Crossbow: MoteWorks
Getting Started Guide

Presented by Catherine Greene, Bretny Khamphavong, Chloe Norris, and Nancy White
Sections 1-3
Presented by Catherine Greene
MoteWorks

- MoteWorks
  - end-to-end enabling platform for the creation of wireless sensor networks
  - easy-to-use wireless original equipment manufacturer
    - OEM refers to the company that originally manufactured the product (wikipedia)
    - Allows for more freedom
      - Unique differentiation to applications
      - Innovative solutions to the market quickly

Information taken from: http://en.wikipedia.org/wiki/Original_equipment_manufacturer
MoteWorks Getting Started Guide
Software Tiers

- A wireless network deployment is composed of three distinct software tiers
  - Mote Tier
    - Xmesh located
      - The software that has the networking algorithms that form a reliable communication to connect all the nodes in the mesh cloud to the server
  - Server Tier
    - Always on
    - The facility that handles translation and buffering of data coming from the wireless network and provides the bridge between the wireless motes and the internet clients
  - Client Tier
    - Provides a graphical interface and software (MoteView) for managing the network
      - Software is made for low-power battery-operated networks and provides an end-to-end solution across all the tiers of the wireless sensor networking applications

Information and Images taken from: MoteWorks Getting Started Guide
Software Tiers

Xmesh Landscape

Information and Images taken from: MoteWorks Getting Started Guide
Un/Installing MoteWorks

- One needs:
  - PC with Windows
  - 1 GB or more of free space in destination drive
  - 550 MB or more of space in C drive

- How to Install:
  - Insert MoteWorks CD in CD-ROM drive
  - Double click on MoteWorks_<version>_Setup.exe
  - InstallShield Wizard will come up and guide you on what to do

- How to Uninstall:
  - Can use the remove option for MoreWorks which can be found under Start>Control Panel>Add/Remove Programs
    - Removes *MoteWorks* Tree, Programmer’s Notepad and MoteConfig
      - but other components like viz., *Graphviz*, *XSniffer*, PuTTY and TortoiseCVS have to be removed separately from the add/remove programs wizard.

Information and Images taken from: MoteWorks Getting Started Guide
MoteWorks

- Comes with Programmer’s Notepad
  - Simple IDE for NesC code
  - Start>Programs>Crossbow>PN
- Comes with Cygwin
  - Unix/Linux emulation
  - Optional interface for compiling and downloading Mote applications in MoteWorks
  - Double clicking the icon on your desktop

Information and Images taken from: MoteWorks Getting Started Guide
Setting up Aliases

- It's recommended that you setup aliases
  - Commonly used commands
  - Aliases are to be edited at the bottom of the file called profile which is located in `<install dir>/cygwin/etc/
  - Useful for quickly changing to commonly used directories while in the Cygwin shell
  - Some the aliases appear as two lines, all are written as one line

```bash
alias cdMoteWorks="cd <install dir>/cygwin/opt/MoteWorks"
alias cdtools="cd <install dir>/cygwin/opt/MoteWorks/tools"
alias cdapps="cd <install dir>/cygwin/opt/MoteWorks/apps"
```

Information and Images taken from: MoteWorks Getting Started Guide
Compiling and Platforms

- Compiling MoteWorks applications can be done in a Cygwin window
  - "make <platform>"

<table>
<thead>
<tr>
<th>Processor/Radio Platform</th>
<th>For &lt;platform&gt; use</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICAz (MPR2400 series)</td>
<td>micaz</td>
</tr>
<tr>
<td>MICA2 (MPR4x0 series)</td>
<td>mica2</td>
</tr>
<tr>
<td>MICA2DOT (MPR5x0 series)</td>
<td>mica2dot</td>
</tr>
<tr>
<td>M2100 or XM2100</td>
<td>m2100</td>
</tr>
<tr>
<td>M2110 or XM2110</td>
<td>m2110</td>
</tr>
<tr>
<td>M9100 or XM9100</td>
<td>m9100</td>
</tr>
</tbody>
</table>

Information and Images taken from: MoteWorks Getting Started Guide
Programming

- Micro In-System Programmer (UISP)
  - Standard programming software
  - Takes various arguments according to the programmer (erase, verify, program, etc.).
    - You need to specify the type of device you are using and how to communicate with it
      - Done using environment variables

Information and Images taken from: MoteWorks Getting Started Guide
Installing MW Apps into a Mote

- Programming tools include a method of programming unique node addresses without having to edit source code.
- To set the node address/ID during program load, the syntax for installing is "make <platform> re/install,<n> <programmer>,<port>"
  - <programmer> and <port> are the name of the programmer the port ID or address or number of the host PC to which the programmer is attached.
  - <n> is an optional number (in decimal) to set the node ID or address.
    - Assigning a node ID ("", <n>") is optional.
  - <platform> is the type of Mote processor/ radio hardware platform.
  - "install,<n>" compiles the application for the target platform, sets the node ID/address and programs the Mote.
  - "reinstall,<n>" sets the node ID/address and downloads the pre-compiled program (into the mote) and it does not recompile, using this option is a lot faster.

Information and Images taken from: MoteWorks Getting Started Guide
Automated Tools

- Build command filters out the compile output to highlight only error messages and warnings
- Buildall command performs an automated build of all applications under that application folder
- Flash command flashes an image onto the Mote
- Flashall command flashes an image onto a test bed of motes
- Fuses command allows the user to read or write the fuse settings of the mote on the programming interface board
- Motelist command lists MIB=520 and Telos devices that are attached to the USB port
- Gettos command allows the user to see how their current TinyOS environment is configured
- Settos command allows a user to switch to a new MotesWorks tree by changing the symbolic link
  - The first time this is run it renames your current MoteWorks tree to the specified version
- The usetos command allows a user to switch between MoteWorks and a legacy TinyOS environment
  - usetos switches to MoteWorks, usetos tinyos switches to TinyOS 1.x, etc..
- The make command (make <platform>) allows users to compile their nesC code with many options from the command line

Information and Images taken from: MoteWorks Getting Started Guide
Reviewing TinyOS and nesC
Sections 4 and 5

Presented by Bretny Khamphavong
Primary Concepts of TinyOS

- **Application**: set of components linked together to form a run-time executable
- **Component**
  - Module - implements one or more interfaces
  - Configuration – “wires” other components together
- **Interface**
  - Bidirectional - specify both commands that a module must implement and events that modules must handle
Application Make Up: Makefiles

- *Makefiles* and nesC files that implement and wire up the application
- *Makefile* and *Makefile.component* define the dependencies for an application
  - *Makefile* tends to have the same contents across all applications
  - *Makefile.component* can be used to specify dependencies for this particular application
Application Make Up: nesC files

- nesC files can be identified because they use the extension "\.nc" for all source files—interfaces, modules, and configurations.
- Comments inside these files can either be single line "//" style comments, or multiline "/* */" style comments.
**nesC Example Code**

- Modules are nesC files that perform two main functions:
  - Define the interfaces the module provides
  - Implement those interfaces with nesC code
- **StdControl** interface with implementation that returns SUCCESS when each function is called
- The interfaces provided and implementation are separated into a provides and implementation block respectively

```nesC
module ModuleName {
  provides {
    interface StdControl;
  }
}
implementation {
  command result_t StdControl.init() {
    return SUCCESS;
  }
  command result_t StdControl.start() {
    return SUCCESS;
  }
  command result_t StdControl.stop() {
    return SUCCESS;
  }
}
```
Wired Configurations

- A configuration can also provide interfaces by wiring components together into more complex interface providers, but it is not required to.
- In the implementation section of an application configuration, the modules are wired together.
- For example:

  ```
  Main.StdControl -> MyAppM.StdControl;
  ```

- Tells the compiler that the Main.StdControl interface is provided for by the StdControl interface in MyAppM.
All Applications Must Have “Main” Component

- Referred to as the scheduler, or driver, of the application
- All nesC application execution starts in this component
- It must be properly wired into the application with the application configuration
Sensing Application and XMesh Sections 6 and 7

Presented by Chloe Norris

All Information and Images taken from: MoteWorks Getting Started Guide
Section 6

A Simple Sensing Application

All Information and Images taken from: MoteWorks Getting Started Guide
Hardware Requirements

- two standard edition Motes
  - of MICA2 (MPR4x0), MICAz (MPR2400), XM2100, XM2110 or XM9100 or OEM editions MPR600, MPR2400, M2100, M2110 or M9100
- one sensor or data acquisition board
  - MDA100, MTS300 or MTS310
- one gateway board
  - MIB510, MIB520, or MIB600 and the associated hardware (cables, power supply) for each
- Windows PC with MoteWorks installed.
A Simplified Sensing Application

- Take light readings using one of the following sensors boards: MTS300/310 or MDA100
- Use the Mote serial port (UART) and radio to send sensor data to the base station
- Blink the yellow LED when the sensor is sampled
- Blink the green LED when the sensor data message is successfully sent to the base station
- Compile and debug if necessary

All Information and Images taken from: MoteWorks Getting Started Guide
Getting Started

- Application’s configuration is located in the MyApp.mc file
- To create the applications configuration, the illustration to the left would be entered into the Programmers Notepad

```cpp
#include "sensorboardApp.c"

/**
* This module shows how to use the Timer, LED, ADC and Messaging components.
* Sensor messages are sent to the serial port
* @Author Crossbow Technology Inc.
**/ 
configuration MyApp {
}
implementation {
    components Main, MyAppM, Timer0, Leds0, Photo, GenericComm as Comm;
    Main.StdControl -> TimerC.StdControl;
    Main.StdControl -> MyAppM.StdControl;
    Main.StdControl -> Comm.Control;
    MyAppM.Timer -> TimerC.Timer[unique("Timer")];
    MyAppM.Leds -> LedsC.Leds;
    MyAppM.PhotoControl -> Photo.PhotoStdControl;
    MyAppM.Light -> Photo.ExternalPhotoADC;
    MyAppM.SendMsg -> Comm.SendMsg[AM_XSXMSS];
}
```

All Information and Images taken from: MoteWorks Getting Started Guide
• Blinking lights every seconds
  ◦ Firing of the timer, sampling light sensor, and then sending message back to base station
  ◦ Red: 1 second timer event fired
  ◦ Yellow: light sensor has been sampled
  ◦ Green: Sensor message has been sent back to base station
**XServe**

- **XServe** is an application that installs with MoteWorks for the purpose of displaying sensor message packet contents as they arrive on the PC over serial port.

All Information and Images taken from: MoteWorks Getting Started Guide
Sending Sensor Data over the Radio

- One change needed in the code of the MyAppM.nc file
- SendMsg.send command decides where the message packet should be sent
- TOS_BCASE_ADDR tells the communications component to send the message through the radio.

```c
if (call SendMsg.send(TOS_BCAST_ADDR, sizeof(XDataMsg), &msg_buffer) != SUCCESS)
```

All Information and Images taken from: MoteWorks Getting Started Guide
Using Xsniffer to View Sensor Data Sent Over The Radio

- XSniffer used to eavesdrop on messages sent over the Mote radios.
  - Monitor messages sent from modified sensing application
- Modify the sensing application in the /lesson_3 folder onto a Mote.
- Tools>shell
- make mica2 install, 1 mib510, com1
- Remove the Mote from the programming board
- plug one of the sensorboards onto the Mote and turn it on

All Information and Images taken from: MoteWorks Getting Started Guide
Using Xsniffer to View Sensor Data Sent Over The Radio Continued

- Install the XSniffer application onto another Mote
- Node id of 2
- Start Xsniffer by double clicking desktop icon
- Options>General Packet Type
- Go back to Log Tab
- Select COM port connected to programming board
- Click start to begin “Sniffing”
Using Xsniffer to View Sensor Data Sent Over The Radio Continued

- Elapsed time the messages are begin sent about 1 second apart
- Each time the LEDs blink you should see a new message captured by XSniffer.
Using a Sensorboard

- Specify the sensorboard
- Send a message containing the sensor data back to the base station
- GenericComm - used to send messages through the UTART port over to the radio
XSensor Applications Supported in MoteWorks

- Crossbow’s sensor and data acquisition boards supported with *XSensor enabled applications*
- *XSensor* applications are test applications for Crossbow’s sensor data acquisition boards.
- Quickly and easily test sensor and data acquisition boards
- Send data over one hop

All Information and Images taken from: MoteWorks Getting Started Guide
Section 7

XMesh enabled Sensing Application

All Information and Images taken from: MoteWorks Getting Started Guide
Hardware Requirements

- Two Motes
  - standard editions of MICA2 (MPR4x0), MICAz (MPR2400), XM2100, XM2110 or XM9100 or OEM editions MPR600, MPR2400, M2100, M2110 or M9100.

- One sensor or data acquisition board
  - MDA100, MTS300 or MTS310

- One gateway board
  - MIB510, MIB520, or MIB600 and the associated hardware (cables, power supply) for each

- A Windows PC with *MoteWorks installed*
A Simple Sensing Application

- Simple sensing application using the XMesh multi-hop networking service would:
  - Take light readings
  - Use the Mote serial port (UART) and radio to send sensor data to the base station
  - Blink the yellow LED when the sensor is sampled
  - Blink the green LED when the sensor data message is successfully sent to the base station
  - Compile and debug if necessary

All Information and Images taken from: MoteWorks Getting Started Guide
Getting Started

- Create folder for code
- To create the application’s configuration, enter the text shown on the right in the Programmer’s Notepad
- Save File

```cpp
#include sensorboardApp;

/**
 * This module shows how to use the Timer, LED, ADC and Messaging components.
 * Sensor messages are sent to the serial port
 *
 * @author Crossbow Technology Inc.
 */

configuration MyApp {
}

implementation {
    components Main, MyAppM, TimerC, LedsC, Photo, GenericComm as Comm;
    Main.StdControl -> TimerC.StdControl;
    Main.StdControl -> MyAppM.StdControl;
    Main.StdControl -> Comm.Control;
    MyAppM.Timer -> TimerC.Timer[unique("Timer")];
    MyAppM.Leds -> LedsC.Leds;
    MyAppM.PhotoControl -> Photo.PhotoStdControl;
    MyAppM.Light -> Photo.ExternalPhotoADC;
    MyAppM.SendMsg -> Comm.SendMsg[AM_X8XMSG];
}
```
Using XSniffer to View Sensor Data Through the Network

- Monitor the messages being sent from the sensor node
- Remove the XMeshBase programmed Mote from the programming board
- Install the XSniffer application onto a third Mote that you will plug into your programming board (base station)
- Node id of 2
- Start the XSniffer application by double clicking on the icon on your desktop
- Options > XMesh > Log tab

All Information and Images taken from: MoteWorks Getting Started Guide
Using XSniffer to View Sensor Data Sent Over the Radio Continued

- Select COM port connected to programming board
- Click Start to begin “Sniffing”
- You should see message packets displayed in Xsniffer
- Remove the XSniffer Mote from the programming board and plug the XMeshBase Mote back into the programming board
- File>Connect>Connect to Database.
- mts310_results and click Apply
- MoteView main menu select File>Connect>Connect to MIB510/MIB520/MIB600/Stargate.

All Information and Images taken from: MoteWorks Getting Started Guide
Using XSniffer to View Sensor Data Sent through the Network

- Set the COM port value
- **XMTS310** application
- **Advanced** tab
- In **Data Logging Options** menu, check the box for **Spawn Separate Shell**
- Click Start to begin “Sniffing”
- All of Crosbow’s sensor and data acquisition boards are supported with **XMesh** enabled applications.

All Information and Images taken from: MoteWorks Getting Started Guide
XMesh Advanced Features
Sections 8 and 9

Presented by Nancy White
8.1 Hardware Requirements

- Two motes
- One gateway board
- A Windows PC with MoteWorks
End-to-End Acknowledgements

- In the MyApp subdirectory /lesson5 it shows how to use XMesh end-to-end acknowledgment, which have code to modify transport requests to the base station.
- A yellow LED light blinks when a message is received.
MyApp will need to be installed on two mote’s, one of the mote’s will be the sensor node while the other one will function as the base station.

- The mode you wish to use as the sensor node should be plugged into the programming board
- The red and green lights will flash until a network is formed, once the network is formed the yellow light will flash.
ReceiveAck file allows for interface writing and requires a callback function that is generated by XMesh.

- MODE_UPSTREAM_ACK tells XMesh to send a message acknowledging that the message was received to the base station
- ReceiveAck.receive is another acknowledgment message that confirms a message has arrived from the base station and the LED light will flash green
MyApp subdirectory /lesson6

- Shows how to implement command processing
- Requires 2 mote’s, one will function as the sensor node, and the other as the base station, which is plugged into the programming board and connected to your PC.
- `get_config`, which will return the current configuration parameters for a mote
set_rate and is used to change the motes sampling rate
XCommand component provides the functionality for processing downstream commands
XCommand provides a single event name received which implements the application module and is signaled when a command arrives to the node.
Data Logging Application
Data Logging Application

- This section teaches you how to read and write data from external flash on a mote.
- Requires a Windows PC with MoteWorks, two motes, and one gateway board.
- Allows the user to read and write operations at the external flash is ByteEEPROM.
ByteEEPROM

- Allows you to log the number of light sensor readings in the external flash.
- When a new reading comes it over-writes the previous reading.
- Once the new reading is written to the external flash the logged data is read back from the flash and is placed in a data packet on the computer.
ByteEEPROM  Cont’d

- The node that has a node id of 0 will always be the base station
- Uses XServe to display the incoming packets on the computer
- ByteEEPROM component is required to request memory in the external flash and carry out read and write operations
- All changes that need to be made use the interface AllocationReq, ReadData, and WriteData of ByteEEPROM
To Sum it up...

- The MoteWorks Getting Started Guide is a very helpful reference when aid is needed with:
  - Uninstalling/reinstalling software
  - How-to’s with commands and programming
  - TinyOS and NesC help
  - Running several different applications
  - Several different MoteWorks features