Introduction to TinyOS

Jianjun Wen
Outline

- Hardware Platforms
- Introduction to TinyOS
- Environment Setup
- Project of This Semester
## Hardware Platform

<table>
<thead>
<tr>
<th>mote type</th>
<th>Telosb</th>
<th>intelmote2</th>
</tr>
</thead>
<tbody>
<tr>
<td>processor</td>
<td>- msp430 16 bits&lt;br&gt;- 8 MHz</td>
<td>- pxa271 xscale 32bits&lt;br&gt;- 13 MHz ~ 104 MHz</td>
</tr>
<tr>
<td>memory</td>
<td>- 48 KB ROM&lt;br&gt;- 10 KB RAM</td>
<td>- 32 MB FLASH&lt;br&gt;- 256 KB SRAM&lt;br&gt;- 32 MB SDRAM</td>
</tr>
<tr>
<td>storage</td>
<td>1 MB external Flash</td>
<td>- 32 MB FLASH (shared)</td>
</tr>
<tr>
<td>radio</td>
<td>CC2420 (IEEE 802.15.4)</td>
<td>CC2420 (IEEE 802.15.4)</td>
</tr>
<tr>
<td>programming</td>
<td>USB</td>
<td>JTAG</td>
</tr>
<tr>
<td>sensors</td>
<td>Light, Temperature, Humidity</td>
<td>on sensor board:&lt;br&gt;- 3D accelerometer&lt;br&gt;- temperature, humidity&lt;br&gt;- light</td>
</tr>
</tbody>
</table>
TelosB

CC2420 radio

temp./ humidity sensor

light sensor
IntelMote2

CC2420 radio
PXA271 processor
reset button
power input
IntelMote2 - sensor board

- Temperature sensor (I2C)
- Humidity sensor (I2C)
- 3D accelerometer (SPI)
TinyOS

- operating system, but not full-fledged
- designed for low-power wireless embedded systems
- task scheduler + collection of device drivers
- component based and event driven
- static resource management (no “malloc” and “free”)
- running as “single application”
- written in NesC (C dialect)
A TinyOS Application

- an example of application

configuration comC{
  implementation{
    componentA.interfaceA->componentB.interfaceA
    ...
  }
}

module comP{
  uses interface InterfaceA;
  ...
  implementation{
    sometask...
    ...
  }
}

interface A{
  command cmd_1( ..);
  ...
  event void event_x( ..);
  ...
}

hardware platform
Basic Concept of TinyOS

- **component**
  - reusable pieces of code
  - similar to Class, groups related functionality into a single unit. e.g LedsC, TimerC
  - two types: *module* and *configuration*

- **interface**
  - access points to a component
  - define the operation specification: *commands & events*
module BlinkC @safe()
{
  uses interface Timer<TMilli> as Timer0;
  uses interface Timer<TMilli> as Timer1;
  uses interface Timer<TMilli> as Timer2;
  uses interface Leds;
  uses interface Boot;
}
implementation
{
  event void Boot.booted()
  {
    call Timer0.startPeriodic( 250 );
    call Timer1.startPeriodic( 500 );
    call Timer2.startPeriodic( 1000 );
  }
  event void Timer0.fired()
  {
    call Leds.led0Toggle();
  }
  ... ...
}

interface Leds {
/**
 * Toggle LED 0; if it was off, turn it on, if was on, turn it off.
 */
  async command void led0Toggle();
/**
 * Turn on LED 1. The color of this LED
 */
  async command void led1On();
}

configuration BlinkAppC
{
}
implementation
{
  components MainC, BlinkC, LedsC;
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2;

  BlinkC -> MainC.Boot;
  BlinkC.Timer0 -> Timer0;
  BlinkC.Timer1 -> Timer1;
  BlinkC.Timer2 -> Timer2;
  BlinkC.Leds -> LedsC;
}
Component: Module

- **module: two parts**
  - interface specification
    -> provide and use
  - code implementation
    -> define variables, functions
    -> implement interface commands and events
Component: Module

- **module: two parts**
  - interface specification
    -> provide and use
  - code implementation
    -> define variables, functions
    -> implement interface commands and events

```c
module BlinkC
{
  uses interface Timer<TMilli> as Timer0;
  uses interface Timer<TMilli> as Timer1;
  uses interface Timer<TMilli> as Timer2;
  uses interface Leds;
  uses interface Boot;
}
implementation
{
  ... unique name ...
}
```
Component: Module

- module: two parts
  - interface specification
    -> provide and use
  - code implementation
    -> define variables, functions
    -> implement interface commands and events
Component: Configuration

- configuration
  - similar to module
  - difference: connect components into larger abstractions

wiring: connect two components together
Component: Configuration

- configuration
  - similar to module
  - difference: connect components into larger abstractions

wiring: connect two components together
Component: Configuration

- configuration
- similar to module
- difference: connect components into larger abstractions

wiring: connect two components together

wiring operator: ‘->’, ‘<-’, ‘=’
Component: Configuration

- similar to module
- difference: connect components into larger abstractions
Generic Component

- singleton component: only one instance exists
  - components are singletons in TinyOS by default

- generic component: can have multiple instances

```cpp
configuration BlinkAppC {
}
implementation {
    components MainC, BlinkC, LedsC;
    components new TimerMilliC() as Timer0;
    components new TimerMilliC() as Timer1;
    components new TimerMilliC() as Timer2;
}
```

```cpp
generic configuration TimerMilliC() {
    provides interface Timer<TMilli>;
}
implementation {
}
```
Interface

- a group of logically related functions
- commands & events
- bidirectional interaction

```c
interface Timer<precision_tag> {  
  /**  
   * Set a periodic timer to repeat every dt time units.  
   */  
  command void void startPeriodic(uint32_t dt);  
  ... ...

  /**  
   * Signalled when the timer expires  
   * (one-shot) or repeats (periodic).  
   */
  event void fired();  
  ... ...
}
```

**must** be implemented in provider
called by user

**must** be implemented in user
signalled by provider
How they combined

- **BlinkAppC**
  - **Boot**
    - **MainC**
      - **RealMainP**
        - **Init**
          - **PlatformC**
    - **Leds**
      - **LedsC**
        - **TinySchedulerC**
      - **LedsP**
    - **Timer**
      - **Timer0 (TimerMilliC)**
      - **Timer2 (TimerMilliC)**
      - **HilTimerMilliC**
      - **AlarmMilli32C**
      - **PlatformLedsC**
      - **GeneralIO**
      - **Timer**

- Uses and provides relationships between components.
cross compiling process

1. race detection
2. generate a single c file
A **cross compiler** is a compiler capable of creating executable code for a platform other than the one on which the compiler is running [wikipedia]
Download and install TinyOS source code

```
wget http://github.com/tinyos/tinyos-release/archive/tinyos-2_1_2.tar.gz
```
```
tar -zxvf tinyos-2_1_2.tar.gz
```

Setup environment: put following lines into tinyos.env

```
1 export TOSROOT=/home/wnsn/wen/tinyos-2.1.2
2 export TOSDIR=$TOSROOT/tos
3 export CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java/tinyos.jar:$TOSROOT/support/sdk/java
4 export MAKERULES=$TOSROOT/support/make/Makerules
5 export PYTHONPATH=$PYTHONPATH:$TOSROOT/support/sdk/python
6
7 echo "setting up tinyos environment on path $TOSROOT"
```

* to make these environment variables permanently accessible, add following line to “~/.profile”:

```
source <file-location>/tinyos.env
```
Installation - tool chains

- Install cross compiler for imote2
  - xscale-elf-* is the specific tool chains for “xscale” processor
  - common arm-cross-tool-chain can also be used here e.g *arm-none-eabi-*

  wget http://sing.stanford.edu/intelmote2/tools/linux/xscale-elf-gcc-3.4.3-2.i386.deb
  sudo dpkg -i <tool-name>.deb
Installation - tool chains (cond.)

- OpenOCD: open on-chip debugger
  - debugging
  - in system programming

- installation, please refer to:

Install nesc tool chains
- xscale-elf-* is the specific tool chains for “xscale” processor
- common arm-cross-tool-chain can also be used here
  e.g *arm-none-eabi-*

```bash
wget http://tinyos.stanford.edu/tinyos/dists/ubuntu/pool/main/n/nesc/nesc_1.3.4-20120709_i386.deb
wget http://www.tinyos.net/dist-2.1.0/tinyos/linux/tinyos-deputy-1.1-1.fc9.i386.rpm
wget http://tinyos.stanford.edu/tinyos-rpms/tinyos-tools-1.4.0-3.ubuntu.i386.rpm
sudo alien -i <tool-name>.rpm
```
Let’s do it!
Our Project

- project description (summer semester 2016)
  
  - task 1: sensing
    - temperature, humidity, 3D accelerometer
    - transmission strategy:
      1. send what you got immediately;
      2. combine them in one packet.
  
  - task 2: MAC protocol
    - duty cycling -> energy efficiency
    - contention based, contention free …
      X-MAC, RI-MAC, or TDMA …
Our Project (cond.)

❖ project description (summer semester 2016)

- task 3: routing protocol
  - multi-hop
  - CTP ?

- task 4: data collection (sensor network to PC )
  - basestation -> PC, serial communication
  - printf like?
  - data storage: database or file-like
Our Project (cond.)

- project description (summer semester 2016)
  - task 5: presentation
    - GUI application on PC
    - display sensing data in real time
    - other statistical analysis: max, min, average
  - task 6: integration
    - integrate all above together and test.
Project Overview

- task 1
- sensor nodes
- relay nodes
- task 2 + task 3
- serial comm
- basestation / gateway
- task 4
- task 5
- task 6
some hint …

• some useful interfaces and components
  
  • task 1: sensing
    
    - components LIS3L02DQC -> 3D accelerometer
    - components TMP175C -> temperature
    - components Tsl2561C -> light
    - components SensirionSht11C -> humidity/temperature
    - interface Timer<TMilli>
    - interface Read<uint16_t>

* please refer to:
  $TOSROOT/tos/sensorboards/im2sb/examples/
some hint ...

* some useful interfaces and components

- task 2 + 3: communication in WSN
  - components ActiveMessageC
  - components AMsenderC
  - components AMReceiverC
  - interface LowPowerListening
  - interface AMSend
  - interface Receive

* please refer to:
  $TOSROOT/apps/tests/TestLpl
  $TOSROOT/apps/tests/TestNetwork or TestNetworkLpl
some hint ...

- some useful interfaces and components

  - task 4: serial communication (mote side)
    - components ActiveMessageC
    - components SerialActiveMessageC
    - interface AMSend
    - interface Receive

  - task 4: pc sides
    - database: sqlite, mysql .. or save as “cvs” file

* please refer to:
  $TOSROOT/apps/BaseStation
some hint …

- some useful interfaces and components
  - task 5: presentation
    - python, java, c/c++, whatever you prefer
    - some framework you can use:
      - wxpython, AWT?, Qt4..
about problem

- virtual machine
  virtual machine: https://www.dropbox.com/s/ntclm5zgrurjgqe/VM-wsn-project.zip?dl=0

- mailing list
  mailing list: https://mailman.zih.tu-dresden.de/groups/listinfo/sya-rn-wsn
  using subject: [WSN]: problem
[3] “Programming TinyOS”, David Culler, Phil Levis, Rob Szewczyk, Joe Polastre University of California, BerkeleyIntel Research Berkeley
Thanks!