Programming Sensor Networks

Distributed Computing Group

Nicolas Burri

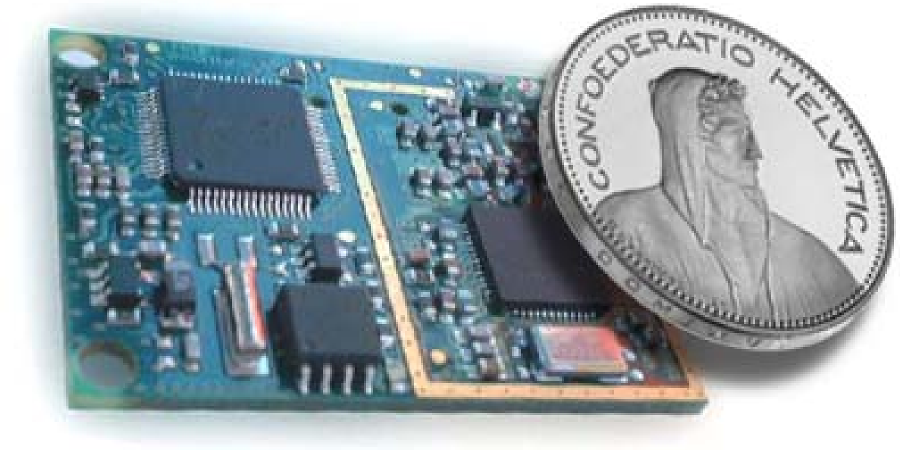
Pascal von Rickenbach

Overview

- TinyOS Platform
- Program Development
- Current Projects

Sensor Nodes

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

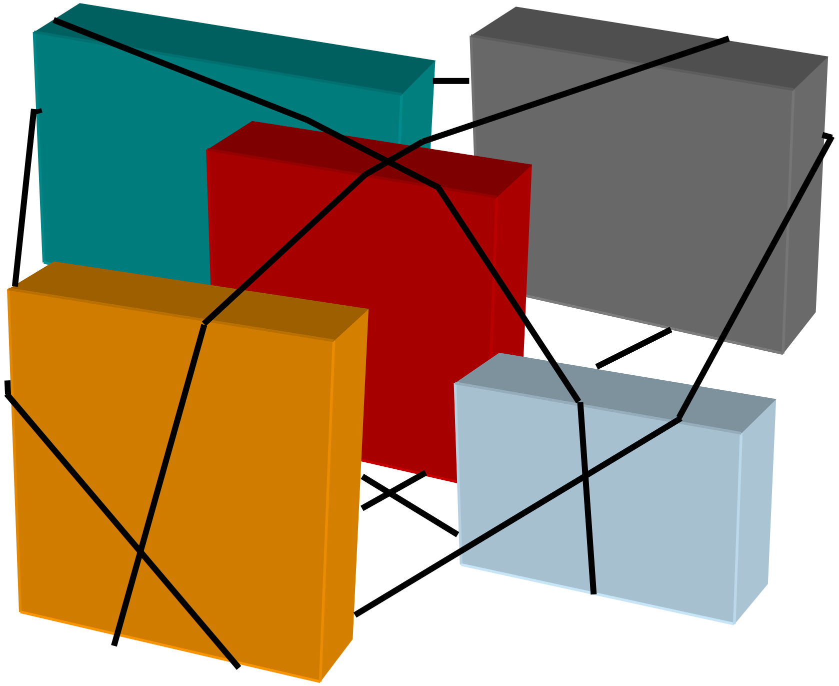


Operating System Requirements

- Measure real-world phenomena
  - Event-driven architecture
- Resource constraints
  - 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 Platform

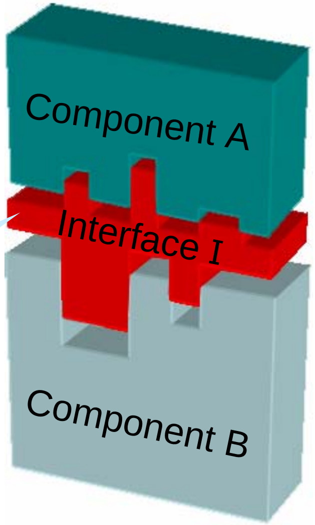
- TinyOS consists of a scheduler & graph of components



Programming Model

provides „hooks“ for component wiring

- Separate construction and composition
- Programs are built out of components specified by an interface
- Two types of components
  - Modules: Implement behavior
  - Configurations: Wire components together
- Components use and provide interfaces



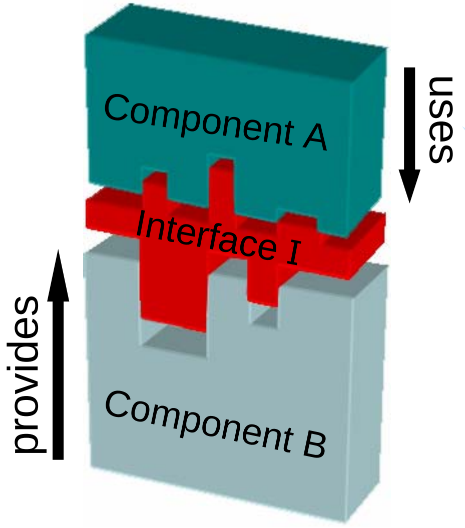
Interfaces are bidirectional

Programming Model

- Interfaces contain definitions of
  - Commands
  - Events

- Components implement the events they use and the commands they provide.

must implement commands, can signal events

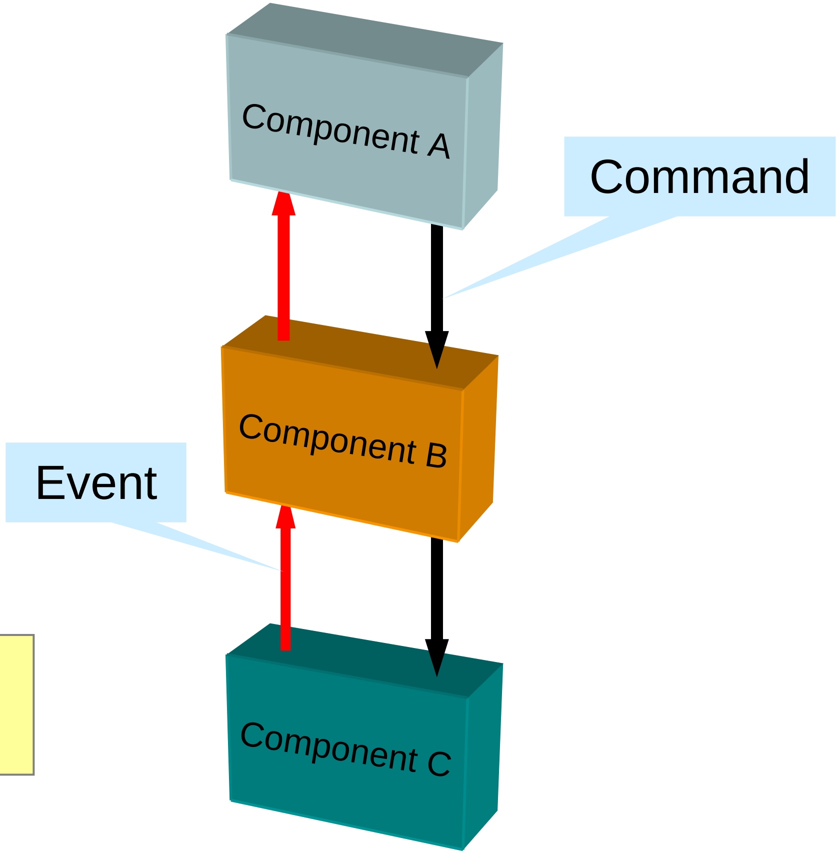


can call commands, must implement events

Programming Model

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

Modular construction kit



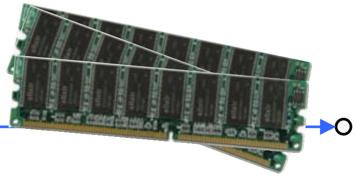
Concurrency Model

Actually single threaded!

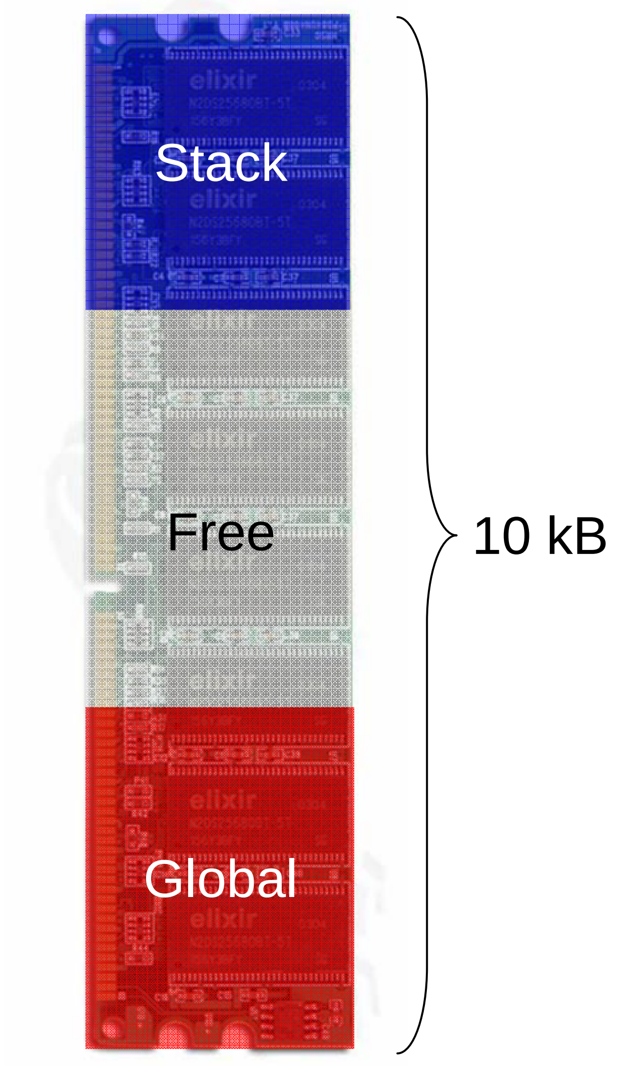
- Coarse-grained concurrency only
  - Implemented via tasks
- Tasks run sequentially by TinyOS scheduler
  - “Multi-threading” is done by the programmer
  - Atomic with respect to other tasks (single threaded)
  - Longer background processing jobs
- Events (interrupts)
  - Time critical
  - Preempt tasks
  - Short duration (hand off computation to tasks if needed)

Note that “event” is overloaded

Memory Model



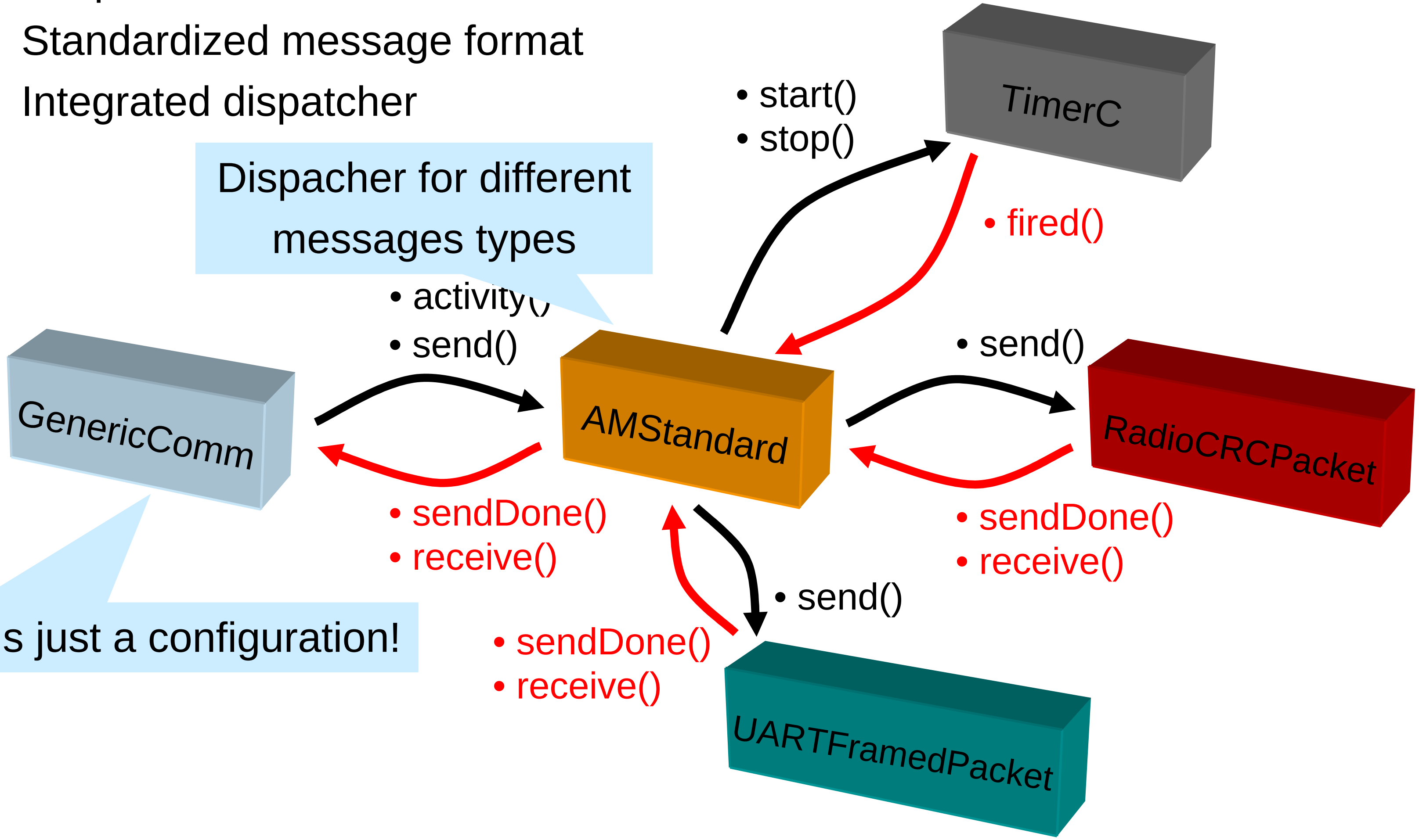
- Static memory allocation
  - No heap (malloc)
  - No function pointers
- Global variables
  - One frame per component
- Local variables
  - Declared within a method
  - Saved on the stack



> - Conserve memory
> - Use pointers, don’t copy buffers

Network Stack

- Ready-to-use communication framework
  - Simple hardware abstraction
  - Standardized message format
  - Integrated dispatcher



Dispacher for different messages types

This is just a configuration!

TinyOS Distribution

- TinyOS is distributed in source code
  - – nesC as programming language
- nesC
  - – Dialect of C
  - – Embodies the structuring concepts and execution model of TinyOS
    - Module, configuration, interface
    - Tasks, calls, signals
  - – Pre-processor producing C code
- nesC limitations
  - – No dynamic memory allocation
  - – No function pointers

nesC – Hello World

All involved components

```
configuration Blink {
}
implementation {
  components Main,BlinkM,TimerC,LedsC;

  Main.StdControl -> BlinkM.StdControl;
  Main.StdControl -> TimerC;

  BlinkM.Timer -> TimerC;
  BlinkM.Leds -> LedsC;
}
```

Wiring the components

```
module BlinkM {
  provides {
    interface StdControl;
  }
  uses {
    interface Timer;
    interface Leds;
  }
}
implementation {
  …
  command result_t StdControl.start() {
    return call Timer.start(TIMER_REPEAT, 1000);
  }

  task void processing() {
    call Leds.redToggle();
  }

  event result_t Timer.fired() {
    post processing();
    return SUCCESS;
  }
}
```

Timer fires every second

Schedule the actual computation

TinyOS Development

- Application development on PC

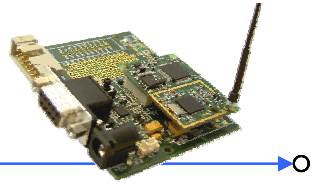
- Programs are compiled to platform specific binaries

- Transfer of binary code using programming boards
  - Serial port
  - Ethernet
  - USB

[mib600 data sheet]

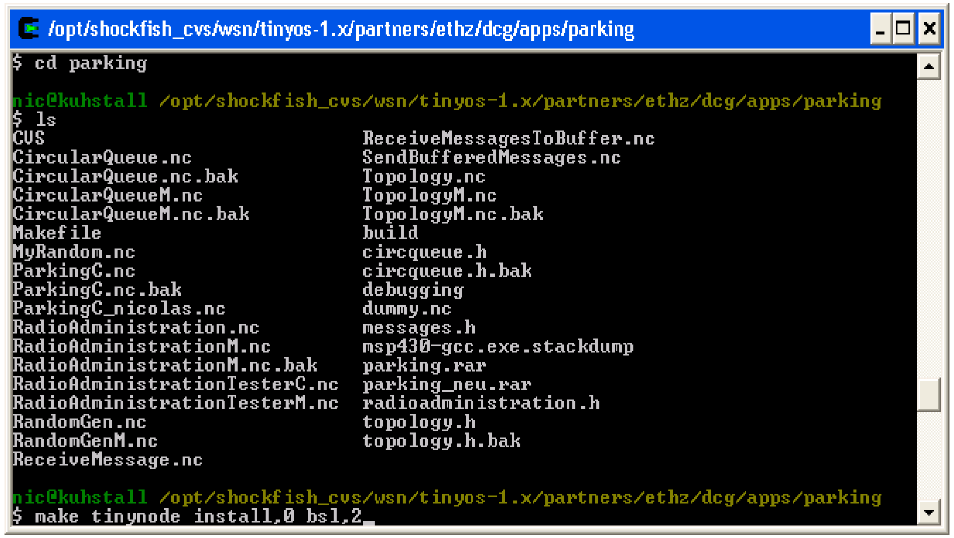
[tinynode manual]

TinyOS Development Today



- **Text Editor**
  - No editor with inbuilt nesC support available
  - Programming in generic text editors
    - UltraEdit
    - Emacs
- **Shell**
  - Make system
    - Compiling of programs
    - Flashing of nodes
  - Additional tools
- **File Browser**
  - Project files
  - Interface definitions
  - System libraries

make tinynode install,0 bsl,2

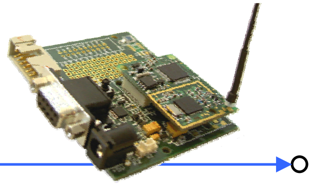


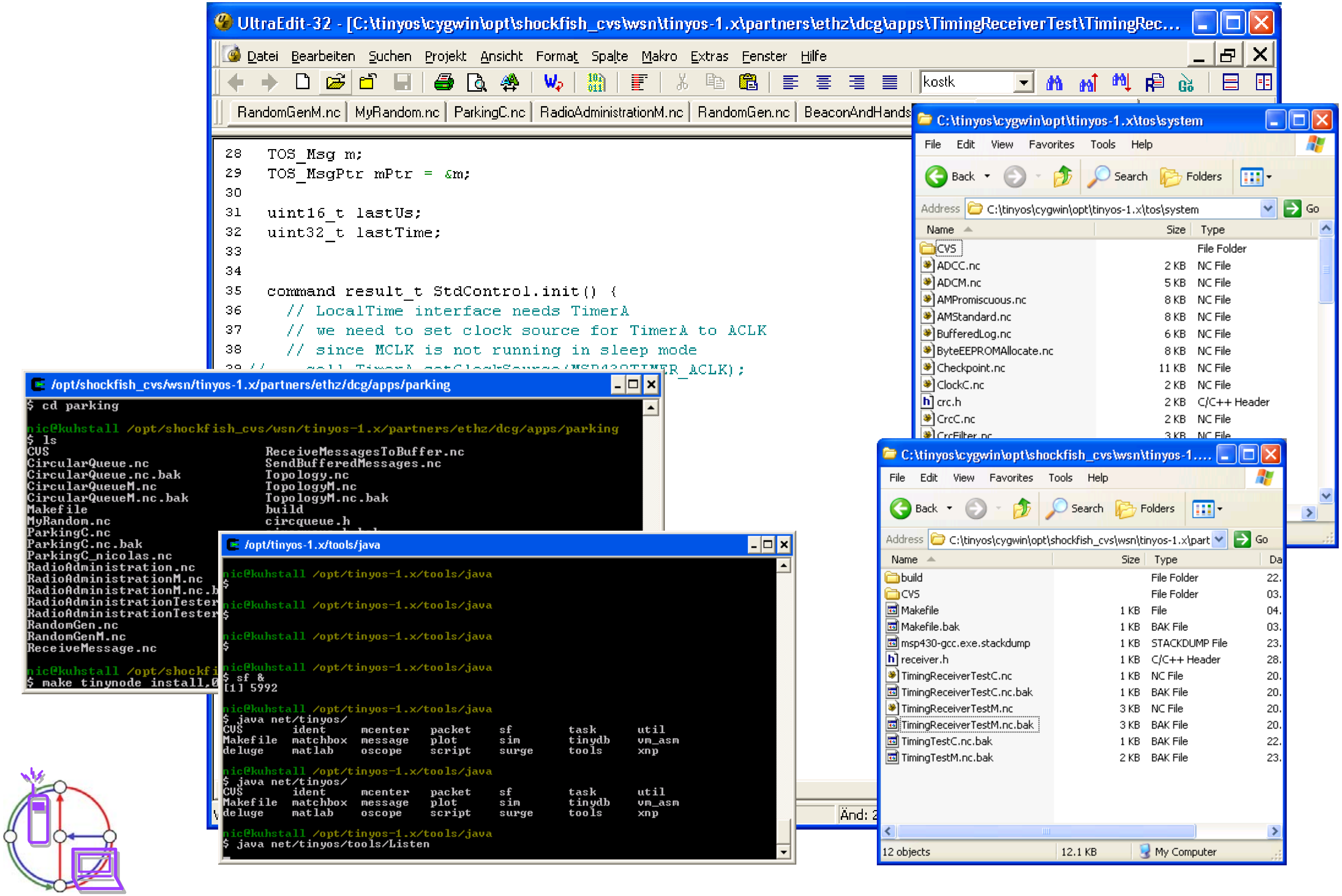
```
$ cd parking

nic@kuhstall /opt/shockfish_cvs/wsn/tinyos-1.x/partners/ethz/dcg/apps/parking
$ ls
CVS                            ReceiveMessagesToBuffer.nc
CircularQueue.nc               SendBufferedMessages.nc
CircularQueue.nc.bak           Topology.nc
CircularQueueM.nc              TopologyM.nc
CircularQueueM.nc.bak          TopologyM.nc.bak
Makefile                       build
MyRandom.nc                    circqueue.h
ParkingC.nc                    circqueue.h.bak
ParkingC.nc.bak                debugging
ParkingC_nicolas.nc            dummy.nc
RadioAdministration.nc         messages.h
RadioAdministrationM.nc        msp430-gcc.exe.stackdump
RadioAdministrationM.nc.bak    parking.rar
RadioAdministrationTesterC.nc  parking_neu.rar
RadioAdministrationTesterM.nc  radioadministration.h
RandomGen.nc                   topology.h
RandomGenM.nc                  topology.h.bak
ReceiveMessage.nc

nic@kuhstall /opt/shockfish_cvs/wsn/tinyos-1.x/partners/ethz/dcg/apps/parking
$ make tinynode install,0 bsl,2
```

TinyOS Development Today





UltraEdit-32 - [C:\tinyos\cygwin\opt\shockfish_cvs\wsn\tinyos-1.x\partners\ethz\dcg\apps\TimingReceiverTest\TimingRec...
Datei Bearbeiten Suchen Projekt Ansicht Format Spalte Makro Extras Fenster Hilfe
kostk
RandomGenM.nc MyRandom.nc ParkingC.nc RadioAdministrationM.nc RandomGen.nc BeaconAndHands

```
TOS_Msg m;
TOS_MsgPtr mPtr = &m;

uint16_t lastUs;
uint32_t lastTime;


command result_t StdControl.init() {
  // LocalTime interface needs TimerA
  // we need to set clock source for TimerA to ACLK
  // since MCLK is not running in sleep mode
//     call TimerA.setClockSource(MSP430TIMER_ACLK);
```

```
$ cd parking

nic@kuhstall /opt/shockfish_cvs/wsn/tinyos-1.x/partners/ethz/dcg/apps/parking
$ ls
CVS                              ReceiveMessagesToBuffer.nc
CircularQueue.nc                 SendBufferedMessages.nc
CircularQueue.nc.bak             Topology.nc
CircularQueueM.nc                TopologyM.nc
CircularQueueM.nc.bak            TopologyM.nc.bak
Makefile                         build
MyRandom.nc                      circqueue.h
ParkingC.nc
ParkingC.nc.bak
ParkingC_nicolas.nc
RadioAdministration.nc
RadioAdministrationM.nc
RadioAdministrationM.nc.b
RadioAdministrationTester
RadioAdministrationTester
RandomGen.nc
RandomGenM.nc
ReceiveMessage.nc

nic@kuhstall /opt/shockfi
$ make tinynode install,0
```

```
nic@kuhstall /opt/tinyos-1.x/tools/java
$

nic@kuhstall /opt/tinyos-1.x/tools/java
$

nic@kuhstall /opt/tinyos-1.x/tools/java
$

nic@kuhstall /opt/tinyos-1.x/tools/java
$ sf &
[1] 5992

nic@kuhstall /opt/tinyos-1.x/tools/java
$ java net/tinyos/
CVS        ident      mcenter    packet     sf         task       util
Makefile   matchbox   message    plot       sim        tinydb     vm_asm
deluge     matlab     oscope     script     surge      tools      xnp

nic@kuhstall /opt/tinyos-1.x/tools/java
$ java net/tinyos/
CVS        ident      mcenter    packet     sf         task       util
Makefile   matchbox   message    plot       sim        tinydb     vm_asm
deluge     matlab     oscope     script     surge      tools      xnp

nic@kuhstall /opt/tinyos-1.x/tools/java
$ java net/tinyos/tools/Listen
```

C:\tinyos\cygwin\opt\tinyos-1.x\tos\system
File Edit View Favorites Tools Help
Back
Search
Folders
Address
Go

Address: C:\tinyos\cygwin\opt\tinyos-1.x\tos\system

Name	Size	Type
CVS		File Folder
ADCC.nc	2 KB	NC File
ADCM.nc	5 KB	NC File
AMPromiscuous.nc	8 KB	NC File
AMStandard.nc	8 KB	NC File
BufferedLog.nc	6 KB	NC File
ByteEEPROMAllocate.nc	8 KB	NC File
Checkpoint.nc	11 KB	NC File
ClockC.nc	2 KB	NC File
crc.h	2 KB	C/C++ Header
CrcC.nc	2 KB	NC File
CrcFilter.nc	3 KB	NC File

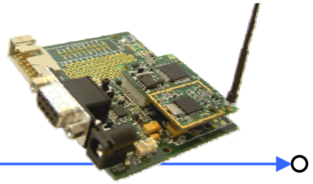
C:\tinyos\cygwin\opt\shockfish_cvs\wsn\tinyos-1....
File Edit View Favorites Tools Help
Back
Search
Folders
Address
Go

Address: C:\tinyos\cygwin\opt\shockfish_cvs\wsn\tinyos-1.x\part

Name	Size	Type	Da
build		File Folder	22.
CVS		File Folder	03.
Makefile	1 KB	File	04.
Makefile.bak	1 KB	BAK File	03.
msp430-gcc.exe.stackdump	1 KB	STACKDUMP File	23.
receiver.h	1 KB	C/C++ Header	28.
TimingReceiverTestC.nc	1 KB	NC File	20.
TimingReceiverTestC.nc.bak	1 KB	BAK File	20.
TimingReceiverTestM.nc	3 KB	NC File	20.
TimingReceiverTestM.nc.bak	3 KB	BAK File	20.
TimingTestC.nc.bak	1 KB	BAK File	22.
TimingTestM.nc.bak	2 KB	BAK File	23.

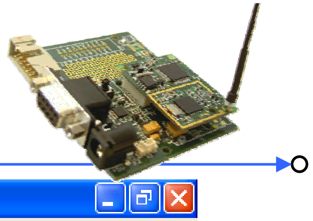
12 objects 12.1 KB My Computer

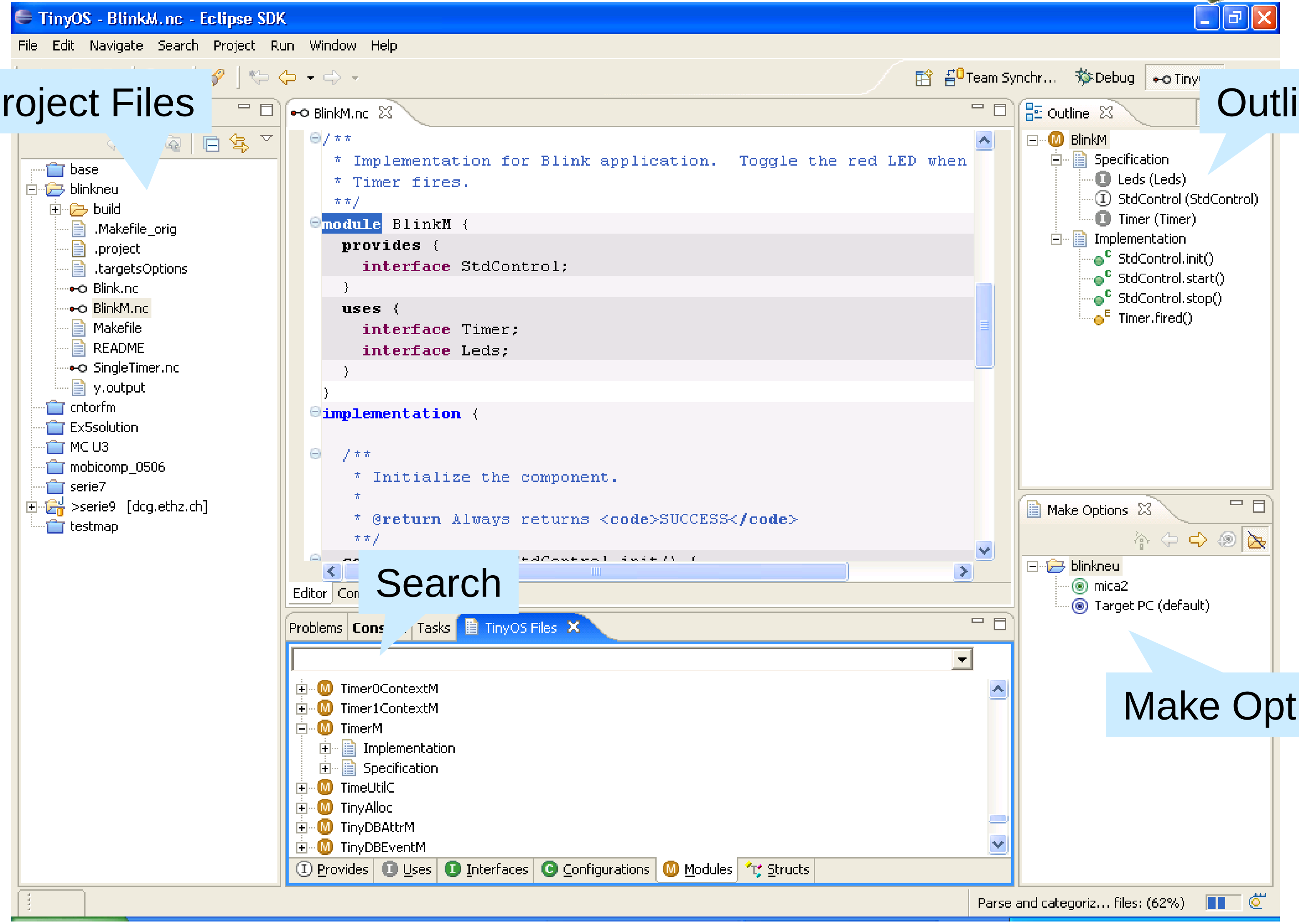
What needs to be improved



- Getting started
  - Setting up the environment is tricky
  - Frustrating without the help of an expert
- Syntax check before compiling
  - Compiling takes up to 1 min even for small programs
- Better debugging support
  - Only three LEDs to show the current state of the application
- Reference
  - What interfaces exist?
  - Which module implements this interface?

TinyOS Plugin for Eclipse





```
/**
 * Implementation for Blink application.  Toggle the red LED when
 * Timer fires.
 **/
module BlinkM {
  provides {
    interface StdControl;
  }
  uses {
    interface Timer;
    interface Leds;
  }
}
implementation {

  /**
   * Initialize the component.
   *
   * @return Always returns <code>SUCCESS</code>
   **/
```

Project Files

Outline

Search

Make Options