Power Management Praktikum
WS 2009/10

Frequency Scaling: Implementierung des PAST-Algorithmus

Tamara Mendt
Martin Fietz

PAST-Algorithmus (1)

- Annahme: CPU-Auslastung ist im nächsten Intervall genauso wie im vorherigen (gemessenen)
- Betrachtung von
  - Running Cycles
  - Hard- und Soft-Idle-Cycles
  - Excess Cycles (wegen zu geringer Taktung „übrig gebliebene“ Zyklen)

PAST-Algorithmus (2)

- bei mehr Excess Cycles als Idle Cycles wird sofort auf Maximalwert getaktet
- überschreitet die Auslastung einer oberen Schwelle (70%) wird die Taktung erhöht, unterschreitet sie untere (50%) wird mit Einbezug der Auslastung runtergetaktet

PAST-Algorithmus (3)

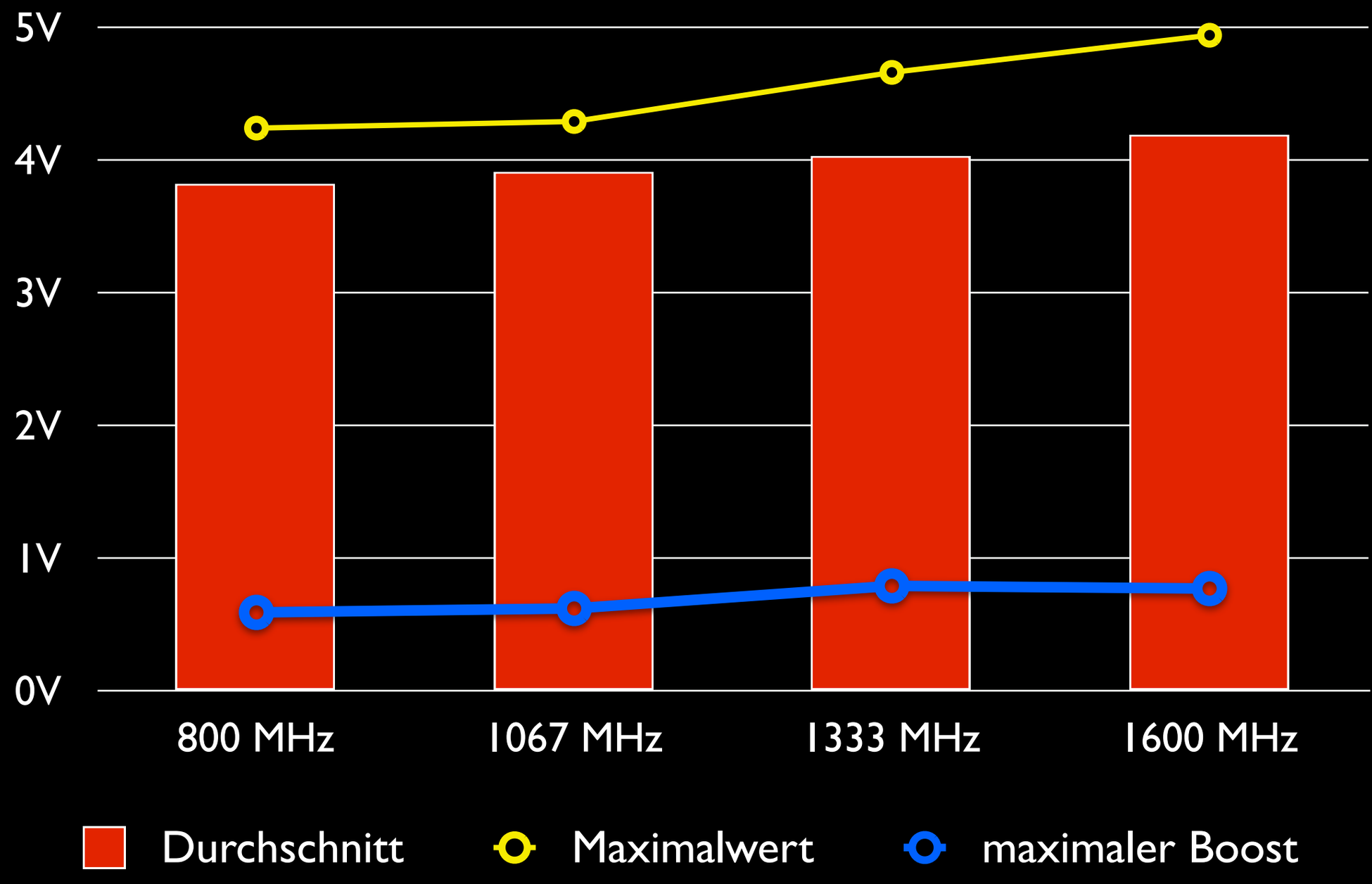
- Hard Idle (TASK_UNINTERRUPTIBLE)
  - nicht eliminierbar
  - Warten auf Hardware, z.B. Festplatte
- Soft Idle (TASK_INTERRUPTIBLE)
  - Warten auf Benutzereingabe
  - kann in der Theorie durch Re-Scheduling „eliminiert“ werden

Implementierung

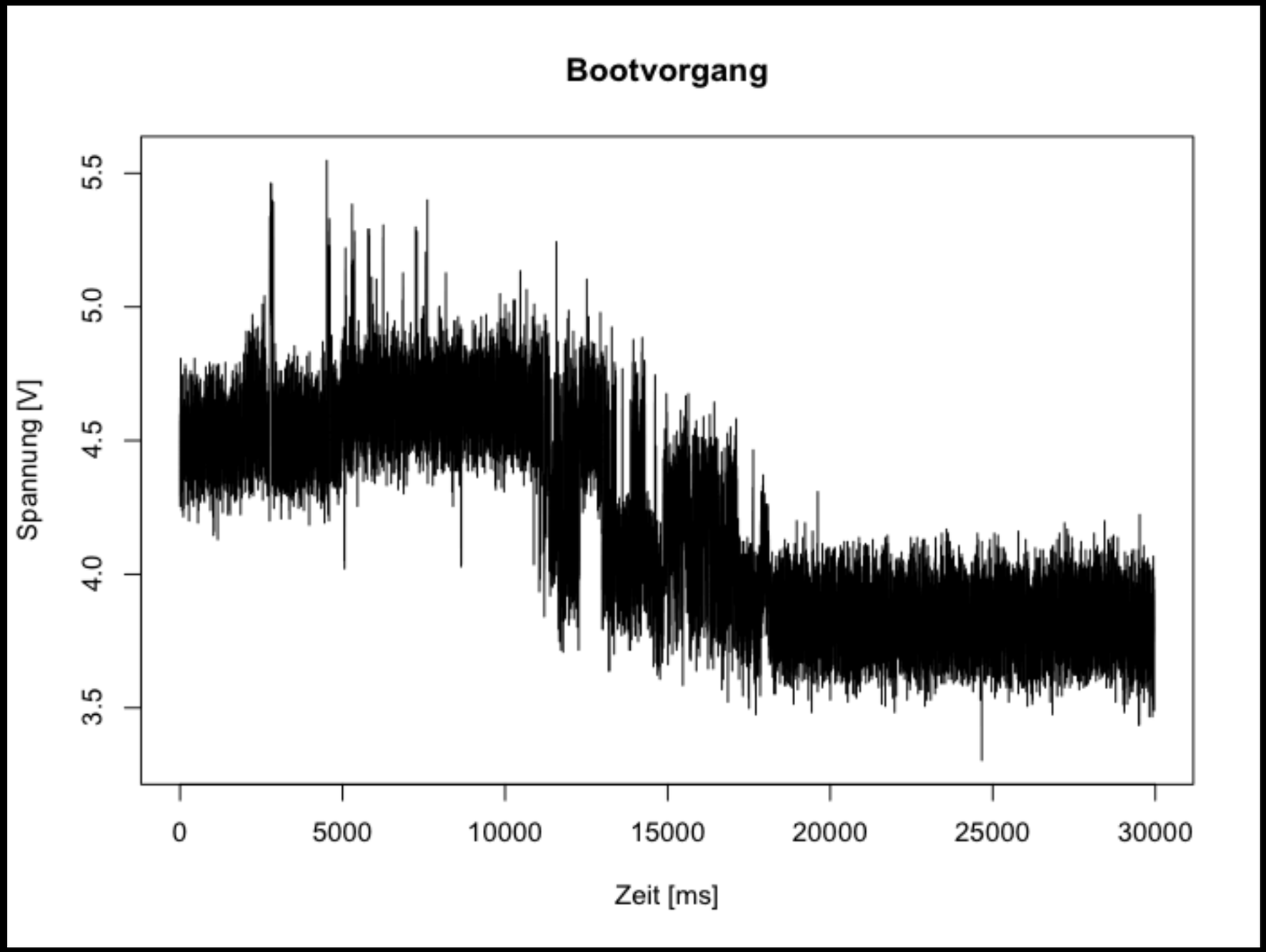
- Linux Scheduler anpassen:
  Scheduler zählt bereits Running-, Idle- und IO-Wait-“Cycles“, wir nun zusätzlich Hard- und Soft-Idle-Cycles

- CPU Frequenz entsprechend anpassen
  Anpassung des cpufreq Governors *Convervative*

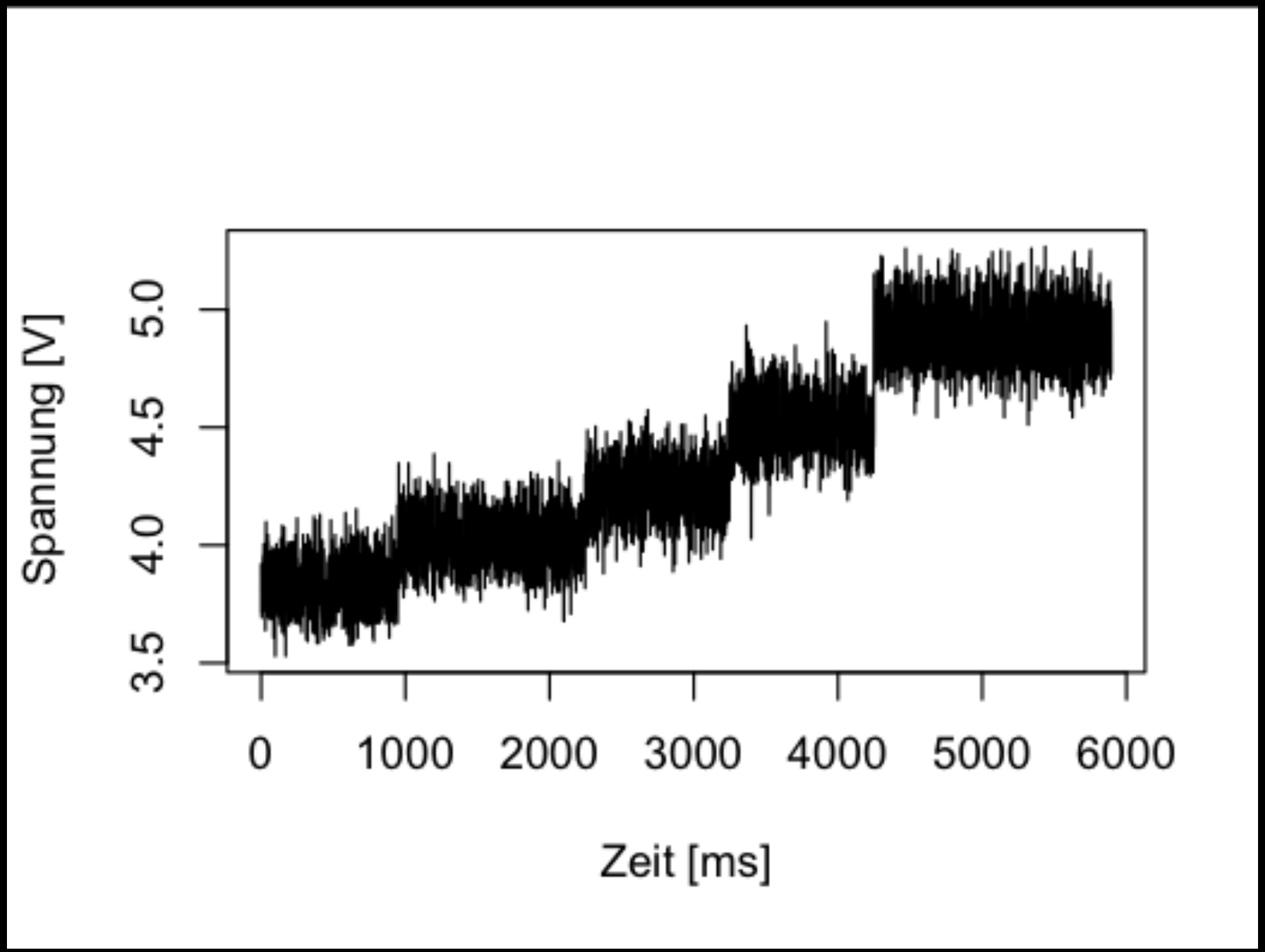
Messergebnisse



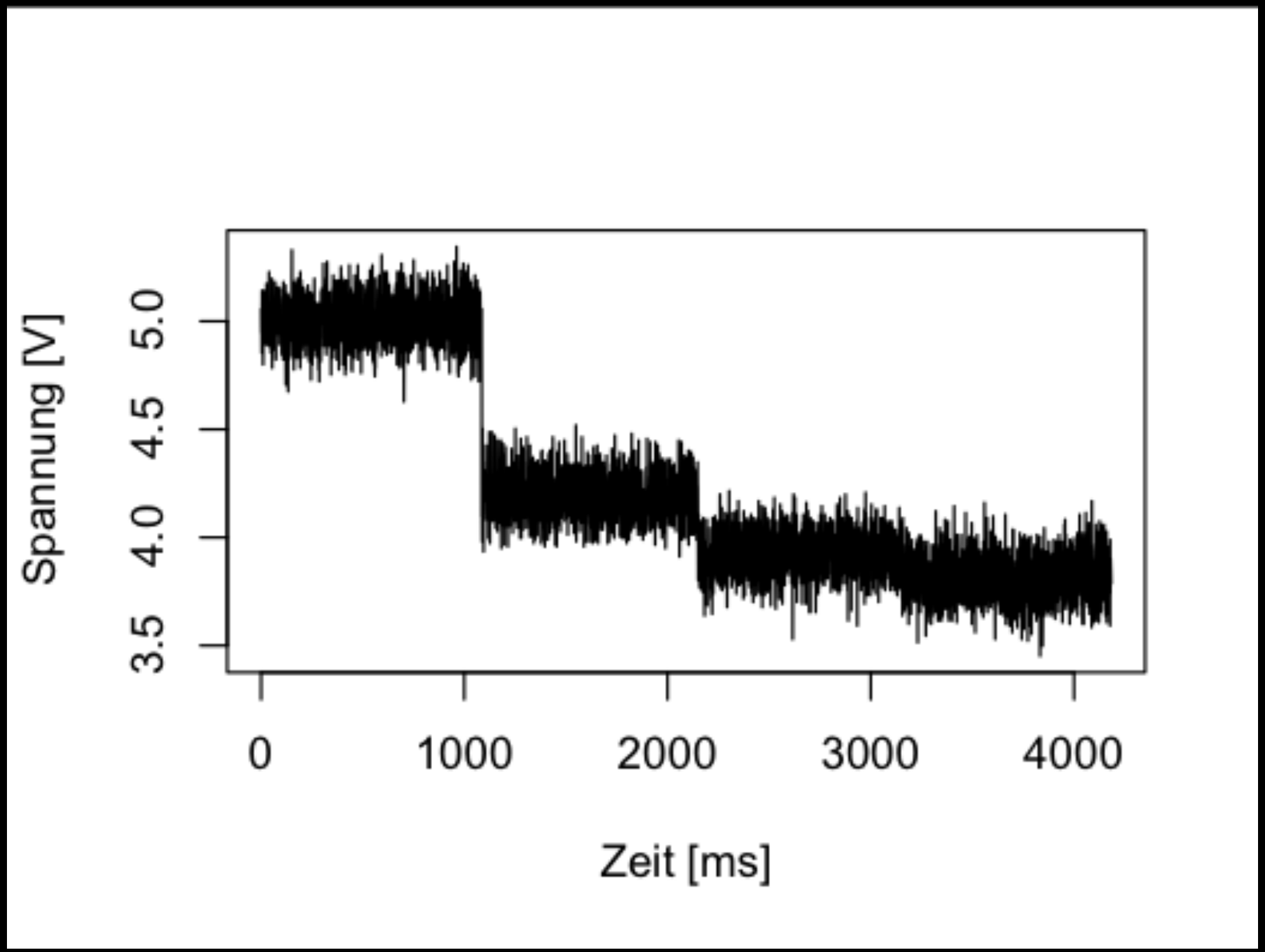
Messergebnisse (I)



Messergebnisse (2)



Messergebnisse (3)



Ergebnisse (1)

- Frequency Scaling an sich vorteilhaft, hier (Atom-CPU) jedoch kaum Unterschied feststellbar
- Hard-Idle-Cycles treten praktisch nie auf

Ausblick (1)

- mögliche Verbesserungen
  - PAST eher träge: kleineres sogar dynamisches Zeitfenster
  - Erkennung von Zugriffsmustern (jedoch sehr komplex) bzw. mehrstufige Analyse (Gewichtung mehrerer Zeitfenster)
  - Anpassung der Schwellwerte für Taktveränderung

Ausblick (2)

- cpufreq Governor Conversative
  - sehr ähnlich zu PAST
  - keine Unterscheidung zwischen Hard-Idle- und Soft-Idle-Cycles
  - bei Auslastung über 80% (unter 20%) wird schrittweise (5%) hoch (runter) getaktet
  - für Batterie-Betrieb geeignet (laut Dokumentation)