Sun™ Small Programmable Object Technology (Sun SPOT) Owner's Manual

Sun Labs

August 2006

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Revision 1.0, August 2006

Please
Recycle

Sun™ Small Programmable Object Technology (Sun SPOT) Owner's Manual

This document provides a quick introduction to the Sun Small Programmable Object Technology (Sun SPOT) kit.

Contents of the Sun SPOT kit

A Sun SPOT kit contains the following:

- one basestation Sun SPOT unit
- two free-range Sun SPOT units
- a USB cable for connection between a standard USB port and a Sun SPOT unit
- one Sun SPOT CDROM disk
- two mounting brackets, each allowing a Sun SPOT unit to be wall-mounted
- one mounting bracket, designed to allow mounting of a Sun SPOT to a circuit board

How to open a SPOT

You must open the Sun SPOT unit lid to be able to reach the switches and LEDs on the sensor board. To open the lid, press down *firmly*, down and back, on the edge of the lid near the small raised dot. You can think of that small-raised dot as the fingernail-catching dot. The closer to the edge of the lid that you press, the easier the lid will open. The opposite end of the lid will pop up.

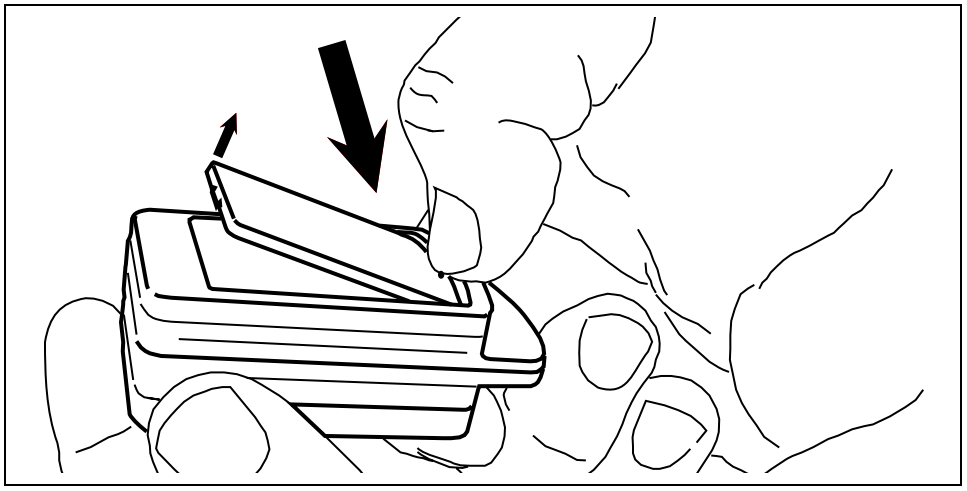


FIGURE 1 Press down *firmly* on the edge of the lid marked with a small raised dot.

After the lid will pops up, pull the lid out and away.

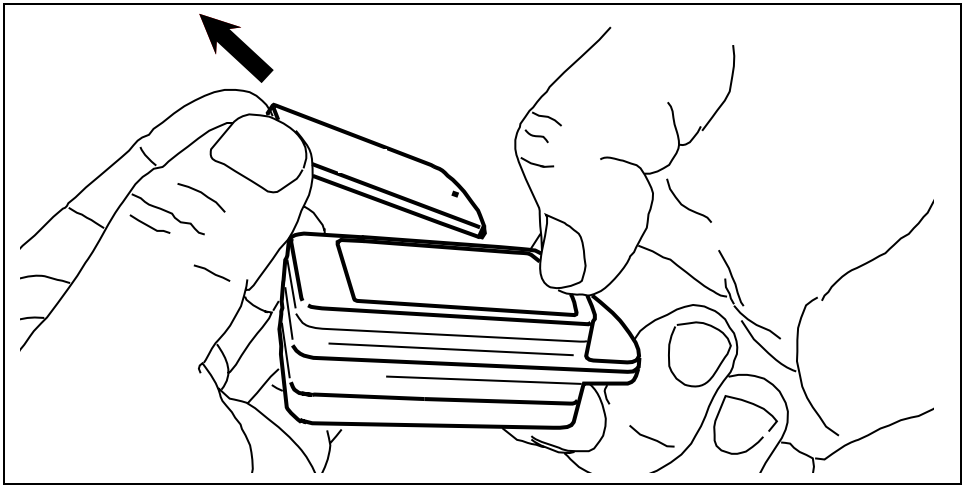


FIGURE 2 Pull the lid out and away.

Guided tour of SPOT switches and LEDs

The Sun SPOT unit has one switch and one connector which are accessible without removing the case lid. These are shown below:

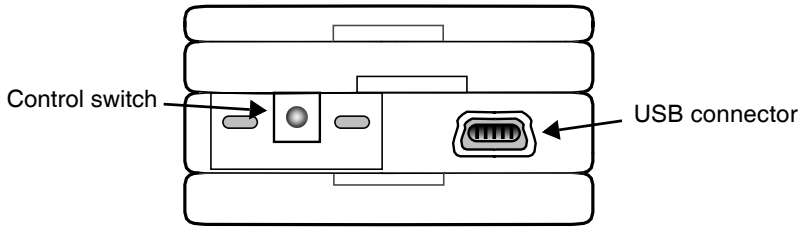


FIGURE 3 Sun SPOT exterior switch and connector

The connector is the micro-USB connector which allows the Sun SPOT unit to be connected to a host workstation.

The switch is the Sun SPOT unit control switch. If the Sun SPOT unit is off, pressing the switch will turn the Sun SPOT unit on and cause it to boot. If the Sun SPOT unit is on, pressing the control switch will cause the Sun SPOT unit to reboot. If the Sun SPOT unit is on, pressing the control switch and holding it down will turn the Sun SPOT unit off.

This end of the Sun SPOT unit also has two LEDs behind the plastic casing.:

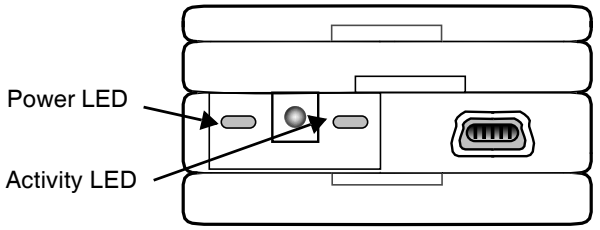


FIGURE 4 Sun SPOT unit LEDs

The power LED is to the left of the power switch. This LED will exhibit the following behaviors:

TABLE 1 Power LED Behavior

Power State	Power LED Behavior
Powering up	One bright green pulse, sharp on, soft off
Powering down	Three bright red flashes
Charging the battery while CPU is active	Slowly alternate between a dim green and a bright green on a eight second cycle
Charging the battery when CPU is asleep	Slowly alternate between off and a dim green on an eight second cycle
External power supplied, but not charging, CPU active	Steady dim green
Battery low	Slowly alternate between off and a dim red
Power fault	Two short red flashes
CPU going to sleep	Short red flash, short green flash
External interrupt or alarm, including button tap.	One short green flash

The activity LED is to the right of the power switch. This LED is under Java program control and can be used in your applications, but it is usually used by the system software. Some of these uses are:

- When the Sun SPOT unit is attempting to synchronize with a host workstation, the activity LED will flash amber 16 times a second. This flashing will last for two seconds or until synchronization is complete, whichever comes first.
- When the Sun SPOT unit is being used as a basestation, that is, for wireless communication between a host workstation and free-range Sun SPOT units, the green component of the activity LED will change state, i.e. switch from off to on or the reverse, for every packet received on the Sun SPOT unit from the host workstation. The red LED component of the activity LED will change state for every packet sent to the host workstation from the Sun SPOT unit.

If the Sun SPOT unit lid has been removed, there are two switches and eight multi-color LEDs that become accessible.

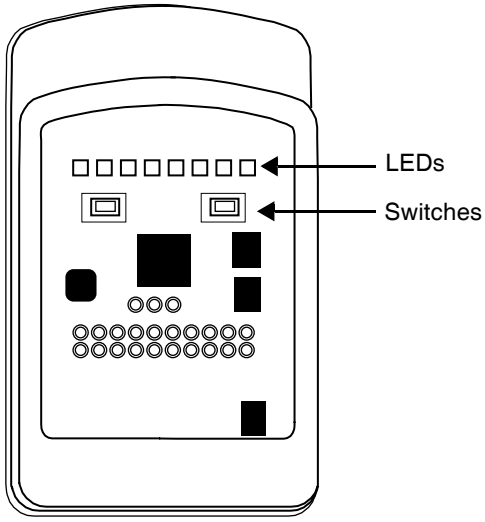


FIGURE 5 The Sun SPOT unit after the lid has been removed

The LEDs have red, blue and green components. The switches and LEDs have no fixed purpose and are under Java program control.

Powering a SPOT

The capacity of the built-in battery is 720 milliampere-hours.

The drain of the system varies with use.

TABLE 2 Power usage for typical a Sun SPOT unit

Processor board state	Radio	Sensor Board	Current draw
Deep sleep mode	Off	Any	~33 microamperes
Shallow sleep[1]	Off	Not present	~24 milliamperes
Shallow sleep	On	Not present	~40 milliamperes
Awake, actively calculating	Off	Not present	~80 milliamperes
Awake, actively calculating	On	Not present	~98 milliamperes
Shallow sleep	Off	Present	~31 milliamperes
Shallow sleep	On	Present	~46 milliamperes
Awake, actively calculating	Off	Present	~86 milliamperes
Awake, actively calculating	On	Present	~104 milliamperes

1. Shallow sleep means devices active, but no active threads.

Changing the transmit power of the radio effects the current draw slightly. Reducing the transmission power from 0db to -25db results in a savings of about 3 milliamperes.

LEDs also use power.

TABLE 3 Power draw for a single Sun SPOT LED

LED	Current draw
All elements, full brightness	25 milliampere
Blue element, full brightness	10 milliampere
Red element, full brightness	9 milliampere
Green element, full brightness	5 milliampere

The current draw for the LEDs is reasonably linear. An LED at half-brightness will draw approximately half the current of an LED at full brightness. Reducing LED brightness to the minimum required for your situation is often a good way to conserve power. Often LED levels of 20, out of a possible 255, are reasonably visible.

The approximate length of time that a full-charged Sun SPOT unit can operate is shown below for most of the conditions of interest:

TABLE 4

Sun SPOT state	Battery life estimate
Deep sleep	909 days
Shallow sleep, no radio	23 hours
Shallow sleep, radio on	15 hours
CPU busy, no radio	8.5 hours
CPU busy, radio on	7 hours
Shallow sleep, 8 LEDs on, no radio	3 hours

A power fault occurs when one of these conditions occurs:

- external power exceeds 5.5V
- Vbatt exceeds 4.9V
- Vcc ± 10% of 3.0V
- Vcore ± 10% of 1.8V
- Battery Discharge current exceeds 500ma

A Sun SPOT may also be powered by removing the demo sensor board and supplying power to pins J1 and J2 of the CPU-board top connector. The power should be between 4.5v and 5.5v and at least 1A.

Programming a SPOT

The easiest way to begin programming a Sun SPOT is to copy a demo project. The copy will get you the ancillary files that NetBeans and the Ant scripts use and it will make sure you get the right subdirectory structure. Alternately, there is a sample application in
[SunSPOTdirectory]`/Demos/CodeSamples/SunSpotApplicationTemplate`
You can copy that directory to get an extremely simple, “Hello, World” application from which you can work. There is also a `SunSpotHostApplicationTemplate` in the same directory which contains a simple application using the basestation.

Note – If you start with one of these templates and rename the main class of the project, you must update the contents of the `[projectdirectory]/resources/META-INF/MANIFEST.MF` file to match.

Sun SPOT applications are just Java applications, so they aren't hard to program. The main issues for most programmers will be (1) debugging the code and (2) determining how to get access to the peripherals on the demo sensor board. We will discuss both in the sections below.

Debugging on a Sun SPOT

On a Sun SPOT connected directly to a USB, the best way to debug is with print statements. If your SPOT has the Over The Air (OTA) command server enabled and is able to connect to a Sun SPOT basestation, then the best way is to use the NetBeans debugger.

OTA Debugging

There are three steps to doing OTA debugging. The first step enable an OTA link between a Sun SPOT basestation and a free range Sun SPOT. The second is to deploy and run, in debugging mode, the application on the free-range SPOT. The final step is to attach the NetBeans debugger to the application and debug the application.

IEEE Extended MAC Address

OTA communication between Sun SPOT requires the IEEE extended MAC address for all Sun SPOTs involved. The IEEE extended MAC address is a 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 visible through the translucent plastic on the radio antenna fin. A typical sticker would say something like "`0000.0106`" and that would imply an IEEE address for that SPOT of `0014.4F01.0000.0106`.

You can also get the IEEE address for a Sun SPOT using the `ant slots` command. To get the IEEE address this way, connect your Sun SPOT to the USB cable, open a command window on the host workstation, navigate to any Sun SPOT project directory, and execute the command:

```
ant slots
```

You will get output that looks like:

```
[java] [waiting for reset] ** VM stopped: exit code = 0 **
[java] [waiting for reset]

[java] Sun SPOT bootloader (1159-20060804)
[java] SPOT serial number = 0014.4F01.0000.00AE
[java] 0: C:\Program Files\Sun\SunSPOT\Demos\BounceDemo\BounceDemo-
OnSPOT/suite/image (Mon Aug 21 10:56:56 PDT 2006) (28632 bytes) at
0x10140000
[java] 1: C:\Program Files\Sun\SunSPOT\Demos\BounceDemo\BounceDemo-
OnSPOT/suite/image (Mon Aug 21 11:18:15 PDT 2006) (28632 bytes) at
0x101a0000 (current)
[java] OTA Command Server is disabled
[java] Ignoring application suite at startup
[java] Exiting
[delete] Deleting: C:\Program Files\Sun\SunSPOT\sdk\temp\spot-temp-
143556030
BUILD SUCCESSFUL
```

The IEEE extended MAC address is the number that follows "SPOT serial number:" In this case, the MAC address is `0014.4F01.0000.00AE`.

Enable an OTA Link

1. **Enable the OTA command server on the free-range SPOT.**

   The OTA command server is enabled on Sun SPOTs direct from the factory. To test whether the OTA command server is enabled, execute the "`ant slots`" as described directly above. If the output will say either "OTA Command Server is enabled" or "OTA Command Server is disabled". To enable the OTA command server, connect your Sun SPOT to the USB cable, open a command window on the host workstation, navigate to any Sun SPOT project directory, and execute the command:

   ```
   ant enableota
   ```

   If you later decide to disable OTA communication, you can use the command

   ```
   ant disableota
   ```

   You can test whether or not the command worked by repeating the "`ant slots`" command. Disconnect the free-range SPOT from the USB cable.

2. **Enable the SPOT basestation.**

   Connect the basestation SPOT to the USB cable. If you execute an "ant slots" command, you will see a line which either says:

   ```
   [java] Ignoring application suite at startup
   ```

   or

   ```
   [java] Not ignoring application suite at startup
   ```

The “Ignoring” line means that the SPOT is running in basestation mode. The “Not ignoring” line means that the SPOT is not running in basestation mode.

To put the Sun SPOT into basestation mode, enter the command:

```
ant selectbasestation run
```

and the SPOT will be put into basestation mode. You can confirm that it is in basestation mode with the `ant slots` command.

Now the free-range SPOT and the basestation are capable of communicating with each other in debugging mode.

Deploy and Run the Application in Debugging Mode

The next step is to deploy the application code to the free-range Sun SPOT and run it in debug mode. To do this:

1. **Open a command window.**

   Under Windows, this is usually available from Start > All Programs > Accessories > Command Prompt. On a Macintosh or a Linux machine, any command line window will do.

2. **Navigate to the project directory for the application which you wish to debug.**

3. **Deploy the application to the free-range SPOT using the command:**

   `ant -DremoteId=`*`nnnn.nnnn.nnnn.nnnn`* `deploy`

   where “*nnnn.nnnn.nnnn.nnnn*” is the IEEE extended MAC address for the free-range SPOT. Ant command options are case-sensitive. The options “-DremoteID”, “-DRemoteID” and “-DRemoteId” will not work. It must be “-DremoteId”.

4. **Launch the application in debug mode, using the command:**

   `ant -DremoteId=`*`nnnn.nnnn.nnnn.nnnn`*
   `-Dbasestation.addr=`*`mmmm.mmmm.mmmm.mmmm`* `debug-run`

   where “*nnnn.nnnn.nnnn.nnnn*” is the IEEE extended MAC address for the free-range SPOT and ““*mmmm.mmmm.mmmm.mmmm*” is the IEEE extended MAC address for the basestation SPOT.

   The command line output will rapidly scroll through some output, then wait for 30 seconds to a minute at the line “Writing configuration to remote SPOT.” Finally, you should get output that ends with:

```
-do-debug-proxy-run:
[java] Starting hostagent...
[java] My IEEE address is 0000.0000.0000.0001
[java] Done starting hostagent
```

```
[java] Trying to connect to VM on radio://0014.4F01.0000.0106:9
[java] Established connection to VM (handshake took 70ms)
[java] Waiting for connection from debugger on serversocket://:2900
```

Note the socket number indicated in the last line.

Attach to the Debugger

The final step is to attach the NetBeans debugger to the application running on the free-range SPOT.

1. **Within NetBeans, open the project which you wish to debug.**

2. **Ask NetBeans to attach the debugger.**

   You can do this either through the "Attach Debugger" command from the Run menu on the main toolbar, or you can press the Attach Debugger icon in the upper right. It is a blue triangle, pointing to the right, with a small red square just to the left of it.

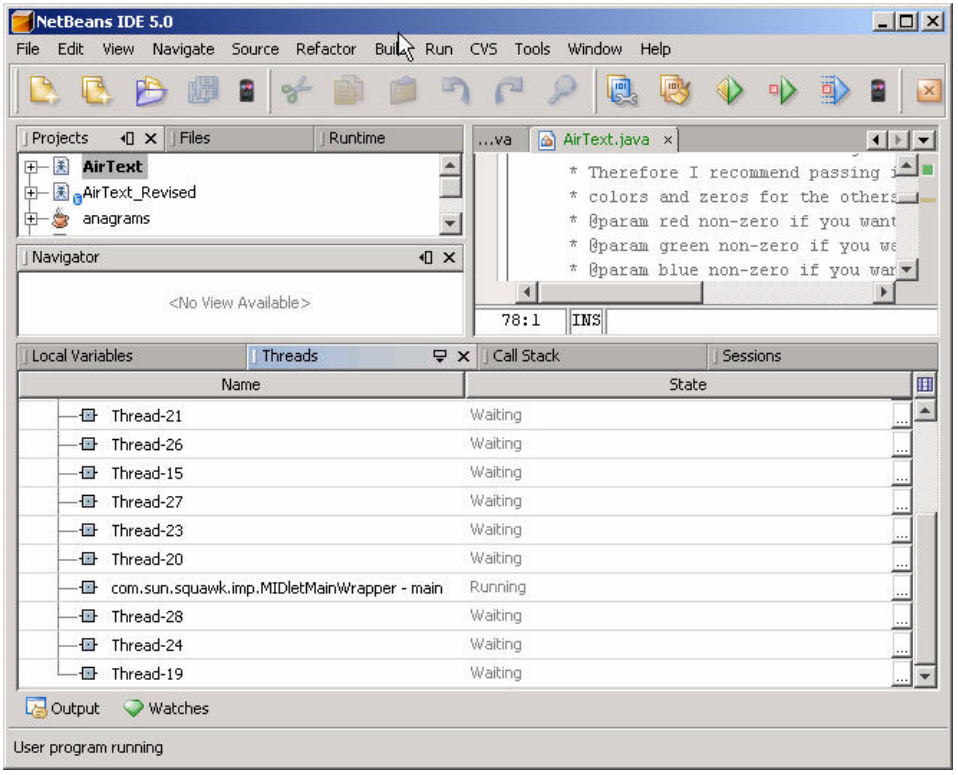
   A dialog box will display.

3. **Specify the debugging attachment.**

   For the connector type, select "SocketAttach". This will give you an opportunity to enter the port number for the socket. Enter the socket number that you got in command line window after doing the `debug-run` command. Click on "OK". After about ten or fifteen seconds, the debugger should attach to the application on your free-range SPOT.

4. **Find the application thread and debug.**

   The debugger will look something like:



   The application thread is the one labeled “MIDletMainWrapper”.

   It is beyond the scope of this document to explain NetBeans and debugging commands, but NetBeans has a good help system and has a typical set of debugging commands.

Print Debugging

For print debugging, load the application to be debugged onto a Sun SPOT which is connected to the host workstation. As you debug, add calls to `System.out.println()`. For example:

```
System.out.println("Got to the Foobar method call");
```

Start the application using either the Run command from within NetBeans or by issuing an “ant run” command at command-line prompt from within the project directory. The debugging output will appear on the host workstation. In NetBeans, it

will appear in an Output panel, normally along the bottom of the Netbeans window. In a command-line window, the debugging output will appear as part of the output from the "ant run" command.

The process that runs Sun SPOT applications from the host produces fair amount of output to the command line or output window. When debugging, be sure to scroll back through that output to see if anything important has been printed but scrolled out of sight.

Accessing the Sensor Board

The sensor board includes a 3D accelerometer, a temperature sensor, a light sensor, eight LEDs, two switches, five general-purpose I/O pints and four high current output pins. In this section, we give a rapid introduction to using these components in a Sun SPOT Java program. For more details, you see the Javadoc pages in the *[SunSPOTdirectory]*`/sdk/doc/javadoc/transducerlib` directory and see the demonstration applications in the `[SunSPOTdirectory]/Demos/CodeSamples` directory. These applications are simple working applications that show the details of using one or two sensor board devices.

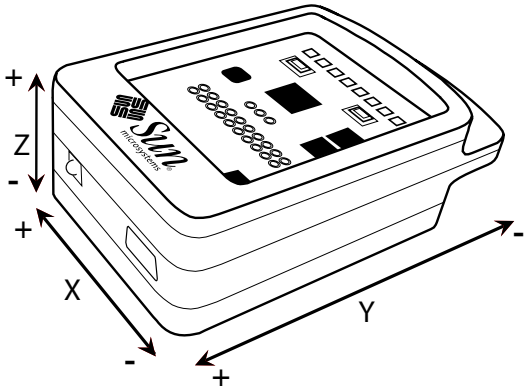
The sensor board devices are also described in the sections below.

Accelerometer

The accelerometer has two modes of operation that determine what range of values it will measure. The ranges are, approximately: -2g to 2g and -6g to 6g. The raw reading from the accelerometer will be a number between 0 and 1023, where 0 indicates the lower end of the range and 1023 represents the higher end of the range. The zero gravity reading will be approximately 466.5.

There are three axes on which the accelerometer measures acceleration. The Z-axis is perpendicular to the Sun SPOT 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.

FIGURE 6 Accelerometer X, Y and Z axes



In FIGURE 6, the plus (+) on the end of an axis indicates that when the device's acceleration vector increases in that direction, the associated accelerometer readings will grow larger.

To use the accelerometer:

1. **Create an accelerometer interface instance:**

```
Import com.sun.spot.sensorboard.EDemoBoard;
Import com.sun.spot.sensorboard.IAccelerometer3d;

IAccelerometer3D ourAccel =
EDemoBoard.getInstance().getAccelerometer();
```

2. **Set the range over which you want the accelerometer to operate. Sending a zero to the method** `IAccelerometer.setRange()` **will set the range to approximately -2g to 2g. A one will set the range to approximately -6g to 6g. After setting the range, the accelerometer must settle for about 100 milliseconds before it will generate accurate readings:**

```
// 0 is the 2g range, 1 is the 6g range
ourAccel.setRange(0);//use 2g scale
// sleep for 100 milliseconds to allow the accelerometer to settle
try{
      Thread.sleep(100);
} catch (InterruptedException ex) {
      // ignore any exceptions
}
```

3. **Read from the accelerometer to get the raw accelerometer reading:**

```
int x-accel = ourAccel.getXaxis().getValue();
int y-accel = ourAccel.getYaxis().getValue();
int z-accel = ourAccel.getZaxis().getValue();
```

The raw reading may be enough for some applications. The formula for conversion of a raw reading on the 2g scale into g-forces is:

$$gForce = \frac{rawReading - 465.5}{186.2}$$

The formula for conversion of a raw reading on the 6g scale into g-forces is:

$$gForce = \frac{rawReading - 465.5}{62}$$

The 465.5 is, in each case, the estimate of the raw reading with the sensor for that axis in a zero-gravity position. Experimentation with particular units may give you a more accurate estimate for that unit.

Anytime that a Sun SPOT is at rest in the earth's gravitational field, the accelerometer will register one gravity of acceleration downward. We can use that fact to calibrate the raw readings from any particular Sun SPOT's accelerometer. If you wish to get a precise value for the zero gravity reading for a particular axis, you can orient the Sun SPOT so that axis is perpendicular to the earth's gravity. For example, if the Sun SPOT rests a table as show in FIGURE 6, the X- and Y-axes will be in be giving their zero gravity reading. The Z-axis sensor will be giving a one-gravity reading.

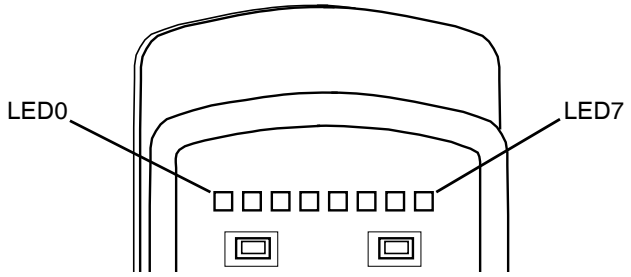
In the formulas given above, the 186.2 (in the 2g formula) and the 62 (in the 6g formula) are the estimated amounts that the raw readings will change for each one g in force applied to the Sun SPOT. If you need more precise values, you can take determine a zero gravity reading as above, then take both one gravity readings for the axis and average their deviations from the zero gravity reading.

For example, if Z1 was the raw reading along the Z-axis with the SPOT in the position shown in FIGURE 6, and Z2 was the raw reading along the Z-axis with the SPOT turned upside-down from that position, then the constant to use in the conversion formula for that scale would be $\left|\frac{Z1 - Z2}{2}\right|$

LEDs

There are eight three-color LEDs on the demo sensor board, in a row, with LED0 on the left and LED7 on the right.

FIGURE 7 LED Layout



Each LED has a red, a green, and a blue emitter as part of the LED. Each individual color can have an intensity from 0 to 255, with 0 being off and 255 being as bright as possible.

To use the LEDs:

1. **Instantiate the LED object array.**

```
Import com.sun.spot.sensorboard.EDemoBoard;
Import com.sun.spot.sensorboard.ITriColorLED;
ITriColorLED[] ourLEDs = EDemoBoard.getInstance().getLEDs();
```

2. **Set the LED color desired.**

   Colors are specified with the `setRGB(int red, int green, int blue)` method.

```
// set the LED color desired
// set the first two LEDs to bright red, the next two to bright green,
// the next two to bright blue, and the last two to white.

// First two = bright red
ourLEDs[0].setRGB(255,0,0); ourLEDs[1].setRGB(255,0,0);

// Next two = bright green
ourLEDs[2].setRGB(0,255,0); ourLEDs[3].setRGB(0,255,0);

// Next two = bright blue
ourLEDs[4].setRBG(0,0,255); ourLEDs[5].setRGB(0,0,255);

// Last two = white
ourLEDs[6].setRGB(255,255,255); ourLEDs[7].setRGB(255,255,255);
```

3. **Turn the LEDs on.**

   //turn the LEDs on

```
for (int i = 0; i < 8; i++){
```

```
      ourLEDs[i].setOn()
}
```

4. **If desired, turn the LEDs off.**

```
// turn the LEDs off
for (int i = 0; i < 8; i++){
      ourLEDs[i].setOff()
}
```

You can also query the state of the LEDs using the `isOn()`, `getRed()`, `getGreen()`, and `getBlue()` methods.

Switches

The sensor board has two switches on it. These are represented in the `EDemoBoard` object as an array of type `ISwitch`. You may query the state of the switches using the `isOpen()` and `isClosed()` methods. 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 Sun SPOT to stop and wait for the switch state to change. When you want the SPOT to wait for the state switch to change, you would use the `waitForChange()` method.

1. **Instantiate the switch array.**

```
Import com.sun.spot.sensorboard.EDemoBoard;
Import com.sun.spot.sensorboard.ISwitch;
ISwitch[] ourSwitches = EDemoBoard.getInstance().getSwitches();
```

2. **Look for a switch press.**

If you wanted a switch press and were willing to wait for it:

```
if(ourSwitches[0].isOpen()){
// if it is open, wait for it to close
      ourSwitches[0].waitForChange();
}
// Whether it was closed before or just closed, wait for it to open
ourSwitches[0].waitForChange();
```

Light Sensor

The light sensor returns an integer that ranges from 0 to 1023. Zero represents complete darkness. Peak sensitivity of light sensor is at 600nm wavelength. An illustration of how the raw readings map to luminance values is given in the table below:

TABLE 5 Specified typical values for light in luminance (lx) and light sensor readings

Luminance	Raw Reading
1000 lx	497
100 lx	50
10 lx	5

To use the light sensor:

1. **Instantiate a light sensor object.**

```
Import com.sun.spot.sensorboard.EDemoBoard;
Import com.sun.spot.sensorboard.peripheral.ILightSensor;

ILightSensor ourLightSensor =
EDemoBoard.getInstance().getLightSensor();
```

2. **Get the light sensor raw reading.**

```
int lightSensorReading = ourLightSensor.getValue();
```

Temperature Sensor

The temperature sensor is the simplest of the sensors. There are no raw readings and no parameters to set. However, it is, inevitably, close to some heat sources in the Sun SPOT. More accurate temperature readings could be obtained with an external temperature sensor tied to the I/O pins on the sensor board.

1. **Instantiate the temperature sensor object.**

```
Import com.sun.spot.sensorboard.EDemoBoard;
Import com.sun.spot.sensorboard.io.ITemperatureInput;

ITemperatureInput ourTempSensor = EDemoBoard.getADCTemperature();
```

2. **Read the temperature.**

```
// The temperature can be read in Celsius

double celsiusTemp = ourTempSensor.getCelsius();

// or in Fahrenheit

double fahrenheitTemp= ourTempSensor.getFahrenheit();
```

The Ectoplasmic Bouncing Ball Demo

Before the Demo

Your Sun SPOT kit should have come with two free-range Sun SPOT units and one basestation unit. The basestation unit is thinner and does not have a battery board.

If you want to use the host workstation and the basestation unit in the demo, you must first install the Sun SPOT development software on the host workstation, then attach the basestation to the host workstation.

If you do not want to use the host workstation and the basestation in the demo, you can still run the demo on the free-range Sun SPOT units alone.

Starting the Demo

To start the demo, turn the free-range Sun SPOT units on. The power switch is located on one of the narrow ends of the Sun SPOT unit. If the free-range Sun SPOT units have a charged battery, they each will go through a boot process lasting two or three seconds. After they have booted, each Sun SPOT unit will start to run the demo.

If the free-range Sun SPOT unit does not boot, it probably needs to charge its battery. To charge a Sun SPOT unit battery, attach the unit, using the supplied USB cable, to the USB port on a working computer. The USB power will charge the Sun SPOT unit in approximately three hours.

To start the demonstration on the host workstation and the basestation Sun SPOT unit, open NetBeans and select the BounceDemo from the list of projects on the left. Select “Host Run” from the menu bar. A window will open containing an image of a Sun SPOT unit. This Sun SPOT unit can participate in the demo in the same way as a physical Sun SPOT unit. Your mouse can be used to manipulate this soft Sun SPOT unit. The basestation Sun SPOT unit does not participate in the demo except to pass radio packets between the free-range Sun SPOT units and the soft Sun SPOT unit running (virtually) on the host workstation.

The Effect

The row of LEDs on the top board of each Sun SPOT unit represents a tube. The red LEDs at each end of the LED row represents a cork in the tube. Any other LED which is lit represents an ectoplasmic ball. At first the balls on any Sun SPOT unit

will be blue. However, if the Sun SPOT units are able to communicate with each other, they will allocate colors for the ectoplasmic balls to avoid, as much as possible, duplication.

Pick up a Sun SPOT unit and tilt it. Note how the red corks keep the ectoplasmic ball from escaping the tube.

If you press the switch next to the LED that represents a cork, that cork will partially open. To reach the switch, you will need to open the lid of the plastic case. To open the lid, press *firmly* on the lid, down and back, on the edge of the lid near the small raised dot. You can think of that small-raised dot as the fingernail-catching dot. The closer to the edge that you press, the easier the lid will open. The opposite end of the lid will pop up. Grasp that opposite end to remove the Sun SPOT unit's lid.

See the illustration below:

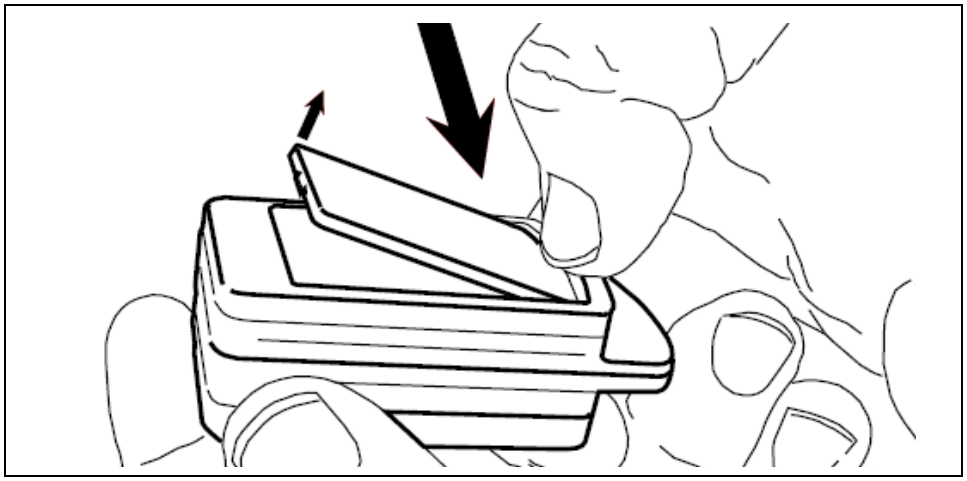
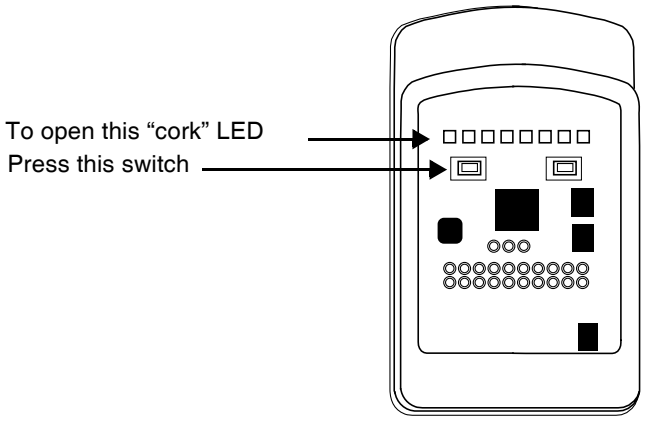


FIGURE 8 Opening a Sun SPOT Unit Lid

Once you have opened the Sun SPOT lid, press the switch nearest a cork LED to open it.



We don’t want to lose any of the precious ectoplasm, though, so the cork won’t fully open until there is another Sun SPOT unit, within range, with a cork that is also half-open. If two Sun SPOT unit with half-open corks find each other, both corks will fully open, and you can pour ectoplasmic balls from one Sun SPOT unit to another.

A cork that is closed displays as a steady red LED. A cork that is half-open displays as a blinking red LED. A cork that is fully open does not show at all. To close a cork that is open, press the switch closest to it.

The ectoplasm is sticky. See if you can get two ectoplasmic balls to come to rest in the same place. They will then merge into one ball. The color of the new ball will be the merged color of the original balls. For example, a red ball and a blue ball will merge to form a purple ball.

To restart the demo, press the control switch momentarily. The Sun SPOT unit will reboot and restart the demo.

The Implementation

The Sun SPOT units use radio communication at the beginning of the demonstration to choose colors for each ectoplasmic ball. They poll the 3-D accelerometer to determine the orientation of the Sun SPOT unit. They use the orientation information to determine the movement of the ectoplasmic balls. The switches are also polled to determine the appropriate cork state. Radio communication is used to determine if there is another Sun SPOT unit within range and if there is an open cork on that unit. If there is, radio communication is used to pass the ball back and forth.

Source code for the demonstration is in the Sun SPOT SDK directory at the location: *[SunSPOTdirectory]*`/Demos/BounceDemo/BounceDemo-OnSPOT`.

Troubleshooting

All Platforms

Problem: But I don't want to use NetBeans!

Solution: One of the things that NetBeans does is set some needed environmental variables. If you don't use NetBeans, do the following:

- Modify your `PATH` to include the `bin` subdirectory for your Java Development Kit (JDK).
- Modify your `PATH` to include `bin` subdirectory for the Ant directory.
- Set JVMDLL to the location of your Java virtual machine.
- Set JAVA_HOME to the location of your top level Java SDK directory

NetBeans also provides a user interface for the several Ant commands used to deploy and run code on the Sun SPOTs from the host workstation. Read through the Sun SPOT Developer's Guide and make sure you understand how to use the Ant commands described there.

Problem: You get the following error message any time you attempt to communicate with a Sun SPOT over the USB cable, including any of the many `ant` commands:

```
[java] WARNING:  RXTX Version mismatch
[java] Jar version = RXTX-2.1-7
[java] native lib Version = RXTX-2.1-7pre17
```

Solution: The host workstation uses a serial communication package called RXTX to communicate with the Sun SPOTs. The version in use on the host workstation and on the Sun SPOT must match. The correct version of the RXTX library is installed on the host workstation when the SDK is installed, but if another version is on the load path, a mismatch can occur. Look for an old version of the RXTX library somewhere in your load path. The name of the file will vary with the operating system of the host workstation:

TABLE 6 RXTX File Names on Different Operating Systems

Operating System	RXTX File Name
Windows	`rxtxSerial.dll`
Macintosh	`librxtxSerial.jnilib`
Linux	`librxtxSerial.so`

Change your PATH variable to avoid the other version of RXTX or remove the excess version of RXTX.

Problem: My Sun SPOT has got into a state where it continually restarts and I get an error whenever I try to deploy to it. How can I recover?

Solution: First you need to get the SPOT into a state where it is listening to commands from the host rather than continually restarting. To do that, follow this procedure:

- Disconnect the Sun SPOT from the USB cable
- Kill all the ant and java processes listening on the port
- Hold the control button in for a few seconds until a double red flash indicates that the Sun SPOT has powered down
- Type this command at a command prompt:

  ```
  ant -Dport=COMnn slots
  ```

  substituting the correct communication port/device name for COM*nn*.

If you don't know the correct port name just enter a dummy value, e.g. "X", and complete the procedure. The output should list the correct port name. Then repeat the procedure with the correct port.

As soon as the ant script starts to complain that the port isn't available, saying something like:

```
[java] Port COM31 unavailable...
[java] Available ports: COM1 COM2 COM3 LPT1
[java] retrying...
```

immediately plug in the Sun SPOT and immediately press the control button.

The `ant slots` command should now operate correctly. Once it has finished, you can take the necessary recovery action, normally to reinstall the system software using `ant upgrade` and then redeploying your (possibly corrected) application.

Problem: When I try to create a suite (via "`ant suite`" or "`ant deploy`") I get the following error:

```
Unable to locate tools.jar Expected to find it in C:\...
```

What should I do?

Solution: Most likely your system environment variables are not setup correctly. Please make sure your PATH variable has your JDK directory as its FIRST value. This also ensures that Windows is not using the `java.exe` commonly located in `C:\Windows\System32`. Also ensure that JAVA_HOME is set correctly.

Problem: How do I modify the `sunspot.home` property for the `ant` build system?

Solution: In Windows, you need to modify the file `.sunspot.properties` located in: `C:\documents and settings\(your account)`

Problem: Is there an API to query the id of a SPOT at run-time?

Solution: `System.getProperty("IEEE_ADDRESS").`

Problem: Can you tell me the default wait time for an `ACK` before the `NoAckException` is thrown?

Solution: Currently set to 11 attempts with a timeout of 10mSec per attempt, so about 110ms plus the 11 pre-transmit backoff times (they are normally short).

Problem: How can my application determine how much memory is left?

Solution: Runtime.getRuntime().freeMemory().

Linux

Problem (Linux): When trying to deploy an application to a Sun SPOT I an get error messages from RXTX, saying that the device is not available, or that it doesn't have permission to access it or the lock file.

Solution (Linux): Make sure that you followed the instructions in the section "Adjust Permissions (Linux)" on page 14 of the *Installation Instructions*. In particular don't forget to logout after performing the changes.

If you still cannot deploy to the Sun SPOT, make sure that the `/var/lock` directory exists, and that it is has read, write, and execute permissions for the group to which you added the user. If it doesn't exist, you should create it. The recommended permissions are `755`, and the group should be `lock`. Also check the permissions and group of the device file. The name of the device should be given in the error message; common names are `/dev/ttyACMx` or `/dev/usbmodem`.

If this doesn't fix the problem, verify that there is no link to `/var/lock` or `/var/spool/lock` as this may cause RXTX to detect the lock file twice and to fail.

Problem (Linux): When trying to deploy an application to a Sun SPOT I get an error message: "No Sun SPOT devices found. ...." or similar.

Solution (Linux): Besides the obvious things, like making sure that a Sun SPOT is actually connected to the USB port, it may be that your Linux installations doesn't have the `cdc_acm` driver which is required to access the Sun SPOT. Try calling "`dmesg | grep usb`" in a command window. There should be a message that looks something like "`usb ...: New full speed USB using ...`". Furthermore there should also be a message referring to `cdc_acm`. If you see the general messages but you do not see any `cdc_acm` messages, you probably have to install the `cdc_acm` driver onto your system.

If you see the `cdc_acm` messages, and still get a "`No Sun SPOT devices found...`" error, it may be that the spot-finder script has failed to detect the device. This may occur if the `hal_device` command is not available and the device gets assigned a name not matching the expected `/dev/ttyACM*` pattern. We recommend you install `hal_device` on your system in this case. Alternatively, you may specify the appropriate device name manually in the `ant port` property. You define this property in the build.properties for a project.

Problem (Linux): When trying to deploy an application to a Sun SPOT, I get an error message:

```
Port  Error: An SELinux policy prevents this sender from sending this
message to this recipient unavailable..."
```

or something similar.

Solution (Linux): This means that the SELinux policy is too restrictive and is not allowing access to the serial device. This is, for example, the case in the default settings for Fedora Core 5.

If you don't require the additional security, SELinux allows you to either disable it or set it to a less restrictive setting. For example, in Fedora Core 5, changing the SELinux setting to permissive solves this problem. You can change the SELinux setting in the `System/Administration/Security Level and Firewall` menu. If you don't want to give up this security level, you need to define a SELinux policy which gives RXTX permission to the device.

Battery Warnings

Do not short-circuit battery. A short-circuit may cause fire, explosion, and/or severe damage to the battery.

Do not drop, hit or otherwise abuse the battery as this may result in the exposure of the cell contents, which are corrosive.

Do not expose the battery to moisture or rain. Keep battery away from fire or other sources of extreme heat. Do not incinerate.

Exposure of battery to extreme heat may result in an explosion.

No other battery substitutions or different chemistry batteries should be used.

Do not bypass the battery protection circuit.

Dispose of batteries properly. Do NOT throw these batteries in the trash. Recycle your batteries, if possible.

Federal Communications Commission Compliance

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try and correct the interference by one or more of the following measures: Reorient or locate the receiving antenna. Increase the separation between the equipment and receiver. Connect the equipment into an outlet on a circuit different from that to which the receiver is connected. Consult the dealer or an experienced radio/TV technician for help.

The Sun SPOTs are supplied with a shielded USB cable. Operation with a non-shielded cable could cause the Sun SPOTs to not be in compliance with the FCC approval for this equipment. The antenna used with this transmitter must not be co-located or operated in conjunction with any other antenna or transmitter; to do so could cause the Sun SPOTs to not be in compliance with the FCC approval for this equipment. Any modifications to the Sun SPOTs themselves, unless expressly approved, could void your authority to operate this equipment.

FCC Declaration of Compliance:

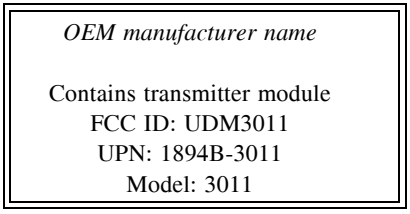
Responsible Party:
Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
Phone: US 1-800-555-9786; International 1-650-960-1300

Products:
SLS-E5-XXXX
where “X” is any alphanumeric character or a blank.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following conditions: this device may not cause harmful interference and this device must accept any interference received, including interference that may cause undesired operation.

This device can be used as is (stand-alone) or as a module (part of a final host product). If the device will be used a module these rules must be followed:

1. **Integrator must place a label outside their product similar to the example shown**



2. **Caution: Exposure to Radio Frequency Radiation.**

   To comply with FCC RF exposure compliance requirements, a separation distance of at least 20 cm must be maintained between the antenna of this device and all persons. This device must not be co-located or operating in conjunction with any other antenna or transmitter.

   Module 3011 and antenna tested with must be integrated in the end product in such a way that the end user cannot access the either the module, cables or antennas.

   The installer of this radio equipment must ensure that the antenna is located or pointed such that it does not emit RF field in excess of Health Canada limits for the general population; consult Safety Code 6, obtainable from Health Canada's website `www.hc-sc.gc.ca/rpb`.