Java is a trademark of Sun Microsystems, Inc.

JavaOne℠

FIRST: For Inspiration and Recognition of Science and Technology

Eric Arseneau
Brad Miller
Derek White

Agenda

- What is *FIRST*
- Talk about building robots
- Zero in on the control system in the robots
- Talk about providing Java Technology for the robots
- More about the Squawk JVM
- Talk about WPILib - the robotics library used by teams

What is *FIRST*

FIRST's mission: to inspire young people to be science and technology leaders, by engaging them in exciting mentor-based programs that build science, engineering and technology skills, that inspire innovation, and that foster well-rounded life capabilities including self-confidence, communication, and leadership.

"To transform our culture by creating a world where science and technology are celebrated and where young people dream of becoming science and technology heroes."

Dean Kamen, Founder

What is *FIRST*

- Programs
  - Junior *FIRST* LEGO League for 6 to 9 year-olds
  - *FIRST* LEGO League for 9 to 14 year olds
  - *FIRST* Tech Challenge for high-school students
  - *FIRST* Robotics Competition (FRC) for high-school students
    - 1683 teams from U.S., Brazil, Canada, Chile, Germany, Israel, Mexico, the Netherlands, and the Philippines, Turkey, and the U.K.
    - 42,075 high-school students

So How Does FRC Work?

- Kickoff on first Saturday in January
  - Game challenge for the year disclosed
  - Teams get kit of parts
    - Includes motors, control system, minimal base chassis, and some miscellaneous parts
- 6 weeks and 2 days later teams have...
  - Decided how to play the game
  - Brainstormed robot ideas
  - Built prototypes
  - Built, programmed, tested and drove their robot
  - Shipped finished robot (and a bunch of other activities)

The Challenge

- Game challenge changes every year
- Generally similar characteristics
  - Autonomous period of game (10-15 sec)
  - Tele-op period (2 minutes)
- Teams play in 3 on 3 alliances
- Competition is 3 days
  - 1 day of practice
  - 1-1/2 days of qualification rounds
  - 1/2 day of elimination rounds

Then...

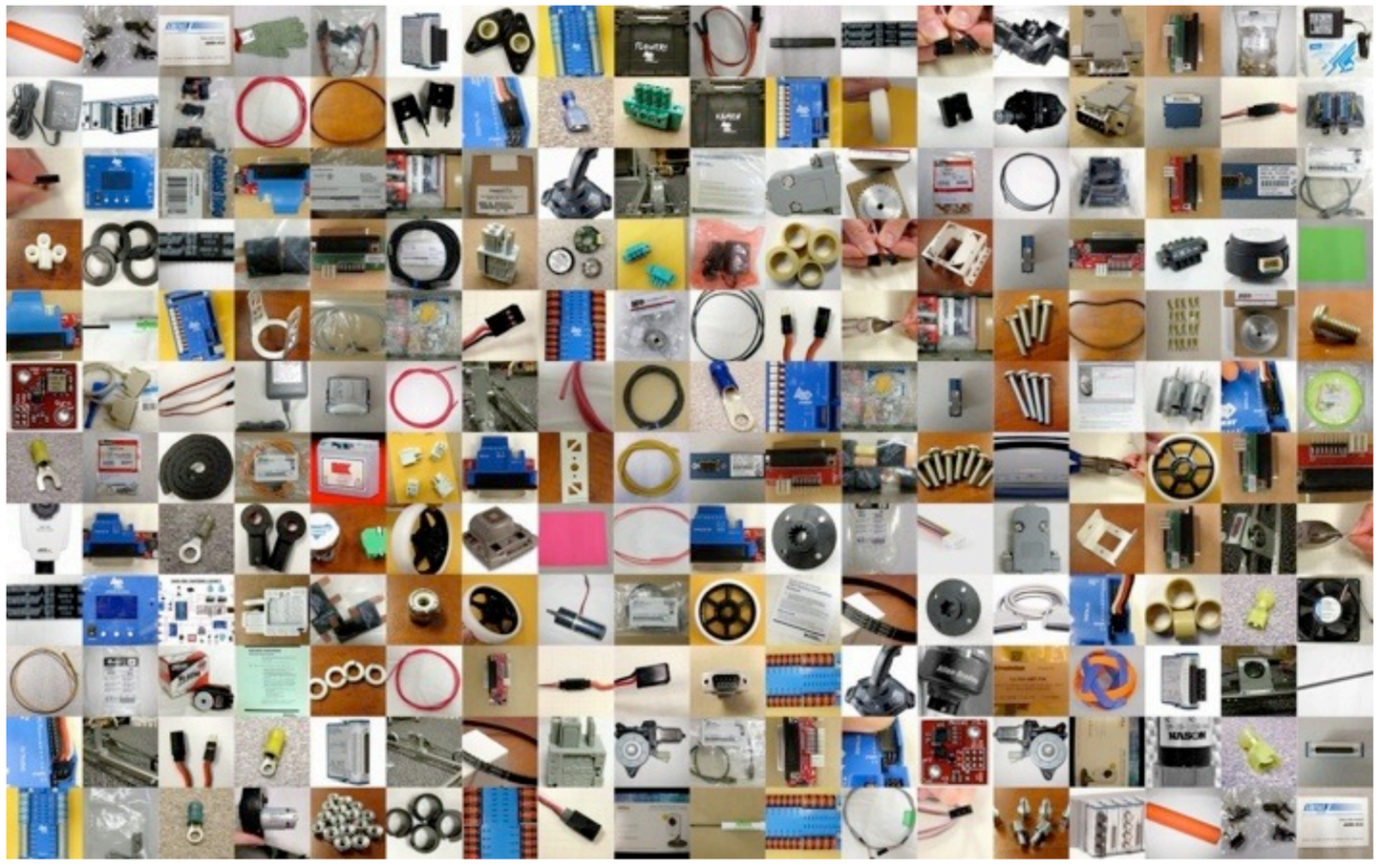
- 6 weeks of local competitions world-wide
  - 40 Regional competitions (US, Canada, Israel)
  - Winning teams go on to...

The Championship

- Championship at the Georgia Dome
  - About 20,000 people in attendance
  - 350 teams will have qualified
  - One winning alliance

The Robots

- Built from a standard kit of parts provided by FIRST
  - Motors, control system, minimal base chassis, assorted other components
- Everything else is supplied by the teams



- Robots specs:
  - 120 lbs MAX
  - About 3’x3’ and 5’ tall
  - Use SLA 12V battery
  - Travel 8-15 ft/sec

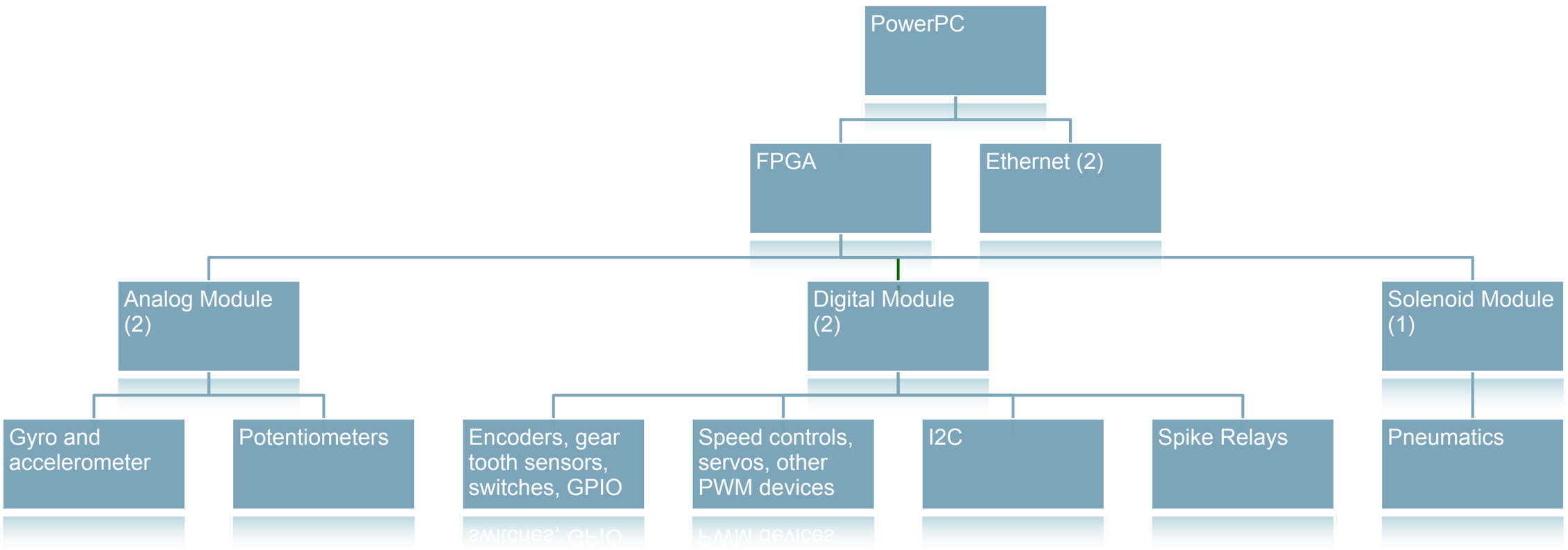
Sun microsystems

Robot Controller

- National Instruments Compact RIO
  - Rugged, embedded control and data aquisition system
  - 400 MHz industrial real-time processor for control, data logging, and analysis
  - 64Mb RAM and 128Mb flash
  - 2M gate, 8-slot FPGA chassis for custom I/O timing, control, and processing
  - Two 10/100BASE-T Ethernet ports; RS232 serial port for connection to peripherals
  - Expandable through plug-in modules

Explain FPGA

Architecture



cRIO Modules

- 5 I/O modules for chassis
  - 2-Digital I/O modules
    - 14 DIO
    - 10 PWM outputs
    - 8 Relay outputs
    - I2C port
  - 2-Analog I/O modules
    - 8 12-bit analog inputs
    - 500Khz sample rate per module (about 60K samples/sec)
  - 1-Solenoid (digital output) module
    - 8 pneumatic actuators

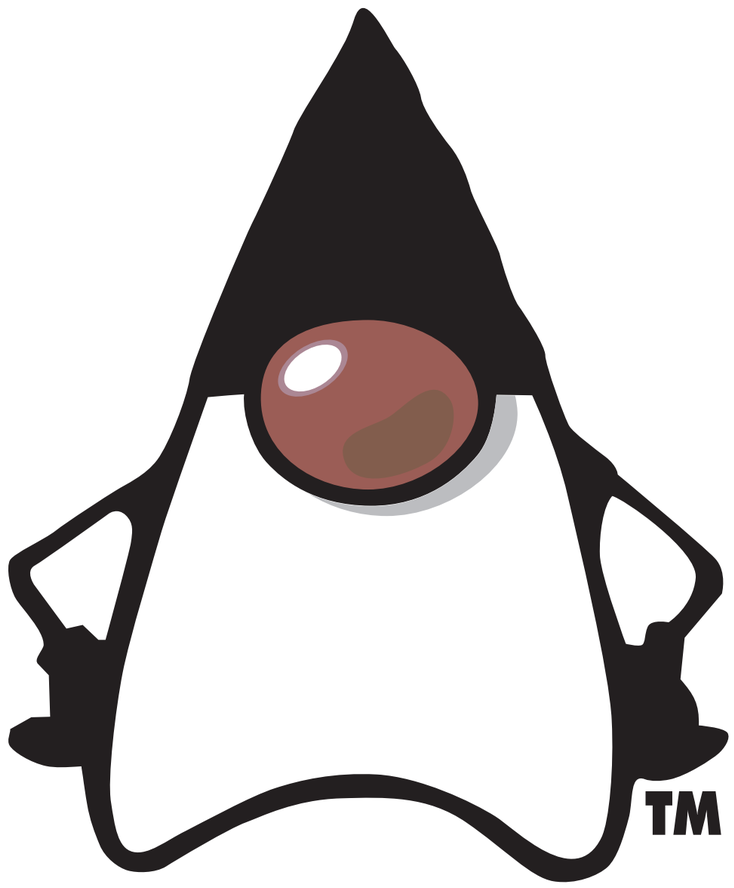
FRC FPGA - Facilitates Data Aquisition

- Analog Input
- Oversample / Average
- Accumulator
- Analog Trigger
- Digital Input / Output
- Slow Digital Output
- Hobby PWM Output
- I2C Bus
- Digital Input Filtering
- Solenoid Output
- Watchdog timers
- Counter / Timer
- SPI Engine
- Time / Alarm
- Routable Interrupts
- Direct Memory Access

Available

Now with...

- cRIO Controlle … stry standard real time opera
- Choice of WP…
  - C++
    - Uses Wind Riv… DE
    - Full C++ langu… e
    - Debugging
  - LabVIEW
    - Graphical … nstruments
    - Data fl… ll devices



Java Technology

Goals of WPILib Java

- Reduce the barrier to entry
  - Easy programs should be easy to write
  - More in-line with what students are doing in school
  - Easier, safer language for many students
  - Compatible interfaces with existing C++ WPILib implementation
- Better access to open tool suites
  - Everything is open source
- Completely extensible
  - Experienced programmers can extend the system to meet the most complex needs
- Easy reuse of components

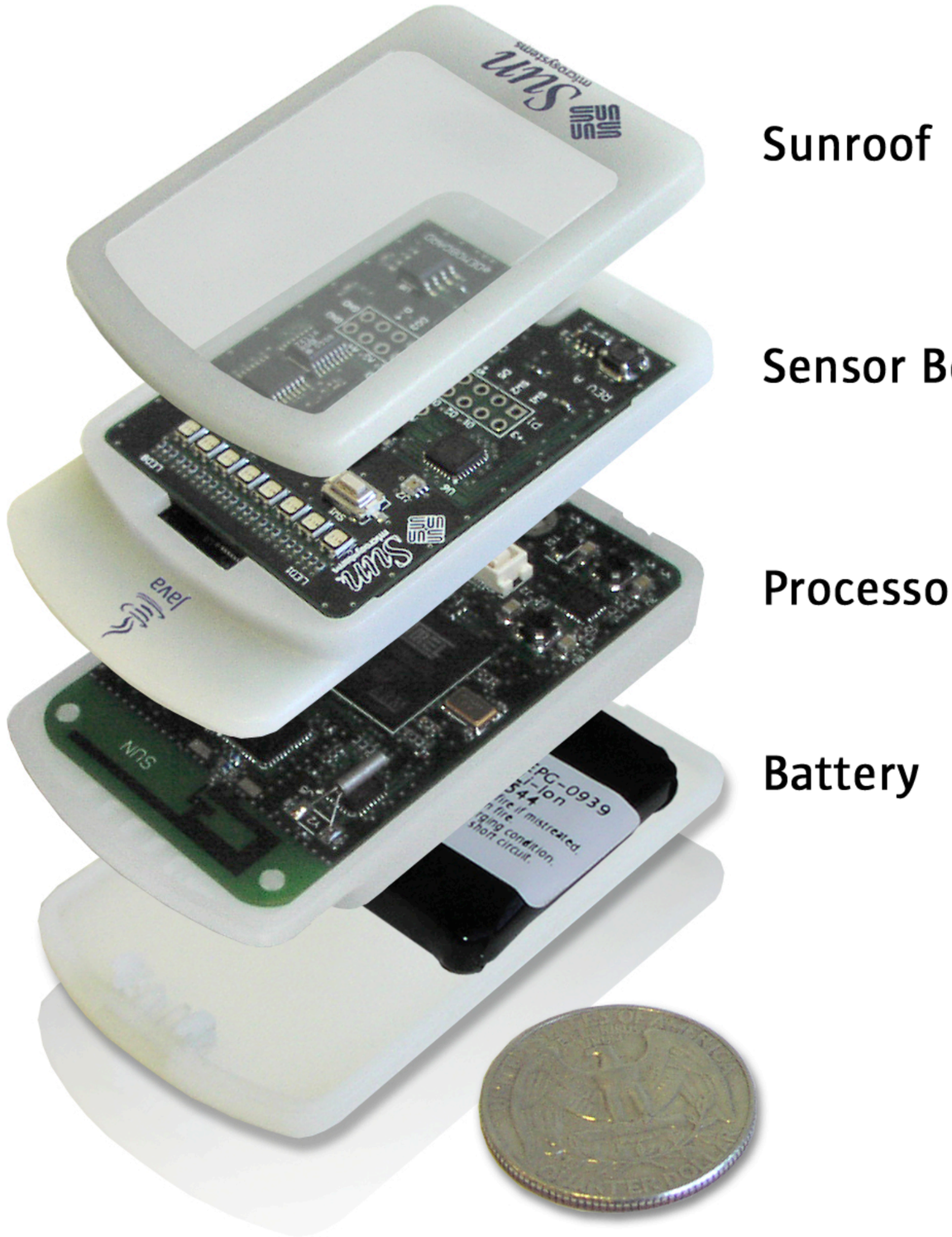
Sun SPOT Platform

- Software platform to enable
  - Java Technology
  - Common driver platform
  - Common services platform
  - Ease of development
- Intended to be ported to different devices
  - eSPOT
  - National Instrument Compact RIO
  - more...

What is a eSPOT?

- Built to inspire
- Designed for flexibility
  - Make hardware projects into software projects
  - Not designed for cost effective deployment today
  - Substitute money for time
- We want to
  - Build a Community of Developers
  - Enable New Devices and Services
  - Engage with new potential

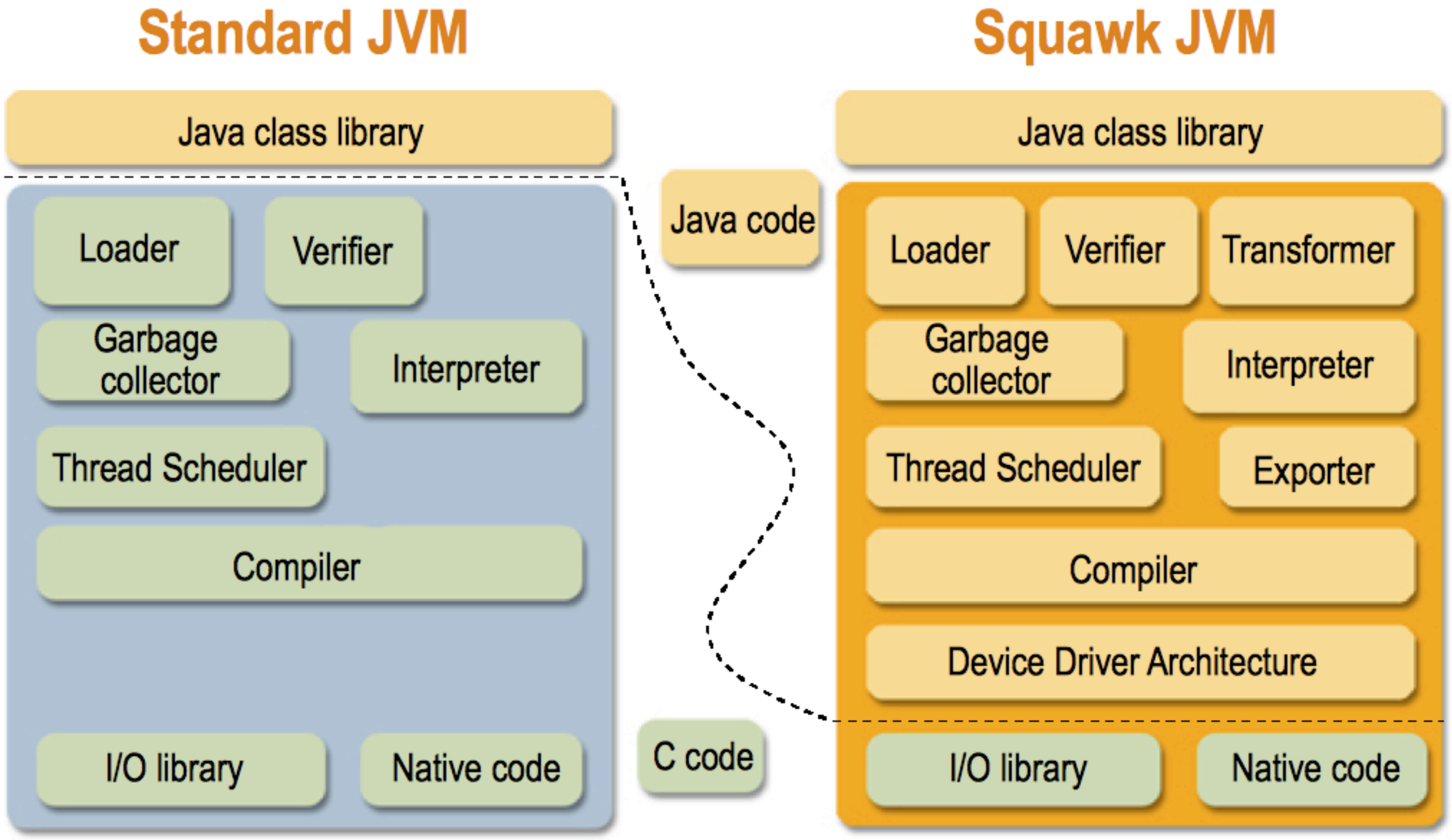
- Basic device has three layers
- User programs the device entirely in Java using standard Java tools



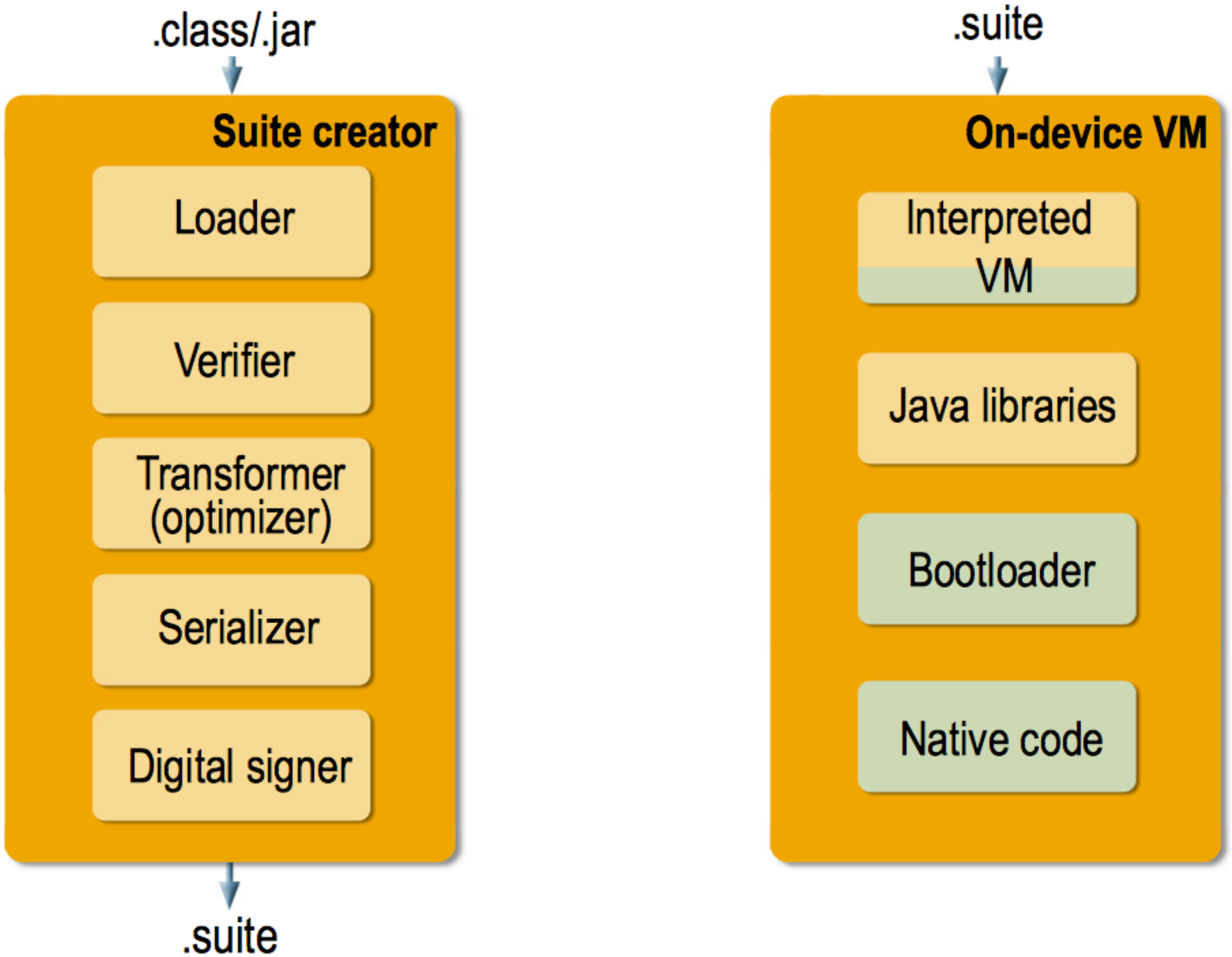
Sun SPOT Platform Java VM

- Squawk Virtual Machine
- J2ME CLDC 1.1 IMP Profile
- Designed for memory constrained devices
- Written mainly in Java
- Runs multiple applications (isolates), can “migrate” across devices
- Also runs on Solaris, Linux, Mac and Windows
- Easy to port

Squawk - Java in Java, Java on Java



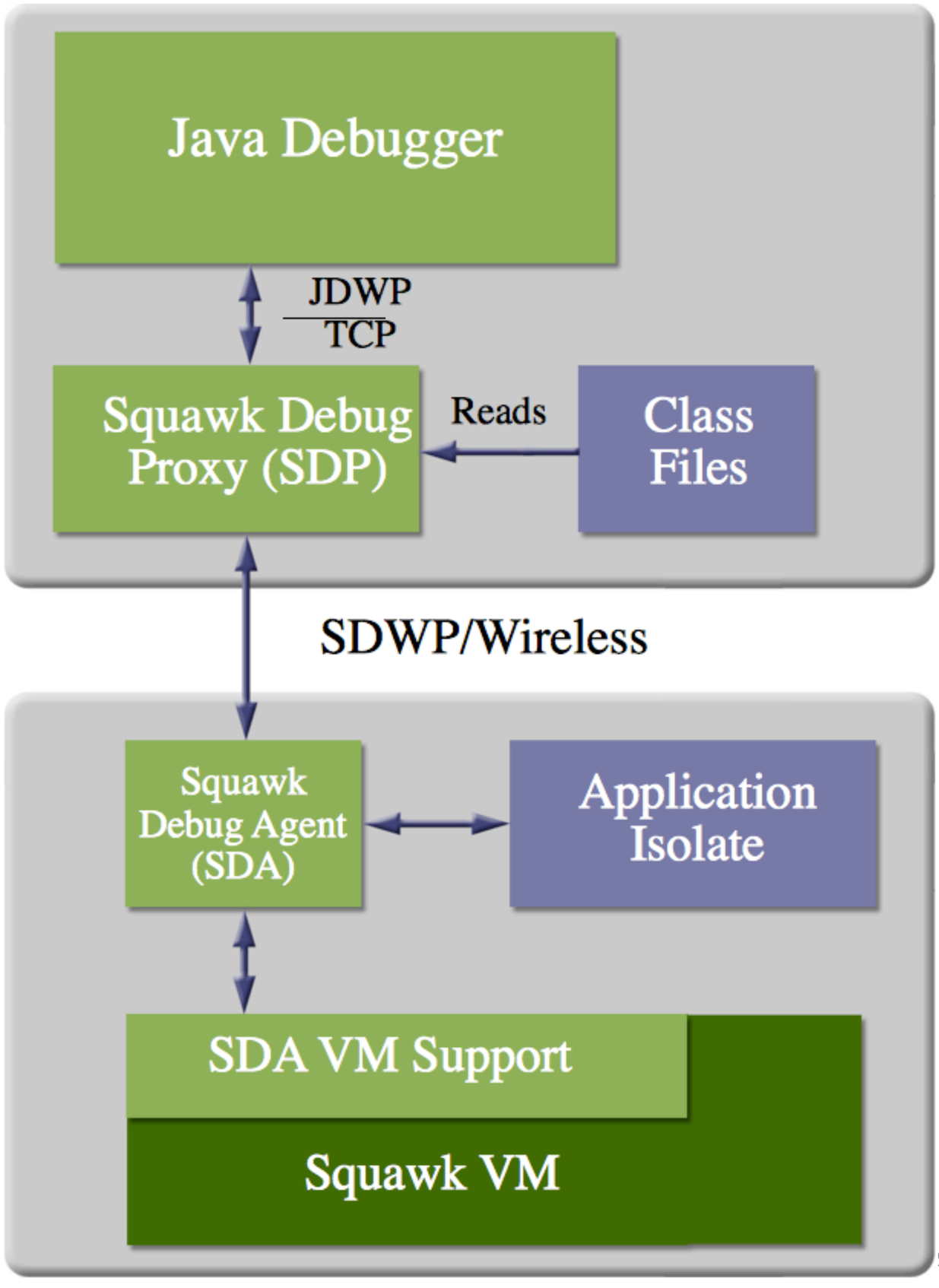
Split VM


.class/.jar
Suite creator
Loader
Verifier
Transformer (optimizer)
Serializer
Digital signer
.suite
.suite
On-device VM
Interpreted VM
Java libraries
Bootloader
Native code

Debugger

Developer Workstation

Sun SPOT



Squawk Ports

- Designed to run on “bare metal” (without OS)
  - Sun SPOT
  - Handles interrupts, radio packets in Java
- Also runs on top of Java SE
  - Java SE handles IO, etc.
  - Development and emulation
  - Windows, Mac, Linux & Solaris

Original design goal of Squawk was to run on the bare metal.

For development and emulation, Squawk also runs on common OSes. But for simplicity, a Java SE JVM was used to do all IO and other OS interactions.

Squawk - Native OS Ports

- Some devices have OS
  - May be required for security, legacy code,...
- Need to run on Posix-style OS
  - file I/O, sockets, etc.
- No JNI in Squawk
- JNI - mostly about giving C access to Java objects
  - Outsourcing / Exporting / Importing
- Need new scheme to access native C code

But there is a class of embedded devices that can benefit from Java, but already have an OS.
The existing code on the OS may be required for security purposes, or there's too much legacy to to reasonably port to Java
So Squawk needs to be able to run natively

Java Native Access

- Need to call C functions
  - Call
  - Peek, poke, access to C memory
- JNA (jna.dev.java.net)
  - Provides abstractions over call/peek/poke
  - For Java SE - uses reflection to generate calls to C

The bare minimum need to call out to C is a way to call a function, and a way to read/write to memory (structures, buffers, etc)

But it's nicer to have some high level abstractions. There are several open source projects that do this. The one that looked simple enough for small devices was JNA.
But JNA use Java SE features like reflection that are not available in Java ME CLDC

JNA for CLDC

- Needed simplified JNA
  - Wrappers for C code created “by hand” or scripts
  - Reflection not needed

```
//class LibCImpl:
Function fcntlPtr = jnaNativeLib.getFunction("fcntl");

public int fcntl(int arg0, int arg1, int arg2) {
    return = fcntlPtr.call3(arg0, arg1, arg2);
}

// use:
int res = libc.fcntl(fd, LibC.F_SETFL, flags);
```


For Squawk we designed a simplified version of JNA suitable for CLDC
You can write wrappers by hand.

Code:
Declare interface & class for C functions
Implement wrapper
Lookup C functions (“dlsym”)

Then can call out to

Squawk Port for cRIO

- Environment
  - VxWorks OS
  - FPGA automatically generated C library
- Solution
  - Start with Sun SPOT SDK and tools
  - Automatically generated JNA wrappers for FPGA
  - IDE and Debugger support
  - Program deployment

Sun SPOT Platform SDK

- NetBeans Module
- ant tasks
- Documentation
- Build tools

Java Technology to *FIRST* teams!

- Started as a WPI senior capstone project (MQP)
  - Done by 3 ECE students, 7 weeks for proof of concept
    - Squawk JVM port to PowerPC / VxWorks
    - Hardware interface libraries for FPGA
    - WPILib C++ library ported to Java with some cool home grown automated tools
  - Co-advised by Derek White from Sun in Burlington, MA
  - Currently in development of production code

WPILib

- Framework to support competition programs
  - Automatic or manual sequencing between autonomous and operator control code
- Parity between C/C++ and LabVIEW libraries

Programming Paradigm

- Subclass one of the robot classes
  - SimpleRobot
    - Assumes code polls iteratively
  - IterativeRobot
    - More advanced model that handles devices with varying timing requirements
- Base classes take care of field communication and match states (autonomous and teleop)

Robot Program Definition

```
public class RobotExample extends SimpleRobot
{
    private RobotDrive drive;
    private Joystick stick;
    private Watchdog dog;

// autonomous and operatorControl code codes here
// by overriding methods in SimpleRobot

}
```

code to sequence through competition and communicate with driver station

need to explain about autonomous and teleop before this

Initialization (constructor)

```
public RobotExample()
{
    drive = new RobotDrive(1, 2);
    stick = new Joystick(1);
    dog = Watchdog.getInstance();
}
```

Define the robot base

Joystick on Driver Station USB port 1

Watchdog timer instance

Autonomous part of program

```
public void autonomous()
{
    dog.setEnabled(false);
    drive.drive(0.6, 0.0);
    Timer.delay(2.0);
    drive.drive(-0.6, 0.0);
    Timer.delay(2.0);
    drive.drive(0.0, 0.0);
}
```

Disable watchdog timer

Drive forwards for 2 seconds, then backwards for 2 seconds, then stop

KEY TO THE EXAMPLE:

myRobot.drive(*speed*, *curve*)

speed: a value from -1.0 to 1.0 where 0.0 is stopped

curve: a value from -1.0 to 1.0 where 0.0 is no turn

Tele-Op part of program

```
public void operatorControl()
{
    dog.setEnabled(true);
    while (!isEnabled())
    {
        drive.arcadeDrive(stick);
        dog.feed();
        Timer.delay(0.01);
    }
}
```

Enable watchdog timer

Drive robot using one joystick

Feed the dog

Wait 10 ms

What kinds of objects are there?

- About 48 objects currently:
  - Robot definition
  - Motors and servos
  - Driver station
  - Sensors
  - Utility classes

Using an Encoder

- Encoder returns rotational information
  - FPGA gets counts and period (rate of rotation)
  - Quadrature encoders also return direction

```
public void autonomous() {

    encoder.reset();
    encoder.start();
    drive.drive(0.6, 0.0);
    while (encoder.get() < 16384) {
        Timer.delay(0.01);
    }

    drive.drive(0.0f, 0.0f);
}
```

Using a Gyro (Proportional Control)

- Gyro provides rate as a voltage
  - Integrated by FPGA accumulator to get heading
  - Proportional control provides smooth operation

```
public class GyroTest extends SimpleRobot {
    final Gyro gyro = new Gyro(1);
    final RobotDrive drive = new RobotDrive(1,2);

    public void autonomous() {
            gyro.reset();
            while (isAutonomous())
            {
                    double angle = gyro.getAngle();
                    drive.drive(-1.0, -angle / 30.0);
                    Timer.delay(0.004);
            }
            drive.drive(0.0, 0.0);  // stop robot
    }
}
```

How Did We Implement WPILib

- Low level JNA
- Re-implementing the library in Java
  - Done as a new implementation rather wrapping the C++ classes

Adopted by *FIRST* as new language

- Demonstrated at the *FIRST* Championship in April
- Now in full speed development
- Few teams alpha testing today
- Full beta test in fall 2009
- Release for the 2010 FRC competition

Everything Open Source!

- Community ownership – nothing proprietary
  - Allowing community contributions make for a more acceptable standard code base
- Nothing hidden “under the covers”
  - Gives teams a better understanding of how code works
  - Teams can now be more self supporting without relying on a single person/company for support
- Teams can easily extend and adapt code for their own needs

Sound Like Fun?

- Since 2003 FRC teams had C
- In 2009 they got C++ and LabVIEW
- In 2010 they're getting Java
- Lots of opportunities to help
  - Need mentors who know Java
  - Volunteers for events
  - Sponsors for teams

Your chance to "Change (Y)our World!

Where to Learn More

- BOF - Tonight at 6:30 Room 124 North Hall
- Pavilion
- Friday morning keynote
- On the Internet
  - http://www.sunspotworld.com/frc
  - http://www.usfirst.org
  - http://first.wpi.edu/FRC

JavaOne℠

Thank You

Eric Arseneau
eric.arseneau@sun.com
Brad Miller
bamiller@wpi.edu

Driver Station

- Linux-based driver station
  - Sends control data to robot
  - Receives status
- Can create custom UIs
  - 4 USB Joystick ports
  - 8 digital outputs
  - 8 digital inputs
  - 4 analog inputs

Driver Station