Embedded Systems

Power Sources & Low-Power Design

Jem Berkes
`http://berkes.ca`

ECE, University of Manitoba

Review of basics

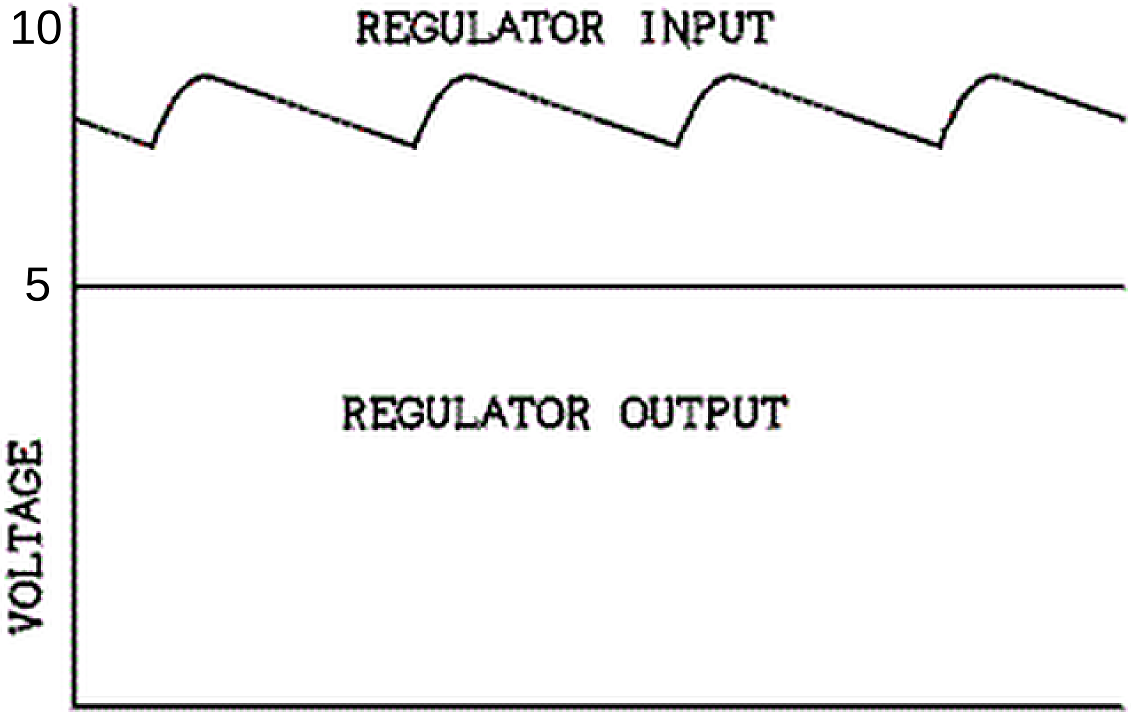
Powering Embedded Systems

- Generally need DC
- Regulated voltage
- Must supply enough current
- Motivations for lower power design

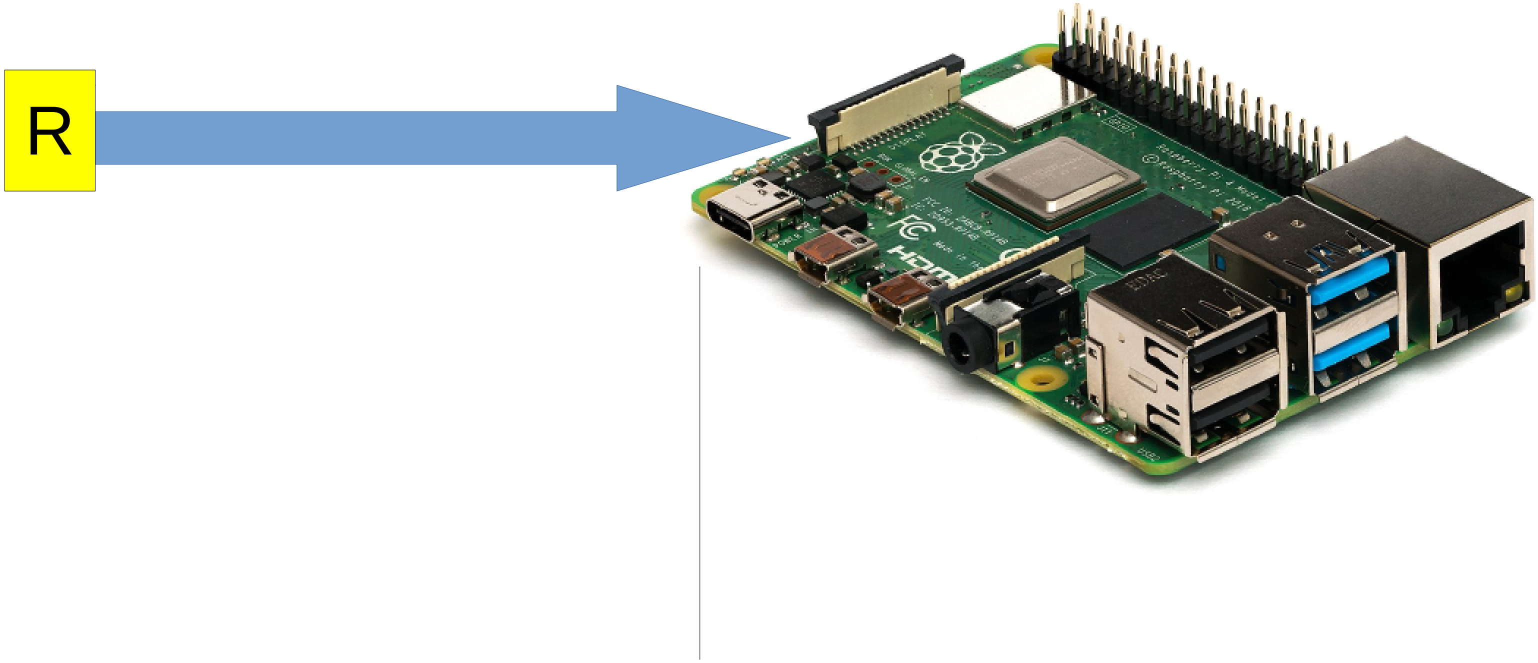
Voltage Regulation

Uneven input

Steady output



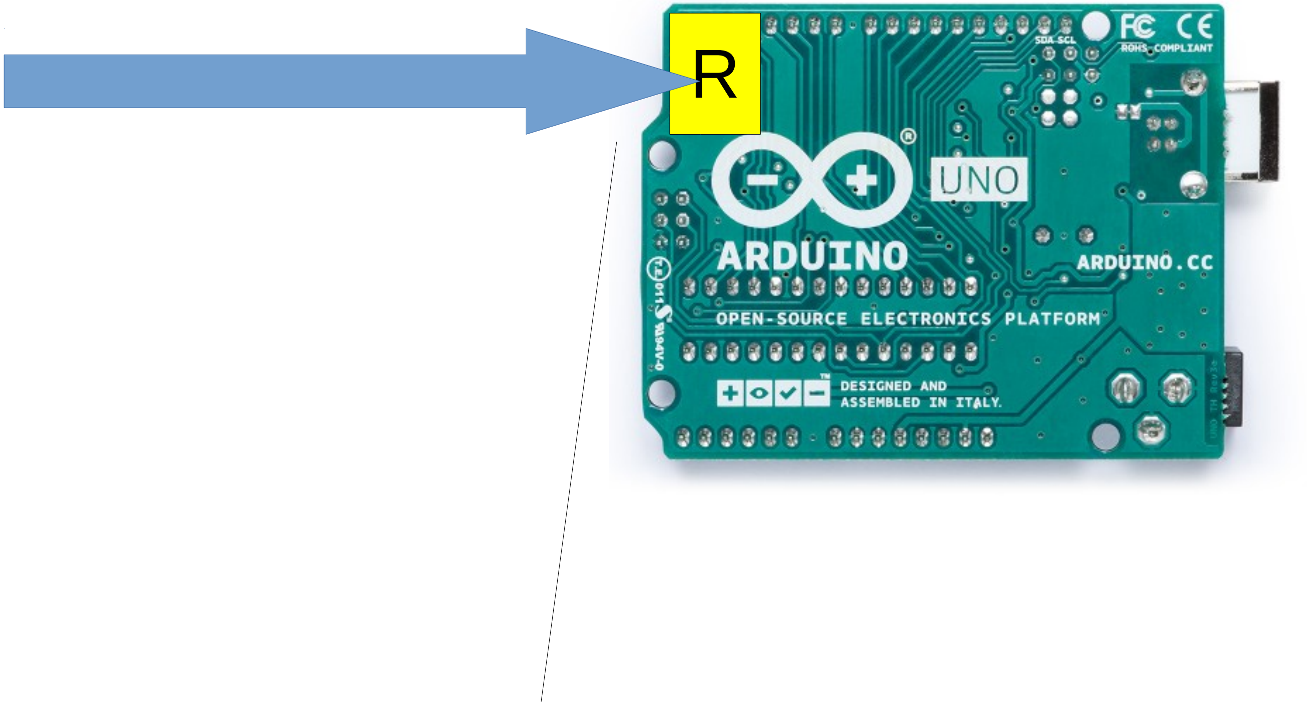
Where is regulator? (Pi)



Expects 4.8 – 5.2 V

Input must be regulated

Where is regulator?


R
UNO
ARDUINO
ARDUINO.CC
OPEN-SOURCE ELECTRONICS PLATFORM
DESIGNED AND ASSEMBLED IN ITALY.
SDA SCL
ROHS COMPLIANT

Expects 7 – 12 V

Board has regulator

Current Draw

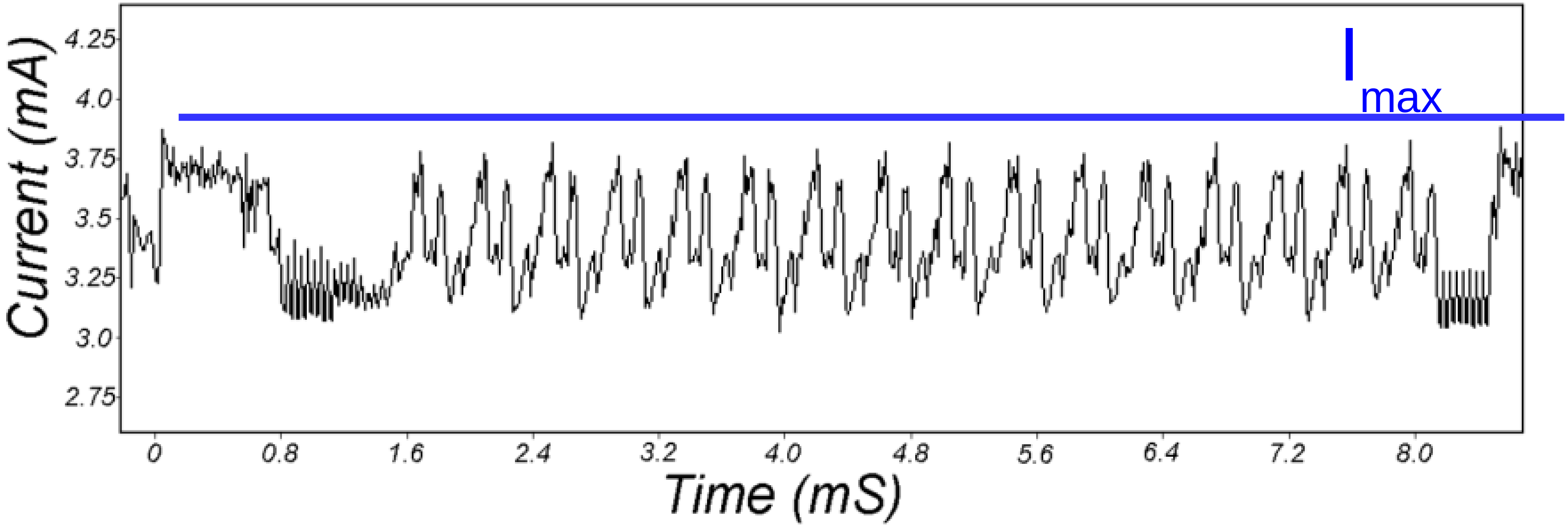
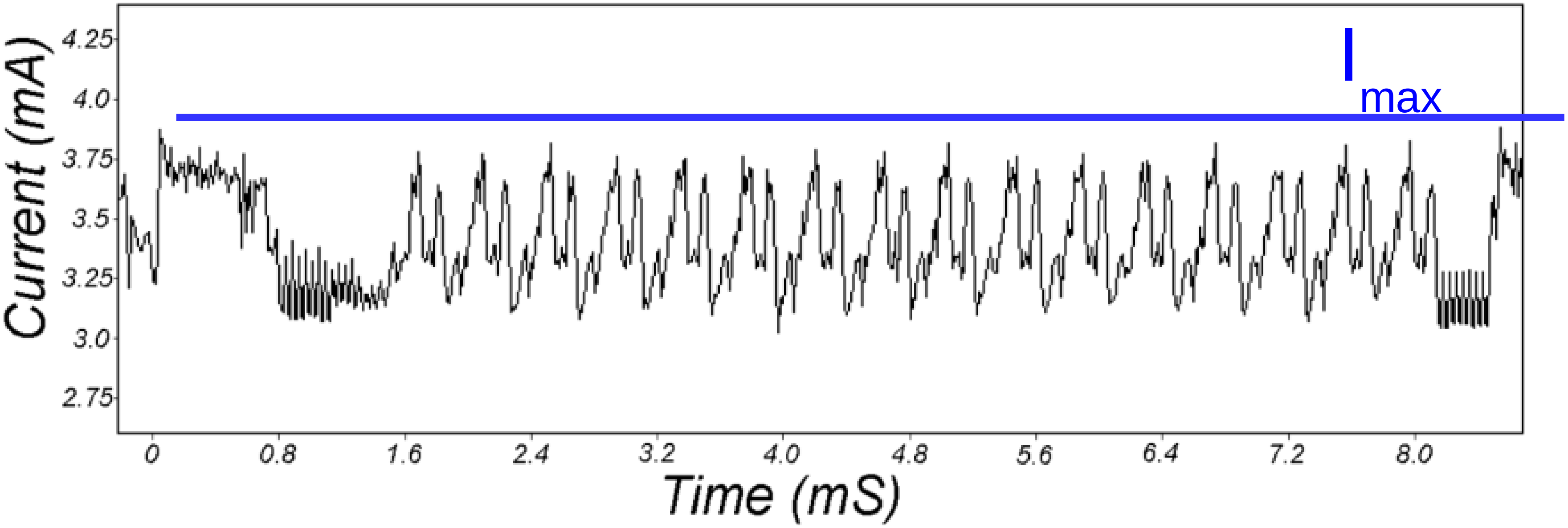


Image from *Differential Power Analysis* (Kocher, Jaffe, Jun)

Current Draw



- Current can spike; plan conservatively
- Power supplies are rated for max current
- If power supply can't keep up, device malfunctions

Image from *Differential Power Analysis* (Kocher, Jaffe, Jun)

Question

- What would the **power consumption** plot look like?

Software Design
for Low Power

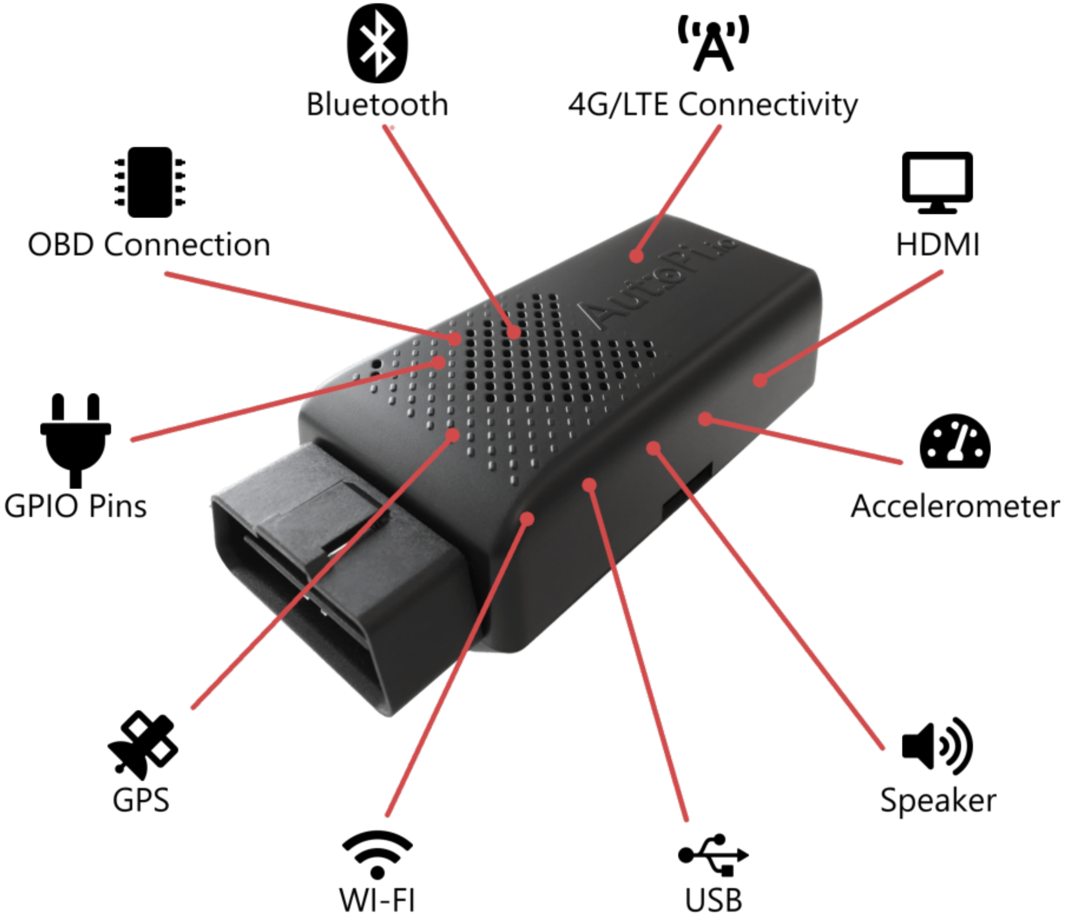
Best Practices

- Turn things off
- Idle is good!
  - Read sensors intermittently (low sample rate)
  - Allows CPU to save power
- System-wide sleep/suspend
- Wi-Fi Sleep

- Mechanisms are highly system-dependent
  - Optimizing requires real work

Turn things off

- Disable interfaces (peripherals, wireless)



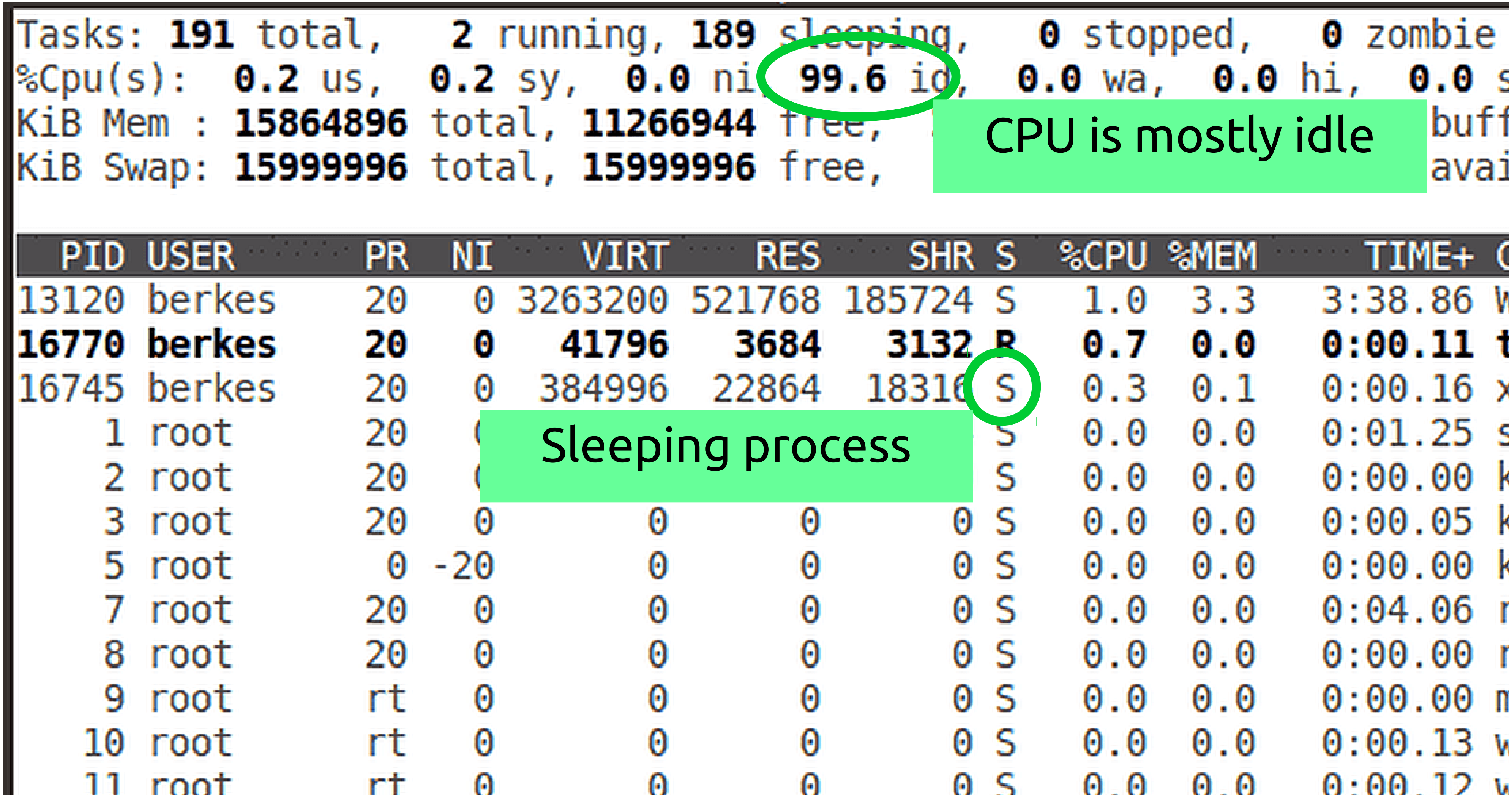
Idle is Good!

- Wait in the right way. Avoid “busy wait”

```
while (!user_interrupt)
   sigsuspend (&oldmask);
```

- Can suspend and wait for event
  - UNIX signals, timers
  - External inputs
- Sample external sensors at very low rates
  - Sleep in between

‘top’ gives clues



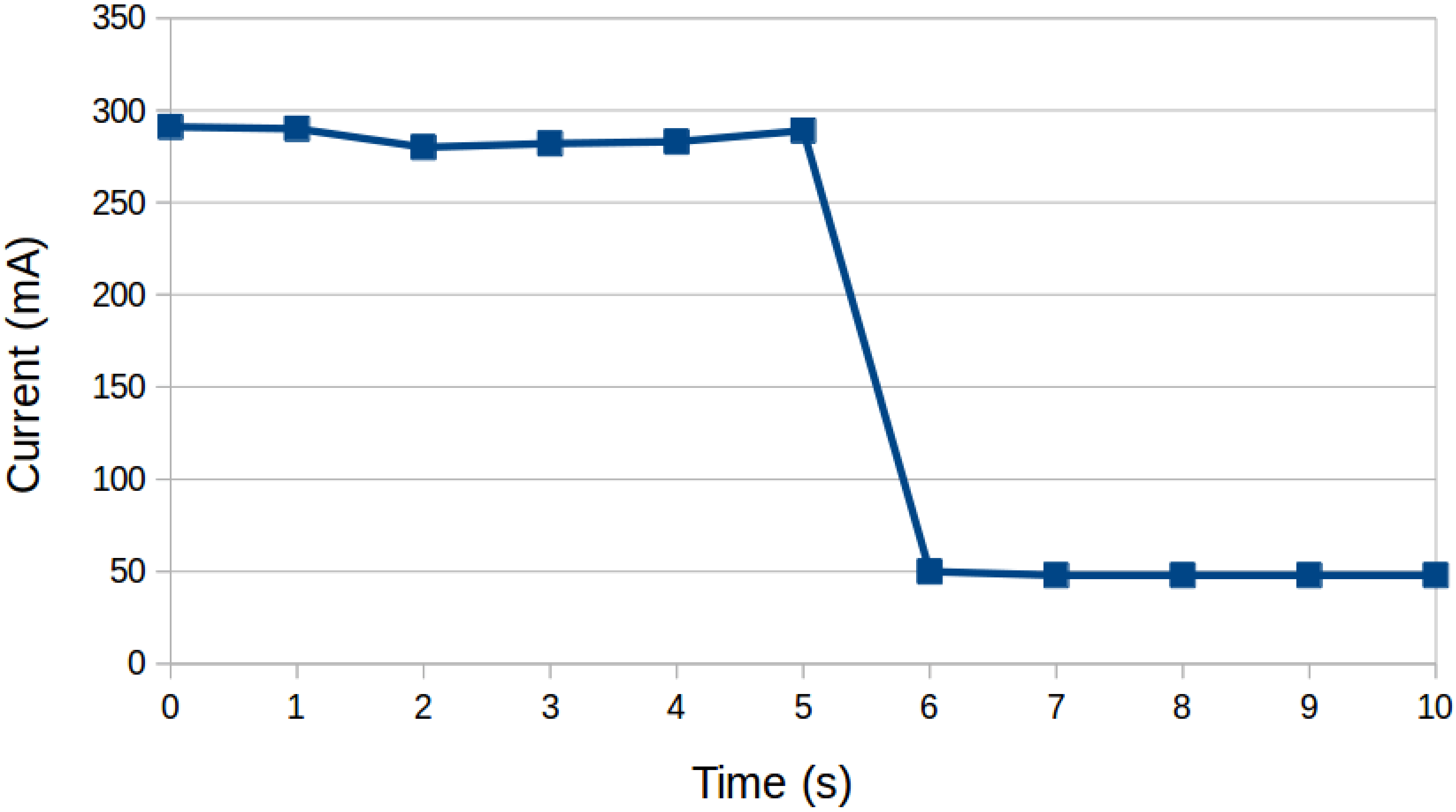
```
Tasks: 191 total,   2 running, 189 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.2 sy,  0.0 ni, 99.6 id,  0.0 wa,  0.0 hi,  0.0 s
KiB Mem : 15864896 total, 11266944 free,                      buff
KiB Swap: 15999996 total, 15999996 free,                      avai

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ C
13120 berkes    20   0 3263200 521768 185724 S   1.0  3.3   3:38.86 W
16770 berkes    20   0   41796   3684   3132 R   0.7  0.0   0:00.11 t
16745 berkes    20   0  384996  22864  18316 S   0.3  0.1   0:00.16 x
    1 root      20   (                       S   0.0  0.0   0:01.25 s
    2 root      20   (                       S   0.0  0.0   0:00.00 k
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.05 k
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 k
    7 root      20   0       0      0      0 S   0.0  0.0   0:04.06 r
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 r
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 m
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.13 w
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.12 w
```

CPU is mostly idle

Sleeping process

System-wide sleep/suspend

```
$ cat /sys/power/state
freeze standby mem disk
$ echo standby > /sys/power/state
```

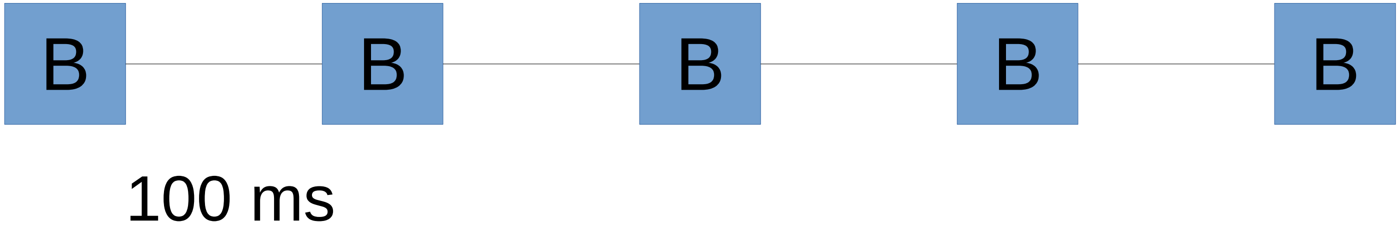


Wi-Fi, Bluetooth, etc

- WiFi has its own controller
- Many have interfaces to **sleep**
- Automatic or on-demand, tunable sleep interval

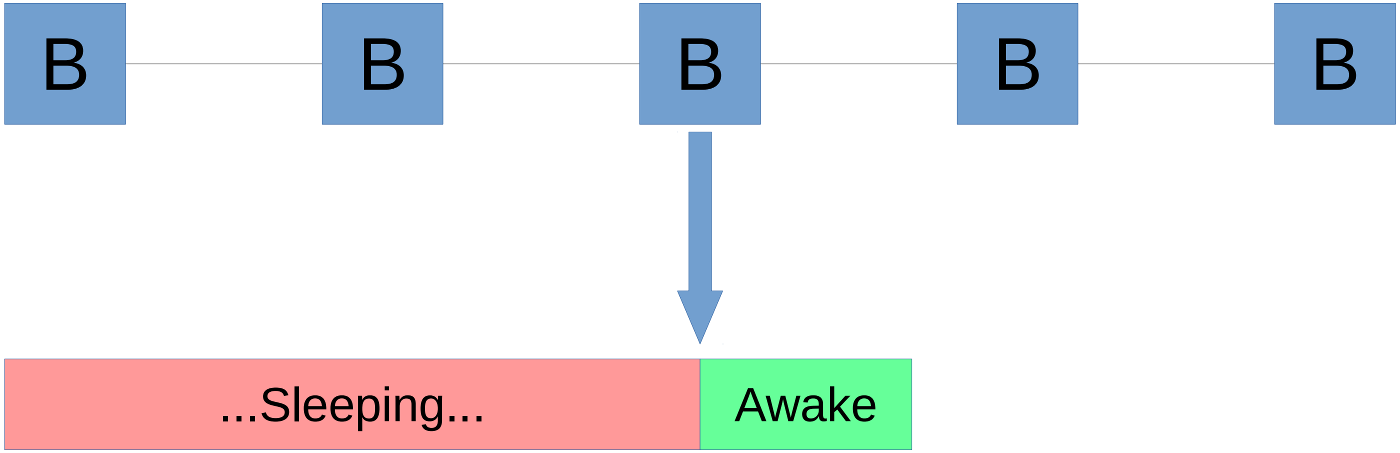
Wi-Fi Sleep

- Access Points send regular “beacon” frames
- Even sleeping Wi-Fi clients can receive them



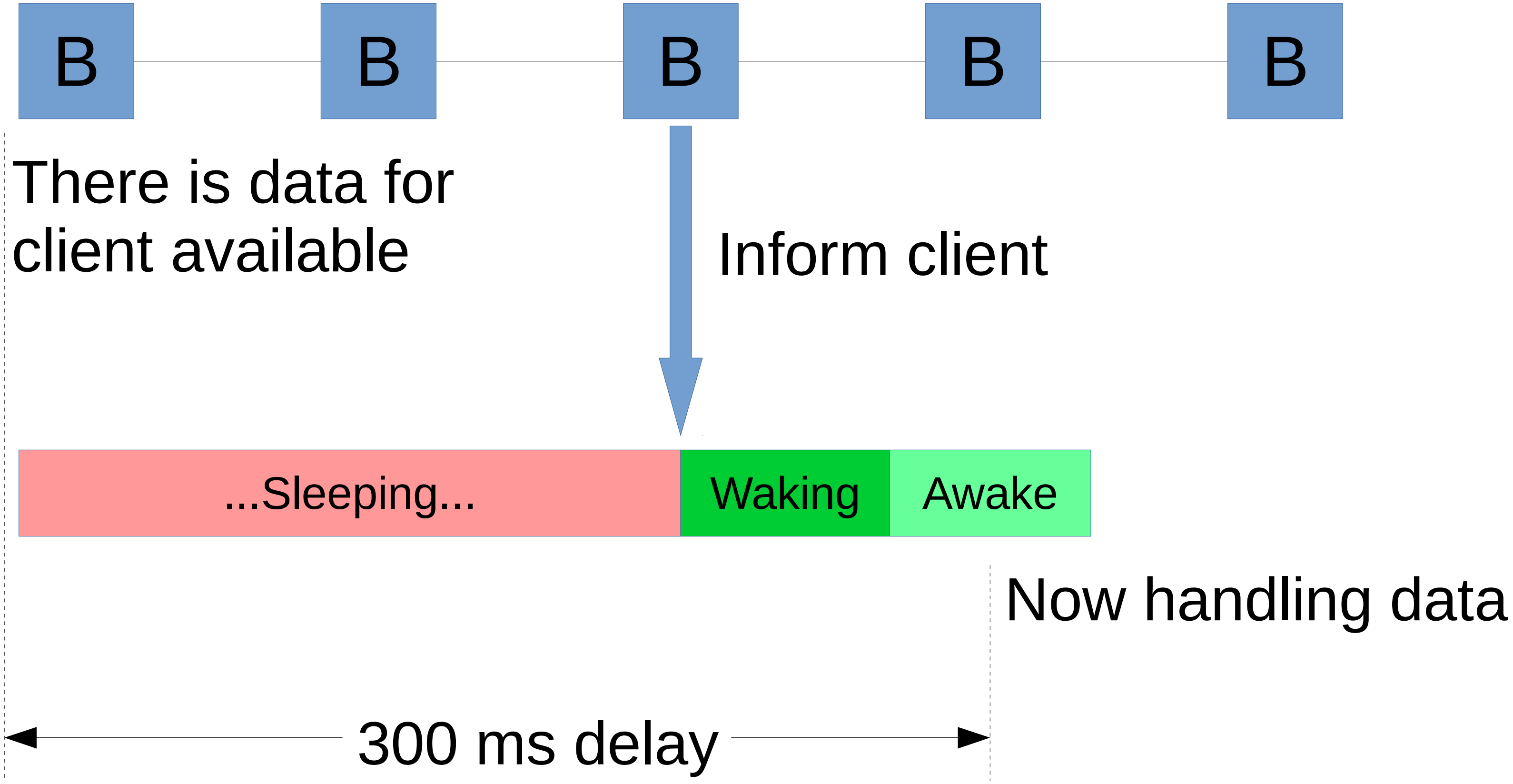
Wi-Fi Sleep

- Clients have a “DTIM Interval” (sleep interval) setting
- Pictured: interval=3



Wi-Fi Sleep

- Sleep might cause latency and even packet loss



Wi-Fi Design Considerations

- Sleeping is good... saves power
- But
  - Latency and dropped packets
  - Connections might *break*
- Designing for sleep
  - Select sleep interval carefully
  - Design robust protocol, handle disconnects

Batteries

Calculations

- Determine I_{max} and ensure supply can provide it
- Determine I_{avg}
- Learn battery’s milliamp-hours (mAh) rating
- Caveats
  - Voltage regulators lose power
  - Batteries age
  - Power packs often use marketing trick
    (overstates mAh)

Conservative Adjustments

- Reduce battery mAh rating by 30%
  - Accounts for marketing trick/regulator loss
- Plan for 50% empty battery
  - Accounts for aging and safety margin

Example Calculation

- Raspberry Pi with I_{avg} = 600 mA and I_{max} = 1200 mA
- USB 5V Mobile Charger, “5000 mAh”, max 2500 mA
- Check
  - $I_{max} <$ 2500 mA (ok)
- Adjust battery capacity down to 3500 mAh
- Time on battery = 3500 mAh / 600 mA = 5.8 hours
- Plan for 50% battery
  - **Conservative answer is 2.9 hours**