

LABORATORUL 01

Banc de test energetic pe Raspberry Pi 5 și Pi 3

Durată: 3 ore | **Suport:** Capitolele 4, 5 și 6 | **Platformă:** Raspberry Pi 5 sau Pi 3 + afișaj LCD
Carte capitolul de referință

CUPRINS

1.	Obiectivele lucrării
2.	Obiectul lucrării
3.	Materiale necesare. Pi 5 sau Pi 3
4.	De unde vine consumul
5.	Explorator DVFS: unde este optimul
6.	Pregătirea sistemului
7.	Citirea stării procesorului
8.	Măsurarea internă a puterii (Pi 5)
9.	Măsurarea externă cu INA219
10.	Afișajul în timp real
11.	Programul complet
12.	Experimentul și interpretarea
13.	Sarcini de lucru
14.	Aplicație de aprofundare
15.	Întrebări de verificare
16.	Resurse

Un procesor nu consumă o putere fixă: consumul depinde de frecvență, de tensiune și de cât timp stă degeaba. În această lucrare transformăm plăcuța Raspberry Pi (5 sau 3) într-un instrument care se măsoară pe sine - afișează pe LCD puterea consumată în timp real și ne lasă să comparăm, cu cifre, ce înseamnă „rapid” și ce înseamnă „economic”.

1 Obiectivele lucrării

- ☐ Înțelegerea relației $P = C \cdot V^2 \cdot f$ și a motivului pentru care tensiunea contează mai mult decât frecvența
- ☐ Citirea stării procesorului din `sysfs`: frecvență curentă, guvernator, limitări termice
- ☐ Măsurarea puterii consumate prin două metode independente: circuitul intern de gestiune a alimentării (numai pe Pi 5) și un modul extern INA219 (singura metodă pe Pi 3)
- ☐ Schimbarea guvernatorului DVFS și observarea efectului asupra timpului de execuție și a energiei
- ☐ Construirea unui afișaj LCD care raportează în timp real puterea, frecvența și temperatura
- ☐ Calculul **energiei pe sarcină** și al raportului **performanță pe watt** - indicatorii care contează cu adevărat

2 Obiectul lucrării

Construim un **banc de test energetic**: un program care rulează o sarcină de calcul identică de mai multe ori, o dată pentru fiecare politică de frecvență a procesorului, măsurând de fiecare dată cât a durat și câtă energie a consumat. Rezultatele apar pe LCD în timp real și se salvează într-un fișier CSV pe care îl puteți deschide în Excel pentru referat.

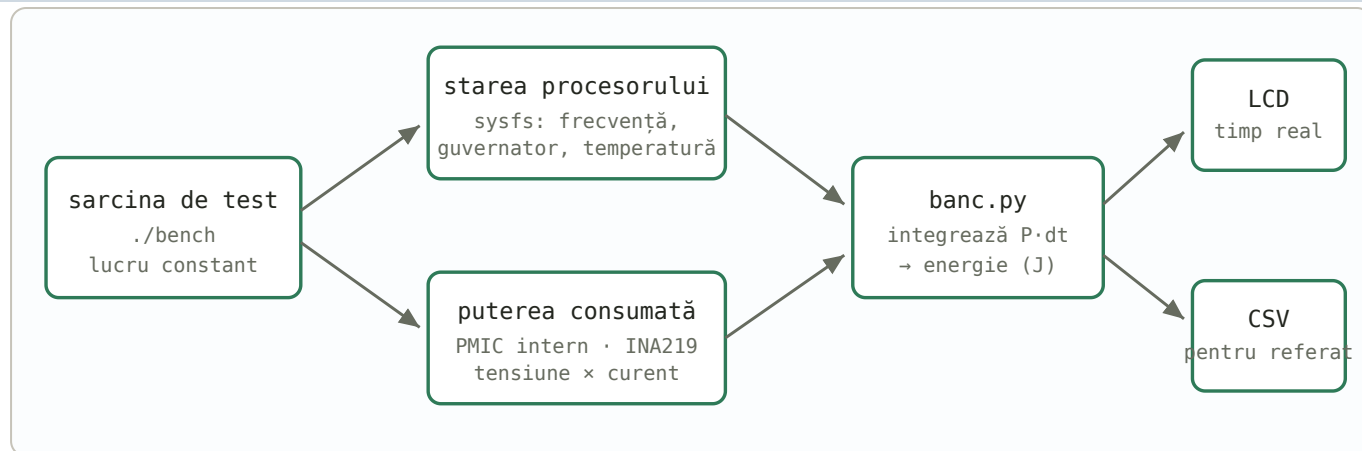


Fig. 1 - Structura bancului de test. Sarcina de calcul este mereu aceeași; se schimbă doar politica de frecvență. Măsurăm simultan starea procesorului și puterea, iar programul le combină în energie consumată.

DE CE RASPBERRY PI 5 ȘI NU UN MICROCONTROLLER

Pentru că Pi 5 are exact ce ne trebuie ca să studiem consumul: un procesor Cortex-A76 cu mai multe trepte de frecvență, un guvernator care le schimbă automat și - spre deosebire de aproape orice altă plăcuță de dezvoltare - un circuit intern de gestiune a alimentării care **își raportează singur curenții pe fiecare linie**. Putem începe să măsurăm înainte să lipim un singur fir.

Lucrarea se poate face și pe **Raspberry Pi 3**, cu o diferență importantă: Pi 3 nu are acest circuit, deci puterea se măsoară numai din exterior, cu modulul INA219. Diferențele dintre cele două plăci sunt adunate în secțiunea următoare, iar acolo unde pașii diferă, pagina o spune explicit.

3 Materiale necesare

Pentru ambele variante

1

Card microSD 32 GB cu Raspberry Pi OS (64 biți, Bookworm)

1

Afișaj LCD 16×2 cu adaptor I²C (PCF8574)

6

Fire mamă-mamă

Varianța Raspberry Pi 5

1

Raspberry Pi 5 (4 sau 8 GB)

1

Alimentator oficial 27 W USB-C

1

Răcitor activ oficial (recomandat - altfel apare limitarea termică)

1

Modul INA219 sau INA226 (opțional, pentru partea a doua)

Varianta Raspberry Pi 3

1

Raspberry Pi 3
Model B sau B+

1

Modul INA219
(obligatoriu - este
singura
măsurătoare de
putere)

1

Sursă de laborator
reglabilă 5 V / 3 A,
cu limitare de
curent

2

Fire cu crocodil sau
tată-mamă, pentru
sursă

1

Radiator pe
procesor
(recomandat)

Pi 5 sau Pi 3: ce se schimbă

Aflați mai întâi ce placă aveți în față. Comanda următoare o spune direct:

```
$ modelul plăcii  
  
cat /proc/device-tree/model; echo
```

Răspunsul arată de forma Raspberry Pi 5 Model B Rev 1.0 sau Raspberry Pi 3 Model B Plus Rev 1.3. Notați-l în referat - de el depind secțiunile 8, 9 și 12.

	Raspberry Pi 5	Raspberry Pi 3 (B / B+)
Procesor	BCM2712, 4 × Cortex-A76	BCM2837, 4 × Cortex-A53
Trepte de frecvență	mai multe, aproximativ 1500–2400 MHz	doar două: 600 și 1200 MHz (1400 MHz pe B+)
Măsurare internă a puterii	da, <code>vcgencmd pmic_read_adc</code>	nu există - comanda răspunde <code>Command not registered</code>
Cum măsurăm puterea	PMIC, cu INA219 ca martor	numai INA219, intercalat pe alimentarea plăcii
Alimentare	USB-C, 5 V / 5 A	micro-USB, 5 V / 2,5 A (în lucrare: sursă de laborator)
Secțiuni care diferă	-	8 (se citește, nu se rulează), 9 (montaj obligatoriu), 11 și 12

FĂRĂ LCD? FĂRĂ INA219?

Lucrarea este construită în trepte. Pe **Pi 5**, secțiunile 6–8 funcționează pe placa goală, fără niciun fir conectat; LCD-ul se adaugă la secțiunea 10, iar INA219 la secțiunea 9, și experimentul complet se poate face doar cu măsurătoarea internă. Pe **Pi 3**, secțiunile 6, 7 și 10 funcționează și fără INA219, dar fără el nu aveți nicio măsurătoare de putere: montajul de la secțiunea 9 este obligatoriu pentru experimentul complet.

4 De unde vine consumul

Cele două componente ale puterii

Puterea consumată de un circuit digital are două surse, iar ele se comportă complet diferit:

Componentă	Formulă	Când apare	Cum o reducem
Dinamică - comutarea tranzistoarelor	$P_d = \alpha \cdot C \cdot V^2 \cdot f$	doar când circuitul lucrează	scădem V și f , sau oprim ceasul (<i>clock gating</i>)
Statică - curenți de scurgere	$P_s \approx V \cdot I_{leak}$	permanent , cât timp circuitul e alimentat	tăiem alimentarea (<i>power gating</i>)

Observația cheie este exponentul: puterea dinamică depinde de tensiune **la pătrat**, dar de frecvență doar liniar. Dacă reducem frecvența la jumătate și putem astfel reduce și tensiunea cu 20 %, puterea scade la $0,5 \cdot 0,8^2 = 32 \%$ din valoarea inițială. De aceea cele două se reglează întotdeauna împreună - de aici și numele tehnicii: **DVFS**, *Dynamic Voltage and Frequency Scaling*.

Capcana: energia nu este puterea

Aici se împiedică majoritatea studenților. O putere mai mică nu înseamnă automat o energie mai mică, pentru că sarcina durează mai mult:

RELAȚIA FUNDAMENTALĂ

$$E = P \cdot t \quad [J = W \cdot s]$$

Bateria se descarcă proporțional cu **energia**, nu cu puterea. Un procesor care consumă 2 W timp de 10 secunde consumă exact la fel de mult ca unul care consumă 10 W timp de 2 secunde.

Dacă înjumătățim frecvența, sarcina durează de două ori mai mult. Energia dinamică per sarcină rămâne aproape neschimbată (P scade liniar cu f , t crește liniar cu $1/f$ - se anulează). Ce se schimbă cu adevărat este **energia statică**, care crește pentru că circuitul stă pornit de două ori mai mult, și **energia dinamică** prin scăderea tensiunii.

Cele două strategii opuse

Strategie	Cum sună	Când câștigă
Race to idle (aleargă și adormi)	rulează la frecvență maximă, termină cât mai repede, apoi intră într-o stare de consum redus	când puterea de repaus și scurgerile sunt mari - adică pe procesoare de aplicație moderne
Just in time (exact la timp)	alege cea mai mică frecvență care încă respectă termenul	când repausul costă aproape nimic și puterea dinamică domină - microcontrolere simple

Nu există un răspuns universal. Care câștigă depinde de raportul dintre puterea de repaus și cea activă - și asta putem descoperi singuri, cu simulatorul de mai jos.

5 Explorator DVFS: unde este optimul

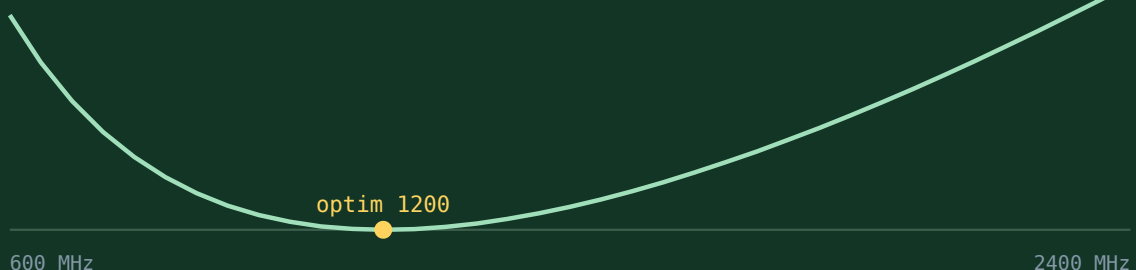
Aveți o sarcină de un anumit număr de cicli și un termen până la care trebuie terminată. Alegeți frecvența. Graficul arată energia totală consumată până la termen, pentru fiecare frecvență posibilă - punctul galben este optimul.

► FRECVENȚĂ, TENSIUNE, ENERGIE ȘI TERMEN LIMITĂ

FRECVENȚĂ (MHZ) LUCRU DE FĂCUT (MCICLI) TERMEN LIMITĂ (MS) PUTERE DE SISTEM (W)

500 1000 0.5

energie totală până la termen (J)



tensiune la 2400 MHz: **1.000 V**

putere: **7.00 W** = 6.00 dinamic + 0.50 scurgeri + 0.50 sistem

durata sarcinii: **208.3 ms** apoi 791.7 ms de somn

energie totală: **1.458 J** vs. 1.250 J la optimul de 1200 MHz (+16.7 %)

Optimul este la mijloc: sub această frecvență platforma pornită prea mult timp depășește economia, peste ea tensiunea crescută costă prea mult.

Model simplificat: $P_{\text{dinamic}} = k \cdot V^2 \cdot f$, tensiunea crescând liniar cu frecvența. **Puterea de sistem** este tot ce rămâne pornit *cât timp sarcina rulează* - memorie, periferice, regulatoare. Presupunem că, odată sarcina terminată, întregul sistem intră în somn adânc și nu mai consumă nimic până la termen.

Versiune interactivă pe site.

EXPERIMENTAȚI CU PUTEREA DE SISTEM

Lăsați ceilalți parametri neschimbați și mișcați doar câmpul „putere de sistem” - adică tot ce rămâne pornit în jurul procesorului cât timp acesta lucrează: memoria, perifericele, regulatoarele.

Putere de sistem	Frecvența optimă	Ce înseamnă
0 W	≈ 800 MHz	procesorul e singur pe lume; contează doar tensiunea, deci lucrați lent
0,5 W	≈ 1200 MHz	echilibru între tensiunea mai mică și timpul mai lung
3 W	≈ 2200 MHz	fiecare secundă costă scump - terminați și adormiți tot sistemul

Optimul se deplasează **continuu**, nu sare: nu există un răspuns universal „repede” sau „încet”, ci un punct care depinde de raportul dintre cât consumă procesorul și cât consumă restul plăcii.

AICI SE ASCUNDE ÎNTREAGA IDEE DE „RACE TO IDLE”

Strategia „aleargă și adormi” nu are sens decât dacă, după terminarea sarcinii, sistemul **chiar poate** să adoarmă profund. Dacă memoria rămâne înprospătată și perifericele pornite, terminarea rapidă nu vă aduce nimic - ați plătit tensiunea mare degeaba. De aceea, pe un procesor de aplicație, decizia despre frecvență nu se poate lua separat de decizia despre stările de somn ale platformei.

ATENȚIE LA MODEL

Simulatorul folosește un domeniu de 600–2400 MHz, tipic pentru un procesor de aplicație. Raspberry Pi 5 are un domeniu mai îngust în practică (aproximativ 1500–2400 MHz), iar Raspberry Pi 3 are doar două trepte (600 și 1200 MHz, respectiv 1400 MHz pe B+), deci efectele reale pe care le veți măsura vor fi mai mici decât cele din simulare. Modelul presupune de asemenea o legătură liniară între frecvență și tensiune, ceea ce este o aproximare. Ideea rămâne însă aceeași, iar **diferența dintre model și măsurătoare merită discutată explicit în referat** - un model care nu se potrivește perfect vă spune ceva despre sistemul real.

6 Pregătirea sistemului

Parcurgeți pașii în ordine. Fiecare pas se poate verifica imediat - nu treceți mai departe până nu vedeți rezultatul așteptat.

1**Actualizați sistemul și instalați uneltele**

Presupunem Raspberry Pi OS pe 64 de biți (Bookworm sau mai nou), cu acces la terminal.

```
$ actualizare
```

```
sudo apt update && sudo apt full-upgrade -y
sudo apt install -y python3-pip python3-venv i2c-tools build-essential stress-ng
```

2**Activați magistrala I²C**

Prin meniul de configurare: Interface Options → I2C → Yes.

```
$ configurare  
  
sudo raspi-config  
sudo reboot
```

După repornire, verificați că magistrala există:

```
$ verificare  
  
ls -l /dev/i2c-1  
i2cdetect -y 1
```

Trebuie să apară o grilă de adrese. Fiind încă nimic conectat, toate căsuțele sunt --. Dacă apare eroarea „No such file or directory”, I2C nu s-a activat - reluați pasul.

3

Creați un mediu virtual Python

Pe Bookworm nu se mai instalează pachete Python direct în sistem. Opțiunea `--system-site-packages` ne lasă totuși să folosim bibliotecile deja instalate prin apt.

```
$ mediu de lucru  
  
python3 -m venv --system-site-packages ~/si-lab  
source ~/si-lab/bin/activate  
pip install smbus2 RPLCD adafruit-circuitpython-ina219
```

DE ȚINUT MINTE

La fiecare sesiune nouă de terminal trebuie să rulați din nou `source ~/si-lab/bin/activate`, altfel Python nu găsește bibliotecile.

4

Creați folderul de lucru

```
$ folder  
  
mkdir -p ~/si-lab/lab01 && cd ~/si-lab/lab01
```

5

Verificați ce guvernatoare are procesorul

```
$ guvernatoare disponibile  
  
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors  
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor  
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

Ar trebui să vedeți o listă care conține cel puțin `ondemand`, `powersave`, `performance` și `schedutil`, iar frecvența curentă exprimată în kHz (de exemplu `1500000` = 1,5 GHz).

Pe **Pi 3** frecvența curentă poate fi doar 600000 sau 1200000 (1400000 pe B+) - procesorul are numai aceste două trepte.

CE ESTE /SYS

Nu este un folder cu fișiere pe card. Este o fereastră către structurile din nucleul sistemului de operare: când citiți `scaling_cur_freq`, nucleul interoghează pe loc controlerul de ceas și vă întoarce răspunsul ca text. Când scrieți în `scaling_governor`, nucleul schimbă efectiv politica. Este aceeași idee ca regiștrii mapate în memorie de la microcontrolere, doar mutată cu un nivel mai sus.

7 Citirea stării procesorului

Primul program citește tot ce ne poate spune sistemul despre starea procesorului. Nu are nevoie de niciun fir conectat.

► stare_cpu.py

```
#!/usr/bin/env python3
"""Citeste starea procesorului (Raspberry Pi 5 sau 3) din sysfs si din vcgencmd."""

import pathlib
import subprocess

NUCLEE = 4
CPUFREQ = "/sys/devices/system/cpu/cpu{}/cpufreq/{"

def citeste(cale):
    """Citeste un fisier din sysfs; intoarce None daca nu exista."""
    try:
        return pathlib.Path(cale).read_text().strip()
    except OSError:
        return None

def frecventa(nucleu=0):
    """Frecventa curenta a unui nucleu, in MHz."""
    khz = citeste(CPUFREQ.format(nucleu, "scaling_cur_freq"))
    return int(khz) / 1000 if khz else None

def guvernator(nucleu=0):
    return citeste(CPUFREQ.format(nucleu, "scaling_governor"))

def trepte_disponibile():
    """Lista treptelor de frecventa, in MHz."""
    val = citeste(CPUFREQ.format(0, "scaling_available_frequencies"))
    if not val:
        # Unele nuclee expun doar limitele, nu si lista completa.
        jos = citeste(CPUFREQ.format(0, "cpuinfo_min_freq"))
        sus = citeste(CPUFREQ.format(0, "cpuinfo_max_freq"))
        return [int(jos) / 1000, int(sus) / 1000] if jos and sus else []
    return [int(v) / 1000 for v in val.split()]
```

```
def temperatura():
    """Temperatura procesorului, in grade Celsius."""
    val = citeste("/sys/class/thermal/thermal_zone0/temp")
    return int(val) / 1000 if val else None

# Fiecare bit din vcgencmd get_throttled semnaleaza o problema de alimentare
# sau de temperatura. Bitii 0-3 = acum, bitii 16-19 = s-a intamplat de la pornire.
STEAGURI = {
    0: "subtensiune ACUM",
    1: "frecventa limitata ACUM",
    2: "limitare termica ACUM",
    3: "limita soft de temperatura ACUM",
    16: "a existat subtensiune",
    17: "a existat limitare de frecventa",
    18: "a existat limitare termica",
    19: "s-a atins limita soft de temperatura",
}

def limitari():
    """Intoarce lista problemelor semnalate de firmware."""
    try:
        iesire = subprocess.run(["vcgencmd", "get_throttled"],
                                capture_output=True, text=True, check=True).stdout
    except (OSError, subprocess.CalledProcessError):
        return ["vcgencmd indisponibil"]
    valoare = int(iesire.strip().split("=")[1], 16)
    return [text for bit, text in STEAGURI.items() if valoare & (1 << bit)]

if __name__ == "__main__":
    print("guvernator activ :", guvernator())
    print("trepte (MHz)      :", trepte_disponibile())
    print("temperatura       : %.1f gradeC" % temperatura())
    print()
    for n in range(NUCLEE):
        print(" nucleul %d: %7.1f MHz" % (n, frecventa(n)))
    print()
    probleme = limitari()
    print("limitari          :", ", ", ".join(probleme) if probleme else "niciuna")
```

Rulați-l de două ori: o dată pe un sistem în repaus, apoi din nou în timp ce încărcăți procesorul. Diferența ar trebui să fie vizibilă imediat.

```
$ test
```

```
python3 stare_cpu.py                # sistem în repaus

stress-ng --cpu 4 --timeout 30s &   # încarcă toate cele 4 nuclee
sleep 5 && python3 stare_cpu.py      # și acum?
```

DACĂ APARE „SUBTENSIUNE” SAU „LIMITARE TERMICĂ”

Rezultatele măsurătorilor devin nefolositoare: procesorul nu mai rulează la frecvența pe care i-o cereți, ci la cea pe care i-o permite alimentarea sau temperatura. Folosiți alimentatorul oficial de 27 W și un răcitor activ. Verificați acest lucru **înainte** de fiecare experiment - este cea mai frecventă cauză de rezultate care „nu au sens”.

PE RASPBERRY PI 3: FRECVENȚA ȘI TENSIUNEA, VĂZUTE DE FIRMWARE

Firmware-ul plăcii raportează direct frecvența nucleelor și tensiunea lor de alimentare:

```
$ frecvență și tensiune

vcgencmd measure_clock arm
vcgencmd measure_volts core
```

Rulați comenzile o dată în repaus și o dată sub stress-ng. Pe unele plăci (mai ales pe B+) tensiunea urcă odată cu frecvența, pe altele rămâne aproape constantă. Ambele rezultate sunt informative: fără scădere de tensiune, DVFS-ul câștigă mult mai puțin decât promite formula din secțiunea 4. Tot pe Pi 3, „subtensiunea” apare des cu încărcătoare de telefon: placa are nevoie de 5 V / 2,5 A.

8 Măsurarea internă a puterii (numai Pi 5)

AVEȚI UN RASPBERRY PI 3? CITIȚI SECȚIUNEA, DAR MĂSURAȚI LA SECȚIUNEA 9

Pi 3 nu are un circuit de alimentare cu convertoare de măsură. Pe el, comanda de mai jos răspunde cu `vc_gencmd_read_response returned -1 și error=1 error_msg="Command not registered"`: firmware-ul nu cunoaște comanda. Nu este o greșeală de instalare și nu se rezolvă prin actualizare. Citiți totuși secțiunea, ca să înțelegeți ce vă lipsește, apoi continuați cu secțiunea 9, unde INA219 devine măsurătoarea principală.

Pe un **Pi 5**, aceeași eroare înseamnă firmware prea vechi: rulați `sudo apt full-upgrade` și `sudo rpi-eeprom-update -a`, apoi reporniți placa.

Raspberry Pi 5 are un circuit dedicat de gestiune a alimentării (PMIC) care distribuie tensiunile către procesor, memorie și periferice. Acest circuit are convertoare analog-numerice pe fiecare linie și **își poate raporta singur curenții și tensiunile**. Comanda este:

```
$ citire brută

vcgencmd pmic_read_adc
```

leșirea arată așa (extras):

ieșire tipică

```

3V7_WL_SW_A current(0)=0.00224000A
3V3_SYS_A current(1)=0.31752000A
1V8_SYS_A current(2)=0.29892000A
VDD_CORE_A current(7)=2.61856000A
...
3V7_WL_SW_V volt(16)=3.69946289V
3V3_SYS_V volt(17)=3.30712891V
VDD_CORE_V volt(23)=0.82031250V
EXT5V_V volt(24)=5.09843750V

```

Fiecare linie de alimentare apare de două ori: o dată cu sufixul `_A` (curentul) și o dată cu `_V` (tensiunea). Puterea unei linii este produsul lor, iar puterea totală este suma peste toate liniile. Programul următor face exact asta:

► putere_pmic.py

```

#!/usr/bin/env python3
"""Citeste puterea consumata direct din circuitul de alimentare al placii Pi 5."""

import re
import subprocess

# Exemplu de linie: VDD_CORE_A current(7)=2.61856000A
LINIE = re.compile(r"^\s*(?P<linie>\S+?)_(?P<tip>[AV])\s+([w\(\d+\))=(?P<val>[0-9.]+)[AV]\s*$")

def masoara():
    """Intoarce (putere_totala_W, dictionar cu puterea pe fiecare linie)."""
    try:
        iesire = subprocess.run(["vcgencmd", "pmic_read_adc"],
                                capture_output=True, text=True, check=True).stdout
    except (OSError, subprocess.CalledProcessError):
        return None, {}

    brut = {}
    for rand in iesire.splitlines():
        m = LINIE.match(rand)
        if m:
            brut.setdefault(m.group("linie"), {})[m.group("tip")] = float(m.group("val"))

    # Numai liniile care raporteaza si curent, si tensiune pot da o putere.
    # EXT5V si BATT raporteaza doar tensiunea, deci raman pe dinafara.
    detaliu = {nume: v["A"] * v["V"] for nume, v in brut.items() if "A" in v and "V" in v}
    if not detaliu:
        return None, {} # placa nu are PMIC cu masurare (de exemplu Pi 3)
    return sum(detaliu.values()), detaliu

if __name__ == "__main__":
    total, detaliu = masoara()
    if total is None:
        raise SystemExit("PMIC indisponibil - exista doar pe Raspberry Pi 5 (pe Pi 3 folositi putere_ina219.py)")

    for nume, p in sorted(detaliu.items(), key=lambda x: -x[1]):
        procent = 100 * p / total
        bara = "#" * int(procent / 2)

```

```
print("%-12s %6.3f W %5.1f%% %s" % (nume, p, procent, bara))
print("-" * 46)
print("%-12s %6.3f W" % ("TOTAL", total))
```

CE VEȚI VEDEA

Linia VDD_CORE - alimentarea nucleelor procesorului - domină în sarcină și devine aproape neglijabilă în repaus. Rulați programul o dată în repaus și o dată sub stress-ng și comparați repartiția. Aceasta este o măsurătoare pe care nu o puteți face pe majoritatea plăcilor de dezvoltare.

CE NU INCLUDE ACEASTĂ MĂSURĂTOARE

Puterea raportată este cea consumată **după** regulatoarele plăcii. Puterea trasă efectiv de la priză este mai mare cu pierderile din regulatoare (tipic 10–20 %) și cu tot ce alimentează portul USB. De aceea, la secțiunea următoare, o comparăm cu o măsurătoare externă.

9 Măsurarea externă cu INA219

Modulul INA219 măsoară căderea de tensiune pe un rezistor de $0,1 \Omega$ plasat în serie cu sarcina și o convertește în curent. Pe Pi 5 îl folosim ca „martor” pentru măsurătoarea internă, iar pe Pi 3 este **singura** măsurătoare de putere. Comunică pe I²C, deci împarte magistrala cu LCD-ul.

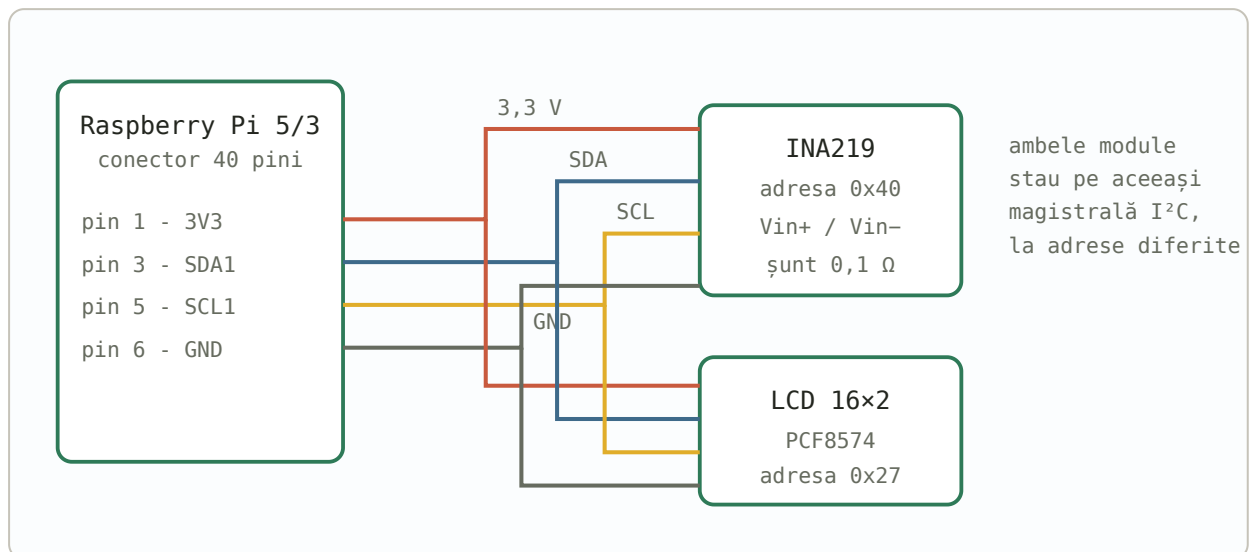


Fig. 2 - Cablarea. I²C este o magistrală cu două fire: toate modulele se leagă în paralel pe aceleași linii SDA și SCL, iar adresa le deosebește. De aceea LCD-ul și INA219 pot funcționa simultan cu doar patru fire. Conectorul de 40 de pini este identic pe Pi 5 și Pi 3, deci aceste legături sunt aceleași pe ambele plăci.

PE PI 5: MĂSURAREA CONSUMULUI PLĂCII ÎN SINE

Pentru a măsura consumul *întregii* plăci, INA219 trebuie intercalat pe firul de +5 V dinspre alimentator, înainte de Pi. Asta înseamnă tăierea unui cablu USB-C - o operație care trebuie făcută cu atenție și pe un cablu de sacrificiu. **Alternativa sigură și recomandată la laborator:** lăsați INA219 să măsoare o sarcină separată (de exemplu ventilatorul răcitorului sau un LED de putere alimentat din pinul 5 V) și folosiți-l pentru a înțelege principiul, iar consumul procesorului citiți-l de la PMIC.

Pe Raspberry Pi 3: INA219 pe alimentarea plăcii

Pe Pi 3 nu avem PMIC, deci INA219 trebuie să vadă **tot** curentul plăcii. Cea mai simplă cale, fără să tăiem cabluri, este să alimentăm placa de la o sursă de laborator prin pinul de 5 V al conectorului, cu șuntul modulului intercalat pe firul de +5 V. Legăturile I²C (3V3, SDA, SCL, GND) rămân cele din fig. 2.

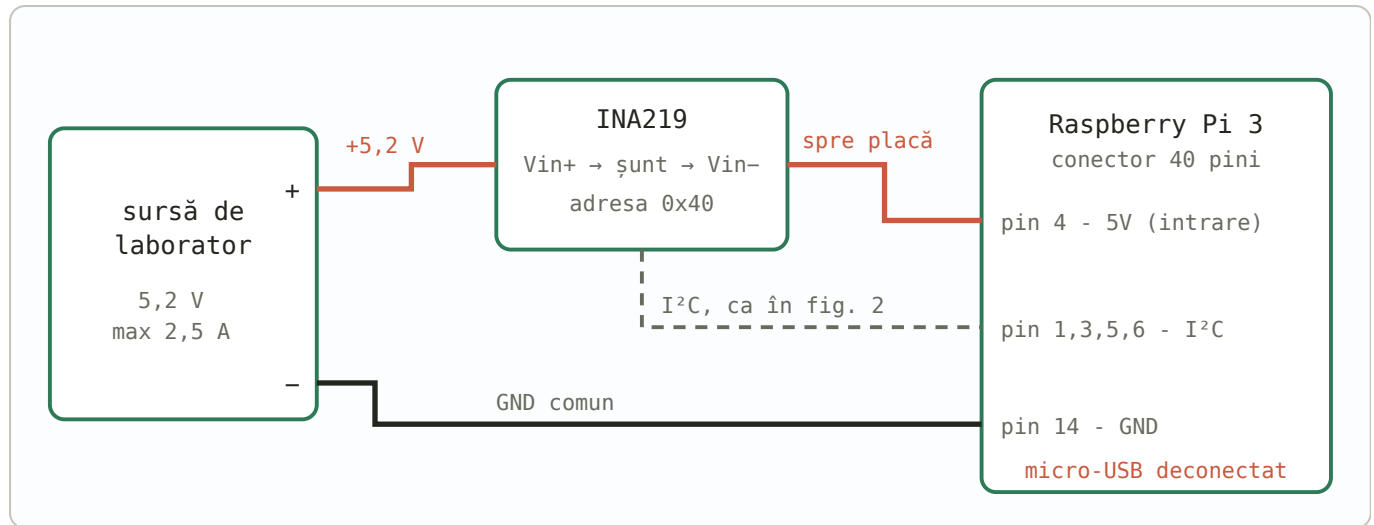


Fig. 3 - Montajul pentru Pi 3. Tot curentul plăcii trece prin șuntul INA219 (roșu), deci modulul măsoară consumul întregii plăci. Firul negru închide circuitul direct, iar firele I²C sunt aceleași ca în fig. 2.

REGULI PENTRU ALIMENTAREA PRIN PINUL DE 5 V

1. **Deconectați cablul micro-USB.** Placa nu se alimentează niciodată din două surse în același timp.
2. Reglați sursa la 5,15–5,2 V și limitați curentul la 2,5 A **înainte** de a o conecta la placă. Nu depășiți 5,25 V.
3. Pinul de 5 V ocolește siguranța de protecție a plăcii: verificați polaritatea de două ori înainte de pornire.
4. Șuntul și firele coboară tensiunea cu aproximativ 0,1 V la 1 A, de aceea sursa se reglează puțin peste 5 V. Dacă `stare_cpu.py` raportează „subtensiune”, creșteți ușor tensiunea sursei, fără a depăși limita de mai sus.

Fără sursă de laborator se poate folosi un cablu micro-USB de sacrificiu, cu firul roșu (+5 V) tăiat și trecut prin Vin+ → Vin–; celelalte fire rămân neîntrerupte. Varianta cu sursă este însă mai sigură la laborator.

Pe oricare placă, verificați apoi că modulele sunt văzute pe magistrală:

```
$ detectare
```

```
i2cdetect -y 1
```

Trebuie să apară 40 (INA219) și 27 sau 3f (LCD). Dacă nu apare nimic, verificați GND-ul - este firul uitat cel mai des.

```
> putere_ina219.py
```

```
#!/usr/bin/env python3
"""Citeste tensiunea, curentul si puterea de la un modul INA219."""

import time

import board
```

```

from adafruit_ina219 import ADCResolution, BusVoltageRange, INA219

i2c = board.I2C()          # foloseste automat magistrala I2C-1
senzor = INA219(i2c)       # adresa implicita 0x40

# Configurare pentru masuratori linistite: rezolutie maxima si mediere pe
# 128 de esantioane. Reduce zgomotul, cu pretul unei citiri mai lente.
senzor.bus_adc_resolution = ADCResolution.ADCRES_12BIT_128S
senzor.shunt_adc_resolution = ADCResolution.ADCRES_12BIT_128S
senzor.bus_voltage_range = BusVoltageRange.RANGE_16V

def masoara():
    """Intoarce (tensiune_V, curent_mA, putere_W)."""
    # Tensiunea reala pe sarcina = tensiunea pe magistrala + caderea pe sunt.
    tensiune = senzor.bus_voltage + senzor.shunt_voltage
    curent = senzor.current          # mA
    return tensiune, curent, tensiune * curent / 1000.0

if __name__ == "__main__":
    print("%8s %10s %10s" % ("V", "mA", "W"))
    try:
        while True:
            u, i, p = masoara()
            print("%8.3f %10.2f %10.4f" % (u, i, p))
            time.sleep(0.5)
    except KeyboardInterrupt:
        print("\nnoprit")

```

DE CE ADUNĂM CĂDEREA PE ȘUNT

INA219 măsoară tensiunea la ieșirea din rezistorul de măsură. Sarcina „vede” însă tensiunea de dinaintea lui. La un curent de 2 A pe un șunt de 0,1 Ω, diferența este de 0,2 V - deloc neglijabilă. Este o eroare clasică în montajele găsite pe internet.

PE RASPBERRY PI 3: CE VEȚI VEDEA

Cu montajul din fig. 3, `putere_ina219.py` afișează puterea **întregii plăci**. Rulați-l o dată în repaus și o dată sub `stress-ng --cpu 4`. Ca ordin de mărime, așteptați-vă la câțiva wați, cu o parte importantă consumată deja în repaus (memorie, USB, rețea). Nu aveți repartiția pe linii de la Pi 5, dar aveți ceva ce Pi 5 nu vă dă din PMIC: consumul real de la sursă.

Un singur modul de măsură pentru ambele plăci

Ca programele următoare să ruleze neschimbate pe Pi 5 și pe Pi 3, ascundem sursa de măsură în spatele unui modul mic, care o alege singur: PMIC-ul dacă există, altfel INA219. Ambele variante întorc aceeași formă de rezultat, (`putere_totala_W`, `detaaliu`).

► `putere.py`

```

#!/usr/bin/env python3
"""Alege automat de unde citim puterea:
Pi 5 -> circuitul PMIC intern (putere_pmic.py),
Pi 3 -> modulul INA219 de pe alimentarea placii (putere_ina219.py)."""

```

```
import putere_pmic

SURSA = None

def masoara():
    """Nicio sursa disponibila: intoarce acelasi format, dar gol."""
    return None, {}

if putere_pmic.masoara()[0] is not None:
    SURSA = "PMIC"
    masoara = putere_pmic.masoara
else:
    try:
        import putere_ina219          # initializeaza senzorul; esueaza daca lipseste
        putere_ina219.masoara()       # o citire de proba
    except Exception as e:
        print("Fara PMIC, iar INA219 nu raspunde (%s)" % e)
    else:
        SURSA = "INA219"

        def masoara():
            """Aceasi forma ca putere_pmic.masoara(): (total_W, detaliu)."""
            _, _, p = putere_ina219.masoara()
            return p, {"PLACA_5V": p}

if __name__ == "__main__":
    total, _ = masoara()
    if total is None:
        raise SystemExit("nu am cu ce masura puterea - vezi sectiunile 8 si 9")
    print("sursa: %s   putere: %.3f W" % (SURSA, total))
```

```
$ test
```

```
python3 putere.py      # Pi 5: sursa PMIC   Pi 3: sursa INA219
```

10 Afișajul în timp real

LCD-ul are 32 de caractere. Trebuie să alegem ce merită afișat - un exercițiu util în sine, pentru că în sistemele încorporate resursa limitată este mereu regula, nu excepția.

```
► afisaj.py
```

```
#!/usr/bin/env python3
"""Afișaj LCD 16x2 pentru bancul de test energetic."""

from RPLCD.i2c import CharLCD

# Daca i2cdetect arata 3f in loc de 27, schimbati adresa mai jos.
ADRESA = 0x27

class Panou:
    """Invelis peste LCD care nu se plange daca ecranul lipseste."""
```

```

def __init__(self, adresa=ADRESA):
    self.lcd = None
    try:
        self.lcd = CharLCD("PCF8574", adresa, cols=16, rows=2, auto_linebreaks=False)
        self.lcd.clear()
        # Simbol propriu pentru grade, in memoria de caractere a ecranului.
        self.lcd.create_char(0, [0b01100, 0b10010, 0b10010, 0b01100,
                                0b00000, 0b00000, 0b00000, 0b00000])
    except Exception as e:
        print("LCD indisponibil (%s) - continui doar in consola" % e)

def scrie(self, sus, jos):
    """Scrie doua randuri de maximum 16 caractere fiecare."""
    sus, jos = sus[:16].ljust(16), jos[:16].ljust(16)
    if self.lcd is None:
        print("\r[%s|%s]" % (sus, jos), end="", flush=True)
        return
    # Scriem pe pozitii fixe, nu cu clear(): altfel ecranul palpaie.
    self.lcd.cursor_pos = (0, 0)
    self.lcd.write_string(sus)
    self.lcd.cursor_pos = (1, 0)
    self.lcd.write_string(jos)

def inchide(self):
    if self.lcd is not None:
        self.lcd.clear()
        self.lcd.close(clear=True)

if __name__ == "__main__":
    import time
    from stare_cpu import frecventa, temperatura
    from putere import masoara

    panou = Panou()
    try:
        while True:
            p, _ = masoara()
            panou.scrie("%4.0fMHz  %4.1fW" % (frecventa(), p or 0),
                       "temp %4.1f gradeC" % temperatura())
            time.sleep(0.5)
    except KeyboardInterrupt:
        panou.inchide()
        print("\n\noprit")

```

DE CE NU FOLOSIM CLEAR() LA FIECARE CADRU

Ștergerea completă a ecranului durează câteva milisecunde și lasă afișajul gol în acest timp, ceea ce se vede ca o pâlpâire. Suprascrierea pe poziții fixe, cu text completat cu spații până la 16 caractere, dă o imagine stabilă. Aceeași tehnică se folosește și la ecranele grafice, unde se numește *double buffering*.

11 Programul complet

Punem totul cap la cap. Avem nevoie mai întâi de o sarcină de calcul care să facă **exact același lucru** de fiecare dată, indiferent de frecvența procesorului - altfel comparația nu are sens.

bench.c - sarcina de referință

```

/* Sarcina de test: un numar fix de operatii pe numere intregi.
   Rezultatul este afisat la sfarsit, ca sa nu poata fi eliminat de compilator. */
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

static uint64_t munca(uint64_t n) {
    uint64_t x = 88172645463325252ULL; /* generator xorshift64 */
    for (uint64_t i = 0; i < n; i++) {
        x ^= x << 13;
        x ^= x >> 7;
        x ^= x << 17;
    }
    return x;
}

int main(int argc, char **argv) {
    uint64_t n = (argc > 1) ? strtoull(argv[1], NULL, 10) : 400000000ULL;
    printf("%llu\n", (unsigned long long) munca(n));
    return 0;
}

```

\$ compilare

```

gcc -O2 -o bench bench.c
time ./bench 400000000 # cât durează? ajustați numărul pentru ~10 secunde

```

Pe **Pi 3**, același număr de pași durează de câteva ori mai mult. Alegeți valoarea după `time`, tot pentru aproximativ 10 secunde la frecvența maximă, și treceți-o în `CICLI` din `banc.py`.

DE CE -O2 ȘI NU -O3

Cu optimizări agresive, compilatorul poate vectoriza sau chiar elimina bucla, iar „sarcina constantă” nu mai este constantă. Ne întoarcem la vectorizare în mod deliberat la laboratorul 5; aici avem nevoie de o referință stabilă.

Iar acum bancul propriu-zis:

▸ banc.py

```

#!/usr/bin/env python3
"""Banc de test energetic: ruleaza aceeasi sarcina sub fiecare guvernator
si compara timpul, puterea medie si energia consumata."""

import csv
import subprocess
import sys
import time

from afisaj import Panou
from putere import SURSA, masoara
from stare_cpu import frecventa, guvernator, limitari, temperatura

GUVERNATOARE = ["powersave", "ondemand", "schedutil", "performance"]
CICLI = 400_000_000 # ajustati ca sa dea aproximativ 10 secunde (pe Pi 3: mai putin)
PERIOADA = 0.2 # intervalul de esantionare a puterii, in secunde

```

```

NUCLEE = 4

def seteaza_guvernator(ume):
    """Scrie acelasi guvernator pe toate nucleele. Necesita drepturi de root."""
    for n in range(NUCLEE):
        cale = "/sys/devices/system/cpu/cpu%d/cpufreq/scaling_governor" % n
        subprocess.run(["sudo", "tee", cale], input=ume.encode(),
                        stdout=subprocess.DEVNULL, check=True)
    time.sleep(2)          # lasam guvernatorul sa se aseze

def masoara_repaus(secunde=5):
    """Puterea medie cu sistemul in repaus - referinta pentru energia utila."""
    probe = []
    sfarsit = time.time() + secunde
    while time.time() < sfarsit:
        p, _ = masoara()
        if p:
            probe.append(p)
        time.sleep(PERIOADA)
    return sum(probe) / len(probe) if probe else 0.0

def ruleaza(panou, ume_guvernator, p_repaus):
    """Ruleaza sarcina si esantioneaza puterea cat timp aceasta lucreaza."""
    seteaza_guvernator(ume_guvernator)

    proces = subprocess.Popen(["./bench", str(CICLI)], stdout=subprocess.DEVNULL)
    probe, frecvente = [], []
    t0 = time.time()

    while proces.poll() is None:
        p, _ = masoara()
        if p:
            probe.append(p)
        f = frecventa()
        if f:
            frecvente.append(f)
        panou.scrie("%-11s%4.1fW" % (ume_guvernator[:11], p or 0),
                    "%4.0fMHz  %4.1fC" % (f or 0, temperatura()))
        time.sleep(PERIOADA)

    durata = time.time() - t0
    p_medie = sum(probe) / len(probe) if probe else 0.0
    f_medie = sum(frecvente) / len(frecvente) if frecvente else 0.0

    return {
        "guvernator": ume_guvernator,
        "durata_s": round(durata, 3),
        "frecventa_medie_MHz": round(f_medie, 1),
        "putere_medie_W": round(p_medie, 3),
        # Energia totala consumata pe durata sarcinii.
        "energie_J": round(p_medie * durata, 2),
        # Energia peste consumul de repaus - partea "platita" efectiv pentru calcul.
        "energie_utilita_J": round((p_medie - p_repaus) * durata, 2),
        # Cat calcul obtinem pentru fiecare joule cheltuit.
        "M cicli_pe_J": round(CICLI / 1e6 / max(p_medie * durata, 1e-9), 2),
        "temperatura_C": round(temperatura(), 1),
    }

```

```

def main():
    if SURSA is None:
        raise SystemExit("Nu am cu ce masura puterea - vezi sectiunile 8 si 9.")
    print("Sursa de masura a puterii: %s" % SURSA)

    # Pastram doar guvernatoarele pe care nucleul le are efectiv.
    with open("/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors") as f:
        disponibile = f.read().split()
    lista = [g for g in GUVERNATOARE if g in disponibile]

    initial = guvernator()
    panou = Panou()

    probleme = limitari()
    if probleme:
        print("ATENTIE - alimentare sau temperatura:", ", ".join(probleme))
        print("Rezultatele nu vor fi comparabile. Continui oricum in 5 secunde...")
        time.sleep(5)

    print("Masor consumul in repaus...")
    p_repaus = masoara_repaus()
    print("Repaus: %.3f W\n" % p_repaus)

    rezultate = []
    try:
        for g in lista:
            print("Rulez cu guvernatorul %s..." % g)
            r = ruleaza(panou, g, p_repaus)
            rezultate.append(r)
            print("    %5.2f s    %5.2f W    %7.1f J    %6.0f MHz"
                  % (r["durata_s"], r["putere_medie_W"], r["energie_J"],
                     r["frecventa_medie_MHz"]))
    finally:
        seteaza_guvernator(initial)      # lasam sistemul cum l-am gasit
        panou.inchide()

    with open("rezultate.csv", "w", newline="") as f:
        scriitor = csv.DictWriter(f, fieldnames=list(rezultate[0].keys()))
        scriitor.writeheader()
        scriitor.writerows(rezultate)

    print("\n%-12s %8s %8s %9s %10s" % ("guvernator", "durata", "putere", "energie",
    "Mcicli/J"))
    for r in rezultate:
        print("%-12s %7.2fs %7.2fW %8.1fJ %9.2f"
              % (r["guvernator"], r["durata_s"], r["putere_medie_W"],
                 r["energie_J"], r["Mcicli_pe_J"]))

    cel_mai_rapid = min(rezultate, key=lambda r: r["durata_s"])
    cel_mai_econom = min(rezultate, key=lambda r: r["energie_J"])
    print("\ncel mai rapid : %s" % cel_mai_rapid["guvernator"])
    print("cel mai econom : %s" % cel_mai_econom["guvernator"])
    if cel_mai_rapid["guvernator"] == cel_mai_econom["guvernator"]:
        print("-> Acelasi castigator: 'race to idle' functioneaza pe aceasta platforma.")
    else:
        print("-> Castigatori diferiti: exista un compromis intre viteza si energie.")

if __name__ == "__main__":
    sys.exit(main())

```

```
$ rulare
```

```
cd ~/si-lab/lab01
source ~/si-lab/bin/activate
python3 banc.py
```

12 Experimentul și interpretarea

Pe **Raspberry Pi 5**, un tabel tipic de rezultate arată cam așa (valorile dumneavoastră vor diferi în funcție de răcire, alimentare și versiunea firmware-ului):

Guvernator	Durată	Frecvență medie	Putere medie	Energie	Mcicli/J
powersave	16,8 s	1500 MHz	4,1 W	68,9 J	5,81
ondemand	10,6 s	2380 MHz	6,5 W	68,9 J	5,81
schedutil	10,5 s	2390 MHz	6,6 W	69,3 J	5,77
performance	10,5 s	2400 MHz	6,6 W	69,3 J	5,77

REZULTATUL CARE SURPRINDE

Energia este aproape aceeași peste tot, deși puterea diferă cu 60 %. Exact asta prezice teoria: puterea dinamică scade liniar cu frecvența, dar durata crește cu $1/f$, iar produsul rămâne constant. Ce mai rămâne pe masă este energia statică - și pe un procesor de aplicație cu scurgeri însemnate, ea înclină ușor balanța spre „termină repede”.

CE SĂ AȘTEPTĂȚI PE RASPBERRY PI 3

Cu doar două trepte de frecvență, tabelul se împarte în două grupuri: powersave rulează la 600 MHz, iar ondemand, schedutil și performance urcă toate la frecvența maximă și dau rezultate aproape identice. Diferența importantă față de Pi 5 este ce măsurați: INA219 vede **toată placa**, cu tot consumul de repaus al memoriei, al portului USB și al rețelei. În limbajul exploratorului din secțiunea 5, „puterea de sistem” este mare, iar teoria prezice că powersave va consuma **mai multă** energie totală, deși puterea lui este mai mică. Verificați dacă măsurătoarea confirmă predicția și comparați și **energia utilă**, nu doar pe cea totală.

Ce trebuie să comparați în referat

- **Durata** - cât de repede s-a terminat sarcina
- **Energia totală** - cât a costat efectiv bateria
- **Energia utilă** (peste consumul de repaus) - cât s-a plătit pentru calculul propriu-zis, fără infrastructura din jur
- **Mcicli/J** - eficiența, adică performanța pe watt
- **Temperatura finală** - pentru a verifica dacă a intervenit limitarea termică

Cum verificați că măsurătoarea este validă

- ☐ Rulați fiecare configurație de trei ori; dacă rezultatele diferă cu peste 5 %, ceva variază necontrolat
- ☐ Lăsați plăcuța să se răcească între rulări (verificați cu `vcgencmd measure_temp`)
- ☐ Închideți interfața grafică: `sudo systemctl isolate multi-user.target`
- ☐ Verificați `vcgencmd get_throttled` după fiecare rulare - trebuie să întoarcă `0x0`
- ☐ Pe Pi 5: comparați puterea de la PMIC cu cea de la INA219; diferența constantă este pierderea în regulatoare
- ☐ Pe Pi 3: `vcgencmd get_throttled` trebuie să rămână `0x0` și cu INA219 montat; dacă apare subtensiune, șuntul și firele coboară prea mult tensiunea (vezi regulile de la secțiunea 9)

13 Sarcini de lucru

- Rulați `stare_cpu.py` în repaus și sub încărcare. Notați frecvențele celor patru nuclee în ambele situații și explicați de ce nu sunt neapărat egale.
- Rulați `putere_pmic.py` în repaus și sub `stress-ng --cpu`
- Realizați un tabel cu puterea fiecărei linii de alimentare în ambele cazuri și calculați cât la sută din creșterea totală provine de la `VDD_CORE`.
Pe Pi 3: rulați `putere_ina219.py` în aceleași două situații. Nu aveți repartiția pe linii, deci calculați creșterea puterii întregii plăci și notați, cu `vcgencmd measure_volts core`, cum se schimbă tensiunea nucleului.
- Montați LCD-ul și rulați `afisaj.py`. Modificați programul pentru ca al doilea rând să alterneze la fiecare două secunde între temperatură și energia acumulată de la pornire.
- Rulați bancul complet și completați tabelul din secțiunea anterioară cu propriile valori.
- Trasați, pe baza fișierului CSV, două grafice: energia în funcție de guvernator și eficiența (Mci/J) în funcție de guvernator. Comentați dacă cele două conduc la aceeași concluzie.
- Fixați manual frecvența maximă la 1500, 1800, 2100 și 2400 MHz și repetați măsurătoarea:

```
$ fixarea frecvenței
```

```
echo 1800000 | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_max_freq
```

Trasați curba energie–frecvență și comparați forma ei cu cea din exploratorul DVFS de la secțiunea 5.

Pe Pi 3 există doar două trepte, deci curba are două puncte: fixați 600000 și 1200000 (1400000 pe B+) și comparați cele două energii. Discutați ce vă poate spune o curbă din două puncte și ce nu.

14 Aplicație de aprofundare

GUVERNATORUL PROPRIU

Scrieți în Python un guvernator în spațiul utilizator care să respecte un **buget de putere** fix, dat de utilizator (de exemplu 4 W). Programul monitorizează continuu puterea (de la PMIC pe Pi 5, de la INA219 pe Pi 3) și, dacă aceasta depășește bugetul, coboară `scaling_max_freq` cu o treaptă; dacă rămâne sub buget mai mult de trei secunde, o urcă înapoi cu o treaptă. Afișați pe LCD bugetul, puterea curentă și limita de frecvență în vigoare.

Aceasta este, în esență, exact ceea ce fac procesoarele de server sub numele de *RAPL power capping*.

Întrebarea la care trebuie să răspundeți în referat: **cât de repede trebuie să reacționeze bucla ca sistemul să nu oscileze?** Încercați perioade de 0,1 s, 1 s și 5 s și descrieți comportamentul în fiecare caz.

Pe Pi 3, cu numai două trepte, bucla nu poate decât să comute între 600 MHz și frecvența maximă. Alegeți un buget între puterea de repaus și cea de sarcină maximă și comparați oscilația cu cea de pe Pi 5.

15 Întrebări de verificare

Reducem frecvența la jumătate și, odată cu ea, tensiunea cu 20 %. Cu cât scade puterea dinamică?

☐ La 50 %☐ La 32 %☐ La 40 %☐ La 80 %

Un procesor rulează o sarcină în 10 s la 6 W, iar la frecvență redusă în 20 s la 3 W. Ce s-a întâmplat cu energia?

☐ S-a înjumătățit☐ S-a dublat☐ A rămas aceeași (60 J)☐ Nu se poate ști fără tensiune

De ce curenții de scurgere favorizează strategia „race to idle”?

Pentru că cresc cu frecvența

Pentru că apar cât timp circuitul e alimentat, deci se acumulează cu timpul

Pentru că depind de temperatura mediului

Pentru că apar doar la comutarea tranzistoarelor

Ce indică bitul 2 din valoarea întoarsă de vcgencmd get_throttled?

Subtensiune la alimentare

Limitare termică activă în acest moment

Cardul microSD este lent

Guvernatorul este powersave

De ce trebuie adunată căderea de tensiune pe șunt la citirea INA219?

Ca să compensăm eroarea convertorului

Pentru că senzorul măsoară după rezistorul de măsură, iar sarcina vede tensiunea de dinaintea lui

Pentru că șuntul se încălzește

Nu trebuie adunată, este o greșeală

De ce nu include măsurătoarea PMIC întregul consum de la priză?

Pentru că nu măsoară procesorul

Pentru că raportează puterea de după regulatoare, fără pierderile acestora și fără consumul porturilor USB

Pentru că are o rezoluție prea mică

Pentru că funcționează doar în repaus

Ce măsoară indicatorul „Mcicli pe joule”?

Viteza procesorului

Eficiența energetică, adică performanța obținută pe fiecare joule cheltuit

Temperatura per watt

Numărul de nuclee active

Pe un Raspberry Pi 3, vcgencmd pmic_read_adc răspunde „Command not registered”. Ce înseamnă?

Firmware-ul trebuie actualizat

Placa nu are circuit de alimentare cu măsurare, deci puterea se măsoară din exterior, cu INA219

Lipsește pachetul i2c-tools

Comanda trebuie rulată cu sudo

Răspunsuri: 1-b, 2-c, 3-b, 4-b, 5-b, 6-b, 7-b, 8-b

1. $P = C \cdot V^2 \cdot f$. Frecvența la jumătate dă factorul 0,5; tensiunea la 80 % dă factorul $0,8^2 = 0,64$. Împreună: $0,5 \cdot 0,64 = 0,32$, adică 32 %.
2. $E = P \cdot t$. $6 \text{ W} \cdot 10 \text{ s} = 60 \text{ J}$, iar $3 \text{ W} \cdot 20 \text{ s} = 60 \text{ J}$. Aceasta este capcana centrală a lucrării: puterea mai mică nu înseamnă automat energie mai mică.
3. Puterea statică curge continuu, indiferent dacă se lucrează sau nu. Cu cât sarcina durează mai mult, cu atât se acumulează mai multă energie statică - deci merită terminată repede, urmată de o stare de consum redus.
4. Bitul 2 semnalează limitare termică activă. Biții 0–3 descriu situația din acest moment, iar biții 16–19 rețin ce s-a întâmplat de la ultima pornire.
5. Tensiunea reală pe sarcină este suma dintre tensiunea de magistrală și căderea pe șunt. La 2 A pe 0,1 Ω înseamnă 0,2 V - o eroare de aproximativ 4 % dacă o ignorăm.
6. PMIC-ul raportează liniile interne de alimentare. Pierderile din reglatoare (10–20 %) și consumul perifericelor alimentate separat nu apar în această sumă.
7. Este raportul dintre lucrul util și energia consumată. Doi guvernatori pot da timpi foarte diferiți dar aceeași eficiență - și exact asta descoperim în experiment.
8. Comanda există doar pe Raspberry Pi 5, care are un PMIC cu convertoare analog-numerice pe fiecare linie. Pe Pi 3 nicio actualizare nu o aduce: puterea se măsoară cu șuntul modulului INA219, intercalat pe alimentarea plăcii.

16 Resurse

Capitolul 4 - Consumul de energie în sistemele embedded

Capitolul 5 - Tehnici de optimizare a consumului energetic

Capitolul 6 - Software power management

Documentația oficială Raspberry Pi - vcgencmd și configurarea plăcii

Documentația nucleului Linux - subsistemul cpufreq și guvernatoarele

Texas Instruments - pagina și foaia de catalog a circuitului INA219
