# 9. Energy efficiency

## Saving the energy, Edge computing, Cloud computing for WSN

Wireless sensor networks

Martin Úbl
ublm@kiv.zcu.cz

2023/24

<DCSE> DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# Energy efficiency

- A typical WSN node is powered by battery
- what type of battery to use?
  - what is the operating voltage and average current drain?
  - what temperatures it must endure?
  - should it be rechargeable? (e.g., from solar panel)
  - what is the desired mission time?
  - can we perform a maintenance?

# Energy efficiency

## Battery capacity

- for us, a battery capacity is an important parameter
- usually given in *mAh*
- how long the battery will last?
  - $t_{max}[h] = \frac{C[mAh]}{A[mA]}$
  - where $C$ denotes capacity and $A$ the average current draw
- for example:
  - battery capacity $C = 900mAh$ at its nominal voltage (let's say $3.7V$)
  - average current draw $A = 45mA$
  - the battery will last $t_{max} = \frac{900}{45} = 20$ hours
- this calculation gives us a very rough estimate

# Energy efficiency

## Battery capacity

- the other way around: if we want the node to last at least one year on the battery
- for example:
  - battery capacity $C = 1500mAh$
  - desired mission time $t_{max} > 1$ year ($t_{max} > 8765$ hours)
  - what is the maximum average current draw?
  - $8765[h] < \frac{1500[mAh]}{A}$
  - $A < \frac{1500}{8765}$
  - $A < 0.1711[mA]$

# Energy efficiency

- how can we achieve such energy consumption?
- for example
  - active time consumes $A_a = 35$ mA
  - sleep time consumes $A_s = 0.015$ mA ($15\mu A$)
  - we need $A_a * t + A_s * (1 - t)$ to equal the boundary $A$ value
  - $35 * t + 0.015 * (1 - t) = 0.1711$
  - by reorganizing and factoring out $t$, we obtain: $t \doteq 0.0045$
  - this means, that we can spend less than 0.45% in active time
  - practically, if the lowest possible active time is 5 seconds, we would need to sleep for 18 minutes
- equation for $t$ based on known consumptions
  - $t = \frac{A - A_s}{A_a - A_s}$

# Energy efficiency

- can we achieve such battery life?
- of course
  - lots of applications report theoretical battery life of more than 10 years
- let us assume, that we have correctly designed hardware part
- we will focus on software control
- goal is to:
  - minimize active mode current draw
  - minimize sleep mode current draw
  - minimize active time
  - maximize sleep time

# Energy efficiency

## Minimizing current draw

- *minimize active mode current draw*
- the first thing to consider
- sources of largest power consumption:
  - radio transmitter
  - radio receiver
  - CPU core operation
  - peripherals
  - clock and support electronics (loss on buck converters, ...)

# Energy efficiency

## Current draw

- example: ESP32-WROOM-32E
- https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf
  - page 15 for radio RX/TX
- WiFi transmit: average 165 mA (802.11n @ 13dBm), 239 mA (802.11b)
- WiFi receive: average 118 mA (802.11n)
- single active CPU core operation: average 20-35 mA
- peripherals: up to 40 mA more in average
- that's way too much!
  - fortunately, there is much we can do to lower these values
  - these are average current draws at a specific "average" setting

# Energy efficiency

## Current draw

- example: CC1101
- sub-Ghz transceiver
- `https://www.ti.com/lit/ds/symlink/cc1101.pdf?ts=1703853203301`
- radio transmit: average 35 mA (maximum TX power)
- radio receive: average 15 mA
- single active CPU core operation: average 8 mA
- peripherals: up to 10 mA more in average
- that's way better, but we must use the sub-Ghz radio instead of well-established WiFi
  - to communicate with Internet, we need dual-technology gateway
- one remark: CC1101 can sleep on less than 0.2 $\mu A$!

# Energy efficiency

## Current draw

- *lowering radio consumption*
- first thing to try: lower overal radio consumption
- use radio standard that saves more energy
  - WiFi standard 802.11n and newer
  - Bluetooth with Bluetooth Low Energy support
  - sub-Ghz radio (power consumptions often lower than 10 mA for RX/TX)
- lower the TX power
  - balance the radio reach with multi-hop routing
- lower the data rate
  - there's not much of a difference when transmitting a few bytes
  - we don't need 50 Mbps WiFi rate, a few kbps suffices

# Energy efficiency

## Current draw

- depending on hardware, we may control a few extra things
- preamble length
- shift keying (OOK draws less than 64-QAM)
- forward error checking (FEC)
- use **low-power listening** (LPL) when possible
    - RX in LPL may drop to e.g., less than 1 mA

# Energy efficiency

## Current draw

- *CPU core operation current draw*
- we may reduce CPU consumption by very limited means
- reduce computational parts
  - simpler algorithms
  - better optimized algorithms
  - push calculation to edge or cloud nodes
- use lower core frequency
- utilize efficient hardware parts (makes a little difference)
  - defer calculation to specialized HW parts
    - e.g., encryption/decryption/hashing to the AES coprocessor if available, etc.
  - lower memory usage to use CPU cache more efficiently

# Energy efficiency

## Current draw

- *peripherals current draw*
- use only peripherals that are required
  - some needs to be explicitly shut down
- power on peripherals only before actual use
- power off peripherals right after they fulfilled their purpose
- but...
  - some peripherals may require lengthy initialization phase, stabilization
  - e.g., when we need GPS coordinates every minute, there is no point in switching GPS off, because finding satellites may take up to a minute
    - peripherals like GPS often support a reduced power mode (low power mode)

# Energy efficiency

## Current draw

- *sleep mode*
- every IoT/WSN MCU have sleep modes
- different sleep modes for different purposes
- example: ESP32-WROOM-32E
  - *modem-sleep* – up to 40 mA
    - radio chip is powered down
  - *light sleep* – up to 1 mA
    - CPU is paused (clock disconnected)
  - *deep sleep* – 10, 100 or 150 $\mu A$
    - only RTC and RTC memory is powered
    - Ultra-Low Power coprocessor (ULP) is running
  - *hibernation* – 5 $\mu A$
    - only RTC memory and RTC is running on slow clock
  - *power off* – 1 $\mu A$
    - everything powered off
    - can be woken up only via RST

# Energy efficiency

## Current draw

- *sleep modes*
- we may combine the sleep modes as we wish
- important things to consider:
  - recovery time from sleep mode
  - in some sleep modes, RAM is powered down (loses contents)
  - TDMA requires precise wake-up time
    - may not be suitable to use some sleep modes or radio technologies at all

# Energy efficiency

## Active time

- *reducing active time*
- can be done exclusively in software
- choice of protocols
- choice of topologies
- data aggregation
  - e.g., 5 sensors in a room, but an average of all 5 suffices
- data combining
  - e.g., we may measure temperature every minute
  - but transmit the result set every 5 minutes
- reducing errors
  - transmission errors: forward error checking (self-correcting codes)
  - parasitic data: repeated measurements and health-check

# Energy efficiency

## Active time

- *maximizing sleep time*
- if the node can sleep, it should sleep
- the ratio of sleep time to active time should be maximum possible
- depending on application, the ratio could be:
  - $\sim$ 10% on fast changing quantities (e.g., sound, ...)
  - $\sim$ 1% on moderately-fast changing quantities (e.g., ambient light, ...)
  - $\sim 0.1 - 0.01\%$ on moderately-slow changing quantities (e.g., temperature)
  - $\sim$ 0.001% and less on slowly changing quantities (e.g., daily screening of greenhouse $O_2$ concentration)

# Energy efficiency

## Active time

- common scenario: sensor duty cycles are shorter than radio duty cycles
- example: average hourly temperature
- sensor wakes up every 5 minutes for 1 second to perform a measurement every wakeup
- calculates moving average on measured values
- after one hour, it sends the averaged value to the sink node (approx. 5 seconds)
- average sensing active time: 0.3%
  - on reasonably low-power hardware: $\sim 5mA$
- average radio active time: 0.14%

# Energy efficiency

## Reducing power draw

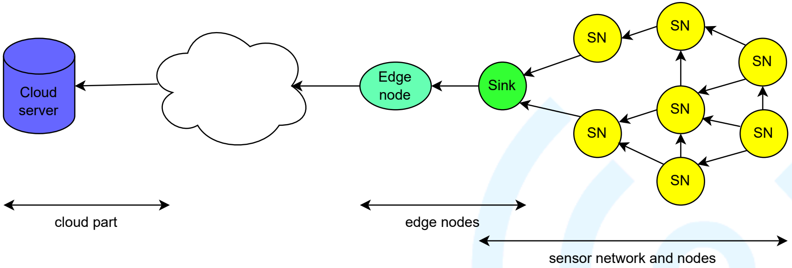
- sometimes, it is more convenient to employ additional hardware
  - sub-Ghz radio with ultra low-power operation
  - AES coprocessor
  - image processing chip (for e.g., object detection use-cases)
  - custom FPGA-based processing unit (rare cases)

# Edge and Cloud

- some tasks are inconvenient for sensor nodes
- **edge** and **cloud computing**
- defer calculations to other nodes with "unlimited" power supply

# Edge and Cloud

- common scenario for WSN-Edge-Cloud network



# Edge and Cloud

## Edge computing

- **edge computing**
- business-oriented definition: *A distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers.*
- simply put:
  - nodes with unlimited power, but limited computational resources
    - not as limited, as sensor nodes, but still more than full-blown servers
  - sits right at the exit of sensor network (or IoT network, ...)
  - allows:
    - data manipulation at the edge of the network
    - network control
    - lower the data and analysis delivery delay

# Edge and Cloud

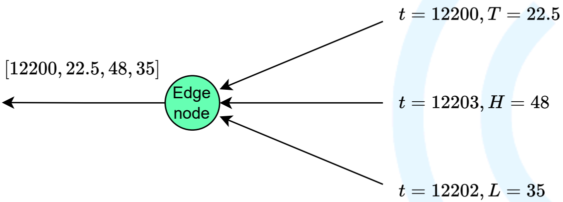
## Edge computing

- typical roles of edge nodes:
  - *data pairing and aggregation*
  - *pre-processing*
  - *filtering*
  - *WSN health-check*
  - *node monitoring*
  - *network querying*
  - *network control*

# Edge and Cloud

## Edge computing

- *data pairing and aggregation*
- sensor nodes measure and transmit independent data
- e.g., temperature data, humidity, ambient light, ...
- edge nodes may "pair" these data to form tuples
- server receives the paired tuples



# Edge and Cloud

## Edge computing

- *data pre-processing and filtering*
- if the WSN is large, we obtain fairly high amount of data
- may be as high, as 1000 values per minute
  - depends on application
- edge node may calculate some intermediate data
- e.g., temperature field from temperature data

# Edge and Cloud

## Edge computing

- *data filtering*
- when having a complete data set, it is easier to detect outliers
- some sensor nodes do not detect, if the measurement is correct
  - some even don’t have means of doing so
- may be a momentary faulty measurement
  - edge node drops it
- may be a long-lasting problem
  - edge node must monitor it

# Edge and Cloud

## Edge computing

- edge node categorization
- as usual, every business and tech section have their own categorization and definitions using various buzzwords
- one layerization:
  - *tiny edge* – closest to WSN
  - *far edge* – middle layer, servers and local processing
  - *near edge* – upper layer, infrastructure and data storage
- another one:
  - *consumer/deep edge* – closest to WSN, includes e.g., mobile phones
  - *edge layer* – middle layer
  - *fog layer* – upper layer, infrastructure, local servers, data storage
- let's not fall for buzzwords and try to understand the principles

# Edge and Cloud

## Edge computing

- from WSN perspective, edge and cloud is a way to analyze, process, store data
- edge offers additional computing power at relatively low cost
- cloud offers "unlimited" computing power for a bigger cost
- we can exploit both to save energy
- namely the edge is the most beneficiary for us

# Edge and Cloud

## Edge and Cloud

- from WSN perspective, cloud is "too far away" – WSN does not directly interact with it
- WSN interact with cloud through the edge
- more on edge and cloud computing: KIV/DCE – Distributed Computing Environments

# Edge and Cloud

## Final remarks

- final remarks: edge as query mediator
- sometimes the WSN is driven from external source
- e.g., server decides, when to make measurements and how to store them
- cloud server (instructed by the user through some UI) can send precise instructuons to the edge node
- the edge node then schedules events and "scatter" the instructions
  - either to schedules for nodes
  - or schedules itself and sends commands at the right time
- this is a pretty straightforward way to *active sensor networks*
  - "reprogrammable" sensor networks