SunSPOT

February 9, 2010
• Introduction to SunSPOT
• Programming with SunSPOT
• References
• Demonstration
What is this?

- Sun SPOT stands for SUN’s Small Programmable Object Technology
- The SunSPOT device is a small, wireless and battery powered platform that:
  - includes a range of built-in sensors
  - interacts through a number of LEDs and buttons
  - it is programmed entirely in Java
- SunSPOT provides a flexible hardware and software platform for developers to
  - experiment
  - prototype whatever they can imagine
  - innovate
This is ...

... a SunSPOT!!!!
• The Sun SPOT project started on November 2004
• Follows-on to Epsilon and Anteater projects
• Used in a number of applications:
  • teaching tool in the classroom
  • research tool (e.g., Actuation/Control and robotics), for wide range of research from wireless networks to gesture interfaces to new security devices
  • hobbyist tools, easy to program and easy to interface
Some of the SunSPOT application examples:

- computer interaction device for 3D-application
- water quality management (pH, redox, turbidity)
- monitoring a small rocket’s flight from a laptop
- detecting microwaves
- video game controller
- motor control
- simple apps (e.g., games) running only in the SunSPOT
Application Examples

SunSPOT
• SunSPOT has three layers: Battery, Processor Board with Radio, Sensor Board

• User programs the device entirely in Java using standard Java tools
Hardware Characteristics

- SunSPOT have common characteristics with mobile phones
- SunSPOT Processor Board:
  - 180MHz 32-bit ARM processor, 512K RAM, 4M Flash
  - 2.4GHz radio, IEEE 802.15.4 compliant, with an antenna
  - USB interface to connect to a PC
  - rechargeable lithium-ion battery
Hardware Characteristics

- SunSPOT Sensor Board has the following characteristics:
  - 8 tri-color LEDs
  - 2 momentary switches
  - a 3-axis accelerometer
  - a temperature sensor
  - a light sensor
  - analog to digital input, gpio, high current output pins
Hardware Characteristics

- Each SunSPOT has a unique IEEE 64-bit address
  - Expressed as four sets of four-digit hexadecimal numbers: nnnn.nnnn.nnnn.nnnn.
  - The first eight digits will always be 0014.4F01
  - The last eight digits should be printed on a sticker on the SunSPOT
• SunSPOT have the following software characteristics:
  • run a Java VM (Squawk VM) that provides basic OS functionality
  • Squawk VM implements Java Micro Edition (JME)
  • all major parts of the Sun SPOT project are open source: hardware, operating system/virtual machine, drivers and libraries, applications

• The SPOTManager and SPOTWorld tools are provided for managing SDK and SunSPOT devices

• A SunSPOT emulator is also provided that runs applications in virtual SunSPOT
The SPOTManager tool provides the following functionalities:

- allows you to query and change the configuration of individual SunSPOT
- allows you to download and install versions of the SunSPOT SDK
- launches SPOTWorld, a tool for managing individual SunSPOT and simulating virtual SunSPOT
- gives you access to the contents of discussion groups on the SunSPOT website
- displays standard output for the SPOTManager tools
SunSPOT are usually used in the following topology:
• The free-range SunSPOT communicate wirelessly with other SunSPOT and the basestation
• The basestation is a SunSPOT connected, through USB, with a host
• The host applications interact with the SunSPOT through the basestation application
• The host application is a J2SE program, that also has access to a subset of the API of the libraries used by SunSPOT
• A host application communicates with a SunSPOT via a basestation using code identical that which you would use to communicate between two SPOTs
• The basestation may run in either dedicated or shared mode

• In the shared model, the host application has its own address, distinct from that of the basestation

• The main advantage of shared mode is that more than one host application can use the same basestation simultaneously

• The disadvantage of shared mode is that run-time manipulation of the basestation SunSPOT’s radio characteristics is not possible
Operating System:

- The SunSPOT development software has been tested on Windows XP, Macintosh OS X 10.4 and Linux
- Successful installation of the SunSPOT SDK for both Vista and Leopard have been reported

For the development of SunSPOT applications a number of Integrated Development Environments (IDE) can be used:

- NetBeans: very good integration with SunSPOT
- IntelliJ Idea
The following editions of the Java platform are available:

- Standard Edition (Java SE)
- Enterprise Edition (Java EE)
- Micro Edition (Java ME)

SunSPOT use a fully capable Java ME implementation, called Squawk
Java ME offers a framework for developing Java applications for embedded devices, such as mobile phones.

Java ME has two defined subsets of libraries, called configurations:

- the more capable Connected Device Configuration (CDC)
- the more restricted Connected Limited Device Configuration (CLDC) that has less requirements from the mobile device

On top of these configurations are profiles, which add specific class libraries.

Mobile devices, such as mobile phones and PDAs, use the Mobile Information Device Profile (MIDP).

SunSPOT supports CLDC 1.1 and MIDP 1.0.
Sun SPOT Libraries

- Standard JME libraries (e.g., javax.microedition.*)
- Hardware libraries: Sensor board hardware driven by Java technology (e.g., com.sun.spot.sensorboard. *)
- Radio libraries (e.g., com.sun.spot.peripheral.radio. *)
- Network libraries: Connection framework interface (e.g., com.sun.spot.io.j2me.radiogram. *)
- Desktop libraries (e.g., java.util)
- Testing libraries (e.g., junit.framework)
All application root directories have the same layout:

- The root directory contains two files that control the ant script used to build and run applications
  - build.xml
  - build.properties
- The root directory also contains two main sub-directories
  - src, is the root of the source code tree for this application
  - resources, contains the manifest file that defines the application, plus any other resource files that the application needs at run time
- other IDE specific or not directories may exist or appear after the build process
• In Java SE, an application consists of a static main() method defined in one of the loaded classes
• In Java ME, an application is defined as a class that extends the MIDlet class
• The Java ME applications are also called MIDlets
• All Java ME applications implement the three members
  • startApp(): application acquires the resources that it requires and starts executing
  • pauseApp()
  • destroyApp(): application releases resources and stops executing
• If the application wants to exit, it must call notifyDestroyed()
Isolates

- In SunSPOT multiple applications run in one Java Virtual Machine (JVM), instead in many as in standard Java ME.
- Squawk VM supports multiple independent execution spaces, called Isolates.
- Each Isolate runs a separate set of threads.
- Each SunSPOT has a Master Isolate, which runs threads supporting system services.
- A user’s application is by default started in the Master Isolate.
- Multiple applications in a SunSPOT run in child Isolates.
• Some resources are unique, such as a radio connection on a specific port number
• The first Isolate to ask for that port will successfully be given access to it, while any subsequent requests from other isolates will fail
• Other resources are truly shared such as the LEDs
• One Isolate might turn an LED on, and then another might turn it off or change its color
Communication

• Every SunSPOT can act as a mesh router, forwarding packets it receives on to other SunSPOT
• This way multihop communication is possible
• AODV (Adhoc On-demand Distance Vector) multihop protocol is used
• Two communication protocols are supported:
  • The radiostream protocol provides reliable, buffered, stream-based communication
  • The radiogram protocol provides unreliable datagram-based communication
• It is possible to broadcast datagrams
• Http connections can be started from any SunSPOT to any accessible web service
To establish a connection both ends must open connections specifying the same port number and complimentary IEEE addresses:

```java
StreamConnection conn = (StreamConnection);
Connector.open("radiostream://nmm.nmm.nmm.nmm:xxx");
```

Once the connection has been opened, each end can obtain streams to use to send and receive data:

```java
DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos = conn.openDataOutputStream();
```
To establish a connection both ends must open connections specifying the same port number and complimentary IEEE addresses:

```java
DatagramConnection conn = (DatagramConnection) Connector.open("radiogram://" + targetIEEEAddress + ":100");
Datagram dg = conn.newDatagram(conn.getMaximumLength());
dg.writeUTF("My message");
conn.send(dg);
...
conn.receive(dg); String answer = dg.readUTF();
```
Broadcasting and "passive" listening is also possible:

```java
DatagramConnection sendConn = (DatagramConnection)
Connector.open("radiogram://broadcast:100");
dg.writeUTF("My message");
sendConn.send(dg);
...

DatagramConnection recvConn = (DatagramConnection)
Connector.open("radiogram://:100");
recvConn.receive(dg);
String answer = dg.readUTF();
```
Storage

- An area in the Flash is reserved for persistent storage that can be read and written from SunSPOT applications.
- The Record Management Store (RMS) that is part of the standard Java ME, provides a simple record-based mechanism for access to persistent data.
- One can create many "recordstores". Each recordstore is a kind of a very simple database, where each row consists of a record ID, followed by a series of bytes.

<table>
<thead>
<tr>
<th>Record</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data</td>
</tr>
<tr>
<td>2</td>
<td>Data</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Create/Open a recordstore

```java
RecordStore rs = RecordStore.openRecordStore("EventsRecord", true);
```

Insert a record to the recordstore

```java
String newEvent = "Time 11:00:00, Someone shoot me!!";
byte newData[] = newEvent.getBytes();
rs.addRecord(newData, 0, newData.length);
```

Read a record from the recordstore

```java
byte oldData[] = new byte[1];
rs.getRecord(recordID, oldData, 0);
```
The SunSPOT sdk provides API for accessing the sensors (accelerometer, temperature and light sensor, LEDs, switches, pins):

- All of the sensor board input devices also have listener classes associated with them

```java
// Create sensor instances
ITemperatureInput tempS = EDemoBoard.getADCTemperature();
ILightSensor lightS = EDemoBoard.getInstance().getLightSensor();

// Get readings
double celsiusTemp = tempS.getCelsius();
int lightSensorReading = lightS.getValue();
```
There are three axes on which the accelerometer measures acceleration:

- The Z-axis is perpendicular to the SunSPOT boards
- The X-axis is parallel to the row of LEDs on the sensor board
- The Y-axis is parallel to the long edge of the sensor board

// Create an accelerometer instance
IAccelerometer3D ourAccel = EDemoBoard.getInstance().getAccelerometer();

// Read from the accelerometer
double x-accel = ourAccel.getAccelX();
There are eight three-color LEDs on the demo sensor board, in a row, with LED1 on the left and LED8 on the right.

// Create LEDs instance
ITriColorLED[] ourLEDs = EDemoBoard.getInstance().getLEDs();

// Set color
ourLEDs[1].setColor(LEDColor.BLUE);

// Set on and off
ourLEDs[2].setOn();
ourLEDs[3].setOff();
Ordinarily you will implement an event loop which will check the switches used in your application on a periodic basis, or you will ask the SunSPOT to stop and wait for the switch state to change.

```java
// Create Switches instance
ISwitch[] ourSwitches = EDemoBoard.getInstance().getSwitches();
if(ourSwitches[0].isOpen())
{
    // if it is open, wait for it to close
    ourSwitches[0].waitForChange();
}
```
In general there are three ways in order to develop, build, deploy and manage an application and the corresponding SunSPOT:

- command line: develop, build, deploy
- IDE: develop, build, deploy
- SPOTManager: deploy, manage
References

1. https://www.sunspotworld.com/
2. https://www.sunspotworld.com/forums/
3. http://www.sunspotworld.com/docs/Purple/javadoc/
6. https://spots.dev.java.net/
Let’s play :-)

SunSPOT