Announcements (Mar. 1)

- Reading for next Tuesday: MauveDB (review due)
- Next Thursday (Mar. 8): project proposal talk
  - 15 minutes per group; 20% of total grade
  - What is it? Why do we care? Hasn’t it been done before? Plans, thoughts, and preliminary results?

General information

- Available: a total of 10 Moteiv Tmote Sky nodes with humidity, temperature, photosynthetically active radiation (PAR), and total solar radiation (TSR) sensors
  - $110 per node
  - Each team only gets a couple
  - If needed, we will reserve several periods during which one team can have most nodes
  - Simulate and debug before deploying larger networks!

- Resources
  - http://www.cs.duke.edu/courses/spring07/cps296.1/Resources.html
  - Besides searching mailing list archives, don’t be afraid to ask on tinyos-help!
Installing Tmote Sky tools

- Quick Start Guide
- Tool CD-ROM image
  - http://www.cs.duke.edu/courses/spring07/cps296.1/duke-only/tmote-2_0_4-cdrom.tgz
- Automatic installation: follow guide (p. 1-8)

Manual installation on Windows

- Install or update cygwin
  - Make sure you have all packages listed in guide (p. 15)
    - Some are obsolete; newer ones should be okay
- Install the latest Java development kit from Sun
- Unpack tmote-2_0_4-cdrom.tgz, and
  - cd cdrom
  - install/helper.sh install-windows-tinyos-base
  - install/helper.sh install-windows-tinyos-moteiv
- Plug a Tmote Sky node into your PC's USB port, and install USB serial COM driver (p. 7-8)
  - No battery needed when node is connected to USB port
- Reboot

Your first TinyOS application: Blink

- cd /opt/tinyos-1.x/apps/Blink
- Blink(.nc): top-level configuration,
  - Component Main mandatory
  - Optionally wire Main.StdControl to other components that need initialization, or starting and stopping at runtime
- BlinkM(.nc): module implementing LED blinking behavior
  - Relies on Timer and Leds interfaces
  - Want to look up what an interface, module, or configuration does?
    - /opt/moteiv/doc/nesdoc/index.html
Compiling/installing TinyOS programs

- Connect a node to a USB port
  - motelist command checks connection

<table>
<thead>
<tr>
<th>SerialNum</th>
<th>PortName</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M4AFBBKW</td>
<td>COM4</td>
<td>tmote sky</td>
</tr>
</tbody>
</table>

- make tmote
- make tmote install, 2
  - “2” specifies the node id (make sure it’s unique)
- make tmote reinstall, 0 bsl, 3
  - “reinstall” skips re- compilation
  - “bsl, 3” programs the node on COM4 (3 + 1)

TOSSIM

- A simulator for your TinyOS application
- make pc
- build/pc/main.exe
- DBG=am,led build/pc/main.exe –t=10 2
  - “DBG=am,led” prints communication/led activities
  - “–t=10” runs the simulation for 10 virtual seconds
  - “2” runs the simulation on 2 nodes

A sensing application: SenseTask

- cd /opt/tinyos-1.x/apps/SenseTask
  - Periodically samples the photo sensor
  - Maintain recent readings
  - Compute the average of recent readings and display the lowest three bits on LEDs
- A few interesting things in SenseTaskM.nc
  - Use of enum to define constants (why?)
  - Use of circular buffer to hold recent readings
  - Use of regular C functions
  - Use of atomic
  - Use of processData task in ADC.dataReady
Hmm… is the photo sensor working?

- DemoSensorC turns out to be temperature on Tmote Sky!
- Fear not; rewiring is easy
  - HamamatsuC is the photo sensor
  - HamamatsuC.TSR: total solar radiation
  - HamamatsuC.PSR: photosynthetically active radiation
- Another interesting thing in SenseTask.nc: use of parameterized interfaces (TimerC.Timer)
  - Parameter values known at compile time
  - ≈ different instances with different parameters
  - "unique(arg)" is a compile-time function call, every instance of which (with the same arg) will return a unique parameter value

Broadcast example: CountDual

- cd /opt/moteiv/apps/Count/CountDual
  - Node 1 periodically increments a counter, broadcasts its value, and sets its LEDs to this value
  - Other nodes set their LEDs to the value received over radio
- A few interesting things to note
  - ../CountMsg.h defines message type and structure
  - GenericComm: really basic (again parameterized interfaces)
  - SendMsg is split-phase (send/sendDone)
  - TOS_BCAST_ADDR specifies broadcast
  - ReceiveMsg: msg content valid only for duration of the function
  - Race condition?
    - Unless "async" is specified, commands and events cannot be preempted
  - Timer2<TMilli>: interface with type parameter specifying its precision

SP example: CountDualAck

- cd /opt/moteiv/apps/Count/CountDualAck
  - Node 1 periodically sets its LEDs according to a counter value and sends it to node 2
  - Node 2 sets its LEDs to the value received over radio
  - Node 1 only updates its counter if node 2 acks
- A few interesting things to note
  - SPC: the new primary link communication mechanism implementing Sensornet Protocol (SP), SenSys 2005
SP example: CountDualAck (cont’d)

- A few interesting things to note (cont’d)
  - `sp_message_t m_spmsg`: a handle to an SP message
  - `TOS_Msg m_msg`: TOS_Msg packet in the SP message
    - `SPSendNext` interface is for message with multiple packets
  - `SPSend.sendAdv`
    - Supports a number of fancy options
  - `SP_FLAG_C_RELIABLE`: attempts reliable transport (with ack and resend)
  - `SPSend.sendDone`
    - Lets CountDualAckP know whether send was successful

Communicating via serial port

- `cd /opt/moteiv/apps/Count/CountDualUART`
  - Periodically increments a counter, sends its value over UART, and sets LEDs to this value
    - UART (Universal Asynchronous Receiver/Transmitter), for communicating over serial port
  - Uses GenericComm, and specify TOS_UART_ADDR for sending a message through serial port
- `cd /opt/moteiv/apps/TOSBase`
  - A generic “base station” application
  - Send anything receive over radio to serial link
  - Send anything receive through serial link over radio

Accessing data from PC

- Using C
  - `/opt/tinyos-1.x/tools/src/sf/sf 9001 COM5 57600 tmote`
    - Listen on serial port “COM5” and forward data to port “9001”
  - `/opt/tinyos-1.x/tools/src/sf/sflisten localhost 9001`
    - Listen on port “9001” and dump data out
- Using Java
  - `MOTECOM=serial@COM5:tmote java net.tinyos.tools.Listen`
    - Listen on serial port “COM5” and dump data out
    - Source code in `/opt/tinyos-1.x/tools/java/net/tinyos/tools/`
Multi-hop example: Delta/Trawler

- cd /opt/moteiv/apps/Delta
  - Base station (connected to PC) is node 0
  - Everybody measure light periodically and sends it to base station
    - Requires replacing DemoSensorC with HamamatsuC
  - Visualize network and results using
    MOTECOM=serial@COM5:tmote java com.moteiv.trawler.Trawler
- MultiHop implements multi-hop node-to-root communication
  - Send: send to base station
  - Intercept: intercept messages that are supposed to be forwarded
    (may decide to drop or aggregate instead)

Not covered

- How to convert raw readings into meaningful measurements
  - [http://www.moteiv.com/community/Getting_Data_from_Tmote_Sky%27s_Sensors](http://www.moteiv.com/community/Getting_Data_from_Tmote_Sky%27s_Sensors)
- How to use the 1Mb external Flash