

**UNIVERSIDADE DO MINHO**

**4º ANO DO MESTRAZO INTEGRADO EM ENGENHARIA  
INFORMÁTICA**

## **Gestão de Redes**

**Relatório Trabalho Prático III**

**Implementação de Agentes SNMP em micro-computadores  
Caso de Estudo Raspberry Pi Modelo B+ 2014**

**Filipe Costa Oliveira  
a57816**

---

# Conteúdo

<b>1 Considerações</b>	<b>2</b>
1.1 Versão SNMP . . . . .	2
<b>2 Motivação</b>	<b>3</b>
<b>3 Especificações do Ambiente de Teste</b>	<b>4</b>
3.1 Especificações de Hardware . . . . .	4
3.1.1 Especificação das características do nosso micro-computador em estudo – Raspberry Pi B+ . . . . .	4
3.1.2 Esquematização para a obtenção dos valores de temperatura . . . . .	4
3.1.3 Especificações de Software do caso de teste . . . . .	5
3.2 Especificações de Software do caso de teste . . . . .	5
<b>4 Criação da MIB GREDES-SENSOR-MIB</b>	<b>6</b>
<b>5 Representação da informação dos sensores e implementação do Agente SNMP no micro-computador</b>	<b>9</b>
5.1 Representação da informação dos sensores a disponibilizar na tabela gestaoRedesSensorTable . . . . .	9
5.2 Obtenção dos valores medidos pelos sensores a disponibilizar na tabela gestaoRedesSensorTable . . . . .	10
5.3 Implementação em JAVA do Agente SNMP no micro-computador . . . . .	11
5.3.1 Classe JAVA GredesSensorMib . . . . .	11
5.3.2 Classe JAVA SNMPPAgent . . . . .	13
<b>6 Teste da ferramenta desenvolvida no trabalho prático 2 para o micro-computador</b>	<b>14</b>
<b>7 Considerações Finais</b>	<b>16</b>

# 1 Considerações

## 1.1 Versão SNMP

No decorrer do trabalho prático e respetivo relatório as referências ao protocolo SNMP estão diretamente associadas à versão 2c do mesmo.

De seguida apresenta-se uma breve descrição do mesmo:

SNMPv2c—The community-string based Administrative Framework for SNMPv2. SNMPv2c (the "c" stands for "community") is an Experimental Internet Protocol defined in RFC 1901, RFC 1905, and RFC 1906. SNMPv2c is an update of the protocol operations and data types of SNMPv2p (SNMPv2 Classic), and uses the community-based security model of SNMPv1.

---

*Cisco Website*

[http://www.cisco.com/c/en/us/td/docs/ios/12\\_2/configfun/configuration/guide/ffun\\_c/fcf014.html](http://www.cisco.com/c/en/us/td/docs/ios/12_2/configfun/configuration/guide/ffun_c/fcf014.html)

## 2 Motivação

De todas as razões para o recurso a um micro-computador, a redução do custo do hardware talvez seja o fator mais importante. Aliando ao custo ainda o tamanho e consumo energético reduzido deste tipo de dispositivos, temos a conjugação perfeita para a massificação de dispositivos de controlo remoto e automatização.

Ora, o presente trabalho prático tem como objetivo estudar a viabilidade de recurso ao protocolo SNMP, como forma de gestão deste tipo de dispositivos – que pelo seu preço e tamanho podem ser incluídos nas mais diversas utilizações dentro de uma empresa/instituição.

Tal como foi discutido numa das aulas práticas, o acesso à informação é importantíssimo numa grande organização. E quando temos acesso a essa informação, existe ainda o problema de a poder agrupar, filtrar e catalogar por forma a podermos retirar os dados que pretendemos.

É da necessidade anteriormente descrita que a recorrência ao protocolo SNMP poderá ser de grande auxílio, não só para gestores de rede ou infraestruturas mas até para gestor da empresa propriamente dita.

Com as ferramentas que existem para o protocolo SNMP, que pela sua idade e robustez possui uma enorme comunidade, podemos então simplificar o processo de gestão da informação – e isso é crucial.

Se for colocada a uma empresa ou grande organização a possibilidade de mapear, por exemplo, o consumo elétrico por edifício ou divisão, todos eles veriam essa possibilidade de bom grado. Contudo, ninguém quer despender grandes quantidades de tempo/recursos a retirar informação de um aglomerado de dados – é nessa situação que a conjugação micro-computadores / SNMP tem o seu proveito.

Considere o caso da Universidade do Minho e dos 10 Milhões de euros anuais gastos em despesas da rede elétrica. Se for necessário tomar medidas de contenção de despesas, é necessário ter conhecimento quais são os departamentos com maior consumo associado. E dentro de cada departamento quais os pisos ou divisões onde o gasto maior acontece.

Considere que em cada quadro elétrico é colocado um micro-computador (o do nosso caso de estudo por exemplo) e um ou mais sensores de corrente não invasivos de por exemplo 100Amperes.<sup>1</sup>. De uma forma barata, não evasiva e com recorrência a infra-estruturas de rede já existentes, temos então a possibilidade de uma monitorização em tempo real dos consumos.

Denote que este é apenas um dos muitos exemplos dos possíveis usos de micro-computadores e protocolos de gestão de rede robustos.

---

<sup>1</sup>Non-invasive AC current sensor 100A, <http://www.ptrobotics.com/sensores-de-corrente/2143-non-invasive-ac-current-sensor-100a.html>

### 3 Especificações do Ambiente de Teste

#### 3.1 Especificações de Hardware

##### 3.1.1 Especificação das características do nosso micro-computador em estudo – Raspberry Pi B+

Tendo um preço médio de mercado no valor de 23 € o modelo Raspberry Pi B+ apresenta as seguintes especificações de Hardware:

Release	Julho 2014
CPU	700 MHz single-core ARM1176JZF-S
GPU	Broadcom VideoCore IV @ 250 MHz
Memória	512 MB (partilhada com GPU)
On-board network	10/100 Mbit/s Ethernet
Periféricos baixo nível	8 * General purpose input-output (GPIO) connector

Tabela 1: Especificação de Hardware presente no micro-computador em estudo – Raspberry Pi B+

##### 3.1.2 Esquematização para a obtenção dos valores de temperatura

Relativamente à obtenção valores de temperatura do meio existem variadas formas de o conseguir. Assim, dada a preferência do nosso caso de estudo em formas económicas de obtenção de dados, decidi recorrer a um simples ambiente de prototipagem que incluía um Thermistor<sup>2</sup> de 10K, uma resistência de 10K e uma forma de medição de valores analógicos. A diferença entre a saída analógica (Vout) e o valor de referência (Vin 5v) dão-nos a possibilidade de conversão dessa mesma diferença em, por exemplo, graus centígrados.

Atente no seguinte esquema:

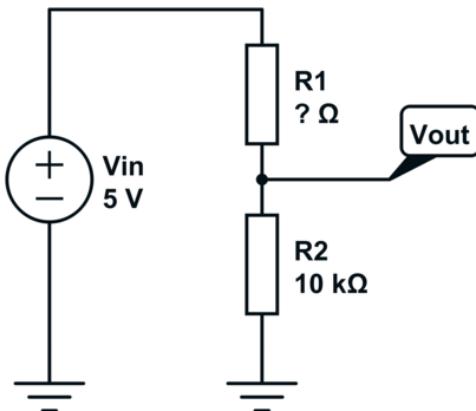


Figura 1: Esquema de ligação para a obtenção de valores de temperatura via entrada analógica

Ora, como iremos ver na especificação do nosso micro-computador, o mesmo tem uma falta de entradas analógicas. Tal foi resolvido recorrendo a um outro micro-computador que possuía, uma placa Arduino Mega 2560, baseado no Circuito Integrado ATmega2560. O mesmo possuía 54 pins para input/output digital.

Assim, o valor da temperatura propriamente dito é recolhido por uma entrada digital na placa Arduino Mega e enviado via comunicação Serial para o micro-controlador Raspberry B+, este ligado à rede.

Denote que a placa Arduino Mega podia ser simplesmente substituída por um chip que permita a conversão de entrada analógica para digital, como é o caso do MCP3008<sup>3</sup>, com um custo de poucos euros. Contudo, dado que possuía ao meu dispor uma placa que me permite a conversão sem custos adicionais, optei pela solução mais económica.

Atente na configuração final do ambiente de teste:

<sup>2</sup><http://www.ptrobotics.com/termicos/2674-temperature-sensor-with-steel-head.html>

<sup>3</sup><http://www.ptrobotics.com/circuitos-integrados-varios/2030-mcp3008-8-channel-10-bit-adc-with-spi-interface.html>

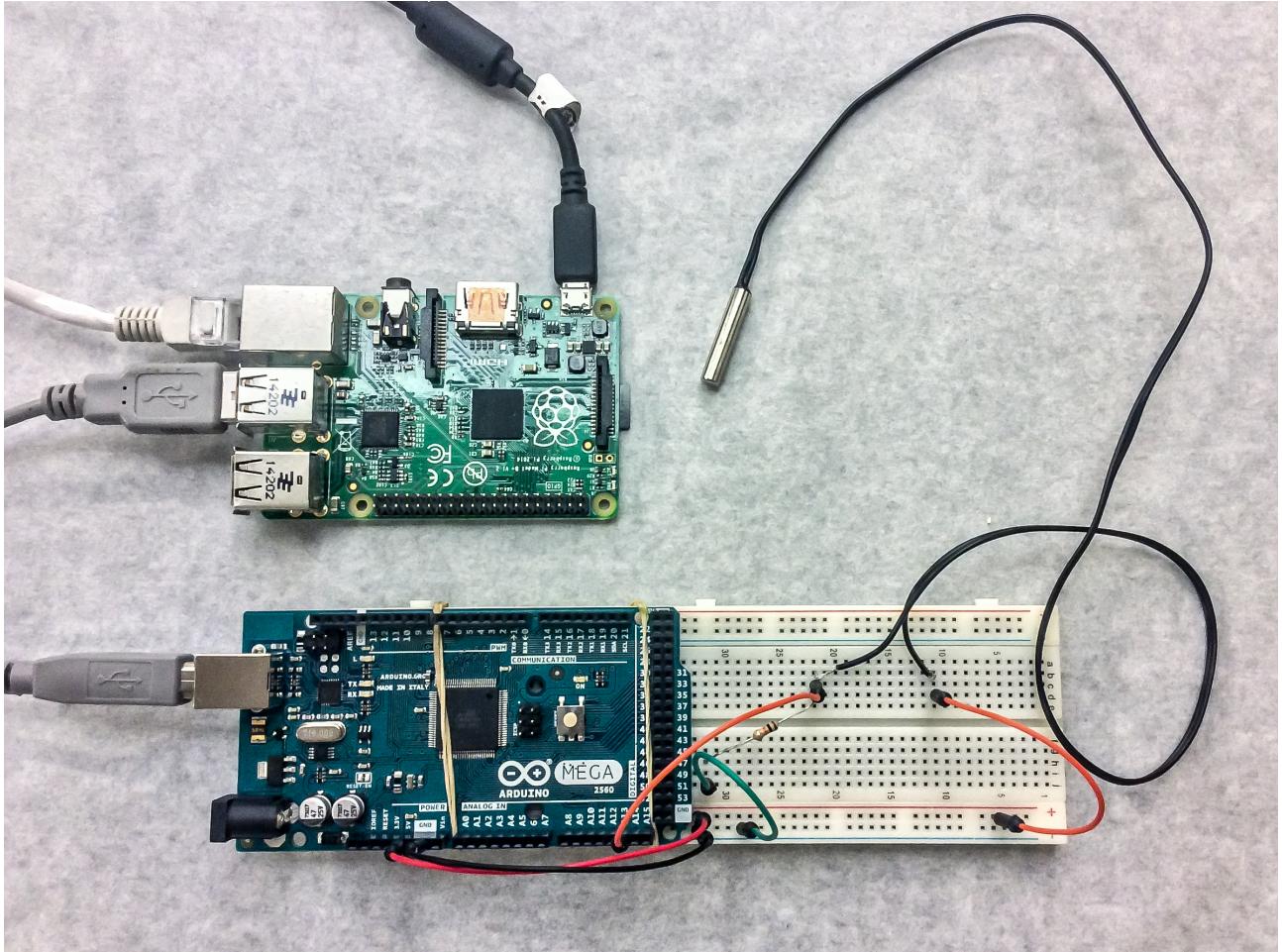


Figura 2: Configuração final do ambiente de teste

Temos portanto a placa Raspberry Pi B+ ligada à rede via Ethernet e ligada a placa Arduino Mega via porta USB (Serial Interface discutido anteriormente). Na placa de prototipagem encontra-se ligado o Thermistor Referido anteriormente.

### 3.1.3 Especificações de Software do caso de teste

## 3.2 Especificações de Software do caso de teste

Relativamente ao sistema operativo, temos uma grande variedade<sup>4</sup> de escolha relativamente ao nosso caso de estudo.

Assim, no nosso micro-computador, foram instalados:

- versão 8 do Raspbian GNU/Linux (jessie). (Sistema Operativo)
- verão "1.8.0" Java Runtime Environment (build 1.8.0-b132)
- Perl 5, version 20, subversion 2
- Python version 2.7.9
- Apache Ant(TM) version 1.9.4 compiled on October 7 2014

---

<sup>4</sup><https://www.raspberrypi.org/downloads>

## 4 Criação da MIB GREDES-SENSOR-MIB

Foi da necessidade de representar de uma forma simples a informação associada aos sensores, aliada também à vertente mais académica desta opção, que decidi criar a minha própria MIB em detrimento de outras. Assim, e recorrendo ao MIB Designer<sup>5</sup> criei a GREDES-SENSOR-MIB com o intuito de representar um conjunto de sensores presentes num Agente. De uma forma sucinta temos uma tabela (**gestaoRedesSensorTable**) contendo a informação dos sensores acessível a partir do OID **1 . 3 . 6 . 1 . 2 . 1 . 4 7 . 1 . 1 . 1**. Essa tabela contém uma sequência de (**gestaoRedesSensorTableEntry**) cada uma contendo a informação relativa a um sensor presente no agente. A informação dos sensores centra-se:

- no seu índice (**gestaoRedesSensorTableEntrySensorIndex**) do tipo inteiro e que está entre os valores 1 e 9999.
- no seu tipo (**gestaoRedesSensorTableEntrySensorType**) do tipo String e que contém mais informação relacionada com o tipo de sensor. Por exemplo no caso de termos mais do que um sensor de temperatura podemos descrever a sua posição ou propriedade que os distinga.
- no seu valor (**gestaoRedesSensorTableEntrySensorValue**) do tipo inteiro e que regista o valor medido pelo sensor físico.
- no timestamp do momento da medição (**gestaoRedesSensorTableEntrySensorValueTimeStamp**) do tipo TimeTicks e que contém o registo do tempo no qual a medição foi efetuada.

```
GREDES-SENSOR-MIB DEFINITIONS ::= BEGIN
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
32
33
34
35
36
37
38
39
40

IMPORTS
entityMIB
FROM ENTITY-MIB
MODULE-IDENTITY,
OBJECT-TYPE,
TimeTicks
FROM SNMPv2-SMI
OBJECT-GROUP
FROM SNMPv2-CONF;

gestaoRedesSensorMIB MODULE-IDENTITY
LAST-UPDATED "201602062029Z" -- Feb 6, 2016 8:29:00 PM
ORGANIZATION ""
CONTACT-INFO
"a57816@alunos.uminho.pt"
DESCRIPTION
"The MIB module for representing multiple sensor
entities supported by a single SNMP agent.
Copyright (C) Filipe Costa Oliveira (2016).
Gestao de Redes, Engenharia de Redes e Servicos ,
Universidade do Minho"
REVISION "201602062029Z" -- Feb 6, 2016 8:29:00 PM
DESCRIPTION
"Initial version."
-- 1.3.6.1.2.1.47.1 --
-- 1.3.6.1.2.1.47.1
::= { entityMIB 1 }

-- Scalars and Tables
--

gestaoRedesSensorMIBObjects OBJECT IDENTIFIER
-- 1.3.6.1.2.1.47.1.1
::= { gestaoRedesSensorMIB 1 }

gestaoRedesSensorTable OBJECT-TYPE
SYNTAX SEQUENCE OF GestaoRedesSensorTableEntry
MAX-ACCESS not-accessible
```

<sup>5</sup>MIB Designer 4.0.1 (Java SE Application JAR) <https://agentpp.com/download.html>

```

STATUS current
DESCRIPTION
"Lists the type, scale, and present value of a sensor listed in the Gestao de Redes Sensor Table."
-- 1.3.6.1.2.1.47.1.1.1 --
-- 1.3.6.1.2.1.47.1.1.1
 ::= { gestaoRedesSensorMIBObjects 1 }

gestaoRedesSensorTableEntrySensorIndex OBJECT-TYPE
SYNTAX INTEGER (1..9999)
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Indicates the index of the entry in the Gestao Redes Sensor Table."
-- 1.3.6.1.2.1.47.1.1.1.1 -- 1.3.6.1.2.1.47.1.1.1.2 --
-- 1.3.6.1.2.1.47.1.1.1.2
 ::= { gestaoRedesSensorTableEntry 2 }

gestaoRedesSensorTableEntrySensorType OBJECT-TYPE
SYNTAX OCTET STRING
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Indicates the type of data reported by the gestaoRedesSensorTableEntrySensorValue."
-- 1.3.6.1.2.1.47.1.1.1.3 --
-- 1.3.6.1.2.1.47.1.1.1.3
 ::= { gestaoRedesSensorTableEntry 3 }

gestaoRedesSensorTableEntry OBJECT-TYPE
SYNTAX GestaoRedesSensorTableEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"Table Entry for the Gestao de Redes Sensor Table"
INDEX {
  gestaoRedesSensorTableEntrySensorIndex }
-- 1.3.6.1.2.1.47.1.1.1
 ::= { gestaoRedesSensorTable 1 }

GestaoRedesSensorTableEntry ::= SEQUENCE {
  gestaoRedesSensorTableEntrySensorIndex          INTEGER ,
  gestaoRedesSensorTableEntrySensorType           OCTET STRING ,
  gestaoRedesSensorTableEntrySensorValue          INTEGER ,
  gestaoRedesSensorTableEntrySensorValueTimeStamp TimeTicks }

gestaoRedesSensorTableEntrySensorValue OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Reports the most recent measurement seen by the sensor."
-- 1.3.6.1.2.1.47.1.1.1.4
 ::= { gestaoRedesSensorTableEntry 4 }

gestaoRedesSensorTableEntrySensorValueTimeStamp OBJECT-TYPE
SYNTAX TimeTicks
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Indicates the age of the value reported by gestaoRedesSensorMIBEntrySensorValue."
-- 1.3.6.1.2.1.47.1.1.1.5
 ::= { gestaoRedesSensorTableEntry 5 }

```

— Notification Types	105
—	106
gestaoRedesSensorMIBEvents OBJECT IDENTIFIER	107
— 1.3.6.1.2.1.47.1.2	108
::= { gestaoRedesSensorMIB 2 }	109
— Conformance	110
—	111
gestaoRedesSensorMIBConf OBJECT IDENTIFIER	112
— 1.3.6.1.2.1.47.1.3	113
::= { gestaoRedesSensorMIB 3 }	114
— Groups	115
—	116
gestaoRedesSensorMIBGroups OBJECT IDENTIFIER	117
— 1.3.6.1.2.1.47.1.3.1	118
::= { gestaoRedesSensorMIBConf 1 }	119
— Compliances	120
—	121
gestaoRedesSensorMIBCompliances OBJECT IDENTIFIER	122
— 1.3.6.1.2.1.47.1.3.2	123
::= { gestaoRedesSensorMIBConf 2 }	124
gestaoRedesSensorMIBBasicGroup OBJECT-GROUP	125
OBJECTS {	126
gestaoRedesSensorTableEntrySensorIndex ,	127
gestaoRedesSensorTableEntrySensorType ,	128
gestaoRedesSensorTableEntrySensorValue ,	129
gestaoRedesSensorTableEntrySensorValueTimeStamp }	130
STATUS current	131
DESCRIPTION	132
"Describes and monitors the values of the GREDES SENSOR MIB gestaoRedesSensorTable ← entries of sensors."	133
— 1.3.6.1.2.1.47.1.3.1.1 —	134
— 1.3.6.1.2.1.47.1.3.1.1	135
::= { gestaoRedesSensorMIBGroups 1 }	136
END	137
	138
	139
	140
	141
	142
	143
	144
	145
	146

## 5 Representação da informação dos sensores e implementação do Agente SNMP no micro-computador

Criada a GREDES-SENSOR-MIB resta-nos portanto implementar o agente que irá responder aos pedidos relativos aos OIDs presentes na mesma.

Assim, e recorrendo ao AgentPro<sup>6</sup> iniciei o processo de implementação do agente na linguagem Java estendendo o agente base presente nas bibliotecas do SNMP4J<sup>7</sup>. Considero que este foi o passo mais moroso de todo o trabalho prático. A falta de informação clara e de exemplos torna a evolução lenta e sujeita a erros. O facto de a MIB criada recorrer a tabelas de OIDs torna a falta de informação ainda mais crítica.

Considere a implementação do agente como contendo dois grandes grupos de software desenvolvido: o primeiro relativo à obtenção dos valores dos sensores e o segundo relativo à implementação propriamente dita do agente SNMP em java.

### 5.1 Representação da informação dos sensores a disponibilizar na tabela gestao-RedesSensorTable

Para podermos implementar o agente e responder corretamente aos pedidos de informação, necessitamos primeiramente de ter acesso a essa informação. Assim, por forma a standardizar os dados relativos aos sensores dos quais o agente irá tratar a informação, criei um formato de ficheiro XML reconhecido pela aplicação final de nome **sensorCatalog.xml**. Atente no seu conteúdo:

```
<?xml version="1.0" encoding="UTF-8"?>
<SensorCatalogue>
  <Sensor>
    <Type>Temperature</Type>
    <Description>Network Management Work Assignment Temperature Sensor</Description>
    <SerialFilePath>snmp_sensor_temperature</SerialFilePath>
  </Sensor>
</SensorCatalogue>
```

1  
2  
3  
4  
5  
6  
7  
8

Como se pode confirmar, o ficheiro XML contém informação extra relativa a cada Sensor. O campo **SerialFilePath** contém os dados relativos à localização no filesystem do valor medido pelo sensor propriamente dito (a forma como obtemos esse valor será descrita posteriormente).

<sup>6</sup>AgenPro 4.0.3 (Java SE Application JAR) <https://agentpp.com/download.html>

<sup>7</sup>SNMP4J <http://www.snmp4j.org>

## 5.2 Obtenção dos valores medidos pelos sensores a disponibilizar na tabela gestaoRedesSensorTable

Como foi referido anteriormente aos sensores a serem disponibilizados na tabela gestaoRedesSensorTable está associado um ficheiro contendo o valor medido pelo sensor propriamente dito. Ora, como foi descrito na secção que especifica o ambiente de teste, para o nosso caso exemplo os valores são obtidos por comunicação serial entre o micro-computador Raspberry Pi B+ e a placa Arduino Mega.

Atente no código presente na placa Arduino Mega:

```
/*
Analog Input
Demonstrates analog input by reading an analog 10K temperature sensor on analog pin ←
A13 and
sending the converted Temperature to Celsius throw Serial

Created by Filipe Oliveira

This example code is in the public domain.

*/
#include <math.h>

double Thermistor(int RawADC) {
    double Temp;
    Temp = log(10000.0*((1024.0/RawADC)-1));
    Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );
    Temp = Temp - 273.15;           // Convert Kelvin to Celcius
    return Temp;
}

int sensorPin = A13;      // select the input pin for the potentiometer
int sensorValue = 0;      // variable to store the value coming from the sensor

void setup() {
    Serial.begin(9600);
}

void loop() {
    // read the value from the sensor:
    sensorValue = analogRead(sensorPin);
    Serial.println(Thermistor(sensorValue)); // display Celcius
    delay (1000);
}
```

Como se pode confirmar pelo código presente no método loop() o valor registado do Thermistor é enviado pela porta Serial a cada segundo, sendo que o tipo de variável enviado é **double**. Um valor exemplo enviado seria **16.77**.

Atente no código presente no micro-computador Raspberry Pi B+, responsável pela receção dos dados provenientes da placa Arduino Mega :

```
#!/usr/bin/python
import serial
import time

ser = serial.Serial('/dev/ttyACM0', 9600)
while True :
    f = open("snmp_sensor_temperature", "w")
    temperature = ser.readline().rstrip()
    print temperature
    f.write(temperature)
    f.close()
    time.sleep(1) # delays for 1 second
```

Tal como acontece no código presente na placa Arduino Mega, este código Python também recebe valores via comunicação serial com um intervalo de 1 segundo após receção. Ou seja, espera ativamente por dados, e após receber uma linha via o Serial Interface '**'dev/ttyACMO'**' aguarda 1 segundo até repetir o processo.

### 5.3 Implementação em JAVA do Agente SNMP no micro-computador

O recurso ao AgentPro<sup>8</sup> permitiu obter código base relativamente à Classe JAVA **GredesSensorMib**. É nela que contemos a informação relativa aos OIDs a serem registados no nosso **Agente**, dado que esta Classe estende a Classe **MOGroup** pertencente às bibliotecas do SNMP4J<sup>9</sup>.

### 5.3.1 Classe JAVA GredeSensorMib

Em adição aos métodos criados automaticamente pelo AgentPro foi necessário:

- adicionar as seguintes variáveis:
    - **<HashMap Integer, String>SerialFilePathMap** – que associa o índice de um sensor na tabela de sensores ao seu SerialFilePath.
    - **int actualIndex** – que contém o índice actual do último sensor adicionado à tabela.
  - adicionar os seguintes métodos:
    - **public int initializeSensorsXML ( String fileName )** – que inicializa a tabela de sensores com os sensores descritos no ficheiro XML (de formato explicado anteriormente) passado pela variável do tipo String e id fileName.

```
1 public int initializeSensorsXML ( String fileName ) {  
2     int numberSensors = 0;  
3     File fXmlFile = new File(fileName);  
4     DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();  
5     DocumentBuilder dBuilder;  
6     try {  
7         dBuilder = dbFactory.newDocumentBuilder();  
8         Document doc = dBuilder.parse(fXmlFile);  
9  
10        doc.getDocumentElement().normalize();  
11        System.out.println("-----");  
12        System.out.println("Reading Devices from :" + fileName );  
13        System.out.println("Root element :" + doc.getDocumentElement().  
14           getNodeName());  
15        NodeList nList = doc.getElementsByTagName("Sensor");  
16        System.out.println("-----");  
17  
18        for (int temp = 0; temp < nList.getLength(); temp++) {  
19            Node nNode = nList.item(temp);  
20            System.out.println("\nCurrent Element :" + nNode.getNodeName());  
21            if (nNode.getNodeType() == Node.ELEMENT_NODE) {  
22                Element eElement = (Element) nNode;  
23                String type = eElement.getElementsByTagName("Type").item(0).  
24                   getTextContent();  
25                String description = eElement.getElementsByTagName("Description").  
26                   item(0).getTextContent();  
27                String filePath = eElement.getElementsByTagName("SerialFilePath").  
28                   item(0).getTextContent();  
29                System.out.println("\tType : " + type);  
30                System.out.println("\tDescription : " + description);  
31                System.out.println("\tSerialFilePath : " + filePath);  
32                numberSensors++;  
33                /*SNMP OIDs initilization for Table Row*/  
34                Variable[] vars = new Variable[4];  
35                vars[0]=new Integer32(actualIndex);  
36                vars[1]=new OctetString(type);  
37                vars[2]=new Integer32(0);  
38            }  
39        }  
40    }  
41 }
```

<sup>8</sup>AgenPro 4.0.3 (Java SE Application JAR) <https://agentpp.com/download.html>

<sup>9</sup>SNMP4J <http://www.snmp4j.org>

```

vars[3]=new TimeTicks();
MOMutableTableModel model = (MOMutableTableModel) ←
    gestaoRedesSensorTableEntry.getModel();
model.addRow(new DefaultMOMutableRow2PC(new OID(new int[] {←
    actualIndex}), vars));
SerialFilePathMap.put(actualIndex, filePath);
actualIndex++;
}
}
System.out.println("\n-----");
} catch (ParserConfigurationException e) {
e.printStackTrace();
} catch (SAXException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
return numberSensors;
}

```

- **public void add\_sensor( String sensorType, int value)** – que adiciona um novo sensor à tabela de sensores.

```

public void add_sensor( String sensorType, int value){
Variable[] vars = new Variable[4];
vars[0]=new Integer32(actualIndex);
vars[1]=new OctetString(sensorType);
vars[2]=new Integer32(value);
vars[3]=new TimeTicks();
MOMutableTableModel model = (MOMutableTableModel) ←
    gestaoRedesSensorTableEntry.getModel();
model.addRow(new DefaultMOMutableRow2PC(new OID(new int[] {actualIndex}), ←
    vars));
actualIndex++;
}

```

- **public void updateSensorsData (MOserver server, OctetString context)** – que realiza o atualização aos dados dos sensores presentes na tabela com base no filePath presente na variável **SerialFilePathMap**. Relativamente à atualização dos valores presentes nos campos da tabela, tentei que o mesmo fosse apenas realizado anteriormente a pedidos get ao OID específico a ser atualizado. Contudo, foi infrutífera a minha alteração aos métodos get dos OIDs (os valores não eram atualizados). Foi então necessário realizar uma atualização constante de todos os valores mutáveis dos sensores presentes na tabela de sensores.

```

public void updateSensorsData (MOserver server, OctetString context){
for (Integer index : SerialFilePathMap.keySet()){
String fileToOpen = this.SerialFilePathMap.get(index);
System.out.println ("Getting sensor value from :" + fileToOpen);
FileInputStream fstream;
try {
fstream = new FileInputStream(fileToOpen);
BufferedReader br = new BufferedReader(new InputStreamReader(fstream))←
;
String strLine;
if ((strLine = br.readLine()) != null) {
Double temperatura = Double.parseDouble(strLine);
MOMutableTableModel model = (MOMutableTableModel) ←
    gestaoRedesSensorTableEntry.getModel();
int pos = 1;
for (Iterator<MOTableRow> i = model.iterator(); i.hasNext() && pos ←
    <= index ;) {
MOMutableTableRow mot = (MOMutableTableRow) i.next();
if (pos == index) {

```

```
mot.setValue( 2 , new Integer32(temperatura.intValue()) );
mot.setValue( 3 , SNMPv2MIB.getSysUpTime(context).get() );
System.out.println ("\tUpdating " + index + " to " + temperatura←
    .intValue() + " C at:" + SNMPv2MIB.getSysUpTime(context).←
        get().toString());
}
}
}
br.close();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
}
}
```

### 5.3.2 Classe JAVA SNMPPAgent

Relativamente à implementação do Agente SNMP foi necessário criar a Classe JAVA **SNMPAgent** que estende a Classe **Base Agent** pertencente às bibliotecas do SNMP4J<sup>10</sup>.

As alterações aos métodos presentes no BaseAgent são mais visíveis nos métodos:

- **public void start()** – que inicializa as respostas por parte do agente a pedidos, e invoca o método updateSensorsData da Classe GRedesSensorMIB a cada segundo.

```
public void start()  {
    super.run();
    if (super.agentState == super.STATE_RUNNING){
        System.out.println("Agent running on: " + this.address);
    }
    while(super.agentState == super.STATE_RUNNING){
        sensors.updateSensorsData(this.getServer(), this.getDefaultContext());
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

- **public static void main(String[] args)** – que cria o agente que responderá a pedidos na porta 2001, inicializa-o e regista os Manageable Objects (OIDs da GREDES-SENSOR-MIB, e invoca por fim o método start).

```
public static void main(String[] args) throws IOException, ←
    DuplicateRegistrationException {
    SNMPAgent agent = null;
    agent = new SNMPAgent("0.0.0.0/2001");
    agent.sensors = new GRedesSensorMib(DefaultMOFactory.getInstance());
    agent.initialize();
    agent.sensors.registerM0s(agent.getServer(), agent.getDefaultContext());
    int totalInitializedSensors = agent.sensors.initializeSensorsXML("←
        sensorCatalog.xml");
    System.out.println("Initialized GRedesSensorMib Table with " + ←
        totalInitializedSensors + " sensors");
    agent.start();
}
```

<sup>10</sup>SNMP4J <http://www.snmp4j.org>

## 6 Teste da ferramenta desenvolvida no trabalho prático 2 para o micro-computador

Dado que dispunha do trabalho prático 2 por forma a analisar visualmente o comportamento do micro-processador, achei que seria um bom exercício de validação da correta implementação do Agente SNMP no caso de estudo.

Os pedidos SNMP serão realizados para o endereço IPv4 192.168.1.115 (Raspberry Pi B+) na porta 2001. Atente na execução do seguinte comando:

```
snmpwalk -v2c -c public 192.168.1.115:2001 1.3.6.1.2.1.47.1.1.1.1  
SNMPv2-SMI::mib-2.47.1.1.1.2.1 = INTEGER: 1  
SNMPv2-SMI::mib-2.47.1.1.1.3.1 = STRING: "Temperature"  
SNMPv2-SMI::mib-2.47.1.1.1.4.1 = INTEGER: 41  
SNMPv2-SMI::mib-2.47.1.1.1.5.1 = Timeticks: (197161) 0:32:51.61
```

1  
2  
3  
4  
5

Temos portanto interesse em analisar o OID 1.3.6.1.2.1.47.1.1.1.4.1, onde está registada a temperatura do sensor de temperatura do nosso sistema. Por fim de simular a evolução do registo de diferentes temperaturas coloquei o nosso Thermistor num copo com água quente. Foi realizado Fetching a cada 1 segundo sendo que os resultados gráficos serão de seguida apresentados.

Denote que para ”compilar” corretamente o projeto basta executar a seguinte sequência de comandos:

```
ant clean && ant && ant agent
```

1

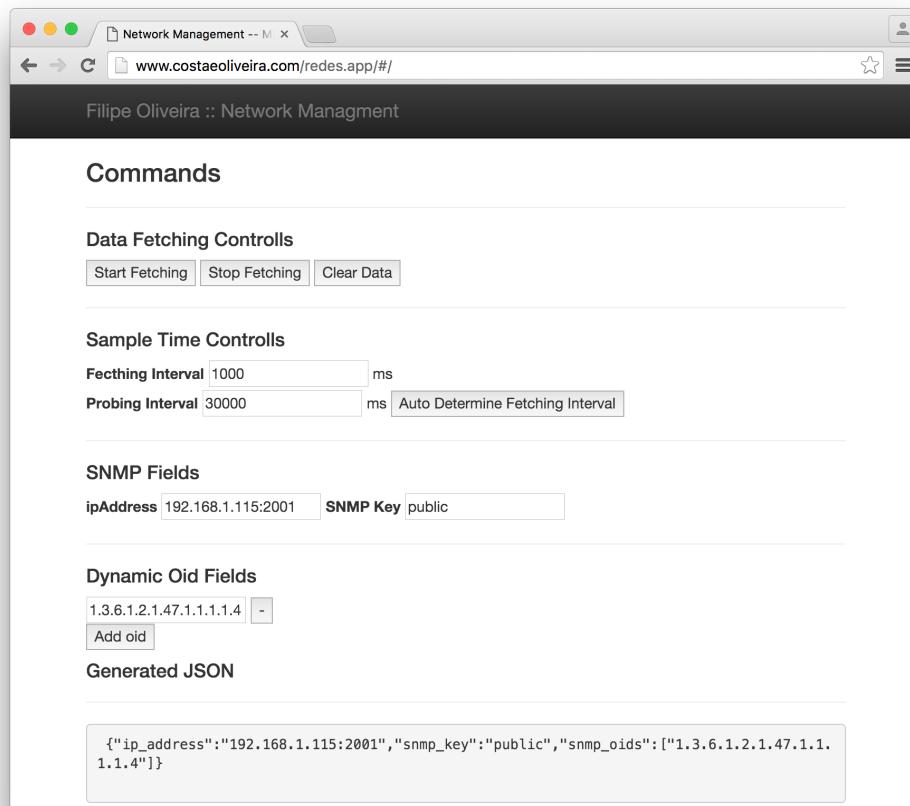


Figura 3: Configuração para a análise do OID 1.3.6.1.2.1.47.1.1.1.4.1

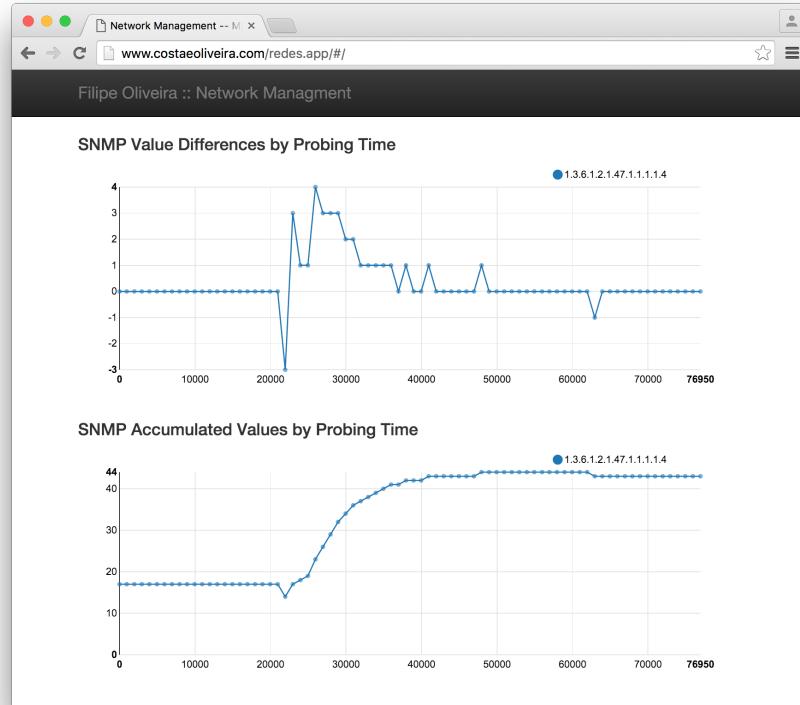


Figura 4: Resultado gráfico do feching do OID **OID 1.3.6.1.2.1.47.1.1.1.4.1** no agente remoto

```

Terminal Shell Edit View Window Help filipeoliveira — pi@raspberrypi: ~/GREDES-DEVICE-MIB — bash — 108x44
17.88 [java] Getting sensor value from :snmp_sensor_temperature
17.88 [java] Updating 1 to 17 °C at:0:29:18.99
17.88 [java] Getting sensor value from :snmp_sensor_temperature
17.88 [java] Updating 1 to 17 °C at:0:29:20.00
17.88 [java] Getting sensor value from :snmp_sensor_temperature
17.88 [java] Updating 1 to 14 °C at:0:29:21.00
17.88 [java] Getting sensor value from :snmp_sensor_temperature
17.88 [java] Updating 1 to 17 °C at:0:29:22.00
17.88 [java] Getting sensor value from :snmp_sensor_temperature
17.88 [java] Updating 1 to 18 °C at:0:29:23.01
17.79 [java] Getting sensor value from :snmp_sensor_temperature
17.79 [java] Updating 1 to 19 °C at:0:29:24.01
14.98 [java] Getting sensor value from :snmp_sensor_temperature
17.79 [java] Updating 1 to 23 °C at:0:29:25.01
18.05 [java] Getting sensor value from :snmp_sensor_temperature
19.98 [java] Updating 1 to 26 °C at:0:29:26.01
23.32 [java] Getting sensor value from :snmp_sensor_temperature
26.88 [java] Updating 1 to 29 °C at:0:29:27.02
29.99 [java] Getting sensor value from :snmp_sensor_temperature
32.53 [java] Updating 1 to 32 °C at:0:29:28.02
34.66 [java] Getting sensor value from :snmp_sensor_temperature
36.35 [java] Updating 1 to 34 °C at:0:29:29.02
37.77 [java] Getting sensor value from :snmp_sensor_temperature
38.92 [java] Updating 1 to 36 °C at:0:29:30.02
39.87 [java] Getting sensor value from :snmp_sensor_temperature
40.63 [java] Updating 1 to 37 °C at:0:29:31.03
41.28 [java] Getting sensor value from :snmp_sensor_temperature
41.83 [java] Updating 1 to 38 °C at:0:29:32.03
42.27 [java] Getting sensor value from :snmp_sensor_temperature
42.61 [java] Updating 1 to 39 °C at:0:29:33.03
42.95 [java] Getting sensor value from :snmp_sensor_temperature
43.18 [java] Updating 1 to 40 °C at:0:29:34.03
43.40 [java] Getting sensor value from :snmp_sensor_temperature
43.52 [java] Updating 1 to 41 °C at:0:29:35.04
43.75 [java] Getting sensor value from :snmp_sensor_temperature
43.86 [java] Updating 1 to 41 °C at:0:29:36.04
43.86 [java] Getting sensor value from :snmp_sensor_temperature
43.98 [java] Updating 1 to 42 °C at:0:29:37.04
44.09 [java] Getting sensor value from :snmp_sensor_temperature
44.09 [java] Updating 1 to 42 °C at:0:29:38.04
44.09 [java] Getting sensor value from :snmp_sensor_temperature
44.09 [java] Updating 1 to 42 °C at:0:29:39.05
44.09 [java] Getting sensor value from :snmp_sensor_temperature
44.21 [java] Updating 1 to 43 °C at:0:29:40.05

```

Figura 5: Evolução dos dados do Agente via ligação ssh ao micro-computador

## **7 Considerações Finais**

Acredito que em termos académicos, este trabalho experimental enriqueceu bastante a minha experiência tanto com protocolos orientados à gestão de redes como na gestão da implementação de MIBs próprias.

O facto de bibliotecas como o caso da SNMP4J serem open-source facilitam-nos o acesso a informação e capacidades computacionais que doutra forma não teríamos, mas leva-nos também à dura constatação que quando não existe uma "obrigação" comercial em documentar corretamente "features" específicas estas mesmas são de difícil compreensão.

Acredito que assim como eu perdi bastante tempo na correta implementação de por exemplo tabelas de OIDs recorrendo à biblioteca do SNMP4J, também muitos acham a compreensão de invocação de métodos algo mal documentado.

Não existem muitos artigos/tutoriais nesta área, apesar de ser de enorme interesse académico e empresarial, o que me leva a concluir que existe uma forte possibilidade de criação de ferramentas nesta área – quer open-source que de licença comercial.