Introducción a los sistemas empotrados

PID_00247325

Màrius Montón Macián
Ignasi Vilajosana Guillén

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Índice

1. Qué es un sistema empotrado 5

2. Aplicaciones 7
2.1. Industria militar y aeroespacial 8
2.2. Automoción 9
2.3. Tecnología médica 9
2.4. Comunicaciones 9
2.5. Electrodomésticos y electrónica de consumo 9
2.6. Automatismos industriales y procesos de control 10
2.7. Logística 10
2.8. Agricultura 11
2.9. Infraestructuras 11
2.10. Seguridad 11

3. Consideraciones de diseño 12

4. El proceso de desarrollo 15
4.1. Otros aspectos que hay que tener en cuenta en el diseño 16
4.1.1. Determinación de los requisitos del sistema 16
4.1.2. Consideraciones en cuanto el proyecto 18
4.1.3. Consideraciones en cuanto al diseño del sistema empotrado 18

1. Qué es un sistema empotrado

Un **sistema empotrado** es un sistema informático de uso específico que está encapsulado totalmente por el dispositivo que controla. Los sistemas empotrados constituyen un sistema computacional fruto de la combinación de hardware y software. Esta combinación tiene como misión llevar a cabo una funcionalidad o un conjunto de funcionalidades determinadas. Se denominan *empotrados* porque normalmente forman parte de un sistema completo o con funcionalidades más generales.

Normalmente, un sistema empotrado está basado en un microcontrolador (μC), que controla una función o funciones específicas de un sistema. Sin embargo, el sistema no está diseñado para ser programado por el usuario final, como podría suceder con un PC, en el que ahora lo utilizamos como un procesador de textos y al cabo de un rato le instalamos un juego. Es decir, un usuario final puede configurar el dispositivo empotrado, pero normalmente no puede modificar la funcionalidad para la que ha sido construido.

Así pues, un sistema empotrado está diseñado para llevar a cabo específicamente la tarea para la que ha sido programado. La tarea puede no ser única y, por lo tanto, se pueden incluir en ella varias opciones que el usuario puede seleccionar (como si se tratara de los distintos programas de lavado en una lavadora). Este rasgo es diferencial con relación a los ordenadores, o al menos lo era hasta hace muy poco. Actualmente, un grupo minoritario de los dispositivos empotrados han experimentado una fuerte evolución y ya casi ofrecen funcionalidades más cercanas a los PC que a los electrodomésticos.

Ejemplo

Un buen ejemplo del acercamiento de los electrodomésticos a las funciones próximas al PC lo tenemos en la telefonía móvil, con dispositivos con capacidades equivalentes a las de un ordenador.

Las técnicas de diseño de sistemas empotrados han posibilitado el desarrollo de productos más pequeños, más rápidos, más robustos y, sobre todo, más baratos que se están introduciendo en casi todos los dispositivos que rodean nuestra vida diaria. El diseño VLSI*, ha permitido crear transistores extremadamente pequeños que se pueden integrar por millones en pequeños circuitos integrados. Gracias a esto, se han podido construir sistemas más complejos de una manera modular.

*** Sigla de *very large scale integrated circuits*, circuitos integrados a muy gran escala.**

El desarrollo de un sistema empotrado está condicionado siempre por su robustez y eficiencia, y con el condicionante de que ninguno de sus usuarios final es consciente de su existencia.

Ejemplo

El sistema de control del nivel de carburante de un vehículo está constituido por varios sistemas empotrados. Curiosamente, solo somos conscientes de su existencia cuando dejan de funcionar.

En la década de 1970 aparecieron los primeros microcontroladores de 8 bits producidos por Motorola (M6800) e Intel (8080), junto a las primeras memorias progra-

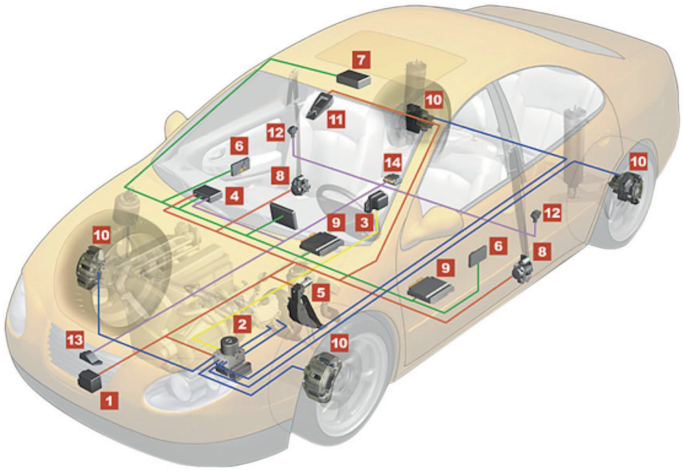
mables (PROM*). Las aplicaciones basadas en los primeros microcontroladores eran muy sencillas (microsoftware, *firmware*, como lo conocemos ahora). Normalmente, hacían uso del lenguaje de ensamblador, que permitía desarrollar aplicaciones de como máximo centenares de líneas de código, con las que se podía sustituir diseños de hardware muy complejos por sistemas programables mucho más sencillos.

*** Sigla de *programable read only memory*.**

Hoy en día, se están desarrollando aplicaciones para sistemas empotrados de miles de líneas de código, utilizando múltiples microcontroladores, componentes integrados VLSI, diferentes niveles de memoria, convertidores analógico-digitales y otros muchos componentes que permiten que los sistemas empotrados sean usados en casi todos los dispositivos y máquinas que existen en la actualidad.

Los sistemas empotrados interactúan con una gran variedad de dispositivos analógicos y digitales. El desarrollador de sistemas empotrados se debe enfrentar a la interacción con dispositivos analógicos y digitales, tiene que comprender cómo funcionan sensores y transductores y cómo sus señales son digitalizadas o viceversa. También, cada día más, los sistemas empotrados utilizan conexiones de red, en muchos casos con protocolos específicos condicionados por la naturaleza de estos dispositivos. De este modo, la tarea de desarrollar un sistema empotrado es heterogénea y requiere que el desarrollador tenga un amplio abanico de habilidades y conocimientos.

Figura 1. Sistemas empotrados en un vehículo



1. Control de crucero adaptativo
2. Sistema electrónico de frenos MK60E
3. Grupo de sensores
4. Entrada del transmisor de datos
5. Retroalimentación del pedal de aceleración
6. Unidad de control de las puertas
7. Unidad de control del techo solar
8. Pretensor reversible del cinturón de seguridad
9. Unidad de control de los asientos
10. Frenos
11. Sistema prechoque óptico (*closing velocity sensor*)
12. Satélites laterales
13. Sensor delantero
14. Unidad de control de los airbags

2. Aplicaciones

El desarrollo tecnológico de los últimos años en el ámbito de los sistemas empotrados y de la microelectrónica en general ha facilitado el desarrollo de aplicaciones existentes y también ha favorecido la aparición de nuevas.

Los sistemas empotrados han permitido el desarrollo de sistemas y aplicaciones:

- de bajo coste,
- en muchos casos descentralizadas,
- con bajos consumos energéticos,
- con más robustez y tolerancia a los fallos,
- con mejor apoyo a la operación con tiempo real,
- con potencial para ser conectadas en red,
- con control o acceso remoto,
- con más densidad de sensores o actuadores,
- con alto nivel de integración en otros sistemas u objetos y
- con tamaños más reducidos o con la posibilidad de ser usadas donde antes no era posible.

Una gran parte de los sistemas empotrados se pueden considerar dispositivos *sensores* que mediante interfaces son capaces de obtener datos. En esta categoría se incluyen todos los dispositivos que utilizan sensores (sensores de temperatura, humedad, sísmicos, acústicos, magnéticos, etc.) y también todo tipo de contadores y dispositivos de medida mecánica (como tacómetros, amperímetros, contadores de pulsos, etc.). Por otro lado, los **sistemas actuadores** son aquellos que gracias a estímulos recibidos llevan a cabo acciones en su ámbito de actuación.

Ejemplo

Los sistemas actuadores accionan motores, abren o cierran interruptores y notifican eventos, entre otras funciones.

En los últimos años y gracias al auge de las comunicaciones inalámbricas, muchos de los sistemas empotrados existentes han evolucionado para operar de una manera remota sin la necesidad de conexiones cableadas. Esto ha permitido el desarrollo de nuevas aplicaciones que hasta el momento no eran posibles. Sin embargo, han aparecido retos nuevos, puesto que el hecho de poder comunicar un sistema empotrado inalámbrico ha permitido ubicarlos en lugares donde ni siquiera hay energía eléctrica. Por lo tanto, el consumo y la eficiencia de los dispositivos ha devenido crucial.

Tanto es así que la mayoría de los esfuerzos en el desarrollo de la microelectrónica y del software (sistemas operativos y bibliotecas) han ido dirigidos a la reducción de los consumos energéticos de estos dispositivos. Esto se ha hecho patente en diferentes vertientes:

1) Microelectrónica energéticamente eficiente. Gracias a la reducción del tamaño de los chips y de los componentes electrónicos que integran un sistema empotrado, junto con técnicas que permiten apagar literalmente algunos componentes del dispositivo cuando no son usados, se ha reducido de una manera notable el consumo de estos dispositivos. También hay que destacar la aparición de sistemas de recogida de energía del entorno (*energy harvesting*), que han permitido la prolongación de la vida operativa de los sistemas empotrados.

2) Sistemas operativos en tiempo real. Los avances en sistemas operativos en tiempo real de propósito específico, incluyendo planificadores de tareas prioritarias y multitarea y en muchos casos con conceptos de justicia (*fairness*, en inglés), han permitido una optimización del tiempo de procesador. También han aparecido técnicas que permiten detectar cuándo no hay más tareas por ejecutar y, de una manera automática, poner el sistema en un modo dormido (*sleep*). Los sistemas basados en eventos y asíncronos han favorecido la utilización de interrupciones y han evitado también el uso innecesario de esperas o encuestas.

3) Estándares de comunicación que optimizan el consumo energético. Las comunicaciones inalámbricas son las que más han evolucionado. Han aparecido estándares para la comunicación pensados para entornos con altas restricciones energéticas en los que la pila de protocolos está pensada para minimizar este aspecto. El 802.15.4 es la especificación hecha por el IEEE del protocolo para redes de sensores y entornos industriales en el que se optimiza la eficiencia energética de la capa física. Define tasas de transmisión, modulaciones y estructura de los paquetes. Han aparecido diferentes implementaciones de la capa MAC, que, entre otras funciones, permiten un acceso aleatorio CSMA/CA en redes con poco tránsito, o acceso múltiple TDMA/FDMA para redes con tránsito constante, basándose en protocolos complejos de sincronización y de encaminamiento de la información.

Gracias a estos avances, encontramos aplicaciones de los sistemas empotrados en casi todos los sectores. A continuación, detallamos los más representativos.

2.1. Industria militar y aeroespacial

Encontramos los sistemas de propósito específico en sistemas de control, ubicación y monitorización en rescates u operación en desastres naturales, también en sistemas de control de proyectiles y armamento o de control de dispositivos voladores no tripulados, entre otros. Una parte significativa del coste de los aviones proviene de los equipos de procesamiento de información, incluyendo sistemas de control de vuelo, sistemas para evitar colisiones o sistemas de información para los pilotos. En muchos casos, la operación en tiempo real, la robustez y los mecanismos de posicionamiento y comunicación inalámbricos son clave para este tipo de dispositivos.

UAV

España y Estados Unidos son dos de los fabricantes más importantes de vehículos aéreos no tripulados (en inglés, *unattended aerial vehicle*, UAV).

2.2. Automoción

La mayoría de los países han establecido leyes que obligan a los fabricantes de vehículos a que incluyan ciertos sistemas de seguridad, como son los sistemas de airbags controlados por sistemas de control empotrados al vehículo, sistemas de control del motor, controles del sistema de freno (ABS) o sistemas de estabilidad (ESP), entre otros. Otros componentes de los vehículos son también sistemas empotrados, los GPS, los sistemas de alarma, los controles de climatización, etc.

2.3. Tecnología médica

Gracias a sus características y tamaños, los sistemas empotrados son usados en el ámbito de la medicina e integran todo tipo de equipamiento para la monitorización y el control en tiempo real. Incluso en muchos casos, los sistemas son adheridos a los pacientes para ser observados de una manera permanente, sobre todo en el caso de gente mayor. Hay algunas experiencias de múltiples sistemas sensores adheridos al cuerpo del paciente y que se comunican entre ellos para encaminar la información hacia el centro receptor y formar una red de comunicaciones personal llamada *red de área personal**.

*** PAN, *personal area network*, en inglés.**

2.4. Comunicaciones

La mayoría de los dispositivos que conforman las redes de comunicaciones están constituidos por un microcontrolador, su memoria y una interfaz de comunicación. Así, los encaminadores (*routers*) o los conmutadores (*switch*), entre otros, no dejan de ser sistemas de propósito específico. El auge de las comunicaciones ha provocado que cada vez aparezcan más dispositivos multiinterfaz que permiten la comunicación utilizando diferentes tecnologías. En este apartado no podemos olvidar los teléfonos móviles, que son la tecnología que ha experimentado un crecimiento más destacado en los últimos años. En este ámbito, son especialmente importantes las comunicaciones de radiofrecuencia y los diseños de sistemas energéticamente eficientes.

2.5. Electrodomésticos y electrónica de consumo

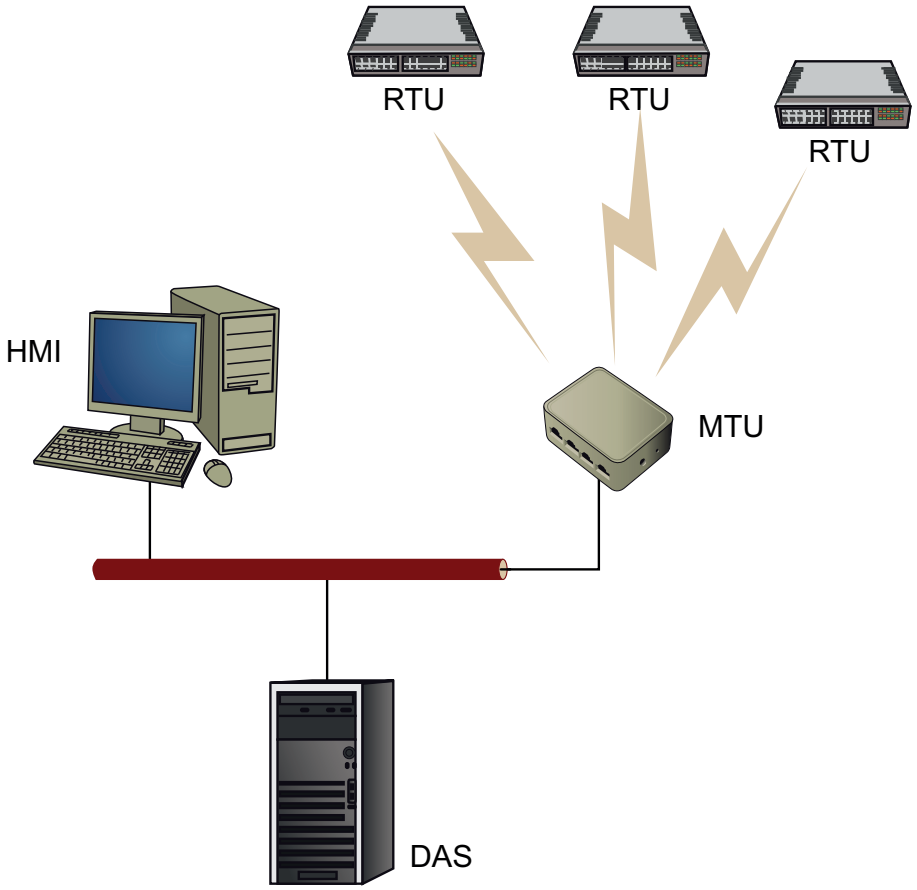
Desde los teléfonos móviles hasta los reproductores de música, pasando por las cocinas de inducción, los aires acondicionados y las calefacciones, encontramos uno o varios sistemas de propósito específico que rigen el funcionamiento de estos dispositivos. Tanto es así que muchos de los electrodomésticos que tenemos en casa ya incluyen interfaces o puertos estándar de expansión que permiten extender las funcionalidades de los aparatos, por ejemplo, añadiendo un control remoto. En este contexto, no hay que olvidar los sistemas de entretenimiento y videojuegos, extendidos cada vez más con dispositivos de hardware que aumentan su interactividad mediante sensores y actuadores.

2.6. Automatismos industriales y procesos de control

Los sistemas SCADA* y autómatas de control son implementados cada vez más con sistemas de propósito específico. A pesar de que la industria tiene una gran inercia a la hora de integrar nuevas tecnologías, los sistemas empotrados con comunicaciones inalámbricas abaratan en extremo los costes de instalación, y los hace muy interesantes y rápidamente amortizables. Hace unos años, los sistemas de control solían estar centralizados en un computador central y gestionados por un software SCADA. Estos ordenadores eran un punto crítico para la operación de la industria, puesto que en caso de fallo se paraban muchos procesos. El abaratamiento de los sistemas empotrados, junto con las comunicaciones inalámbricas, ha permitido llevar un control descentralizado, puesto que los propios sistemas sensores son capaces de procesar los datos y enviarlos a sistemas web en los que la replicación y la tolerancia a fallos es mucho más sencilla.

* ***Supervisory control and data acquisition*, en inglés.**

Figura 2. Arquitectura SCADA



Un servidor central, un centro de control y los dispositivos sensores.

2.7. Logística

En muchas aplicaciones de logística se equipan las materias primas con identificadores de radiofrecuencia (*tags*), o RFID, que permiten hacer el seguimiento de los productos.

Tanto el control como la lectura de estos identificadores se lleva a cabo mediante sistemas de control basados en sistemas de propósito específico.

2.8. Agricultura

La aparición de múltiples sistemas de irrigación, fertilización del suelo y control del crecimiento de vegetales y plantas empieza a introducir un cambio en los modelos de agricultura tradicionales. A pesar de los costes de introducir la tecnología en el campo, algunos estudios demuestran grandes aumentos de la productividad, lo que hace muy interesante esta tecnología para el sector. Por ejemplo, se han empezado a aplicar controles exhaustivos en la viticultura y la huerta de regadío dada la relevancia de las mejoras que los dispositivos empotrados pueden aportar a la producción agrícola en estos sectores y el incremento evidente del volumen de negocio que ello significa.

Control de las viñas haciendo uso de un dispositivo empotrado dotado de interfaz de comunicación inalámbrica.

2.9. Infraestructuras

Gracias a las capacidades de comunicación inalámbrica, los bajos consumos energéticos y las reducidas dimensiones, los sistemas empotrados pueden ser usados como sistemas de medición en lugares casi inaccesibles en todo tipo de infraestructuras. Esto permite controlar la salud de edificios, puentes, presas, etc. La evolución de estos sistemas ha dado pie a la aparición de términos como *smart cities* o *smart infraestructures*, que denominan ciudades llenas de sistemas de medición inalámbricos puestos al servicio del control de las grandes infraestructuras o ciudades. Incluyen sistemas de control de tráfico y de polución, sistemas de control acústico o de luminosidad a gran escala, por mencionar algunos.

SmartSantander

En el 2011, la ciudad de Santander ha iniciado un proyecto, llamado SmartSantander, que pretende instalar más de 20.000 sensores en toda la ciudad con el objetivo de dar servicio a la ciudadanía.

2.10. Seguridad

En los últimos años, los sistemas de seguridad han ido incorporando todo tipo de sistemas de propósito específico, desde sistemas de captura y procesamiento de imágenes hasta sistemas sensores y actuadores, pasando por sistemas de autenticación de personas, por ejemplo, mediante la huella dactilar o el escáner de retina.

3. Consideraciones de diseño

El proceso de diseño de un sistema empotrado se basa en la creación de un modelo del dispositivo en cuestión. Normalmente, el proceso de diseño requiere una metodología específica que debe tener en cuenta los requisitos siguientes:

- **Jerarquía**. Los seres humanos generalmente no son capaces de comprender sistemas muy complejos formados por muchos objetos fuertemente interrelacionados. De este modo, hacer uso de jerarquías permite concebir los sistemas de una manera estructurada y permite focalizar el diseño en cada una de las partes. Podemos considerar dos tipos de jerarquías:

  – **Jerarquías de comportamiento**. Contienen los objetos necesarios para describir el comportamiento del sistema: estados, eventos, señales de entrada y de salida, etc.

  – **Jerarquías estructurales**. Describen la composición física del sistema. Por ejemplo, procesadores, memorias, actuadores y sensores. A su vez, los procesadores incluyen registros, multiplexores, sumadores, etc. Los multiplexores están constituidos, a su vez, por puertas lógicas.

- **Diseño orientado a componentes**. El sistema debe poder ser especificado usando componentes con funcionalidades definidas. Tiene que ser sencillo derivar el comportamiento de un sistema del comportamiento de sus componentes. Si dos componentes están interconectados, el comportamiento de ambos ha de ser predecible.

- **Concurrencia**. Dado que el sistema es una composición de componentes, hemos de esperar que estos trabajen de una manera simultánea y, por lo tanto, dar pie a situaciones de concurrencia. Para el diseñador es difícil prever las situaciones de concurrencia, y por ello es un aspecto crítico que hay que tener en cuenta en la fase de diseño. Cabe señalar que, en muchos casos, los posibles fallos se deben a un conocimiento incompleto de las situaciones de concurrencia.

- **Sincronización y comunicación**. Los componentes se deben poder comunicar y sincronizar. Se han de poder expresar prioridades y ponerse de acuerdo en el uso de recursos (exclusión mutua).

- **Comportamiento temporal**. Muchos de los sistemas empotrados son sistemas en tiempo real. Así, los requisitos de tiempos son críticos y tienen que ser especificados durante la fase de diseño del sistema. Es crucial en el diseño de algunos

algoritmos que se ejecutarán en un sistema empotrado demostrar que el algoritmo finaliza de manera determinista en un tiempo determinado. Generalmente, se emplean técnicas para lo siguiente:

– **Medir el tiempo empleado**. Para muchas aplicaciones es necesario medir el tiempo que ha pasado desde un cierto instante hasta que se ha acabado la ejecución.

– **Retardar procesos durante un periodo**. Típicamente los lenguajes de programación para sistemas en tiempo real ofrecen métodos para introducir retardos. Sin embargo, en muchos casos estos métodos no son muy precisos, puesto que utilizan el planificador de tareas del sistema operativo, que introduce ciertos retardos debido a los cambios de contexto.

– **Posibilidad de especificar temporizaciones (*timeouts*)**. En muchas situaciones lo que se quiere es esperar un cierto acontecimiento. Puede suceder, sin embargo, que este acontecimiento no suceda en un periodo de tiempo finito y, por lo tanto, queramos que esto se nos notifique para evitar que el sistema espere eternamente. Un caso común lo podríamos encontrar cuando se espera un paquete de respuesta a través de la red y queremos ser notificados si este no se ha recibido en un periodo de tiempo determinado. Los lenguajes de programación en tiempo real suelen ofrecer este tipo de funcionalidades.

- **Métodos para especificar retardos y planificaciones**. Para ciertas aplicaciones, hay que completar ciertos cálculos en un periodo de tiempo concreto. Desafortunadamente, la mayoría de los lenguajes de programación no permiten imponer restricciones de tiempo; a la vez, el hardware presenta cada vez más un comportamiento más impredecible (temporalmente hablando), debido al uso de memorias caché, *pipelines* de ejecución, ejecución especulativa, priorización de tareas, interrupciones, etc., lo cual hace muy difícil que se pierda el tiempo de ejecución de un algoritmo.

Ejemplo

Si el conjunto de los sensores de aceleración de un coche detecta una fuerte desaceleración, el airbag se debe poner en marcha con un tiempo no superior a 10 ms; en este contexto, el sistema tiene que garantizar que se tomará la decisión de activar o no el airbag en este tiempo.

- **Comportamiento definido por estados**. Es muy útil utilizar máquinas de estado autómatas para definir el comportamiento del sistema. Sin embargo, la temporización no se puede modelar en una máquina de estados ni tampoco se pueden modelar jerarquías complejas.

- **Gestión de eventos**. Fruto de la naturaleza reactiva de los sistemas empotrados, se requieren mecanismos para describir eventos. Los eventos pueden ser externos (causados por el entorno) o internos (causados por el comportamiento del propio sistema).

- **Comportamiento definido por excepciones**. En la mayoría de los sistemas hay excepciones. Para que nuestros sistemas sean robustos, se tienen que poder definir acciones que gestionen las excepciones de una manera sencilla.

- **Soporte al diseño de sistemas grandes**. En ocasiones, se deben desarrollar sistemas grandes y complejos con funciones muy específicas. Se tienen que usar meto-

dologías para llevar a cabo estos diseños; entre otras, la orientación a objetos o a componentes son capitales. Se deben reaprovechar las técnicas y las metodologías existentes para minimizar los esfuerzos en este sentido.

- **Legibilidad**. El diseño se ha de especificar en un documento que tiene que ser legible. La documentación es tan importante o más que el propio diseño. Se deben desarrollar documentos que puedan ser leídos por humanos y también especificaciones que puedan ser fácilmente transformadas en programas.

- **Portabilidad y flexibilidad**. Las especificaciones han de ser independientes del hardware específico para que puedan ser utilizadas de una manera sencilla en otras plataformas o componentes. Idealmente, se debería poder cambiar la plataforma de hardware sin que esto afectara a la especificación –aunque, en la práctica, siempre son necesarios algunos cambios.

- **Finalización**. Tiene que ser posible identificar el ciclo de vida de la aplicación a partir de su especificación. Se deben poder determinar sus estados de finalización.

- **Soporte para dispositivos de E/S* no estándar**. Muchos sistemas de propósito específico emplean dispositivos de E/S no estándar. Se tienen que describir las entradas y salidas de una manera conveniente.

- **Propiedades no funcionales**. Se deben tener en cuenta también otras propiedades no funcionales, como la tolerancia a fallos, el tamaño, la extensibilidad, la esperanza de vida del sistema, el consumo, el peso, la usabilidad, el aspecto, la compatibilidad electromagnética, etc. Se tienen que definir estas propiedades formalmente.

*** Entrada/Salida (*Input*/*Output*, I/O, en inglés).**

4. El proceso de desarrollo

La concepción del sistema en su totalidad es parte de una idea de una persona, o de un equipo de diseño o de un encargo a medida hecho para un «cliente» que quiere resolver un problema determinado.

Podemos establecer un serie de tareas previas que culminarán en la elaboración de un anteproyecto, a partir del cual se tomarán las decisiones de seguir adelante o descartar el proceso:

1) Determinación de los requisitos globales del sistema.

2) Selección del microprocesador, microcontrolador o DSP más adecuados.

3) Selección de la tecnología de fabricación más adecuada.

4) Elección de la memoria y del sistema operativo, si procede.

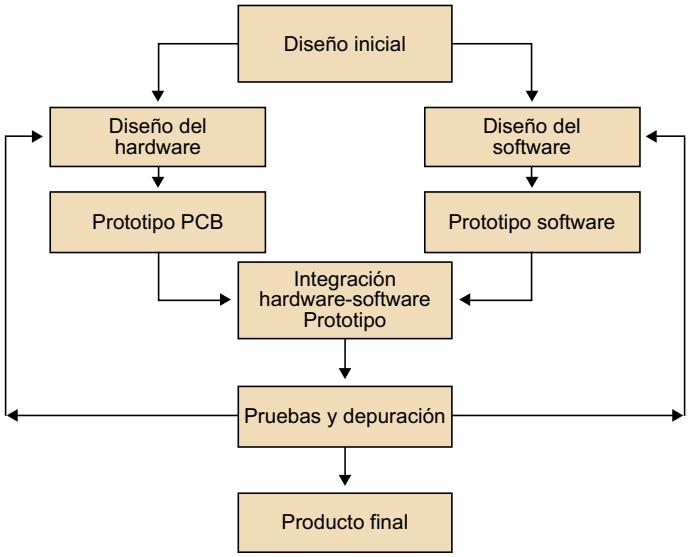
5) Determinación de las entradas/salidas, comunicaciones, etc.

6) Determinación de las necesidades de homologación en función de la aplicación.

7) Selección del equipo humano más oportuno para su desarrollo, selección de proveedores hardware y software, etc.

8) Elaboración de un anteproyecto, cuanto más detallado mejor, que nos permita evaluar la viabilidad técnica y económica del sistema, y también generar un presupuesto de costes tan veraz como sea posible. Desde la primera fase, conocida como **diseño previo del sistema**, hasta la última, que consiste en la decisión del producto final, se debe vigilar el orden de ejecución de las tareas y encabalgar todas las que sea posible, de modo que se obtenga un producto fiable en el mínimo tiempo posible de acceso al mercado (*time-to-market*).

Figura 3. Diagrama de fases para el diseño de un sistema empotrado



En esta figura se muestra un diagrama simplificado con las fases que hay que seguir y que comentamos a continuación:

1) Diseño inicial del sistema, que incluye toda una serie de tareas que acabarán en la elaboración de un esquema eléctrico del sistema y en un diseño de necesidades de software.

2) A partir del esquema y de la forma física de cada uno de los componentes que intervienen en él, diseño hardware del sistema. Esta tarea incluye el posicionamiento de cada uno de los componentes y el encaminamiento de las pistas conductoras que llevarán a cabo las interconexiones necesarias entre los pivotes de los componentes, lo que generará un prototipo de placa de circuito impreso (PCB). A partir de esta PCB, una vez producida, se lleva a cabo el montaje o el ensamblaje de todos y cada uno de los dispositivos mediante el procedimiento de soldadura más adecuado. Este proceso acaba en un prototipo de hardware.

3) Desarrollo del prototipo de software con la programación inicial del microcontrolador o de los microcontroladores que formen parte del sistema empotrado.

4) Integración del hardware/software mediante el vertido o la programación en el circuito de los microcontroladores. Así, se dispondrá del primer prototipo preparado para proceder a su test y depuración.

5) Pruebas y depuración del software y hardware mediante el uso de prototipos hasta llegar a la versión final. Si se detectan errores en el hardware, será necesario rediseñar la placa y volver a empezar el proceso. Si los errores son de software, el proceso es similar, salvo que es menos costoso en cuanto a materiales que en cuanto a horas de ingeniería.

6) Obtención del producto final. Después del resultado satisfactorio en todas las pruebas se conseguirá el producto final. En el caso de previsiones de fabricación masiva, habrá que fabricar preseries y probarlas para minimizar, así, los imprevistos de cara a la fabricación en serie de altas cantidades.

4.1. Otros aspectos que hay que tener en cuenta en el diseño

A continuación, se comentan una serie de aspectos que hay que tener en cuenta para el éxito del proyecto de sistema empotrado que se tiene que llevar a cabo.

4.1.1. Determinación de los requisitos del sistema

Una de las partes más importantes del desarrollo de un sistema empotrado es la definición de los requisitos técnicos y funcionales para poder cumplir las especificaciones de la aplicación que se quiere construir. Cuanto más cuidadosamente se lleve a cabo esta fase previa a la creación de un prototipo, menos probables serán los cambios no

deseados tanto en el hardware como en el software. Aun así, hay que recordar que el desarrollo del sistema generalmente es un proceso iterativo. Al recorrer los últimos pasos del proyecto, puede surgir la necesidad de revisar las primeras fases para conseguir un producto más fiable, más consistente y que cumpla todos los requisitos especificados previamente.

Habrá que tener en cuenta una serie de consideraciones técnicas, como por ejemplo las siguientes:

- **Definición de las interfaces de control**
  - Los tipos de dispositivos que habrá que controlar o con los cuales se tendrá que comunicar, y también las propiedades eléctricas y mecánicas de sus interfaces.
  - Una definición general de los requisitos de tipos de memoria (volátil, no volátil, estado sólido, magnética, etc.).
  - La definición de cómo el sistema interactuará con las personas (monitor, teclado y lector de códigos de barras, entre otros).

- **Definición de la aplicación de software**
  - Descripción específica de todas las características de la aplicación de software.
  - Características de depuración.
  - Posibilidad de actualización del software.
  - Previsión de controladores (*drivers*) para el funcionamiento del hardware externo.

- **Requisitos de alimentación**
  - Posibilidad de alimentación mediante batería o pilas.
  - Alimentación de seguridad para condiciones de caída de la alimentación principal.
  - Previsiones generales de consumo del sistema.

- **Requisitos térmicos del sistema**
  - Margen de temperatura nominal.
  - Refrigeración por ventilador o por convección.
  - Ubicación de los componentes para una gestión térmica efectiva.
  - Condiciones ambientales externas para determinar las necesidades de control térmico.
  - Tiempo de vida esperado del sistema frente a las necesidades de control térmico.

- **Diseño de la forma y el tamaño**
  - Restricciones en cuanto a dimensiones.
  - Robustez del producto.

- **Necesidades de rendimiento**
  - Necesidades de velocidad de procesamiento.
  - Capacidad de manejar interfaces gráficas.
  - Capacidad de ejecutar software exterior.
  - Capacidad de comunicarse con otras interfaces de alta velocidad.

- **Definición de los subsistemas principales**

– Los subsistemas principales con suficiente complejidad deben tener una definición similar a la del sistema completo para evitar problemas o costes de integración elevados.

4.1.2. Consideraciones en cuanto el proyecto

1) **Previsión del tiempo que el producto vivirá en el mercado**

- Disponibilidad de componentes.
- Soporte y mantenimiento del producto vendido.

2) **Estimación del tiempo de desarrollo**

- Ventana temporal de la oportunidad de mercado.
- Velocidad para sacar los primeros prototipos.
- Disponibilidad de encontrar desarrolladores con experiencia.

3) **Nivel de experiencia de diseño o desarrollo**

- Experiencia de diseño de hardware.
- Familiarización con los entornos de programación.

4) **Actualizaciones del software**

- Número esperado de actualizaciones.
- Previsión del crecimiento del tamaño del programa.
- Posibilidad de realizar actualizaciones remotas del programa.

5) **Efectividad de la plataforma de desarrollo**

- Tiempo estimado para la finalización del primer prototipo.
- Disponibilidad de herramientas.
- Capacidad de utilización o integración de periféricos.
- Facilidad de uso.
- Mantenimiento:
  - Procedimiento de actualización del hardware y del software.
  - Mantenimiento o sustitución de componentes.
  - Mantenimiento o sustitución de subsistemas.
  - Mantenimiento térmico del sistema.
  - Actualizaciones del hardware.
  - Estudio de las necesidades de actualización.
  - Expansión y modularidad del diseño.

4.1.3. Consideraciones en cuanto al diseño del sistema empotrado

Los fabricantes de semiconductores proporcionan los microprocesadores, microcontroladores y DSP en forma de chip bajo una amplia gama de encapsulados, en general

cada vez con dimensiones más reducidas y más adaptados a nuevos procesos industriales de soldadura que permiten reducir costes y automatizar procesos. Normalmente, el hardware se diseña en dos fases: creación de los esquemas eléctricos y creación del formato (*layout*) de la PCB.

Elaboración de los esquemáticos

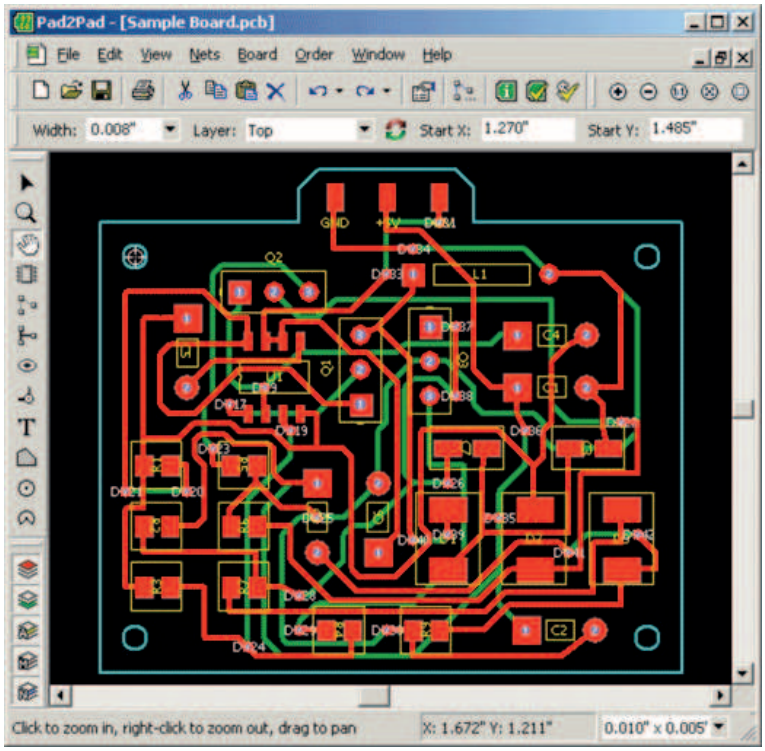
El diseño electrónico del sistema se lleva a cabo mediante unos diagramas llamados **esquemas eléctricos**, que representan los componentes mediante símbolos y sus interconexiones mediante líneas, de manera que sea fácil o digerible de visualizar. Estos diagramas se hacen con herramientas de software tipo CAD*, como por ejemplo PSPICE, Orcad, Altium o VeriBest de Intergraph.

*** Sigla del término inglés *computer aided design*, diseño asistido por ordenador.**

Unos buenos esquemas deben incluir información extra requerida para entender el funcionamiento del sistema. Entre otras cosas, han de figurar nombres representativos en las líneas que representan buses o señales entre dos componentes.

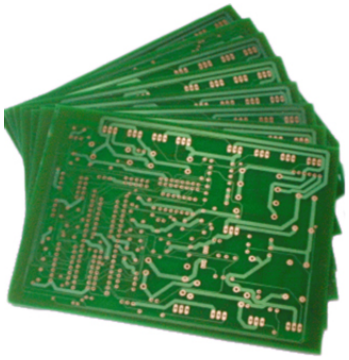
Todos los componentes que se colocarán en la placa del circuito impreso deben tener su representación en los esquemas mediante símbolos, normalmente rectangulares, con sus pivotes alrededor. El diseñador de los esquemas debe consultar la documentación técnica o las especificaciones de los componentes utilizados para que la conexión entre ellos se realice correctamente. Se deben considerar aspectos como la temporización en las comunicaciones entre ellos y las cargas (esto es, que una salida proporcione suficiente corriente eléctrica para las entradas a las que llega). Sobre todo a altas frecuencias de trabajo del sistema (por encima de 50 MHz), hay que controlar la adaptación entre componentes, esto es, que no se produzcan reflexiones de ondas eléctricas en las interconexiones entre componentes, puesto que harán imposible la transferencia de información legible.

Figura 4. Esquemático. Sistemas empotrados en un vehículo



Creación del formato de la PCB

El formato (*layout*) consiste en una representación exacta de las *huellas* (*footprint* en inglés) de los componentes y de las pistas que unen sus pivotes o terminales. Se entiende por *huella de un componente* los puntos de material conductor que necesita en la placa base para poder ser soldado y fijado, mientras que las pistas son las líneas de material conductor de ancho determinado que permitirán la propagación eléctrica de las señales entre los componentes.



PCB.

La placa está formada por una lámina de fibra de vidrio o similar con varias capas de cobre (dos exteriores y posibles internas, como si fuera un sándwich). Mediante los ***gerbers***, el fabricante de PCB puede construir la placa con las huellas y las pistas necesarias. Después de esto se pueden soldar los componentes. El formato también se hace mediante una herramienta CAD, integrada con la de creación de esquemas. Así, los esquemas dirigen la creación del formato estableciendo cuáles deben ser las uniones entre los componentes, ahora en forma de pistas que recorrerán el camino necesario para unir dos puntos.

Gerbers

Los *gerbers* son unos dibujos que representan el formato.

Entre los aspectos que hay que tener en cuenta en la creación de la PCB, se encuentra la posibilidad de elegir encapsulados de diferentes formas o tipos para un mismo componente, y también la compatibilidad electromagnética de la placa (esto es, que las señales de alta frecuencia no interfieran con otras de la misma placa ni con otras del exterior). Mediante la herramienta del encaminador o del *autorouter*, el programa CAD puede dibujar todas las pistas simplemente eligiendo el lugar donde irá cada componente (ubicación o *placement*). Dependiendo de la densidad de pistas, pueden ser necesarias más o menos capas de pistas de la PCB. Aun así, esto solo funciona para diseños sencillos, y el procedimiento de encaminar placas PCB complejas acaba siendo una tarea que requiere mucha experiencia.

Existe una jerarquía de diseño de un sistema empotrado, que podemos expresar según el nivel de integración en los puntos siguientes:

1) **Nivel de chip**. Utilización de sistemas CAD para diseño electrónico para proceder a su diseño y simulación.

2) **Nivel de tarjeta (PCB)**. Sobre un sustrato aislante con unas dimensiones físicas determinadas, se procede a diseñar el formato de la tarjeta. Posteriormente, se sueldan los dispositivos. Las tecnologías de diseño se suelen definir como:

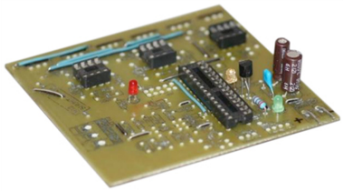
- convencionales,
- montaje superficial (SMT) o
- híbrida de capa fina y capa gruesa.

A su vez, según la manera de interconectar los dispositivos, podemos hablar de:

- simple capa,
- doble capa o
- multicapa.

Montaje de los componentes sobre la PCB

El prototipo es un equipo hardware que tiene partes idénticas a las del producto final pero que no está completo. Incluso, puede ser diferente del producto final pero con fuertes parecidos. Pero tiene que servir para ir probando módulos de software y con ello el hardware, que también ha de ser depurado. Este prototipo se puede construir adquiriendo los componentes necesarios y placa *layout* y componentes de un prototipo fabricando una PCB, o se puede comprar a un tercero una placa de evaluación del microprocesador utilizado (de este modo, se consigue ahorrar tiempo en el diseño). Las placas de evaluación permiten emular el funcionamiento del prototipo final y comprobar el funcionamiento del software diseñado. Sin embargo, en última instancia siempre se deberá probar el software en el prototipo final en una fase de integración.



Placa *layout* y componentes de un prototipo.

Integración del sistema

En esta fase es la primera vez que se prueba el sistema completo (tanto de software como de hardware conjuntamente). En realidad, lo que se prueba es el hardware, utilizando el software desarrollado antes. Normalmente, antes de esta fase, el software es verificado después de las pruebas llevadas a cabo con los kits de desarrollo y emuladores, que ofrecen un comportamiento casi idéntico al que tiene que ofrecer el hardware.

La integración es el momento de la verdad, en el que se verifica el buen funcionamiento del hardware desarrollado. Este es un paso de mucha ansiedad en el proceso de desarrollo, puesto que es cuando se detectan los errores más costosos y que deben ser solucionados rápidamente.

Por desgracia, la integración del sistema suele tener lugar cerca del final del ciclo de vida del proyecto y, por lo tanto, el tiempo disponible para solucionar los errores normalmente es corto. Dependiendo del tipo de sistema empotrado, el coste añadido a la hora de corregir un error aumenta de 2 a 10 veces por cada fase de vida.

Hay numerosas estrategias para reducir el tiempo de integración de los sistemas. Una de las mejores consiste en crear un prototipo de hardware dentro de la fase de diseño del hardware y hacer pruebas de diferentes partes del software que se va desarrollando. Además, el desarrollo del software se acelera, dado que no hay que perder tiempo implementando software que simule el hardware todavía inexistente.

Pruebas, depuración y producto final

La prueba final consiste en la comprobación de que el sistema cumple todos los requisitos esperados. Las pruebas pueden ser alfa, si se hacen en la misma empresa de desarrollo del sistema, o beta, si se hacen con el cliente o usuario en el lugar que él de-

signe. Después de esta fase, solo queda la política de mantenimiento (principalmente software) que se establezca entre ambas partes.

Otras consideraciones sobre el diseño del software

El software que hay que crear para que el sistema cumpla su funcionalidad se suele empezar a desarrollar en paralelo con el diseño del hardware. Una vez acabados el hardware y el software, se podrá efectuar la integración del sistema. Hasta que no llega este momento, el software se desarrolla, depura y prueba con emuladores de software o hardware que proporcionan los fabricantes de los microcontroladores empleados u otras empresas especializadas en estas fases de diseño.

Si se utiliza una arquitectura de tipo PC, se puede simplificar el desarrollo del software, dado que en el mercado hay numerosos módulos ya implementados que se pueden comprar. Donde podría haber una complejidad mayor es en los requisitos de tiempo real que pueda tener el sistema. El software se empieza a desarrollar por una fase inicial de captura de **requisitos de usuario** (lo que el usuario final espera obtener del sistema). Después de esto, se establecen los **requisitos de sistema** (lo que el sistema debe tener para poder cumplir los requisitos de usuario). Hay que destacar, también, la importancia de documentar el software, igual que el hardware. En el código, hay que añadir líneas de comentarios que describan la funcionalidad o la utilidad de las variables y de las funciones, e incluso explicaciones de trozos de código. También se ha de generar un documento con la descripción del software. Este queda prácticamente constituido con los diagramas que se van generando con la metodología UML.

Requisitos de sistema y de usuario

Los requisitos de sistema son más técnicos que los del usuario y en general hay más, puesto que para cubrir un requisito de usuario pueden ser necesarios más de un requisito de sistema.