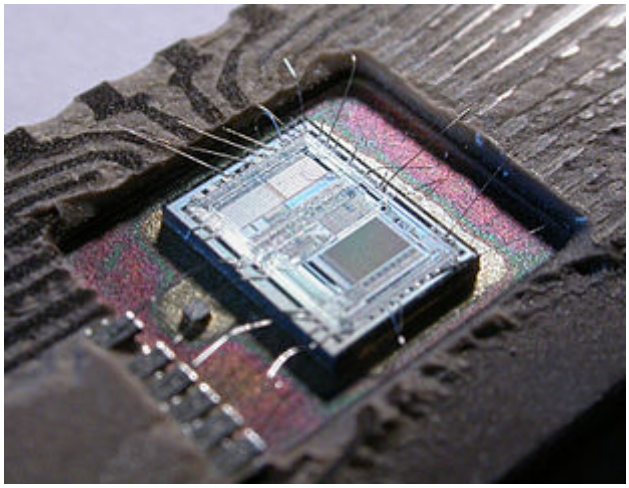
Sistema embedded

Da Wikipedia, l'enciclopedia libera.

Un **sistema *embedded*** (letteralmente *immerso o incorporato*, tradotto in italiano con *sistema integrato*), nell'informatica e nell'elettronica digitale, identifica genericamente tutti quei sistemi elettronici di elaborazione a microprocessore progettati appositamente per un determinato utilizzo (*special purpose*) ovvero non riprogrammabili dall'utente per altri scopi, spesso con una piattaforma hardware *ad hoc*, integrati nel sistema che controllano e sono in grado di gestirne tutte o parte delle funzionalità richieste.



Microcontrollore per sistemi embedded senza plastica di protezione (Intel 8742)

Indice

Storia

Uno tra i primi sistemi *embedded* fu l'Apollo Guidance Computer, sviluppato da Charles Stark Draper al MIT Instrumentation Laboratory. Per ogni volo lunare ne veniva utilizzato uno nell'orbiter CSM (modulo di comando e servizio) ed un altro identico nel LEM, entrambi incaricati del sistema di guida inerziale.

Al momento della concezione, l'Apollo Guidance Computer era considerato uno dei più rischiosi oggetti all'interno del progetto Apollo. L'utilizzo dei nuovi circuiti integrati monolitici per ridurne la taglia e il peso, aumentò considerevolmente il rischio.

Il primo sistema *embedded* prodotto in massa fu il computer che si occupava della guida del missile Minuteman nel 1961. Era il sistema di guida computerizzato *Autonetics D-17*, che utilizzava circuiti logici realizzati con transistor e un hard-disk come memoria principale. Quando il Minuteman II venne prodotto nel 1966, il D-17 fu rimpiazzato da un

nuovo computer basato su circuiti integrati e che fu il primo caso d'impiego di tali componenti in grandi volumi, contribuendo ad abbassarne i prezzi ed a diffonderne di conseguenza l'uso.

La principale caratteristica di questo sistema era la possibilità di modificare successivamente l'algoritmo di guida per rendere il missile più preciso. Il computer inoltre eseguiva la diagnostica del missile consentendo un risparmio sul peso di cavi e connettori.

In seguito, i sistemi *embedded* hanno subito una riduzione dei costi così come una enorme crescita della capacità di calcolo e delle loro funzionalità. La cosiddetta Legge di Moore si è rivelata esatta su tutto questo periodo. Il primo microprocessore progettato per essere messo in commercio è stato l'Intel 4004, che venne montato su calcolatrici ed altri sistemi di piccole dimensioni. Esso richiedeva, comunque, dei chip di memoria esterni ed altra logica di supporto esterna. Verso la fine degli anni settanta, i microprocessori ad 8 bit erano la norma, ma necessitavano sempre di memoria esterna, logica di decodifica e di un'interfaccia con il mondo esterno. In ogni caso, i prezzi erano in caduta libera e sempre più applicazioni cominciarono ad adottare questo approccio, piuttosto che metodologie di progetto di circuiti logici personalizzate.

Verso la metà degli anni ottanta, un maggiore grado di integrazione permise il montaggio di altri componenti, in precedenza collegate esternamente, sullo stesso chip del processore. Questi sistemi integrati vennero chiamati *microcontrollori* piuttosto che microprocessori e fu possibile la loro utilizzazione di massa. Con un così basso costo per componente, questa alternativa divenne molto più interessante che costruire interamente circuiti logici dedicati. Ci fu un'esplosione del numero di sistemi *embedded* distribuiti sul mercato, così come delle componenti fornite da vari produttori per facilitare la progettazione di tali sistemi. Per esempio, vennero prodotti molti circuiti integrati con un'interfaccia seriale (piuttosto che parallela, più tradizionale) che potevano essere interconnessi con microcontrollori con meno connettori. In questo contesto apparve il bus I²C. Verso la fine degli anni ottanta, i sistemi *embedded* rappresentavano la regola piuttosto che l'eccezione per quasi tutti i dispositivi elettronici, tendenza che continua tuttora.

Descrizione

In questa area si collocano sistemi di svariate tipologie e dimensioni, in relazione: al tipo di microprocessore, al sistema operativo, ed alla complessità del software che può variare da poche centinaia di byte a parecchi megabyte di codice. Appartengono a questa categoria di sistemi microelettronici di elaborazione i microcontrollori.

Contrariamente ai computer generici riprogrammabili (*general purpose*), un sistema *embedded* ha dei compiti noti già durante lo sviluppo, che eseguirà dunque grazie ad una combinazione hardware/software specificamente studiata per la tale applicazione. Grazie a ciò l'hardware può essere ridotto ai minimi termini per contenerne lo spazio occupato limitando così anche i consumi, i tempi di elaborazione (maggiore efficienza) ed il costo di fabbricazione. Inoltre l'esecuzione del software è spesso in tempo reale per permettere un controllo deterministico dei tempi di esecuzione.

Un esempio tipico e diffuso di *Sistema embedded* sono le centraline elettroniche installate a bordo degli autoveicoli per il controllo del motore e dell'ABS.

Caratteristiche

I sistemi *embedded* sono *sistemi di calcolo*, nel significato più generale del termine. Questa definizione, infatti, include tutti i computer del mondo, tranne quelli progettati per essere di utilità generica (*general purpose*). Gli esempi di sistemi embedded spaziano dai lettori portatili di musica ai controlli in tempo reale di sistemi fisici quali lo Space Shuttle.

La maggior parte dei sistemi *embedded* è progettata per eseguire ripetutamente un'azione a costo contenuto. La maggior parte di questi sistemi, ma non tutti, deve soddisfare inoltre dei vincoli di prestazioni minime, come ad esempio la necessità di operare in tempo reale. Può anche accadere che un sistema debba essere in grado di eseguire

molto velocemente alcune funzioni, ma possa tollerare velocità inferiori per altre attività. Questi sistemi rispettano i vincoli di prestazione con una combinazione di hardware e software appositamente progettati.

Risulta difficile caratterizzare la velocità o i costi di un sistema *embedded* generico, anche se, soprattutto per sistemi che devono processare una grande quantità di dati, il progetto stesso assorbe la maggior parte dei costi. Per la maggior parte dei sistemi *embedded* le prestazioni richieste possono essere soddisfatte con una combinazione di hardware dedicato e una quantità limitata di software ottimizzato. A titolo di esempio, basti pensare ad un *decoder* per una televisione satellitare. Nonostante un sistema come questo debba processare decine di megabit di dati al secondo, la maggior parte del lavoro è svolta da hardware dedicato che separa, regola e decodifica il flusso digitale multicanale in un'uscita video. Alla CPU *embedded* spetta di determinare i percorsi dei dati nel sistema, o di gestire gli *interrupt*, generare e disegnare la grafica, e così via. Spesso, quindi, gran parte dell'hardware di un sistema *embedded* deve sottostare a requisiti di prestazioni molto meno severi di quelli che, invece, deve rispettare l'hardware primario del sistema stesso. Questo permette all'architettura di un sistema *embedded* di essere semplificata rispetto a quella di un computer generico che deve eseguire le stesse operazioni, usando ad esempio una CPU più economica che tutto sommato si comporta discretamente anche per queste funzioni secondarie.

Per sistemi *embedded* che non devono gestire una grossa mole di dati, possono essere utilizzati anche dei personal computer, riducendo il numero di programmi installati, oppure utilizzando un sistema operativo real-time. In questo caso l'hardware dedicato può essere sostituito con una o più CPU ad alte prestazioni. Ciononostante, alcuni sistemi *embedded* potrebbero richiedere in ogni caso CPU potenti, hardware dedicato ed una grande quantità di memoria per eseguire una certa attività.

Nel caso di sistemi che devono essere commercializzati in massa, come un lettore di musica portatile, ridurre i costi diventa una priorità. Sistemi di questo genere, infatti, spesso sono dotati di alcuni *chip*, una CPU altamente integrata, un *chip* dedicato a tutte le altre funzioni ed un singolo banco di memoria. In questo caso ogni componente è selezionato e progettato per ridurre il più possibile i costi.

Il software scritto per molti sistemi *embedded*, in particolare quelli senza *hard disk*, è talvolta chiamato *firmware*. Il firmware è un tipo di software che, ad esempio, è possibile trovare nei *chip* delle memorie ROM o Flash.

I sistemi *embedded* spesso richiedono di essere attivi continuamente per anni senza errori, pertanto il software ed il *firmware* sono progettati e testati con molta più attenzione rispetto al software dei personal computer. Molti sistemi *embedded*, infatti, evitano di incorporare componenti con parti meccaniche in movimento (come gli *hard disk*), poiché meno affidabili rispetto a componenti allo stato solido come le memorie Flash.

In aggiunta, i sistemi *embedded* possono essere fisicamente inaccessibili (come per le batterie di perforazione dei pozzi di petrolio, oppure i componenti lanciati nello spazio), pertanto i sistemi che li contengono devono essere capaci di resettarsi autonomamente in caso di perdita o corruzione dei dati. Questa funzionalità è molto spesso ottenuta con l'inserimento di un componente elettronico chiamato *watchdog* che ripristina il processore se il programma presente sullo stesso non azzera con una certa frequenza il *timer* interno del componente.

Esempi di sistemi embedded

- Personal computer dedicati all'automazione industriale e il controllo di processo.
- Sistemi elettronici per avionica
- Sportelli Bancomat e apparecchi POS.
- Elettronica aeronautica, come sistemi di guida inerziale, hardware/software di controllo per il volo e altri sistemi integrati nei velivoli e nei missili.
- Telefoni cellulari e, in generale, dispositivi mobili.
- Centralini telefonici.
- Apparecchiature per reti informatiche come router, timeserver e firewall, switch.
- Stampanti e Fotocopiatrici.
- Sistemi di memorizzazione dati come hard disk, floppy disk o compact disc.

- Sistemi di automazione casalinghi come termostati, condizionatori e altri sistemi di controllo della sicurezza.
- Distributori automatici di bevande e cibo, di carburante, di biglietti, ecc.; stazioni automatizzate di riscossione dei pedaggi autostradali o dei parcheggi, ecc. ; in generale distributori o di vari prodotti o servizi.
- Elettrodomestici come forni a microonde, lavatrici, apparecchi televisivi, lettori o scrittori di DVD, impianti Hi-fi o home video ma anche un banale spazzolino elettrico o un rasoio elettrico di alta fascia.
- Apparecchiature biomediche come ecografi, scanner medici per risonanza magnetica.
- Equipaggiamenti medici.
- Strumenti di misura come oscilloscopi digitali, analizzatore logico, e analizzatore di spettro. Ma pure una banale bilancia elettronica da casa, nonché il comune termometro digitale.
- I PLC (*Programmable Logic Controller*) utilizzati per l'automazione industriale.
- Console per videogiochi fisse e portatili.
- Centraline di controllo incorporate nei veicoli per la gestione elettronica di vari impianti e servizi.
- Strumenti musicali digitali quali tastiere workstation, mixer digitali o processori audio.
- Decoder per TV digitale.
- Sistemi per la domotica.
- Attrezzi e utensili (sia casalinghi che professionali): dal trapano elettrico ad una friggitrice, dalla lavatrice al una livella multifunzione laser, ma l'elenco sarebbe assai lungo.

In pratica la diffusione di questi sistemi è capillare nella società e tutti i dispositivi elettronici non general purpose, ovvero i computer riprogrammabili propriamente detti, possono essere definiti sistemi *embedded* evidenziando così la loro notevole importanza. Il fatto che il sistema *embedded* sia incorporato e dunque non visibile agli occhi dell'utilizzatore induce il senso comune a pensare ad una prevalenza dei computer *general purpose* quando in realtà la situazione è diametralmente opposta. Anche perché per la stragrande maggioranza delle persone risulta difficile pensare che questi prodotti/sistemi contengano dei "computer" a tutti gli effetti.

Progettazione

Piattaforma

La piattaforma sulla quale può venir sviluppato un sistema *embedded* varia drasticamente a seconda della sua complessità, consumi (elettrici), costo e campo di utilizzo. Si passa dai PLC e microcontrollori più semplici ad architetture più complesse basate su circuiti integrati sofisticati (System-on-a-chip - vedere di seguito.)

Alcune piattaforme di sviluppo (*reference board* o *reference design*) molto usate sono basate su architetture ARM, MIPS, Coldfire/68k, H8, SH, V850, FR-V, M32R, eccetera. Talvolta sono impiegate le più comuni architetture IBM compatibili, eventualmente adattate, con CPU X86 o PowerPC.

Altre architetture più semplificate sono basate su microcontroller PICmicro, Intel 8051, Atmel AVR, eccetera.

L'evoluzione, che porta a maggiore ingegnerizzazione dei componenti, si sta spostando, specie per sistemi *embedded* ad alto volume di produzione, verso i "System-on-a-chip" (o SoC). I SoC racchiudono, in un singolo circuito integrato di tipo ASIC, il microcontroller/CPU e/o il DSP, memoria, oscillatori e clock, regolatore di tensione, eventuali interfacce AD/DA, e verso l'esterno (USB, ethernet, eccetera).

Data la complessità crescente, non è raro che il produttore dell'hardware fornisca un BSP (o Board Support Package) per semplificare il supporto e integrazione tra il software sviluppato ad hoc, l'ambiente operativo sottostante e l'hardware.

Un altro comune metodo di progetto prevede l'utilizzo di FPGA (*Field-Programmable Gate Array*), con la programmazione di tutta la logica interna, inclusa la CPU. La maggior parte dei FPGA sono progettati proprio per questo scopo. Tipicamente si fa uso di FPGA affiancandoli ad altri circuiti integrati per l'interfacciamento.

Questa situazione è in contrasto con quella del mercato dei *desktop computer*, che al momento è composto solo da poche architetture concorrenti, principalmente l'Intel/AMD x86 ed il PowerPC di Apple/Motorola/IBM, quest'ultimo usato nei computer Apple Macintosh fino al 2005.

Utile menzionare anche lo standard PC/104, per quanto riguardante solo il *form factor* (la taglia della scheda madre e simili) e il bus di comunicazione, è presente in ambito industriale e tipicamente impiega elettronica comune nei sistemi desktop (CPU X86) con riadattamenti per questi usi specifici. Usualmente, quindi, impiegano anche i medesimi sistemi operativi (principalmente DOS, Linux od un sistema operativo real-time, come ad esempio QNX, o Inferno).

Interfacce utente

> **Questa voce o sezione sull'argomento sistemi operativi non cita le fonti necessarie o quelle presenti sono insufficienti**.

Le interfacce utente per i sistemi embedded variano anche di molto tra sistema e sistema e quindi meritano qualche commento aggiuntivo.

I progettisti di interfacce come PARC, Apple, Boeing ed HP tendono a minimizzare il numero di diverse interazioni dell'utente. Ad esempio, i loro sistemi utilizzano due soli bottoni (il minimo assoluto) per controllare un menu di sistema (un bottone verrebbe utilizzato per selezionare la successiva voce di menu, l'altro per attivare quella selezionata).

Un touch screen oppure dei bottoni ai bordi dello schermo possono anche essere utilizzati per minimizzare le interazioni con l'utente.

Un'altra comune pratica è quella di minimizzare e semplificare il tipo di output. A volte si usa anche associare una lampadina allo stato di ogni interfaccia, oppure ad ogni situazione di errore. Un'altra economica variazione è quella di creare due file di lampadine, accoppiate ad una matrice di errori che possono verificarsi (l'utente può allora incollare delle etichette che spieghino l'errore nella lingua che preferisce).

Per esempio, lo standard della Boeing per una interfaccia di test è composta da un bottone e alcune lampadine. Quando si preme il bottone, tutte le luci si accendono. Quando il bottone viene rilasciato, le lampadine associate agli errori verificatisi rimangono accese. Le etichette sono in inglese semplificato.

I progettisti usano molto i colori e le sensazioni ad essi collegate. Il rosso indica che l'utente può farsi male (si pensi al sangue). Giallo indica che qualcosa può andare storto. Verde indica che lo stato è positivo. Questa combinazione ricorda da vicino quella di un semaforo, perché molte persone sono in grado di comprenderla, anche istintivamente.

Molti progettisti fanno sì che l'immagine visualizzata cambi immediatamente dopo l'interazione con l'utente. Se la macchina sta per fare qualcosa, normalmente si avvia nel giro di 7 secondi, oppure fornisce un rapporto sul procedere dell'operazione.

Se è necessario uno schermo, molti progettisti si avvalgono di testo semplice, principalmente perché gli utenti sono abituati a leggere. Una interfaccia grafica è gradevole allo sguardo e permette di fare qualunque cosa, ma tipicamente aggiunge circa un anno di ritardo per la progettazione (ideazione, approvazione, traduzione, ecc.) e necessita di uno o due programmatori aggiuntivi, senza aggiungere alcun reale contenuto al sistema. Inoltre, una interfaccia troppo affollata in realtà tende a confondere gli utenti, perché può utilizzare simboli dall'aspetto poco familiare.

Se in un progetto è necessario riferirsi ad alcune parti della struttura (come in una fotocopiatrice), queste sono spesso etichettate con numeri ben visibili sulla struttura.

Una interfaccia di rete è semplicemente uno schermo remoto e si comporta grosso modo come una qualsiasi altra interfaccia utente.

Una delle interfacce generiche basate su schermo più famose è quella composta da due bottoni ed una linea di testo nella lingua madre dell'utente. È utilizzata ad esempio nei cercapersone, nelle stamparti di fascia media, negli switch di rete ed in genere in tutte le situazioni che richiedono una elevata interazione con l'utente.

Quando c'è del testo, il progettista sceglie una o più lingue: quella predefinita è normalmente una delle più conosciute dal gruppo di utenti a cui il prodotto è destinato.

La maggior parte dei progettisti tende ad utilizzare il set di caratteri nativo della lingua, nonostante ciò possa rivelarsi difficoltoso. Le persone che utilizzano set di caratteri particolari preferiscono, infatti, leggere testi scritti in questa maniera.

Il testo è normalmente tradotto da personale specializzato, anche se nello staff del progetto ci sono persone madrelingua.

Le organizzazioni straniere spesso tendono a dare ad un grosso distributore il compito di aggiornare e correggere le traduzioni nella loro lingua. Ciò spesso previene le critiche provenienti da altre persone madrelingua, che tendono a credere che nessuna organizzazione straniera può conoscere la propria lingua così bene come loro.

Quando possibile si tende a rendere il più chiaro possibile sul display i metodi di funzionamento del macchinario.

Nelle organizzazioni meglio gestite, una persona approva l'interfaccia utente. Spesso si tratta di un cliente, di un distributore oppure di qualcuno direttamente responsabile per la vendita del sistema. I committenti, infatti, tendono a prendere decisioni poco rapidamente, oppure a non prenderle affatto. Ciò causa costosi ritardi che potrebbero essere evitati come sopra riportato.

Strumenti di sviluppo

Come per altri software, i progettisti di sistemi embedded utilizzano compilatori, assembler e debugger per sviluppare i software relativi al sistema. Tuttavia possono usare anche alcuni programmi più specifici:

- Un in-circuit emulator (ICE) è un dispositivo hardware che sostituisce o si interfaccia con il microprocessore, ed offre funzionalità per caricare velocemente e effettuare il debugging di codice di prova all'interno del sistema.
- Per velocizzare e rendere economica la diagnostica e il debugging, specie su sistemi prodotti su larga scala, viene integrata, a livello di SoC, microcontroller o CPU, l'interfaccia JTAG (alias IEEE 1149.1), una norma di interfacciamento semplice ed economico atto a sospendere il normale funzionamento del processo e interrogarne le fasi tramite collegamento a un personal computer.
- Alcuni programmi aggiungono un controllo di ridondanza (checksum) o un Cyclic redundancy check (CRC) al programma, in modo da permettere al sistema *embedded* di controllare la validità del programma.
- Per sistemi che utilizzano Digital Signal Processor (DSP), i progettisti possono usare uno strumento algebrico come MathCad o Mathematica.
- Compilatori e *linker* specifici possono essere utilizzati per migliorare l'ottimizzazione di hardware particolari.
- Un sistema *embedded* può avere un proprio linguaggio specifico o programma di sviluppo, oppure offrire miglioramenti ad un linguaggio esistente.
- Data la complessità crescente, non è raro che il produttore dell'hardware fornisca un BSP (o Board Support Package) per semplificare il supporto e integrazione tra il software sviluppato ad-hoc, l'ambiente operativo sottostante e l'hardware.

I programmi per la generazione del software possono avere provenienza diversa:

- Compagnie produttrici di software specializzate nel mercato dei sistemi *embedded*
- Possono essere tool provenienti dal progetto GNU (si veda anche cross compiler)
- Talvolta possono essere utilizzati strumenti di sviluppo per personal computer se il processore *embedded* è molto simile ad un comune processore per PC

Sistema operativo

La presenza o meno di un completo Sistema operativo su di un sistema *embedded* varia drasticamente a seconda della complessità della sua architettura e campo di utilizzo. Nei casi più semplici i dispositivi *embedded* potrebbero essere sprovvisti di un sistema operativo vero e proprio.

Su microcontrollori semplici, tipicamente opererà ciclicamente un singolo programma di pochi byte, talvolta indicato come programma di "monitor" in quanto dedicato principalmente a monitorare lo stato delle porte di I/O, senza alcuna sovrastruttura (gestione dei processi, eccetera); su ambienti complessi possono trovare applicazione i medesimi sistemi operativi impiegati comunemente per scopi generali (Linux, Windows CE, ecc.) eventualmente personalizzati (per operare in un ambiente con risorse minimali, in gergo *a basso footprint*), oppure più specializzati per gestire eventi in sistema operativo in tempo reale (come Vxworks o QNX) o estremamente specializzati e non disponibili sul mercato (come i sistemi operativi dei cellulari GSM di prima generazione, per esempio, sviluppati tipicamente dal produttore degli apparati stessi).

Un telefono pubblico Internet utilizzante Windows XP.

Auto-verifica interna

Gran parte dei sistemi embedded hanno capacità native di auto-verifica delle proprie funzionalità. Tipicamente in caso di procedure in loop o subentrando in situazioni di deadlock, intervengono dei meccanismi di 'protezione' chiamati watchdog. Ci sono diversi tipi principali di verifiche, divise in base alla funzione o componente controllata:

1. Verifica Calcolatore: CPU, RAM e memoria programmabile. Spesso si effettua una volta all'accensione. Nei sistemi di importanza critica si effettua anche periodicamente o continuamente.
2. Verifica Periferiche: Simula ingressi e misura uscite. Un sorprendente numero di sistemi di comunicazione, analogici o di controllo possono fare queste verifiche, spesso a costo molto basso.
3. Verifica Alimentazione: Solitamente misura ogni linea di potenza, e può controllare anche qual è l'ingresso (batterie o rete). Le alimentazioni sono spesso sfruttate al massimo, con poco margine di scarto.
4. Verifica Comunicazione: Verifica la ricezione di semplici messaggi da parte delle altre unità connesse. Internet, ad esempio, usa il messaggio ICMP "ping".
5. Verifica Collegamenti: Solitamente usano un cavo sistemato a serpentina tra punti rappresentativi dei cavi che devono essere collegati. I sistemi di comunicazione sincroni, come la telefonia, spesso usano "sync test" a questo scopo. Le verifiche dei cavi sono economiche, e estremamente utili quando l'unità ha dei connettori.
6. Verifica Attrezzaggio: Spesso un sistema dev'essere regolato quando viene installato. Questa verifica fornisce indicazioni alla persona che installa il sistema.
7. Verifica Consumi: Misura le risorse che il sistema utilizza, e avvisa quando le quantità sono basse. L'esempio più comune è la lancetta della benzina di un'auto. Gli esempi più complessi possono essere i sistemi di analisi medica automatica che gestiscono inventari di reagenti chimici.
8. Verifica Operativa: Misura varie cose che interessano all'utente lavorando sul sistema. Notare che si effettua mentre il sistema è in funzione. Esempi sono gli strumenti di navigazione aeronautici, il contachilometri di un'auto, le lucine di un disk-drive.
9. Verifica di Sicurezza: eseguita periodicamente secondo un intervallo di sicurezza, assicura che il sistema sia ancora affidabile. La durata dell'intervallo è in genere appena inferiore al tempo minimo entro cui un malfunzionamento può causare danni.

Aggiornamenti

Per quanto riguarda le funzioni di aggiornamento del software (firmware, sistema operativo o driver a seconda) dei sistemi *embedde*d, vi sono attualmente tre possibili scenari:

- 1) il microcontrollore è totalmente "chiuso": il software nasce e muore con il dispositivo e quindi non è upgradabile;

- 2) il microcontrollore aggiornabile ma solo mediante intervento presso la rete dell'OEM ovvero presso un CAT; in questi casi si deve utilizzare un apposito strumento di diagnostica e programmazione del software;
- 3) il microcontrollore è aggiornabile dall'utente esattamente come un normale PC. Esempi classici: il telefono cellulare (via cavo, via bluetooth o via OTA), il firmware della TV (mediante chiavetta USB, inserita nell'apposita porta dell'apparecchio, contenente l'aggiornamento scaricato dal sito del produttore) e altre situazioni analoghe.

È piuttosto semplice prevedere che nel tempo la situazione più frequente sarà la terza, eccetto i casi in cui il sistema *embedded* sovraintenda requisiti di sicurezza.

Voci correlate

- Architettura ARM
- Microcontrollore
- Hardware-in-the-loop
- Software-in-the-loop
- Sistema operativo real-time
- Sistema di sviluppo
- System on a chip

Altri progetti

- Wikimedia Commons (' attr(href) ') contiene immagini o altri file su **sistema embedded (' attr(href) ')**

Collegamenti esterni

- (EN) *Come funzionano i sistemi operativi real time*, su *FreeRTOS.org*. URL consultato il 9 novembre 2018 (archiviato dall'url originale il 4 febbraio 2012).
- (EN) *Portale sui sistemi embedded*, su *Microcontroller.com*.
- (EN) *Centro di sviluppo di sistemi embedded di Microsoft*, su *msdn.microsoft.com*.
- (EN) *Gruppo di discussione sui sistemi embedded*, su *embeddedrelated.com*.
- (EN) *Progettazione di software embedded*, su *eventhelix.com*.
- (EN) *Università che hanno gruppi di ricerca sui sistemi embedded*, su *emlabs.info*. URL consultato il 10 novembre 2005 (archiviato dall'url originale il 26 maggio 2006).
- (EN) *Programmazione di sistemi embedded*, su *embedded.com*.
- (EN) *Compendio di EE - Electronic Engineering e Embedded Systems Programming*, su *ee.cleversoul.com*.
- (EN) *Strumenti e piattaforme per sviluppatori di soluzioni embedded*, su *devicetools.com*.
- *Portale italiano sui sistemi embedded*, su *embedded.it*.
- (EN) *sito web NASA sull'Apollo Guidance Computer*, su *klabs.org*. URL consultato il 15 gennaio 2007 (archiviato dall'url originale il 10 maggio 2012).

Controllo di autorità	LCCN (EN) sh87006632 (' attr(href) ') · GND (DE) 4396978-1 (' attr(href) ') · BNF (FR) cb12410613b (' attr(href) ') (data) (' attr(href) ') · NDL (EN, JA) 01064710 (' attr(href) ')