UNIVERSIDAD DE BUENOS AIRES

Facultad de Ingeniería

Carrera de Especialización en Sistemas Embebidos

Desarrollo de Firmware y Software para programar la CIAA en lenguaje JAVA con aplicación en entornos Industriales

Alumno: Ing. Eric Nicolás Pernia.

Director: MSc. Ing. Félix Gustavo Safar.

Buenos Aires, Argentina.

Presentación:
Noviembre de 2015

RESUMEN

El propósito de este Trabajo Final es la incorporación de nuevas tecnologías en ambientes industriales mediante el desarrollo de arquitecturas de sistemas embebidos novedosas. En particular, permitir crear aplicaciones Real-Time para entornos industriales, utilizando un lenguaje de programación orientado a objetos (POO), sobre la Computadora Industrial Abierta Argentina (CIAA). Además, se espera acercar a programadores informáticos a la disciplina de programación de sistemas embebidos, permitiéndoles aplicar técnicas avanzadas de programación.

Para llevarlo a cabo se ha escogido Java como lenguaje POO y Icecaptools. Icecaptools es un plug-in de Eclipse realizado por Stephan Erbs Korsholm, que convierte al Eclipse en un IDE para la programación en lenguaje Java y permite compilar el programa de usuario para HVM. HVM es una máquina virtual de Java, libre, escrita en lenguaje C, diseñada para correr directamente sobre el hardware. Además, HVM cumple con la especificación Safety-Critical Java (SCJ), permitiendo realizar aplicaciones críticas Real-Time.

Se incluye en este trabajo la implementación y validación de un ambiente de Firmware y Software, basado en Icecaptools y HVM, para programar la Computadora Industrial Abierta Argentina (CIAA) en lenguaje Java, para aplicaciones industriales en tiempo real. Esta consiste en:

- La realización del *porting* de HVM para que corra sobre el microcontrolador NXP LPC4337 que contienen las plataformas CIAA-NXP y EDU-CIAA-NXP.
- Un diseño e implementación de una HAL con API sencilla para permitir controlar el Hardware desde una aplicación Java.
- La realización del *porting* de la capa SCJ de HVM para que corra sobre el microcontrolador NXP LPC4337 permitiendo aplicaciones SCJ.
- El desarrollo de la integración en Icecaptools del *porting* de HVM para completar el IDE de Java SCJ sobre la CIAA.

Para validar el IDE desarrollado se presentan:

- Ejemplos de aplicaciones Java utilizando periféricos de la CIAA y EDU-CIAA.
- Una aplicación Real-Time (SCJ) de referencia para demostrar el funcionamiento real-time del sistema.

Agradecimientos

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

INTRODUCCIÓN GENERAL

1.1. Marco temático Programación Orientada a Objetos en Sistemas embebidos para aplicaciones industriales

Marco temático: Programación Orientada a Objetos en Sistemas embebidos para aplicaciones industriales.

1.2. Plataforma CIAA

Plataforma CIAA-NXP.
Plataforma EDU-CIAA-NXP.

1.3. Lenguaje Java

Se elije Java debido a que es uno de lenguajes de POO más utilizados en la actualidad. Es un lenguaje moderno, con manejo de memoria automático y sintaxis similar a C++. A diferencia de otros, en lugar de compilarse para la arquitectura del controlador objetivo, se compila a un lenguaje similar al assembler (bytecodes) que se ejecuta mediante una Máquina Virtual, o VM, por sus siglas en inglés (Virtual Machine). Esta máquina virtual corre habitualmente sobre un sistema operativo.

1.4. Máquinas Virtuales de Java para sistemas embebidos

Máquinas Virtuales de Java para sistemas embebidos.

1.5. Especificación SCJ

Especificación SCJ.

1.6. Justificación

Con la creciente complejidad de las aplicaciones a realizar sobre sistemas embebidos en entornos industriales, y el aumento de las capacidades de memoria y procesamiento de los mismos, se desea poder aplicar técnicas avanzadas de programación para realizar programas fácilmente mantenibles, extensibles y reconfigurables. El paradigma de Programación Orientada a Objetos (POO) cumple con estos requisitos.

Para aplicaciones industriales es requerimiento fundamental cumplir con especificaciones de tiempo real. Es por esto que se debe elegir un lenguaje POO que soporte de manejo de threads real-time. Ejemplos de aplicaciones con estos requerimientos son, control a lazo cerrado, etc.

1.7. Objetivos

Los objetivos del presente trabajo final son:

- Realizar el *porting* de HVM para que corra sobre el microcontrolador NXP LPC4337 que contienen las plataformas CIAA-NXP y EDU-CIAA-NXP para permitir correr aplicaciones Java sobre la CIAA.
- Diseñar e implementar una HAL para controlar el Hardware desde una aplicación Java mediante una API establecida.
- Realizar el *porting* de la capa SCJ de HVM para que corra sobre el microcontrolador NXP LPC4337 permitiendo aplicaciones Java SCJ sobre la CIAA.
- Integrar en Icecaptools el *porting* de HVM para completar el IDE de Java SCJ sobre la CIAA.

Capítulo 2

DISEÑO

2.1. Elección de la VM de Java, HVM

HVM son las siglas de *Hardware Virtual Machine*. Es una máquina virtual (en adelante VM) de Java, libre, diseñada para ejecutarse *bare-metal*[1] sobre microcontroladores y portable a diferentes arquitecturas. Su implementación de referencia fue realizada por Stephan Erbs Korsholm (Icelab, Dinamarca) para su tesis doctoral y puede ejecutarse sobre Posix o AVR. Esta VM cumple con la especificación Safety-Critical Java (SCJ) Level 0, 1 y 2 permitiendo realizar aplicaciones en tiempo real para sistemas críticos. Provee tres formas de acceso al hardware:

- **Variables Bindeadas.** Es una variable que puedo utilizar en lenguaje Java y tiene correspondencia directa con una variable en otro lenguaje (en este caso particular, lenguaje C).
- ***Hardware Objects.*** Es una abstracción que permite acceder a registros del microcontrolador mapeados en memoria para manipularlos desde el programa en lenguaje Java. De esta forma se puede crear una biblioteca completa dependiente del microcontrolador que maneje un periférico a nivel de registros directamente en Java.
- **Métodos nativos.** Esta alternativa permite utilizar funciones realizadas en otro lenguaje como métodos en lenguaje Java. De esta manera permite ejecutar código *legacy* dando la posibilidad de utilizar bibliotecas completas realizadas en otro lenguaje. Para conectar un método con una función en lenguaje C, deben respetarse ciertan convenciones de nombres y de pasajes de parámetros en las funciones realizadas para que el compilador de Java puede asociarlas.

Se elije métodos nativos como alternativa para proveer al programa de usuario en lenguaje Java el acceso a los periféricos del microcontrolador. Esto es porque ya existen bibliotecas completas de manejo de periféricos y además *stacks* y *file systems* entre otras.

2.2. IDE Icecaptools

Icecaptools se distribuye como un plugin de Eclipse realizado por Stephan Erbs Korsholm, que convierte al Eclipse en un IDE para la programación en lenguaje Java y permite compilar el programa de usuario para HVM. Funciona realizando una traducción de un programa de usuario escrito en lenguaje Java, a un programa en lenguaje C que incluye el código de dicho programa y el código C generado de HVM. De esta manera, logra portabilidad entre diferentes microcontroladores y permite integración con programas escritos previamente en lenguaje C, como por ejemplo, el firmware de la CIAA. Requiere un toolchain de lenguaje C, para el microcontrolador objetivo, que permita compilar el código, generando un binario ejecutable, y su posterior descarga a dicho dispositivo. En la figura [2.1] se incluye un esquema gráfico de su funcionamiento.

[1] Significa que se ejecuta directamente sobre el hardware, sin necesidad de un sistema operativo.

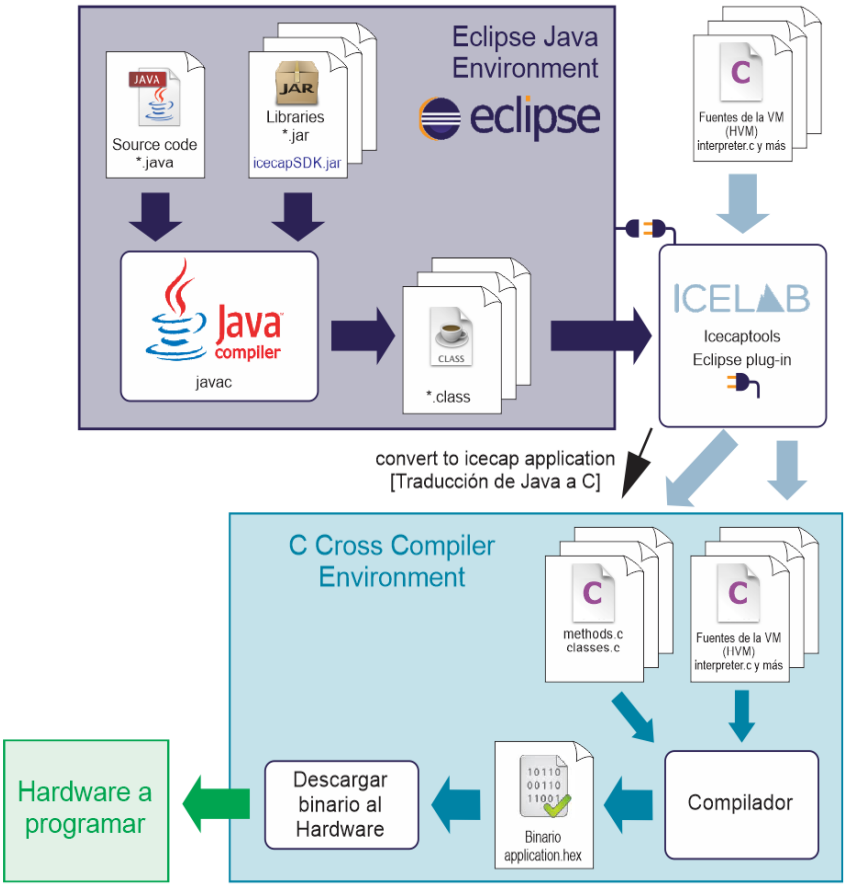

Figura 2.1: Esquema de funcionamiento de Icecaptools.

2.3. Porting de HVM e icecaptools a una nueva arquitecura

Para portar el Firmware HVM a una nueva arquitectura deben realizarse los siguientes pasos:

- Conseguir un ambiente de desarrollo en lenguaje C para la arquitectura.
- Estandarizar los tamaños de tipos de datos.
- Implementar una HAL para la arquitectura.
- Implementar el acceso al hardware.

Para facilitar la inclusión de nuevas arquitecturas a icecaptools y automatizar tareas habituales, que en caso contrario debería realizar el usuario para cada nueva aplicación, se propone:

- Adaptar Icecaptools para la arquitectura ARM mediante su extensión a través de un plug-in que llamaremos HvmForArmCortexM.
- La programación de un segundo plug-in más específico llamado HVM4CIAA.

Finalmente, en la figura 2 se gráfica la solución propuesta para lograr utilizar icecaptools y HVM con la CIAA.
FIGURAAAAAAAAAAA

2.4. Diseño de HAL para manejo de periféricos

Diseño de una HAL para manejo de periféricos con una API sencilla.

2.5. Diseño del IDE

Diseño del IDE.

Capítulo 3

IMPLEMENTACIÓN

3.1. Paradigmas y lenguajes de programación utilizados

Paradigmas y lenguajes de programación utilizados.

3.2. Entorno Eclipse

Entorno Eclipse.

3.3. Port de HVM (Java básico) para CIAA-NXP

Port de HVM (Java básico) para CIAA-NXP.

3.4. Implementación de una HAL para manejo de periféricos CIAA-NXP

Implementación de HAL para manejo de periféricos CIAA-NXP.

3.5. Ejemplos de uso de periféricos desarrollados

Ejemplos de uso de periféricos desarrollados.

3.6. Port de HVM (Java SCJ) para CIAA-NXP

Port de HVM (Java SCJ) para CIAA-NXP.

3.7. Ejemplo de aplicación de tiempo real desarrollado

Ejemplo de aplicación de tiempo real desarrollado.

3.8. Plugins desarrollados

Plugins desarrollados.

3.9. Implementación del IDE

Implementación del IDE.

Capítulo 4

RESULTADOS

4.1. Acerca del IDE obtenido

Acerca del IDE obtenido.

4.2. Comparaciones entre Java y C

4.3. Aplicación de referencia SCJ

Aplicación de referencia SCJ.

Se ha completado el porting de HVM e Icecaptools para la CIAA y se ha creado un conjunto de clases para manejo básico del hardware desde el programa en lenguaje Java. De esta manera puede crearse una aplicación en lenguaje Java para las plataformas CIAA-NXP y EDU-CIAA-NXP, permitiendo controlar sus periféricos.

Enumerar las clases desarrolladas

Como subproducto, se obtiene además un nuevo módulo de Firmware para la CIAA nombrado sAPI (simple API) que permite encapsular los periféricos y con una interfaz muy sencilla pensada para el programador con perfil informático no experto en Sistemas embebidos.

Enumerar el hardware que permite atacar

Se ha logrado portar además portar la capa de HVM que implementa la especificación SCJ obteniendo una plataforma de firmware y software que permite la programación de aplicaciones Real-Time para la CIAA en lenguaje Java.

Para probar el correcto funcionamiento de las herramientas, ha realizado una aplicación de referencia con especificadores de tiempo real y se ha comprobado su correcto funcionamiento.

Poner diagramas temporales de mediciones en ejecución?

Capítulo 5

CONCLUSIONES Y TRABAJO A FUTURO

5.1. Conclusiones

En este trabajo se ha logrado llevar a cabo...

5.2. Trabajo a futuro

Como labor a futuro, pueden realizarse las siguientes tareas:

- fdfdf
- fdfdf

REFERENCIA BIBLIOGRÁFICA

1. Stephan E. Korsholm (2014). *The HVM Reference Manual*. Switzerland: Square Bracket Associates. Consultado en 05-06-2015 en http://http://icelab.dk/resources/HVMRef.pdf.