e do gestor podem ser desenvolvidas numa linguagem de programação à escolha. Podem usar-se APIs, funções ou excertos de código de terceiros para implementar aspetos tecnológicos, desde que devidamente referenciados. Em caso de dúvida consulte o docente para verificar a adequação da sua utilização no contexto do trabalho.

O código deve ser claro e usar convenções de nomeação de variáveis, tipos, funções e constantes. O código deve ser estruturado duma forma o mais modular possível, sem complexidades desnecessárias, e permitir reutilização, sempre que possível. A qualidade e correção do código não se mede pelo seu tamanho! Os alunos devem documentar/explicar o código criado através de comentários nas secções mais relevantes e no relatório. Todos os ficheiros do código devem ter um cabeçalho com a informação relevante que identifique os seus autores e explique as principais funções/classes criadas, tipos de dados e variáveis usadas, etc.

O relatório pode ser escrito no formato e no editor que for mais conveniente e deve incluir, no mínimo:

- Uma primeira página com o título do trabalho e a identificação do autor (incluindo fotografia), universidade, curso, unidade curricular e data de entrega;
- Um índice do conteúdo;

- Uma secção com a discussão das estratégias escolhidas, as opções tomadas, os mecanismos e tecnologias adotados, incluindo eventuais otimizações;
- Uma secção com a explicação e análise crítica das principais funções/classes implementadas, os seus principais méritos e a suas limitações mais importantes;
- Uma secção de conclusões que inclua uma eventual discussão sobre o que gostava de ter feito melhor ou de ter acrescentado e não conseguiu;
- Uma lista de eventuais referências bibliográficas, artigos científicos ou recursos informais na web que tenham sido úteis.

O relatório é para ser avaliado pelo docente por isso não inclua informação genérica e irrelevante que o docente já conheça. Tente ser conciso e claro. Sempre que incluir uma afirmação importante no contexto do relatório e que seja de autoria de terceiros, ou que seja baseada diretamente em afirmações de terceiros ou concluída de informação retirada de recursos alheios, deve referenciar corretamente essas autorias ou proveniências e acrescenta-las na lista das referências.

O relatório pode incluir algumas partes relevantes do código quando estas ajudam às análises e justificações apresentadas. Não inclua código desnecessário no texto do relatório. Sempre que possível, comente antes o próprio código.

No material entregue inclua apenas dois ficheiros: um ficheiro PDF com o relatório e um ficheiro zip com todos os ficheiros do código do projeto dentro duma diretoria com o seguinte nome `GR-24_25-NúmerosAlunos.zip`, como por exemplo, `GR-24_25-53454_345454.zip`.

Por fim, é recomendável que, durante a defesa do trabalho, os alunos tentem responder honesta e concisamente apenas às questões colocadas por forma a que as sessões de apresentação não se arrastem muito para além dos 30-40 minutos.