

# Sistemas Empotrados

Tema 1:

Introducción a los sistemas empotrados

Lección 2:

Sistemas empotrados



Jesús González Peñalver

# 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

# Ventajas de empotrar un procesador en un sistema

## Se facilita el diseño de familias de dispositivos

- Se ofrecen dispositivos con diferentes conjuntos de funcionalidades a diferentes precios
- Estas diferencias se deben básicamente al uso de *firmwares* diferentes y sustitución de algún periférico
- Se facilita la posibilidad de añadir nuevos dispositivos a la familia con nuevas funcionalidades impuestas por los mercados

Televisor



Diferentes modelos según subconjuntos de características

- 3D, Internet, Wi-Fi, multimedia, vídeo bajo demanda, aplicaciones (youtube, skype, etc.)
- Se diseña el modelo completo y luego se hacen diferentes modelos más baratos anulando características
- Reducción de costes y aumento de la oferta

Diferentes tamaños para un modelo

Mismo diseño y mismo *firmware*, sólo cambia el tamaño de la pantalla (y el precio)

# Ventajas de empotrar un procesador en un sistema

## Posibilidad de diseñar sistemas actualizables

Si una parte importante de las tareas que realiza el sistema está implementada en software, se puede cambiar su comportamiento, corregir fallos, etc., reescribiendo el *firmware*

Televisor CRT



Viene con una configuración fija de fábrica

Televisor LED



El fabricante libera periódicamente actualizaciones de *firmware* que corrigen fallos, mejoran funciones, añaden características, etc.

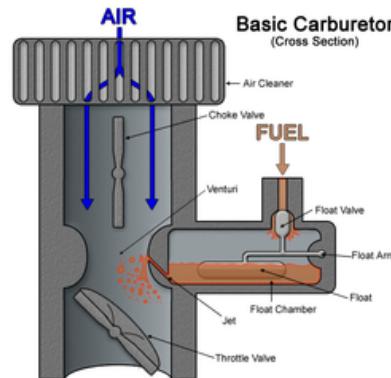
# Ventajas de empotrar un procesador en un sistema

## Posibilidad de diseñar sistemas más complejos

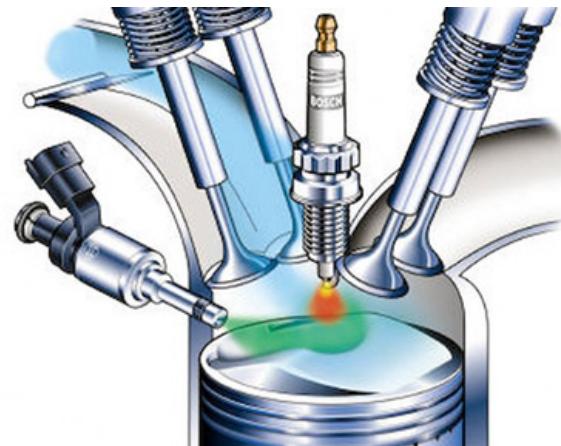
Se sustituyen funciones implementadas físicamente (mecánicamente, mediante circuitos, etc.) por software.

El software permite implementar funciones más sofisticadas y complejas con un menor coste

Carburador



Inyección electrónica (años 80)



Controla la mezcla de combustible y aire mecánicamente

Un procesador ejecuta un sofisticado programa de control que decide en cada momento la mezcla exacta.

Se mejora la combustión a la vez que se reducen las emisiones y el consumo

# 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

# Problemas para encontrar una definición adecuada

E. Sutter. *Embedded Systems Firmware Demystified*. CMP Books, 2002

Un computador oculto dentro de cualquier otro producto

F. Vahid & T. Givargis. *Embedded System Design*. Wiley, 2002

Sistemas de cómputo que están dentro de dispositivos electrónicos de mayor tamaño, llevando a cabo una determinada función repetidamente, y que a menudo no es percibida por el usuario del dispositivo

P. Marwedel. *Embedded System Design*. Springer, 2006

Sistemas de procesamiento de información que están empotrados dentro de un producto mayor y que normalmente no son directamente visibles por el usuario

W. Wolf. *Computer as Components*. Morgan Kaufmann, 2008

Cualquier dispositivo que incluye un computador programable que no es de propósito general

E. White. *Making Embedded Systems*. O'Reilly, 2011

Un sistema computerizado que se ha construido a propósito para su aplicación

[Wikipedia](#)

Sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real

# Características de los sistemas empotrados

## Dedicación a tareas específicas

Su procesador, memoria, periféricos, software, etc., están específicamente escogidos para la(s) tarea(s) que deben desempeñar

La función que debe desempeñar impone restricciones en su diseño, recursos, consumo,...

Cafetera



Televisor



Lee el código de barras de la cápsula y prepara diferentes tipos de cafés, chocolates, etc.

Se debe decodificar el *stream* de vídeo proveniente de la antena y reproducirlo en tiempo real

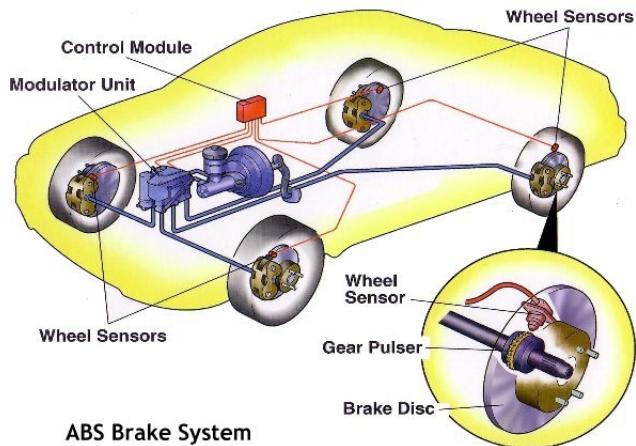
# Características de los sistemas empotrados

## Restricciones de tiempo real

En algunos sistemas, si los datos necesarios no están a tiempo, el sistema deja de funcionar correctamente

Es necesario asegurar un tiempo de ejecución determinista para las tareas. Hay que usar con cuidado elementos como las caches, predictores de saltos, etc., que introducen no determinismo

### Subsistema de frenos ABS de un coche



Si los cálculos no se hacen a tiempo y las señales al freno no se envían correctamente, se podría ocasionar un accidente

### Banda ancha móvil (4G)



Permite videoconferencias HD sin cortes, subir fotos y vídeos desde una cámara digital en tiempo real desde cualquier lugar, etc.

# Características de los sistemas empotrados

## Restricciones de confiabilidad

En algunos sistemas empotrados, los fallos no son tolerables

### Efecto de un fallo en un PC

```
*** STOP: 0x00000019 (0x00000000, 0xC00E0FFF0, 0xFFFFEFD4, 0xC0000000)
BAD_POOL_HEADER
CPUID: GenuineIntel 5.2.c irql:1f  SYSVER: 0xf00000565

Dll Base DateStamp - Name
00100000 3202cb7e - ntoskrnl.exe
00100000 31ee6c59 - actsys.sus
00200000 31ed486bf - 10_8Mx.sys
00200000 31ee6c7a - CLASS2.SYS
00200000 31ee6c6f7 - Floopy.SYS
00200000 31ee6c6f9 - Kbd.sys
00200000 31ee6d7 - KSecDD.SYS
00200000 31ee6d7 - KSecDD.SYS
00200000 31ee6c6f9 - Null.SYS
00200000 31ee6c99 - Beezy.SVS
00200000 31ee6c78 - Uideport.SVS
00200000 31ee6c722 - Uideport.SVS
00200000 31ee6c6f2 - Mga_mill.sus
00200000 31ee6c6c2 - Mga.sus
00200000 31ee6c6c2 - MDIS.SYS
00200000 31ee6c6c2 - NDIS.SYS
00200000 31f91a51 - mgs.dll
00200000 31ee6c6c - Pcap.SYS
00200000 31ee6c6c - Tcpip.sus
00200000 31ee6c6c - Vlan.SYS
00200000 31681a30 - e159x.sus
00200000 31ee6c6c - netbios.sys
00200000 31ee6c6b - networkel.sys
00200000 31ee6c6b - Sexual.SVS
00200000 31ee6c6b1 - Sexual.SVS
00200000 31f7af1ba - ruup.sus

Address dwoxid dump Build 11381
00100000 3202cb7e 00100000 ffdffff0 00070002 - Name
001471c8 88144800 88144800 fdfdf800 c0300001 - KSecDD.SVS
001471dc 88122000 88003f00 f03beee0 e133cd40 - ntoskrnl.exe
00147304 883823f0 0000023c 00000034 00000000 - ntoskrnl.exe

Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option.
```

Son comunes. Sus efectos no son graves. Se suelen arreglar instalando parches de actualización

### Efecto de un fallo en un avión



Pueden suponer una catástrofe. Es necesario invertir mucho tiempo en testar y validar el sistema, así como añadir hw/sw para evitar catástrofes en caso de fallo

# Características de los sistemas empotrados

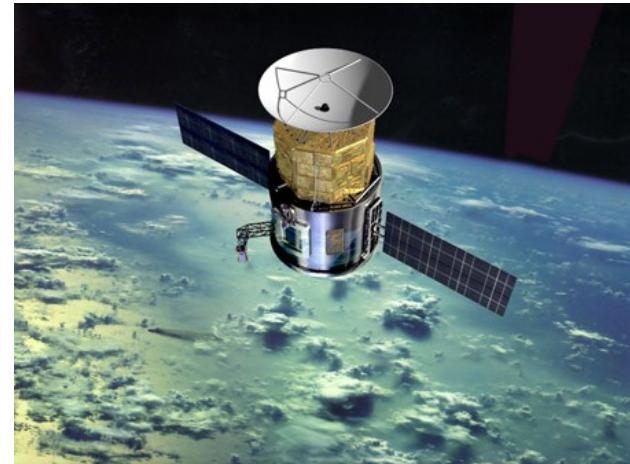
## Restricciones ambientales

Algunos sistemas empotrados deben operar en condiciones atmosféricas extremas. El HW (procesadores, memorias, periféricos, ...) debe pasar muchas pruebas para garantizar su correcto funcionamiento. Esto sube mucho su precio.

Motor de inyección



Satélite



Temperatura óptima de operación: 93°C

La temperatura oscila entre 100°C y -150°C, según le de o no el sol

Además no hay gravedad

# Características de los sistemas empotrados

## Restricciones en el consumo de energía

Menor consumo implica una fuente de alimentación más barata, y además, en los dispositivos móviles, más autonomía.

PC (Intel core i7-960, 3.2 GHz)



TDP (*Thermal Design Power*): **130 W**

Puede alcanzar los 100 °C. Intel recomienda bajar la temperatura a 60 °C

iPad 2 (ARM Cortex A9, dual core, 1GHz)



Consumo entre 0,7 y 1W, dependiendo de la carga de trabajo

Autonomía de 10 horas

No necesita ventiladores, disipadores, etc.

# Características de los sistemas empotrados

## Restricciones en el coste

Restricciones en los recursos (potencia de cálculo, memoria, almacenamiento, etc.)

Necesidad de optimización

PC



Portátil 800€

Sistemas empotrados



Decod. TDT 25€



Microondas 55€

Excepciones



Satélite 500 000 000 €



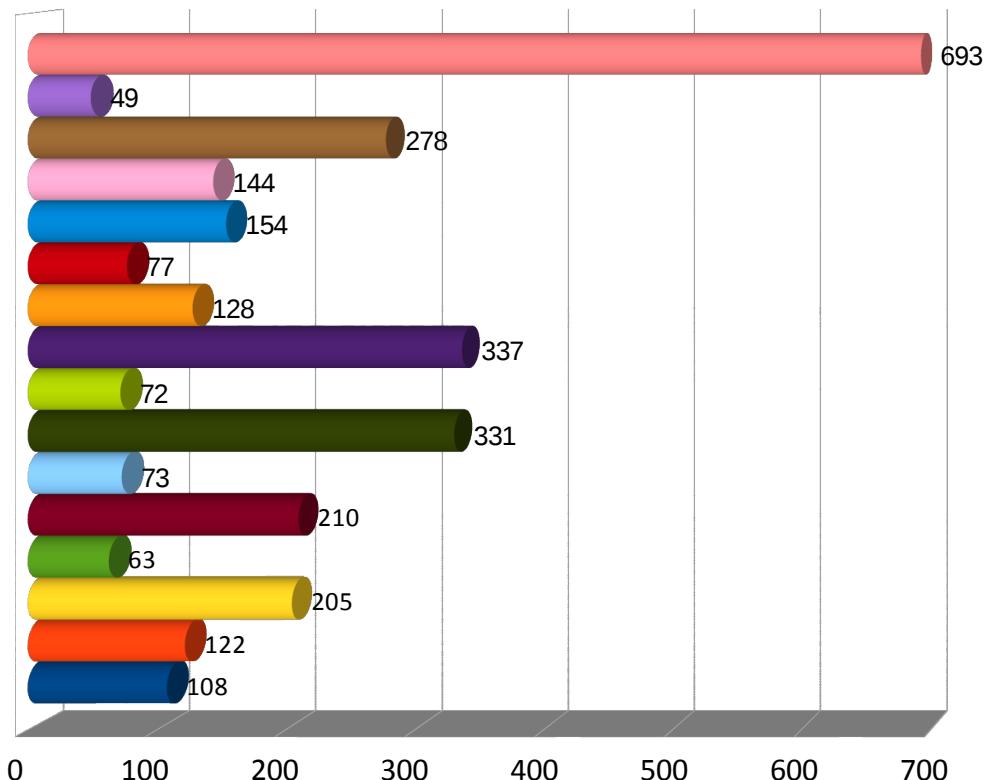
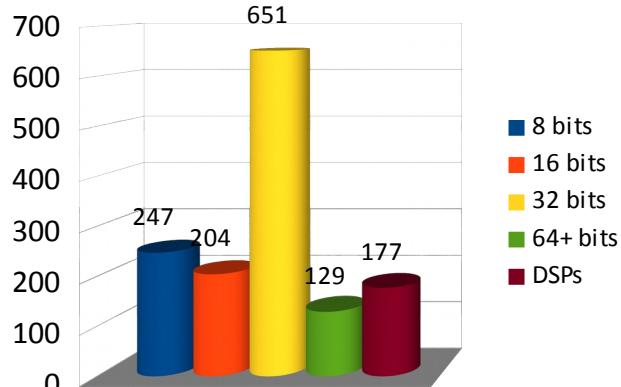
Impresora 50€



Termómetro 3€

# Características de los sistemas empotrados

## Multitud de arquitecturas y procesadores disponibles



Más de 80 empresas diseñan procesadores

Más de 1400 modelos de procesador diferente

Elegir el procesador más adecuado (**relación coste-consumo-prestaciones óptima**) para el sistema no es nada fácil

■ Audio	■ Automotive	■ Communication & Wired	■ Computers & Peripherals
■ Consumer	■ Digital Power	■ General Purpose	■ Imaging & Video
■ Industrial	■ Medical	■ Military & Aerospace	■ Mobile & Wireless
■ Motor Control	■ Security	■ Test & Measurement	■ Other

# Características de los sistemas empotrados

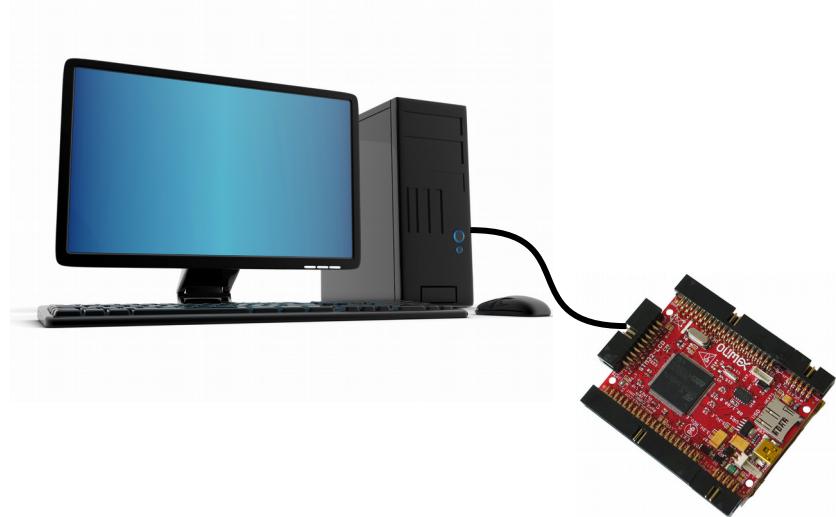
## Desarrollo cruzado

Los sistemas empotrados no suelen disponer de recursos suficientes para desarrollar en el propio sistema

En los PCs



En los sistemas empotrados



Desarrollo en el mismo sistema

La aplicación se desarrolla en la misma plataforma en la que se ejecutará

Desarrollo cruzado

Se desarrolla en un PC, pero se ejecuta en el sistema empotrado

# 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

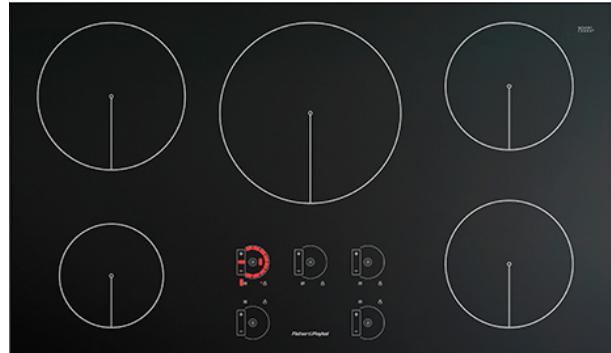
Componentes del Firmware

# Clasificación según su percepción por el usuario

## Imperceptible

Forman parte de un sistema mayor

El usuario no es consciente de su presencia



## Perceptible

El sistema empotrado es el propio dispositivo



# Clasificación según su complejidad

## Baja

Basados en un microcontrolador

Procesador: de 8-16 bits,

Memoria: ROM o Flash

Periféricos habituales:  
conversores A/D, D/A, PWM,  
UART, temporizadores, etc.

Aplicación *stand-alone*



## Media

Basados en un procesador de 32-bits

Controlador para más tipos de memorias, MPU

Periféricos habituales: pantalla táctil, usb, red, cámaras, etc.  
**(depende de la aplicación)**

Aplicación stand-alone o RTOS  
**(depende de la aplicación)**



## Alta

Procesadores distribuidos, multi-core, SoCs

Coprocesadores: MMU, FPU, GPU, codecs de audio y vídeo, comunicación de voz y banda ancha, Wi-Fi, Bluetooth, soporte de tiempo real, seguridad, etc.  
**(depende de la aplicación)**

Periféricos: pantalla táctil, GPS, acelerómetros, cámaras, etc.  
**(depende de la aplicación)**

Sistema operativo de propósito general o RTOS **(según la aplicación)**



# Clasificación según su conectividad

## Sin conectividad

### Independientes

Realizan determinadas funciones dentro de un sistema

No se comunican

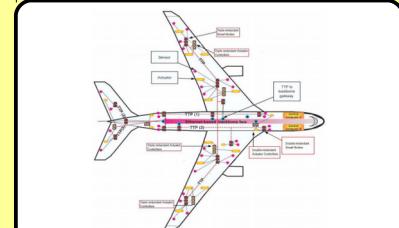
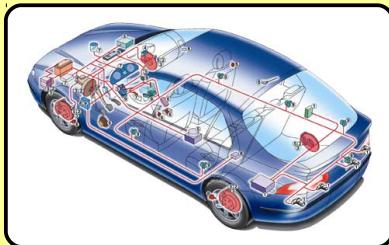


## Con conectividad

### Distribuidos

Varios subsistemas que realizan tareas diferentes dentro de un mismo dispositivo

Se comunican mediante red, normalmente cableada



### Redes de sensores

Multitud de pequeñas motas muy sencillas realizan tareas de monitorización

Normalmente no son móviles

Se comunican mediante redes inalámbricas (ZigBee, 6LoWPAN, etc.)



### Smart devices

Dispositivos móviles siempre conectados (bluetooth, 4G, Wi-Fi, etc.) y orientados comunicaciones y aplicaciones (potencia de cálculo, memoria, almacenamiento, etc.)



# Clasificación según su confiabilidad

## Confiables (*dependables*)

Fiable

Mantenimiento del servicio correcto en el tiempo

Hard  
Real  
Time

Disponible

Prontitud en el uso

Crítico

Fallos controlados y sin consecuencias catastróficas

Seguro

Evitar accesos no deseados



## No confiables

No necesitan alguna o todas las características anteriores.



Soft  
Real  
Time



# 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

# Metodología de diseño clásica

Modelos de computación:

- UML
- Máquinas de estados,
- Grafos de flujo de estados,
- Redes de Petri, o
- Lenguajes de descripción de sistemas (SystemC, SpecC, etc.)



Especificación del producto

Determina qué funciones se implementarán en HW o en SW

Partición hardware/software

Diseño detallado del hardware

Diseño detallado del software

Diseño independiente de cada parte y con diferentes herramientas (no compatibles)

La integración y la verificación son bastante complejas y costosas

**HW:**

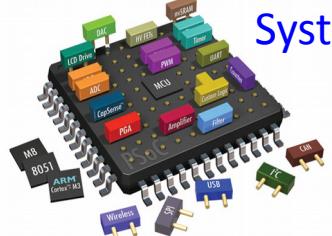
Más apropiado para tareas de cálculo intensivo o que no vayan a cambiar

**SW:**

Más apropiado para control de flujo o tareas actualizables

Integración y verificación

System-on-Chip



# Desarrollo de cada una de los componentes

## Desarrollo de hardware

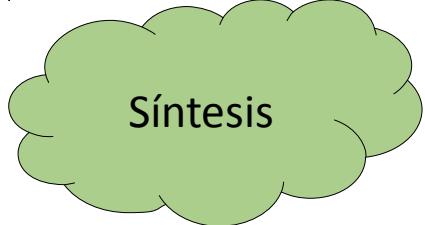


Bibliotecas  
de terceros

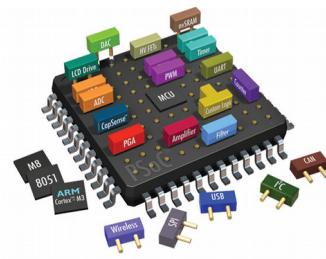


Código  
fuente

Lenguaje de  
descripción de  
hardware (HDL):  
VHDL, Verilog, etc.



Síntesis



System-on-Chip



Netlist



Implementación

## Desarrollo de software



Código  
fuente



Bibliotecas  
de terceros

Lenguaje de  
programación:  
C, C++, Java, etc.



Compilación

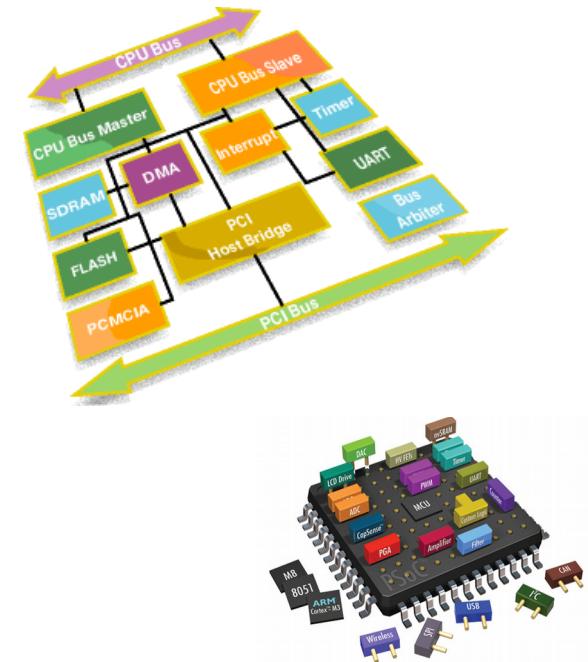


Firmware

# Componentes virtuales o IP cores

## IP (Intellectual Property) core

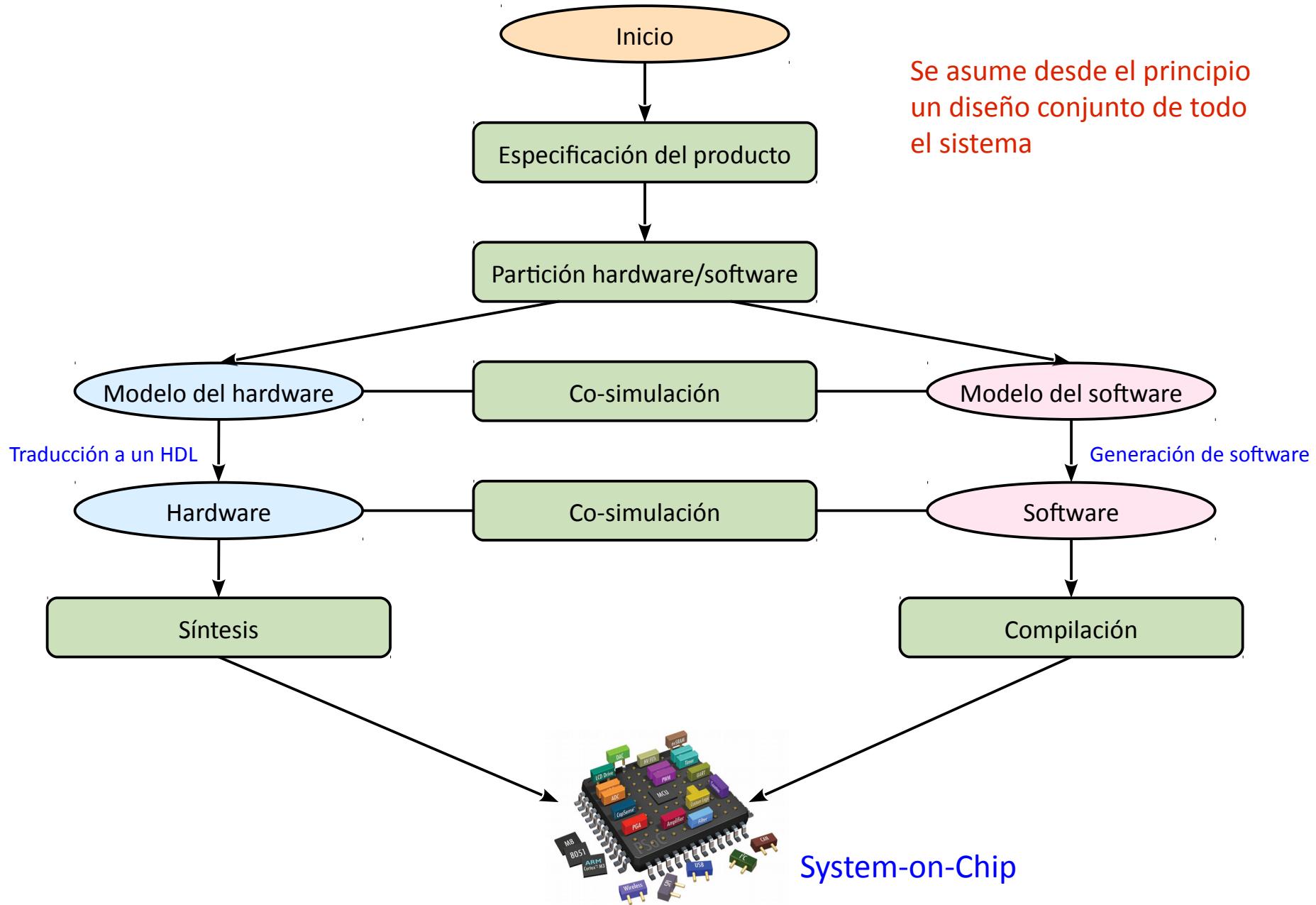
- Diseño reutilizable de un circuito
- Equivalente a las bibliotecas de código en el desarrollo de software
- Se usan como componentes para desarrollar SoCs
- Podemos diseñarlos o bien comprar licencias de *cores* diseñados por terceros
- También hay *cores open source* (*open hardware*)



## Tipos

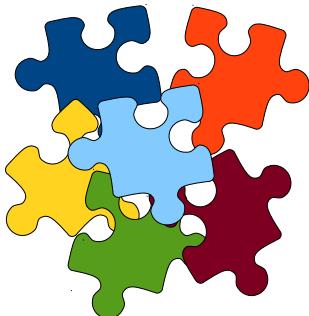
- **Soft core**
  - Distribuido en HDL (permite su modificación) o *netlist* (para evitar la ingeniería inversa)
  - Se puede sintetizar y mapear a cualquier tecnología de implementación
- **Hard core**
  - Descripto a nivel electrónico para una tecnología determinada
  - No se puede modificar
  - Prestaciones y consumo más predecibles

# Co-diseño



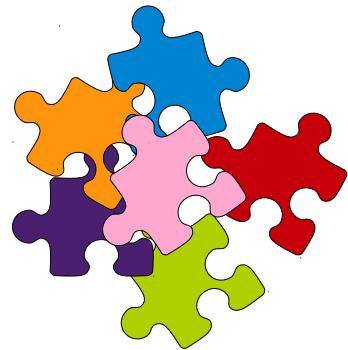
# Diseño basado en componentes

Componentes HW



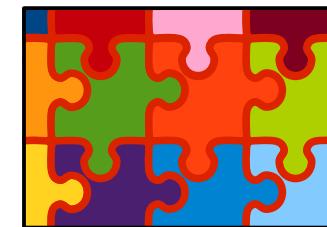
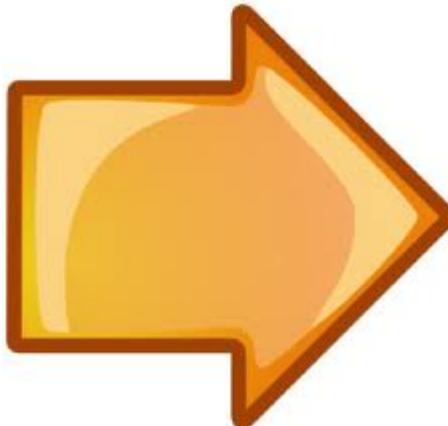
Procesadores, memorias, caches,  
controladores de interrupciones,  
aceleradores gráficos, periféricos,  
etc.

Se parte de componentes  
HW y SW previamente  
desarrollados



Componentes SW

Sistemas operativos, bibliotecas de  
código, *middleware*, drivers, etc.



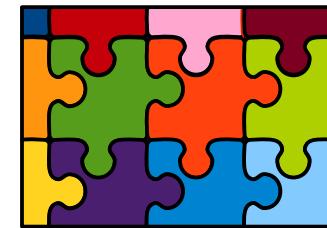
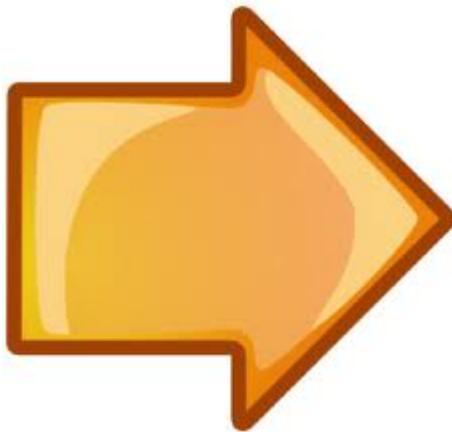
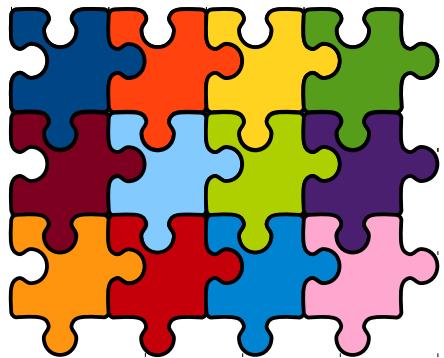
System-on-Chip

El objetivo es integrarlos  
para formar el sistema  
final

# Diseño basado en plataformas

Plataforma:

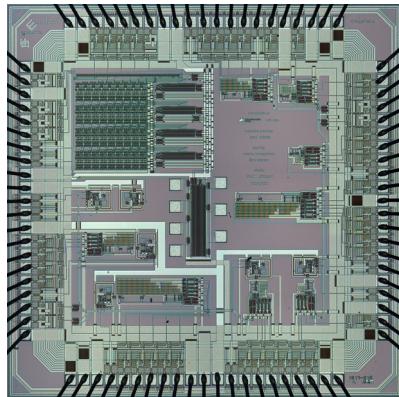
Conjunto de componentes (HW y SW)  
previamente seleccionados e integrados



System-on-Chip

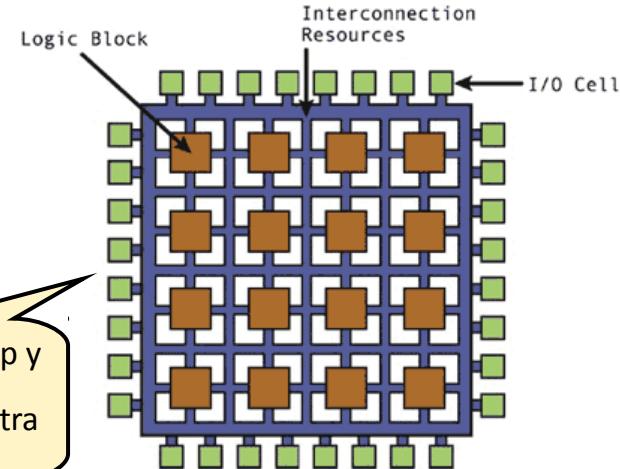
Simplemente hay que seleccionar los componentes (HW y SW) más adecuados para nuestro sistema

# Alternativas de implementación del sistema



Se fabrica un chip a la medida de la aplicación

Se compra un chip y se configura para que ejecute nuestra aplicación



## ASIC

*Application Specific Integrated Circuit*

- Diseño completamente a medida de la aplicación
- Hay que mandar el diseño a una fábrica
- Mejores prestaciones
- Menor consumo
- Menor coste por unidad
- Adecuado para un gran volumen de unidades

## FPGA

*Field-Programmable Gate Array*

- Es necesario mapear el diseño del sistema a la plataforma
- Se puede configurar en casa
- Menor coste de desarrollo y test
- Más flexibilidad (HW reconfigurable incluso en tiempo de ejecución)
- Adecuado para prototipado o pocas unidades

# 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

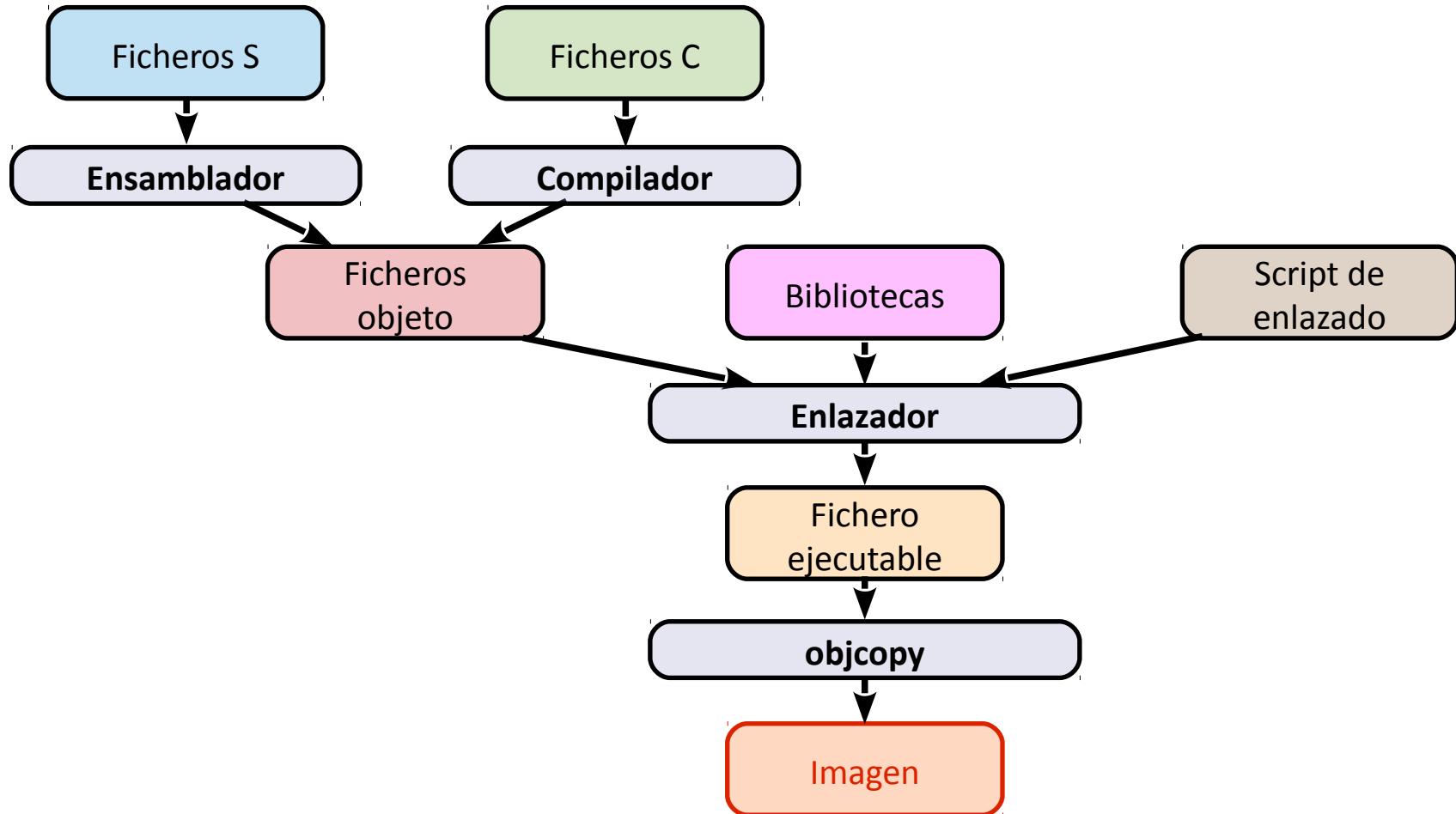
Periféricos

### La parte software

Importancia creciente del software empotrado

Componentes del Firmware

# Creación de una imagen ejecutable



Obtención de una imagen ejecutable del *firmware* a partir del código fuente

# Cadena de herramientas de desarrollo (*toolchain*)

Se denomina cadena herramientas de desarrollo porque las herramientas que la componen se usan en cadena

Ensamblador → Compilador → Enlazador → Depurador

## Alternativas

### Propietaria

- Precio alto de licencia
- Uso y configuración diferente entre distintos fabricantes
- Limitación en el número de arquitecturas soportadas



### GNU

- Genera código de gran calidad
- Soporta la mayoría de arquitecturas



# Herramientas GNU

## Componentes de la cadena de desarrollo

**binutils:** ensamblador, enlazador, etc.

**gcc:** Compilador de la GNU

**gdb:** Depurador GNU

**libC:** Muchas alternativas (newlib, uClibc, diet libc, eglIBC)

## Características

### Se construyen a partir de las fuentes

Para cualquier plataforma de desarrollo (Linux, Windows, Mac OSX, ...)

Para la mayoría de las plataformas de destino (ARM, MIPS, PowerPC, ...)

### No hay que aprender a usar herramientas nuevas

Son las mismas herramientas que se usan en los PC

### Soportan muchos lenguajes de desarrollo

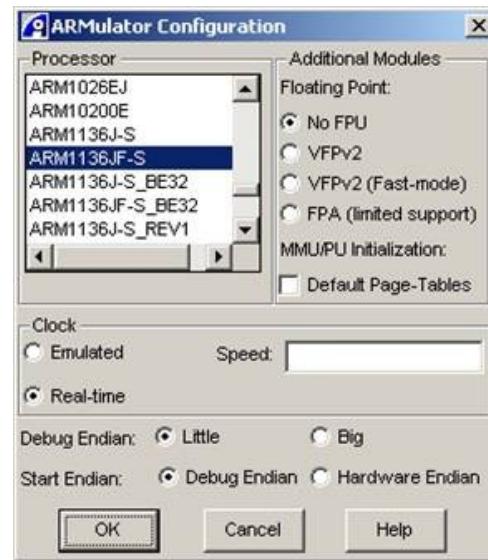
C, C++, Java, ADA, ...

### Generan código de muy buena calidad

# Simuladores

## Utilidad

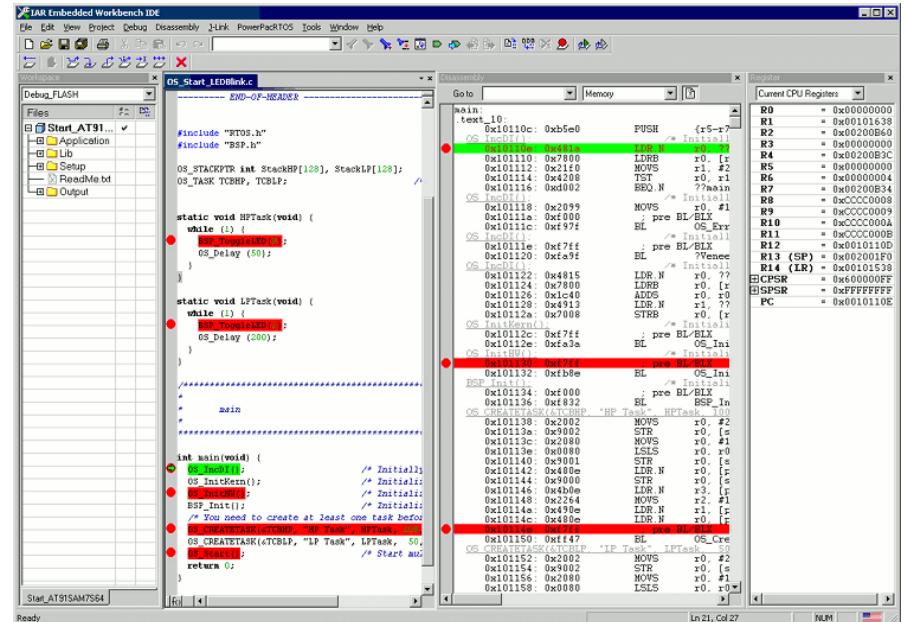
- Se ejecutan en la plataforma de desarrollo
- Simulan la ejecución de una aplicación en la plataforma de destino
- Permiten depurar fallos que afecten a la funcionalidad de la aplicación



ARMulator

## Inconvenientes

- No suelen simular la llegada de interrupciones externas al sistema
- Tampoco suelen soportar la conexión de periféricos externos al procesador
- No pueden simular los efectos del ruido en las entradas y salidas del sistema
- La ejecución de la aplicación suele ser más lenta que en la plataforma de destino



# Placas de evaluación

Permiten testar el SW del sistema mientras se está desarrollando el HW

**Hay para todos los procesadores**

Seleccionaremos una basada en el procesador que vaya a tener nuestro sistema

# Múltiples configuraciones de periféricos (según su precio)

### Memorias (RAM, ROM, FLASH, ...)

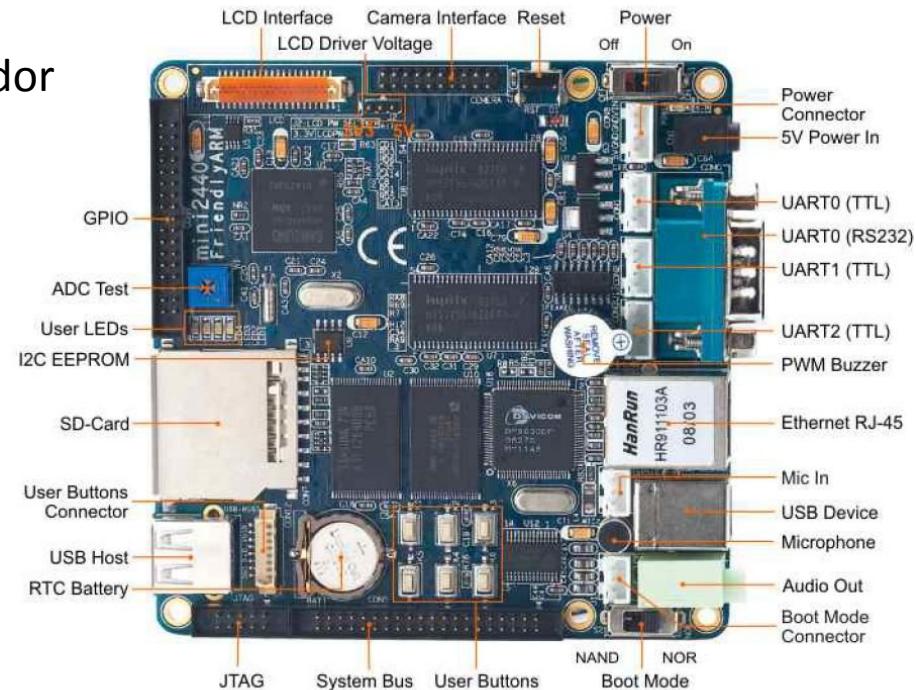
## Interfaces de depuración (JTAG)

## E/S (leds y pulsadores)

## Buses (UART, SPI, USB, IIC, IIS,...)

### Conectividad (Ethernet, Wi-Fi,...)

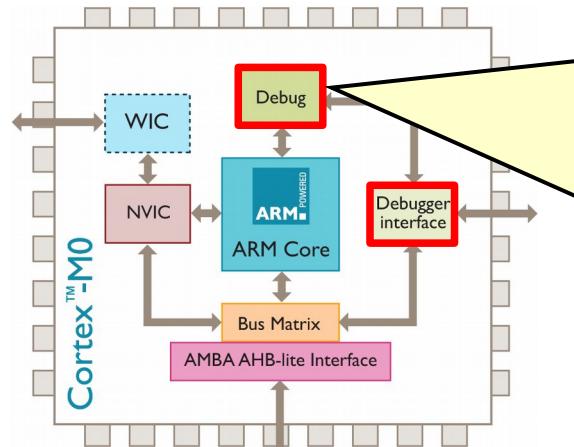
### Pantalla (LCD, táctil, multitouch,...)



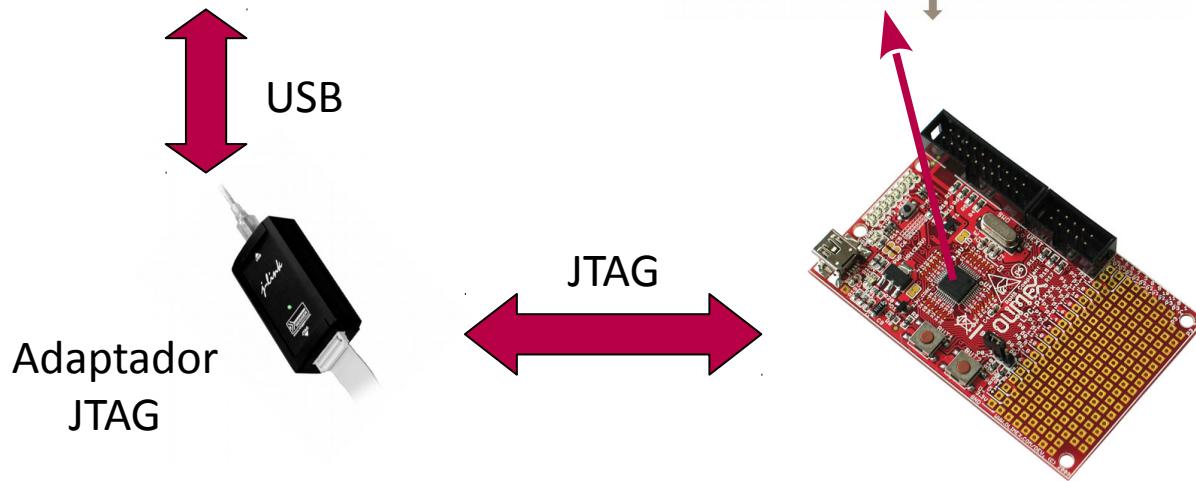
## Mini2440 (basada en un ARM920T)

Seleccionaremos una con los periféricos que necesite nuestra aplicación

# Circuitos de depuración on-chip



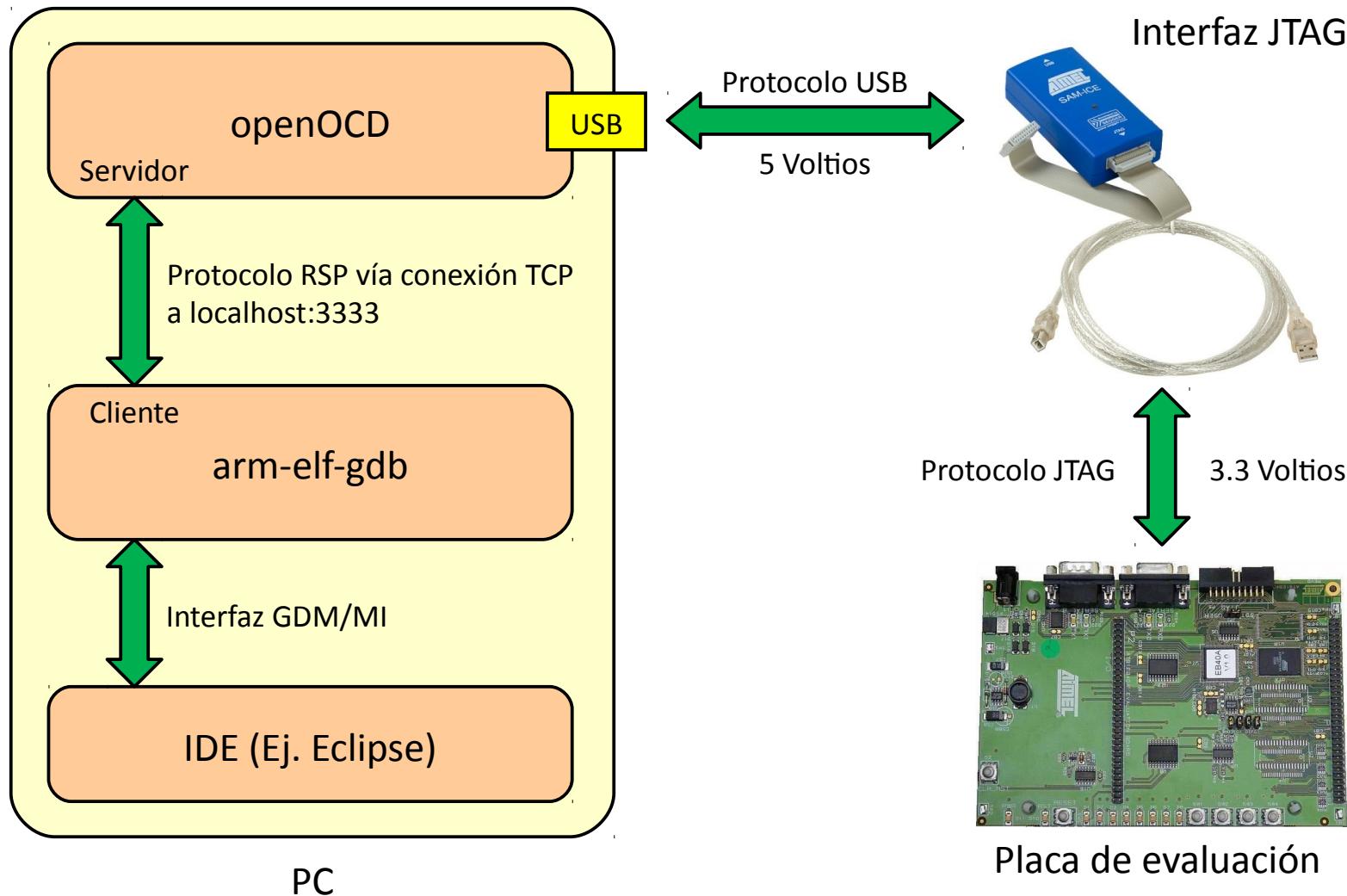
- Puntos de ruptura (RAM/ROM/Flash)
- Ejecución paso a paso
- Inspección y modificación de registros/memoria
- Detener el procesador
- Continuar la ejecución
- Etc.



## Depuración Remota

El debugger se ejecuta en un PC mientras que la aplicación corre en el sistema empotrado

# OpenOCD



Ofrece una interfaz estándar al *debugger*, independiente del tipo de adaptador JTAG de cada fabricante

# Lecturas recomendadas

Definición y caracterización de los sistemas empotrados:

A. S. Berger. *Embedded System Design. An introduction to Processes, Tools, & Techniques.*. CMP Books, 2002. Introducción

S. Heath. *Embedded System Design*. Newness, 2<sup>a</sup> edición, 2003. Capítulo 1

W. Wolf. *Computer as Components*. Morgan Kaufmann, 2<sup>a</sup> edición, 2008. Capítulo 1

Diseño de sistemas empotrados:

A. S. Berger. *Embedded System Design. An introduction to Processes, Tools, & Techniques.*. CMP Books, 2002. Capítulo 1

W. Wolf. *Computer as Components*. Morgan Kaufmann, 2<sup>a</sup> edición, 2008. Capítulos 1 y 9

P. Marwedel. *Embedded System Design*. Springer, 2006. Capítulos 1 y 5

W. Wolf. *A Decade of Software/Hardware Codesign*. IEEE Computer, 36(4): 38-43, Abril 2003

J. P. Diguet, G. Gogniat, J. L. Philippe, Y. le Moullec, S. Bilavarn, C. Gamrat, K. Ben Chehida, M. Auguin, X. Fornari y P. Kajfasz. EPICURE: *A Partitioning and Co-design Framework for Reconfigurable Computing. Microprocessors and Microsystems*, 30(6):367-387, Septiembre 2006

# Lecturas recomendadas

## GNU Toolchain:

Free Software Foundation. *Herramientas de desarrollo con licencia GNU*

**Binutils:** <http://directory.fsf.org/wiki/Binutils>

**GCC:** <http://directory.fsf.org/wiki/GCC>

**Gdb:** <http://directory.fsf.org/wiki/Gdb>

Red Hat, Inc. *The Newlib Homepage*. <http://sourceware.org/newlib/>

L. Edwards. *Embedded Systems Design on a Shoestring*. Newnes, 2003. Capítulo 3

J. P. Lynch. *Using Open Source Tools for AT91SAM7S Cross Development. Rev. C*, 2007,  
<http://gnuarm.alexthegeek.com/atmel/>

YAGARTO. *Yet Another GNU ARM Toolchain*. <http://www.yagarto.de/>

Mentor Graphics. *Sourcery CodeBench Lite Edition*. <http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/editions/lite-edition/>

## Simuladores:

Proteus. *Co-simulador de varios procesadores y SPICE*. <http://www.labcenter.co.uk/>

Qemu. *Generic and open-source machine emulator and virtualizer*. <http://www.qemu.org/>

## Circuitos de depuración en el chip:

A. Berger y M. Barr. *Introduction to On-Chip Debug*. Embedded Systems Design. Mayo 2003  
<http://www.eetimes.com/discussion/beginner-s-corner/4024528/Introduction-to-On-Chip-Debug>

Wikipedia. *Joint Test Action Group*. [http://en.wikipedia.org/wiki/Joint\\_Test\\_Action\\_Group](http://en.wikipedia.org/wiki/Joint_Test_Action_Group)

OpenOCD. *Open On-Chip Debugger*. <http://openocd.sourceforge.net/>