

# **Appunti di Progettazione di sistemi embedded**

Autori:

**Matteo Iervasi**

**Linda Sacchetto**

**Leonardo Testolin**

# Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
<b>2</b>	<b>Modellazione dei sistemi embedded</b>	<b>5</b>
2.1	SystemC Design Flow . . . . .	5
2.2	SystemC TLM . . . . .	5
2.3	SystemC AMS . . . . .	5
2.4	VHDL . . . . .	5

# Prefazione

Questa dispensa si basa sugli appunti presi da Linda Sacchetto e Leonardo Testolin durante il corso di *Progettazione di Sistemi Embedded* dell'anno accademico 2018/2019. Nonostante sia stata revisionata in corso di scrittura, potrebbe contenere errori di vario tipo. In tal caso potete segnalarli inviando una mail all'indirizzo [matteoiervasi@gmail.com](mailto:matteoiervasi@gmail.com).

Matteo Iervasi

# 1 Introduzione

Citando Wikipedia, un sistema embedded, nell'informatica e nell'elettronica, identifica genericamente tutti quei sistemi elettronici di elaborazione digitale a microprocessore progettati appositamente per una determinata applicazione (*special purpose*), ovvero non riprogrammabili dall'utente per altri scopi, spesso con una piattaforma hardware ad hoc, integrati nel sistema che controllano ed in grado di gestirne tutte o parte delle funzionalità richieste.

Storicamente, sono nati prima i sistemi embedded rispetto ai sistemi *general purpose*, basti pensare ai grandi calcolatori degli anni '40. Essi infatti erano costruiti per un utilizzo specifico, anche se in quanto a dimensioni non erano di certo ristretti. Tuttavia il primo vero sistema embedded, in tutti i sensi, fu l'*Apollo Guidance Computer*, che doveva contenere una notevole potenza computazionale per il tempo in spazi ristrettissimi. La produzione di massa di sistemi embedded cominciò con l'*Autonetics D-17* nel 1961 e continua fino ai nostri giorni.

Non possiamo progettare i sistemi embedded come facciamo con i sistemi *general purpose*, perché abbiamo dei vincoli di progettazione e degli obiettivi differenti. Se ad esempio nei sistemi *general purpose* la ricerca si focalizza nel costruire processori sempre più veloci, nei sistemi embedded la CPU esiste solamente come un modo per implementare algoritmi di controllo che comunica con sensori ed attuatori, e diventa invece più interessante trovare processori che usano sempre meno energia.

I vincoli principali ai quali bisogna attenersi durante la progettazione di un sistema embedded sono:

- **Dimensione e peso**  
Si pensi ai dispositivi che devono poter essere tenuti in una mano
- **Energia**  
Molto spesso i dispositivi embedded devono funzionare con una batteria
- **Ambiente esterno ostile**  
Bisogna dover tenere conto di eventuali fluttuazioni di energia, interferenze radio, calore, acqua, ecc.
- **Sicurezza e operazioni *real time***  
Vi sono casi in cui è necessario che il sistema debba garantire sempre il funzionamento, oppure che garantisca un tempo costante per ogni operazione
- **Costi contenuti**  
Oltre a tutto il resto, bisogna tenere i costi bassi altrimenti si rischia di non poter vendere il prodotto

## 2 Modellazione dei sistemi embedded

Nel momento in cui ci accingiamo a pensare a come si progetta un sistema embedded, salta subito alla mente un problema, ovvero come faccio a verificarne il corretto funzionamento? Quando progettiamo del software, abbiamo a disposizione una miriade di strumenti per assicurarci di tenere il numero dei bug il più basso possibile: debugger, unit testing, analisi statica, ecc. Sull'hardware invece non possiamo certamente metterci a rifare ogni volta che sbagliamo, ricordiamoci che dobbiamo tenere i costi bassi! Si pone quindi il problema della *simulazione*, strumento fondamentale per la verifica del nostro sistema. Spesso infatti l'architettura di riferimento è differente da quella del calcolatore che usiamo per lo sviluppo (caso tipico: noi sviluppiamo su architettura X86 per un'architettura di destinazione ARM).

In generale, un sistema embedded è costituito dalle seguenti componenti:

- Piattaforma hardware  
Oltre al microprocessore, vi sono una serie di altre componenti
- Componenti software  
Il software è monolitico, quando infatti devo accendere il sistema, il software deve eseguire in automatico. Possono esserci casi in cui viene caricato un intero sistema operativo e nella maggioranza dei casi in cui serve un OS si fa riferimento a Linux.
- Componenti analogiche (es. sensori e trasduttori)  
Naturalmente se il nostro sistema dovrà interagire con l'ambiente esterno dovremo introdurre componenti analogiche.

Il trend attuale è di portare tutto ciò in un singolo chip (SoC - *System on a Chip*), dove microprocessore, memoria e altre componenti vengono montate su un singolo chip, collegate da un bus. Esempi di SoC moderni sono i Qualcomm® Snapdragon o i Samsung® Exynos.

### 2.1 SystemC Design Flow

### 2.2 SystemC TLM

### 2.3 SystemC AMS

### 2.4 VHDL