What will you open?

The Sun Small Programmable Object Technology (Sun SPOT): Java(tm) Technology-Based Wireless Sensor Networks

Angela Caicedo
Technology Evangelist
Sun Microsystems

solaris Java

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference

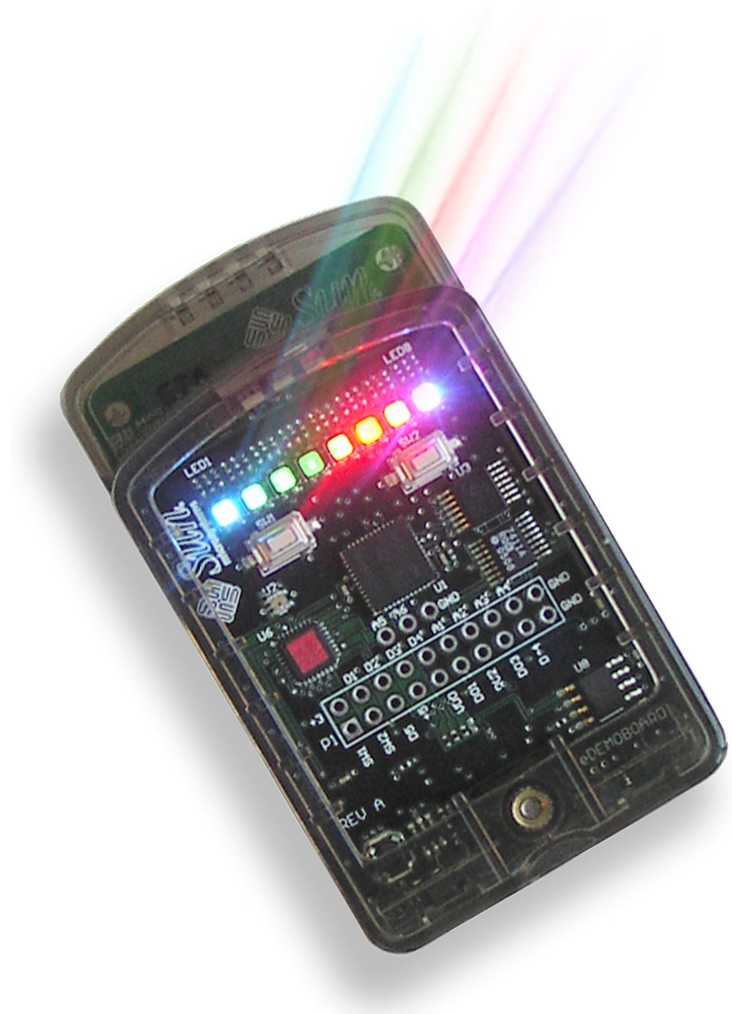
Agenda

- Introduction
- Sun Small Programmable Object Technology
- The Squawk Java VM
- Java Programming for Sun SPOTs
- Sun SPOTs Communication
- Conclusion and Resources

Project Sun SPOT

- Sun has licensed Java on over 1 billion cell phones

- How do we encourage Sun technology in whatever comes next?



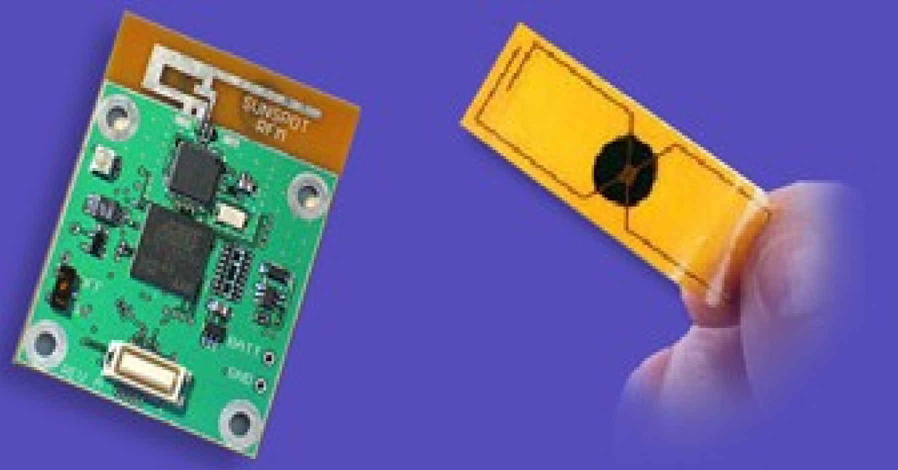
RFID Tags and Sensors

Traditional
Network
Devices

17 Billion

"Internet of Things"

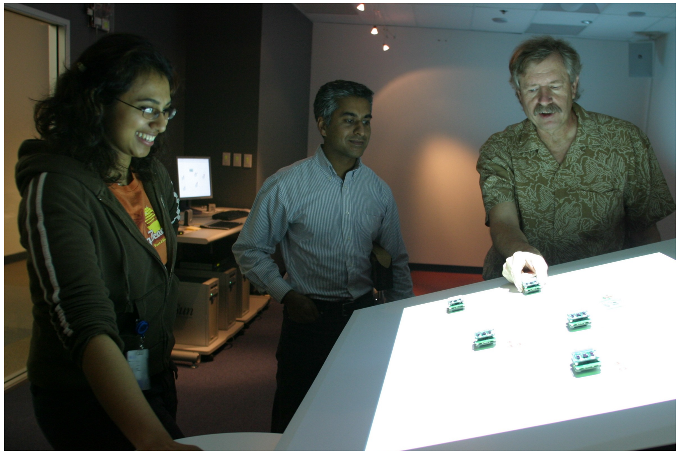
1 Trillion!



Source: IDC Estimates, 2004

Sun SPOT project

- Sun SPOT project started Nov 2004
  - > Follow-on to Epsilon and Anteater projects
  - > Wireless Sensor Networks are a hot research topic
  - > Found difficult tools and limited hardware



- How can we accelerate the development of the internet of things?
  - > Need new tools - HW & SW
  - > Need to inspire new developers

Sun’s Opportunities

- Strengths
  - > Operating Environment - *Squawk VM/Java*
  - > Development/Deployment Tools - *Net Beans, SPOTWorld*
  - > Security/DRM - *Sizzle, OpenMediaCommons.org*
  - > Scalability/Back-end support
- Other Differentiators
  - > Local Processing - *32-bit processing*
  - > Actuation/Control - *robotics, toys, etc*
  - > Platform for experimentation/inspiration - *don’t optimize prematurely, design for flexibility*

The Need for Better Sensor Networks

- 40% of energy costs in an office building is lighting
- U.S. movie theaters
  - > Some have energy costs >$400 per day
  - > Can vary by a factor of 10
- Sensor market in 2001 was ~$11 Billion*
  - > Wiring installation costs > $100 Billion
- Wireless sensor market in 2010 of $7 Billion†
- 1.5 Billion transducer devices installed by 2010‡

* Freedonia Group report on sensors, 2002 † ON World Report ‡Harbour research report

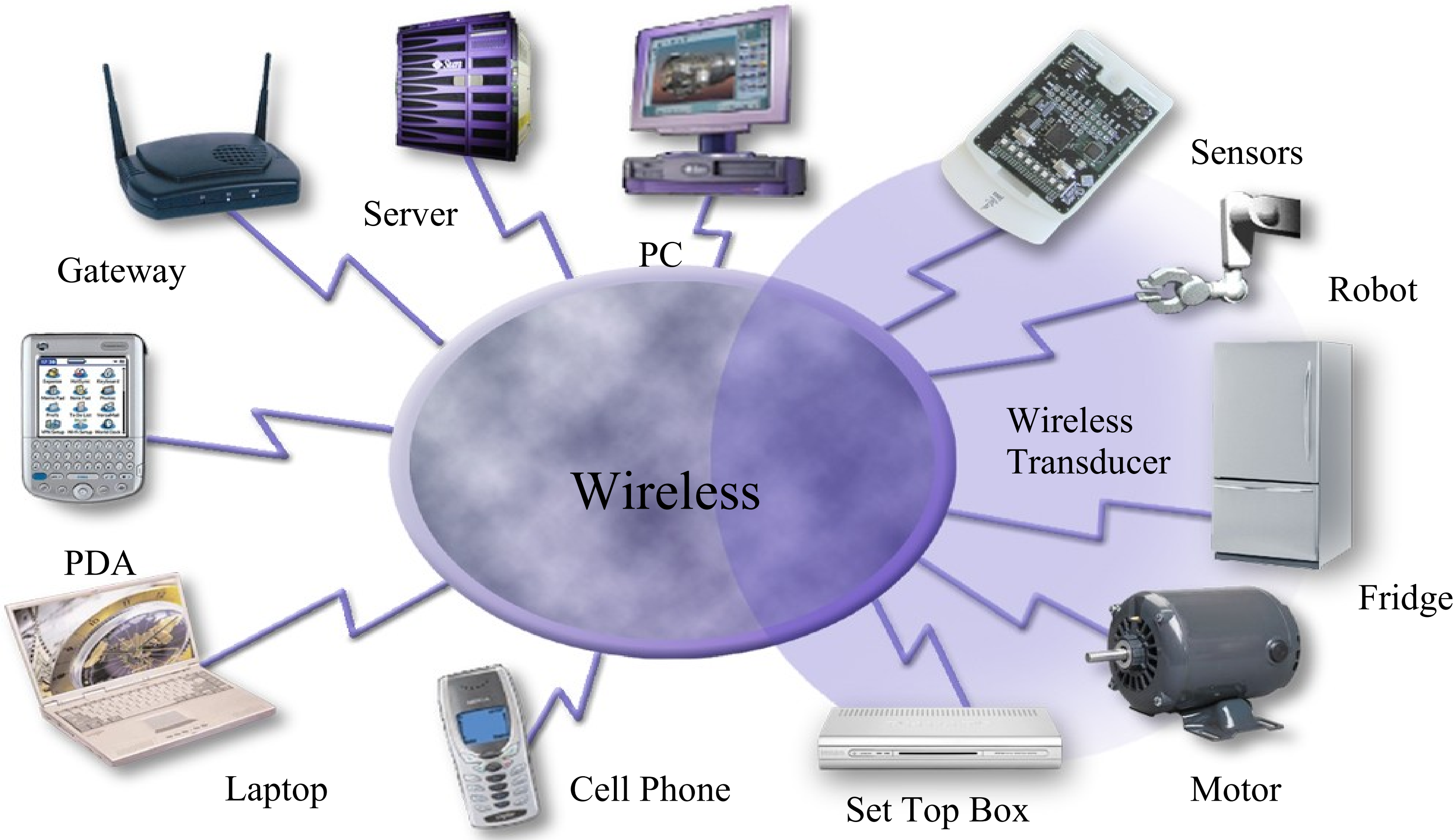
The State of the Art

- Ideas of “Smart Dust”
  - Berkeley, Kris Pister, 1998–2001
- Berkeley motes, TinyOS
  - Mica2, Mica2Dot: 8-bit microcontroller, 7.37/4.0 Mhz clock, 128 KB flash, 4 KB SRAM, CC1000, 512 KB external flash, 2 AA batteries/3V lithium cell battery
- Intel Mote
  - Zeevo module (ARM7 core, SRAM and flash memory, Bluetooth wireless), TinyOS
  - Mote 2: 32-bit Xscale PXA 271 CPU, large RAM and flash memories, runs Linux and the Java VM

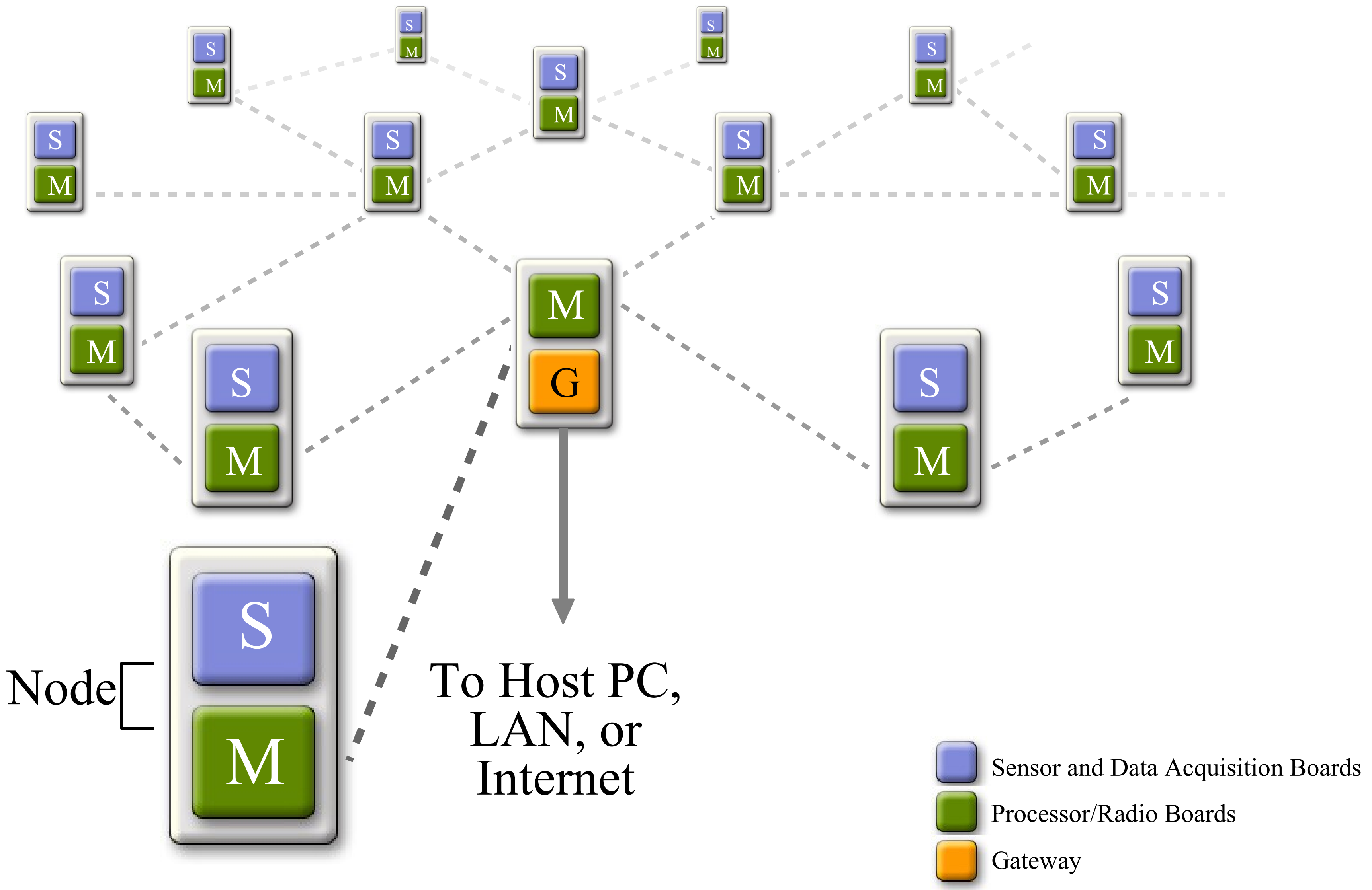
Applications: Chicken and Egg Problem

- Hard to develop applications using current technologies
- Low-level C-like languages
- Unproductive development tools
  - > Hardly any debugging support
- Too many low-level concerns in current systems
  - > Most high-level software developers do not know how hardware works, or even have an appreciation any more
- Not accessible to majority of software developers

The Future: Connected Wireless Networks



Wireless Sensor Networks



Target Audience

- Education
  - > Classroom teaching tool for everything from embedded systems to robotics to design classes
- Research
  - > Flexible, easy to deploy platform for wide range of research from wireless networks to gesture interfaces to new security devices
- Hobbyist
  - > Powerful platform - easy to program and easy to interface

Education

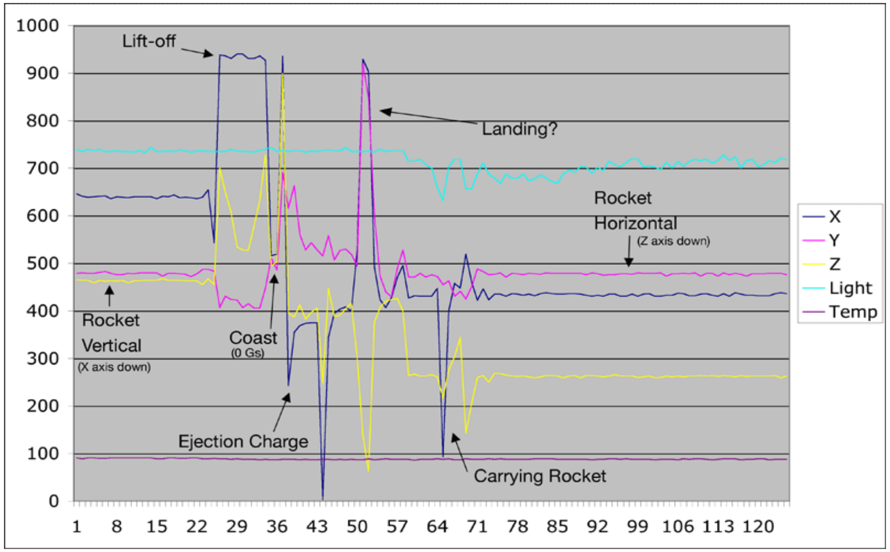
- Essex University
  - Pervasive Computing and Ambient Intelligence
  - Used to teach embedded development
  - Sun SPOTs popular among students and teachers
- Art Center College of Design
  - Designer (not engineers)
  - Single programmer supported projects for ~20 student class
  - Apps ranged from autonomous blimps to consumer music devices

Industrial Research

- Volkswagen Passat Showcar
  - > Quickly built working demo of home security check integra-ted with existing in-car equipment and in-dash display
- Defense Customer
  - > Developed tamper resistant device for authenticating users
  - > Used Sun SPOTs to sense orientation of drum to allow a user to enter a PIN
  - > User is authenticated securely over the radio

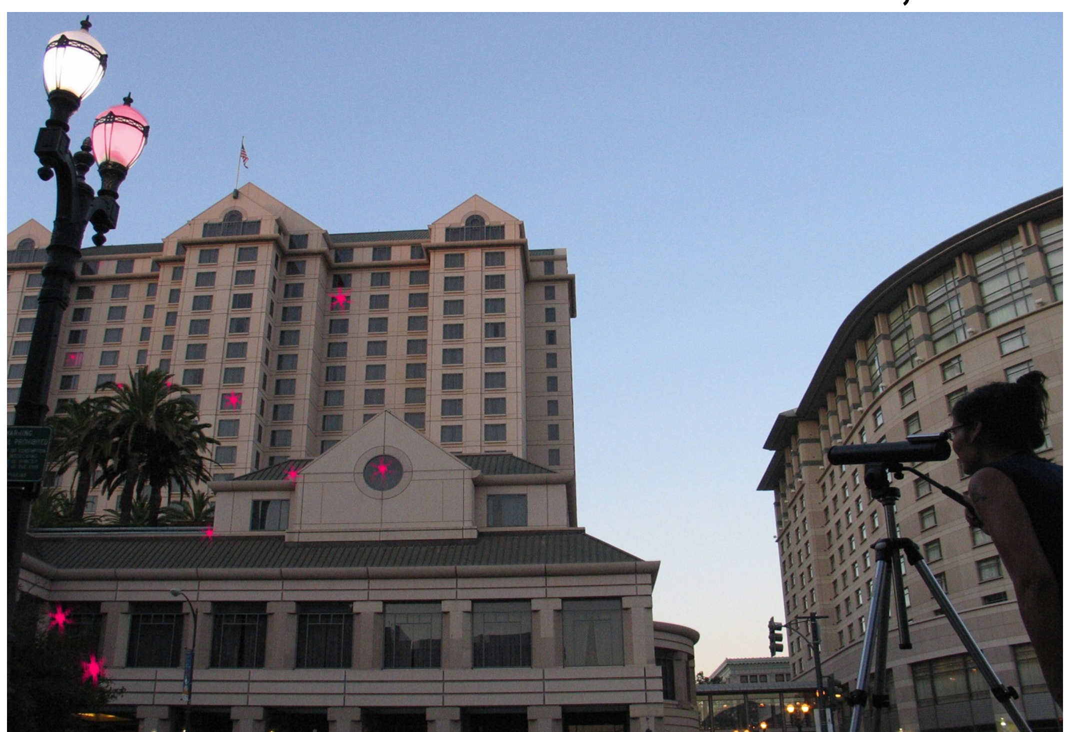
Hobby

- Rocket Science
  - > One day project start to finish
  - > Built rocket and software and launched rocket in the afternoon
  - > Rocket had two Sun SPOTs for redundancy
  - > Data was streamed live to laptops in the parking lot
  - > Recorded 3D acceleration, light and temperature



SPOTs at the ZeroOne San Jose - CA

- Accessed via two telescope-like interface devices situated at ground level in the park.
- Use infra-red beam illuminates or "turns on" embedded elements along the viewer's line of sight.
- For example, lights turn on in rooms at the Fairmont, over 200 feet away.



What will you open?

Sun Small Programmable Object Technology (Sun SPOT)

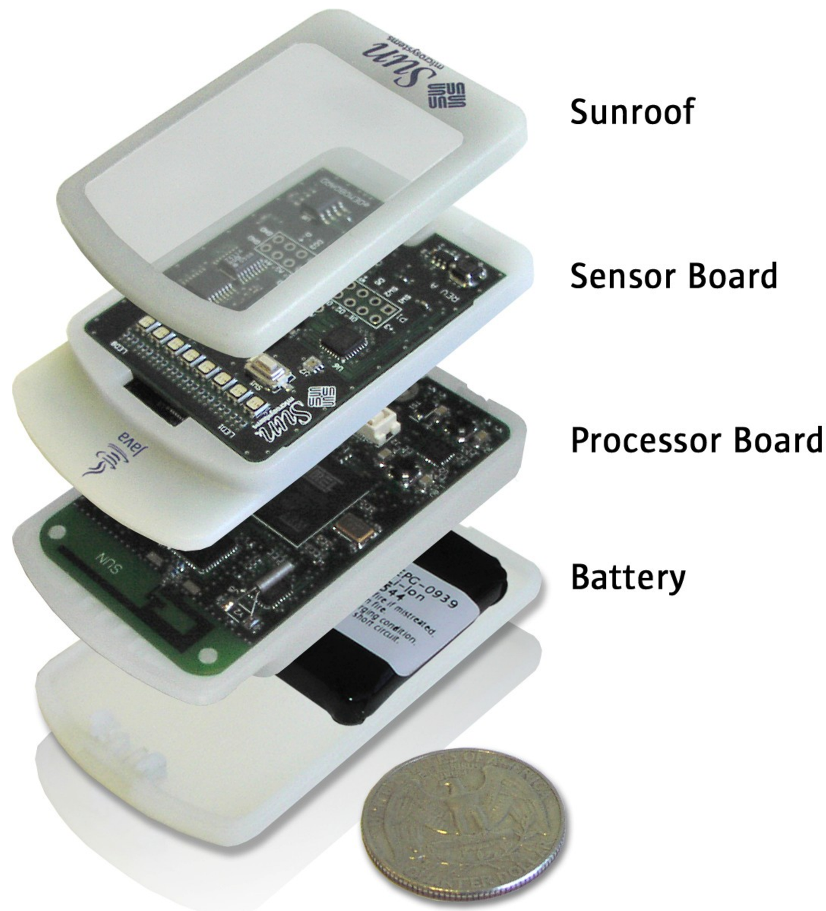
Sun SPOTs

- A Java Platform for Developing Applications for Wireless Networks of Small Devices
- More than just sensors:
  - Robotics
  - Art
  - Toys
  - Personal Electronics
  - Commercial Applications
- Program the world!

Sun SPOT Device

- Basic device has three layers
  - > Battery
  - > Processor Board with Radio
  - > Sensor Board (application specific)

- Processor Board alone acts as a base-station
- User programs the device entirely in Java using standard Java tools



Sun SPOT Hardware

- 180 Mhz 32-bit ARM920T core
  - 512K RAM/4M ROM
- ChipCon 2420 radio
  - 2.4 GHz IEEE 802.15.4
- USB interface
- 3.7V rechargeable 750 mAh prismatic lithium ion battery
- 40 uA deep sleep mode, 40 mA to 100+ mA
- 64 mm x 38 mm
- Double sided connector for stackable boards

Demo Sensor Board

- 8 tri-color LEDs
- 3D accelerometer
- 5 general purpose I/O pins
- 4 hi current output pins
- 1 A/D converter
- Temperature sensor
- Light sensor

The Sun SPOT SDK—Libraries

- Squawk Java VM: desktop and Sun SPOT
- Libraries
  - > Java ME CLDC 1.1 libraries
  - > Hardware libraries
    - > SPI, AIC, TC, PIO drivers all written in the Java programming language
    - > Demo sensor board library
  - > Radio libraries
  - > Network libraries
    - > 802.15.4 MAC layer written in the Java programming language, uses GCF
  - > Desktop libraries

The Sun SPOT SDK—Tools

- DebugClient
- ant tasks
- IDE integrations
- Sample NetBeans™ based projects
- SPOTWorld

Sun SPOT developer's kit

- Two Full Sun SPOTs with eDemoSensor boards and batteries
- One base-station Sun SPOT
- Software
  - > Squawk VM
  - > Java SDK
  - > Netbeans
- USB cable
- Mounting clips
  - > Two wall mounts
  - > One PC board mount
- $499 introductory price
  - > Available Q4 2006

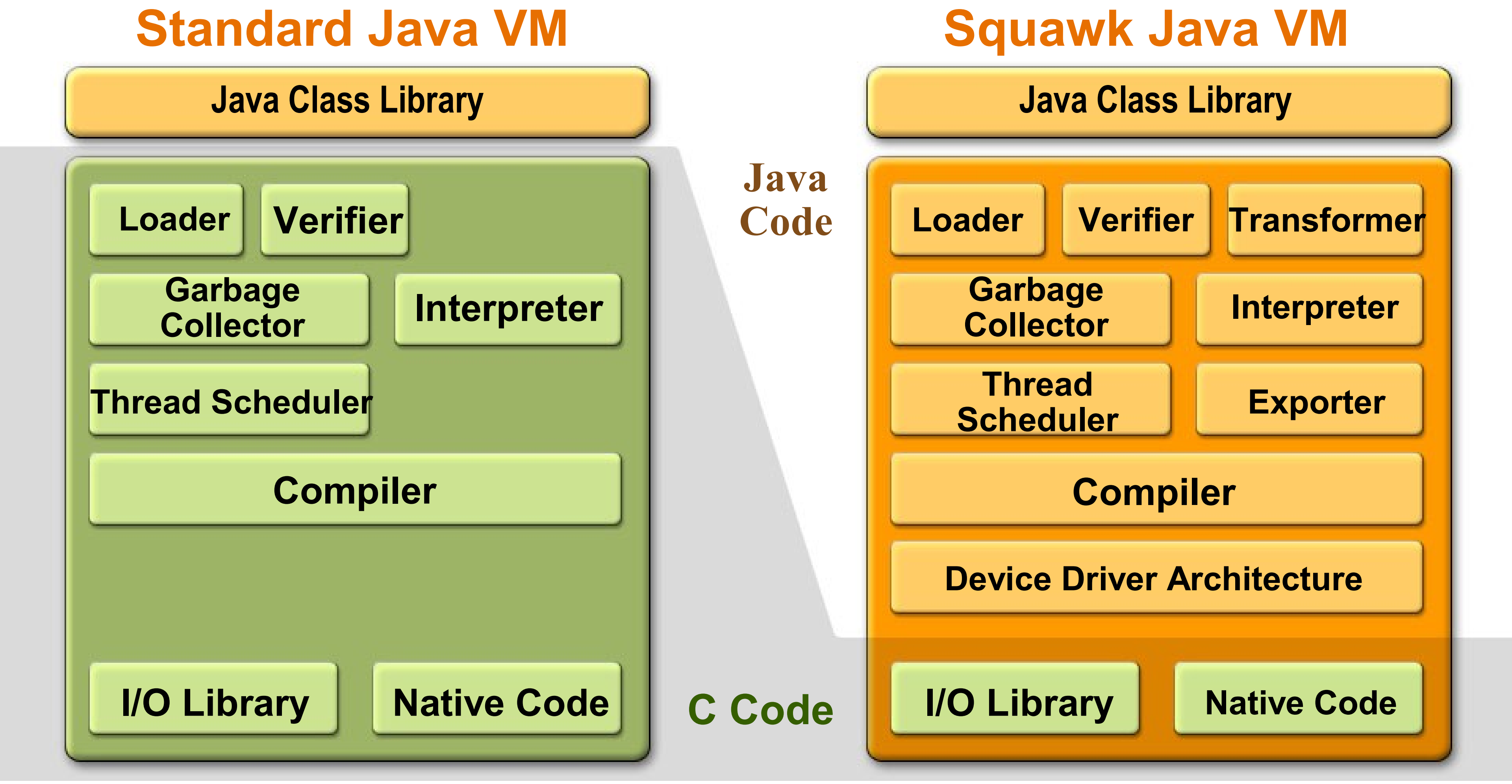
The Squawk Java VM

What will you open?

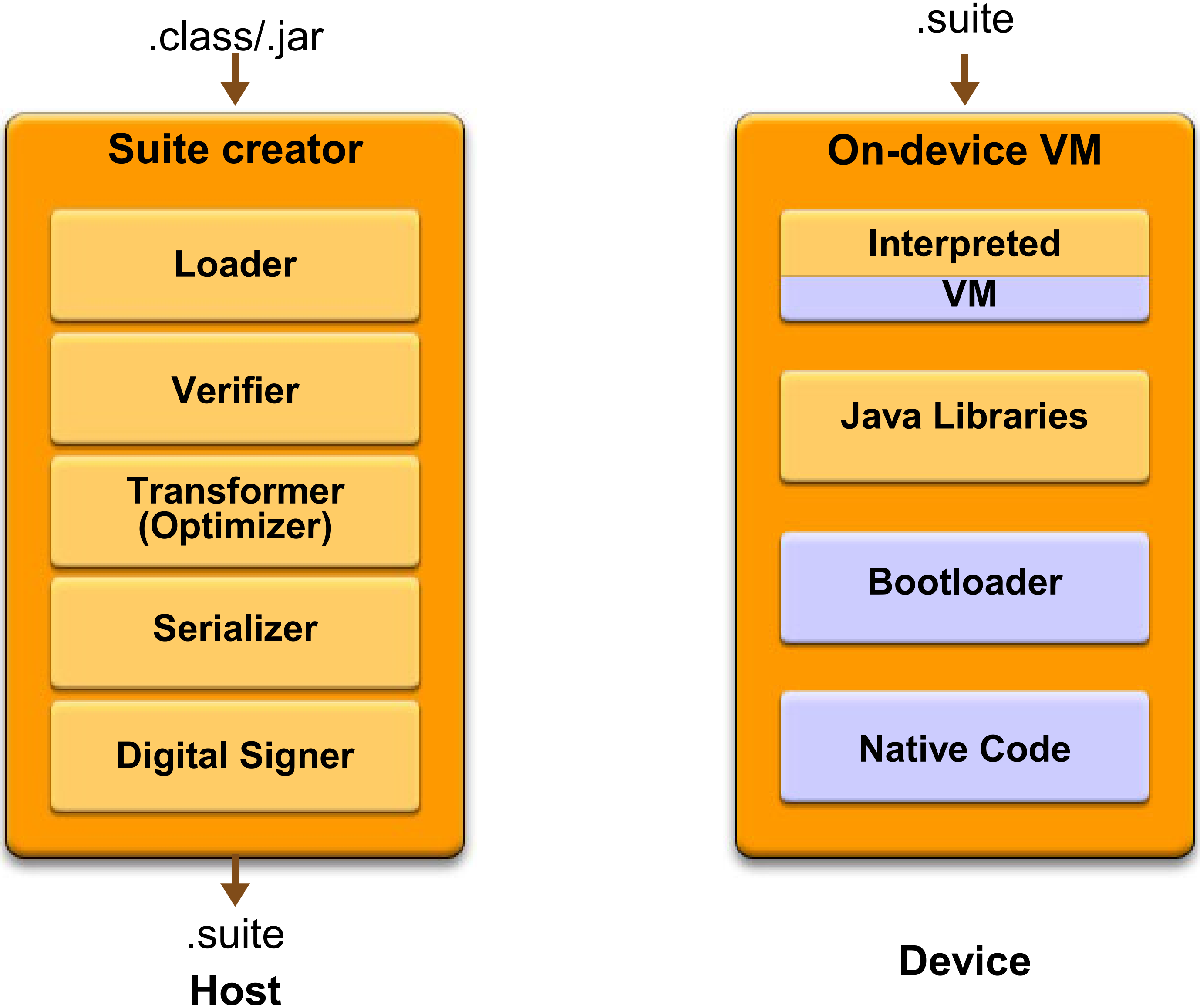
The Squawk Java VM

- Java VM mainly written in the Java programming language
  - > Interpreter written in C
  - > Garbage collector translated from the Java to the C programming language
- Java ME CLDC 1.1
- Extra features
  - > Runs on the bare ARM without an underlying OS
  - > Interrupts and device drivers written in the Java programming language
  - > Supports isolate application model

Standard Java VM Vs. Squawk Java VM



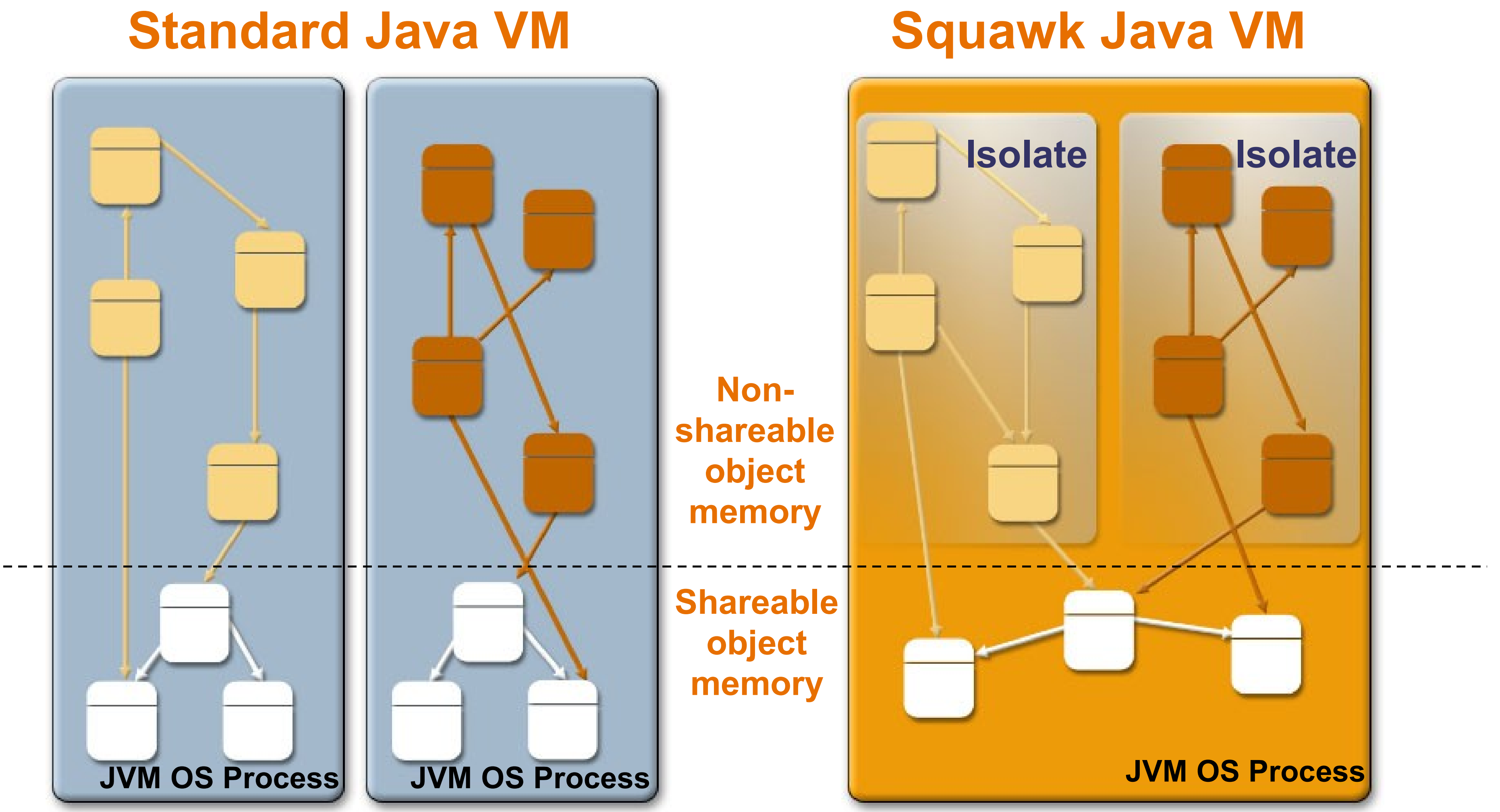
Squawk's Split VM Architecture



Isolate Application Model

- JSR 121: Application Isolation API Specification
- Application state is an object `Isolate`
  - `start` -`exit` -`moveTo`
  - `resume` -`pause`
- Every isolate has its own state for all static variables
- Allows for running multiple applications in one VM
- Inter-isolate communication
  - Provides lower-level asynchronous message delivery
- Can migrate from one device to another

Multiple Isolates (Applications) on the One Java VM



Isolate Migration

- State of a running application is migrated to another device
  - Continues running on target device where it left off
- Target device must have suite of the application
- Constraints on external state
  - There must be none, or
  - I/O sub-system must be homogeneous at both ends (Sun SPOT Squawk), or
  - I/O sub-system must be serializable (desktop Squawk)

Uses of Isolates and Isolate Migration

- Uses of isolates
  - > Represent applications as objects
  - > Support hardware resource sharing
  - > Field hardware replacement
- Uses of isolate migration
  - > Load balancing and fault tolerance
  - > Local debugging of remote application
  - > Seamless client-server programming model

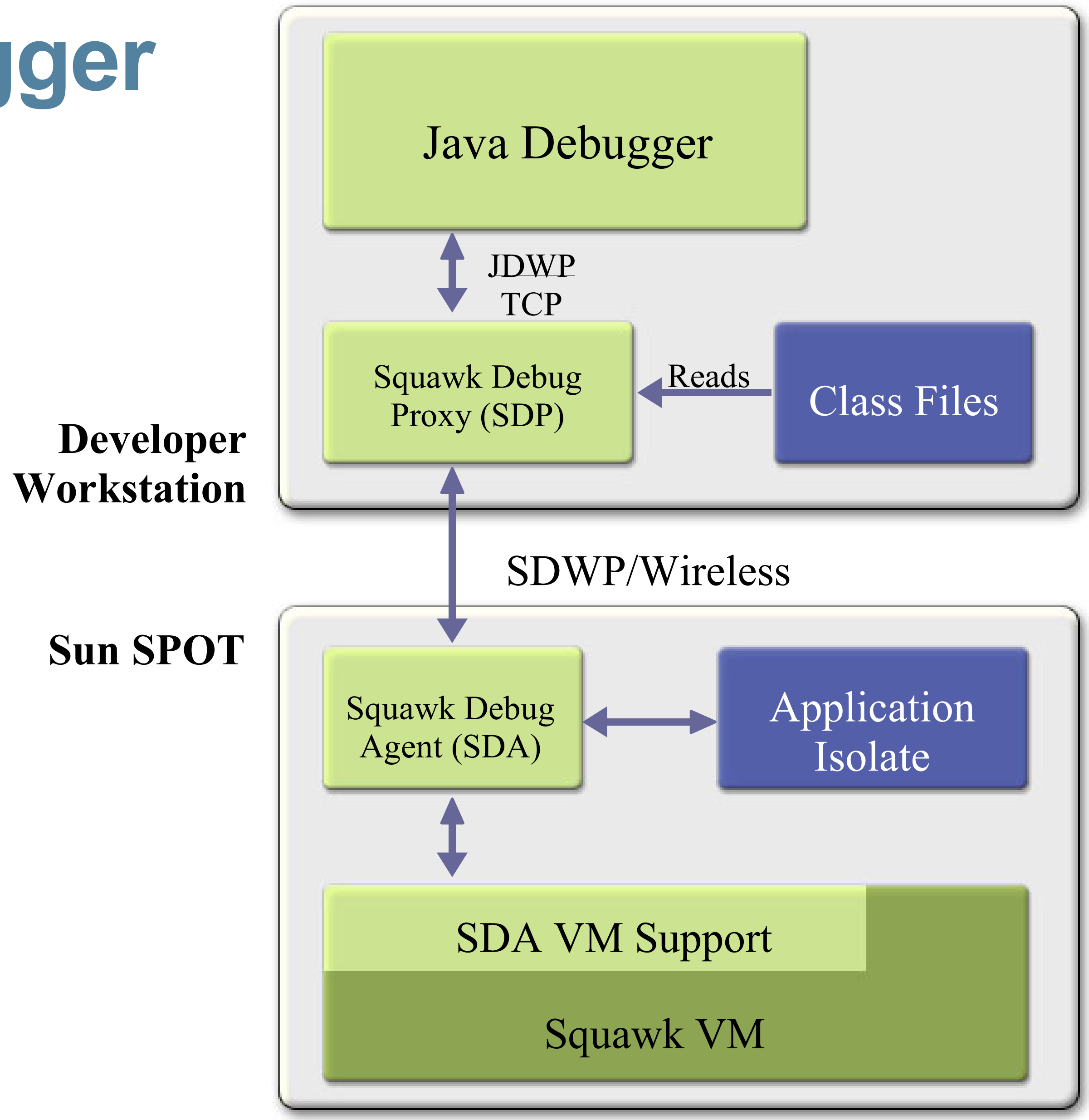
Ease of Development

- Java technology
- Can run on desktop as well as on-device
- Runs with standard IDEs or command-line
- Supports standard Java debuggers
  - Stepping, breakpoint
- On-device debugging

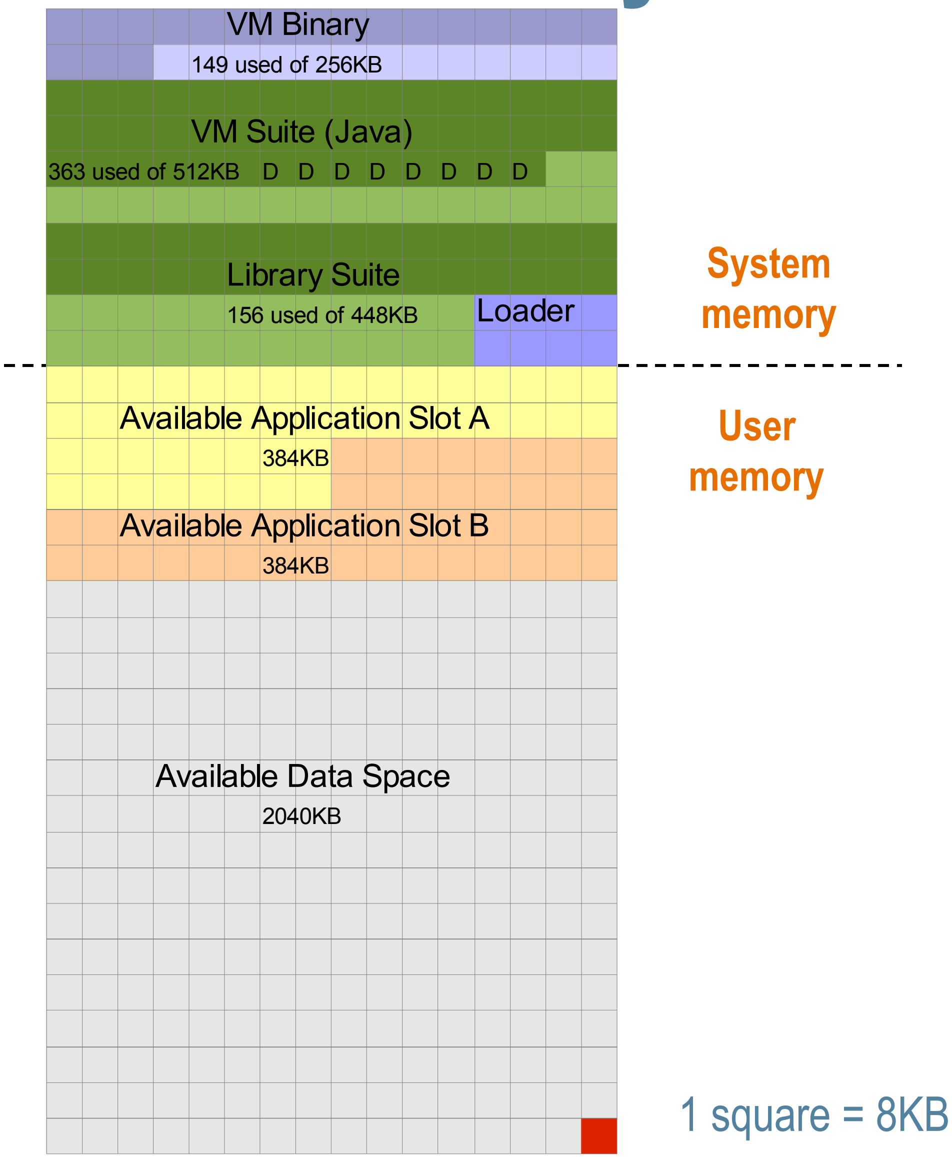
Debugger

- Use standard Java Debug Wire Protocol (JDWP) debuggers
- Debug via USB or over-the-air (OTA) on-device
- Challenges
  - > Not much memory on-device
  - > Slow communication link to debugger client
  - > Bytecode patching too slow (stored in flash)

Debugger

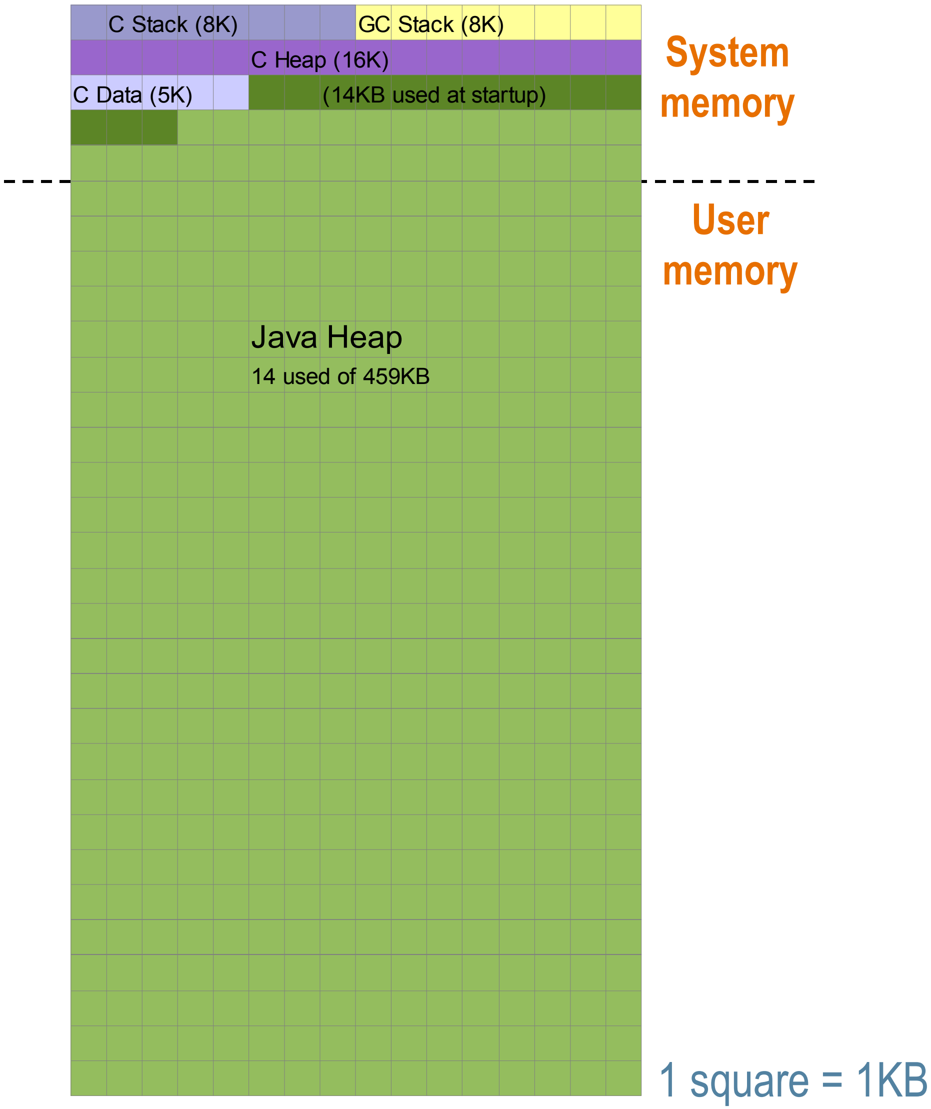

Developer
Workstation
Java Debugger
JDWP
TCP
Squawk Debug
Proxy (SDP)
Reads
Class Files
SDWP/Wireless
Sun SPOT
Squawk Debug
Agent (SDA)
Application
Isolate
SDA VM Support
Squawk VM

Squawk on the Sun SPOT: Flash Memory



- 4 MB flash
  - > Very low power
  - > 1 million cycles/sector endurance
- 1/3 reserved for System
  - > Not all in use
- 2/3 reserved for applications and data

Squawk on the Sun SPOT: RAM



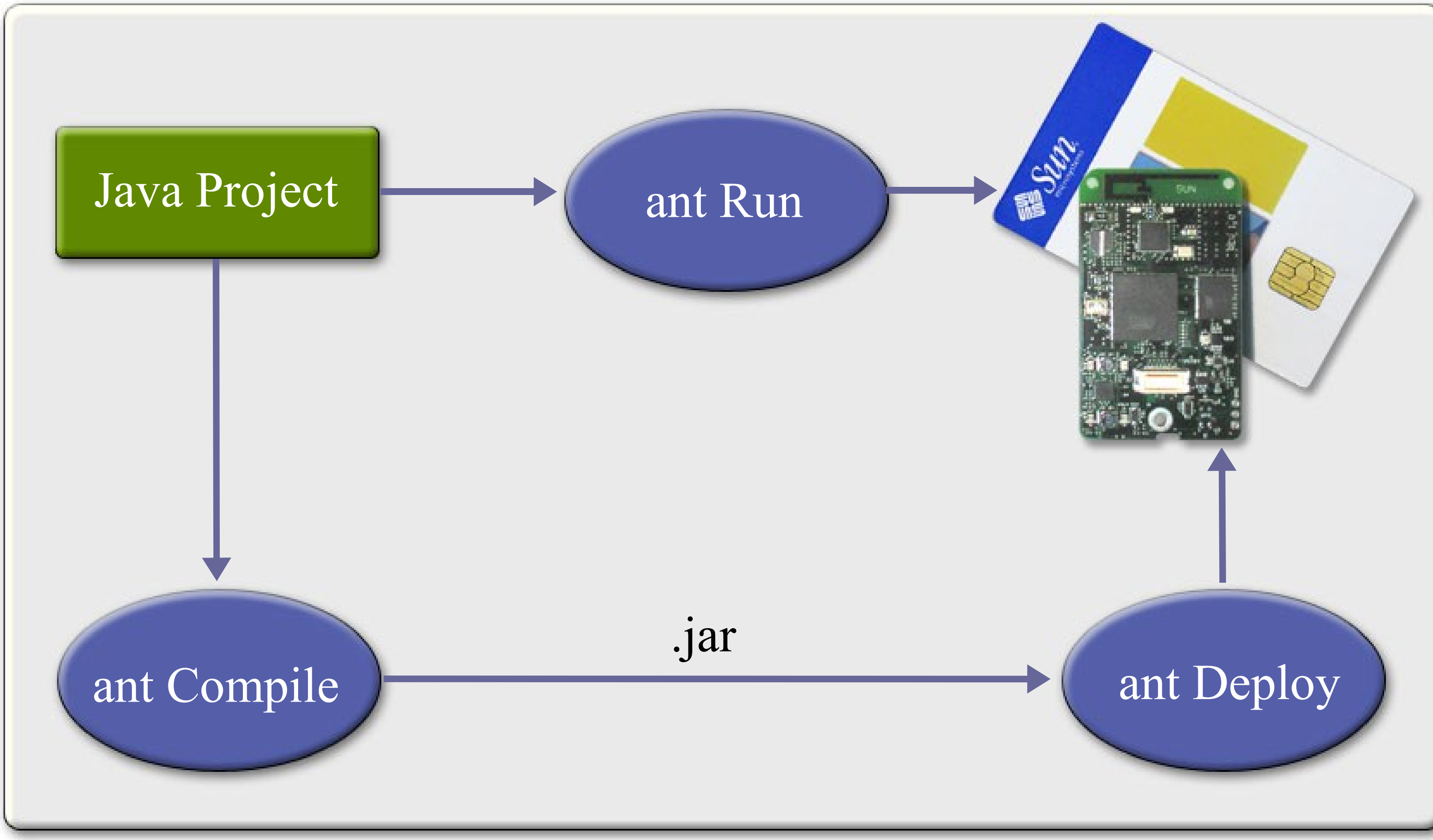
- 512 KB pSRAM
  - > Active current ≈ low mAs
  - > Inactive current ≈ low µAs
- >80% available for application objects

subject to change

Java Programming for Sun SPOTs

What will you open?

Build and Deploy Process



Sun SPOT Programming Environment

- Standard J2ME™ CLDC application environment
- Libraries are CLDC-based with extensions
- Squawk uses a form of JSR 121 isolation API
  - Multiple applications running on one Java Virtual Machine (JVM)
- Connection framework for device specific features
  - radio:// for 802.15.4 communication
  - msg:// for inter-isolate communication (proposed)

Sun SPOT Libraries

- Standard J2ME libraries
  - CLDC 1.0
- Hardware libraries
  - SPI, AIC, TC, PIO drivers all in Java technology
  - Sensor board hardware driven by Java technology (no C)
  - ADCs, GPIO, IrDA, etc.
- Radio libraries
  - To drive Chipcon CC2420 from Java technology (no C)

Sun SPOT Libraries (Cont.)

- Network libraries
  - 802.15.4 MAC layer in Java technology (no C)
  - Connection framework interface
- Desktop libraries
  - Create connections from standard J2SE VM to SPOT
  - Utilize SPOT in testboard as a gateway

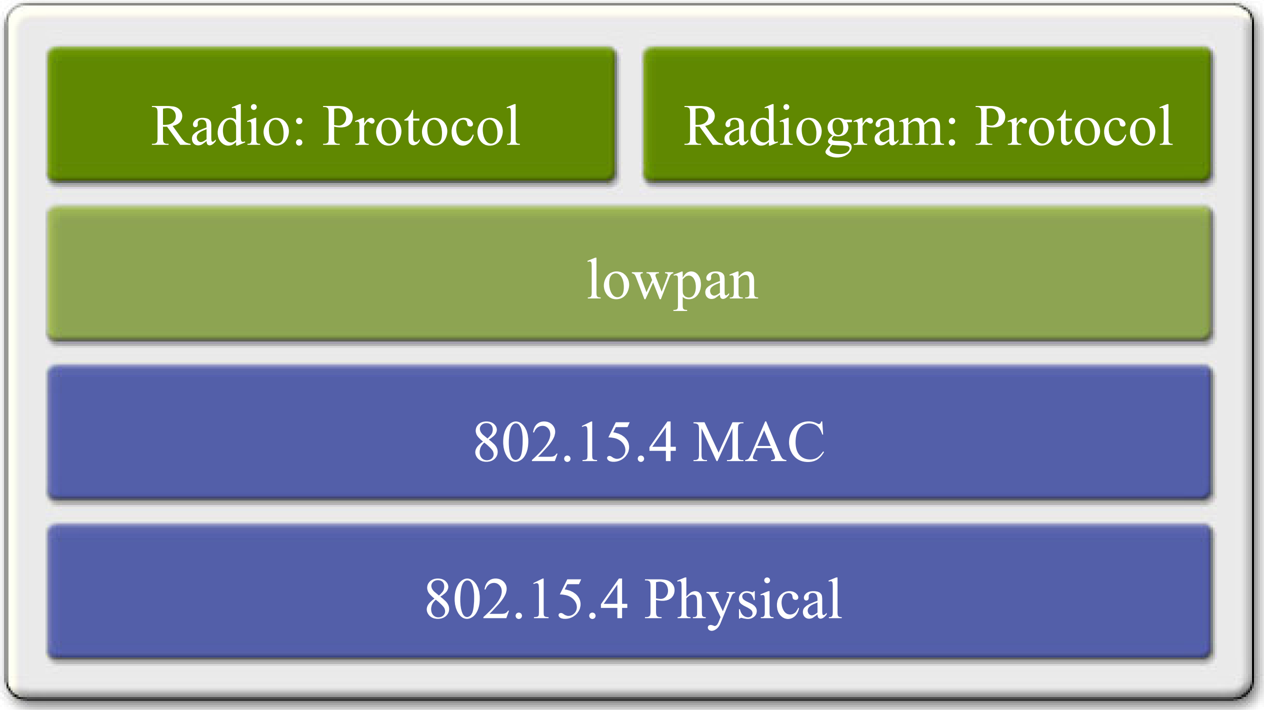
Security and Sun SPOT

- Data sent to your SPOT is cryptographically signed.
  - > Ensure valid bytecodes
  - > Prevent remote attackers from downloading dangerous code to your SPOT.
- SPOT contain public-private key
  - > Created en user first required a key
  - > Only owner is allowed to install new apps
- **`ant deletepublickey`**
- **`ant deploy -Dremote=0014.4f01.0000.0006`**

What will you open?

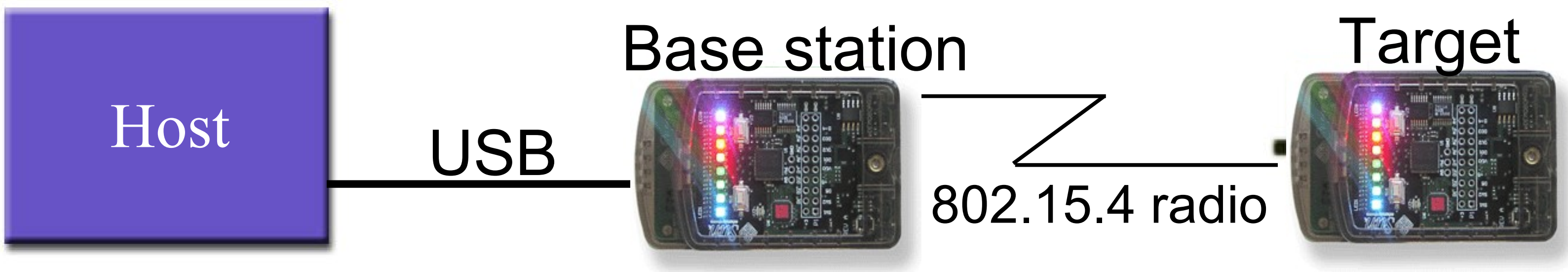
Sun SPOTs Communication

The Sun SPOT Radio Stack



IEEE 805.15.4, 250 kbps OTA

Using the Base Station



- Spot - Spot communication:
  - Sender sent radio package & no acknowledgment from target → NoAckException
- Host – Target communication:
  - No NoAckException only confirms delivery to the base station
  - If base station fail to deliver package to target → System.out warning

SPOT Communication Protocol

- Radio Protocol:
  - > Socket-like peer-to-peer protocol that provides reliable, buffered streambased IO between two devices

```
RadioConnection conn =
   (RadioConnection)Connector.open("radio://<des
   tinationAddr>:<portNo>");
```

- Radiogram Protocol:
  - > Client-server protocol that provides reliable, buffered datagram-based IO between two devices
  - > Server:

```
RadiogramConnection conn =
 (RadiogramConnection)Connector.open
 ("radiogram://:<portNo>");
```

SPOT Communication Protocol

- Radiogram Protocol:
  - Client:

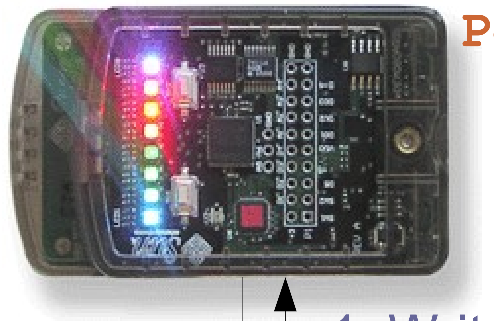
```
RadiogramConnection conn =
  (RadiogramConnection)Connector.open
   ("radiogram://<serveraddr>:<portNo>");
```

- Broadcasting:

```
RadiogramConnection conn =
  (RadiogramConnection)Connector.open
   ("radiogram://broadcast:<portNo>");
```

Example: Welcome Message...

0014:4F01.0000.0006



Port 100

1. Write “Hello up there”
2. Flush
3. Wait for answer

4. Read buffer
5. Answer
6. Flush

6. Print answer

Port 100

0014:4F01.0000.0007

Radio Connection: Program 1

```
RadioConnection conn = (RadioConnection)Connector.
                open("radio://0014:4F01.0000.0006:100");
DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos=conn.openDataOutputStream();
String answer = "";
try {
 dos.writeUTF("Hello up there");
 dos.flush();
 answer = dis.readUTF();
System.out.println ("Answer was: " + answer);
} catch (NoAckException e) {
  System.out.println ("No reply from
                                  0014.4F01.0000.00006");
} finally {
  dis.close();
  dos.close();
  conn.close();
}
```

Radio Connection: Program 2

```
RadioConnection conn = (RadioConnection)Connector.
          open("radio://0014.4F01.0000.00007:100");
DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos=conn.openDataOutputStream();
String question = "";
try {
   question = dis.readUTF();
   if (question.equals("Hello up there") {
   dos.writeUTF("Hello down there");
   } else
     dos.writeUTF("What???");
   dos.flush();
} catch (NoAckException e) {
  System.out.println("No reply from
                                   0014:4F01.0000.0007");
} finally {
  dis.close();
  dos.close();
  conn.close();}}
```

Radiogram Client Connection

```
RadiogramConnection conn = (RadiogramConnection)
 Connector.open("radiogram://0014.4F01.0000.00006:10");
Datagram dg =
      conn.newDatagram(conn.getMaximumLength());
String answer = "";
try {
   dg.writeUTF("Hello up there");
   conn.send(dg);
   conn.receive(dg);
   answer = dg.readUTF();
   System.out.println ("Received: " + answer);
} catch (NoAckException e) {
   System.out.println ("No-reply 0014.4F01.0000.00006");
} finally {
   conn.close();
}
```

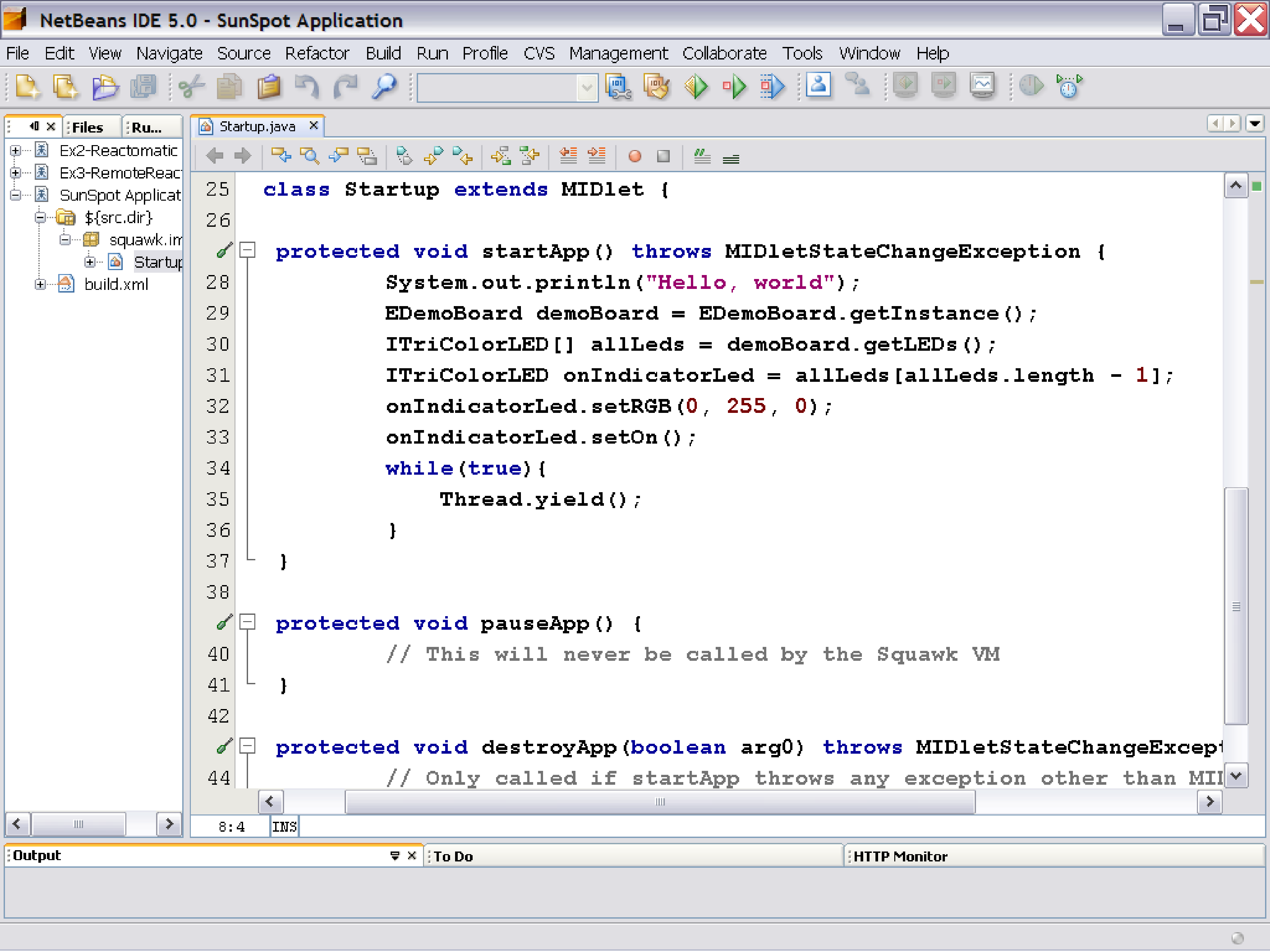
Radiogram Server Connection

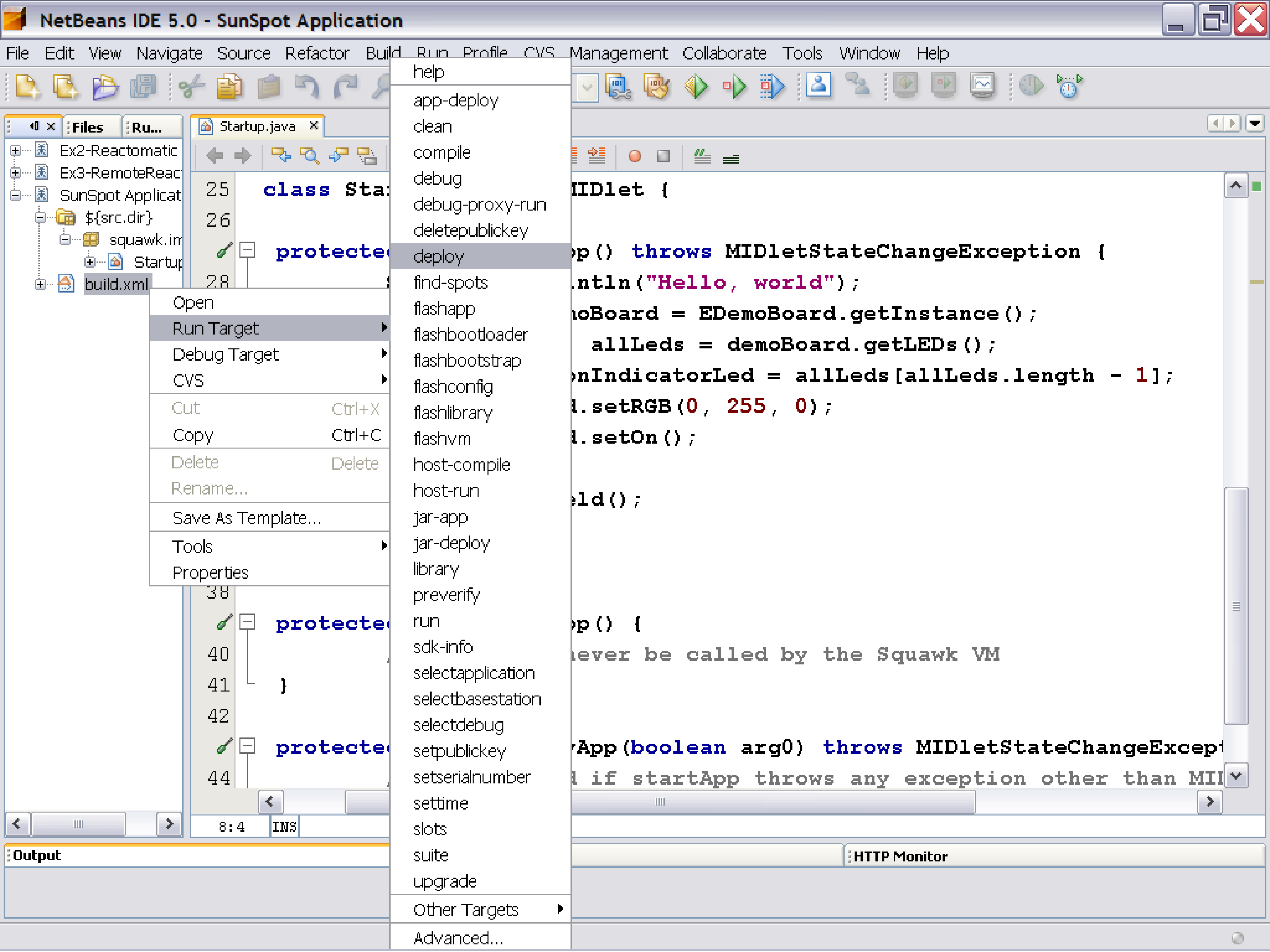
```
RadiogramConnection conn = (RadiogramConnection)
                    Connector.open("radiogram://:10");
Datagram dg =
                 conn.newDatagram(conn.getMaximumLength());
Datagram dgreply =
                 conn.newDatagram(conn.getMaximumLength());
String question = "";
try {
   conn.receive(dg);
   question = dg.readUTF();
   dgreply.reset(); //reset stream pointer
   dgreply.setAddress(dg); //copy reply addr from input
  if (question.equals("Hello up there"){
    dgreply.writeUTF("Hello down there");
   } else {
    dgreply.writeUTF("What???");
  }conn.send(dgreply);
} catch (NoReplyException e) {
    System.out.println ("No reply from " +
                                dgreply.getAddress());
} finally {
    conn.close();
}
```

SPOT Sensor Code

```
EDemoBoard demoBoard = EDemoBoard.getInstance();
ILightSensor a =demoBoard.getLightSensor();
int x= a.getValue()
ITemperatureIcon tempSPOT = demoBoard.getADCTemperature();
//Power on
ITriColorLED[] allLEDs = demoBoard.getLEDs();
ITriColorLED powerOn = allLEDs[allLEDs.length -1];
IAccelerometer3D myAccelerometer = demoBoard.
                                            getAccelerometer();
IScalarInput x = myAccelerometer.getXAxis();
IScalarInput y = myAccelerometer.getYAxis();
IScalarInput z = myAccelerometer.getZAxis();
//Set PowerON
powerOn.setRGB(100,100,100);
powerOn.setOn();
```

Demos: Netbeans 5.0


NetBeans IDE 5.0 - SunSpot Application
File Edit View Navigate Source Refactor Build Run Profile CVS Management Collaborate Tools Window Help
Files
Ru...
Ex2-Reactomatic
Ex3-RemoteReac
SunSpot Applicat
${src.dir}
squawk.im
Startup
build.xml
Startup.java
class Startup extends MIDlet {
protected void startApp() throws MIDletStateChangeException {
System.out.println("Hello, world");
EDemoBoard demoBoard = EDemoBoard.getInstance();
ITriColorLED[] allLeds = demoBoard.getLEDs();
ITriColorLED onIndicatorLed = allLeds[allLeds.length - 1];
onIndicatorLed.setRGB(0, 255, 0);
onIndicatorLed.setOn();
while(true){
Thread.yield();
}
}
protected void pauseApp() {
// This will never be called by the Squawk VM
}
protected void destroyApp(boolean arg0) throws MIDletStateChangeExcept
// Only called if startApp throws any exception other than MII
8:4
INS
Output
To Do
HTTP Monitor


NetBeans IDE 5.0 - SunSpot Application
File Edit View Navigate Source Refactor Build Run Profile CVS Management Collaborate Tools Window Help
Files
Ru...
Startup.java
Ex2-Reactomatic
Ex3-RemoteReac
SunSpot Applicat
${src.dir}
squawk.im
Startup
build.xml
Open
Run Target
Debug Target
CVS
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
Save As Template...
Tools
Properties
help
app-deploy
clean
compile
debug
debug-proxy-run
deletepublickey
deploy
find-spots
flashapp
flashbootloader
flashbootstrap
flashconfig
flashlibrary
flashvm
host-compile
host-run
jar-app
jar-deploy
library
preverify
run
sdk-info
selectapplication
selectbasestation
selectdebug
setpublickey
setserialnumber
settime
slots
suite
upgrade
Other Targets
Advanced...
class Sta
MIDlet {
protecte
pp() throws MIDletStateChangeException {
.ntln("Hello, world");
oBoard = EDemoBoard.getInstance();
allLeds = demoBoard.getLEDs();
nIndicatorLed = allLeds[allLeds.length - 1];
.setRGB(0, 255, 0);
.setOn();
eld();
pp() {
never be called by the Squawk VM
App(boolean arg0) throws MIDletStateChangeExcept
if startApp throws any exception other than MII
8:4
INS
Output
HTTP Monitor

Reactomatic Demo

```
protected void startApp() throws MIDletStateChangeException {
    EDemoBoard demoBoard = EDemoBoard.getInstance();
    ITriColorLED[] allLeds = demoBoard.getLEDs();
    ITriColorLED onIndicatorLed = allLeds[allLeds.length - 1];
    onIndicatorLed.setRGB(0, 255, 0);
    onIndicatorLed.setOn();
    ITriColorLED[] activityIndicatorLeds =
                          ITriColorLED[allLeds.length - 1];
    System.arraycopy(allLeds, 0, activityIndicatorLeds, 0,
                              activityIndicatorLeds.length);
    for (int i = 0; i < activityIndicatorLeds.length; i++) {
        activityIndicatorLeds[i].setOn(); // switch LED on
        activityIndicatorLeds[i].setRGB(0, 0, 0);
    }
```

Reactomatic Demo

```
IAccelerometer3D accelerometer = demoBoard.getAccelerometer();
accelerometer.setRange(0);
IScalarInput x = accelerometer.getXAxis();
...
int lastX = 0, lastY = 0, lastZ = 0;
while (true) {
    int r, g, b;
    try {
        int xValue = x.getValue();
        ...
        r = Math.abs(xValue - lastX) > JITTER ? 255 : 0;
        g = Math.abs(yValue - lastY) > JITTER ? 255 : 0;
        b = Math.abs(zValue - lastZ) > JITTER ? 255 : 0;
        lastX = xValue;
        ...
        } catch (IOException e) {...}
        for (int i = 0; i < activityIndicatorLeds.length; i++) {
            activityIndicatorLeds[i].setRGB(r, g, b);
 }}}
```

Remote Reactomatic Demo

```
StreamConnection conn = (StreamConnection)
 Connector.open("radio://" +otherSpot+ ":100");
DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos = conn.openDataOutputStream();

SensorSender sender=new SensorSender(dos, this);
SensorDisplay display=new SensorDisplay(dis, this);
new Thread(sender).start();
new Thread(display).start();
```

Remote Reactomatic Sender

```
if ((lastR != r) || (lastG != g) || (lastB != b))
{
        output.writeInt(r);
        output.writeInt(g);
        output.writeInt(b);
        output.flush();
        lastR = r;
        lastG = g;
        lastB = b;
}
```

Remote Reactomatic Receiver

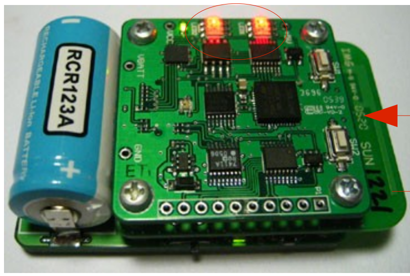
```
while (true) {
            try {
                int r = input.readInt();
                int g = input.readInt();
                int b = input.readInt();
                startup.display(r, g, b);

                Thread.yield();
            } catch (Exception e) {
              startup.showStatusError("Display
           problem.", e);
            }
          }
```

ServoBot Demo: Movement Recognition using the SPOTs

What will you open?

ServoBot: Getting ready



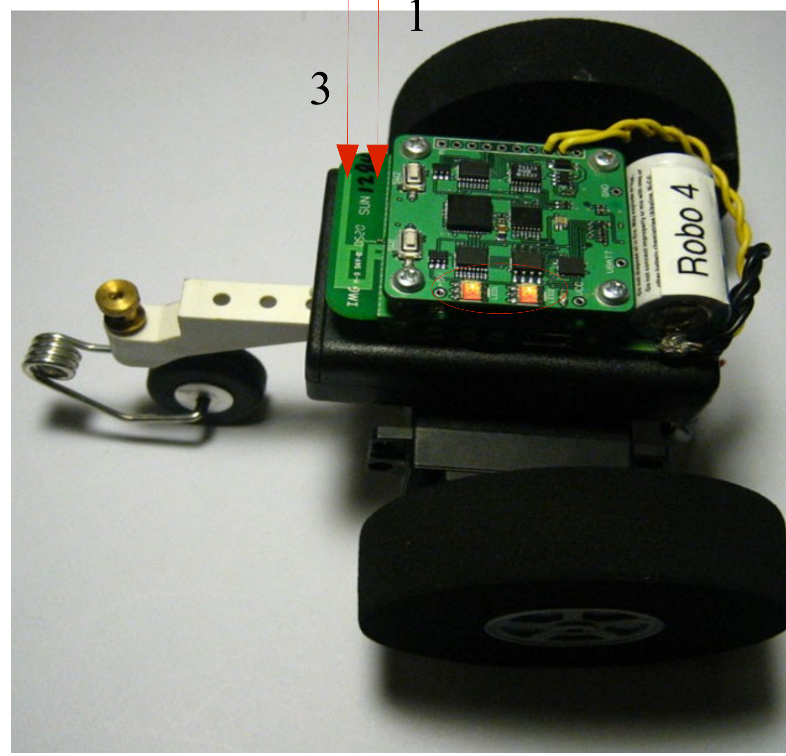
2

datagram who's there

datagram stop

1

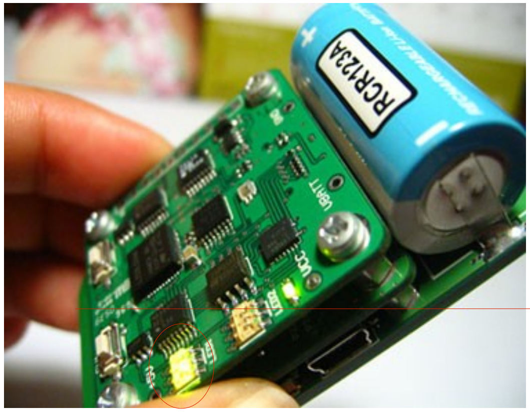
3



Accelerometer X = 0

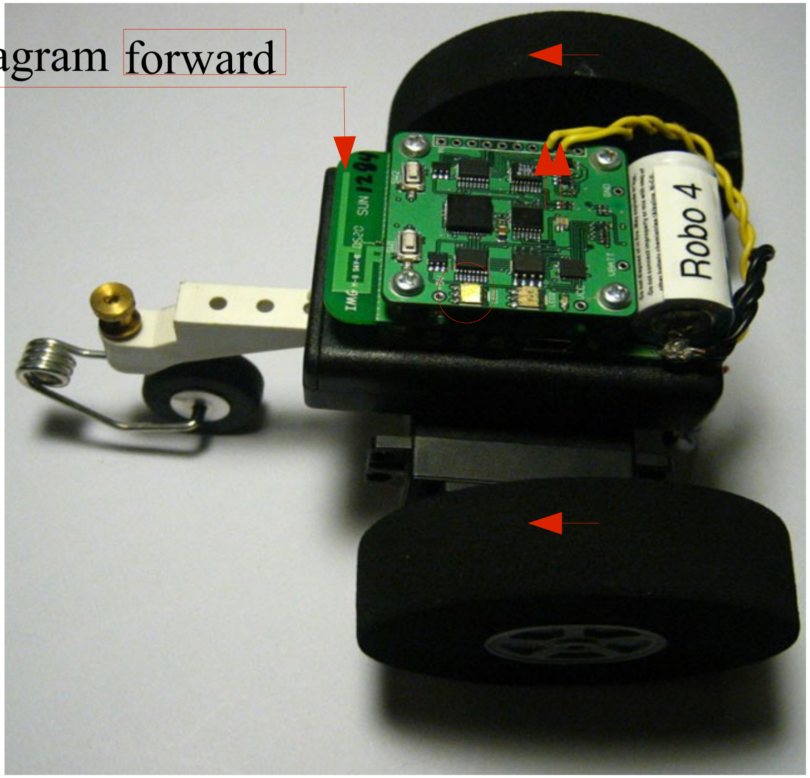
Accelerometer Y = 0

Moving forward



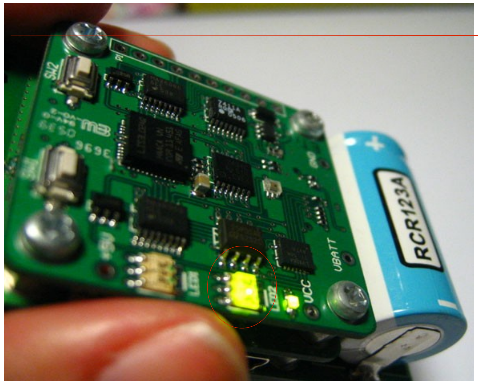
Accelerometer X < 0
Accelerometer Y = 0

datagram forward

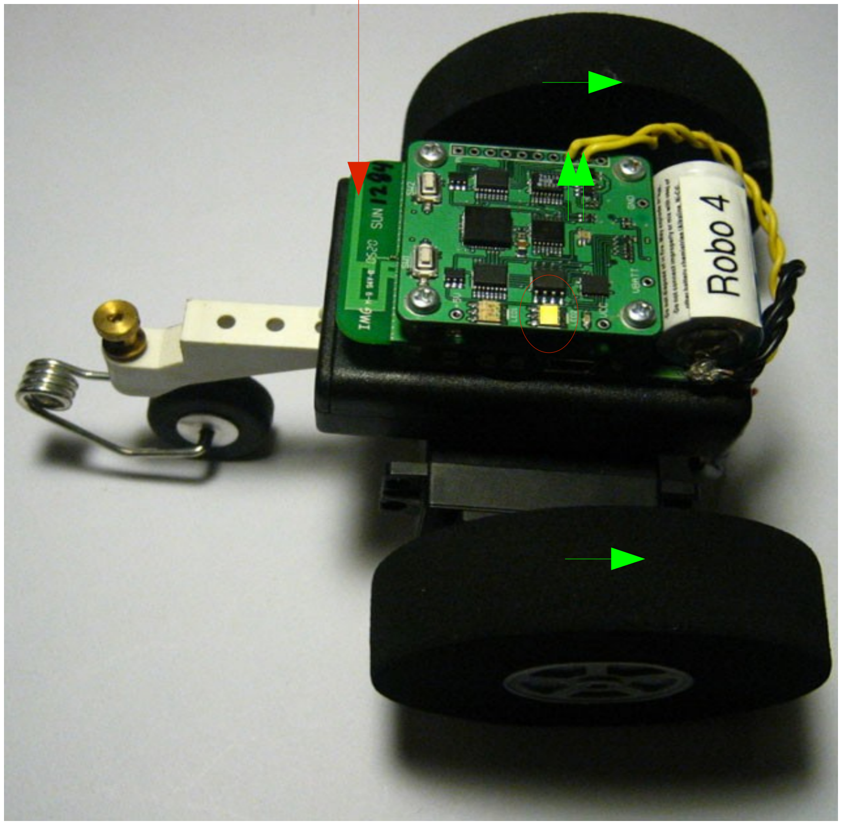


Moving backward

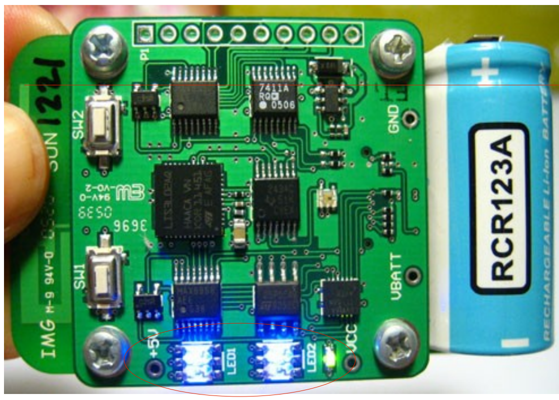
datagram reverse



Accelerometer X > 0
Accelerometer Y = 0

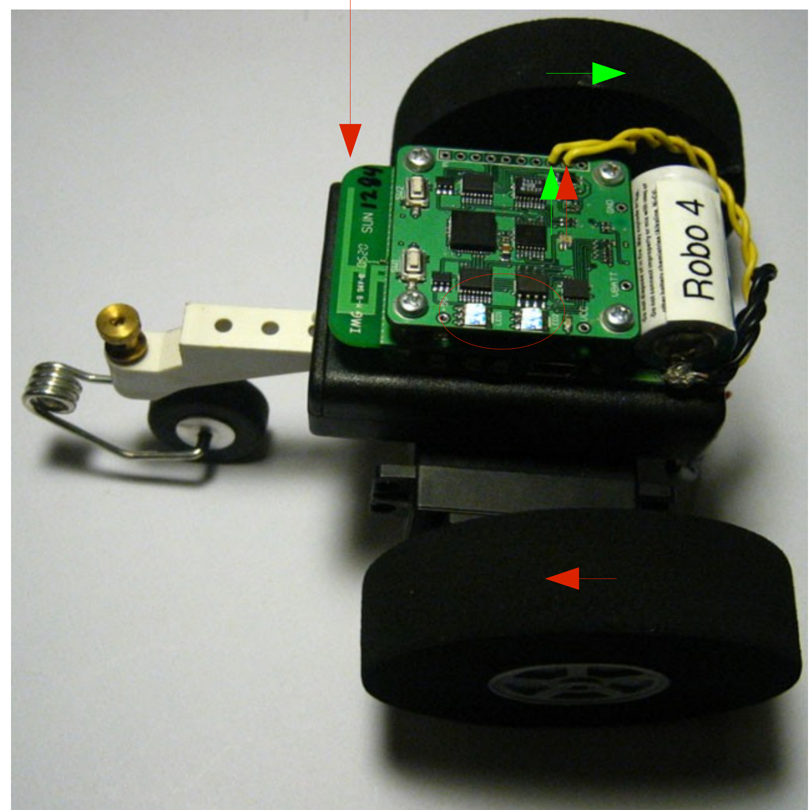


Turning right

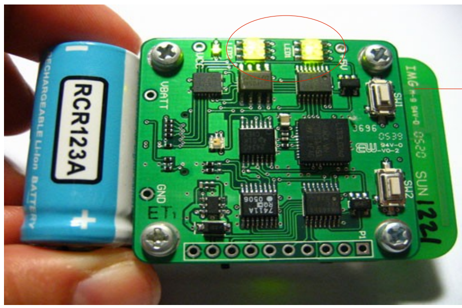


Accelerometer X = 0
Accelerometer Y > 0

datagram right

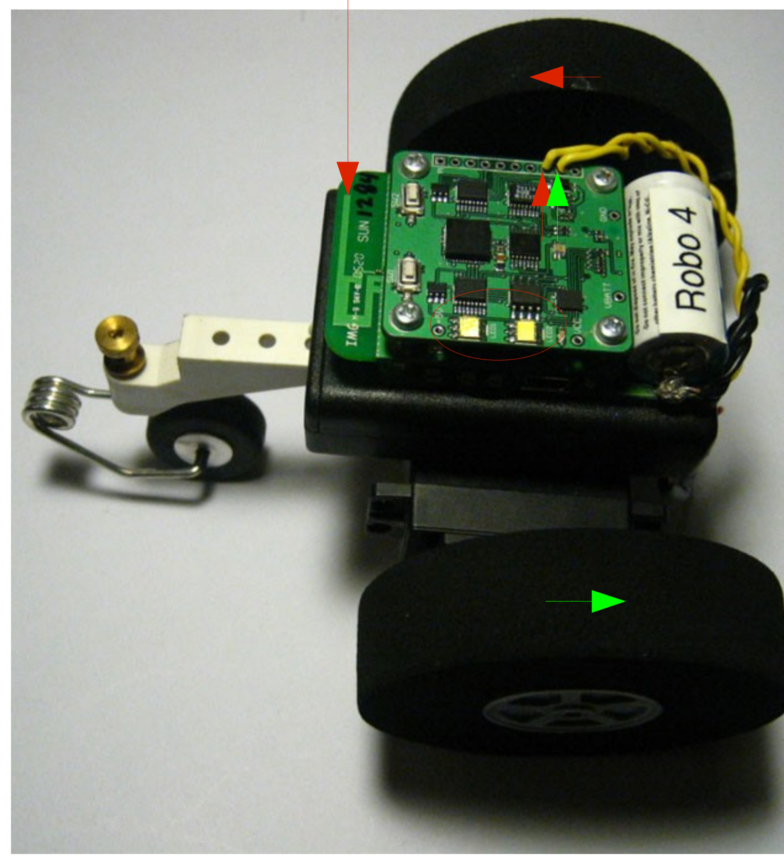


Turning left



Accelerometer X = 0
Accelerometer Y < 0

datagram | left |



Minority Report

Join Us Later at:

18:15-19:15

Scandinavia Scene

Unleash the power of Java!

Angela Caicedo,Simon Ritter og Matt Thompson

Summary

- Java technology on “wireless sensor networks” is here
  - > Better developer experience than the state-of-the-art
- Squawk: small Java-based VM
- Sun SPOT: mid-level sensor device that can be battery powered
  - > Enable exploratory programming
  - > Enable more on device computation and reduce network traffic
  - > Enable over-the-air programming

Future

- Collaborate with qualifying partners
- Use within Sun Labs
  - > Gesture based interfaces, building instrumentation, self-organizing systems, etc.
- Iterate hardware design
  - > Smaller chips, lower power, cheaper, etc.
- Iterate VM
  - > Smaller footprint, faster, smarter interrupts, power management, etc.
- Open schematics and VM to the community

For More Information

- Squawk
  > http://research.sun.com/projects/squawk
- Sun SPOT
  > http://www.sunspotworld.com
- Papers
  > “Java™ on the Bare Metal of Wireless Sensor Devices—The Squawk Java Virtual Machine”, VEE, June 2006

  > “The Squawk Virtual Machine: Java™ on the Bare Metal”, Extended Abstract, OOPSLA, Oct 2005

What will you open?

Thank You!!!

Angela Caicedo
Technology Evangelist
Sun Microsystems