Robótica embebida

Sistemas embebidos

Facultad de Ingeniería
Instituto de Computación

Temas

- Sistemas Embebidos
- Microcontroladores
- System on a Chip
  - Single Board Computers
- Sistemas Operativos

Sistemas Embebidos

¿Qué es un Sistema Embebido?

- Un sistema embebido (S.E) es un sistema computador destinado a una aplicación en particular.
- Los sistemas computadores de propósito general tienen muchas aplicaciones, según el software que se instale.

Características

- Es una combinación de hardware, software y posibles elementos mecánicos.
- Específicos para una tarea por lo que son optimizados para la misma.

Firmware

- Rutinas de software almacenadas en memoria no volátil (Flash, ROM, EEPROM, etc).
  - Muchas veces específicas para ese hardware
- Software que se encuentra inmerso en el dispositivo de hardware a controlar.
- Es software muy acoplado con un hardware particular.

Características de los sistemas embebidos

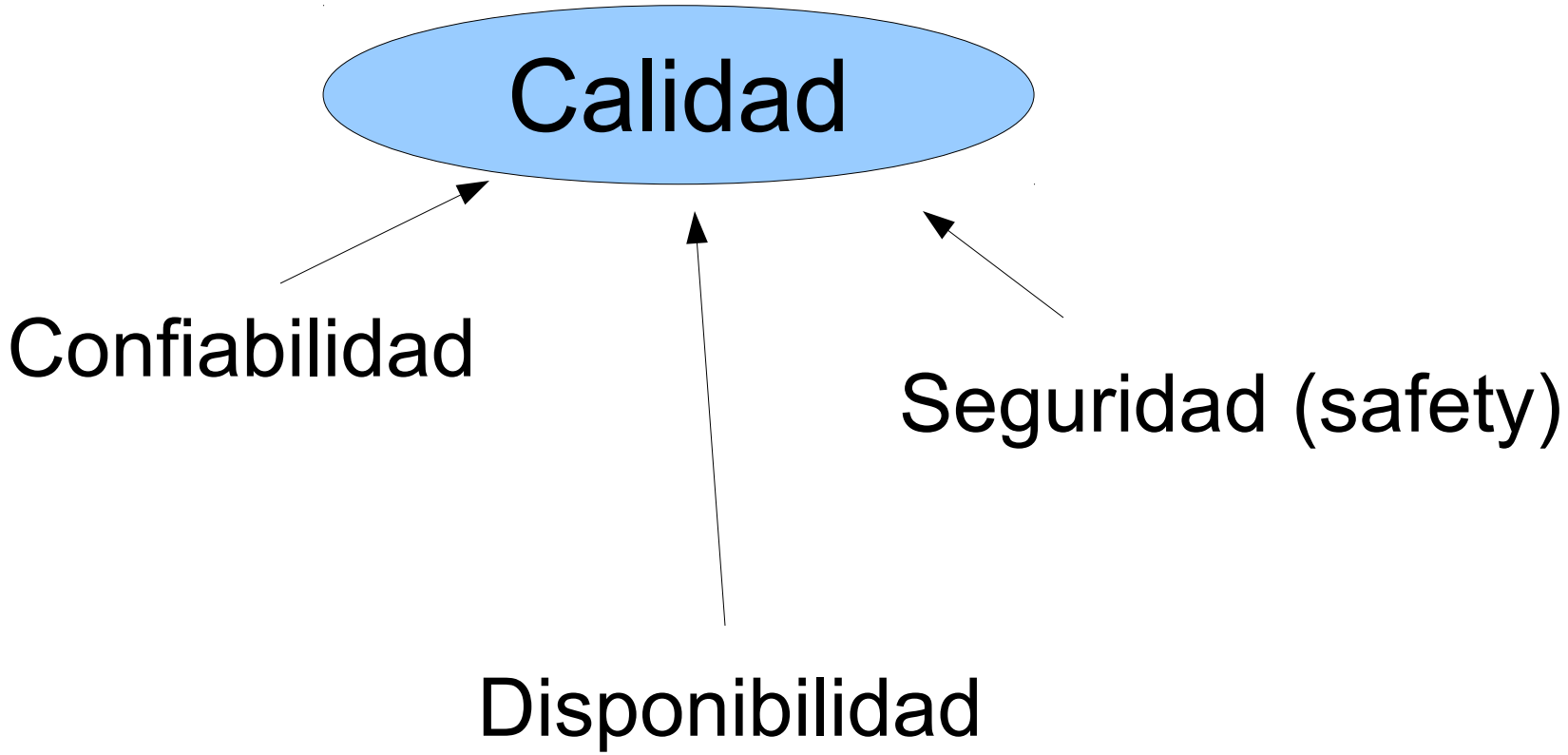
- Interactúan con el entorno
  - Directamente sensando y controlando señales.
  - Comunicándose con otros dispositivos.
- Interacción con restricciones de tiempo real.
- Bajo consumo.

Los S.E en nuestras vidas

- Electrodomésticos, periféricos para computadora, control industrial, teléfonos celulares, GPS, routers, mp3, máquinas de fotos, consolas de videojuegos, equipos para medicina, canaleras para TV, DVD, autos, entre otros.

Características de los sistemas embebidos.

Mayores exigencias



Mayores exigencias

- Confiabilidad: *C(t),* probabilidad que un sistema cumpla sus requerimientos hasta un tiempo *t* cuando opera bajo sus condiciones establecidas de funcionamiento.
- Disponibilidad: *D(t),* probabilidad que el sistema esté operando correctamente en el instante *t* de tiempo.

Mayores exigencias

- Seguridad (safety): S(t) es la probabilidad que condiciones que pueden derivar en una situación adversa, no ocurran, independientemente de si el sistema cumple o no con su misión. Especialmente nos referimos a daños a propiedad, medio ambiente e incluso, vidas humanas.

Características - Mayores exigencias

¿Qué atributo de calidad considera más importante para un sistema de control de presentismo?

Características - Mayores exigencias

¿Qué atributo de calidad considera más importante para un sistema que controla el acceso a un centro de cómputos?

Características - Mayores exigencias

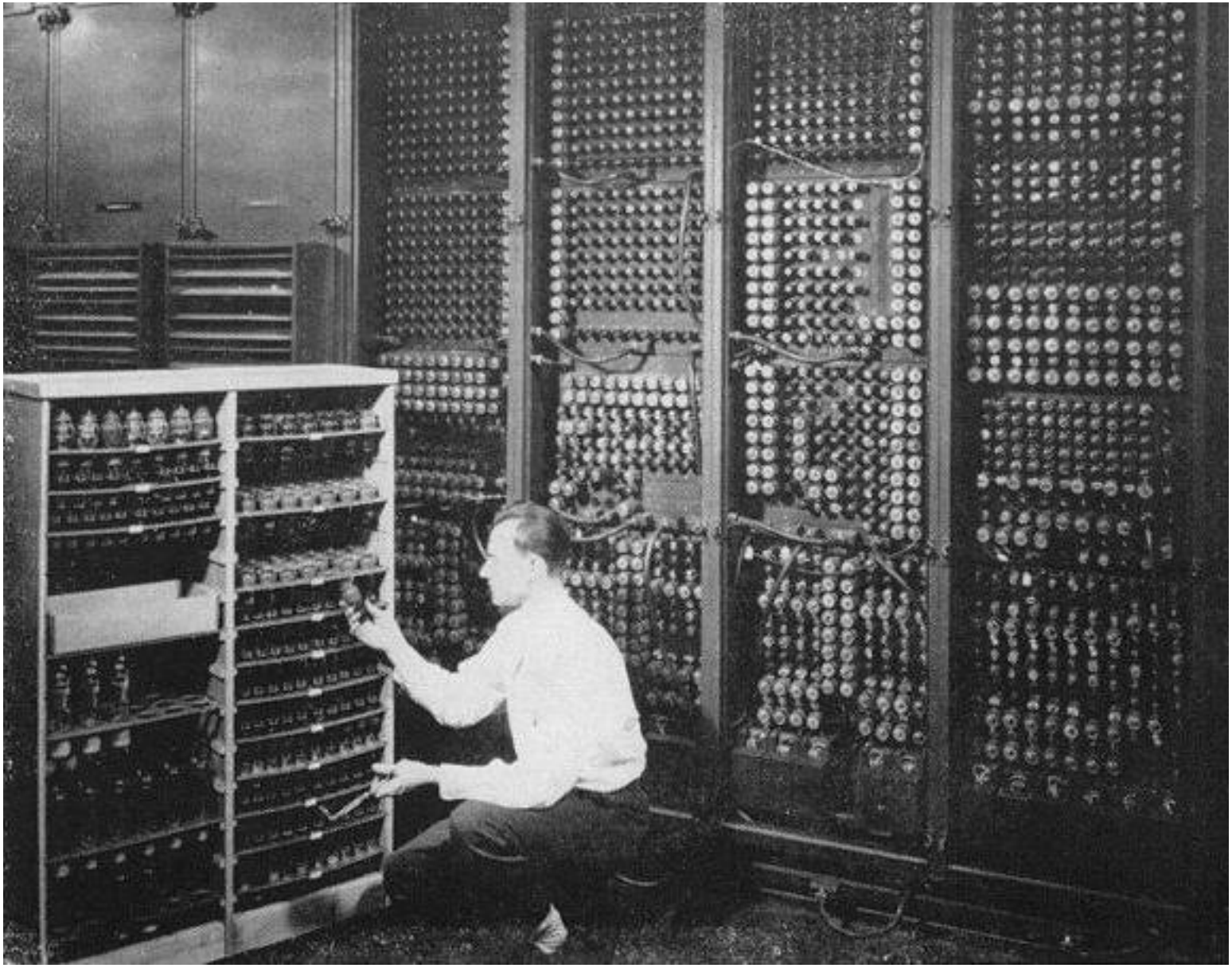
¿Qué atributo de calidad considera más importante para una guillotina de papel?

Hay que probar más

- Aeroespacio: Mariners 1, 3, 8, Ariane 5 (1996).
- Sobredosis radiológicas en Panamá.
- Durante el 2006, el 60% de los automóviles de una marca de lujo alemana volvieron a taller por fallas en el software.

El hardware también falla

- Aún bajo implementaciones “perfectas” de software/firmware los sistemas embebidos se encuentran inmersos en un ambiente externo, el cual puede afectar el correcto funcionamiento
- Este ambiente produce defectos transitorios en el dispositivo debido principalmente a:
  - Interferencia electromagnética (EMI)
  - Rayos cósmicos
  - Temperatura alta
- Su naturaleza transitoria hacen muy difícil de detectar/reproducir/corregir



mina
network management | artificial intelligence

Protección ante fallas

- Watch Dog Timer
  - Es utilizado para prevenir caídas del software.
  - En entornos con ruido eléctrico, puede ocurrir que el Program Counter del CPU se vea afectado y éste comience a ejecutar en un lugar indeterminado.
  - El circuito WDT se encarga de resetear el CPU si el registro WD se desborda.
- Línea de reset externa

Entrada/Salida (E/S)

- Debido a su característica los sistemas embebidos deben interactuar con el ambiente que los rodea.
- Sensando señales del ambiente o actuando sobre el mismo.
- Hay dos maneras de manejar la E/S:
  - Digital
  - Analógica

E/S Digital

- Se intercambian “unos” y “ceros”.
- Esos valores corresponden a voltajes de referencia.
- Utilizado para implementar protocolos de comunicación.
- Útil para controlar algunos dispositivos electrónicos:
  - Prender un led.
  - Leer estado de un botón.

E/S Analógica(1/2)

- Los valores que se intercambian pueden tomar varios valores.
- Es necesario disponer de conversores Digital -> Analógico (D/A) o Analógico -> Digital (A/D).
- Un conversor A/D convierte un voltaje en un pin de entrada a un valor digital.
- La resolución del conversor condiciona la cantidad de valores a representar. Un canal de 10 bits va a permitir representar valores desde 0 a 1023.

E/S Analógica(2/2)

- Un conversor D/A convierte un valor digital a un voltaje en un pin de salida.
- Muchos sensores se manejan de esta manera:
  - Temperatura, humedad, luz, micrófono
- Algunos actuadores se manejan de esta manera:
  - Parlantes, motores

Lectura/escritura de E/S(1/2)

- Paradigmas para implementar la lectura/escritura de E/S:
  - Polling: Donde se utilizan ciclos de CPU para estar constantemente consultando el valor de alguna entrada.
  - Interrupciones: El CPU es notificado externamente. A nivel de software se ejecuta una rutina de atención a la interrupción.

A tener en cuenta...

- Latencia de interrupción: Tiempo entre la ocurrencia de la interrupción y ejecución de la rutina de interrupción.
- En sistemas de tiempo real se trata de acotar a valores mínimos la latencia de interrupción.

Lectura/escritura de E/S(2/2)

También puede existir un enfoque híbrido donde se utilice una interrupción de timer para consultar el estado de un periférico que no genera interrupciones.

Comunicación

- Serial
  - RS-232
  - SPI
  - I^2C
- Paralela
- Hoy día se está comenzando a utilizar otros mecanismos
  - USB
  - Wifi
  - Ethernet
- Síncrona o asíncrona.
- Redes de dispositivos.

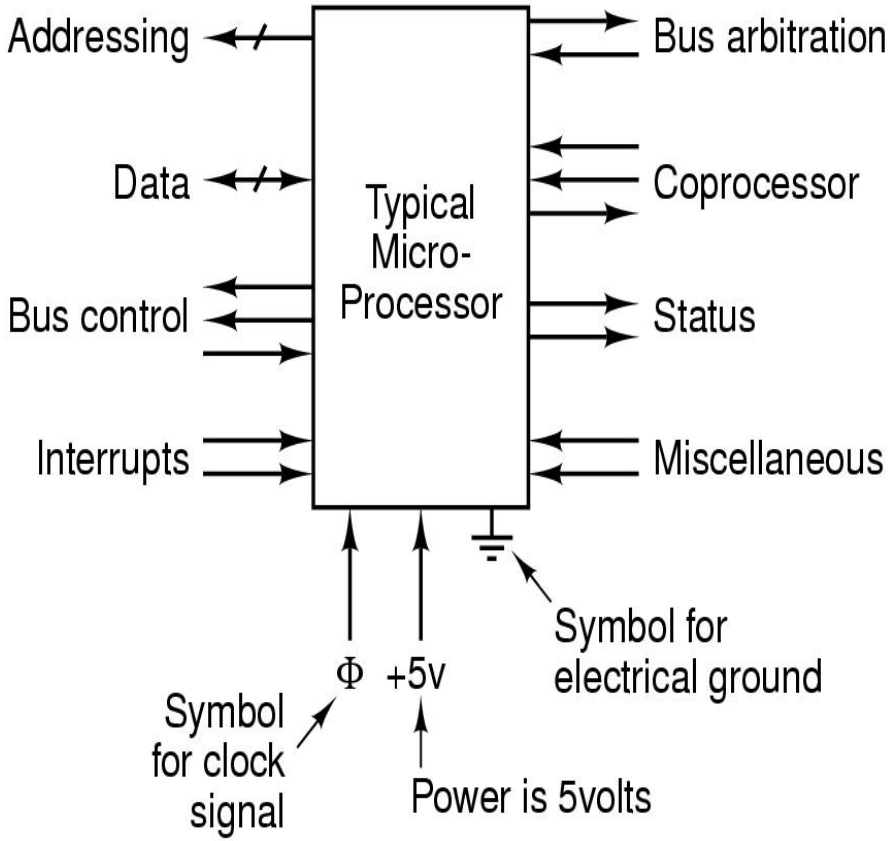
Formas de implementar S.E.

- Basados en microcontroladores
- Basados en Systems on a chip (SOC)
- Híbrido

Microcontroladores

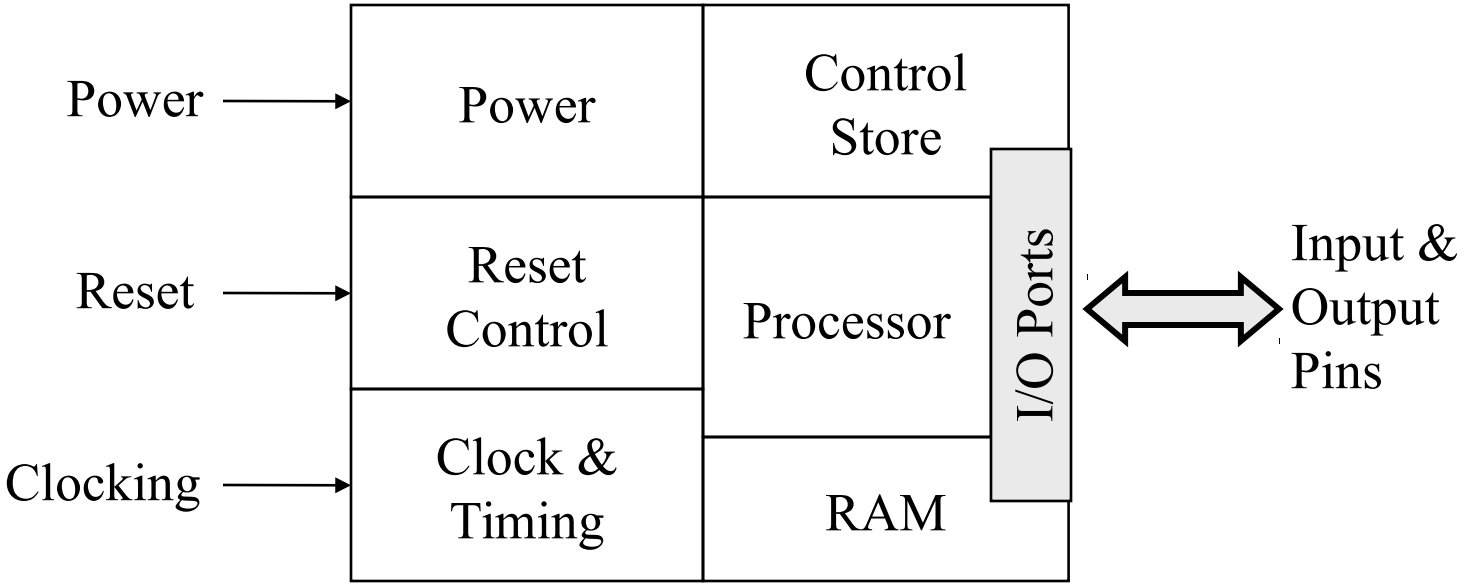
Introducción a los microcontroladores (1/3)

- Un microprocesador (µP) es una CPU en un solo circuito integrado.
- Un computador es una CPU, más memoria y puertos de E/S.
- Un sistema computador es un computador más periféricos.



Introducción a los microcontroladores (2/3)

Un microcontrolador (µC) es un sistema autocontenido donde el microprocesador, soporte, memoria y entrada/salida se presentan dentro de un mismo integrado.



Introducción a los microcontroladores (3/3)

Características

- Fáciles de utilizar.
- Bajo costo.
- Flexibles.
- Debido a su tamaño puede incluirse dentro del dispositivo que gobierna.

Tipos de Microcontroladores

Los microcontroladores se pueden clasificar en:

- Microcontroladores de 8 bits.
- Microcontroladores de 16-32 bits
- Procesadores de señales digitales (DSP)

Microcontroladores de 8 bits (1/3)

- Todos los recursos necesarios están incluidos en el chip.
- Solo necesitan alimentación y reloj.
- Proporcionan control e interfaz con dispositivos externos de manera económica y programable.

Microcontroladores de 8 bits (2/3)

Disponen de:

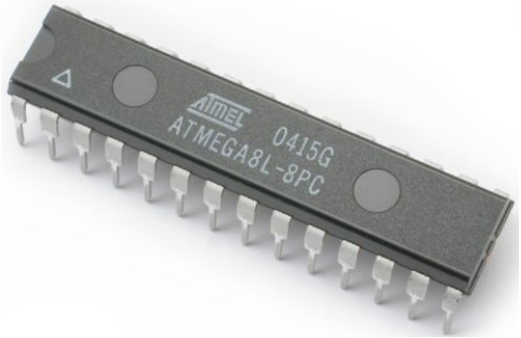
- Reset
- Reloj
- Procesador
- Memoria ROM para el programa e interfaz de programación
- Memoria RAM para variables.
- I/O Pins.

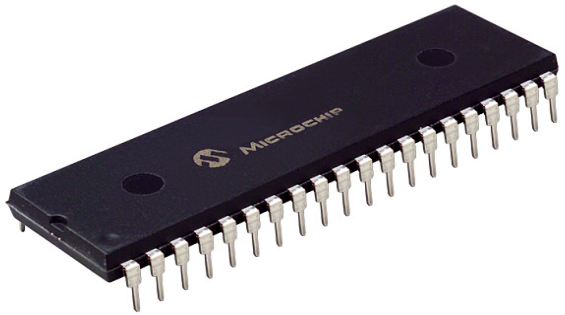
Adicionalmente pueden incluir:

- Capacidad de debugging
- Interrupciones
- I/O analógica
- Comunicación serial y/o paralela
- Interfaz con memoria

Microcontroladores de 8 bits (3/3)

- Muy poca RAM (decenas de KB).
- Muy poca velocidad de reloj (decenas de MHZ).
- Muy utilizados en aplicaciones de control.
- Bajo costo.
- Bajo consumo energético (algunos μW).
- Ejemplo: Control remoto Universal (programable)





Procesador de señales digital

- Es una categoría relativamente nueva de microprocesadores.
- El objetivo de los DSP es tomar un señal analógica y calcular una respuesta apropiada.
- Ejecutan a gran velocidad para permitir el control en tiempo real.
- Ejemplo: reproductor MP3

Registros de E/S

- Son los componentes más utilizados de los microcontroladores .
- Los microcontroladores disponen de registros para controlar los dispositivos de E/S.
- Espacios
  - E/S mapeada en memoria.
  - Mapa E/S y mapa de memoria.

Reloj del sistema

- Los µC están diseñados para ejecutar con poco soporte externo para el reloj del sistema.
- Los µCs corren en el entorno de las decenas de MHz.
- Existen diversos métodos para proporcionarle el reloj al µC:
  - Usando un cristal
  - Resonador cerámico
  - Oscilador RC
  - Utilizando un clock interno en el µC.

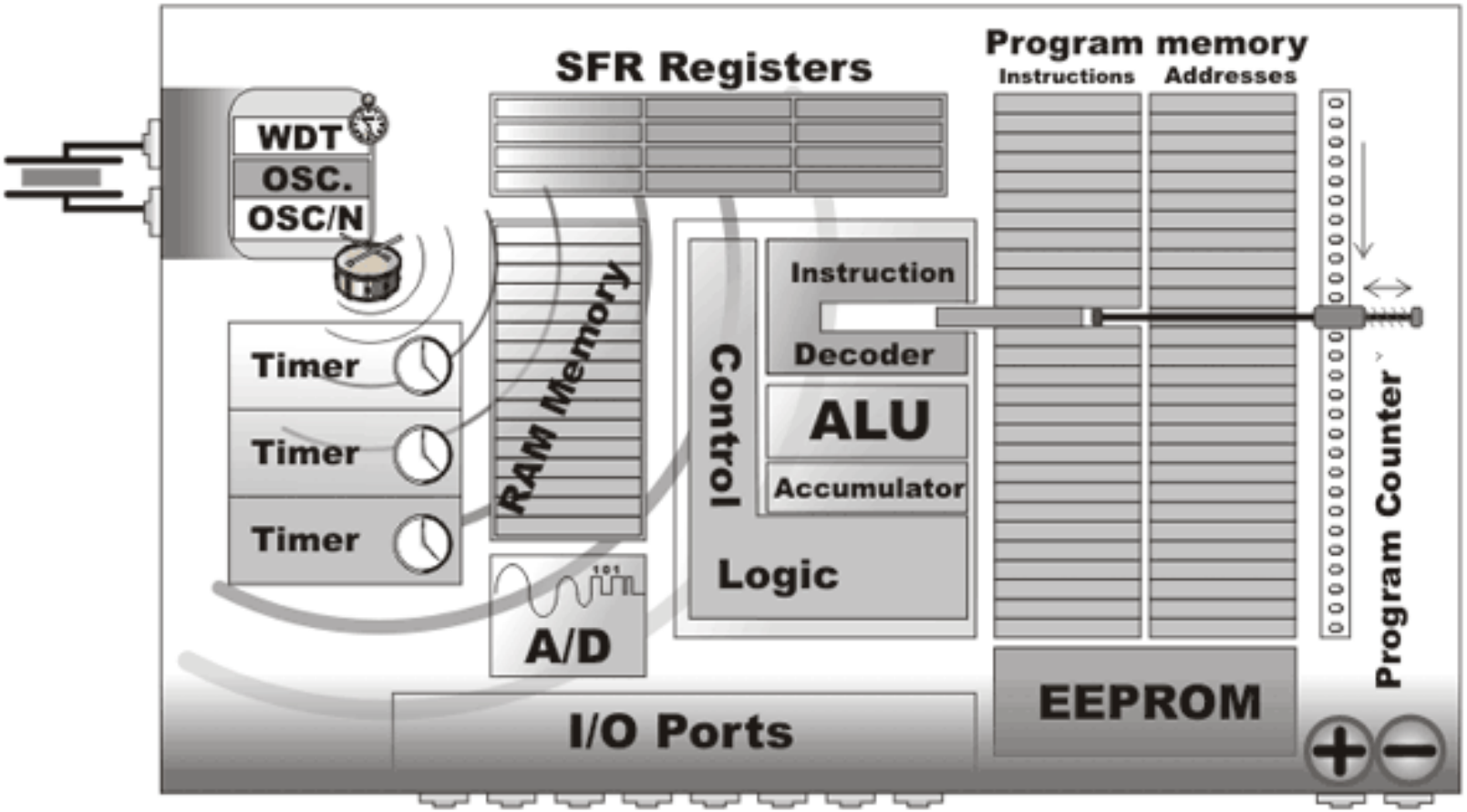
Alimentación

- Minimizar el consumo de corriente es un tema a tener en cuenta.
- Hay que tener en cuenta el consumo:
  - Del microcontrolador en modo normal.
    - Depende de la frecuencia y del voltaje proporcionado.
  - Del microcontrolador en modo sleep.
  - Del los dispositivos conectados a la E/S.
    - Particular de cada aplicación.
    - Pueden gestionarse adecuadamente los dispositivos.

Timers

- Se utiliza para trabajar con eventos de tiempo.
- Contadores: cuentan acontecimientos que suceden en el exterior.
- Temporizadores: controlan períodos de tiempo.

Arquitectura

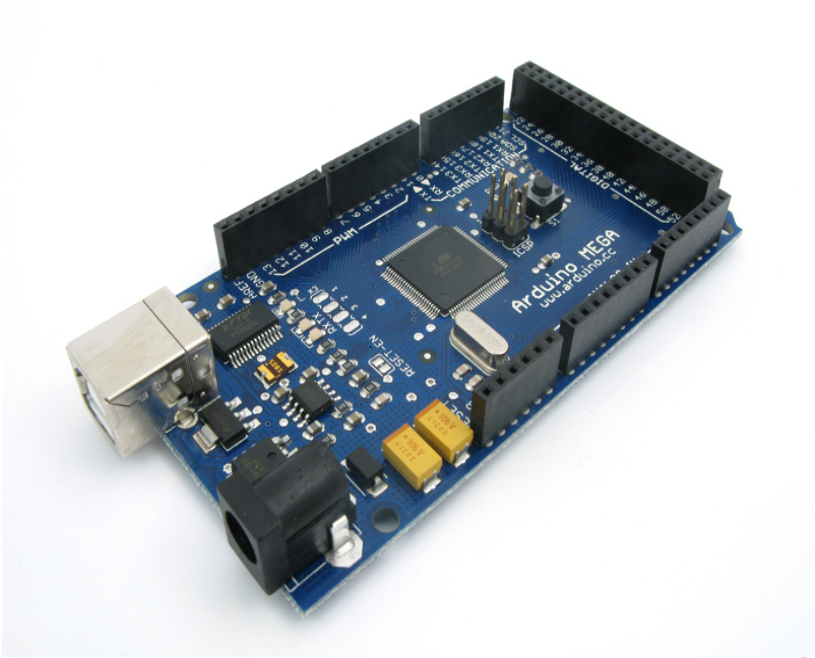


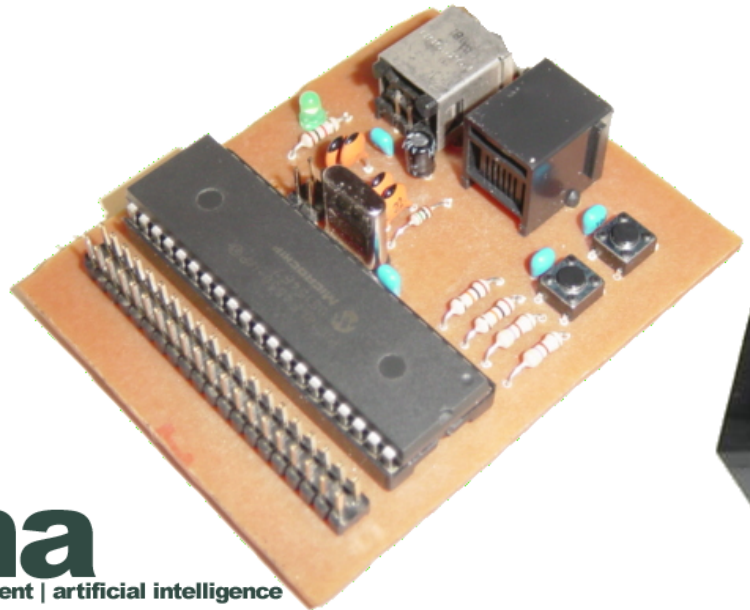
IO Boards(1/2)

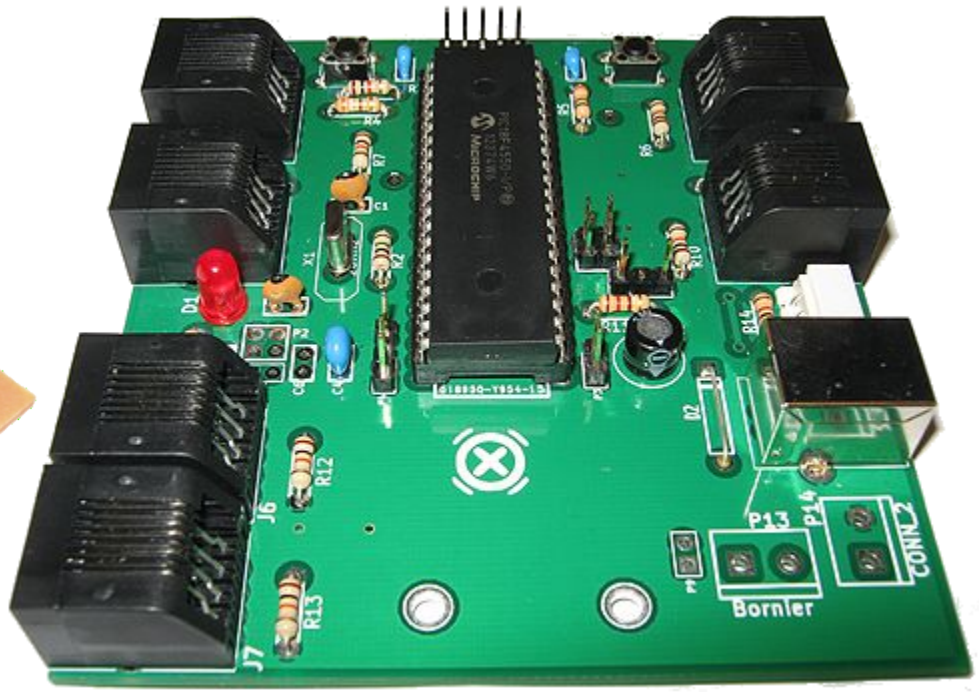
- Existen placas orientadas a trabajar con entrada/salida que permiten realizar diseños embebidos en base a ellas.
- Utilizadas para prototipar o diseñar productos de pocas unidades.
- Incluyen un microcontrolador, señal de reloj, conectores de E/S, memoria flash externa, acondicionamiento de señales.

IO Boards(2/2)

- Arduino
- GogoBoard
- USB4all
- USB4Butiá





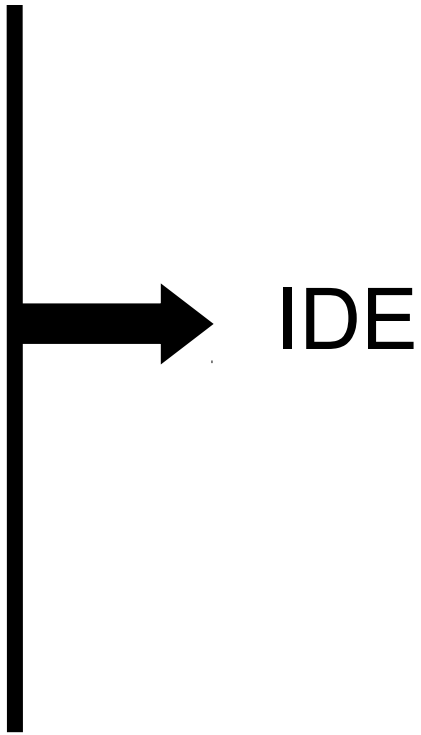


Desarrollo de software

- Herramientas y entornos de desarrollo
- Programación
- Debug

Herramientas y entornos de desarrollo (1/4)

- Editor
- Compilador
- Ensamblador
- Simulador
- Emulador
- Programador



Herramientas y entornos de desarrollo (2/4)

- Assembler
  - Instrucciones assembler.
  - Directivas
  - Debe estar bien comentado

Herramientas y entornos de desarrollo (3/4)

- Lenguaje de alto nivel
  - C, C++, Basic, Forth, JavaMe.
  - Proporcionan
    - Mayor nivel de abstracción
    - Bibliotecas.
    - Tipos de datos.
    - Variables locales y globales.
    - Estructuras de datos y punteros.
    - Asignación de memoria para datos.
    - Acceso a registros.
  - Decrementa el tiempo de desarrollo.

Herramientas y entornos de desarrollo (4/4)

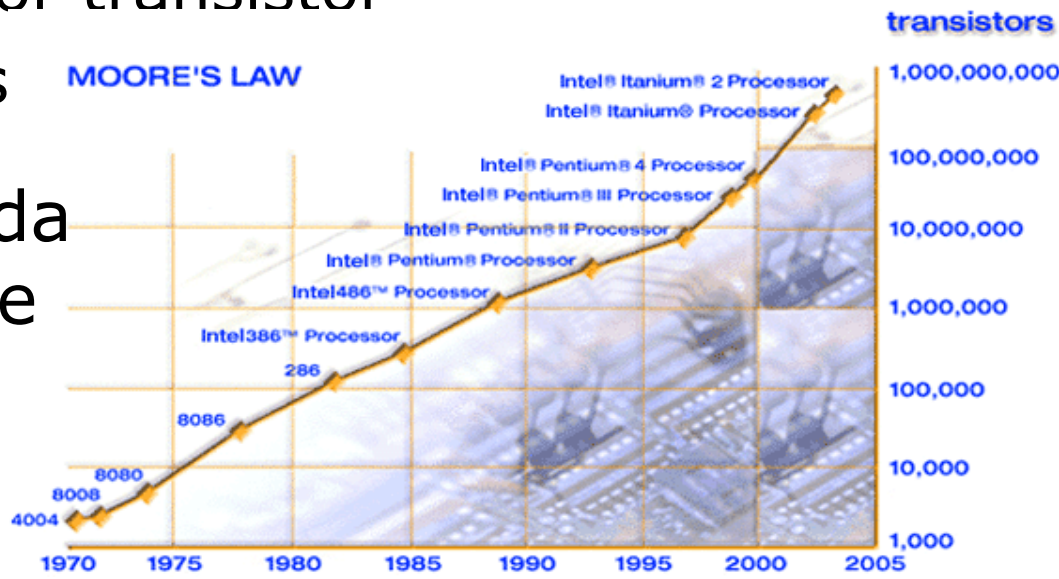
- Real Time Operating System (RTOS)
  - Multitasking
  - Scheduling
  - Context Switching
  - Respuestas en tiempo a eventos del mundo.
  - Comunicación entre procesos.
  - Stack TCP/IP
  - Ejemplos: Salvo, FreeRTOS, µC/OS, ...

Fabricantes

- Intel
  - 8048
  - 8051 (Intel y Otros)
  - 80186, 80186 y 80386 EX.
- Microchip
  - PIC
- Motorola
  - 68HC11 (Motorola y Toshiba)
  - 683xx
- Atmel
  - AVR

Evolución del hardware embebido:

- Cada vez hay más aplicaciones ya que pueden hacerse dispositivos que:
  - Consuman menos por transistor
  - Cuesten menos por transistor
  - Sean más rápidos
- Ley de Moore: Cada 2 años, el doble de transistores en la misma superficie

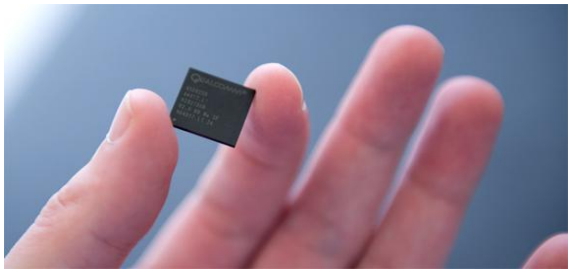


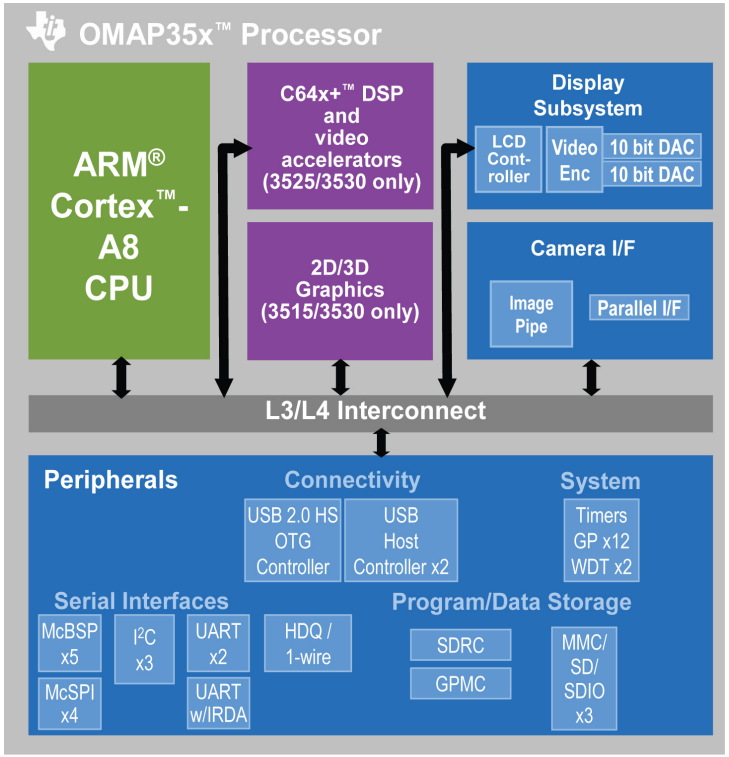
Systems on a chip (SOC)

- Se refiere a integrar todos los componentes de un computador u otro dispositivo electrónico en un chip.
- La principal diferencia con un µC es la memoria disponible y el soporte para MMU.
- En general, los SOC permiten ejecutar sistemas operativos (S.O) tradicionales.
- Cada SOC esta orientado a determinado tipo de aplicaciones.

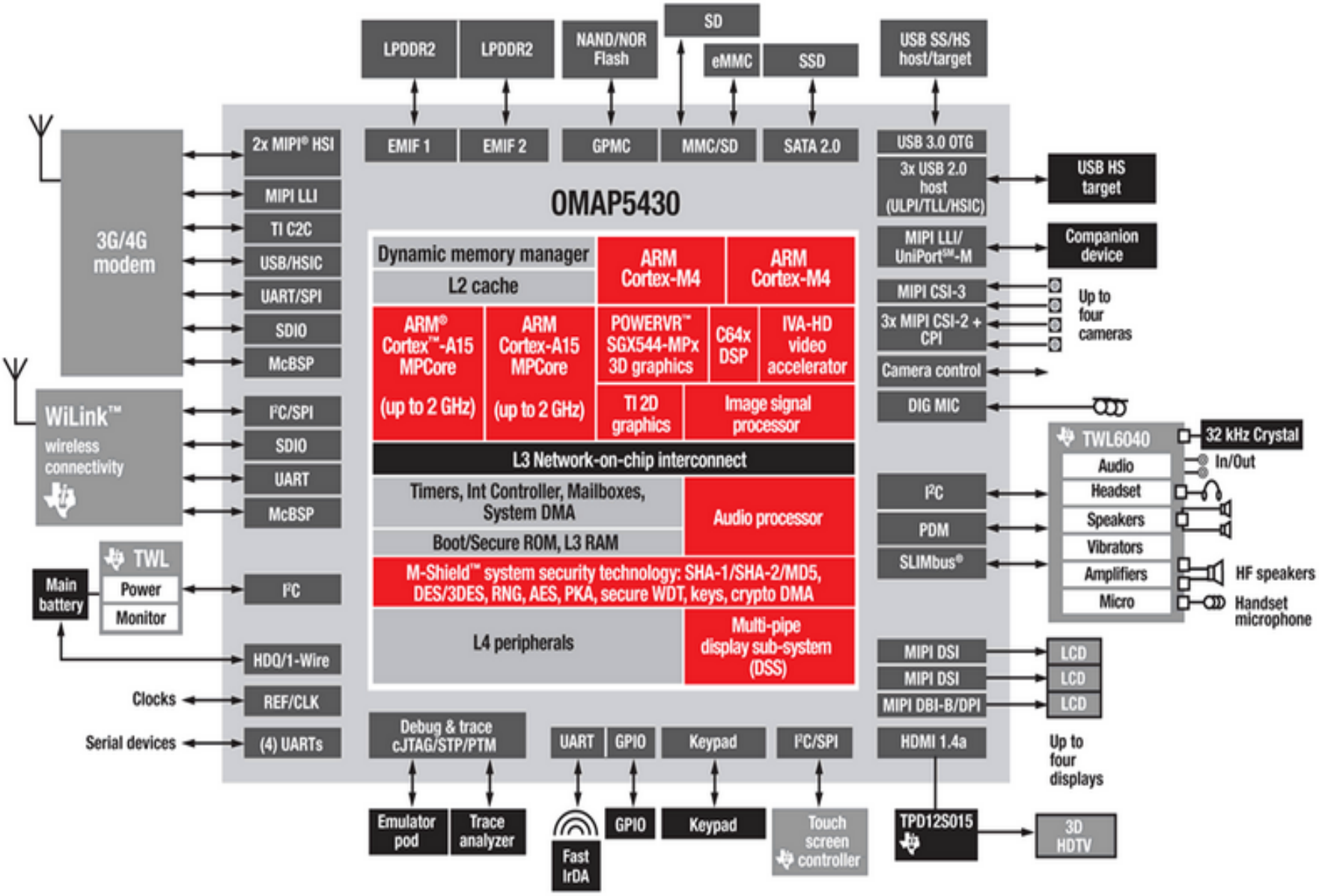
Systems on a chip (SOC)

- Al incrementar el nivel de integración se disminuye el costo de fabricación.
- Permite implementar sistemas más pequeños de altas prestaciones.





TI OMAP5430 SoC



Hardware Embebido basado en SOC(1/3)

- La arquitectura de hardware para los sistemas embebidos por lo general difiere a la de los sistemas de escritorio.
  - ARM, MIPS o PowerPC son muy utilizados por su bajo consumo y buena performance.
  - x86 actualmente no es tan frecuente, pero también es utilizado.

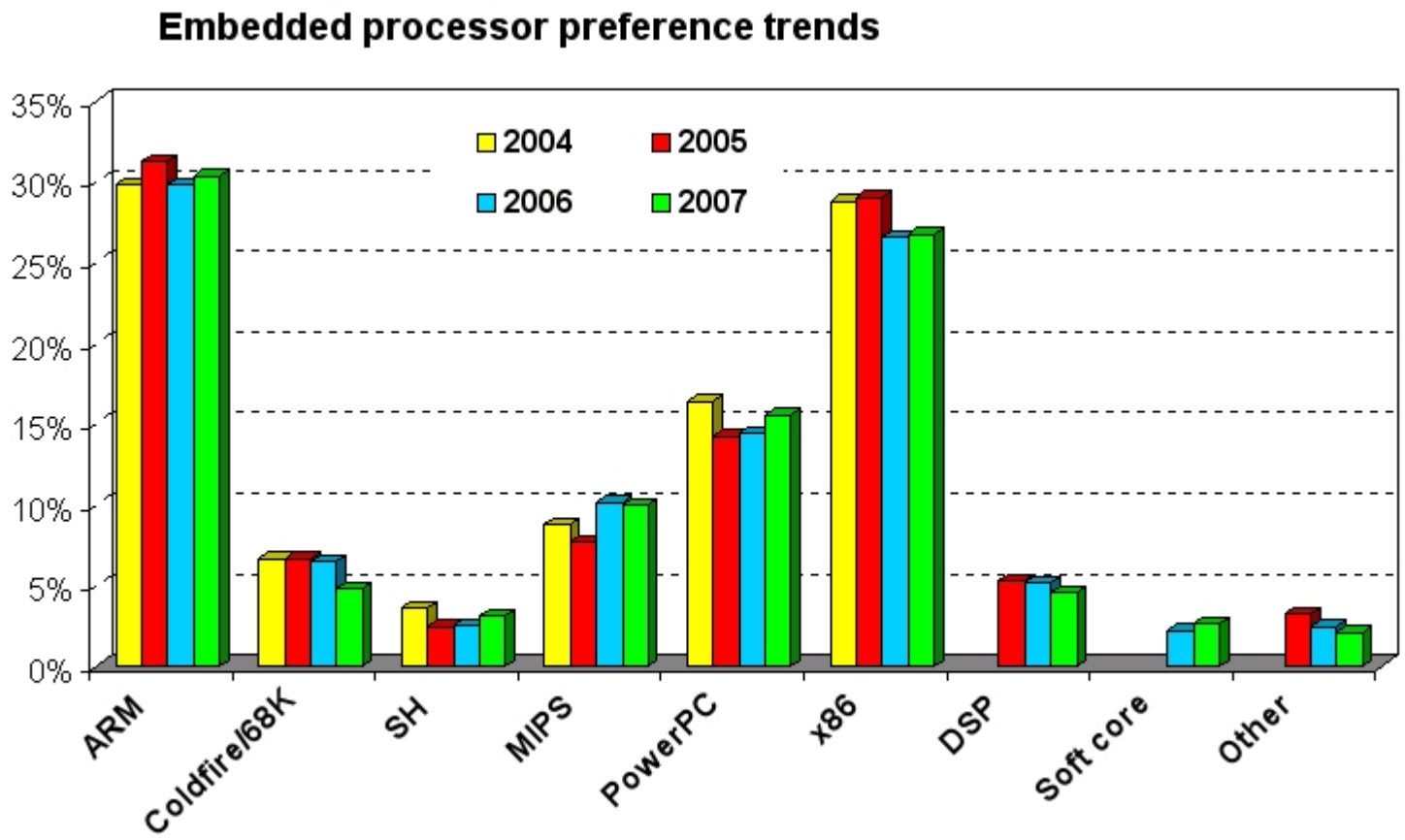
Hardware Embebido basado en SOC(2/3)

- Almacenamiento en flash
  - Hay que tener en cuenta que los ciclos de borrado son limitados, para alargar la vida útil de la memoria.
    - Existen técnicas para “gastar” de forma uniforme la memoria (wear leveling).
    - No es recomendable utilizarlas para implementar espacio de swap.

Hardware Embebido basado en SOC(2/2)

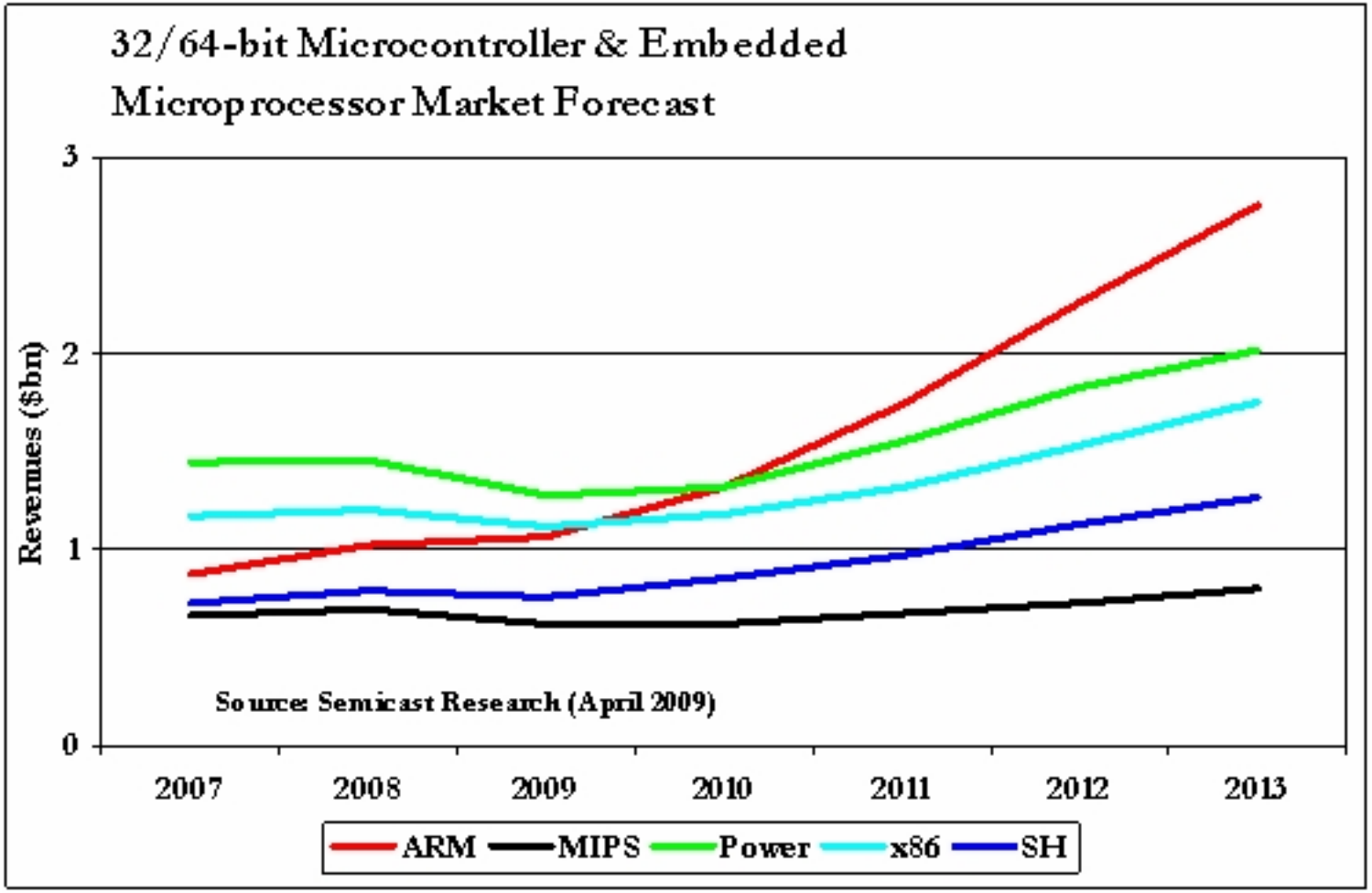
- Cantidad de RAM limitada (desde unos pocos MB a varias decenas de MB)
- Velocidad de CPU inferior a sistemas de escritorio.
- Varios buses de interconexión (I2C, SPI, USB, serial) y puertos de E/S.

Arquitecturas utilizadas en S.E.



Extraído de:
http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Snapshot-of-the-embedded-Linux-market-April-2007/

Arquitecturas utilizadas en S.E.



(Semicast Research)

Ventajas de utilizar SOC

- Dispositivos diseñados con SOC se adaptan mejor a nuevos requerimientos que los diseñados con microcontroladores
- Permiten utilizar Sistemas Operativos de propósito general, software, device drivers y periféricos utilizados en Pcs
- Acortan la curva de aprendizaje

En resumen

- Diseños basados en SOC presentan muchas ventajas respecto los sistemas basados en µC.
- Sin embargo existen casos de dispositivos muy sencillos, o que requieren muy bajo consumo, donde los diseños basados en µc siguen siendo utilizados.

Embedded Single Board Computers (SBC) (1/2)

Single-board computers (SBCs) son computadoras completas fabricadas en una única placa de circuito. El diseño es centrado en un microprocesador con RAM, almacenamiento, E/S y otras características necesarias para ser una computadora funcional en una sola placa.

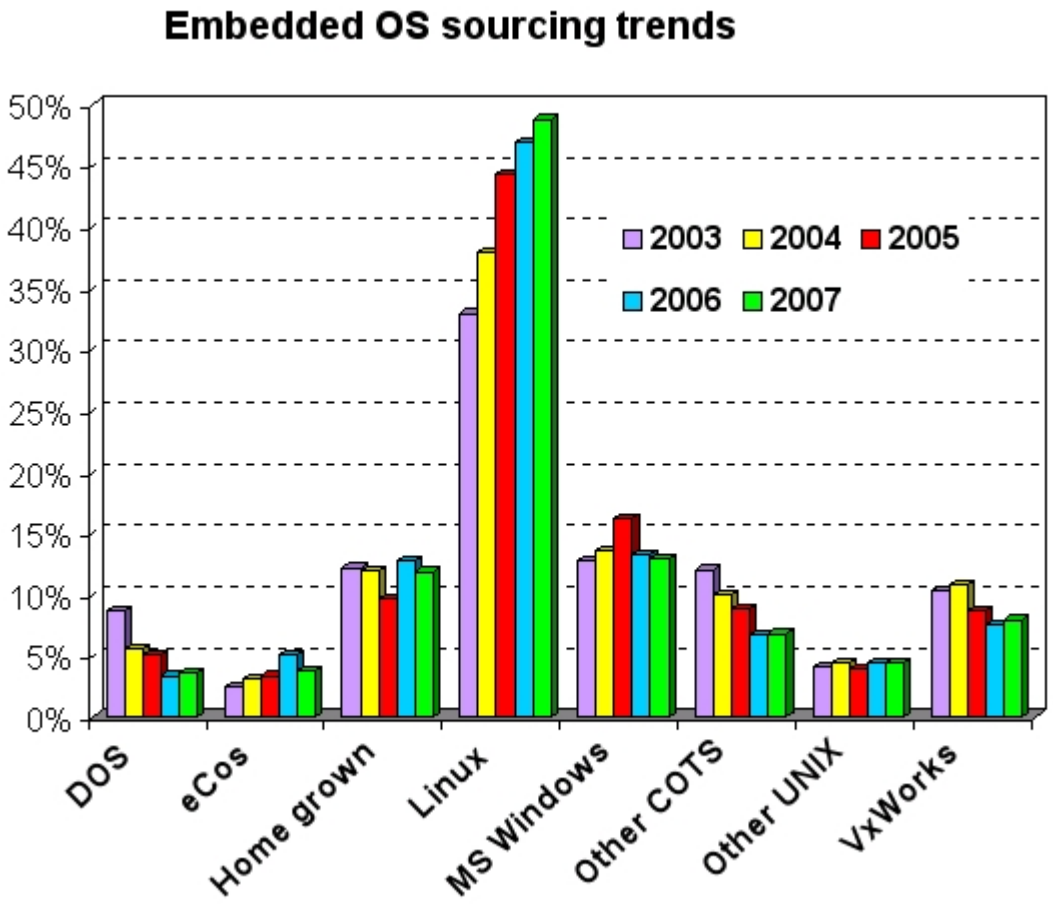
Embedded Single Board Computers (SBC)(2/2)

- Actualmente existe una gran gama de SBC basadas en SOC.
- Un SOC por si solo no puede funcionar, necesita de alimentación, acondicionamientos de señales, conectores, y algún controlador adicional.
- Más pequeñas, más eficientes energéticamente y con mayores posibilidades de E/S que un sistema tradicional.

Ventajas de desarrollar basado en SBC

- Son utilizadas muchas veces para prototipar soluciones.
- Para producción de pocas unidades no es rentable diseñar hardware. Diseñar en base a SBCs resulta una opción a tener en cuenta.
- Existen diferentes SBC según las necesidades del sistema embebido a realizar.
- Orientadas a multimedia, robótica, aplicaciones móviles.

Tendencias en S.O. para embebidos



Extraido de:
http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Snapshot-of-the-embedded-Linux-market-April-2007/

Acertijo

Funciona con Linux, pero para que sirve?

mina
network management | artificial intelligence

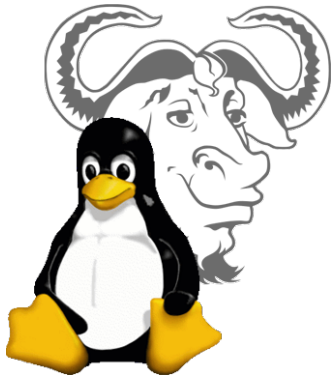
Respuesta:

Para ordeñar vacas

Linux Embebido en nuestras vidas

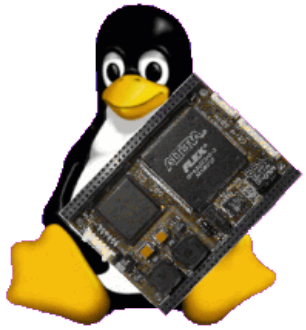
mina
network management | artificial intelligence

GNU/Linux para S.E.



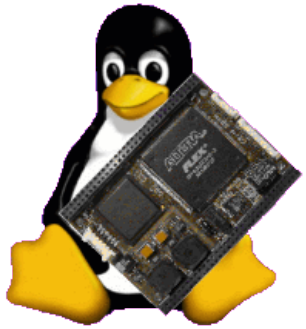
- GNU/Linux: Sistema operativo libre compuesto por el kernel (Linux) y herramientas del sistema GNU.
- El Kernel es el componente del S.O. que nos abstrae del hardware y nos facilita su uso.
- Actualmente es el sistema operativo de uso general más utilizado en sistemas embebidos.

El Kernel Linux



- Originalmente **no** fue pensando como un sistema operativo embebido.
- Escrito originalmente para la arquitectura IA-32, portado por primera vez a procesadores Motorola.
  - El proceso fue costoso, lo que implicó un rediseño de la arquitectura para hacerlo fácilmente portable.
  - Este cambio abrió la puerta para ser portado a otras arquitecturas.

El Kernel Linux



- Soporta numerosas arquitecturas (3.8.3): alpha, arm, arm64, avr32, blackfin, c6x, cris, frv, h8300, hexagon, ia64, m32r, m68k, microblaze, mips, mn10300, openrisc, parisc, powerpc, s390, score, sh, sparc, tile, um, unicore32, x86, xtensa

El kernel de Linux para sistemas embebidos

- No existe un kernel de Linux especial para sistemas embebidos y otro para estaciones de trabajo.
- Es el mismo código fuente que se compila para servidores, estaciones de trabajo o sistemas embebidos.
- Al momento de compilar la imagen del kernel, se incluye soporte únicamente para el hardware a utilizar.

Distribuciones embebidas(1/2)

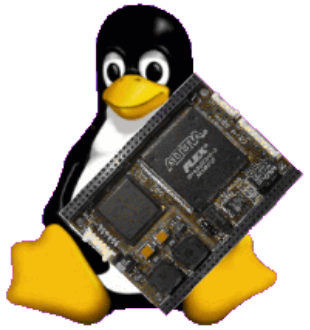
- Meego: http://meego.com/
  - Apunta a celulares, reproductores de multimedia,
  - netbooks, Tvs, IVI.
  - Soportado por Intel y Nokia (más o menos).

- Android: http://www.android.com/
  - Distribución de Google para teléfonos y tablet PCs.
  - Salvo el Kernel, un userspace muy diferente de otras distribuciones. Muy exitosa, muchas aplicaciones disponibles (varias propietarias).

Distribuciones embebidas(2/2)

- Ångström: http://www.angstromdistribution.org/
  - Apunta a PDAs y MIDs (Siemens Simpad...)
  - Binarios para arm little endian.
- Ubuntu ARM

Requerimientos mínimos

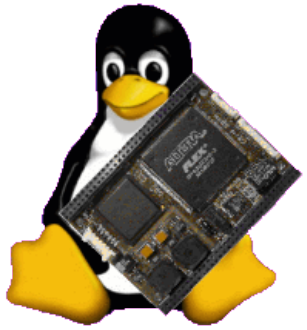


- Un CPU soportado por gcc y el Kernel Linux.
  - CPU de 32 bits
  - 4 MB RAM
  - CPU's sin MMU son también soportados mediante el proyecto uClinux.

Memory Management Unit

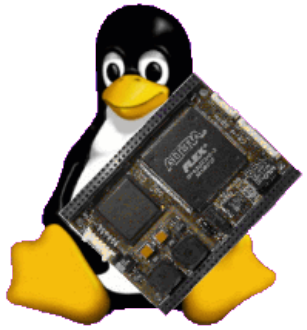
- Dispositivo de hardware.
- En los sistemas operativos modernos cada proceso dispone de un espacio de direcciones lógicas o virtuales.
- En momento de ejecución la dirección virtual es transformada en una dirección física por la *Memory Management Unit* (MMU).
- Permite implementar protección de memoria, memoria virtual.

Ventajas en el uso de GNU/Linux en S.E



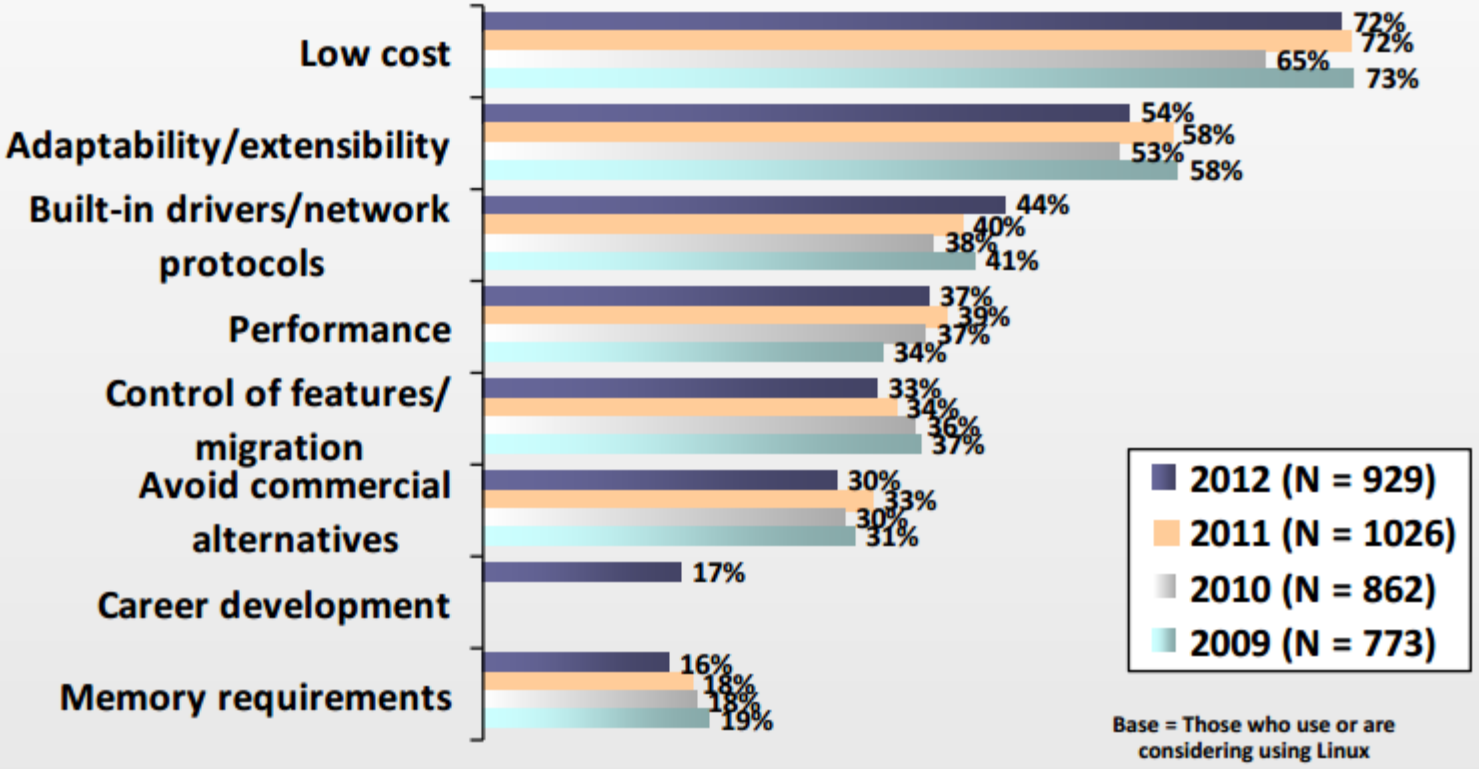
- Reutilización de bibliotecas de software existente.
  - Nos permite centrarnos en dar valor agregado a una solución y no en “reinventar la rueda”.
  - Calidad.
- Uso de lenguajes de programación de mayor nivel de abstracción (Java, Python, Lua).
- Posibilidad de prototipar y debugear en un PC muchos de los componentes del sistema.
- Brinda portabilidad de mi sistema a otras arquitecturas (independencia tecnológica).

Ventajas en el uso de GNU/Linux en S.E



- Extensiones para manejo de tiempo real.
- Mantenibilidad
- Puedo auditar el código fuente.
- Libertad de modificar el código fuente.
- Menor costo en licencias.
- Acceso más sencillo al software y a las herramientas.
- Permite hacer frente a los requerimientos cada vez más complejos impuestos por el mercado.
- Herramientas de desarrollo independientes de la plataforma

Why are you interested in embedded Linux?



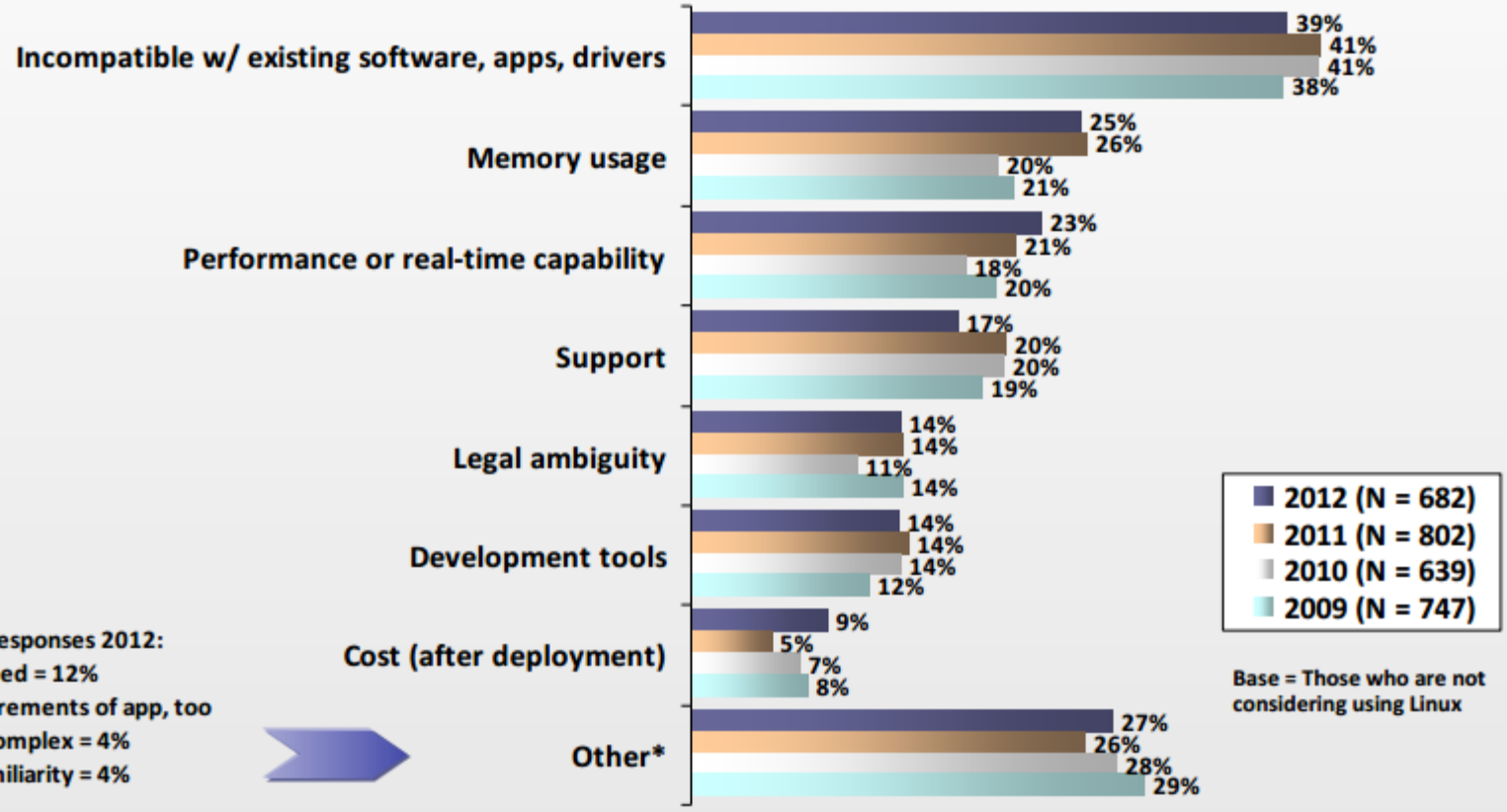
Base = Those who use or are considering using Linux

UBM Electronics

Desventajas de GNU/Linux en S.E

- Algunos drivers propietarios solo existen para versiones viejas del kernel y el fabricante no publica sus fuentes.
- Algunos Sistemas Embebidos basados en GNU/Linux toman un fork del kernel vainilla y nunca integran sus cambios.
- Muchas veces no tengo opción de que distribución usar.
  - Puede ocurrir que ni siquiera exista una distribución (usar Buildroot, OpenEmbedded).

Why are you not interested in embedded Linux?



*Other responses 2012:
- No need = 12%
- Requirements of app, too big/complex = 4%
- Unfamiliarity = 4%

Linux y el tiempo real (RT)

- RTOS: S.O con la habilidad de brindar el nivel requerido de servicio en un período de tiempo acotado (POSIX 1003.1b)
- Con el tiempo Linux fue incorporando parches RT, como los timers de alta resolución.
- Es raro tener restricciones de tiempo blandas que un Kernel Vanilla no pueda proveer (Latencia de peor caso ~ 1ms).
- Si se necesita menor latencia usar parches para el Kernel:
  - PREEMPT_RT Ofrece una latencia máxima de 100µs

Estado actual de los S.E

- Cada vez se dispone de hardware con mayores prestaciones, que incluso permite ejecutar un S.O de propósito general con RT y seguir siendo económicamente viable.
- El mercado le exige más requerimientos a los dispositivos embebidos.

Estado actual de los S.E.

- Hoy en día los S.E. tienen más puntos en común con los sistemas de propósito general que en el pasado.
- Herramientas de desarrollo de mayor nivel de abstracción.
- Reutilización de software y periféricos utilizados en sistemas de propósito general
- Es necesario contar con equipos interdisciplinarios.

Componentes de Software

- Cross-toolchain: Compilador que corre en la máquina de desarrollo, pero genera código para el sistema embebido.
- Bootloader: Iniciado por el hardware. Responsable de la inicialización básica, cargar y ejecutar el kernel.
- Kernel Linux: Contiene el manejo de los procesos, memoria, red, drivers y provee servicios para las aplicaciones de usuario.
- Biblioteca de C: Interfaz entre el Kernel y las aplicaciones (glibc, µClibc).
- Bibliotecas y aplicaciones: reutilizadas o desarrolladas por nosotros.

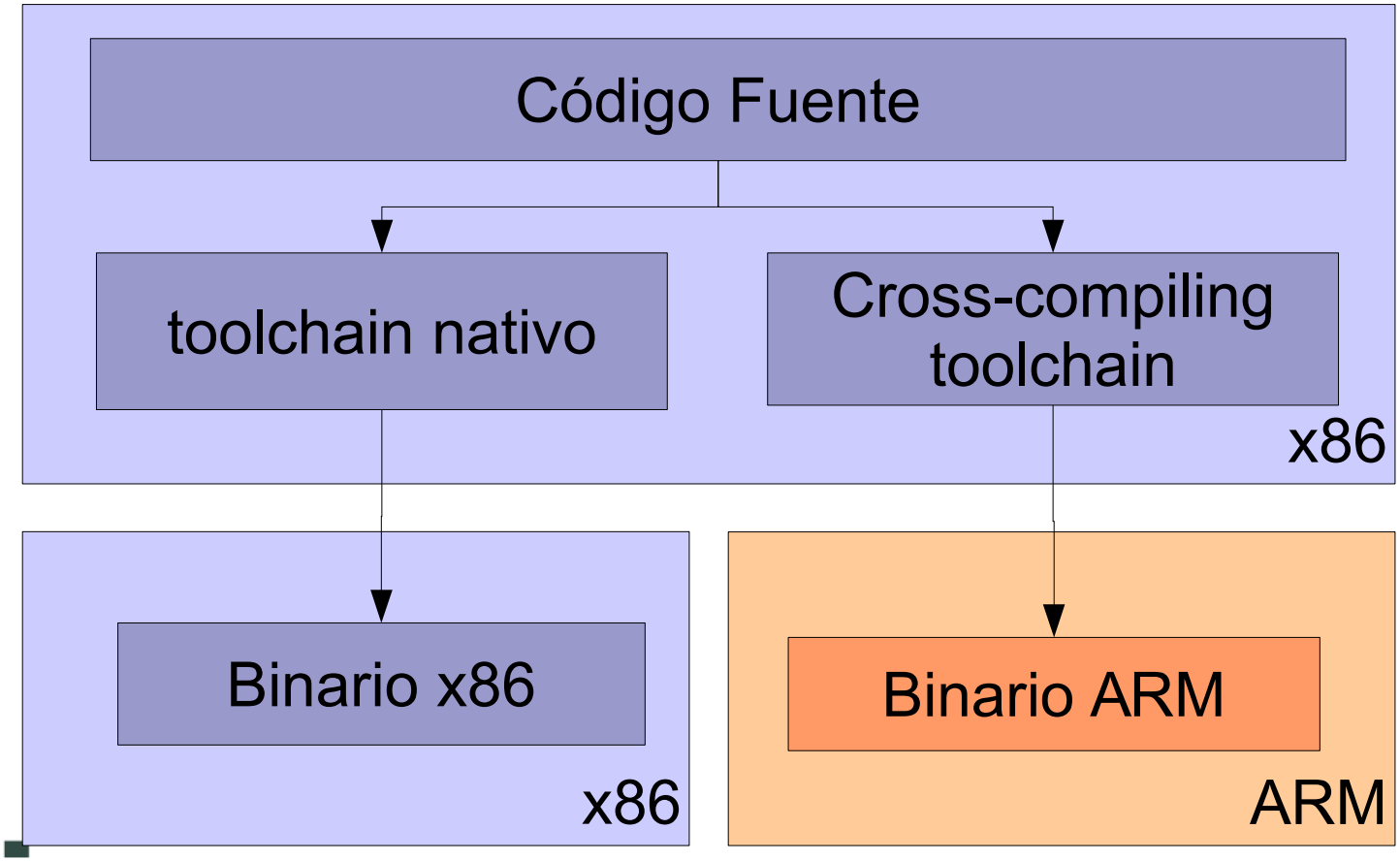
Toolchain

- Las herramientas usuales de desarrollo disponibles en una estación de trabajo GNU/Linux es un **toolchain nativo**
- Este toolchain ejecuta en la estación de trabajo y genera codigo para esta estación de trabajo, generalmente x86

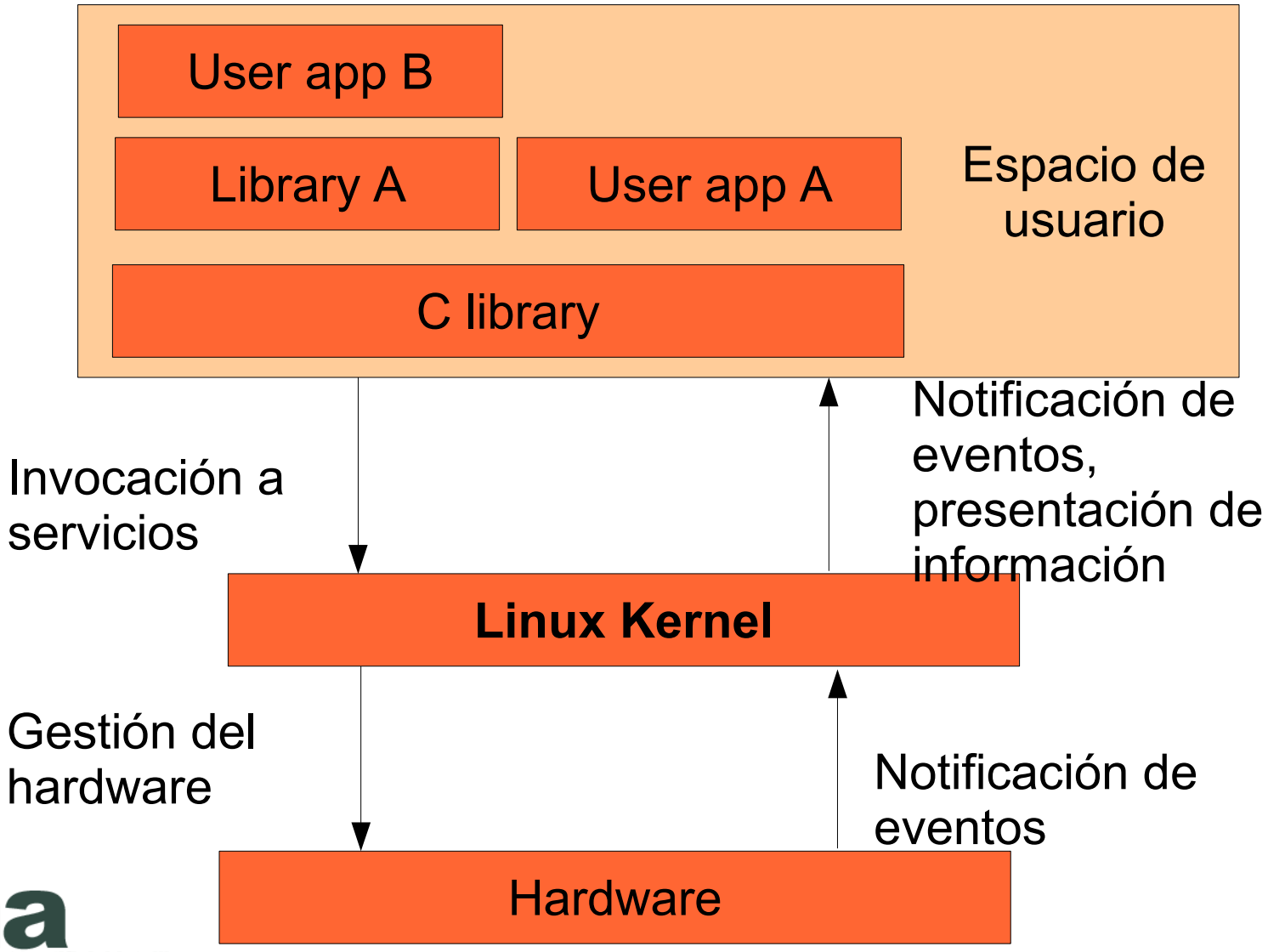
Toolchain

- En sistemas embebidos, es usualmente imposible o poco interesante utilizar un toolchain nativo.
  - El target es muy restrictivo en términos de almacenamiento o memoria.
  - El target es muy lento comparado con la estación de trabajo.
  - Puede no ser deseado de instalar todas las herramientas de desarrollo en el target.
- Por lo tanto **cross-compiling toolchains** son generalmente utilizados. Ejecutan en el workstation pero generan código para el target.

Toolchain


Código Fuente
toolchain nativo
Cross-compiling toolchain
x86
Binario x86
x86
Binario ARM
ARM
Equipo de compilación
Equipo a ejecutar

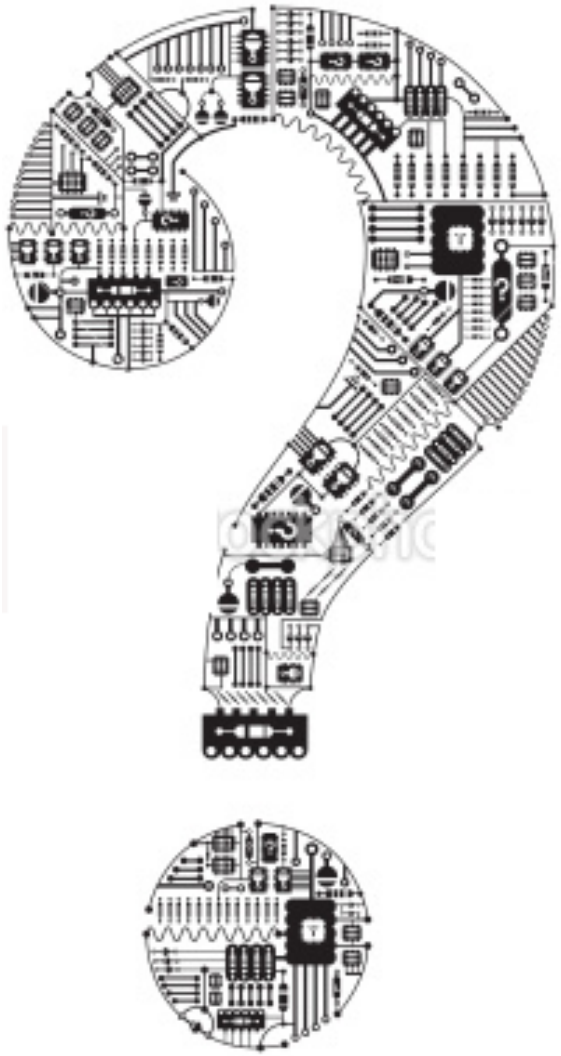
El kernel de Linux en el sistema



Bibliografía

- Inyección de defectos y su aplicación en dispositivos USB, http://www.sase.com.ar/2011/files/2010/11/SASE2011-Inyeccion_defectos_y_su_aplicacion_en_USB.pdf
- http://www.linuxfordevices.com/
- http://free-electrons.com/
- http://kernel.org
- Building Embedded Linux Systems
- Embedded Linux System Design and Development (Auerbach, 2006)

Preguntas



mina
network management | artificial intelligence