Java application platforms for design-to-cost embedded systems

Régis Latawiec, COO

IS2T

www.is2t.com
Embedded Processing Market Share
IS2T - Solutions for Embedded Innovations

Develop software applications and leverage innovations at low Total Cost of Ownership.

Portability across all hardware and software environments

microcontrollers < $5
microprocessors < $15

Edge Devices  Gateway  Cloud
Embedded Market Maturity

• Like servers, workstations and smartphones…
• … cost constrained embedded systems now look at 3rd party “platform” procurement

Make or Buy?

Applications
Service platforms
OS & BSP
Processors

Mature for big embedded systems like Android, iOS, Linux

Mature (buy)
MicroEJ Platforms

MicroEJ Editions

<table>
<thead>
<tr>
<th>Cross Platform Environments</th>
<th>EMBEDDED DEVICE EDITION</th>
<th>HARDWARE AS A SERVICE EDITION</th>
<th>HARD REAL TIME EDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontrollers</td>
<td>Keil RTX, uCOS, ThreadX, FreeRTOS… (ARM Cortex M, Renesas RX, …)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microprocessors</td>
<td>Linux, VxWorks, Integrity, PikeOS (Cortex A, MIPS, …)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobiles</td>
<td>iOS, Android</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td>MacOS, Linux, Windows</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Java Platforms Concept for Embedded systems

Java language

Java technology

MicroEJ Virtual Machines (JVM – mostly software processors)

Different JVM architectures to fit different architecture flavors

Microprocessors
Optimized MicroEJ® VMs

- Code quality
- Productivity
- Reliability
- Portability
- Scalability
- Maintainability
- Code density

- High speed execution
- Determinism (HRT)
- Tiny footprint
- Interface C/asm
- Low power, etc.

32/64-bit CPU

Optimized implementation

32-bit MCU/MPU
Embedded Java Platform Example

• STM32F2x (Cortex-M3) – 120MHz
• 16-bit col. QVGA LCD, Touch
• APIs: B-ON, MicroUI, MWT, SNI
• Boot time (reset to main(String[] args)): 2ms

### Application Memory Requirements

<table>
<thead>
<tr>
<th></th>
<th>Flash (ROM)</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Machine (runtime &amp; GC)</td>
<td>422KB</td>
<td>42KB</td>
</tr>
<tr>
<td>Libraries (graphics, com, float...)</td>
<td>132KB</td>
<td>1KB</td>
</tr>
<tr>
<td>Graphical resources (images)</td>
<td>228KB</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>34KB</td>
<td>13KB</td>
</tr>
</tbody>
</table>

Java needs
GUI Examples on STM32 MCUs

Eclipse IDE
EMBED JAVA TO A LEGACY C BASED APPLICATION

MicroEJ Embedded Devices Edition
What is a MicroEJ Embedded Platform?

- Dual Java Platform
  - Embedded platform (EmbJPF)
  - Simulated platform (SimJPF)

- Integration with legacy
  - RTOS if any
  - Firmware & Driver

- General purpose
  - CLDC/EDC, BON, NLS

- Special packs
  - UI, IoT, Num, SOA
MicroEJ SDK

• Java platform design
  » Integrate to your RTOS
  » Interface to your C code
  » Supports ARM-Keil, GNU, IAR, GreenHills, Windriver

• Java application design
  » Java project editor
  » Simulate to prototype and debug
  » Analyze memory usage
  » Deploy
Easy RTOS Integration (Green Thread)

• Multi-threaded Java execution environment within a single RTOS task

RTOS Examples
• μC/OS, ThreadX, RTX
• FreeRTOS
• Linux, Integrity, VxWorks
• Your RTOS!
Easy RTOS Integration (Green Thread)

• Same Java thread scheduling policy for all RTOS
  » Portability improved
    Not only at binary level, but also scheduling level

• Easy control of CPU resource usage for Java world
  » Java RTOS task priority setting for Java world
  » CPU resource allocation irrespective of the number of threads

• Java threads & native Tasks synchronization means
  » Allows synchronous and asynchronous Java / native calls
Easy Java → C Interface (Calls 1/2)

• SNI (ESR 012) : Simple Native Interface
• Call Java world → C/asm
• Arguments: base types (int, float, double, char)
Easy Java → C Interface (Calls 2/2)

- Easy mapping using naming convention

```java
package GPIO;

public class Main {
    public static native void toggle();

    public static void main(String[] a) throws InterruptedException {
        while (true) {
            toggle();
            Thread.sleep(10);
        }
    }
}
```

```c
#include <sni.h>
#include "gpio.h"

void Java_GPIO_Main_toggle() {
    GPIOE->ODR ^= GPIO_Pin_2 ;
}
```
Easy Java ↔ C Interface (Data 1/2)

• SNI (ESR 012): Simple Native Interface
• Share arrays of base types
• Zero copy buffers and compatible with DMA systems
Java → C Interface

• Immortals are used to share data memory between Java and C

```java
package com.corp.examples;
public class Hello {

    static int[] array = (int[]) Immortals.setImmortal(new int[50]);
public static native int getData(int[] array);

public static void main(String[] args){
    int nb = getData(array);
}
}
```

```c
#include <sni.h>

jint Java_com_corp_examples_Hello_getData(jint* array){
    array[0] = 0xBEEF;
    return 1 ;
}
```
Shielded Plug for Safe & Easy C Integration

- Communication between two separated worlds (Java & native like C/asm)
- Pooling or notification event types
- Spatial & temporal decoupling
- Ideal to add Java tasks on top of a legacy C program

![Diagram showing Publishers and Subscribers with events streaming between them](image-url)
Shielded Plug Java Read Example

```xml
<shieldedPlug>
  <database name="Forecast" id="0" immutable="true" version="1.0.0">
    <block id="0" name="WIND" length="8" maxTasks="1"/>
    <block id="1" name="TEMP" length="4" maxTasks="1"/>
    <block id="2" name="THERMOSTAT" length="4" maxTasks="1"/>
  </database>
</shieldedPlug>
```

```java
public class Wind {
  public int speed; // in ms [0..]
  public int direction; // in degree [0..360]
}
```

```java
public class WindReader implements SPReader {
  private static final int SPEED = 0;
  private static final int DIRECTION = 4;
  public Object readObject(ShieldedPlug database, int blockID) throws EmptyBlockException {
    Wind w = new Wind();
    byte[] data = new byte[database.getLength(blockID)];
    database.read(blockID, data);
    w.speed = ByteArray.readInt(data, SPEED);
    w.direction = ByteArray.readInt(data, DIRECTION);
    return w;
  }
}
```
Shielded Plug C Publish Example

```c
#include <sp.h>

struct Wind {
    int32_t speed;
    int32_t direction;
};

void windPublication()
{
    struct Wind w;
    ShieldedPlug database = SP_getDatabase(Forecast_ID);
    w.speed = speed();
    w.direction = direction();
    SP_write(database, Forecast_WIND, &w);
}
```
Extend the Simulation Platform

• Why building your simulator?
  » Prototype before having hardware available

• Build your virtual device for UI
  » Front Panel Designer (buttons, LCD display, LEDs, etc.)

• Build your peripheral extensions (mocks)
  » Software mocks in Java or C connected to the simulation engine
  » Hardware mocks over workstation communication interfaces
public void runTest(Display display, String message) {
    MessageViewable viewable = new MessageViewable(display);
    viewable.init("Hello, world!");
    viewable.show();
}

public static void blink() {
    while (true) {
        toggle();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
        }
    }
}

public static final native boolean toggle();

public class LED {
    private boolean state;

    public static boolean toggle() {
        state = !state;
        if (state) {
            drawImage("ledOn.png");
        } else {
            drawImage("ledOff.png");
        }
        return state;
    }

    public class LED {
        private boolean state;

        public static boolean toggle() {
            state = !state;
            if (state) {
                drawImage("ledOn.png");
            } else {
                drawImage("ledOff.png");
            }
            return state;
        }
    }
APPLICATION PLATFORMS FOR
SMART OBJECTS (IOT)

MicroEJ - Hardware as a Service
HaaS for Home Energy Management

Dynamic Service Deployment

Service Provider

Remote terminal

Admin Platform

ARM Cortex M0+

ARM Cortex M4

3rd party services

(C) 2005-2014, all rights reserved.
HaaS for Wearable Electronics

- ARM Cortex M4
- ARM Cortex M0+
- Autonomous HaaS platform
IoT Market Challenges

• Energy efficiency
  » No bloatware!

• Cost Effectiveness
  » Small execution environments

• Rich Eco-Systems
  » More software enablers for innovative business models

• Reliability
  » Data integrity, service management

• Security
  » Virtualization, resource management
Solution Alignment

- Various topologies for gateways and edge devices
- Time-to-Market can not wait for specific system availability

Need unified and portable execution environments
HaaS Platform Overview

Application 1  Application n  MicroEJ HaaS

Edge Objects  Gateways  Cloud
New Capabilities

• Let marketing try new ideas
  » Try new services fast

• Share your platform with your Eco-System
  » Provide an open platform with safe isolation capability

• Let your customer choose a product configuration
  » In the field dynamic service deployment and activation

• Keep using your legacy device base
  » Use ubiquitous technology with low constraints on hardware
MicroEJ® Haas Architecture – Kernel

- Standalone (independent from Features)
- Manage Features
  - Life cycle
  - Resource allocation
- Native code allowed
MicroEJ® HaaS Architecture – Features

- Rely on Kernel APIs
- Cannot directly access to other Features (code, objects, threads) → use Kernel as a proxy instead
- Full virtualization (no native code allowed)
K&F Key Features

• **Low consumption & OS agnostic**
  » Kernel & Features: ~20KBytes
  » Run the same on any RTOS

• **Ressources management**
  » CPU and memory allocations
  » All I/O: file system, TCP/IP, UART, USB, etc.

• **Stable & Secure**
  » Kill of a Feature (group of bundles) feasible at any time
    ▪ Threads + objects + code
  » No back door
K&F and OSGi

• Bundle life cycle management
  » Load/unload, enable/disable

• Resource management
  » Bundles cannot access to larger CPU and memory resources than required
  » Bundles cannot access to physical resource when not allowed to

• Isolation
  » Bundles is isolated from each others and interface according to the rules defined by the Kernel

• Stable & Secure
  » Unload has no impact on other Bundles
  » No stale reference, no zombie threads, etc.
THANK YOU!

More information: [www.is2t.com](http://www.is2t.com)
Evaluation kits: [is2t.microej.com](http://is2t.microej.com)