

Sistemas Empotrados

Tema 1:

Introducción a los sistemas empotrados

Lección 4:

La parte software



Contenidos

Tema 1: Introducción a los sistemas empotrados

Presentación de la asignatura

- Motivación

- Descripción de la asignatura

Sistemas empotrados

- Utilidad

- Caracterización

- Clasificaciones

- Diseño e implementación

- Herramientas de desarrollo

La parte hardware

- Procesadores

- Co-procesadores y aceleradores

- Controladores de sistema

- Arquitectura de memoria

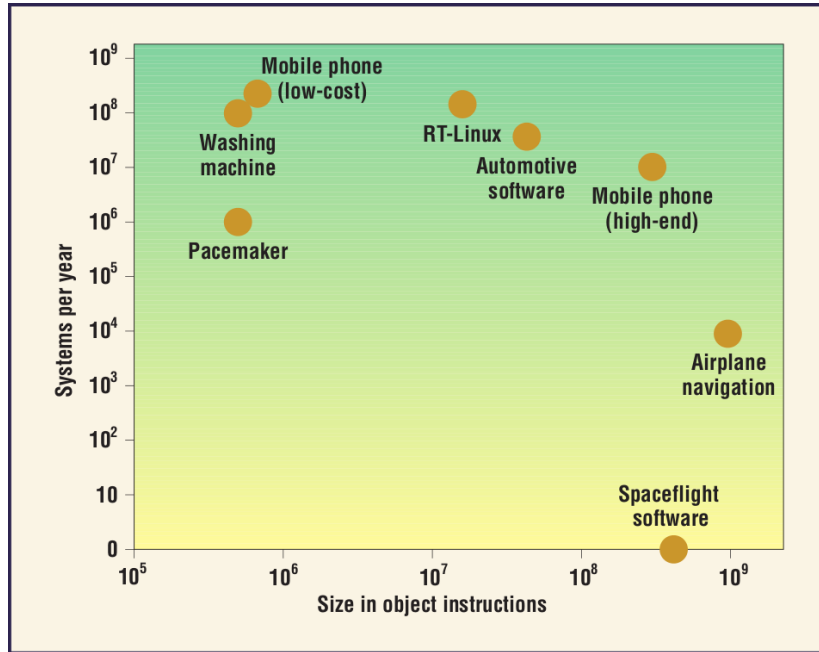
- Periféricos

La parte software

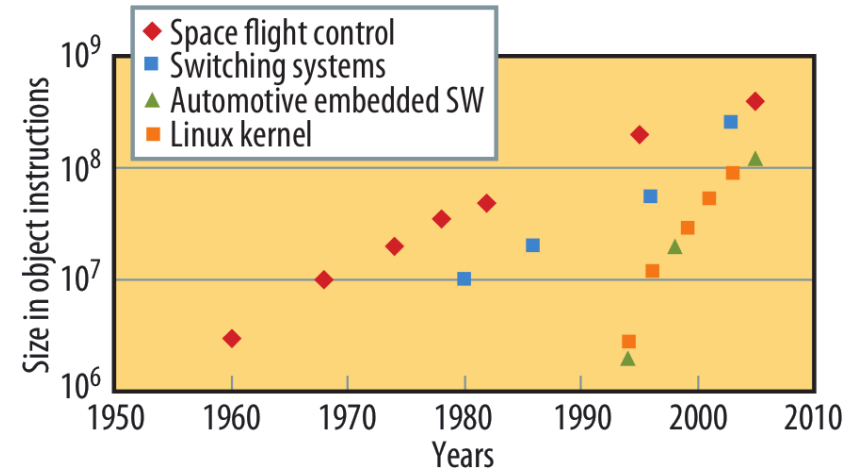
- Importancia creciente del software empotrado

- Componentes del Firmware

Importancia creciente del software empotrado



El tamaño del software y el número de unidades vendidas depende del tipo de sistema empotrado



Crecimiento de la complejidad del software empotrado

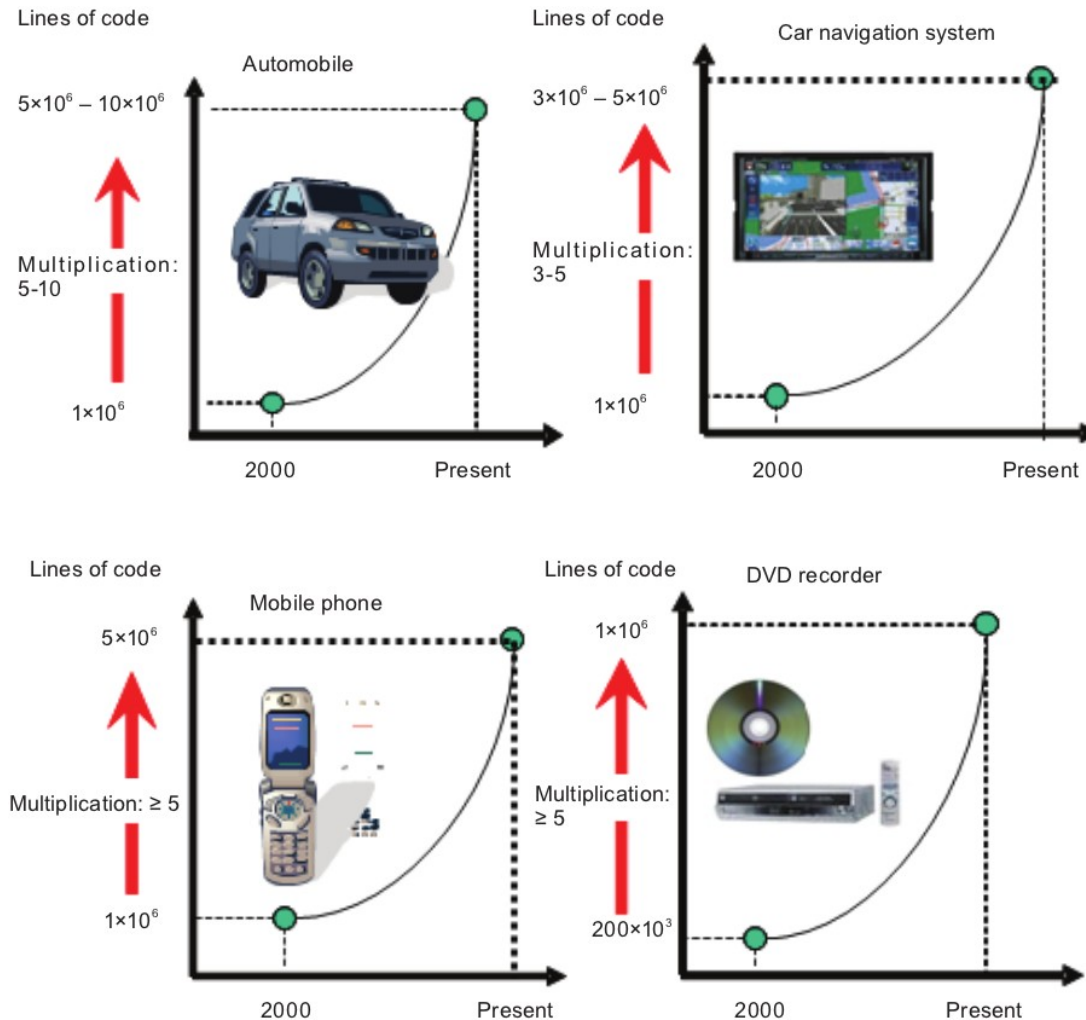
La complejidad y el tamaño del software empotrado crece cada año

Fuente:

C. Ebert y C. Jones. Embedded Software: Facts, Figures and Future. Computer, 42(4), 42-52, 2009

C. Ebert y J. Salecker. Embedded Software – Technologies and Trends. IEEE Software, 26(3), 14-18, 2009

Importancia creciente del software empotrado

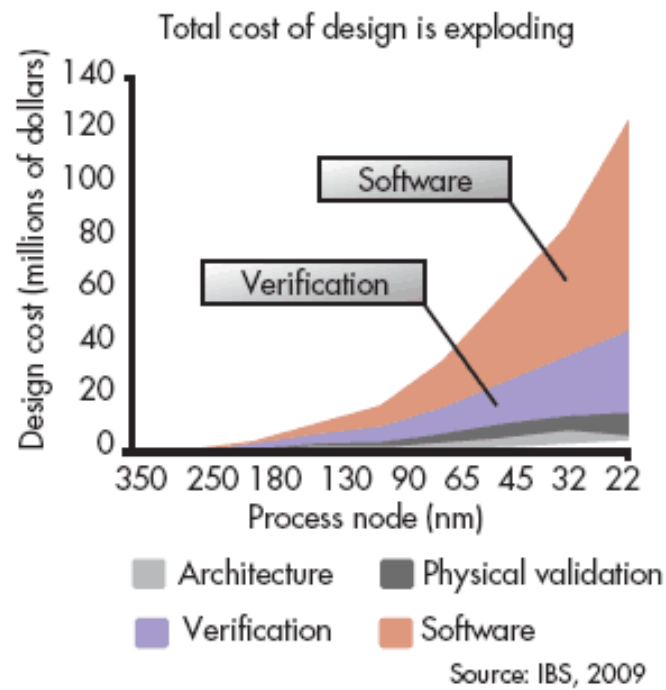


El número de funciones implementadas en software es cada vez mayor

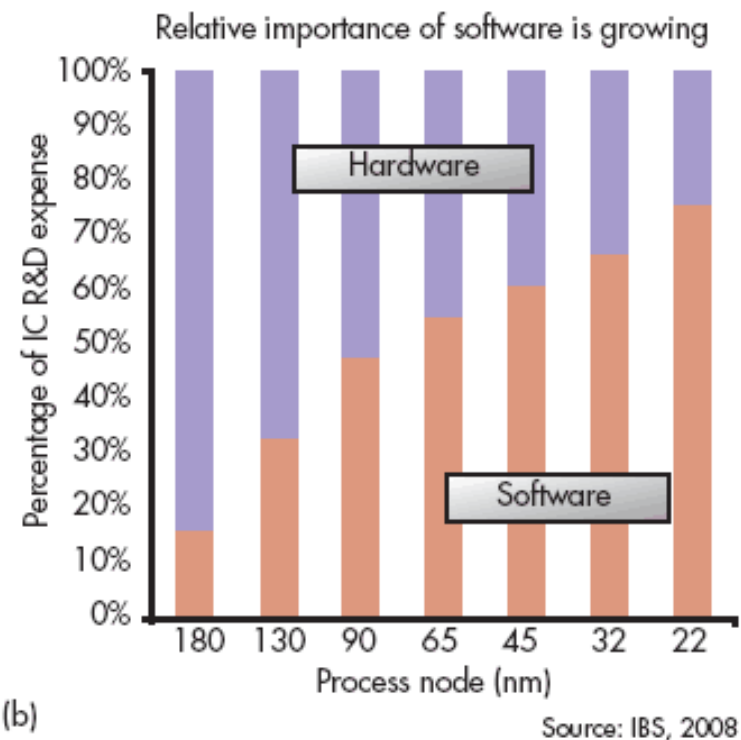
Fuente:

M. Tachikawa. The Direction of Embedded Software Development: Focusing on Japan's Social Characteristics – Reinforcing the basis for software development in electronics-driven durable goods. Science & Technology Trends – Quarterly Review, 42:36-49, enero 2012.

Importancia creciente del software empotrado



(a)



(b)

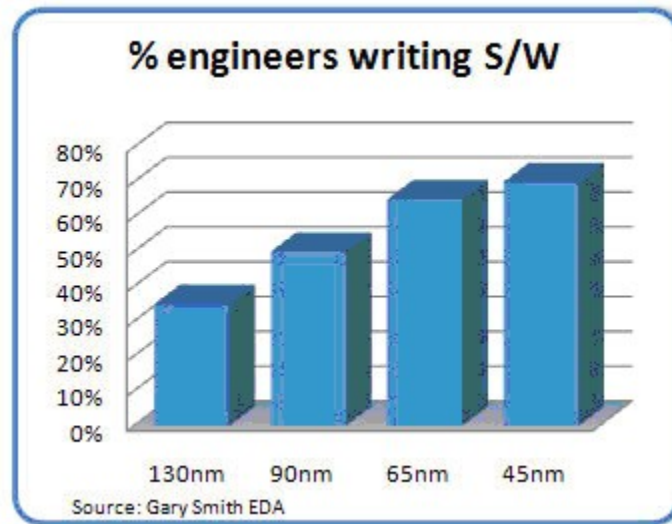
El software cada vez representa un mayor porcentaje en el coste total del sistema

Fuente:

D. Maliniak. Verification And Software Dominate EDA's Future. ElectronicDesign.com, enero 2010.

http://electronicdesign.com/article/eda/verification_and_software_dominate_eda_s_future

Importancia creciente del software empotrado



La demanda de ingenieros que sepan desarrollar software empotrado está creciendo notablemente

Fuente:
N. Songcuan. Accelerating Embedded Software Development with Rapid Prototyping. Chip Design.
<http://chipdesignmag.com/display.php?articleId=3598>

Contenidos

Tema 1: Introducción a los sistemas empotrados

Presentación de la asignatura

- Motivación

- Descripción de la asignatura

Sistemas empotrados

- Utilidad

- Caracterización

- Clasificaciones

- Diseño e implementación

- Herramientas de desarrollo

La parte hardware

- Procesadores

- Co-procesadores y aceleradores

- Controladores de sistema

- Arquitectura de memoria

- Periféricos

La parte software

- Importancia creciente del software empotrado

- Componentes del Firmware

Concepto de firmware



Almacenamiento



Memoria ROM/Flash

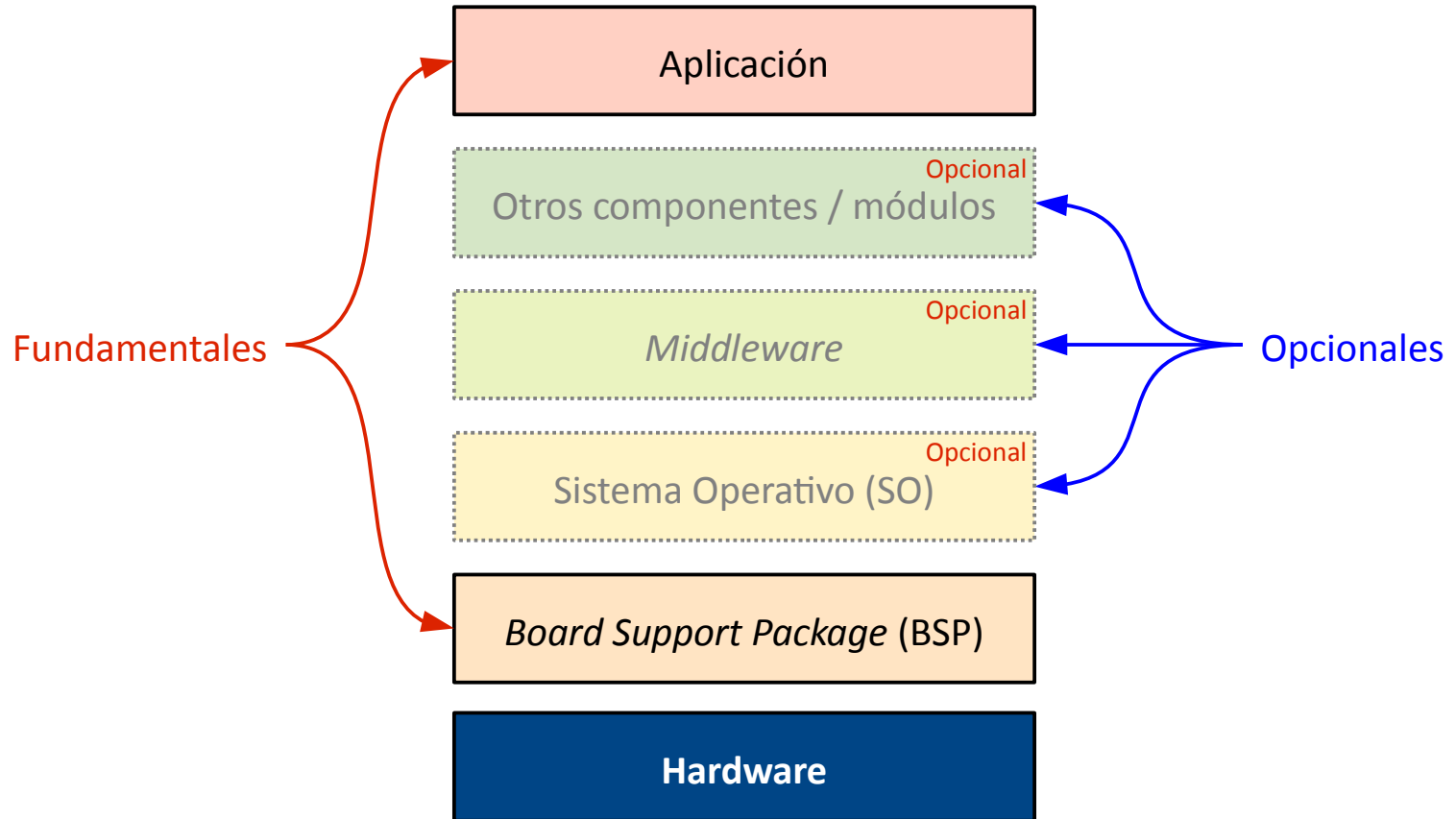
El término **firmware** se usa comúnmente para denominar la parte software de un sistema emputrado



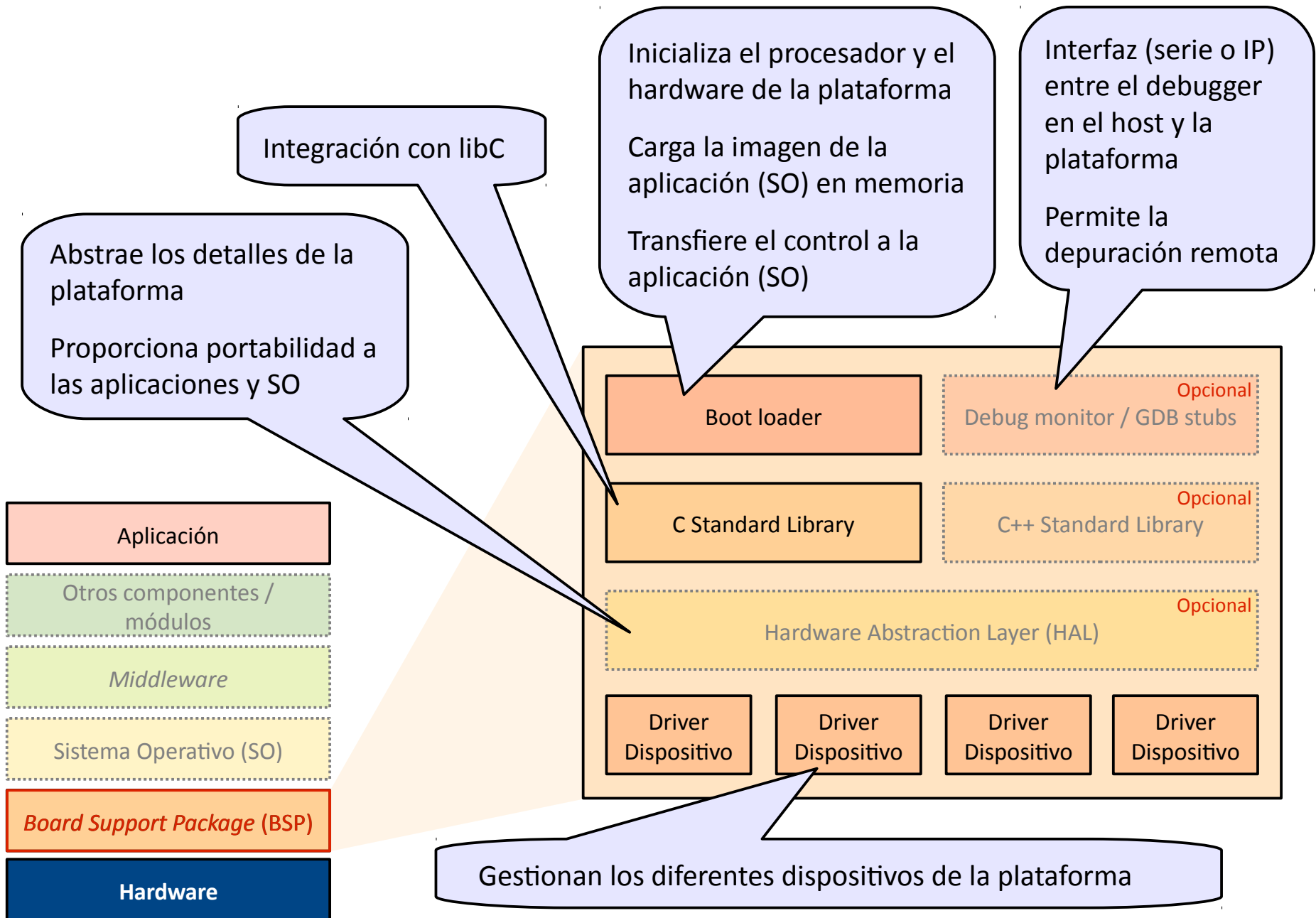
Ejecución



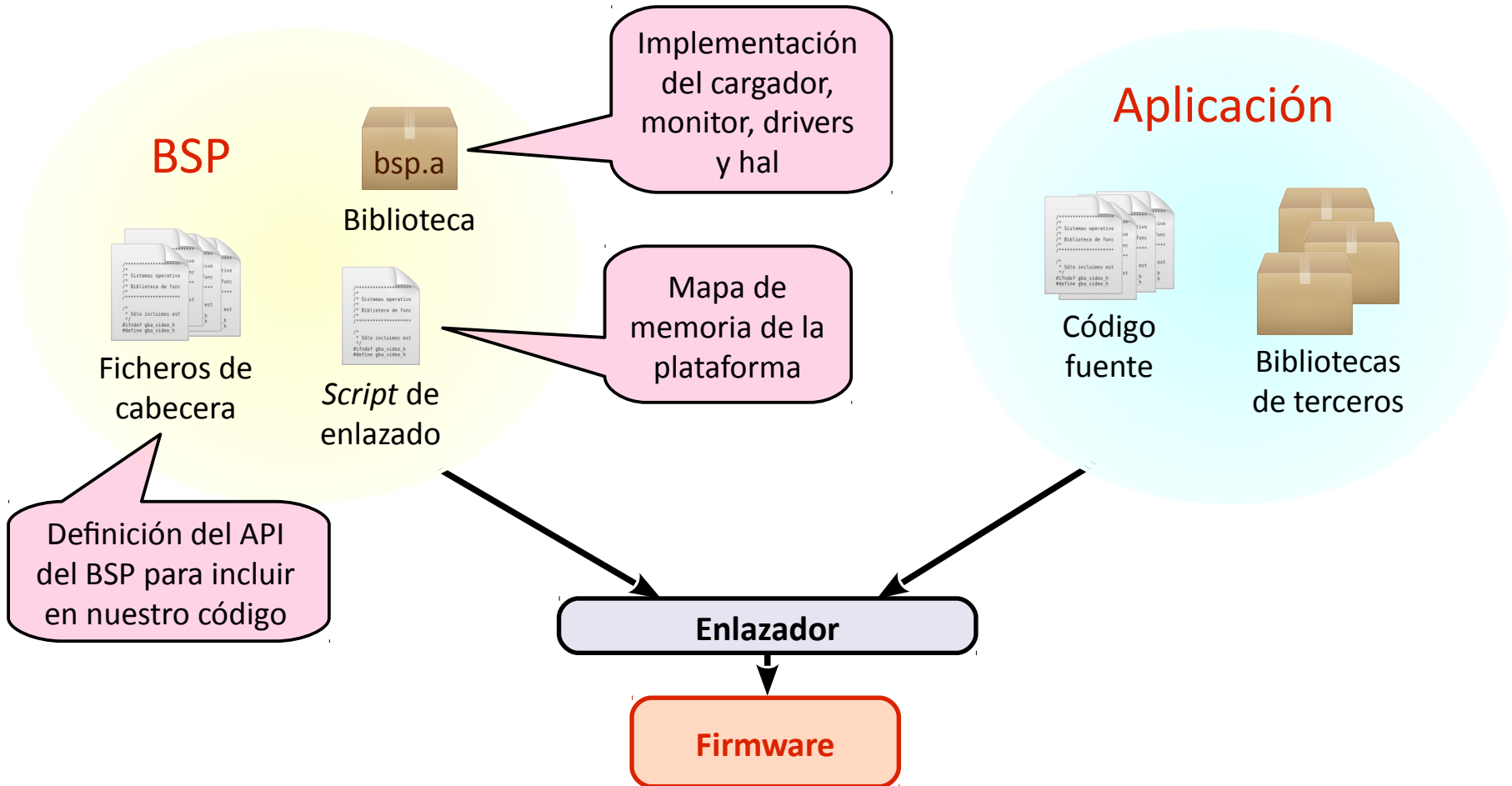
Componentes del firmware



Componentes del *Board Support Package* (BSP)



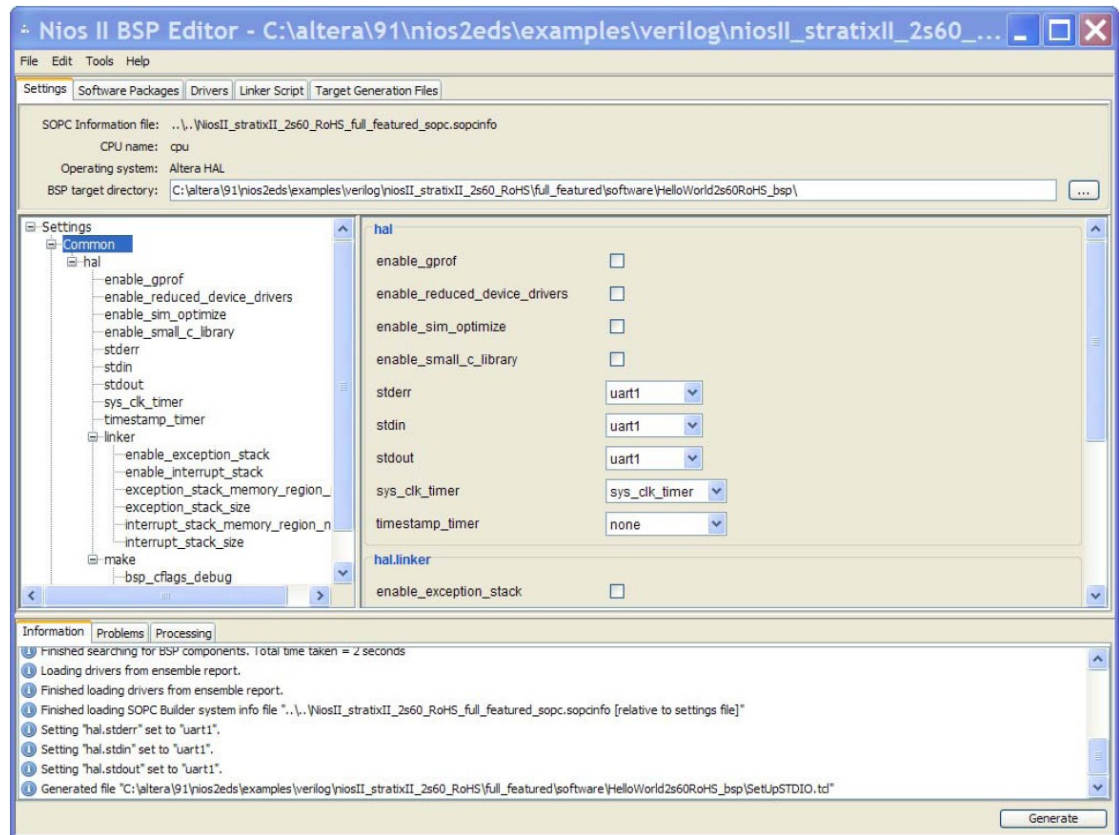
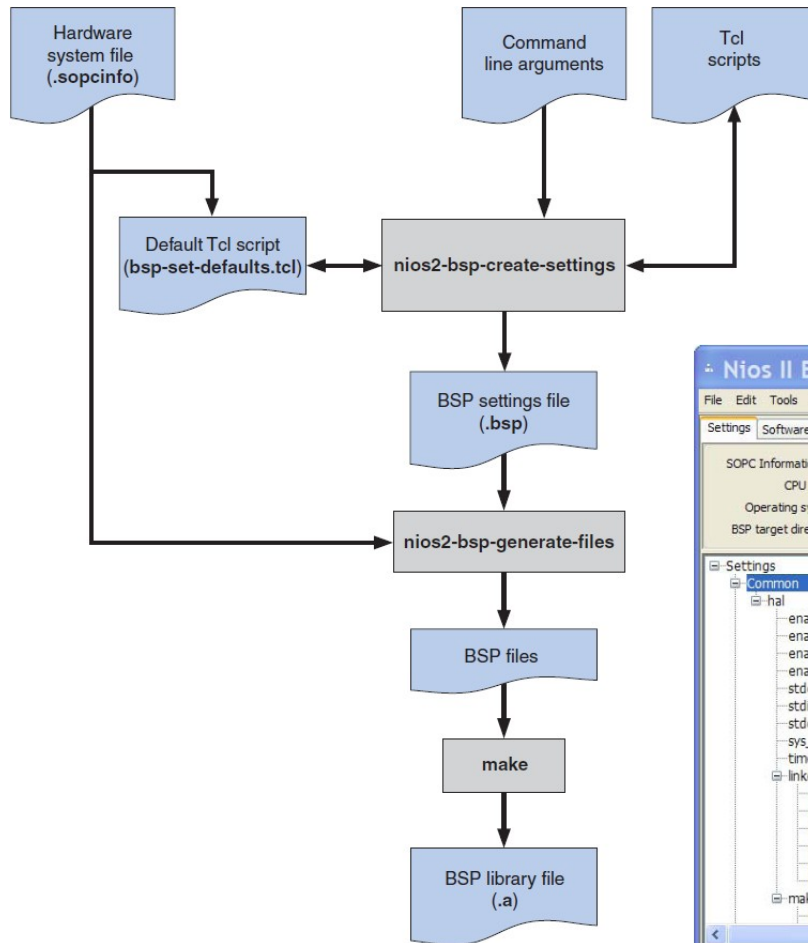
El BSP a efectos prácticos



Dependiendo de la plataforma, el BSP puede ser proporcionado por el fabricante, generado automáticamente o programado completamente desde cero

Ejemplo

Generación automática del BSP
para un SoC basado en el procesador
Nios II de Altera a partir de la
información del sistema



Fuente:

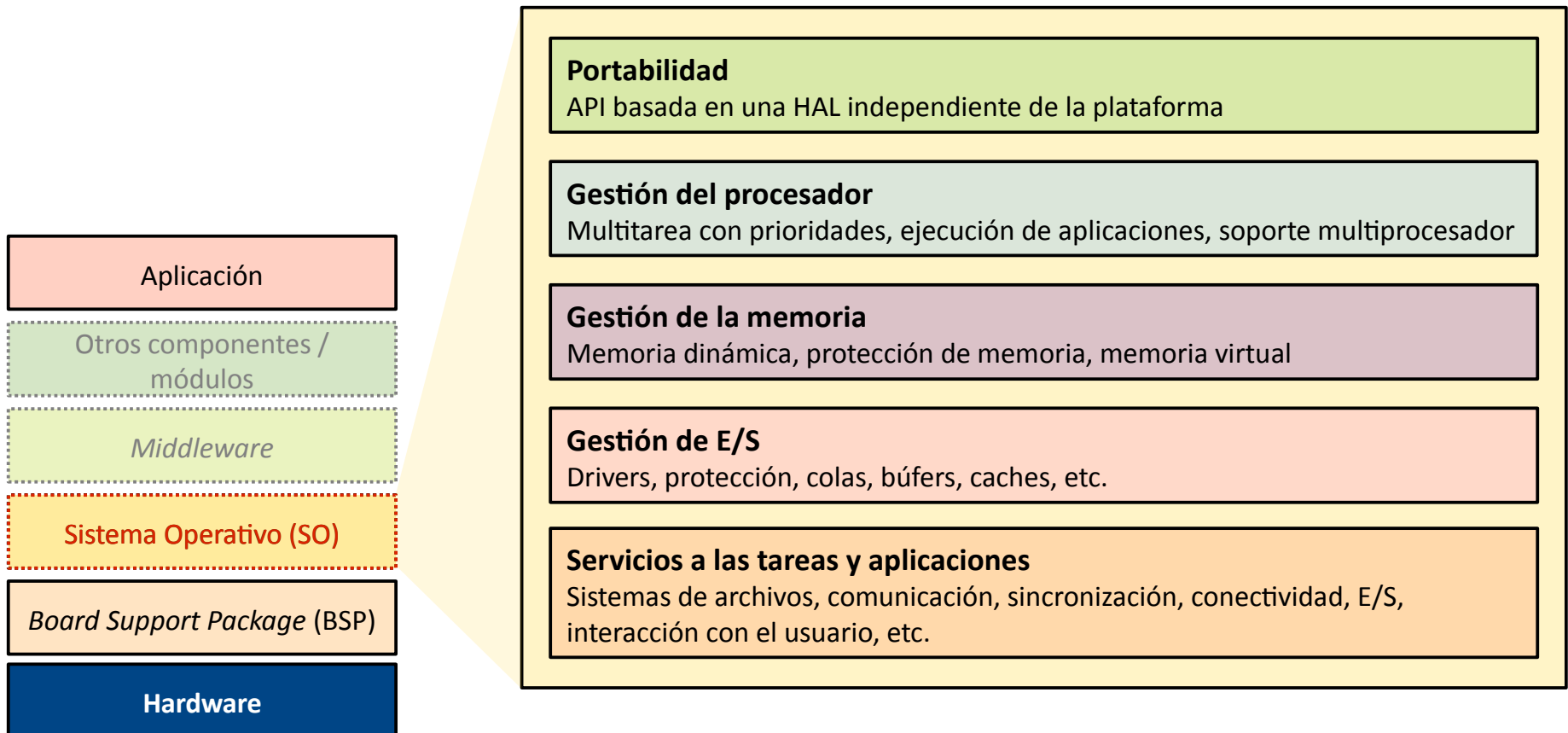
Altera. Nios II Software Developer Handbook. http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf

Necesidad de un Sistema Operativo

Algunos sistemas empotrados necesitan más servicios de los que proporciona el BSP

Se debe seleccionar un SO adecuado para el sistema (RTOS o GPOS)

El SO se configura para soportar sólo los servicios necesarios



Real Time Operating Systems (RTOS)

Objetivo: Reaccionar ante ciertos estímulos dentro de un intervalo de tiempo determinado

Características:

Fiabilidad: Funcionamiento correcto en el tiempo

Determinismo: Comportamiento predecible en el tiempo

Rendimiento: Dadas las restricciones en potencia de cálculo

Compacidad: Dadas las restricciones en la cantidad de memoria disponible

Escalabilidad: Capacidad de adaptación a los requerimientos de cada aplicación
(desde un DVD hasta un avión)

Componentes fundamentales:

Gestión de tareas (cambio de tarea, gestión de la pila de cada tarea, ...)

Planificación de tareas (determinista y con prioridades)

Intercomunicación de tareas (buffers, mailboxes, colas de mensajes, ...)

Sincronización de tareas (regiones críticas, semáforos, eventos, ...)

Gestión de memoria y E/S (protección → **MPU**, memoria dinámica, ...)

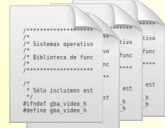
Aplicaciones: Multimedia, automoción, telecomunicaciones, dispositivos médicos, ...

Uso de un RTOS

RTOS



Código fuente



Scripts de configuración

Opcional

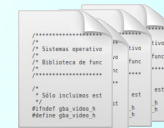


BSP

A veces puede ir incluido en el RTOS

Un RTOS es una biblioteca que proporciona los servicios que necesita nuestra aplicación

Aplicación



Código fuente



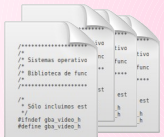
Bibliotecas de terceros

Configuración y construcción

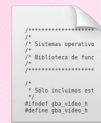
RTOS



Biblioteca



Ficheros de cabecera



Script de enlazado

BSP, Procesos, comunicación, sincronización, etc.

Enlazador

Firmware

Dependiendo de la plataforma y del RTOS puede que ya esté portado a nuestra plataforma o que tengamos que portarlo

General Purpose Operating Systems (GPOS)

Objetivo: Permitir la ejecución de múltiples aplicaciones / widgets

Características:

Memoria virtual → **MMU** (muchos más procesos que un RTOS, más escalable)
Soporte multiprocesador, red, sistemas de ficheros, seguridad, multilengua, etc.
Múltiples lenguajes de programación (Java, Python, Php, Perl, Ruby, TCL, etc.)
Navegador Web, multimedia e interfaz al usuario ya integradas
Interacción multimodal (Ventanas, 3D, multimedia, voz, gestual, etc.)
Conectividad: TCP/IP, Wi-Fi, Bluetooth, 3G, 4G, NFC, ...
Disponibilidad de muchos más drivers y bibliotecas de código
Múltiples distribuidores de software (*open source, markets*, etc.)
Demanda de más memoria y más potencia de cálculo

Ejemplos:

Google Android, Apple iOS, Nokia Symbian, Microsoft Windows 7, Montavista Linux

Aplicaciones:

Smartphones, Tablets, Smart TVs, e-Book readers, routers, Portable Media Players (PMP), ...

Uso de un GPOS

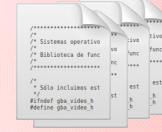
A veces puede ir incluido en el GPOS

BSP

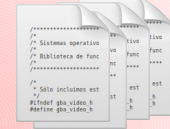
Opcional

Port

GPOS



Código fuente



Scripts de configuración

Las aplicaciones se instalan por el usuario una vez que el GPOS (firmware) está instalado en el sistema

Firmware

Configuración y construcción

Instalación

Aparecen los mercados de aplicaciones

Origen 1



Aplicaciones

Origen 2



Aplicaciones

...

Origen n



Aplicaciones

Uso combinado de RTOS y GPOS

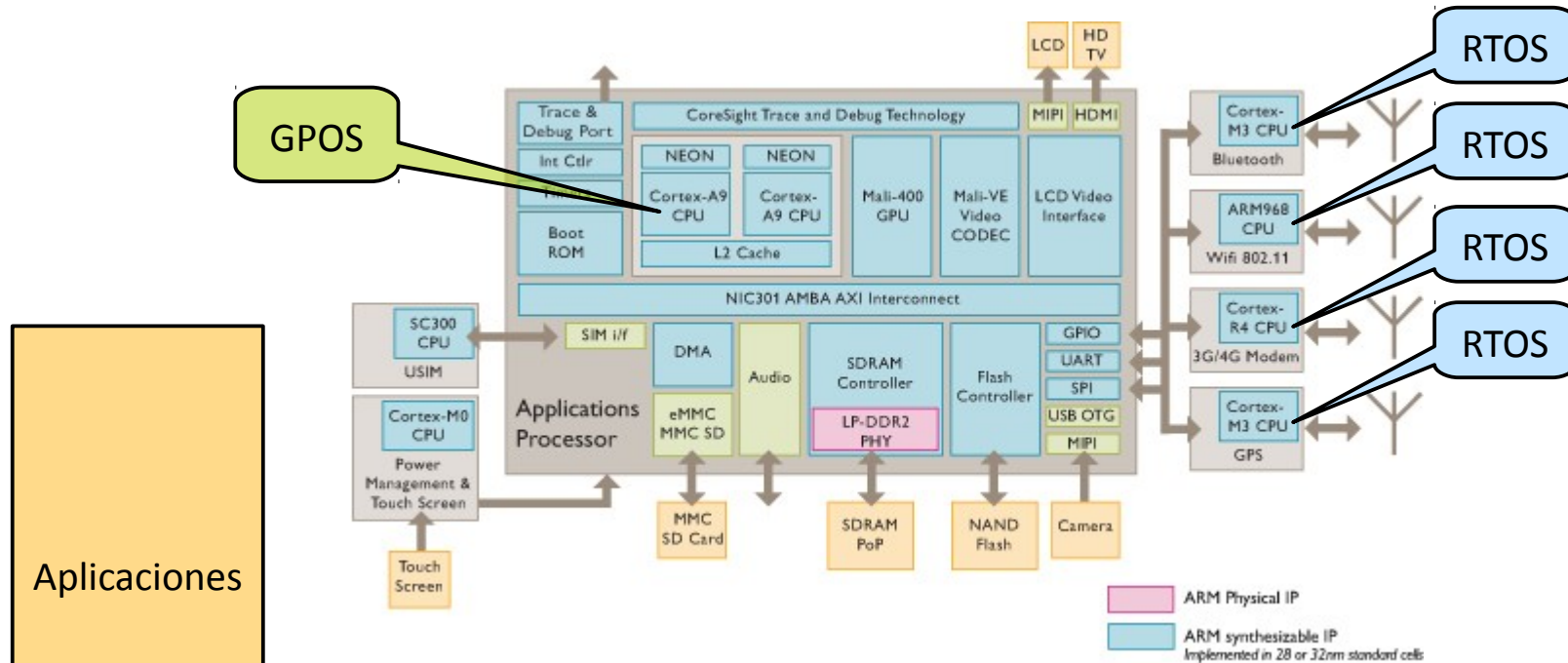
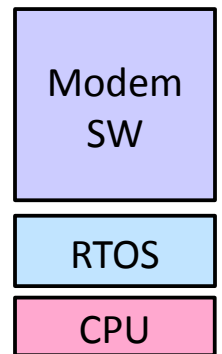
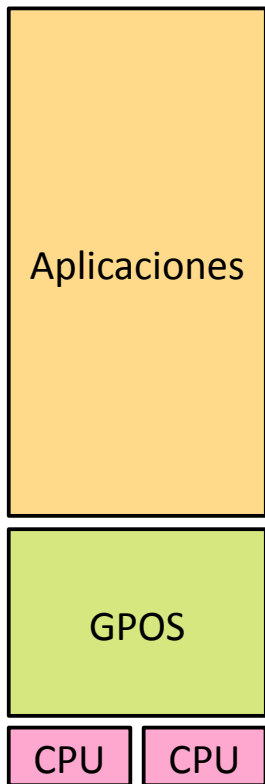
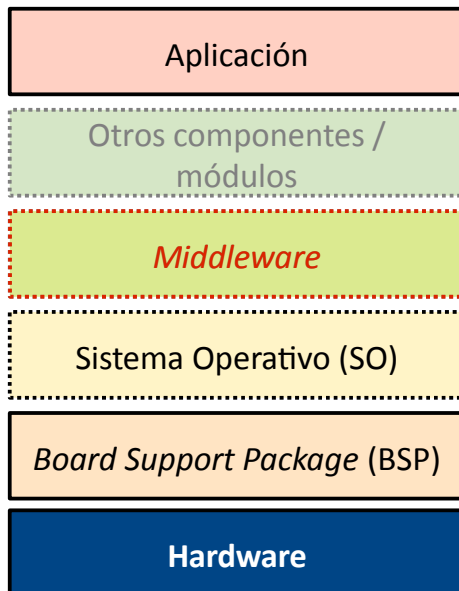


Diagrama de bloques de un smartphone



Necesidad de un *Middleware*

Proporciona interoperabilidad en los sistemas empotrados en red o distribuidos



Posibilita las comunicaciones

Algunos RTOS carecen de drivers y pilas de protocolos

Independencia de los detalles de las comunicaciones

El cliente ignora la existencia de servidores, protocolos, puertos, redes, seguridad, etc. Sólo hace llamadas a funciones/objetos locales

Independencia del lenguaje

Cliente y servidor pueden estar escritos en diferentes lenguajes

Independencia del SO

Los cambios de SO en cliente o servidor no afectan al código de la aplicación

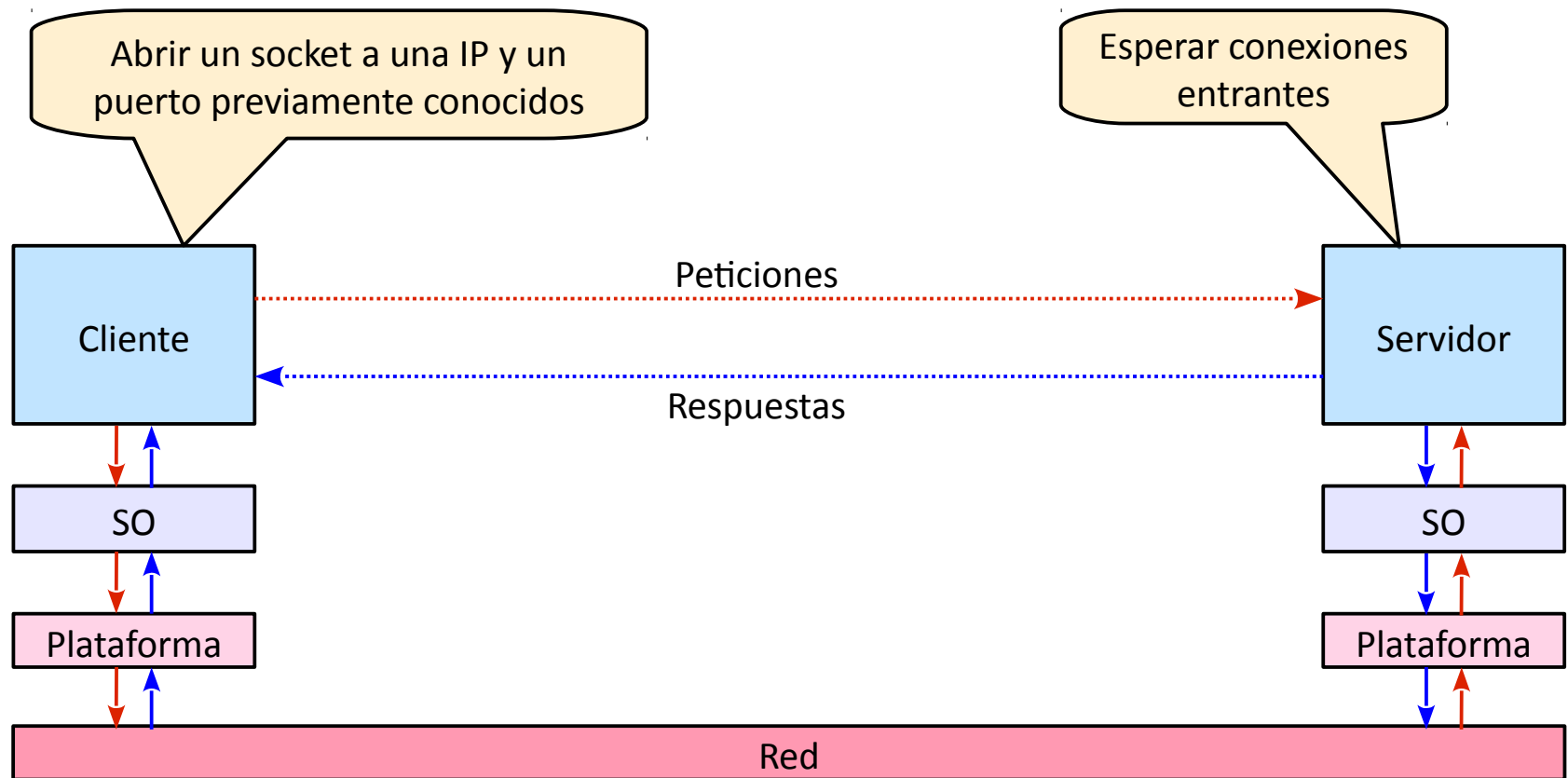
Independencia de la implementación

Los cambios en la implementación de los servicios no afectan al cliente

Dependencia de la interfaz de los servicios

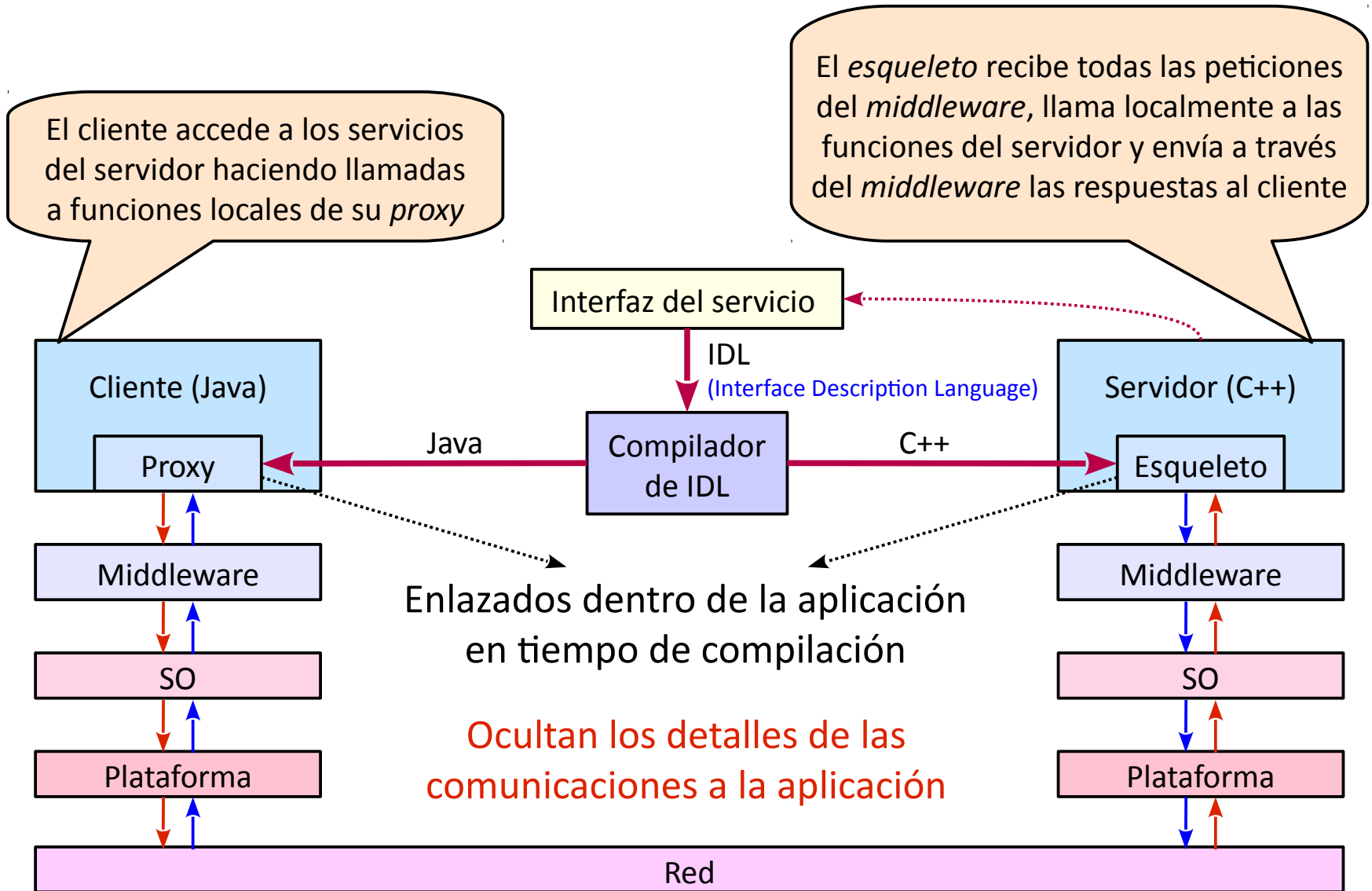
El cambio de la interfaz sí que afecta a los clientes que usan los servicios

Comunicaciones sin *middleware*



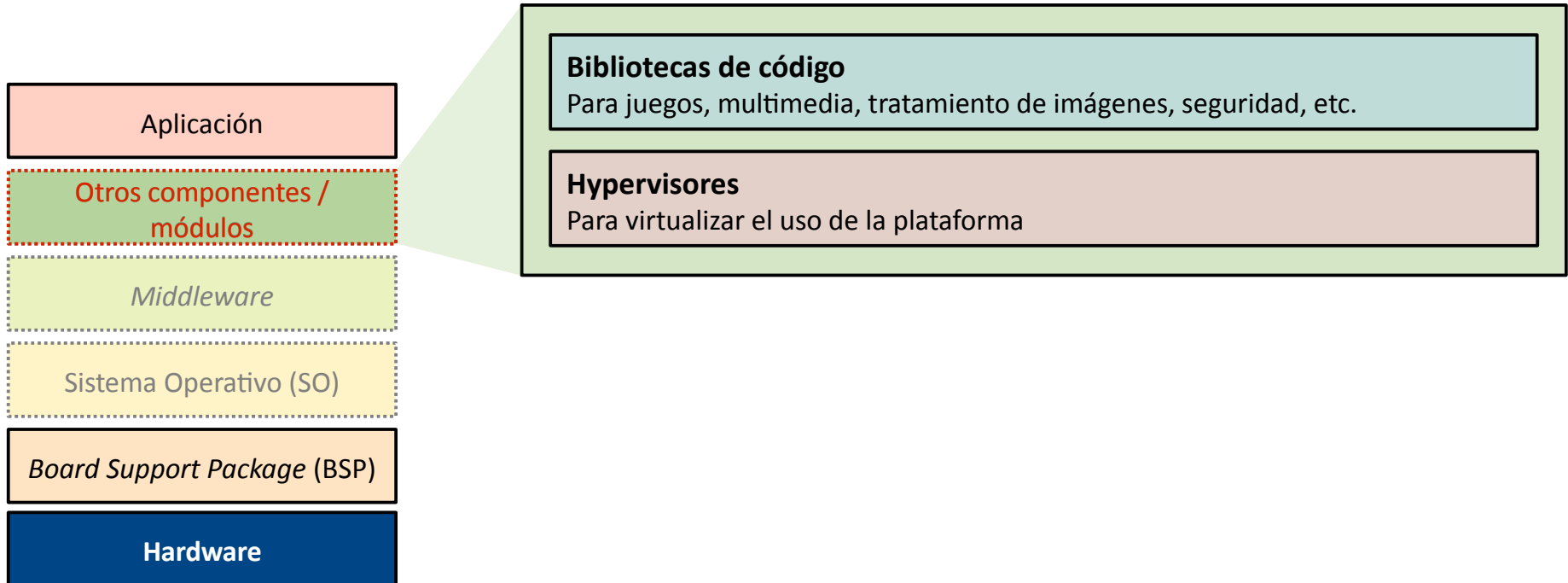
El código de gestión de las comunicaciones está mezclado con la lógica de la aplicación y es dependiente de detalles técnicos, como el protocolo de red , la dirección IP, el puerto, etc.

Comunicaciones con *middleware*



Otros componentes / módulos

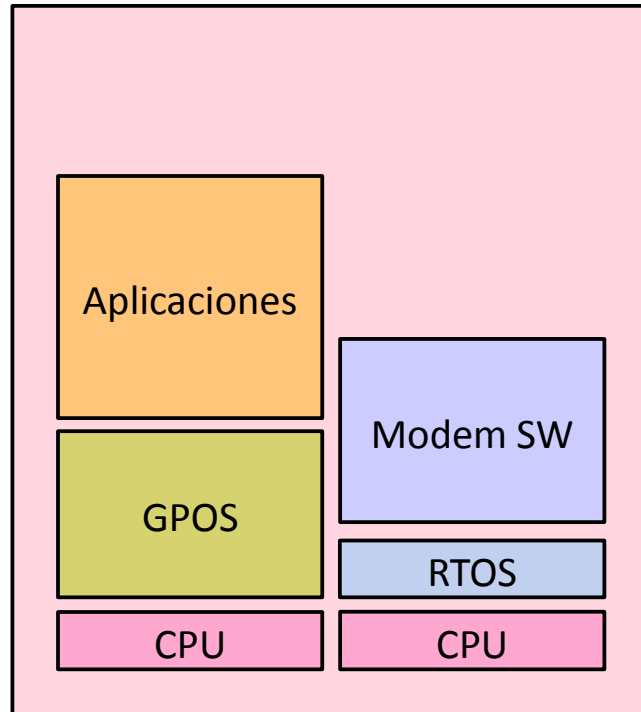
Normalmente dependientes de la aplicación



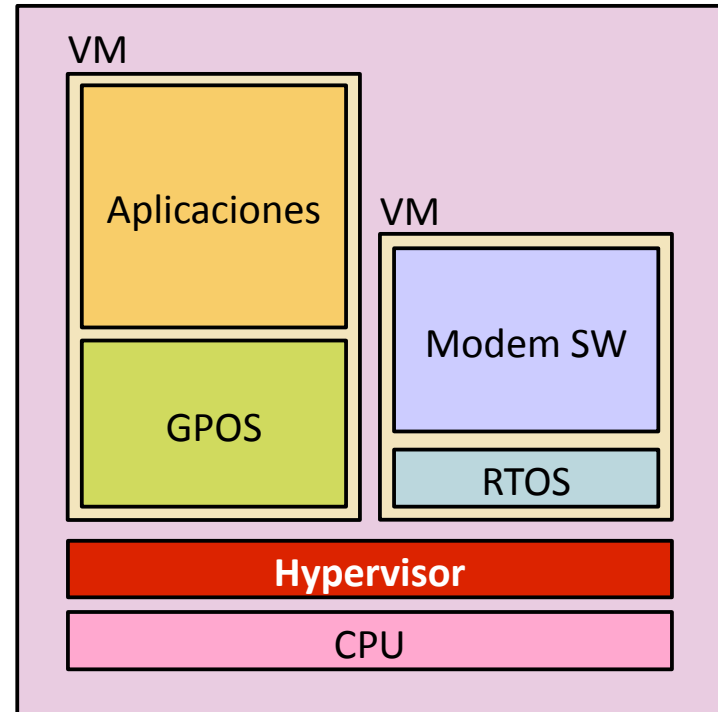
Hypervisores

Proporcionan entornos de ejecución virtuales para ejecutar varios sistemas operativos sobre la misma plataforma

Smartphone gama alta



Smartphone gama baja



Permiten obtener las prestaciones de un *smartphone* con el hardware de un teléfono convencional

Fuente:

S. Subar. Mobile Virtualization – Coming to a Smartphone Near You, junio 2010.

<http://www.visionmobile.com/blog/2010/06/mobile-virtualization-coming-to-a-smartphone-near-you/>

Lecturas recomendadas

Software empotrado:

C. Walls. *Embedded Software. The Works*. Newness, 2006. Capítulo 2

Componentes del Firmware:

Q. Li, C. Yao. *Real-Time Concepts for Embedded Systems*. CMP Books, 2003. Capítulo 3

A. J. Massa. *Embedded Software Development with eCos*. Prentice Hall, 2003. Capítulo 2

A. N. Sloss, D. Symes y C. Wright. *ARM Systems Developer's Guide. Designing and Optimizing System Software*. Morgan Kaufmann, 2004. Capítulo 10

Altera. *Nios II Software Developer Handbook*. Capítulo 5

http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf

Lecturas recomendadas

Sistemas operativos:

- Q. Li, C. Yao. *Real-Time Concepts for Embedded Systems*. CMP Books, 2003. Capítulo 4
- P. A. Laplante. *Real-Time Design and Analysis, 3ª edición*. Willey, 2004. Capítulo 3
- J. Labrose *et al.* *Embedded Software. Know it All*. Newnes, 2008. Capítulo 3
- E. Sutter. *Embedded Systems Firmware Demystified*. CMP Books, 2002. Apéndice B
- P. N. Leroux. *RTOS versus GPOS: What is best for embedded development?* Embedded Computing Design, Enero 2005. <http://embedded-computing.com/pdfs/QNX.Jan05.pdf>
- D. Addison. *Embedded Linux applications: An overview*. IBM DeveloperWorks, Agosto 2001. <http://www.ibm.com/developerworks/linux/library/l-embl/index.html>
- B. Japenga. *Why Use Linux for Real-Time Embedded Systems*.
<http://www.microtoolsinc.com/Why%20Use%20Embedded%20Linux%20for%20Real%20Time%20Embedded%20Systems%20Rev%20A.pdf>

Middleware:

- T. Noergaard. *Embedded Systems Architecture. A Comprehensive Guide for Engineers and Programmer*. Newnes, 2005. Capítulo 10