TinyOS 2.x & nesC
Chapter X
Sensor Nodes

- System Constraints
  - Slow CPU
  - Little memory
  - Short-range radio
  - Battery powered
Operating System Requirements

- Measure real-world phenomena
  - Event-driven architecture

- Resource Contraints
  - Hurry up and sleep!

- Adapt to changing technologies
  - Modularity & re-use

- Applications spread over many small nodes
  - Communication is fundamental

- Inaccessible location, critical operation
  - Robustness
TinyOS

- TinyOS consists of a scheduler & graph of components
Programming Model

- Separate construction and composition
- Programs are built out of components connected by interfaces
- Two types of components:
  - Modules: Implement program logic
  - Configurations: Wire components together
- Components use and provide interfaces

provide „hooks“ for component wiring

Interfaces are bidirectional
Programming Model

- Interfaces contain definitions of
  - Commands
  - Events

- Components implement the events (event handlers) they use and the commands they provide
Programming Model

- Components are **wired** together by connecting interface users with interface providers
- Commands flow downwards
  - Control returns to caller
- Events flow upwards
  - Control returns to signaler
- Commands are non-blocking requests

Modular construction kit
Concurrency Model

- Coarse-grained concurrency only
  - Implemented via tasks

- Tasks are executed sequentially by the TinyOS scheduler
  - "Multi-threading" is done by the programmer
  - Atomic with respect to other tasks (single threaded)
  - Longer background processing jobs

- Events (interrupts)
  - Time critical
  - Preempt tasks
  - Short duration (hand off computation to tasks if necessary)

watch out for data races
Memory Model

- Static memory allocation
  - No heap (malloc)
  - No function pointers

- Global variables
  - One namespace per component

- Local variables
  - Declared within a function
  - Saved on the stack

- Conserve memory
- Use pointers, don’t copy buffers
TinyOS Distribution

• TinyOS is distributed in source code  
  – nesC as programming language

• Nested C (nesC)  
  – Dialect of C  
  – Embodies the structural concepts and execution model of TinyOS  
    – Module, configuration, interface  
    – Tasks, calls, signals  
  – Pre-processor produces native C code

• nesC limitations  
  – No dynamic memory allocation  
  – No function pointers
configuration BlinkAppC{
}

implementation {
    components MainC, BlinkC, LedsC;
    components new TimerMilliC() as Timer0;

    BlinkC -> MainC.Boot;

    BlinkC.BlTimer -> Timer0;
    BlinkC.Leds -> LedsC;
}

module BlinkC {
    uses interface Timer<TMilli>
        as BlTimer;
    uses interface Leds;
    uses interface Boot;
}

implementation{
    event void Boot.booted() {
        call BlTimer.startPeriodic(1000);
    }
    event void BlTimer.fired() {
        call Leds.led0Toggle();
    }
}
The End

Thanks to Pascal von Rickenbach for many of the slides