

IoT – design aspects

MOBILE AND CYBER-PHYSICAL SYSTEMS

STEFANO CHESSA

1

Learning objectives



IoT
characteristics



Energy
efficiency



Duty cycle

most important:

else init. ref.: env.,
sys.)

2

limitation of technology: energy density

1

IoT devices characteristics

Each device is usually:

- Low power, low cost system
- Small
- Autonomous

⇒ put them there,
forget them
respect on what's
happening around

equipped with: Full microsys

- 1) ◦ Processor
- 2) ◦ Memory
- 3) ◦ Radio Transceiver
- 4) ◦ Sensing devices
 - Acceleration, pressure, humidity, light, acoustic, temperature, GPS, magnetic, ...
- 5)
- 6) ◦ Actuators – depending on the case
 - Battery, solar cells, wind turbines

freedom on deployment everywhere

3

Issues in IoT design

Energy efficiency

- Sensors are battery-powered or use energy harvesting → lifetime will be limited
- Need for HW/SW energy efficient solutions

Adaptability to changing conditions

- Need for dynamic network management & programming

Low-complexity, low overhead protocols

- Need at any level of the protocol stack due to limitation of nodes' resources

Security

- At all layers of the stack

Multihop communications

- Need for protocol stacks & routing protocols

Mobility (mobile net communication)

- Need for dynamic routing protocols

Data storage & (pre-)processing

...

on. flag
re-programming
env. for 10 years
→ need adapt
it's standard
or interoperability
with other device

4

devices are low-powered
caps. of mem., communica.,
processing → need algo to
specific to fight with these limitations

1) periodically replace battery by solar cell
2) charge battery
3) use (des.) energy
4) use own

5) replace battery

6) people time

7) (des.) energy

8) use own

9) use own

10) use own

11) use own

12) use own

13) use own

14) use own

15) use own

16) use own

17) use own

18) use own

19) use own

20) use own

21) use own

22) use own

23) use own

24) use own

25) use own

26) use own

27) use own

28) use own

29) use own

30) use own

31) use own

32) use own

33) use own

34) use own

35) use own

36) use own

37) use own

38) use own

39) use own

40) use own

41) use own

42) use own

43) use own

44) use own

45) use own

46) use own

47) use own

48) use own

49) use own

50) use own

51) use own

52) use own

53) use own

54) use own

55) use own

56) use own

57) use own

58) use own

59) use own

60) use own

61) use own

62) use own

63) use own

64) use own

65) use own

66) use own

67) use own

68) use own

69) use own

70) use own

71) use own

72) use own

73) use own

74) use own

75) use own

76) use own

77) use own

78) use own

79) use own

80) use own

81) use own

82) use own

83) use own

84) use own

85) use own

86) use own

87) use own

88) use own

89) use own

90) use own

91) use own

92) use own

93) use own

94) use own

95) use own

96) use own

97) use own

98) use own

99) use own

100) use own

101) use own

102) use own

103) use own

104) use own

105) use own

106) use own

107) use own

108) use own

109) use own

110) use own

111) use own

112) use own

113) use own

114) use own

115) use own

116) use own

117) use own

118) use own

119) use own

120) use own

121) use own

122) use own

123) use own

124) use own

125) use own

126) use own

127) use own

128) use own

129) use own

130) use own

131) use own

132) use own

133) use own

134) use own

135) use own

136) use own

137) use own

138) use own

139) use own

140) use own

141) use own

142) use own

143) use own

144) use own

145) use own

146) use own

147) use own

148) use own

149) use own

150) use own

151) use own

152) use own

153) use own

154) use own

155) use own

156) use own

157) use own

158) use own

159) use own

160) use own

161) use own

162) use own

163) use own

164) use own

165) use own

166) use own

167) use own

168) use own

169) use own

170) use own

171) use own

172) use own

173) use own

174) use own

175) use own

176) use own

177) use own

178) use own

179) use own

180) use own

181) use own

182) use own

183) use own

184) use own

185) use own

186) use own

187) use own

188) use own

189) use own

190) use own

191) use own

192) use own

193) use own

194) use own

195) use own

196) use own

197) use own

198) use own

199) use own

200) use own

201) use own

202) use own

203) use own

204) use own

205) use own

206) use own

207) use own

208) use own

209) use own

210) use own

211) use own

212) use own

213) use own

214) use own

215) use own

216) use own

217) use own

218) use own

219) use own

220) use own

221) use own

222) use own

223) use own

224) use own

225) use own

226) use own

227) use own

228) use own

229) use own

230) use own

231) use own

232) use own

233) use own

234) use own

235) use own

236) use own

237) use own

238) use own

239) use own

240) use own

241) use own

242) use own

243) use own

244) use own

245) use own

246) use own

247) use own

248) use own

249) use own

250) use own

251) use own

252) use own

253) use

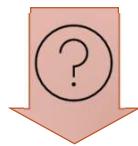
IoT design

Many limitations in the WSN design are related to processing, memory, battery and communication constraints

wireless
sensors
network

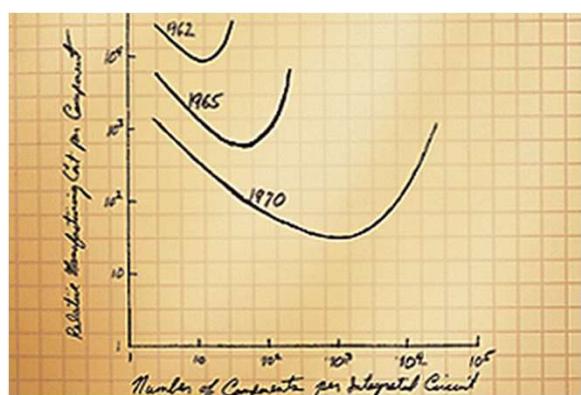
As we have
and want

technology
become
more



The evolution of HW technologies will overcome these constraints?

5



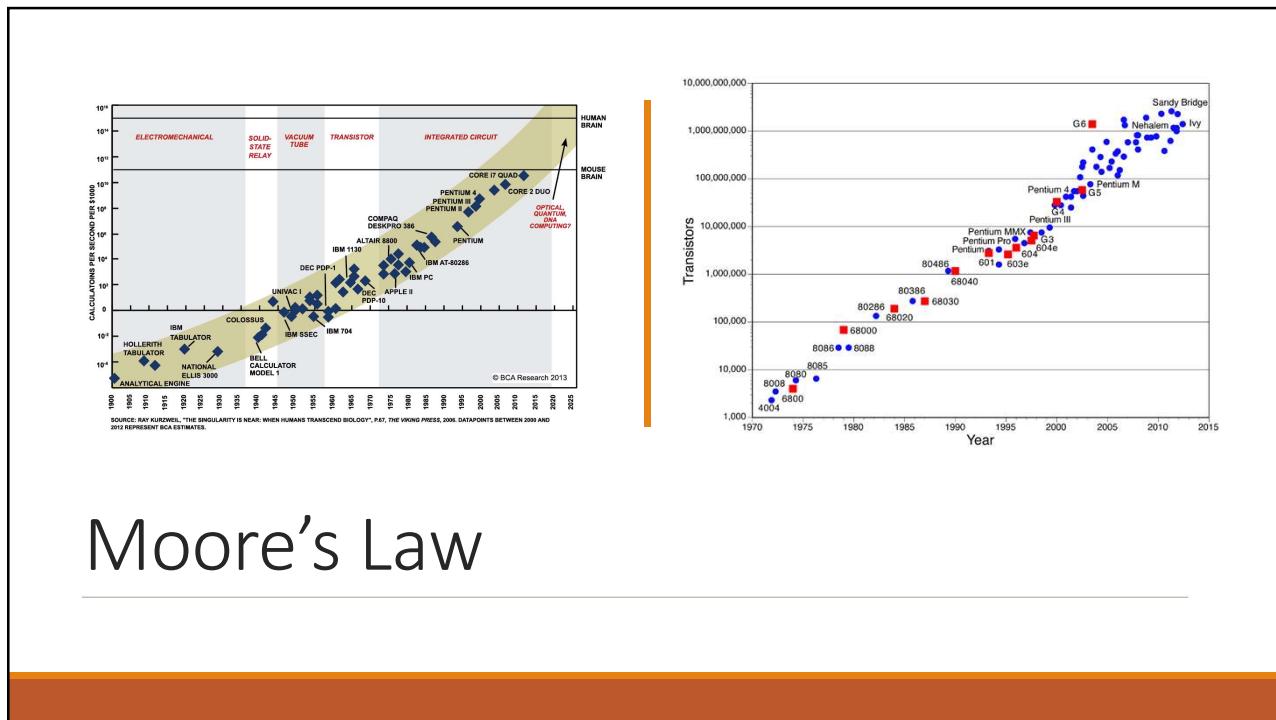
The Moore's law

"The number of transistors that can be (inexpensively) embedded in a chip grows exponentially"

(it doubles every two years)

6

3



Moore's Law

7

Moore's law growth

INTEL 4004 (1971)

2.3 K transistors

740 KHz



4KB program memory

INTEL CORE I9-7980XE (Q3'17)

1.8 G transistors

4.4 GHz

165 W - power consumption

128 GB

8

4

The Moore's law and IoT

The **Moore's law** offers three different interpretations:

1. The **performance doubles every two years** at the same cost
 - Up to now this is true for processors of servers/desktops
2. The **chip's size halves every two years** at the same cost
 - Consequently also the energy consumption is reduced
3. The size and the processing power remain the same but the **cost halves every two years**

The Moore's law and IoT

In IoT all the three interpretations are true...

There are applications that:

- Require small-sized sensors and/or that have low power consumption
- Require higher processing capabilities to the single sensor (in server performance is important, not critical in IoT but large # of devices, responding fast of time is needed to reduce costs)
- The cost is important in (almost) all applications
o ~~considering~~ (since > power || cost)

(~~more~~ performance) ▷ processing power is crucial for cameras

The Moore's law and IoT

Nowadays there exist several IoT HW platforms with different capabilities in terms of processing and energy consumption

⇒ ordering & software

Differently than server/desktop applications the IoT devices use low-power, cheap processors

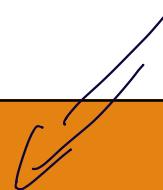
- Even old or refreshed processors that are still on the market

Normally used the cheapest HW that meets the application requirements

- considering the scale factor due to the large number of IoT devices, that has considerable effects on the final costs

11

More not resolve problem on the short time ⇒ relies on something else



Energy efficiency

cur dev. battery powered, everything wireless

⇒ soon or later

battery will die

⇒ must think

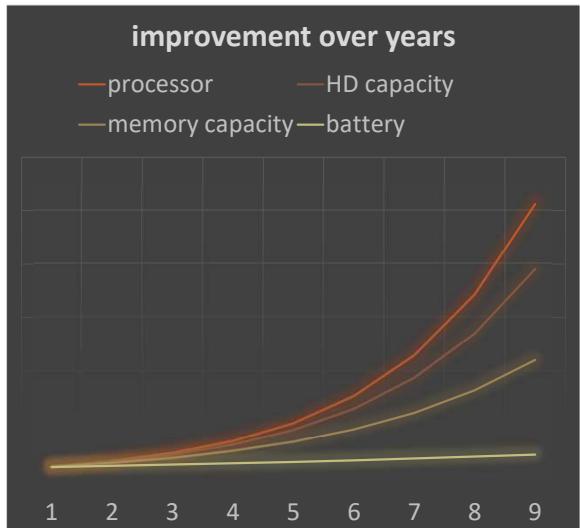
cost (↑)
do seek a better
which is
technic

Very expect to find solutions

in design IoT solutions

12

6



- improvement in
years

Energy efficiency

INTEL VS DURACELL

exp grow (use non- γ SIS)

all shows exp but not fully coping:

13

- Improve energy efficiency

1) Where spend energy
of dev? 

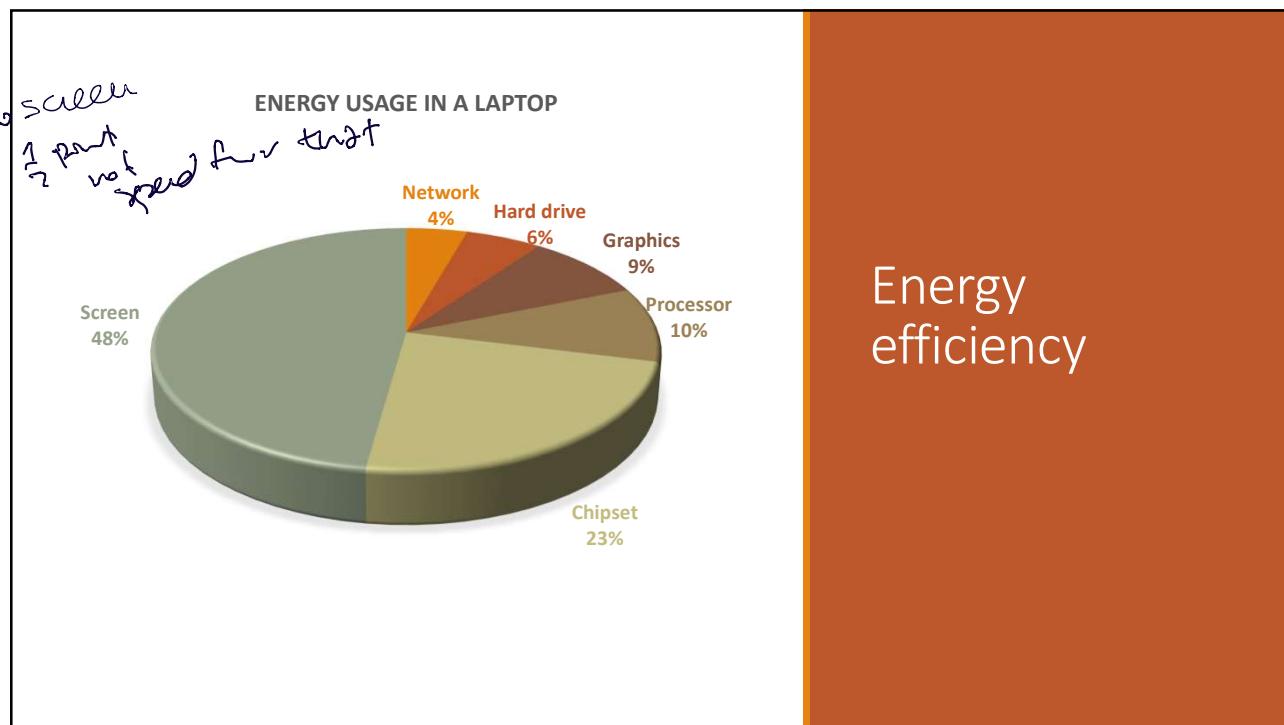
factory
on input

ed every eff,
design solution

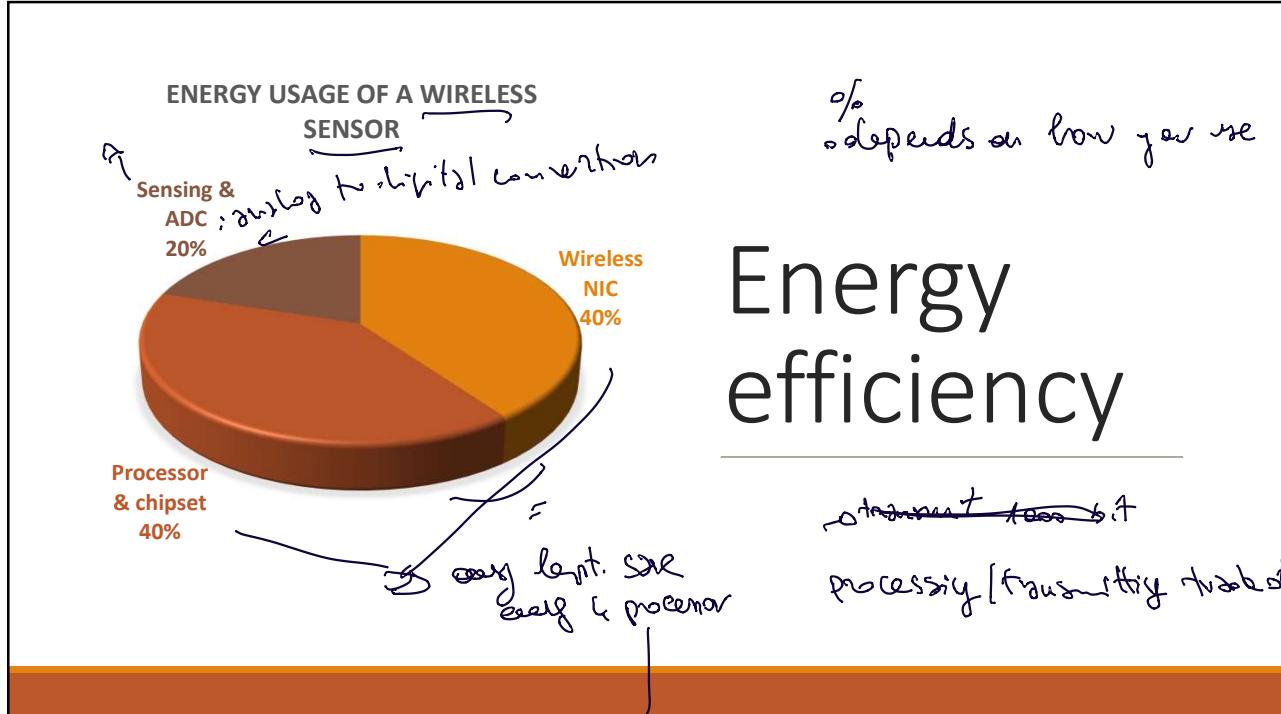
boiled to
physical
properties

to many
low

(wt %)
to monomer
low



field. & sampling (activation of sensing)
 cannot be arbitrary
 \rightarrow red sensing \Rightarrow impact possibly other \hookrightarrow mapping



Energy efficiency

make
consumption
more
efficient
¹⁵

- impact sols.
- ↳ save energy

take decision, knows about radio \Rightarrow easy to design
 - transfer is f: radio to transmit/receive
 \Rightarrow not free to control radio as you
 with other one
 (radio
over)

Example of energy consumption of a WiFi Network Interface	
Sleep mode:	10mA
(radio is forced \Rightarrow no transmit/receive unless it runs it ON)	
Listen mode:	180mA
Receive mode:	200 mA
Transmit mode:	280 mA

\Rightarrow idle energy \Rightarrow transmit less

\Rightarrow Listen \Rightarrow sleep more

Energy efficiency - radio

16

- one interface works on MAC layer

\downarrow , is similar

sort of
awake state
 ↳ no wake
 main: quite
 in short time
 , every 2-3s
 goes
 different with
 states:
 transmit/
 receive
 mode
 listening
 mode
 sleep

8

Energy efficiency - radio	<p>Energy consumption of a sensor (Mote-clone)</p> <p>Sleep mode:</p> <p>0.016 mW</p> <p>Listen mode: <i>consume less than much</i></p> <p>12.36 mW</p> <p>Receive mode:</p> <p>12.50 mW <i>don't have to amplify the signal ⇒ bit decoding analog</i></p> <table border="1"> <tr> <td>Transmit mode</td><td>0.1 power level, 19.2kbps: 12.36 mW</td></tr> <tr> <td></td><td>0.4 power level, 19.2kbps: 15.54 mW</td></tr> <tr> <td></td><td>0.7 power level, 19.2kbps: 17.76 mW</td></tr> </table>	Transmit mode	0.1 power level, 19.2kbps: 12.36 mW		0.4 power level, 19.2kbps: 15.54 mW		0.7 power level, 19.2kbps: 17.76 mW
Transmit mode	0.1 power level, 19.2kbps: 12.36 mW						
	0.4 power level, 19.2kbps: 15.54 mW						
	0.7 power level, 19.2kbps: 17.76 mW						

17

- ~~cost~~ Mote on NOT transmitting! Listen mode *difference next: internal circuit in listen mode very* *⇒ put more power on internal ⇒ more power consumption* *⇒ reach far*
- Keep radio off while net is working

Energy efficiency - radio

Energy consumption of a sensor (Mote-clone):

- In some cases **transmit power < receive power!**
- **listen power ≈ receive power** equivalent transmission (receive less)
- Radio should be turned off as much as possible
(preserving network operating)

18

9

Turn ON/OFF a component requires ENERGY and TIME

→ IoT & from LoPy4: are not general purpose & real time interactions with Humans
⇒ BUSINESS LOGIC (sw) for each specific application

Energy efficiency - processor

Processor power around 30%-50% of total power

- Processor as well should be turned off!

Turning on and off processor and radio consumes power as well...

19

Duty cycle: 1) Read data (from sensor, "sense") : sampling physical data: efficient if periodically, at constant time.
2) Analyze: process & store
3) Transmit/receive

...
4) Wait for the next sample
...

Duty cycle

$$\text{Duty cycle} = \frac{\text{Active time}}{\text{Period of activity}}$$

Saving energy by reducing the period of activity of a sensor:

- The activity of a sensor is (mostly) repetitive:
 - Sense
 - Process & store
 - Transmit/receive
- A sensor alternates a period of activity and a period of inactivity (defines a duty cycle)
- During inactivity the energy consumption is very low
 - But processor, radio and I/O need to be freezed!

- activity of dev. is periodic
- period defines duty cycle

20

can exploit this fact: put component in sleep mode

→ n.b. → \rightarrow sensor, storage, rel., ...
↓ how save energy
↓ Perception when you're not you want network active (now)
↓ waiting in next pend

10

Example of duty cycle: code

TIME REQUIRED

- Lasts 4 milliseconds
 - Processor and sensor active (idle)
 - Lasts 1 milliseconds
 - Only processor active (idle)
 - Lasts 15 milliseconds
 - Processor and radio active (tx)
 - Lasts 380 milliseconds
 - All components idle
- Total: 400 milliseconds

PATTERNS OF ACTIVITY

**NOTE: the milliseconds are not the real ones... they are just to illustrate the concept

```

    in loop; need
    ... read sensor, running operations
    void loop() {
        // reads the input from analog pin 0:
        int sensorValue = analogRead(A0); // read electric
                                         signal input

        // converts value into a voltage (0-5V):
        float voltage = sensorValue * (5.0 / 1023.0);

        // transmits voltage over the radio
        Serial.println(voltage);

        // waits for next loop
        delay(380); wait 380ms, after this time
                     re perform all
                     the previous
                     principle: in addition component off all the time
                     duty
    }

```

21

Duty cycle is time in which component is active
in entire ~~period~~ period
of activity

Example of duty cycle: code

However, this code does not turn off any component when not in use...

SUPPOSE

Let's say you have calls like:

- `turnOn(x)` / `turnOff(x)` that turn on/off the component x forever
- `Idle(y)` that puts the processor in idle state (with low power consumption) for y milliseconds

→ needed call to turn on a component need microprocessor

Should
be on

```

void loop() {
    // reads the input from analog pin 0:
    turnOn(analogSensor);
    int sensorValue = analogRead(A0);
    turnOff(analogSensor);

    // converts value into a voltage (0-5V):
    float voltage = sensorValue * (5.0 / 1023.0);

    // transmits voltage over the radio
    turnOn(radioInterface);
    Serial.println(voltage);
    turnOff(radioInterface);

    // waits for next loop
    idle(380);  // freeze micro.
}

```

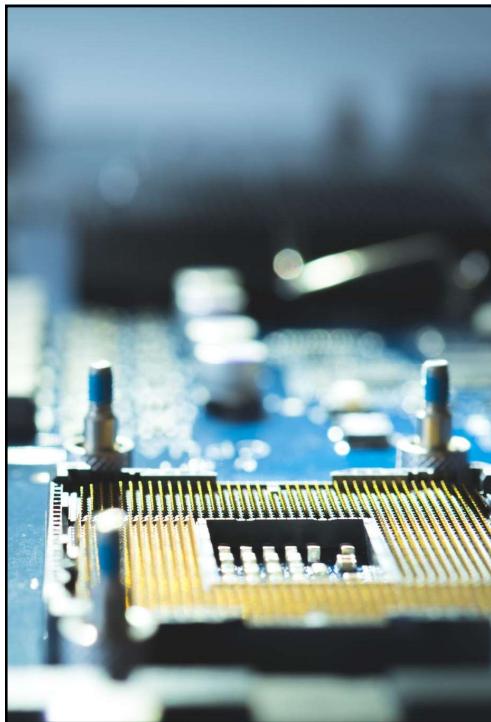
⇒ here I'm
saving very
much energy

cannot be simplified to calculate the code

22

11

3 DC: Sensor,
Microp.



Question

What is now the duty cycle of the processor, of the radio and of the sensor?

```

void loop() {
    turnOn(analogSensor);
    int sensorValue = analogRead(A0);
    turnOff(analogSensor);

    float voltage = sensorValue * (5.0 / 1023.0);

    turnOn(radioInterface);
    Serial.println(voltage);
    turnOff(radioInterface);

    idle(380);
}

```

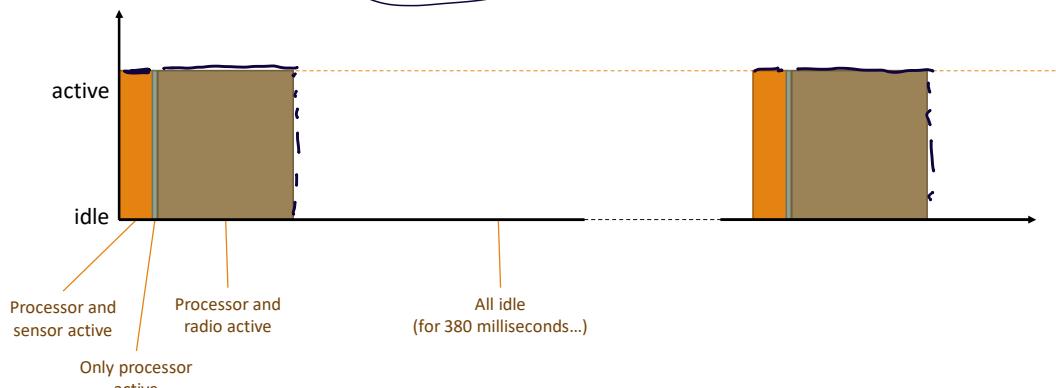
23

$$\text{Processor DC: } \frac{27}{400} \text{ ms} = \frac{1}{20}$$

$$\text{Radio DC: } \frac{3+15}{400} \text{ ms} = \frac{18}{400} \text{ ms} = \frac{3}{40} \text{ ms}$$

$$\text{Sensor: } \frac{41}{400} \text{ ms} = \frac{1}{10}$$

Example of (duty cycle) diagram



current needed by each component in the main STATE:

	Value	units
Micro Processor (Atmega128L)		
current (full operation)	8	mA
current sleep	15	μ A
Radio		
current in receive	19,7	mA
current xmit transmit	17,4	mA
current sleep idle	20	μ A
Logger (storage in the flash memory)		
write	15	mA
read	4	mA
sleep	2	μ A
Sensor Board		
current (full operation)	5	mA
current sleep	5	μ A
Battery Specifications		
Capacity Loss/Yr	3	%

- battery: keeps charge
part of energy is lost when components exist

Specs. of a mote class-sensor

25

- ① ~~loss~~: loss ability to get information
- ② amount of charge you use even ~~if~~ if you don't ~~use~~ use it

	model 1: working AT: 100% DC	model 2: 5% DC	units
<i>cycle</i>			
Micro Processor			
current (full operation)	100 <i>always active</i>	5	%
current sleep	0	95	%
Radio			
current in receive	50 <i>idle</i>	4	%
current xmit	50	1	%
current sleep	0	95	%
Logger			
write	1	1	%
read	2	2	%
sleep	97	97	%
Sensor Board			
current (full operation)	100	1	%
current sleep	0	99	%

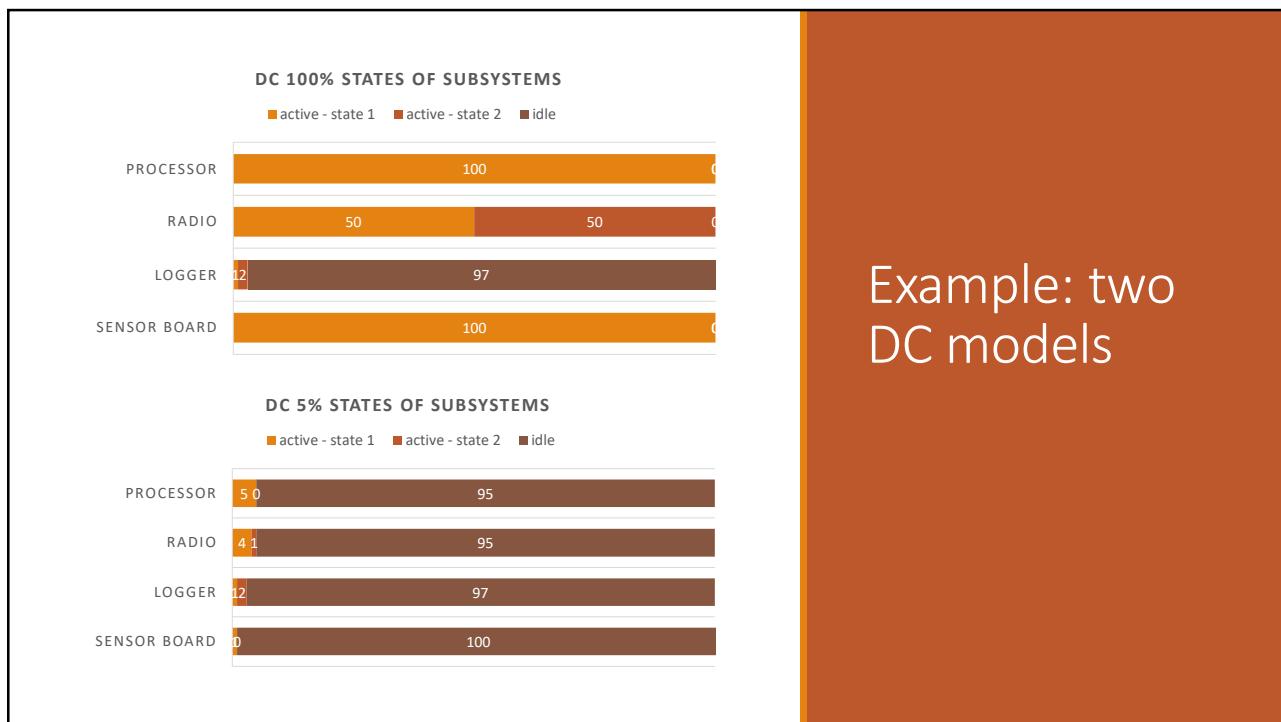
Some components work in parallel

Example: two DC models

Logger duty cycle of 3%

26

13



Example: two DC models

27

Measuring energy – a note

Energy and power are measured in Joule (J) and Watt (W), respectively:

$$1J = 1W \cdot sec$$

In electromagnetism, $1W$ is the work performed when a current of $1Ampere$ ($1A$) flows through an electrical potential difference of $1Volt$ ($1V$):

$$1W = 1V \cdot 1A \quad \Rightarrow \text{in Direct current voltage is constant}$$

Since we use direct current and the electrical potential difference is (almost) constant, the power and the energy only depend on the current (Ampere).

Hence we can «express» both the energy stored in a battery (battery charge) and the energy consumed in mAh .

\downarrow Direct current \downarrow mAh \downarrow Current \downarrow \Rightarrow And to V one ~~one~~ constant Ampere minutes

28

14

Fraction of time active:

400 ms

Duty cycle

11
sample period (not busy)

- Energy cost microprocessor E_μ (per cycle):

$$E_\mu = C_\mu^{\text{full}} \cdot dc_\mu + C_\mu^{\text{idle}} \cdot (1 - dc_\mu)$$

• C_μ^{full} full energy cost microprocessor

• C_μ^{idle} idle energy cost microprocessor

• dc_μ % duty cycle microprocessor

idle account:
Active + Inactive
in Vtage per ms

- Energy cost radio E_p (per cycle):

$$E_p = C_p^T \cdot dc_p^T + C_p^R \cdot dc_p^R + C_p^{\text{idle}} \cdot (1 - dc_p^T - dc_p^R)$$

every job
in more than
transmit

• C_p^T radio transmission energy cost

• C_p^R radio receival energy cost

• C_p^{idle} idle energy cost

energy in idle cost 3 order magnitude less

• dc_p^T % transmit duty cycle radio

• dc_p^R % receive duty cycle radio

Example: computing consumption per duty cycle

29

$$E_\mu = 8 \text{ mA} \cdot \frac{120}{400} + 15 \mu\text{A} \cdot \left(1 - \frac{120}{400}\right)$$

$$\text{mA} \quad \frac{120}{400} = \frac{1}{3.33}$$

$$\frac{1}{20} : \frac{1}{3.33} = \frac{1}{2000}$$

$$E_p = 20 \text{ mA} \cdot \frac{15}{400} \cdot \frac{1}{100} + 0 +$$

$$+ 20 \mu\text{A} \cdot \left[1 - \frac{15}{400} \cdot \frac{1}{100}\right]$$

- Energy cost logger E_λ and sensor board E_σ (same as before)

single read:

- Total energy cost (per duty cycle):

$$E = E_\mu + E_p + E_\lambda + E_\sigma$$

Σ every
spend

at computer

- Lifetime (in number of duty cycles):

$$\text{Lifetime} = \frac{B_0 - L}{E}$$

initial battery charge
as
is
depends on lifetime
battery cost per cycle
"recycle"

• Where B_0 is the initial battery charge and L is the battery charge lost during the lifetime due to **battery leaks**

• Note that L depends on lifetime!

• Based on the charge loss/cycle ε

charge loss

Example: computing lifetime

30

$$\frac{3}{100} = \frac{300}{\text{cycle}}$$

$$\frac{100 - 3}{100} = \frac{97}{100} = 97\%$$

15

Example: computing lifetime

- We thus have a recurrence equation:

$$B_n = B_{n-1} \cdot (1 - \varepsilon) - E$$

at cycle n

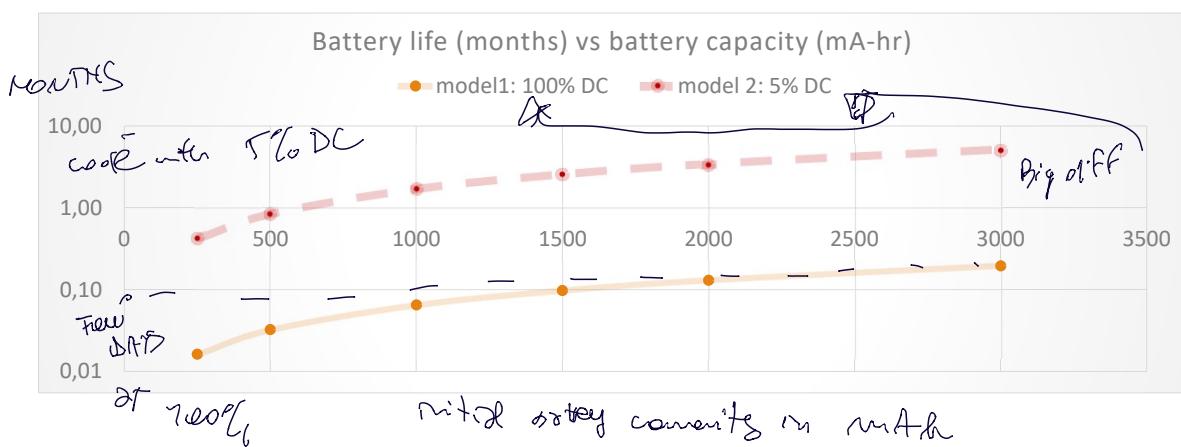
- Where B_n is the battery charge at cycle n

- Solving the recurrence equation:

$$B_n = B_0 \cdot (1 - \varepsilon)^{n-1} + \frac{E((1 - \varepsilon)^n - 1)}{\varepsilon}$$

- Device lifetime is given by $n : B_n = 0$
- ... in practice the device stops working before than that...

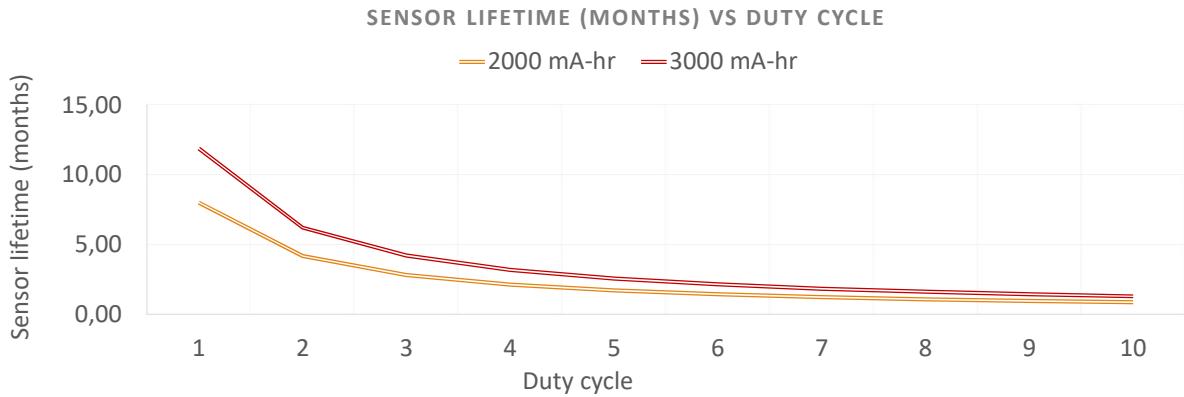
31



Battery life vs DC (I)

32

16



Battery life vs DC (II)

33

Energy efficiency

- Hence the solution is to reduce the DC

However:

Turning off the processor is a local decision

- the node scheduler knows what are the activities and when the processor should run

Turning off the radio is a global decision:

- A device with its radio off does not communicate
- Cannot receive incoming messages/commands
- Cannot act as router in multihop network...
 - ... cannot receive requests, commands, updates...

not always possible, constraint effect perf. of app
 - possible change instructions
 to reduce DC
 - NP: radio to communicate
 with the others, if turn off
 radio, others cannot communicate
 with me
 ✓ use coordination
 with other devle
 C impact on MAC
 other layers

34

17

MAC Protocols

- Low-level communication protocols
 - send/receive packets to/from in-range sensors

In conventional networks, MAC protocols arbitrate the access to the shared communication channel

In IoT they also implement strategies for energy efficiency

- synchronize the devices
- turn off the radio when it is not needed
 - turning off the radio means excluding a device from the network

35

Exercise

Consider this program and the table of energy consumption in the different states. Compute:

- the energy consumption of the device per single hour
- the expected lifetime of the device

```
...
void loop() {
    turnOn(analogSensor);
4 milliseconds int sensorValue = analogRead(A0);
    turnOff(analogSensor);

1 milliseconds float voltage = sensorValue*
    (5.0 / 1023.0);

        turnOn(radioInterface);
15 milliseconds Serial.println(voltage);
    turnOff(radioInterface);

380 milliseconds idle(380);
}
```

	value	units
Micro Processor (Atmega128L)		
current (full operation)	8	mA
current sleep	15	µA
Radio		
current xmit	1	mA
current sleep	20	µA
Sensor Board		
current (full operation)	5	mA
current sleep	5	µA
Battery Specifications		
Capacity	2000	mAh

36

18

solution

energy consumption per hour:

- The processor has a duty cycle of _____
- The radio has a duty cycle of _____
- The sensor has a duty cycle of _____

Energy consumption in one hour: _____

Lifetime of the device: _____

	idle	Active
Processor		
Radio		
Sensor		
Total:		

	value	units
Micro Processor (Atmega128L)		
current (full operation)	8	mA
current sleep	15	µA
Radio		
current xmit	1	mA
current sleep	20	µA
Sensor Board		
current (full operation)	5	mA
current sleep	5	µA
Battery Specifications		
Capacity	2000	mAh

37

Summary

- Issues in IoT design
- Energy efficiency
- Duty Cycle in IoT (different from laptop \Rightarrow general purpose)
- Energy consumption of an IoT device
- Lifetime of an IoT device
 - \hookrightarrow based on DCs tolerate to different environments
 - \hookrightarrow esp. of A services

40



19