INTRODUCTION TO WIRELESS SENSOR NETWORKS

CHAPTER 2: ANATOMY OF A SENSOR NODE

Anna Förster

IEEE PRESS

WILEY

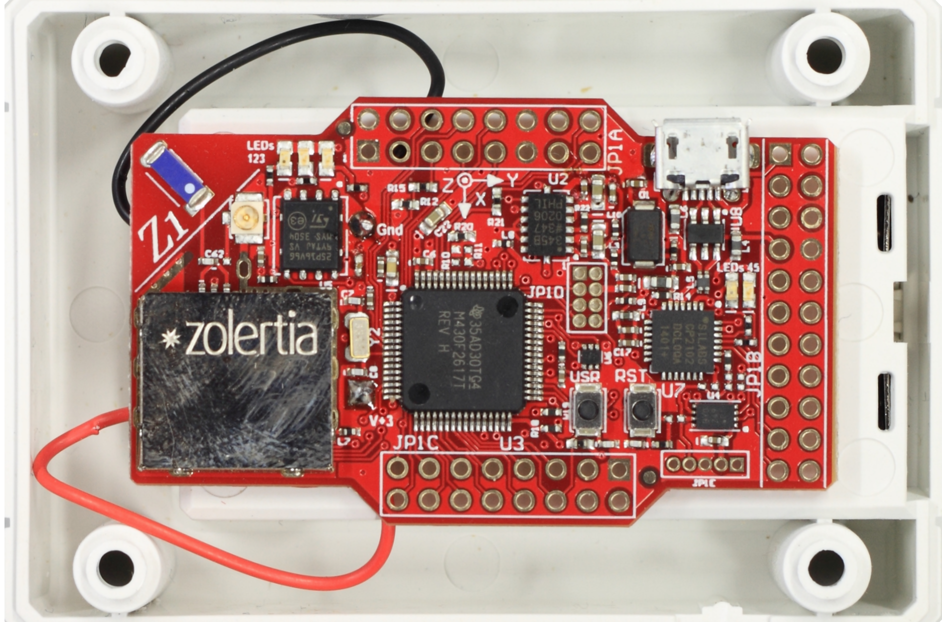
comnets

OVERVIEW

1. Hardware components
2. Power Consumption
3. Operating Systems and Concepts
    1. Memory Management
    2. Interrupts
    3. Tasks, Threads and Events
4. Simulators
5. Communication Stack

HARDWARE COMPONENTS

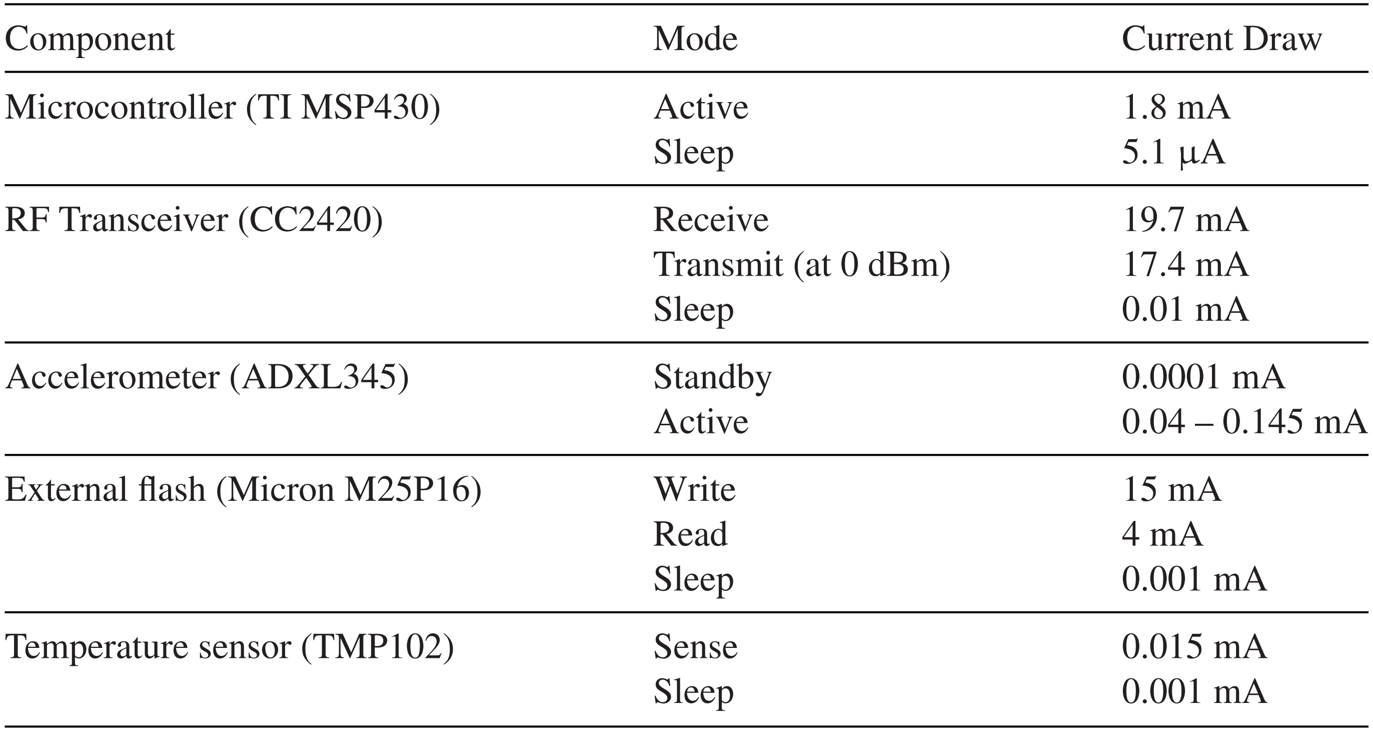
- Microcontroller
- Radio transceiver
- Sensor
- External memory
- Battery
- Serial adapter
- Embedded antenna
- Oscillator



Z1 sensor node.

POWER CONSUMPTION

- Sensor nodes operate on batteries
- Each component on the sensor node consumes power
  - Measured in Ampere (A, current) or milli Ampere (mA).



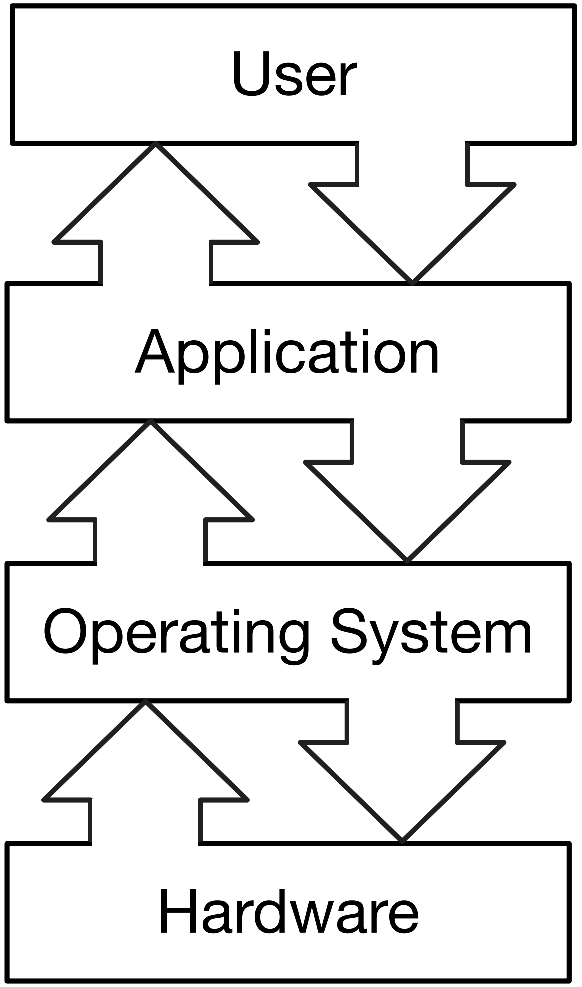
Component	Mode	Current Draw
Microcontroller (TI MSP430)	Active Sleep	1.8 mA 5.1 µA
RF Transceiver (CC2420)	Receive Transmit (at 0 dBm) Sleep	19.7 mA 17.4 mA 0.01 mA
Accelerometer (ADXL345)	Standby Active	0.0001 mA 0.04 – 0.145 mA
External flash (Micron M25P16)	Write Read Sleep	15 mA 4 mA 0.001 mA
Temperature sensor (TMP102)	Sense Sleep	0.015 mA 0.001 mA

Power consumption of Z1 sensor nodes

EXERCISE: BATTERIES

- Explore the batteries in front of you.
- What is their nominal capacity?
- Consider a scenario, where the sensor node is working in cycles. Every cycle, the components are working with the following time percentages:
  - Radio: 10% send, 10% receive, 80% sleep
  - Microcontroller: 10% active, 90% sleep
  - Temperature sensor: 5% sense, 95% sleep
  - The other components are always sleeping /inactive.
- Compute the energy consumption of one cycle with length 1 second.
- Now assume the active time of all components is halved. What is now the energy consumption of 1 cycle of length 1 second?
- For both scenarios above, compute the theoretical lifetime of the sensor nodes, considering the battery capacity in front of you.
- Discuss your results.

CONCEPTS OF OPERATING SYSTEMS



The tasks of the operating system:

- Manage software and hardware
- Allow the user to access easily the hardware
- Allow several programs to run simultaneously
- Manage the memory of the system
- Manage shared resources (memory, CPU, external devices)

Memory Management

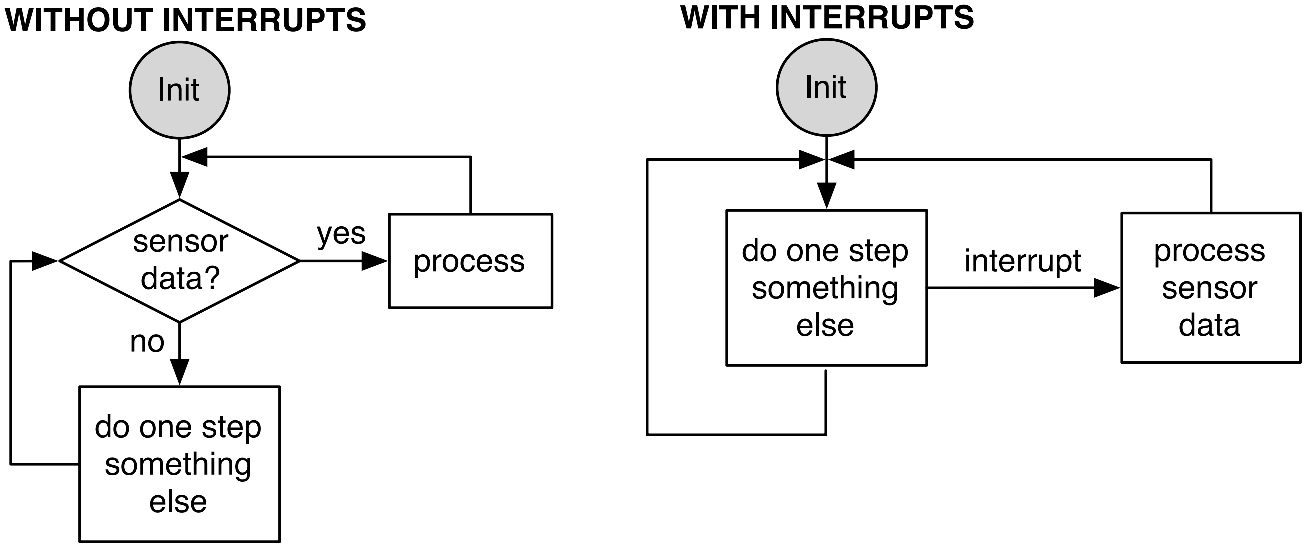
- Static memory management: Reserve all memory at compilation time.
  - Simple, does not need have any overhead for management
  - Inflexible, cannot extend on demand
- Dynamic memory management: A special memory management software reserves memory on demand.
  - Flexible, uses only memory which is really needed
  - Complex to maintain, can easily overflow on memory-restricted devices (like sensor nodes)

EXERCISE: MANAGE MEMORY

- Read Section “Memory Management Example” on page 19.
- Discuss the usage of the Big-O Notation.
- Explain the different implementations and compare against each other.
- Do you have other ideas of how to implement the same functionality?

INTERRUPTS

- **Goal**: react as fast as possible to an external signal (sensor, user, etc.)
- **Example**: ask sensor for data.
- **Problem**: sensor does not answer immediately.



Anna Förster, Introduction to Wireless Sensor Networks, 2016

TASKS

- Interrupts are used to control the flow of programs
- How to enable several programs to run simultaneously?
- **Task**: an independent process in the system with its own resources and memory, which works independently from all other processes
- **Memory management unit**: makes sure memory is allocated to each process and that processes cannot access/corrupt memory of other processes.
- **Multitasking**: the scheduler decides which task is allowed to proceed and which is interrupted.
- **Shared resources**: the process needs access to some shared resource/device, e.g. printer or network interface. This access should not be interrupted.
- **Atomic operations**: runs until completion and nobody can interrupt it.
- DISADVANTAGE: tasks are resource-intensive

EVENT BASED PROGRAMMING

- An alternative to multi-tasking
- Based on **events** as opposed to interrupts: events are not necessarily handled immediately (like interrupts) and can be generated also by software components. For example, a lengthy processing job can create intermediate results and signal those by creating events.
- Each event correspond to one or more **event handlers**, which process the event.
- Event-based programming is based on **finite state machines**.
- **QUESTION**: what is a finite state machine?
- DISADVANTGE: Finite state machines are not trivial to implement and do not scale well.

EXERCISE: PROTOTHREADS

- Inform yourself on the Contiki website about protothreads.
- Implement a small application, which uses protothreads
- Discuss the usage of protothreads in terms of technical efficiency and ease of implementation

SIMULATORS

- A software system, running on a normal computer, which mimics the behavior of some other system and its interactions
- Various tools exist:
  - Cooja for the Contiki operating system
  - OMNeT++ for more sophisticated and general-purpose network simulators
  - Many others

SIMULATION MODELS

Simulation relies on simulation models, which mimic the behavior of individual real components and properties

- **Wireless propagation models**: how does the wireless signal propagates through different environments?
- **Mobility models**: how do the mobile nodes move around?
- **Energy expenditure models**: how much energy is needed for individual components and/or tasks?
- **Traffic model**: how much data is sent and when?

COMMUNICATION STACK

- Reduced sensor network communication stack (as opposed to OSI)

Application	Gather and pre-process sensory data, report data, aggregate and compress data, etc.
Routing	Plan a route from the current node to the final destination, find the next hop, etc.
Link management	Error control of packets, node addressing, link quality evaluation
Medium Access	Plan the access to the wireless medium - listen, send, sleep
Physical	Encode the data to transmit into an electromagnetic wave

PROTOCOLS VS. ALGORITHMS

- **Algorithm**: a general-purpose, parameter-based computation flow.
- **Protocol**: exact implementation of one or more stack layers, parameters are often fixed or a very limited number of options is provided

SUMMARY

- Sensor node consists of microcontroller, radio transceiver, sensors, serial connection, LEDs, flash memory.
- Each component needs energy, with radio and flash being the most “hungry” ones.
- Putting individual components to sleep saves energy.
- Towards the end of battery lifetime, performance of individual components degrades and might deliver false results.
- WSN operating systems are simple and usually have only limited dynamic memory and multi-tasking functionalities.