

# **SIMPLE NETWORK MANAGMENT**

## **PROTOCOL**

### **(SNMP)**

#### **1.Rezumat:**

Proiectul își propune realizarea unui instrument de monitorizare a unei rețele. Soluția propusă se bazează pe protocolul SNMP. Scopul acestui proiect este de a observa, în timp real, starea de funcționare a echipamentelor de comunicație sau a echipamentelor destinate anumitor servicii. Printre lucrurile observabile se enumeră traficul de download și upload, memoria RAM disponibilă, procentul de încărcare al CPU, spațiul liber pe hard disk, dar și alte valori care pot fi exprimate prin unități de măsură (ex: temperatură).

#### **2.De ce avem nevoie de SNMP?**

Nevoia de a monitoriza dispozitivele conectate la rețelele de calculatoare, care creșteau rapid la sfârșitul anilor 1980, a dus la dezvoltarea protocolului SNMP (Simple Network Management Protocol). Scopul principal al acestuia a fost să faciliteze managementul și monitorizarea echipamentelor de rețea, precum routere, switch-uri, servere și alte dispozitive, într-o manieră standardizată și eficientă. SNMP a fost creat pentru a rezolva problemele complexe de administrare a rețelelor, oferind un mod simplu și uniform de a gestiona și controla infrastructura rețelelor de date.

Informațiile obținute prin monitorizare pot fi folosite pentru a determina dacă resursele implicate sunt utilizate corespunzător, pentru a verifica cum lucrează echipamentele care sunt folosite în rețea, a urmări activitatea în cadrul unei rețele și pentru a identifica probleme care apar și de a lua măsurile necesare pentru rezolvarea lor.

#### **3.Geeky description**

SNMP este un protocol, care funcționează la nivelul aplicație al modelului TCP/IP. Acest protocol a apărut din nevoia companiilor de a evita situațiile neplăcute în

care componentele unei rețele să nu funcționeze la parametri optimi.

Majoritatea implementărilor folosesc UDP pentru transferul de mesaje, deoarece este considerat acceptabilă pierderea de pachete în comparație cu funcțiile pe care trebuie să le îndeplinească entitățile administrate.

UDP(User Datagram Protocol) este unul din principalele protocoale de comunicare folosit pentru a trimite mesaje (transmis ca o datagrama sub forma unui pachet de date) către alte gazde prin intermediul IP network.

#### 4.Arhitectura protocolului:

Arhitectura SNMP se bazează pe modelul manager/agent. Modelul constă din următoarele componente (componente SNMP):

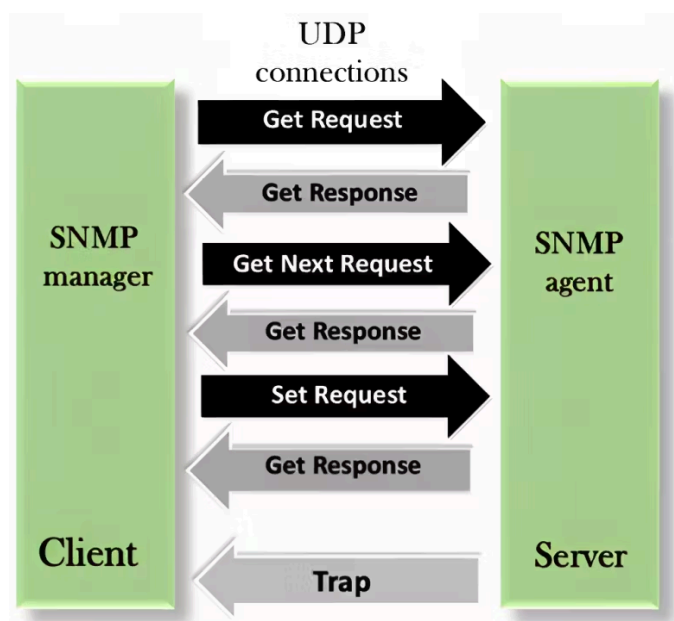


Figura 4.1: Comunicarea între managerul și agentul SNMP

**Manager:** Un manager SNMP furnizează interfața între sistemul de management al rețelei și administratorul rețelei responsabil de sistem. În aplicațiile de monitorizare la distanță, managerul este de obicei un computer sau o stație principală echipată cu un software de management al rețelei SNMP care colectează și procesează informații de la mai mulți agenți SNMP la distanță.

**Agent:** Un agent SNMP oferă interfața dintre manager și dispozitivele monitorizate. În aplicațiile de monitorizare a alarmelor, acesta ar fi o unitate RTU (Unitate de Telemetrie la Distanță) capabilă de SNMP, care gestionează obiectele (componentele) de pe el însuși. RTU-urile pot trimite trap-uri managerului când apare un eveniment de alarmă, iar managerul interoghează continuu RTU-urile pentru colectarea de date.

**Obiecte Gestionate:** Obiectele gestionate sau dispozitivele gestionate sunt obiecte logice în software care corespund, în general, unor lucruri fizice, cum ar fi intrări și ieșiri. În monitorizarea la distanță, un RTU va avea mai multe intrări pentru senzori sau conexiuni seriale și ieșiri pentru a controla echipamente la locații care nu au capacități SNMP.

**Baza de Date:** Baza de date funcționează ca un fel de dicționar pentru a ajuta managerul și agentul să comunice. Se numește MIB și va fi discutată mai detaliat mai târziu.

### **Cum să Înțelegi Baza de Informații de Management (MIB) și Identificatorii de Obiecte (OID)**

Managerul și agentul folosesc o Bază de Informații de Management și un set relativ mic de comenzi pentru a schimba informații. Fișierul MIB este un fișier text formatat care definește obiectele de date utilizate de un anumit echipament. Fiecare obiect de date este definit cu un Identificator de Obiect unic (OID), iar managerul și agentul știu la ce corespunde OID-ul prin intermediul MIB-ului. Pentru a defini OID, acesta este o adresă folosită pentru a identifica în mod unic dispozitivele gestionate și stările acestora. Vrei să afli citirea temperaturii de la un senzor din instalația ta de la distanță, de pe vârful muntelui? Există un OID pentru asta.

De fiecare dată când se dorește adăugarea unui nou tip de dispozitiv în rețea trebuie să încarci un fișier MIB în managerul SNMP pentru ca acesta să știe cum să comunice cu agenții săi. Ei ar trebui să aibă deja MIB-ul încărcat de la producător. Instalarea fișierelor MIB în manager este similară cu instalarea driverelor pe un PC pentru a comunica, de exemplu, cu o imprimantă.

MIB-ul este organizat într-o structură de arbore (vezi Figura 4.2), cu variabile individuale, cum ar fi starea punctului sau descrierea, reprezentate ca frunze pe ramuri. Un identificator numeric lung sau Identificator de Obiect este utilizat pentru a distinge în mod unic fiecare variabilă în MIB și trap-uri SNMP.

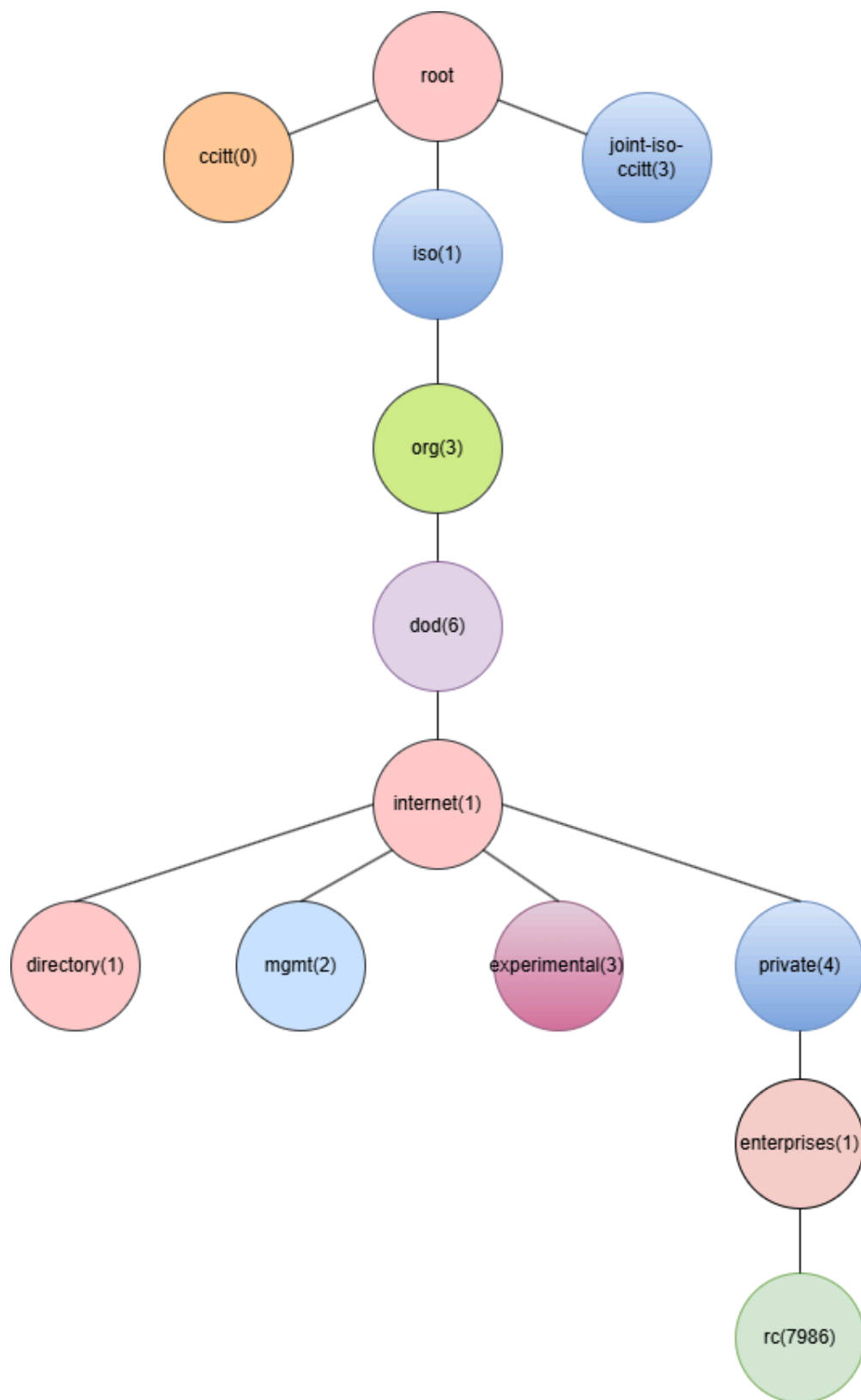


Figura 4.3: Structura arbore MIB standard

### **GET: Cererea de informații în orice moment**

Pentru a obține informații de stare de la agent, managerul poate emite mesaje Get și GetNext pentru a solicita informații pentru o variabilă specifică. Odată ce un mesaj Get sau GetNext este primit, agentul va emite un mesaj GetResponse către manager. Acesta va conține fie informațiile solicitate, fie o eroare care explică de ce cererea nu poate fi procesată.

### **SET: Controlează dispozitivele de la distanță**

Un mesaj Set permite managerului să solicite o modificare a unui obiect gestionat. Agentul va răspunde apoi cu un mesaj GetResponse dacă modificarea a fost efectuată sau cu o eroare care explică de ce modificarea nu poate fi realizată.

### **TRAP: Cel mai comun mesaj SNMP**

Mesajele Trap sunt inițiate de agent și sunt trimise către manager atunci când are loc un eveniment important. Acest lucru face ca trap-urile să fie perfecte pentru raportarea alarmelor către manager - în loc să aștepți o solicitare de stare din partea managerului atunci când ajunge să interogheze agentul.

## **Înțelegerea Tipurilor de Pachete și Structurii SNMP**

Protocoalele de monitorizare la distanță serială de bază, precum TBOS, sunt orientate pe byte, cu un singur byte schimbat pentru a comunica. Protocoalele de telemetrie serială extinse, cum ar fi TABS, sunt orientate pe pachete, având pachete de bytes schimbate pentru a comunica.

Aceste pachete conțin un header, date și un byte de verificare. SNMP este, de asemenea, un protocol orientat pe pachete. Pachetele sunt compuse din tipurile de mesaje discutate anterior: *Get*, *GetNext*, *GetResponse*, *Set* și *Trap*.

Fiecare pachet, sau legătură de variabile (*variable binding*), conține un identificator, un tip și o valoare (dacă este un *Set* sau un *Response*). Agentul folosește MIB-ul său pentru a determina dacă obiectul este administrat și schimbabil (dacă procesează un *Set*). Managerul folosește MIB-ul său pentru a afișa numele citibil al variabilei și uneori pentru a interpreta valoarea acesteia pentru orice tehnician care ar putea trebui să ia măsuri corective.

## **Modelul de Comunicare Stratificat al SNMP**

Pentru a trimite informații, SNMP folosește un model de comunicare stratificat.

- **Stratul 1 - Strat aplicație (stratul SNMP)**

- **Stratul 2 - Strat transport (UDP)**
- **Stratul 3 - Strat Internet (IP)**
- **Stratul 4 - Strat Interfață rețea** (de ex., pereche răsucită de cupru, co-axial RG58 sau fibră)

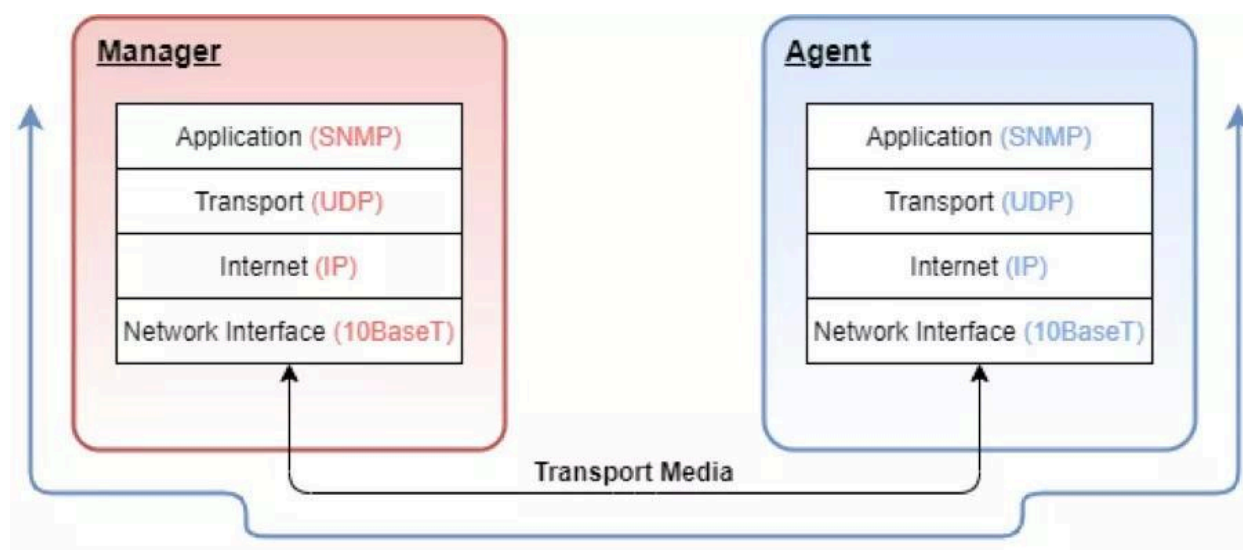


Figura 4.3: Cum sunt transportate datele prin SNMP

Deși acest model pe mai multe straturi poate părea puțin ambiguu, el izolează eficient sarcinile de comunicare și, în final, ajută la proiectarea și implementarea unei rețele.

### Exemplu de Transport SNMP

Pentru a ilustra funcția acestui model stratificat, să ne uităm la o singură solicitare SNMP *GET* din perspectiva agentului.

1. Managerul SNMP dorește să știe care este numele de sistem al Agentului și pregătește un mesaj *GET* pentru OID-ul corespunzător.
2. Apoi trimite mesajul către stratul UDP. Stratul UDP adaugă un bloc de date care identifică portul managerului la care ar trebui să fie trimis pachetul de răspuns.
3. Pachetul este apoi transmis la stratul IP, unde blocul de date conținând adresa IP și adresele Media Access ale managerului și agentului sunt adăugate.
4. Întregul pachet asamblat este transmis la stratul Interfață Rețea. Stratul Interfață Rețea verifică accesul și disponibilitatea media. Apoi plasează pachetul pe mediu pentru transport.

5. După ce își face drumul prin punți și prin routere bazat pe informațiile IP, pachetul ajunge în cele din urmă la agent.
6. Aici trece prin aceleași patru straturi exact în ordinea opusă față de cea de la manager.

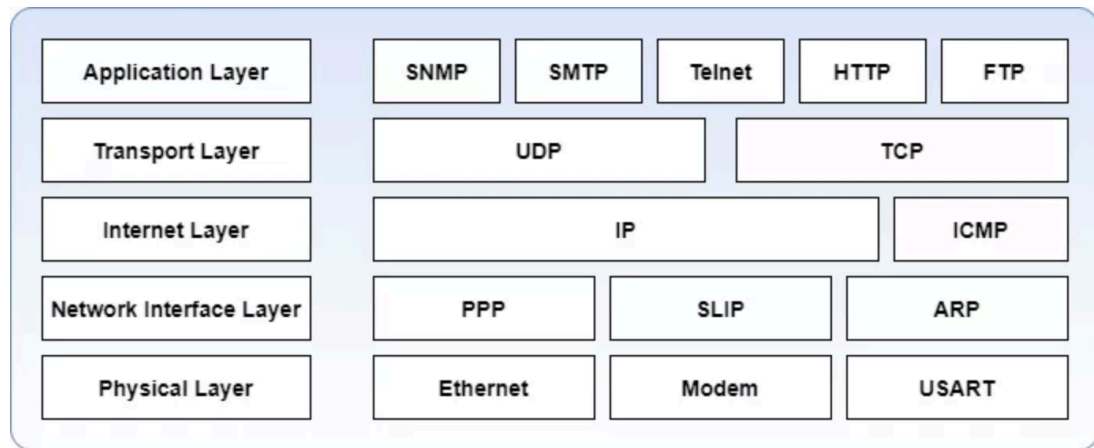


Figura 4.4: Modelul de rețea pe 5 straturi sau Suita de Protocoale Internet

Fiecare pas poate fi verificat independent până când toate etapele funcționează corect pentru o comunicare de la un capăt la altul.

### Care este structura mesajelor SNMP?

Mesajele SNMP sunt structurate cu componente-cheie care facilitează comunicarea între sistemele de management al rețelei și dispozitive. Fiecare mesaj include:

- Un identificator de versiune.
- Un șir de comunitate sau parametri de securitate (în funcție de versiunea SNMP).
- O Unitate de Date de Protocol (PDU).

PDU-ul conține operațiunea ce urmează să fie efectuată (cum ar fi GET sau SET), împreună cu câmpuri precum ID-ul cererii, starea erorii și indexul erorii. De asemenea, include *variable bindings* (VarBinds), care sunt perechi de Identificatori de Obiect (OID-uri) și valorile lor corespunzătoare, specificând datele ce urmează să fie accesate sau modificate. Acest format structurat asigură un management al rețelei precis și eficient.

## **5.Componenta hardware:**

Sistemul de monitorizare a rețelei folosește protocolul SNMP (Simple Network Management Protocol) pentru a colecta și centraliza informațiile despre starea echipamentelor de rețea și a serverelor. Stația de management („Management Station”) joacă un rol central, conectându-se la echipamentele de rețea (cum ar fi switch-urile, routerele și firewall-urile) și la servere pentru a monitoriza parametri esențiali de funcționare.

Stația de management primește date prin protocolul SNMP și este configurată să trimită alerte în cazul detectării unor erori, anomalii sau scăderi de performanță. Utilizatorul poate astfel interveni rapid pentru remedierea problemelor, asigurând continuitatea operațiunilor și o funcționare optimă a infrastructurii IT.

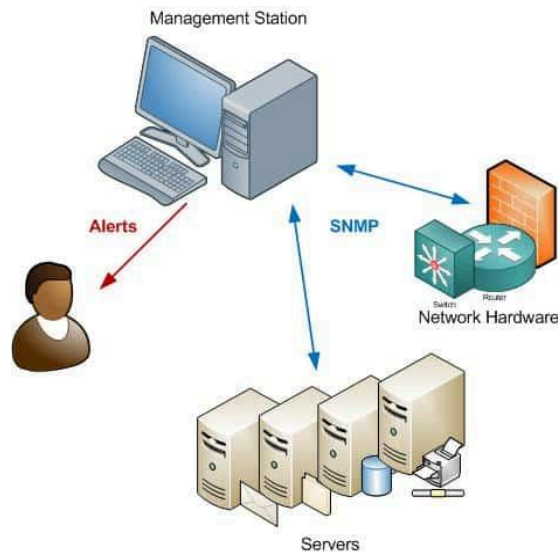


Figura 6: Arhitectura de Monitorizare Hardware prin SNMP



## 6.Componenta software:

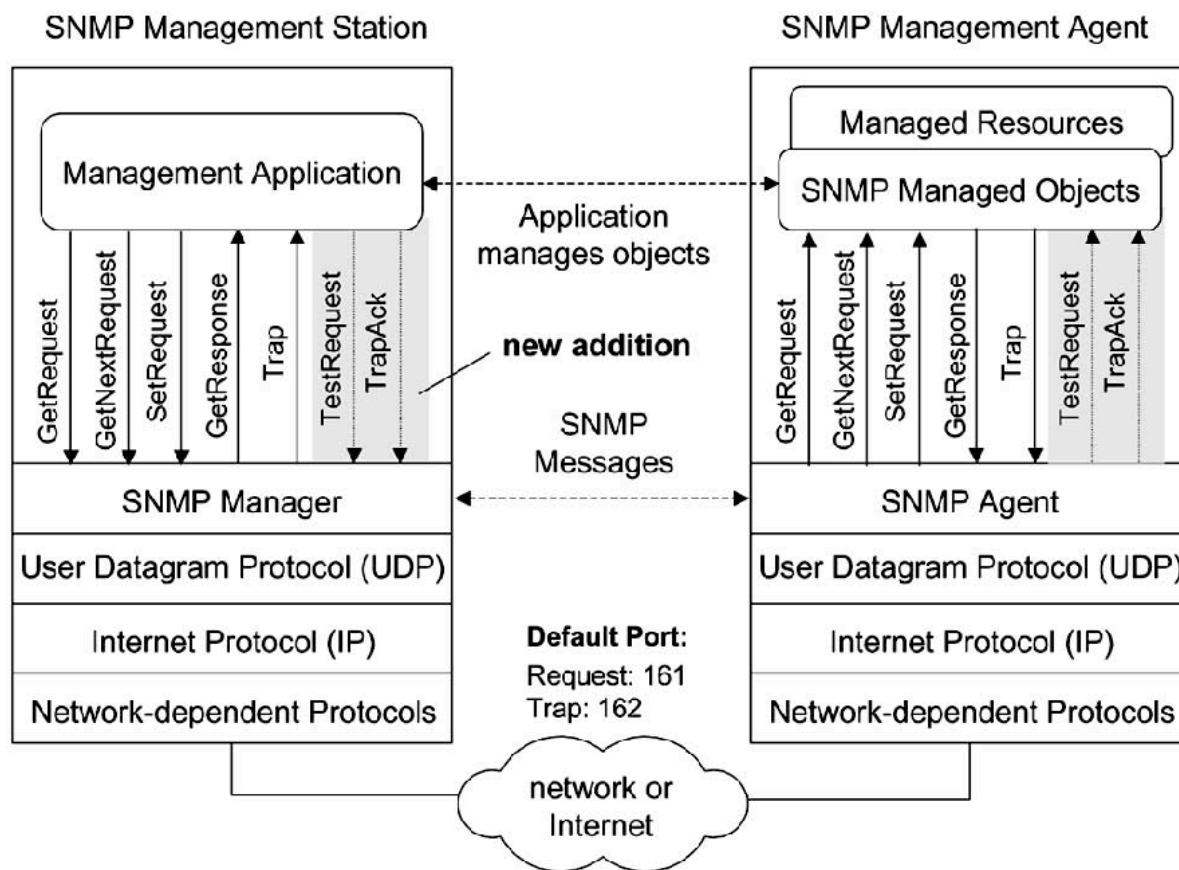


Figura 7: Imaginea arhitecturii comunicării SNMP

Diagrama ilustrează arhitectura software a unui sistem SNMP (Simple Network Management Protocol), în care Stația de Management SNMP comunică cu Agentul de Management SNMP pentru a monitoriza și administra resursele de rețea.

Stația de Management SNMP conține aplicația de management, care coordonează și monitorizează resursele rețelei, și componenta SNMP Manager, care trimite cereri și primește răspunsuri de la SNMP Agent. Comunicarea se face prin protocolul UDP (User Datagram Protocol).

În partea dreaptă a diagramei, Agentul SNMP interacționează cu resursele gestionate (Managed Resources) și cu obiectele administrate (SNMP Managed Objects) pentru a obține sau modifica date, pe care le transmite ulterior către stația de management.

## 7. Detaliile implementării

Până acum, am discutat detaliile tehnice referitoare la structura protocolului SNMP. Acum, pentru a avansa către implementarea acestuia, este necesar să stabilim o nouă structură pentru arborele MIB, adaptată specific rețelei noastre. În rețeaua noastră, vom avea patru tipuri diferite de dispozitive: calculatoare, imprimante, switch-uri și routere.

Din acest motiv, nodul `rc` din MIB va necesita o modificare (vezi Figura 7.1).

O implementare elegantă a SNMP implică utilizarea unui fișier MIB dedicat pentru fiecare dispozitiv din rețea. Astfel, atunci când dorim să adăugăm un nou dispozitiv, managerul va compila fișierul MIB corespunzător, având astfel acces la informațiile necesare pentru gestionarea eficientă a datelor. Această abordare oferă numeroase avantaje, precum claritate, extensibilitate și ușurință în întreținere. Cu toate acestea, complexitatea și dificultatea de integrare a acestei metode pot reprezenta provocări

semnificative. Din această cauză, ne propunem să utilizăm un singur arbore MIB, care va conține informațiile tuturor dispozitivelor din rețea, simplificând astfel gestionarea și monitorizarea acestora.

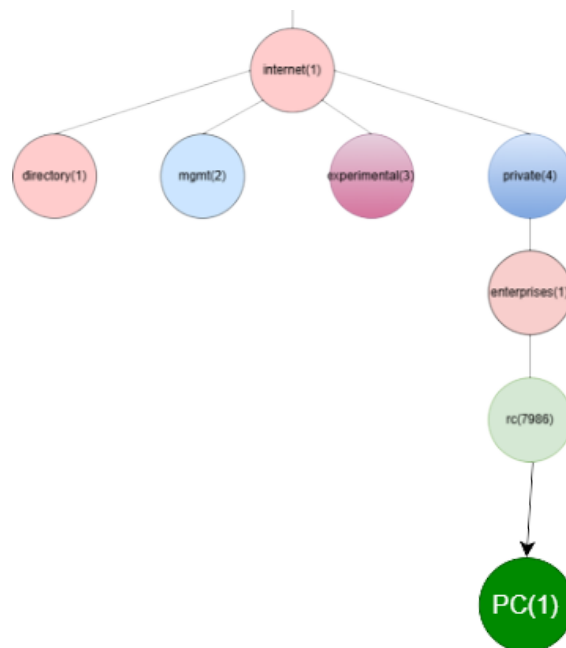


Figura 7.1: Structura arborelui MIB

În continuare, trebuie să stabilim care sunt metricile de interes pentru fiecare dispozitiv:

→ Pentru PC-ul nostru:

- ◆ CPU usage
- ◆ RAM % usage
- ◆ RAM GB usage
- ◆ Temperatura
- ◆ Nume

## **7.1 Structura generală a OID-urilor**

Pornind de la structura MIB, vom crea o ramură dedicată metricilor pentru fiecare tip de dispozitiv. Prefixul OID-ului de bază rămâne:

- 1.3.6.1.4.1.7986

Fiecare dispozitiv va avea o subramură specifică, iar metricile corespunzătoare vor fi definite sub aceasta.

## **7.2 Definirea OID-urilor pentru fiecare metrică**

Pentru dispozitivele de tip PC, vom folosi ramura **1** sub prefixul de bază. Fiecare metrică va avea un identificator unic, astfel:

- Temperatura: 1.3.6.1.4.1.7986.1.1
- Nume: 1.3.6.1.4.1.7986.1.2
- RAM % usage: 1.3.6.1.4.1.7986.1.3
- RAM GB usage: 1.3.6.1.4.1.7986.1.4
- CPU % usage: 1.3.6.1.4.1.7986.1.5

## **7.3 Rezumatul structurii OID**

Structura completă pentru PC (PC):

1.3.6.1.4.1.7986 - rc

|

|-- 1 - computers

|

|-- 1 - Temperatura

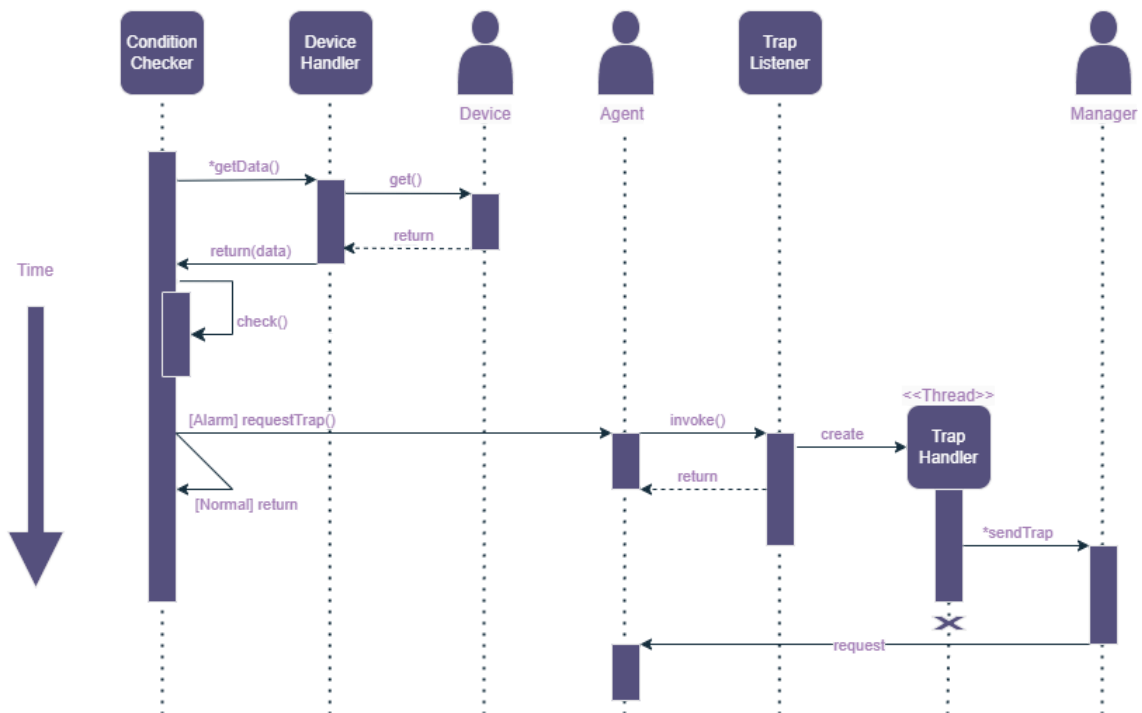
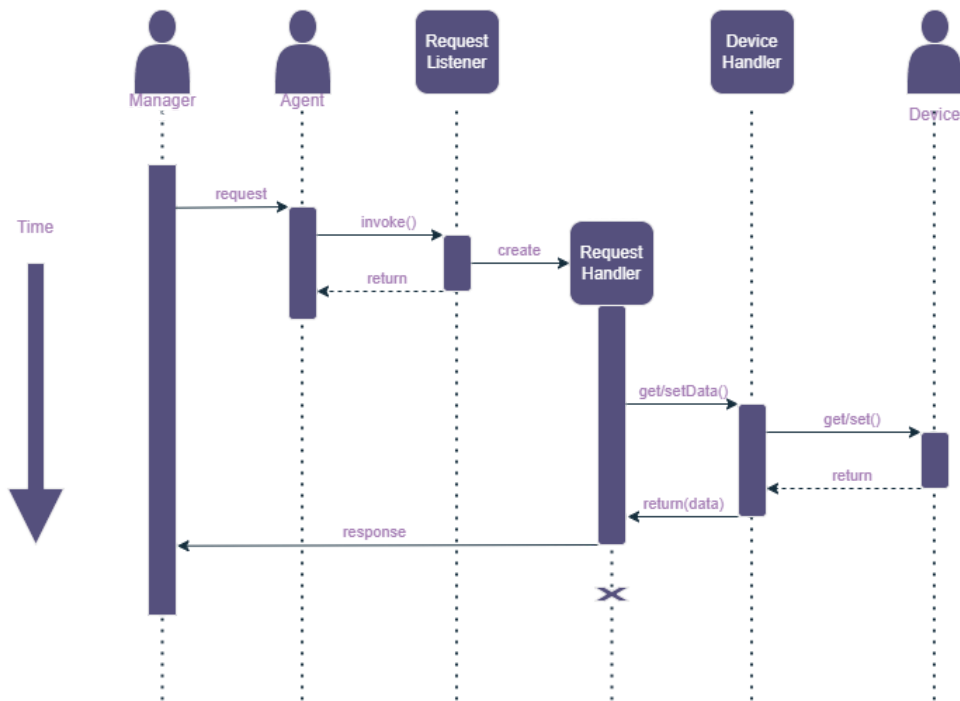
|-- 2 - Nume

|-- 3 - RAM % usage

|-- 4 - RAM GB usage

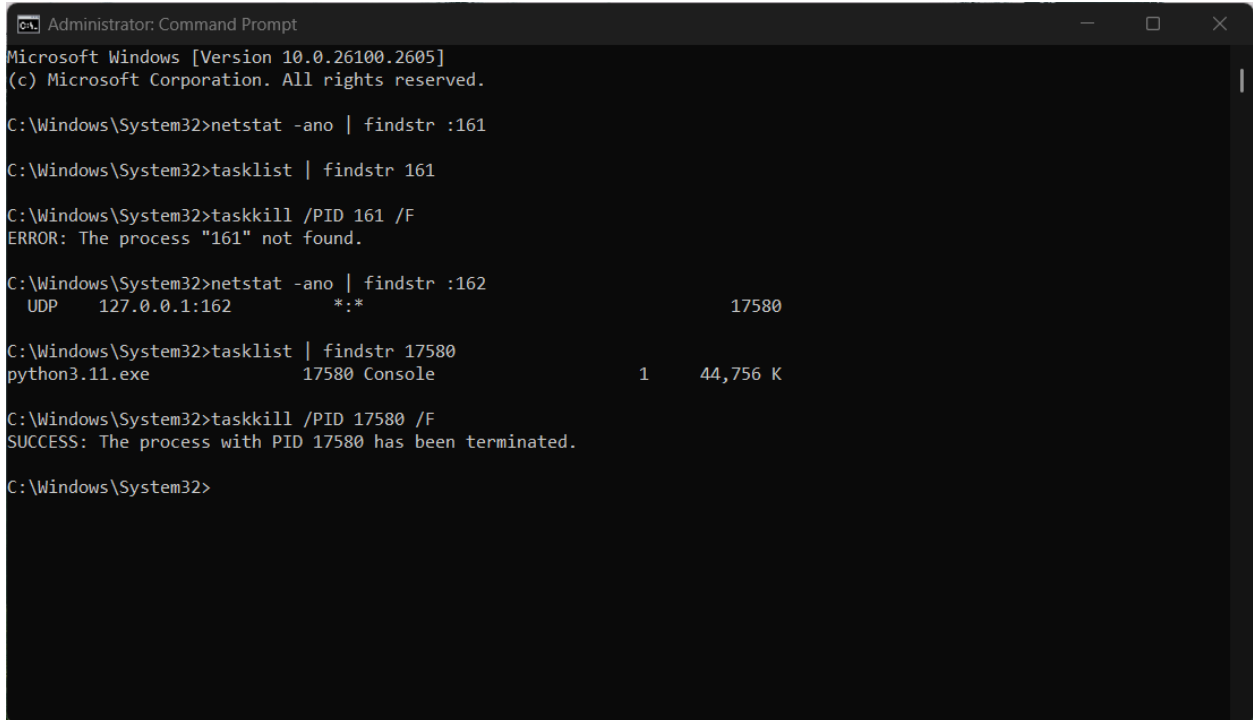
|-- 5 - CPU % usage

## 8. Sequence Diagram



## 9.Probleme intampinate in timpul implementarii

În timpul implementării, am întâmpinat o problemă legată de porturile SNMP 161 și 162, care erau ocupate, ceea ce a împiedicat funcționarea corectă a aplicației. Portul 161 nu avea un proces asociat, însă portul 162 era utilizat de procesul python3.11.exe, identificat cu PID-ul 17580. Pentru a rezolva problema, am utilizat comanda taskkill /PID 17580 /F pentru a termina procesul respectiv și a elibera portul 162. Astfel, porturile necesare au fost deblocate, permițând continuarea implementării fără alte conflicte.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.26100.2605]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>netstat -ano | findstr :161

C:\Windows\System32>tasklist | findstr 161

C:\Windows\System32>taskkill /PID 161 /F
ERROR: The process "161" not found.

C:\Windows\System32>netstat -ano | findstr :162
  UDP      127.0.0.1:162          *:*                17580

C:\Windows\System32>tasklist | findstr 17580
python3.11.exe           17580 Console          1      44,756 K

C:\Windows\System32>taskkill /PID 17580 /F
SUCCESS: The process with PID 17580 has been terminated.

C:\Windows\System32>
```

A doua problema intampinata a fost extragerea temperaturii pe Windows. Am incercat diverse metode de pe diverse site-uri, forumuri, videoclipuri de pe youtube, insa niciunul nu a functionat.

<https://stackoverflow.com/questions/3262603/accessing-cpu-temperature-in-python>

<https://raspberrypi.stackexchange.com/questions/85415/how-to-directly-get-cpu-temp-in-python>

<https://www.youtube.com/watch?v=RVvmB7HFTR4>

<https://python-forum.io/thread-33291.html>

<https://python-forum.io/thread-28680.html>

Intr-un final, am gasit un forum in care un profesor a recomandat ca aceasta operatie de extragere a temperaturii sa se realizeze pe sistemul de operare Linux, ceea ce ne-a determinat sa schimbam sistemul de operare, si astfel problema a fost rezolvata.

Domnul profesor a sugerat folosirea urmatorului script:

```
import sys

import psutil

def main():

    if not hasattr(psutil, "sensors_temperatures"):

        sys.exit("platform not supported")

    temps = psutil.sensors_temperatures()

    if not temps:

        sys.exit("can't read any temperature")

    for name, entries in temps.items():

        print(name)

        for entry in entries:

            print("    %-20s %s °C (high = %s °C, critical = %s °C)" % (

                entry.label or name, entry.current, entry.high,

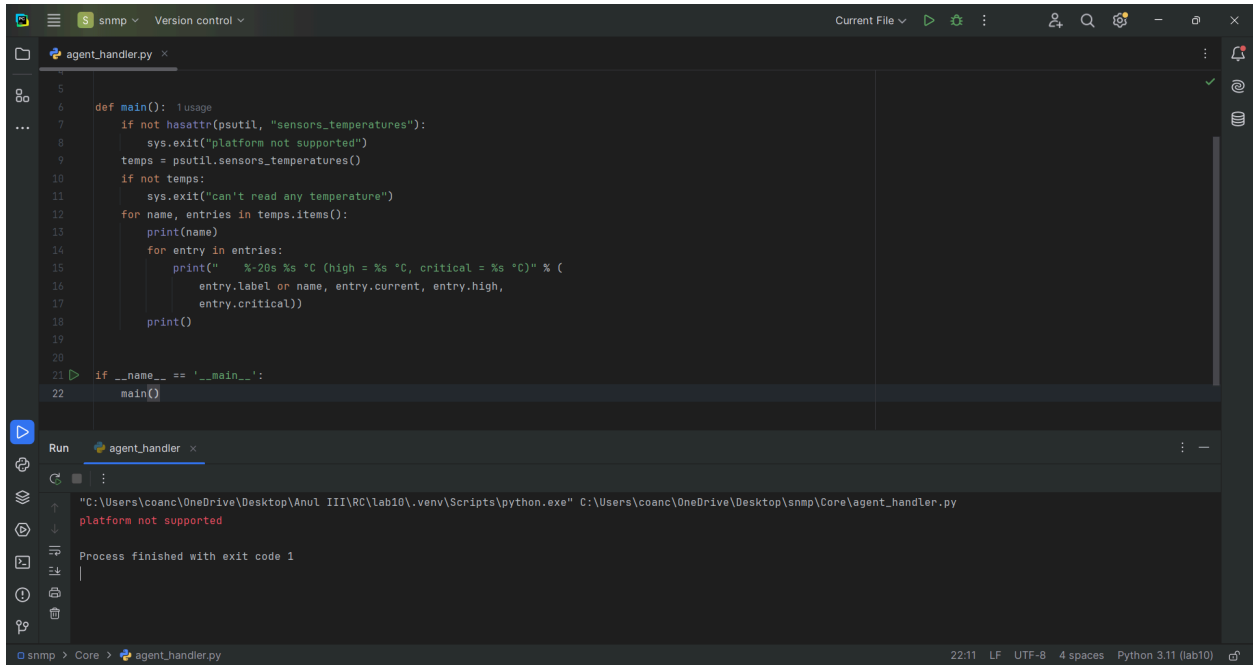
                entry.critical))

        print()

if __name__ == '__main__':

    main()
```

La rularea acestuia pe Windows am obtinut urmatorul rezultat:



The screenshot shows a Windows IDE with a file named `agent_handler.py`. The code is a Python script that attempts to use `psutil` to get system temperatures. The `main` function checks if `psutil` is available and if it can read temperatures. If not, it exits with code 1. The output window shows the error: `platform not supported`.

```
def main(): usage
    if not hasattr(psutil, "sensors_temperatures"):
        sys.exit("platform not supported")
    temps = psutil.sensors_temperatures()
    if not temps:
        sys.exit("can't read any temperature")
    for name, entries in temps.items():
        print(name)
        for entry in entries:
            print("    %-20s %s °C (high = %s °C, critical = %s °C)" % (
                entry.label or name, entry.current, entry.high,
                entry.critical))
        print()

if __name__ == '__main__':
    main()
```

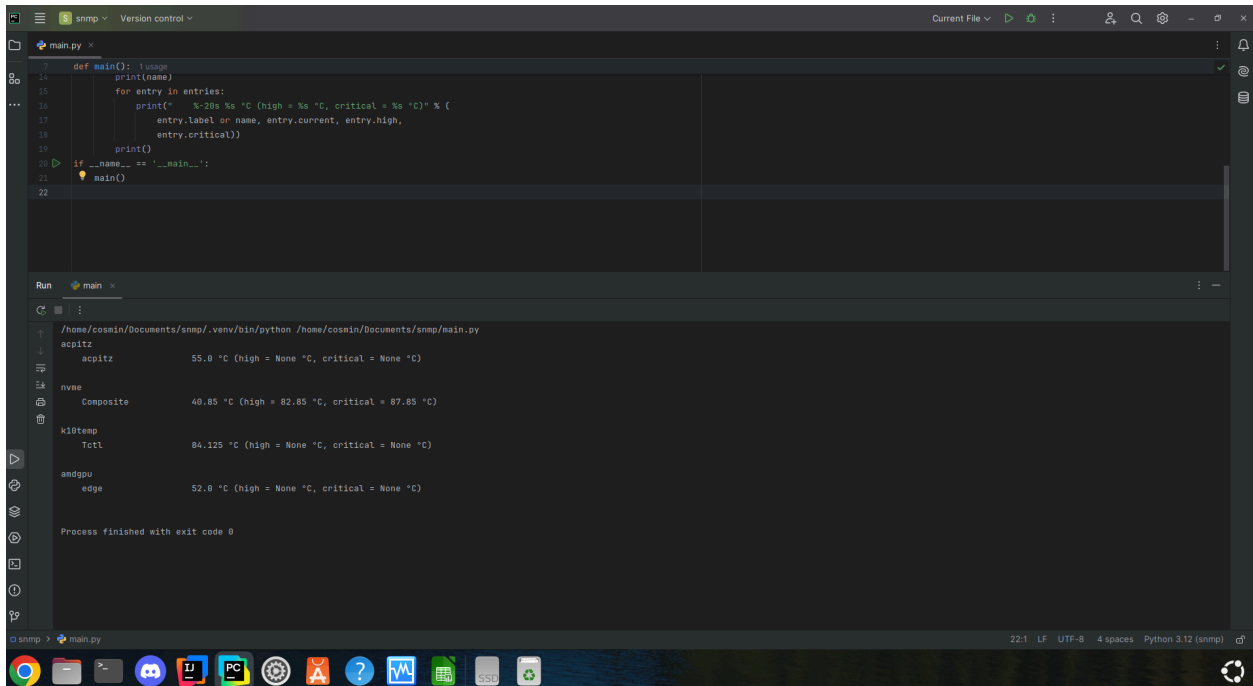
Run agent\_handler

"C:\Users\coanc\OneDrive\Desktop\Anul III\RC\lab10\.venv\Scripts\python.exe" C:\Users\coanc\OneDrive\Desktop\snmp\Core\agent\_handler.py

platform not supported

Process finished with exit code 1

Dupa ce ne-am mutat pe Linux, am reusit intr-un final sa obtinem temperatura procesorului.



The screenshot shows a Linux IDE with a file named `main.py`. The code is a Python script that uses `psutil` to get system temperatures. The `main` function prints the name of each temperature sensor and its current, high, and critical values. The output window shows the results of the script run.

```
def main(): usage
    print(name)
    for entry in entries:
        print("    %-20s %s °C (high = %s °C, critical = %s °C)" % (
            entry.label or name, entry.current, entry.high,
            entry.critical))
    print()

if __name__ == '__main__':
    main()
```

Run main

/home/cosmin/Documents/snmp/.venv/bin/python /home/cosmin/Documents/snmp/main.py

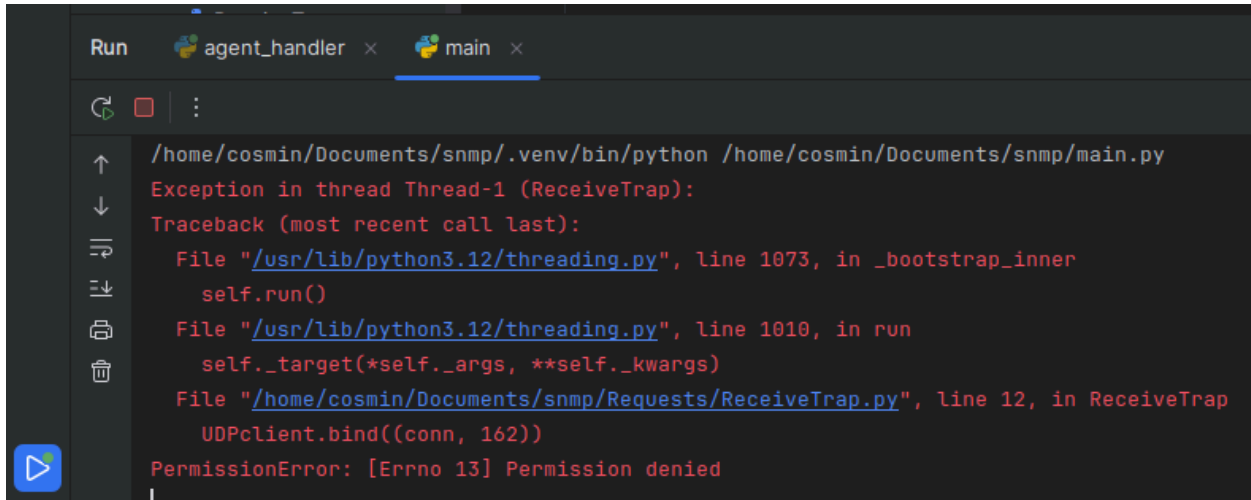
acpitz	55.0 °C (high = None °C, critical = None °C)
nvme	Composite 40.85 °C (high = 82.85 °C, critical = 87.85 °C)
kl8temp	Tctl 84.125 °C (high = None °C, critical = None °C)
andgpu	edge 52.0 °C (high = None °C, critical = None °C)

Process finished with exit code 0

După migrarea pe Linux, am întâmpinat o nouă problemă legată de utilizarea porturilor SNMP 161 și 162, acestea fiind rezervate și necesitând permisiuni speciale pentru acces. Pe Linux, aceste porturi, fiind considerate "privilegiate" (porturi sub 1024), nu pot fi utilizate de aplicații



fără rularea acestora cu privilegii de administrator, ceea ce ar fi complicat implementarea și testarea soluției. Pentru a evita aceste limitări, am decis să folosim porturi de dimensiuni mai mari, 8080 și 8081, care nu necesită permisiuni speciale și sunt frecvent utilizate pentru aplicații web și servicii personalizate. Această schimbare a implicat ajustarea configurației aplicației pentru a redirecționa traficul SNMP către noile porturi, ceea ce a permis continuarea dezvoltării fără alte constrângeri legate de accesul la porturi. Soluția a fost implementată rapid și eficient, asigurând compatibilitatea și funcționalitatea pe Linux.



```
Run agent_handler x main x
Exception in thread Thread-1 (ReceiveTrap):
Traceback (most recent call last):
  File "/usr/lib/python3.12/threading.py", line 1073, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.12/threading.py", line 1010, in run
    self._target(*self._args, **self._kwargs)
  File "/home/cosmin/Documents/snmp/Requests/ReceiveTrap.py", line 12, in ReceiveTrap
    UDPClient.bind((conn, 162))
PermissionError: [Errno 13] Permission denied
```

## **9.Bibliografie:**

### ***Carti:***

Douglas Mauro, Kevin Schmidt, „Essential SNMP”, O'Reilly, 2008.

Kevin R. Fall, W. Richard Stevens, „The Protocols”, Addison-Wesley Professional Computing Series, 2012.

Sidnie M.Feit, SNMP: A Guide to Network Management

Aaron Leskiw, SNMP Tutorial Part 2: Rounding Out the Basics


### ***Site-uri:***

[Simple Network Management Protocol - Wikipedia](#)


[What is Simple Network Management Protocol \(SNMP\)? Definition from SearchNetworking](#)


[Simple Network Management Protocol \(SNMP\) - GeeksforGeeks](#)

### ***Video:***

 [How SNMP Works - a quick guide](#)

 [SNMP Operation \(CCNA Complete Video Course Sample\)](#)

 [SNMP Explained | Simple Network Management Protocol | Cisco CCNA 200-301](#)

 [SNMP Theory + Practical | Simple Network Management Protocol Details | How S...](#)