Introduction to Programming Motes

Mohamed M. El Wakil
http://mohamed.elwakil.net
mohamed.elwakil@wmich.edu

Wireless Sensornets (WiSe) Laboratory
Department of Computer Science
Western Michigan University

Agenda

• Wireless Sensor Nodes
  – Components, Models
• Wireless Sensor Networks
  – Applications
• Handling a Sensor Node
  – Antistatic strap, batteries, sensor boards, programming boards
• TinyOS
• NesC
  – Application 1: Blink Application
  – Syntax and semantics
  – Compilation and burning
  – TOSSIM
• XBow MoteWorks
  – XBow Programmers Notepad 2
  – Application 2: Sensing, and wire transmission application
  – XBow MoteView
  – Application 3: Sensing, and wireless transmission application
  – TOSBase

A wireless sensor anatomy

• Small, but powerful computing devices.
• Processor, storage, wireless communication, and sensors.

- Processor:
  - Atmel 8MHz ATmega128L
- Transceiver:
  - IEEE 802.15.4
- Program Memory:
  - 128K bytes
- Measurements Flash
  - 512K bytes
- Power:
  - 2 AA batteries
- Sensors?

MTS310 sensor board

- 4.6KHz Speaker
- Interface
- Tone Detector
- Light and Temperature
- Microphone
- 2 Axis Accelerometer
- Magnetometer

Mica2Dot (~2001)

- Processor:
  - Atmel 16MHz ATmega128L
- Transceiver:
  - ISM Band with up to 38.4 Kbaud
- Program Memory:
  - 128K bytes
- Measurements Flash
  - 512K bytes
- Sensors:
  - ?
**Mica2Dot (~2001)**

- **Processor:**
  - Atmel 16MHz ATmega128L
- **Transceiver:**
  - ISM Band with up to 38.4 Kbaud
- **Program Memory:**
  - 128K bytes
- **Measurements Flash**
  - 512K bytes
- **Sensors:**
  - MTS510
  - Light, sound, and accelerometer.

**Other Devices**

- **Mica2 (~2002)**
  - 8MHz ATMega128
  - 4kB RAM, 128kB flash
  - 512kB external flash
  - 20kb/s 900MHz radio
  - 2 AA batteries

- **Tmote Sky (2005)**
  - 1MHz TI MSP430
  - 10kB RAM, 48kB flash
  - 512kB external flash
  - 250kb/s 802.15.4 radio
  - Built-in sensors

**MicaZ: R&D Vs. OEM**

- 0.95 x 0.95 in
- 24.13 x 24.13 mm
- 2.25 x 1.25 x 0.25 in
- 58 x 32 x 7 mm
What can we do with them?

- Use them as to form a network of wireless sensor devices.
- Fine, but, what can we do with a network of wireless sensor devices?

Enemy Movement Tracking

Equipment Monitoring
Volcano Monitoring

Lots of things!

- Borders monitoring
- Buildings automation
- Enemy movement tracking
- Environment Observation
  - North pole, Great Duck Island
- Radioactive, and poisonous materials detection
- Many others!

Wireless Sensor Network
WSN is one of the 10 Emerging Technologies That Will Change the World

Handling Sensor Nodes

• Antistatic wrist strap to avoid electrostatic discharges

Programming Boards (MB510)
Steps

- Installing Batteries
- Connecting the mote to the programming board
- Connecting a sensor board
- USB-2-Serial Converter

TinyOS

- Open source OS for sensor nodes (motes)
- Runs a single application
- Eases hardware access
  - Timers, Radio, Sensors, Storage, ....
- Based on NesC programming language
  - A flavor of C language

Installing TinyOS v1

- Sun JDK 1.4
- JavaX COMM
- Cygwin
- TinyOS, NesC (RPms)
- ~700MB
- Windows Installer:
  - http://www.tinyos.net/windows-1_1_0.html
- Manual Installation:
  - http://www.tinyos.net/tinyos-1.x/doc/install.html
- toscheck
NesC

- The NesC is a pre-processor
- A NesC program is built out of components
- A component behavior is described via interfaces.
- An interface may be provided by a component or used by a component.
- The provided interfaces are intended to represent the functionality that the component provides to its user.
- The used interfaces represent the functionality the component needs to perform its job.

Components

NesC

- Interfaces are bidirectional:
  - They specify a set of functions to be implemented by the interface's provider (commands) and a set to be implemented by the interface's user (events).
  - Component A calls command C that belongs to component B.
  - When C is done, component B triggers event E that belongs to component A.
Example

- Components:
  - TimerC, ClockC, HPLClock

NesC Programs

- Three files:
  - Make file
  - Module file
  - Configuration file
- A configuration file
  - Wires components in the module file and provides a control flow

Modules

```c
module C1 {
  uses interface triangle;
}
module C2 {
  provides interface triangle out;
  uses {
    interface triangle in;
    interface rectangle side;
  }
}
module C3 {
  provides interface triangle;
  provides interface rectangle;
}
```
Configuration 1

configuration config1 { }
implementation {
uses c2, c3;
c2.in -> c3.triangle
c2.side -> c3.rectangle;
}

- User → Provider

Configuration 2

component config2 { 
provides interface triangle t1; }
implementation {
uses c2, c3;
t1 -> c2.out;
c2.in -> c3.triangle;
c3 <- c2.side;
}

- User → Provider
Interface StdControl

- StdControl is a common interface used to initialize and start TinyOS components.
- Every component should provide this interface.

```c
interface StdControl
{
    command result_t init();
    command result_t start();
    command result_t stop();
}
```

- Calling init() of a module must make it call init() on modules it uses.
- Same applies to start and stop

Blink Application

- Causes the red LED on the mote to turn on and off at 1Hz
- Three files:
  - `BlinkM.nc`: the Blink module component and implementation of its interface.
  - `Blink.nc`: The configuration component
  - `Makefile`: Make file

Makefile

- COMPONENT=Blink
- include ../Makerules
**BlinkM.nc**

- module BlinkM
  - {
    - provides {interface StdControl;}
    - uses {interface Timer; interface Leds;}
    - }
  - implementation {
    - command result_t StdControl.init()
      - {call Leds.init(); return SUCCESS;}
    - command result_t StdControl.start()
      - {return call Timer.start(TIMER_REPEAT, 1000);}
    - command result_t StdControl.stop()
      - {return call Timer.stop();}
    - event result_t Timer.fired()
      - {call Leds.redToggle(); return SUCCESS;}
    - }

**Blink.nc**

- configuration Blink {}
- implementation {
  - components Main, BlinkM, TimerC, LedsC;
  - BlinkM.Leds -> LedsC.Leds;
  - BlinkM.Timer -> TimerC.Timer[unique("Timer")];
  - Main.StdControl -> TimerC.StdControl;
  - Main.StdControl -> BlinkM.StdControl;
  - }

**Compilation and Burning**

- **Compilation**
  - make micaz
  - make mica2
- **Burning**
  - make micaz install,0 mib510,com4
  - make micaz reinstall,0 mib510,com4
- **TOSSIM**
  - make pc
  - main --t=sec nodelscount
  - main --t=10 2
MoteWorks

• A complete software development environment for wireless sensor network applications.
• Based on TinyOS
• Includes:
  – Cygwin/NesC
  – Programmers Notepad
  – MoteView
• http://www.xbow.com/Products/productdetails.aspx?sid=154

Programmers Notepad

Sensing, and wire transmission

• A program that samples the light sensor and transmits its reading to the PC via the serial cable
Sensing, and wire transmission

- **init()**
  - Initialize the LEDs and the light sensor
  - Declare and initialize data packet
- **start()**
  - Start timer
- **event fired()**
  - Toggle red LED
  - Activate light sensor
- **event dataReady()**
  - Add the light sensor reading to the data packet
  - Send the data packet
  - Deactivate the light sensor
  - Toggle yellow LED
- **event sendDone()**
  - Toggle green LED

Concurrence in NesC

- Two threads:
  - Tasks
  - Events handlers
- Tasks:
  - Deferred start
  - Run to completion
  - Can not preempt other tasks
- Events handlers
  - Triggered in response to a HW interrupt, or user-event
  - Can preempt other events or tasks

Call Vs. Signal Vs. Post

- **Call**
  - Invokes a command immediately
- **Signal**
  - Invokes an event immediately
- **Post**
  - Invokes a task for later execution and returns immediately
atomic keyword

- Atomic blocks are executed asynchronously.
- It must be used when accessing global variables.

Provides/Uses

- includes sensorboardApp;
- module MyAppM {
  - provides {interface StdControl; }
  - uses {
    - interface Timer;
    - interface Leds;
    - interface StdControl as PhotoControl;
    - interface ADC as Light;
    - interface SendMsg;
    - }
  - }

Global Variables

- implementation {
  - bool sending_packet = FALSE;
  - TOS_Msg msg_buffer;
  - XDataMsg *pack;
init()

- command result_t StdControl.init() {
  - call Leds.init();
  - call PhotoControl.init();
  - atomic {
    - pack->xSensorHeader.board_id = SENSOR_BOARD_ID;
    - pack->xSensorHeader.packet_id = 2;
    - pack->xSensorHeader.node_id = TOS_LOCAL_ADDRESS;
    - pack->xSensorHeader.rsvd = 0;
    - }
  - return SUCCESS;
  - }

start()

- command result_t StdControl.start() {
  - return call Timer.start(TIMER_REPEAT, 1000);
  - }

stop()

- command result_t StdControl.stop() {
  - return call Timer.stop();
  - }
fired()  
• event result_t Timer.fired()  
• {  
•  call Leds.redToggle();  
•  call PhotoControl.start();  
•  call Light.getData();  
•  return SUCCESS;  
•  }

dataReady()  
• async event result_t Light.dataReady(uint16_t data)  
• {  
•  atomic pack->xData.datap1.light = data;  
•  post SendData();  
•  call Leds.yellowToggle();  
•  return SUCCESS;  
•  }

SendMessage()  
• void task SendMessage()  
• {  
•  call PhotoControl.stop();  
•  if (sending_packet) return;  
•  atomic sending_packet = TRUE;  
•  if (call SendMessage.send(TOS_UART_ADDR,sizeof(XDataMsg),&msg_buffer) != SUCCESS)  
•  sending_packet = FALSE;  
•  return;  
•  }
sendDone()

- event result_t SendMsg.sendDone(TOS_MsgPtr msg, result_t success)
- {
  - call Leds.greenToggle();
  - atomic sending_packet = FALSE;
  - return SUCCESS;
- }

Receiving the data

- Mote View

Sensing, and wireless transmission

- void task SendData()
- {
  - call PhotoControl.stop();
  - if (sending_packet) return;
  - atomic sending_packet = TRUE;
  - if (call SendMsg.send(TOS_BCAST_ADDR, sizeof(XDataMsg), msg_buffer) != SUCCESS)
    - sending_packet = FALSE;
  - return;
- }
TOSBase

- Captures all the packets that it can hear and report it back to the UART
- Forwards all incoming UART messages out to the radio

Resources

- TinyOS Tutorials
  - http://www.tinyos.net/tinyos-1.x/doc/tutorial/
- NesC Reference
  - www.tinyos.net/api/nesc/doc/ref.pdf
- MoteWorks Getting Started Guide

Questions