Lab 1

• Goal 1: Getting familiar with
  – Cygwin: a linux emulator for Windows
  – TinyOS: an operating system for motes (sensor nodes)
  – NesC: a programming language that extends the syntax of C with components, interfaces, tasks or events
• Goal 2: Compiling a first application
• Goal 3: Running a first application
• These goals will be achieved through a simple application
A simple application

• Problem:
  − Create an application called Blink that makes the red LED of a mote toggle every second

• Solution:
  − Raise a timer every second
  − Each time the timer expires, we need to toggle the red LED
Architecture of Blink

Makefile

BlinkC

Leds

LedsC

BlinkM

Timer

TimerC

StdControl

Main
Makefile

COMPONENT = BlinkC

include /opt/tinyos-1.x/apps/Makerules
configuration BlinkC {
}

implementation {
    components BlinkM, LedsC, TimerC, Main;

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

    BlinkM.Leds -> LedsC;
    BlinkM.BlinkTimer ->
        TimerC.Timer[unique("Timer")];
}

BlinkC.nc
module BlinkM { 
    provides { 
        interface StdControl;
    }
    uses { 
        interface Leds;
        interface Timer as BlinkTimer;
    }
}

implementation { 
    ... // see next slide
}
implementation {
  task void blinkTask() {
    call Leds.redToggle();
  }
  command result_t StdControl.init() {
    call Leds.init();
    return SUCCESS;
  }
  command result_t StdControl.start() {
    call BlinkTimer.start(TIMER_REPEAT, 1024);
    return SUCCESS;
  }
  command result_t StdControl.stop() {
    return SUCCESS;
  }
  event result_t BlinkTimer.fired() {
    post blinkTask();
    return SUCCESS;
  }
}
Blink application (1/2)

• Create a directory for Blink in your home directory (say muc/Blink)

• With a text editor (such as TextPad)
  - create a Makefile (and write the code)
  - create the BlinkM.nc file (and write the code)
  - create the BlinkC.nc file (and write the code)
  - if those files have the .txt extension, it has to be removed (using the Windows rename command)
Blink application (2/2)

- Run Cygwin
  - cd muc
  - cd Blink
  - make pc (to compile Blink)
  - cd build
  - cd pc
  - export DBG=led (to filter the output)
  - main.exe 1 | more (to run the program with 1 mote only)
Explanation: compilation (1/3)

• **The Makefile contains**
  - COMPONENT = BlinkC
  - include /opt/tinyos-1.x/apps/Makerules

• **Description**
  - the Makefile is used by make
  - it tells us that the configuration file is called BlinkC.nc
Explanation: compilation (2/3)

- The BlinkC configuration file contains components BlinkM, Main, LedsC, TimerC;

- Description
  - BlinkM can be found in the current directory
  - the others are basic TinyOS components
    - /opt/tinyos-1.x/tos/platform/pc/Main.nc
    - /opt/tinyos-1.x/tos/platform/pc/LedsC.nc
    - /opt/tinyos-1.x/tos/platform/pc/TimerC.nc
The BlinkM module file contains

- provides { interface StdControl; }
- uses { interface Leds; interface Timer; }

Description

- all the interfaces can be found in
  - /opt/tinyos-1.x/tos/interfaces/
  - simply add .nc to the name of the interface to find the file
Explanation: execution (1/2)

• The program is compiled in
  - muc/Blink/build/pc/main.exe (from your home directory)

• Syntax
  - main.exe -h (for the help)
  - main.exe <number-of-nodes>

• But
  - running main.exe generates too many debug messages
Explanation: execution (2/2)

• The debug messages can be filtered using
  - export DBG=led

• Filters
  - the list is displayed with main -h
  - usually
    • led (for the LEDs)
    • am, radio (for the messages or the radio)
    • task (for the tasks)
    • usr1, usr2, usr3 (for the user debugging messages)