SAPIENZA
UNIVERSITÀ DI ROMA

Implementazione di Sistemi Real Time su Embedded System e PLC

Automazione

Vincenzo Suraci

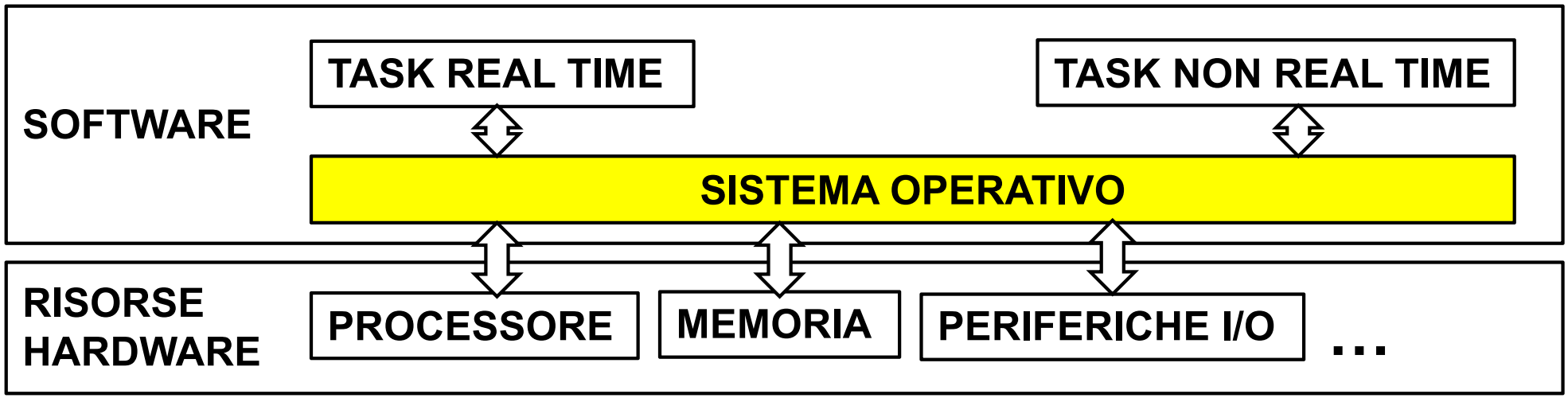
HARDWARE ABSTRACTION LAYER (HAL)

SISTEMA OPERATIVO

Un qualsiasi **sistema di controllo real time** è oggigiorno **implementato** via **software**.

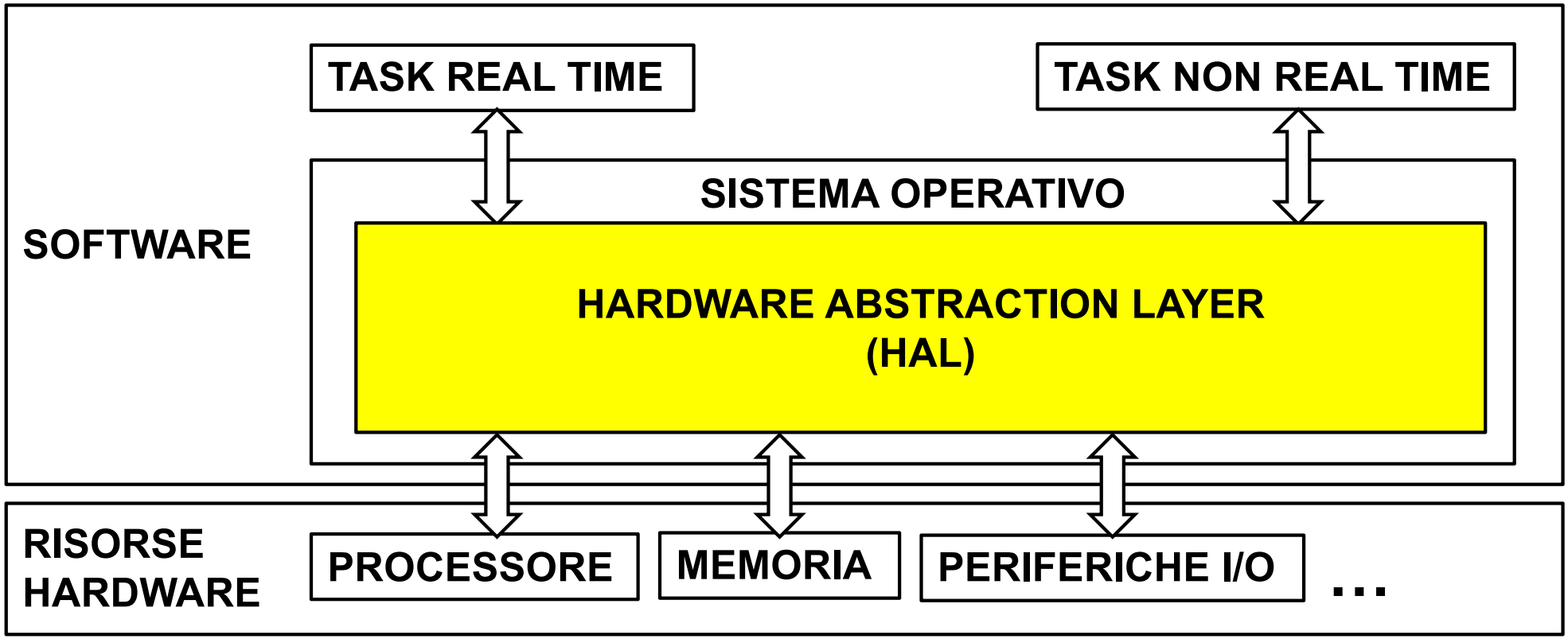
A fare da collante tra il livello **software** che implementa la logica del sistema di controllo e le **risorse hardware** controllate, vi è il **sistema operativo**.

Il **sistema operativo** è **l'interfaccia software** che permette di gestire i **task real time e non real time** che devono essere eseguiti dal **processore**.



HARDWARE ABSTRACTION LAYER

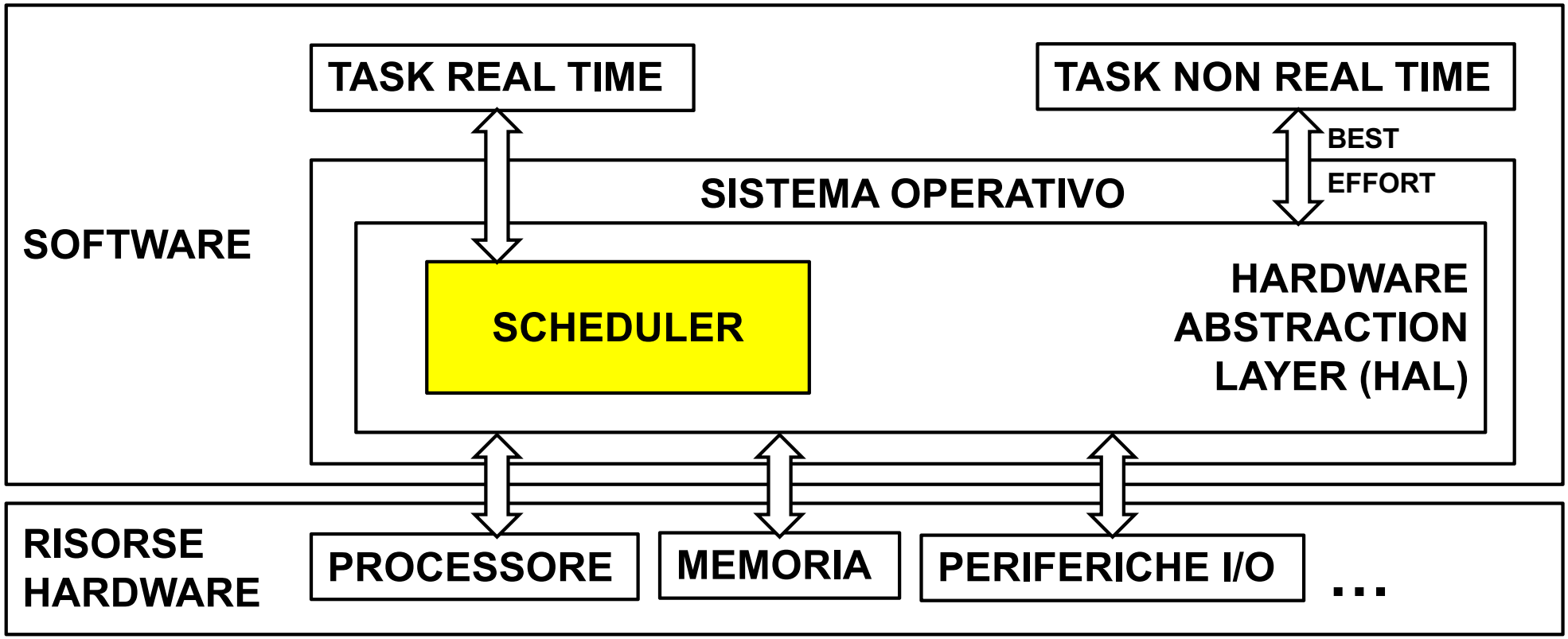
Per disaccoppiare il **sistema operativo** dalle infinite **possibili combinazioni di risorse hardware** viene introdotto un componente chiamato **HARDWARE ABSTRACTION LAYER**.



HARDWARE ABSTRACTION LAYER

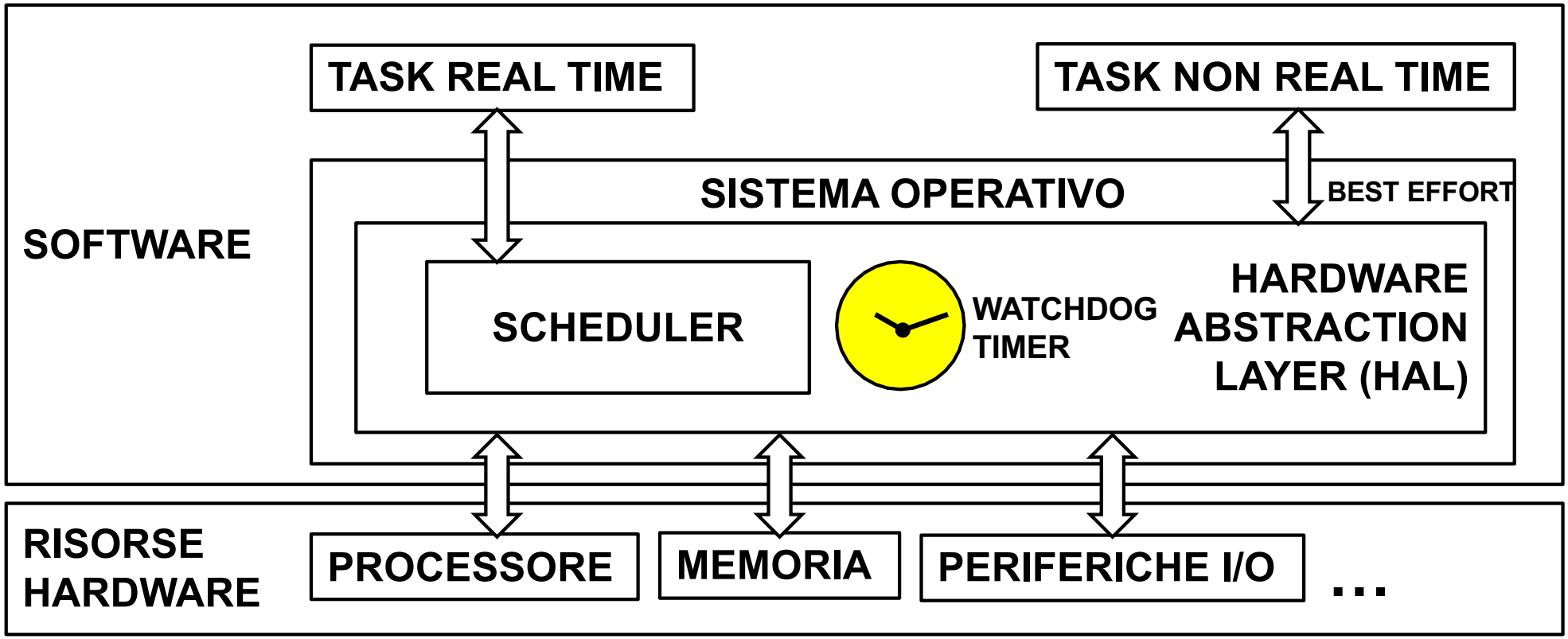
I task NON REAL TIME vengono gestiti dal HAL con politica **BEST EFFORT**.

I task REAL TIME vengono gestiti attraverso lo **SCHEDULER** che ospita uno degli algoritmi di scheduling studiati.



HARDWARE ABSTRACTION LAYER

Il **sistema operativo** controlla **periodicamente** che le **deadline dei task real time** vengano rispettate attraverso un **WATCHDOG TIMER**. Allo **scadere del timer** se una **deadline** è **scaduta**, l'anomalia è segnalata e una **routine di emergenza** è eseguita.

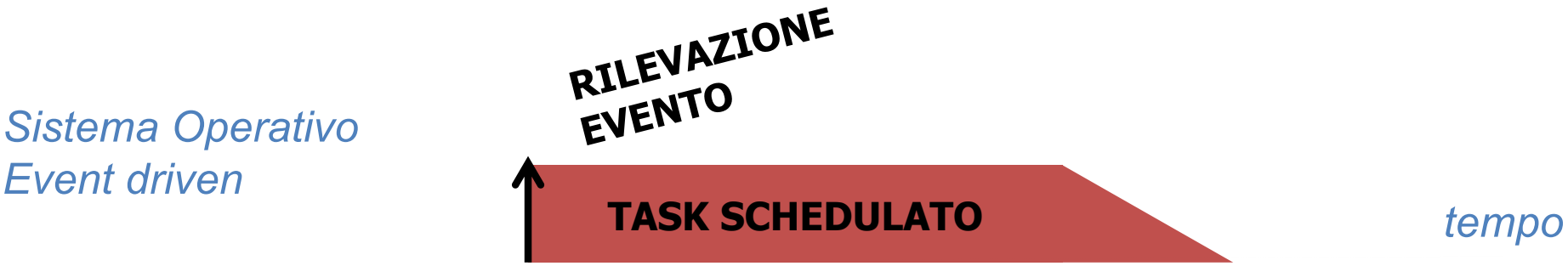


IMPLEMENTAZIONE EVENT-DRIVEN

EVENT DRIVEN

DEFINIZIONE

Un **sistema operativo** si dice **EVENT DRIVEN** se esso è in grado di **schedulare un task** (scartandolo, mettendolo in coda o mandandolo subito in esecuzione) **nello stesso istante** in cui si verifica l'evento che lo ha **attivato**.



EVENT DRIVEN

VANTAGGI

- INTUITIVO: ogni task viene mandato allo scheduler non appena l'evento che lo attiva occorre;
- SEMPLICE DA USARE: ad ogni task può essere **associata una priorità** e l'algoritmo di scheduling pensa a soddisfare anche vincoli hard real time.

SVANTAGGI

- COMPLESSO DA REALIZZARE: **definire un algoritmo di scheduling** generale che risolva il problema della programmazione concorrente **è difficile ed oneroso** se non si conoscono a priori le **caratteristiche dei task in ingresso**.

IMPLEMENTAZIONE TIME-DRIVEN

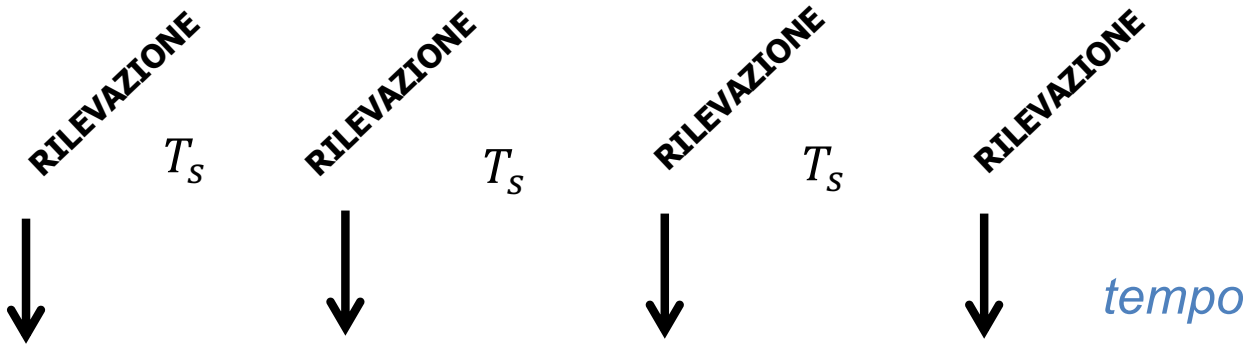
TIME DRIVEN

Un approccio **puramente event-driven** è in realtà **irrealizzabile** se l'unità di elaborazione è di tipo **digitale** e quindi intrinsecamente **quantizzata nel tempo**.

Un approccio realizzabile consiste nel **rilevare periodicamente** l'occorrenza di eventi e di gestire di conseguenza i relativi task. Tale approccio è detto **TIME DRIVEN**.

Il **periodo di tempo** che intercorre **tra due rilevazioni** consecutive è detto **PERIODO DI RILEVAZIONE** (o **PERIODO DI SCANSIONE**) T_s

Sistema Operativo
Time driven



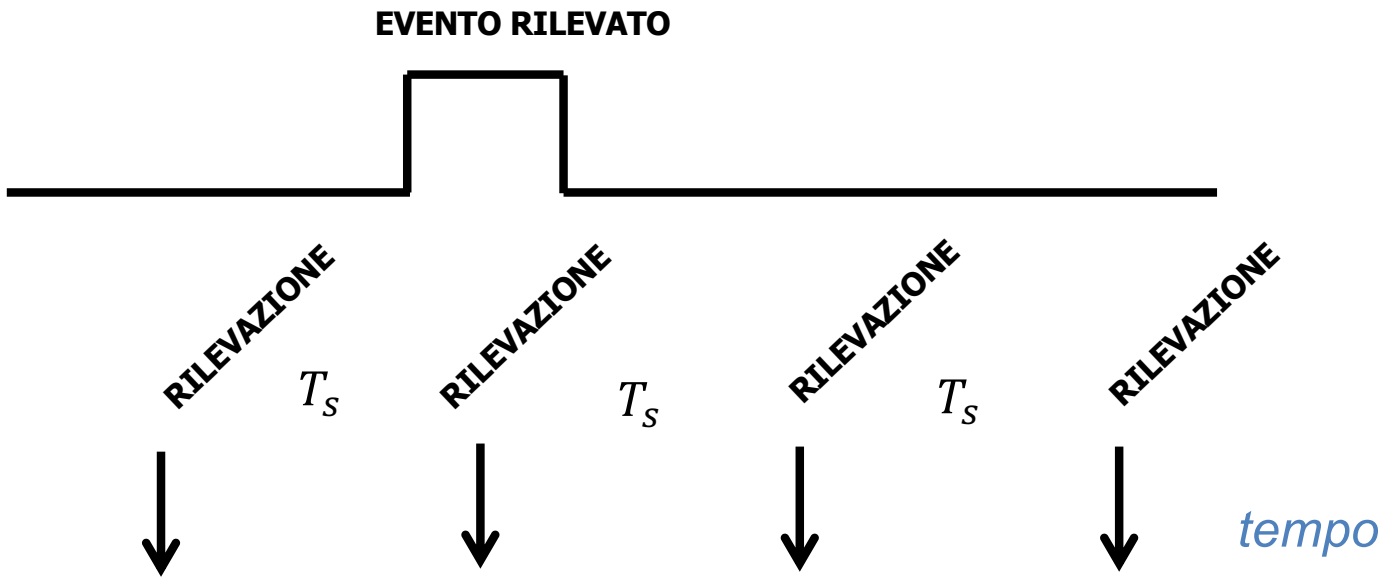
TIME DRIVEN

Un sistema di controllo TIME DRIVEN deve gestire necessariamente le problematiche relative alla **GESTIONE SINCRONA** di **EVENTI ASINCRONI** (che attivano i task).

In un sistema di controllo **DIGITALE**, un **EVENTO** può essere associato al **valore logico** di un **variabile binaria** (SEGNALE LOGICO).

Segnale logico (evento)

Sistema Operativo Time driven



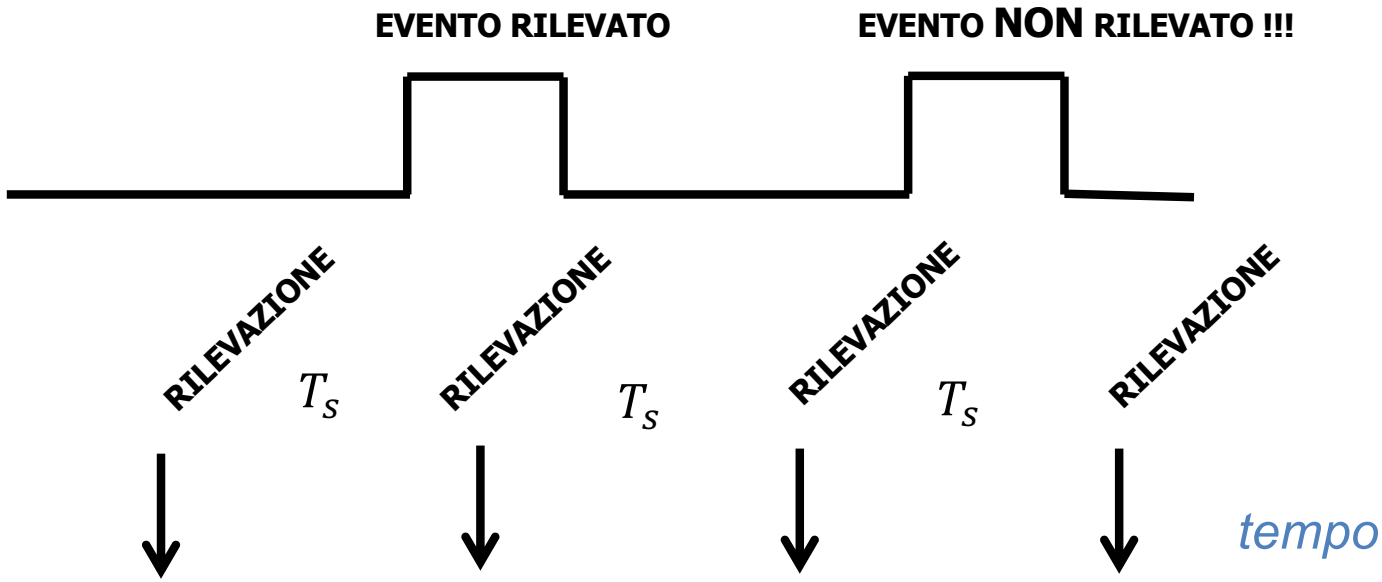
TIME DRIVEN

PROBLEMA 1 – OSSERVABILITÀ DEGLI EVENTI

In un sistema di controllo **time driven** un **evento** può **NON ESSERE OSSERVABILE**. In particolare ciò può avvenire solo se il segnale logico associato all'evento rimane attivo per un **tempo inferiore al periodo di rilevazione**.

Segnale logico (evento)

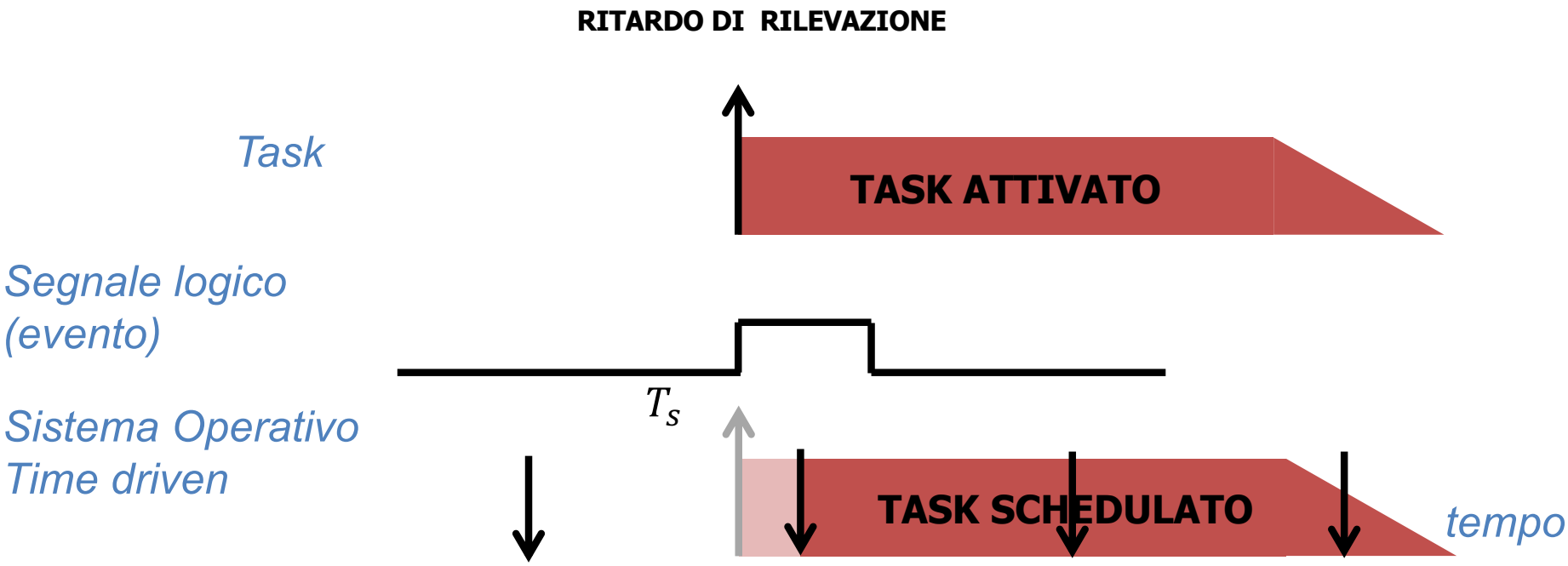
Sistema Operativo Time driven



TIME DRIVEN

PROBLEMA 2 – RITARDO DI RILEVAZIONE

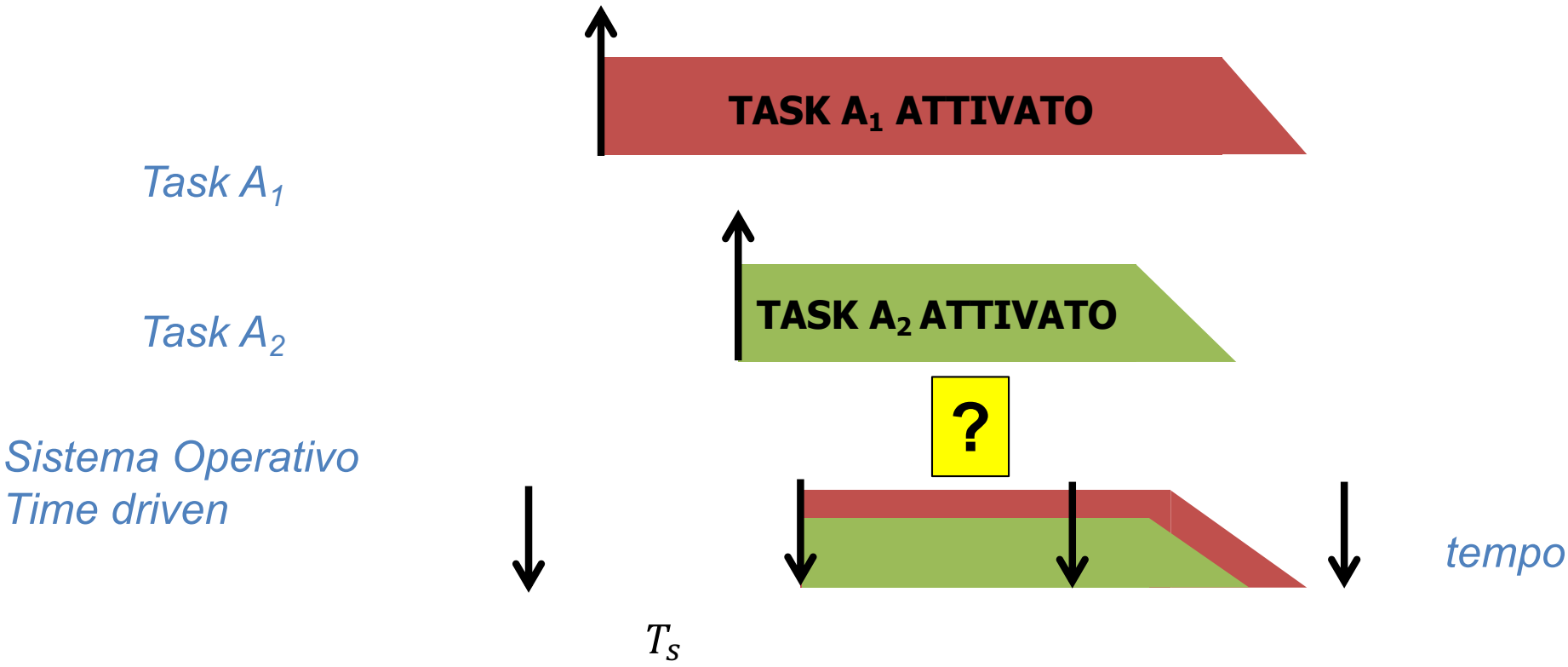
In un sistema di controllo **time driven** ogni occorrenza di un task è soggetta ad un **ritardo di rilevazione** che impatta necessariamente sullo **start time** del task.



TIME DRIVEN

PROBLEMA 3 – ORDINE DI OCCORRENZA

In un sistema di controllo **time driven**, l'**ordine** in cui si presentano due o più eventi occorrenti tra due rilevazioni successive **viene perso**.



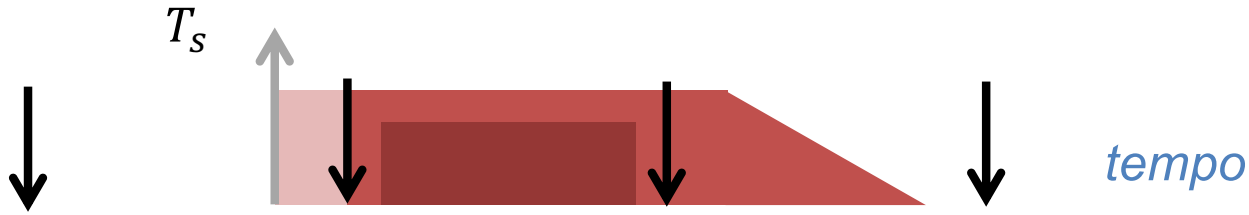
TIME DRIVEN

VANTAGGI

- SEMPLICE DA REALIZZARE: **è sufficiente abilitare un timer** per rilevare **periodicamente** l'occorrenza di **eventi**;

- REATTIVITÀ: ipotizzando che l'elaborazione di qualsiasi task si concluda entro l'intervallo di tempo tra due istanti di rilevazione successivi, è possibile determinare il **limite superiore del response time** del sistema di controllo, pari a **due volte il periodo di rilevazione**.

response time



SVANTAGGI

- FLESSIBILITÀ: le problematiche evidenziate (osservabilità degli eventi, ritardo di rilevazione, ordine di occorrenza) riducono il campo di azione.

SAPIENZA
UNIVERSITÀ DI ROMA

Corso di Laurea: INGEGNERIA
Insegnamento: AUTOMAZIONE
Docente: DR. VINCENZO SURACI

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SISTEMI DI AUTOMAZIONE REAL TIME

OBIETTIVO DEI SISTEMI DI AUTOMAZIONE REAL TIME

OBIETTIVO dei SISTEMI DI AUTOMAZIONE REAL TIME in funzione della PIRAMIDE DELLA AUTOMAZIONE

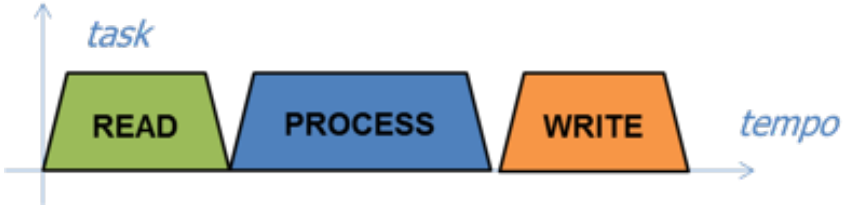
CONTROLLO DI SEQUENZE LOGICHE PER LA SUPERVISIONE

PERIODICI & APERIODICI

CONTROLLO DI SEQUENZE LOGICHE DI COORDINAMENTO

PERIODICI & APERIODICI

CONTROLLO DIGITALE DI VARIABILI ANALOGICHE



PERIODICI

REALIZZAZIONE DEI SISTEMI DI AUTOMAZIONE REAL TIME

REALIZZAZIONE dei SISTEMI DI AUTOMAZIONE REAL TIME in funzione della PIRAMIDE DELLA AUTOMAZIONE

EVENT DRIVEN

TIME DRIVEN

REALIZZAZIONE DEI SISTEMI DI AUTOMAZIONE REAL TIME

OSSERVAZIONE

A livello di **coordinamento** e **conduzione** sono presenti **task misti**, pertanto è intuitivo pensare di usare sistemi di controllo real time **event driven**.

Ma **da un punto di vista implementativo** è molto **più conveniente** realizzare sistemi di controllo finalizzati all'Automazione **completamente time driven**.

REALIZZAZIONE DEI SISTEMI DI AUTOMAZIONE REAL TIME

REALIZZAZIONE dei SISTEMI DI AUTOMAZIONE REAL TIME in funzione della PIRAMIDE DELLA AUTOMAZIONE

EVENT DRIVEN

TIME DRIVEN

DA EVENT-DRIVEN A TIME-DRIVEN

Si può ricondurre il problema del controllo di un sistema di Automazione al problema di gestione di task misti di livello di campo, coordinamento e/o conduzione.

Ipotizziamo di avere m task aperiodici soft real time e n task periodici (equivalenti) real time di periodo T_i $(i = 1,2, \dots, n)$, il sistema di controllo TIME-DRIVEN può essere implementato attraverso l'uso combinato di algoritmi di scheduling di task periodici (ad es., RMPO, EDF, TS) e strategie di scheduling BEST EFFORT di task aperiodici soft real time (ad es., SERVIZIO IN BACKGROUND o PROCESSI SERVER).

OSSERVAZIONE

Come scegliere il periodo di scansione T_s?

DA EVENT-DRIVEN A TIME-DRIVEN

IPOTESI 1

Il problema del **ritardo di rilevazione** può essere **mitigato diminuendo** opportunamente il **periodo di rilevazione** T_s. In particolare si pone:

$$T_s \leq \min_i(T_i)$$

CONSIDERAZIONE

Si potrebbe prendere T_s piccolo a piacere, ma questo aumenterebbe inutilmente la frequenza con cui il sistema di controllo verifica il cambio di stato.

DA EVENT-DRIVEN A TIME-DRIVEN

IPOTESI 2

Ipotizziamo che **ogni task** abbia una **deadline relativa pari almeno al doppio del periodo di rilevazione**:

$$D_i \geq 2T_s \quad \forall\, i = 1,2,\dots,n$$

IPOTESI 3

Ipotizziamo che la **somma dei tempi di calcolo degli n task sia inferiore al periodo di rilevazione**:

$$\sum_{i=1}^{n} C_i < T_s$$

DA EVENT-DRIVEN A TIME-DRIVEN

PROPOSIZIONE (senza dimostrazione)

Dato un sistema di Automazione composto da un sistema di controllo TIME DRIVEN con periodo di rilevazione T_s, e da n task periodici di periodo T_i $(i = 1,2, \dots, n)$ e computation time C_i $(i = 1,2, \dots, n)$ che rispettino le seguenti condizioni:

$$\begin{cases} T_s \leq \min_i(T_i) \\ T_s \leq \frac{1}{2} D_i \quad \forall\, i = 1,2, \dots, n \\ T_s > \sum_{i=1}^{n} C_i \end{cases}$$

La schedulazione dei task può **SEMPRE** avvenire usando un algoritmo **TIMELINE SCHEDULING** scegliendo come **MINOR CYCLE il PERIODO DI RILEVAZIONE**.

SISTEMI OPERATIVI REAL TIME

Consumer Electronic OSes

I sistemi operativi più diffusi (Windows, Linux, Mac OS) **non sono adatti** per gestire sistemi di controllo **real time**.

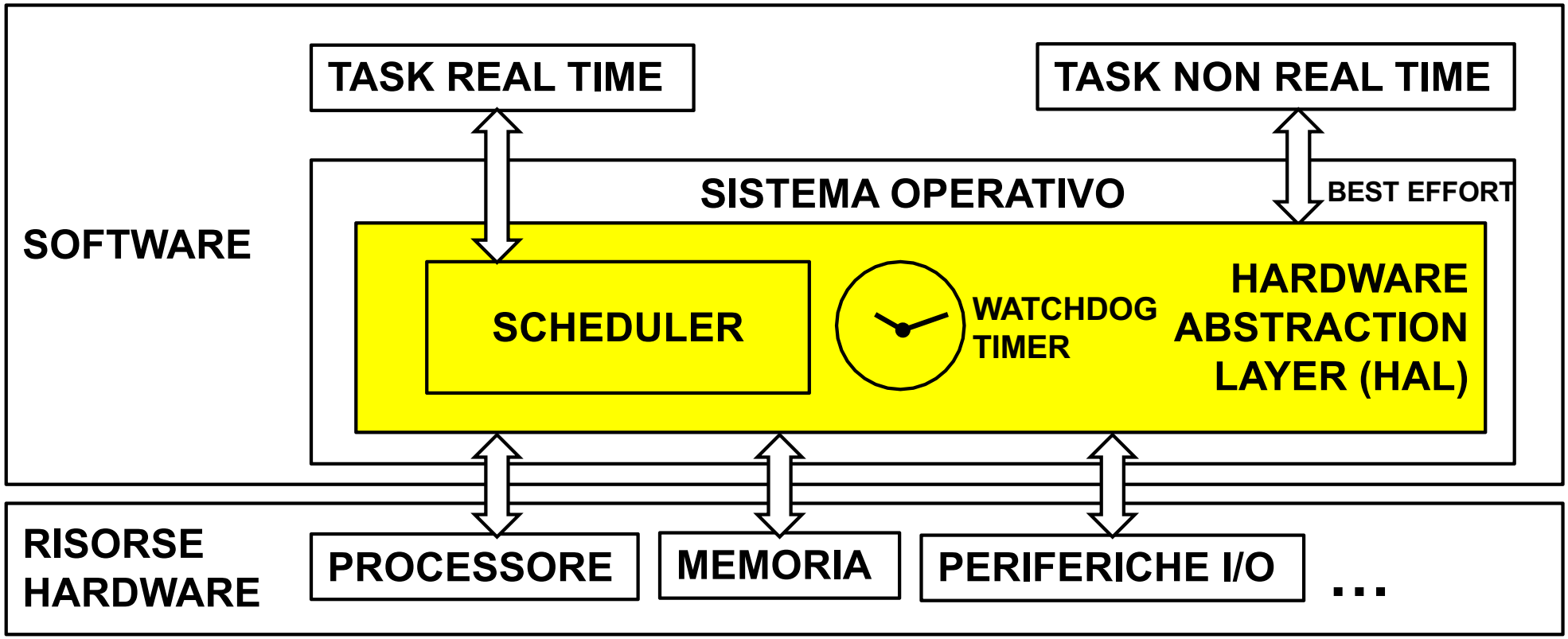
Il problema principale risiede nella **impossibilità di determinare il massimo tempo di esecuzione di un task** (processo o thread che sia).

Un computer **non industriale** è equipaggiato con **risorse** che **bloccano la CPU** e rendono difficilissimo gestire il **determinismo** dello scheduler implementato nel Kernel:

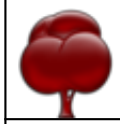
- Perifiche di I/O (Universal Serial Bus - USB);
- DMA (Direct Memory Access) dell'Hard Disk;
- CACHE della CPU (quando si svuota la CPU non può lavorare);
- Memoria Virtuale (paging);
- IRQ (Interrupt Request) da parte di periferiche PCI;
- ACPI (Advanced Configuration and Power Interface) cambia la frequenza della CPU.

Sistemi Operativi Real Time

Per rendere un Sistema Operativo Real Time **è necessario mettere mano al codice** del suo **scheduler** e del suo **HAL** (Hardware Abstraction Layer). Ma questo è possibile solo se il Kernel del sistema operativo è **Open Source**.



Esempi di Sistemi Operativi Real Time – NON COMMERCIALI

	Nome	Caratteristiche
	ChibiOS/RT	Open Source ; Sistemi Embedded
	TinyOS	Open Source; Wireless Sensor Nodes
	RTAI Linux	Open Source (italiano); Computer Industriali; Linux based
	BeRTOS NON SOLO KERNEL	Open Source (italiano); Sistemi Embedded; Arduino
	freeRTOS	Open Source; Cross platform
	Ethernut	Open Source; Sistemi embedded dedicati
	milos	Open Source; Sistemi embedded

SAPIENZA
UNIVERSITÀ DI ROMA

Corso di Laurea: INGEGNERIA
Insegnamento: AUTOMAZIONE
Docente: DR. VINCENZO SURACI

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

CONTROLLO LOGICO SEQUENZIALE

CONTROLLO LOGICO-SEQUENZIALE

Abbiamo visto che un **sistema di controllo real time** deve agire a livello di campo, di coordinamento e di conduzione per eseguire tutti i **task necessari al corretto funzionamento** del sistema complesso.

CONTROLLO DI SEQUENZE LOGICHE PER LA SUPERVISIONE

CONTROLLO DI SEQUENZE LOGICHE DI COORDINAMENTO

CONTROLLO DIGITALE DI VARIABILI ANALOGICHE

SISTEMA DI CONTROLLO REAL TIME

CONTROLLO LOGICO-SEQUENZIALE

In **Automazione** i sistemi di controllo **REAL-TIME CENTRALIZZATI** di **PICCOLA TAGLIA** possono essere realizzati tramite **EMBEDDED CONTROLLER** che eseguono opportuni programmi software sviluppati per il controllo logico sequenziale.

CONTROLLO DI SEQUENZE LOGICHE PER LA SUPERVISIONE

CONTROLLO DI SEQUENZE LOGICHE DI COORDINAMENTO

CONTROLLO DIGITALE DI VARIABILI ANALOGICHE

EMBEDDED CONTROLLER

PROGRAMMA SOFTWARE

CONTROLLO LOGICO-SEQUENZIALE

In Automazione i sistemi di controllo **REAL-TIME CENTRALIZZATI** possono essere realizzati tramite **PLC** o **Soft PLC** che eseguono programmi software per il controllo logico sequenziale.

CONTROLLO DI SEQUENZE LOGICHE PER LA SUPERVISIONE

CONTROLLO DI SEQUENZE LOGICHE DI COORDINAMENTO

CONTROLLO DIGITALE DI VARIABILI ANALOGICHE

PROGRAMMABLE LOGIC CONTROLLER

PROGRAMMA SOFTWARE

CONTROLLO LOGICO-SEQUENZIALE

In Automazione i sistemi di controllo **REAL-TIME DISTRIBUITI** operanti a livello di conduzione sono realizzati tramite **SCADA** (Supervisory Control and Data Acquisition).

CONTROLLO DI SEQUENZE LOGICHE PER LA SUPERVISIONE

SCADA

CONTROLLO DI SEQUENZE LOGICHE DI COORDINAMENTO

CONTROLLO DIGITALE DI VARIABILI ANALOGICHE

CONTROLLO LOGICO-SEQUENZIALE

In Automazione i sistemi di controllo **REAL-TIME DISTRIBUITI** operanti a livello di campo, coordinamento e conduzione, possono anche essere realizzati tramite **DCS** (Distributed Control System).

CONTROLLO DI SEQUENZE LOGICHE PER LA SUPERVISIONE

CONTROLLO DI SEQUENZE LOGICHE DI COORDINAMENTO

CONTROLLO DIGITALE DI VARIABILI ANALOGICHE

SISTEMI DI CONTROLLO DISTRIBUITO

REALIZZAZIONE SU EMBEDDED CONTROLLER

Embedded Controller

DEFINIZIONE

Un qualsiasi sistema realizzato tramite **una singola scheda elettronica** viene chiamato **EMBEDDED SYSTEM**.

DEFINIZIONE

Un **sistema di controllo** realizzato tramite un **embedded system** viene chiamato **EMBEDDED CONTROLLER**.

OSSERVAZIONE

Un **embedded controller** contiene al suo interno **tutto il necessario** sia per **connettere** il controllore al sistema da controllare, sia per **eseguire** gli algoritmi di controllo definiti dall'utente.

Un **embedded controller** viene progettato o scelto in maniera tale che la configurazione **hardware** e **software** sia **ad-hoc** rispetto al problema di **Automazione** da risolvere. Ciò richiede la **conoscenza a priori dei compiti da eseguire**.

Embedded Controller

VANTAGGI

Noto a priori il problema di automazione, l'embedded controller permette di **ridurre** l'**hardware**, lo **spazio**, i **consumi energetici**, i **tempi di realizzazione** ed il **costo** necessari a realizzare il sistema di controllo.

SVANTAGGI

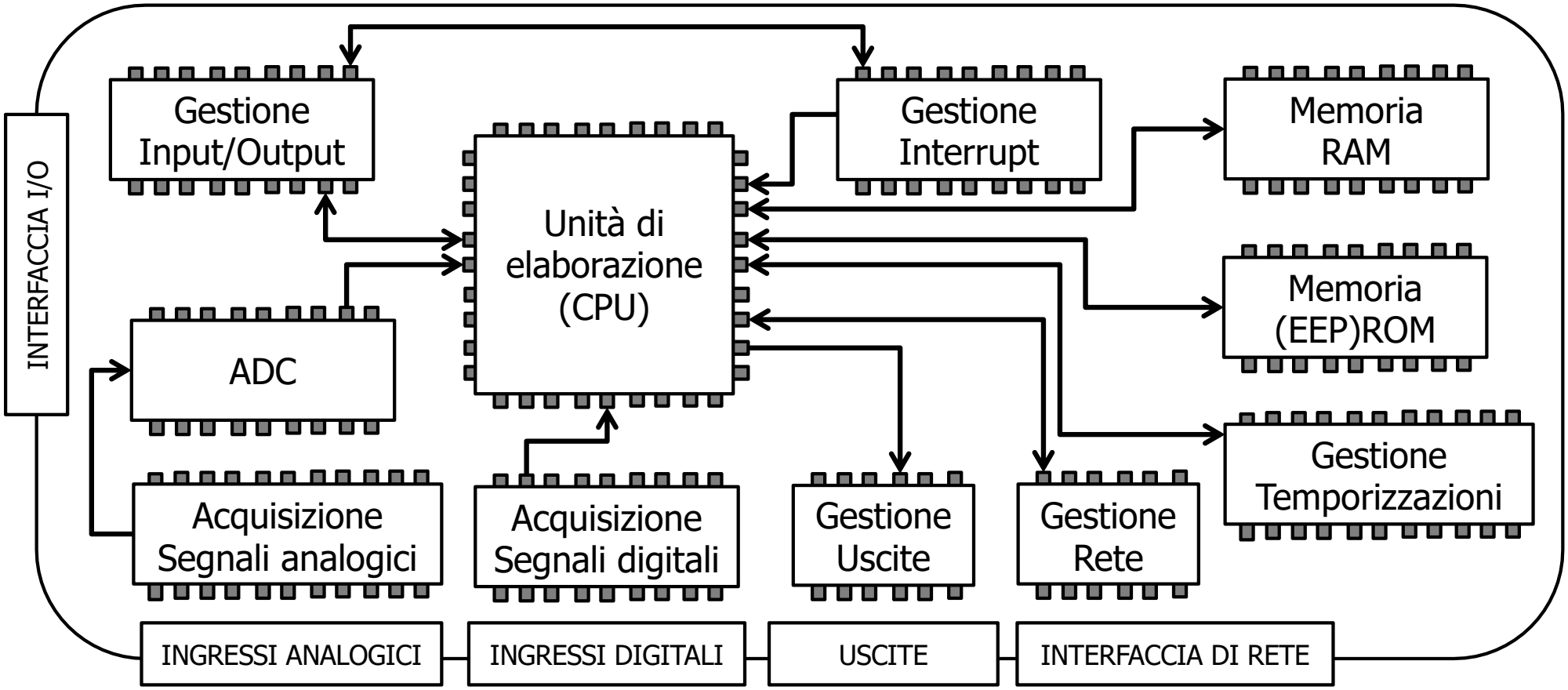
Di contro, l'embedded controller è caratterizzato da **scarsa flessibilità**, **bassissima estendibilità** e **difficile intercambiabilità**.

OSSERVAZIONE

L'uso degli embedded controller nell'ambito della **Automazione** si è diffuso moltissimo soprattutto a **livello di campo**, dove i **task** del sistema di controllo sono ripetuti e **noti a priori**.

Architettura Hardware degli Embedded Controller

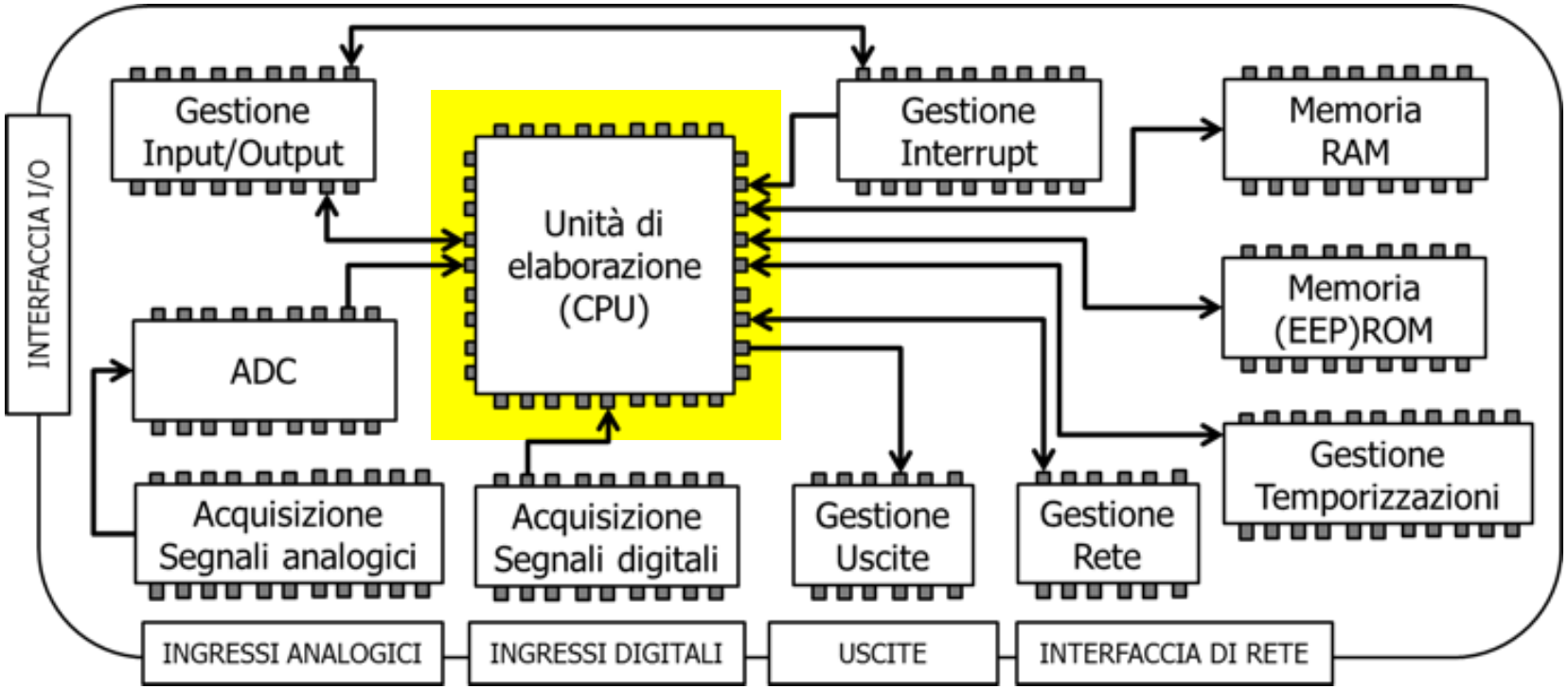
Un embedded controller è realizzato su un'unica scheda, come in figura:



Architettura Hardware degli Embedded Controller

CENTRAL PROCESSING UNIT (CPU)

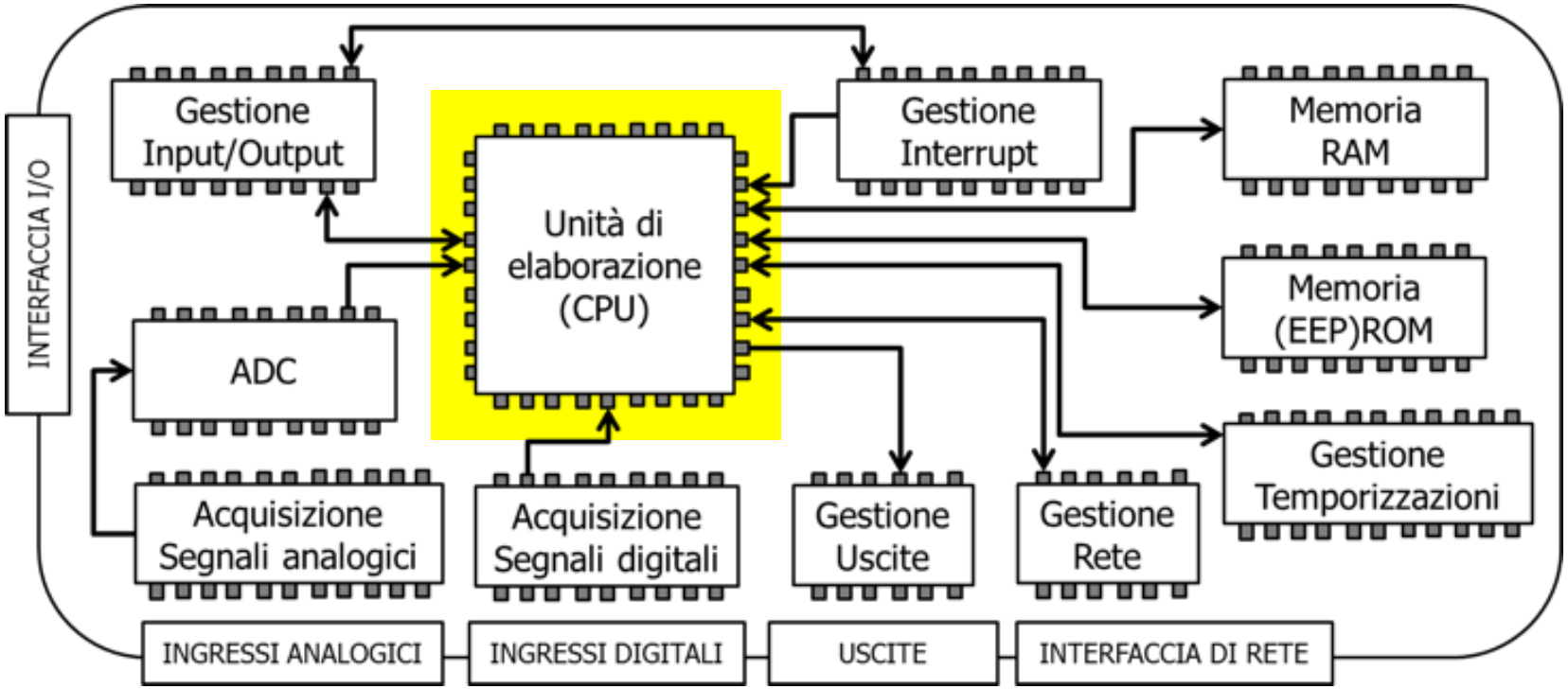
È il **processore** che si occupa di **eseguire** sia il **Basic I/O System** (**BIOS**), ovvero il **sistema operativo**, che il **programma realizzato dall'utente** che implementa **l'algoritmo di controllo**.



Architettura Hardware degli Embedded Controller

CENTRAL PROCESSING UNIT (CPU)

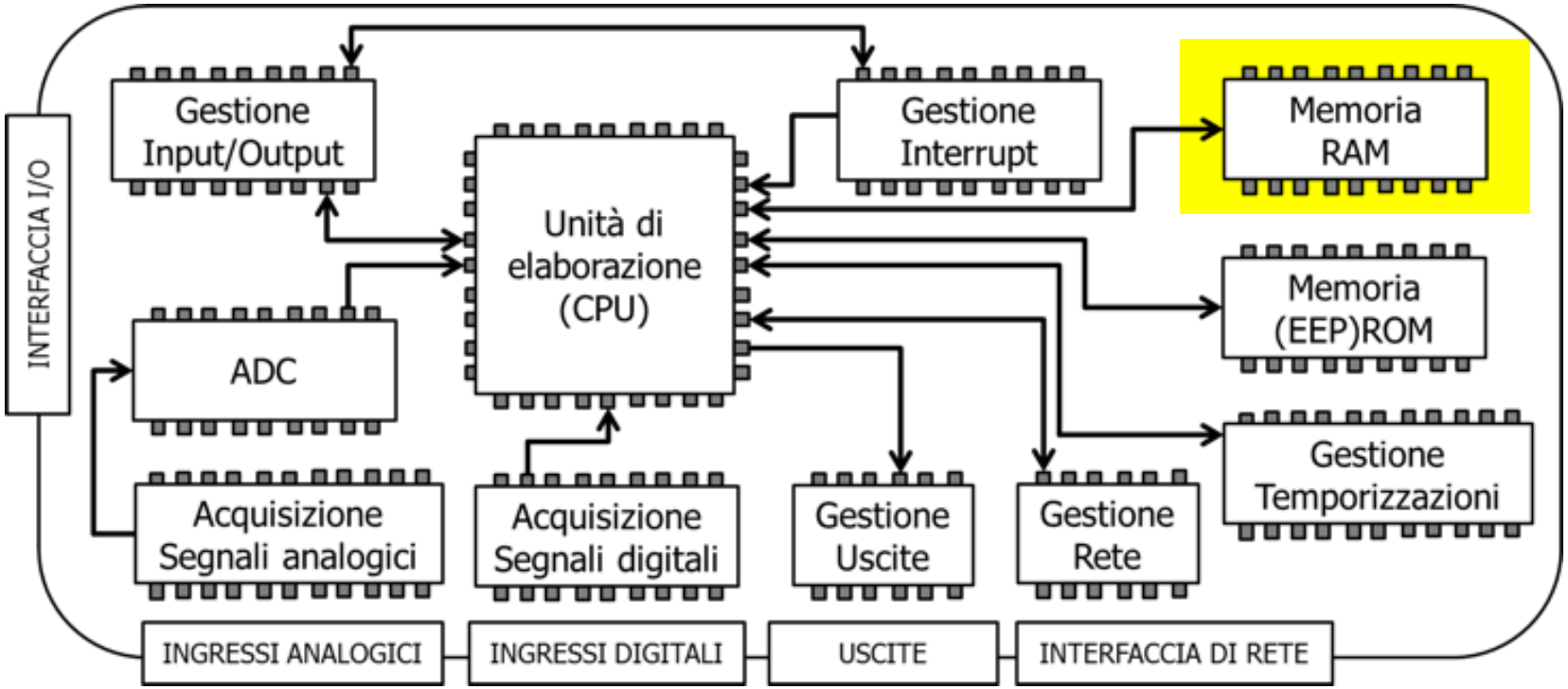
La **CPU** può essere un **microprocessore general purpose** (***numeri interi***), oppure un **digital signal processor** (***numeri interi e in virgola mobile***), o un **field programmable gate array** (***funzioni logiche***)



Architettura Hardware degli Embedded Controller

MEMORIA RAM

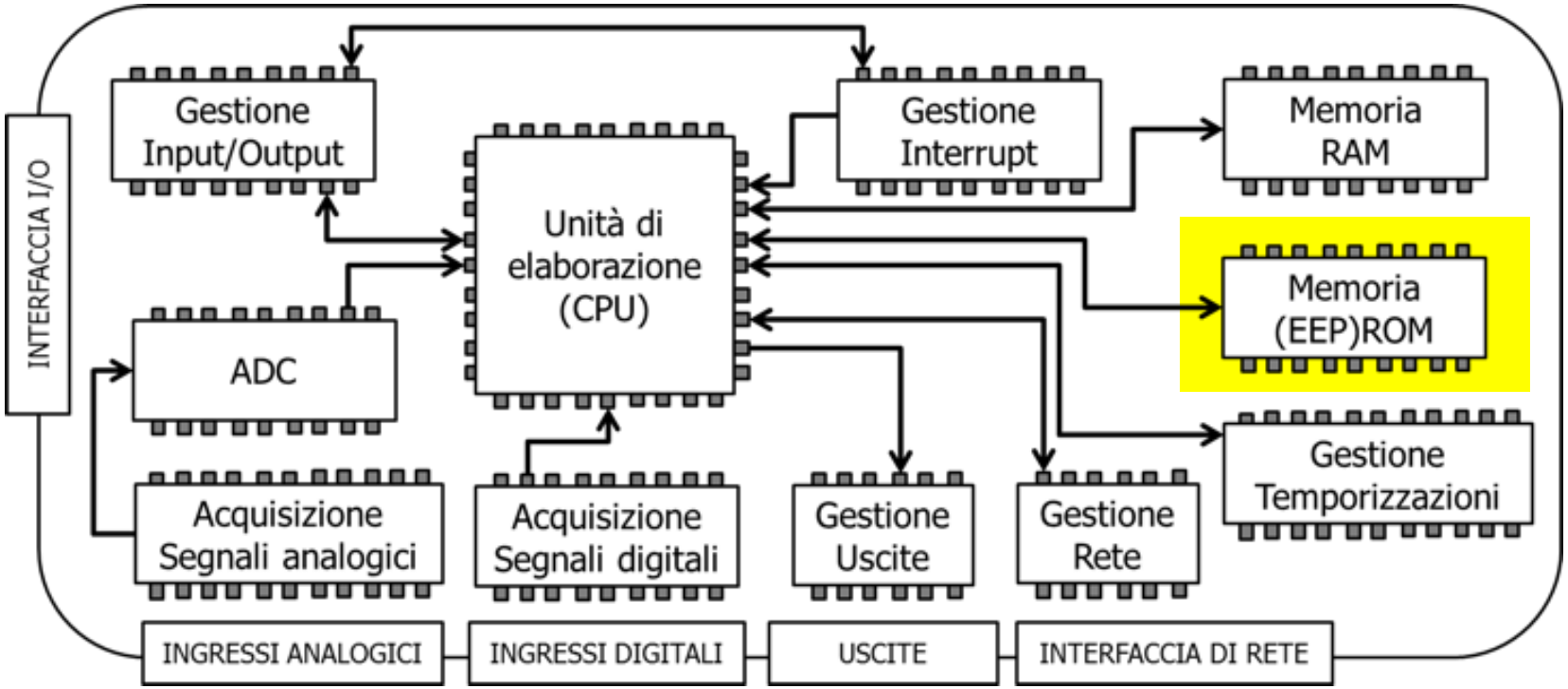
È la **memoria volatile** necessaria per mantenere lo **stato del programma**, ovvero tutti i **dati temporanei** necessari durante l'elaborazione degli **algoritmi di controllo** logico sequenziale.



Architettura Hardware degli Embedded Controller

MEMORIA (EPP)ROM

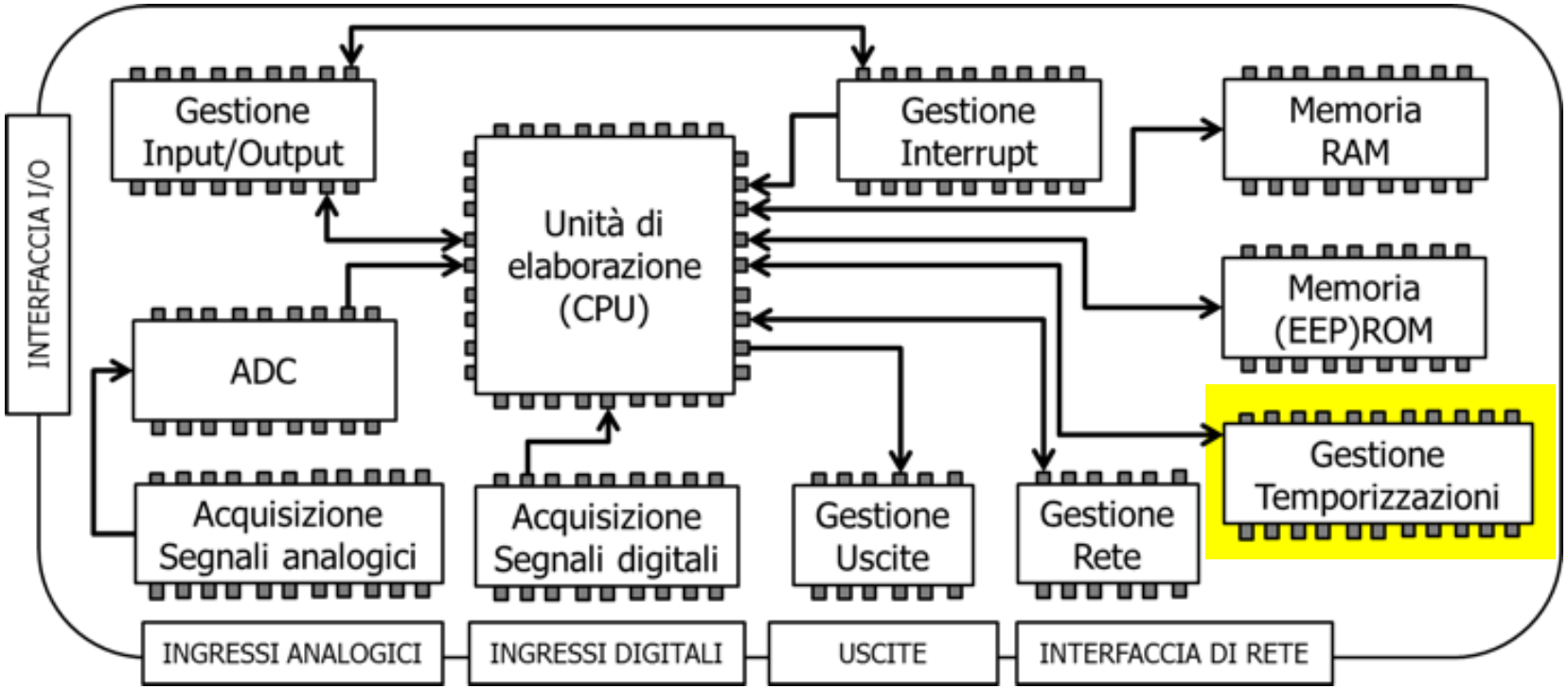
È la **memoria non volatile** necessaria per **memorizzare permanentemente** il **sistema operativo (ROM)** e il **programma utente ([EEP]ROM)**.



Architettura Hardware degli Embedded Controller

GESTIONE TEMPORIZZAZIONI

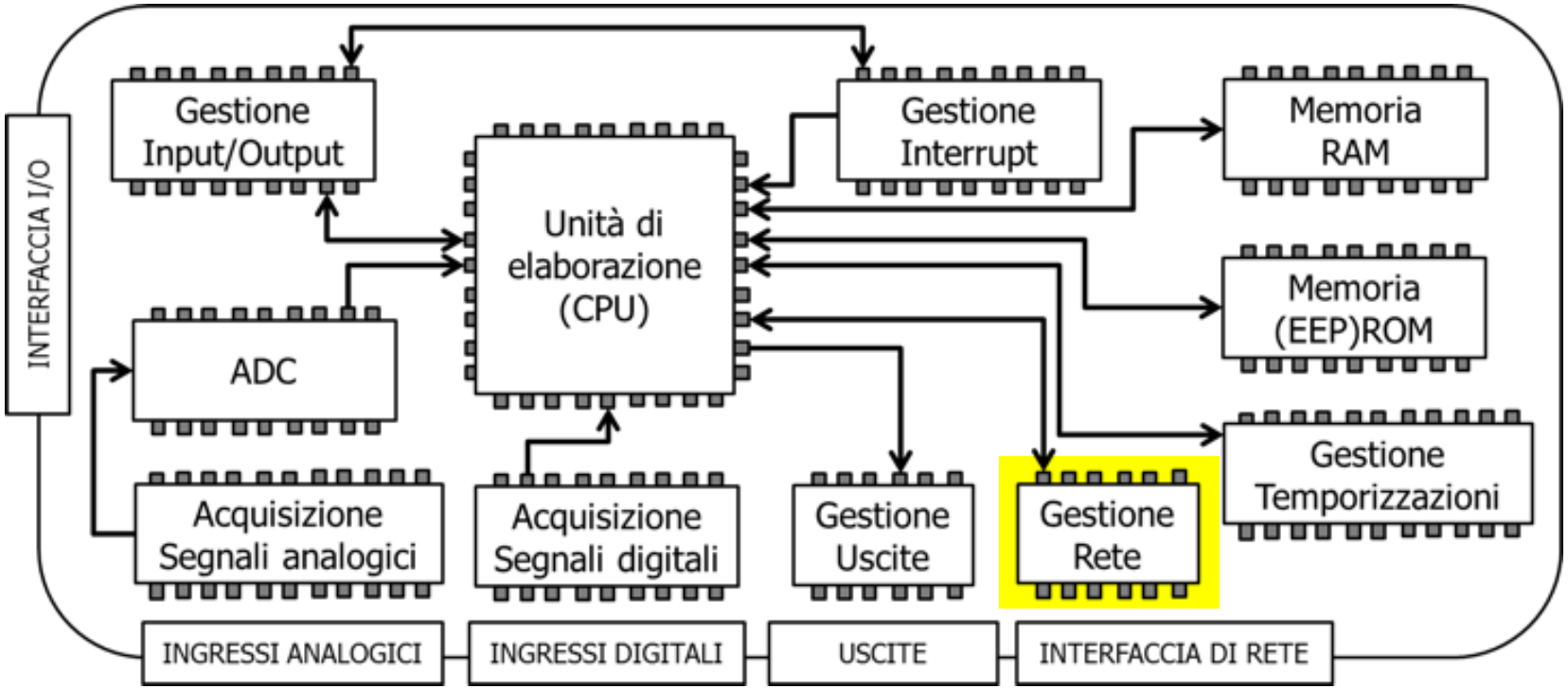
Gestione dei **timer** utili per la **sincronizzazione** e la **temporizzazione** delle attività della CPU.



Architettura Hardware degli Embedded Controller

GESTIONE RETE

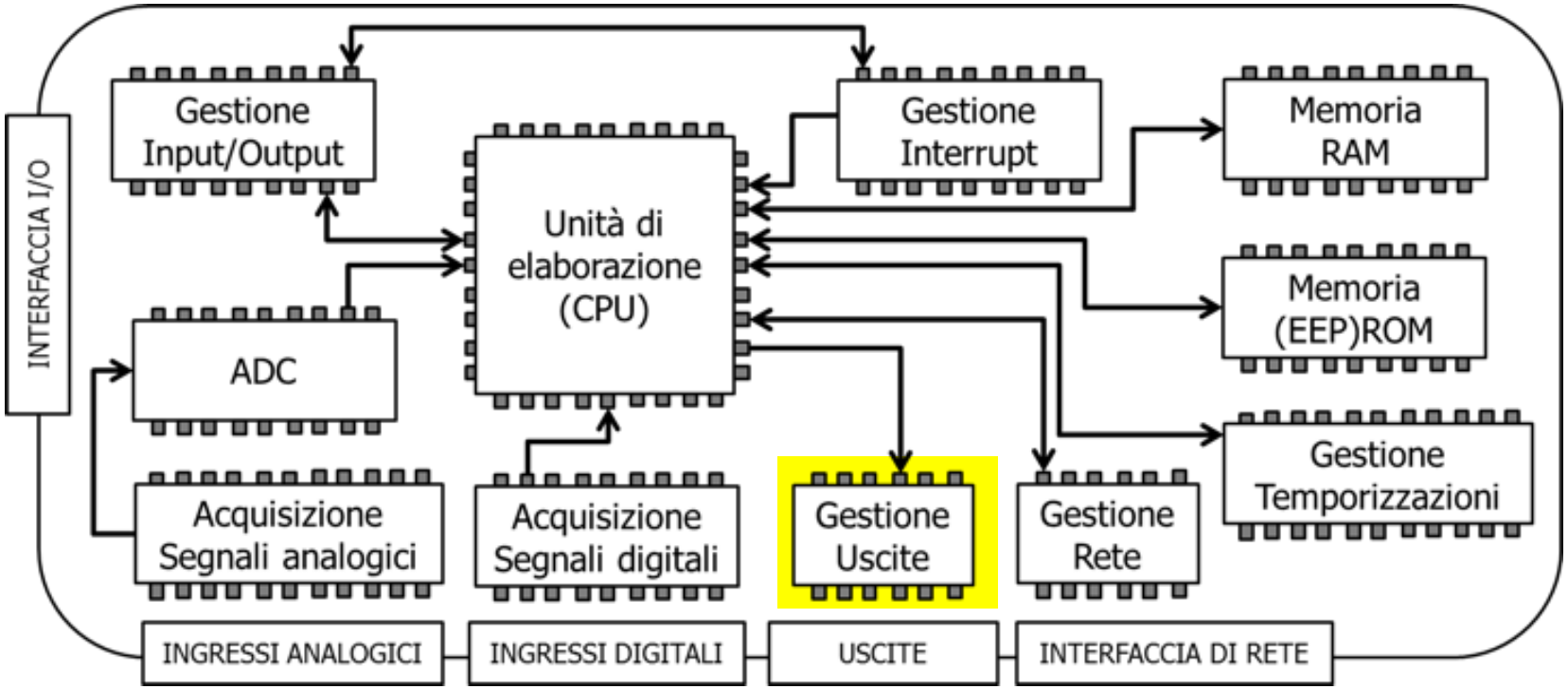
Contiene i **protocolli di base** (di livello **fisico** e di **accesso al mezzo** trasmissivo) per **gestire la comunicazione con altri dispositivi di controllo** (embedded e non).



Architettura Hardware degli Embedded Controller

GESTIONE USCITE

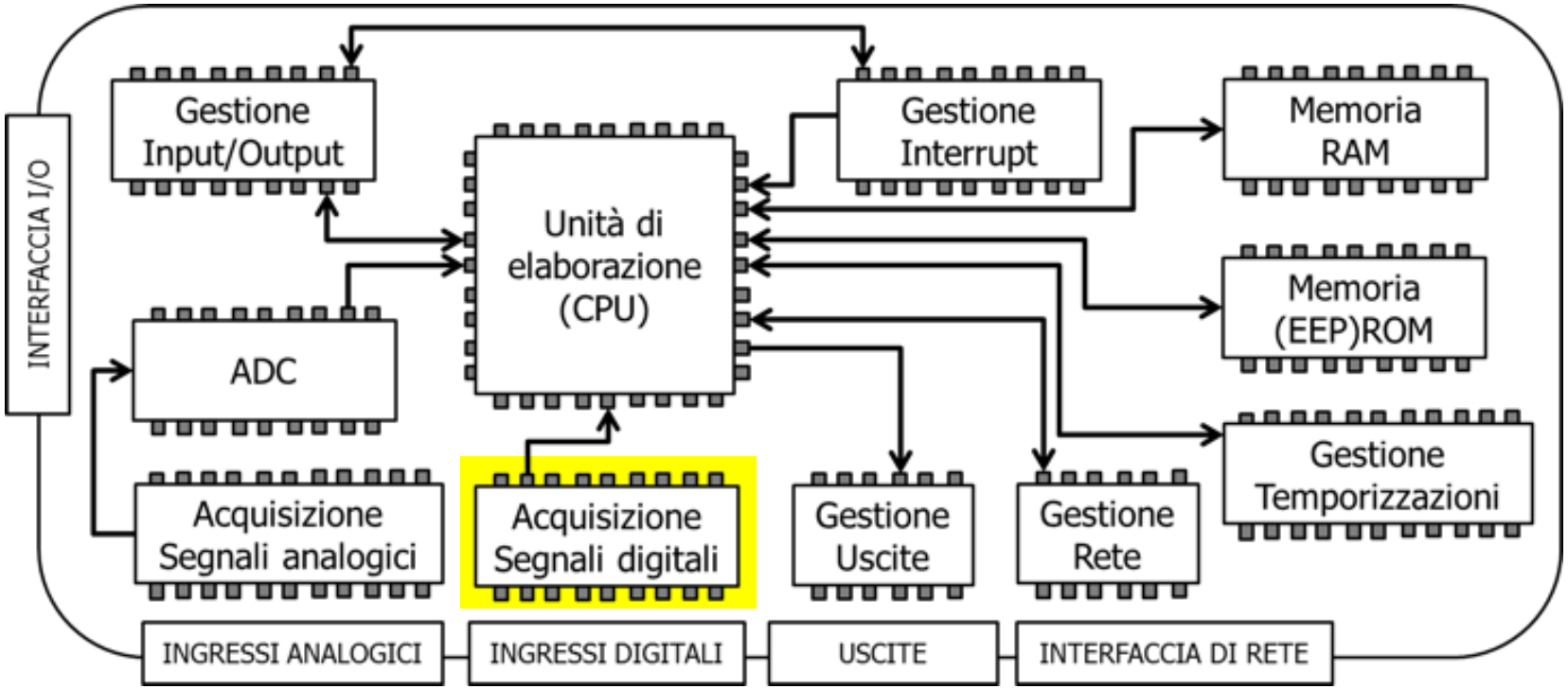
Contiene una serie di circuiti per la **generazione di segnali analogici** (DAC) e **digitali**.



Architettura Hardware degli Embedded Controller

ACQUISIZIONE SEGNALI DIGITALI

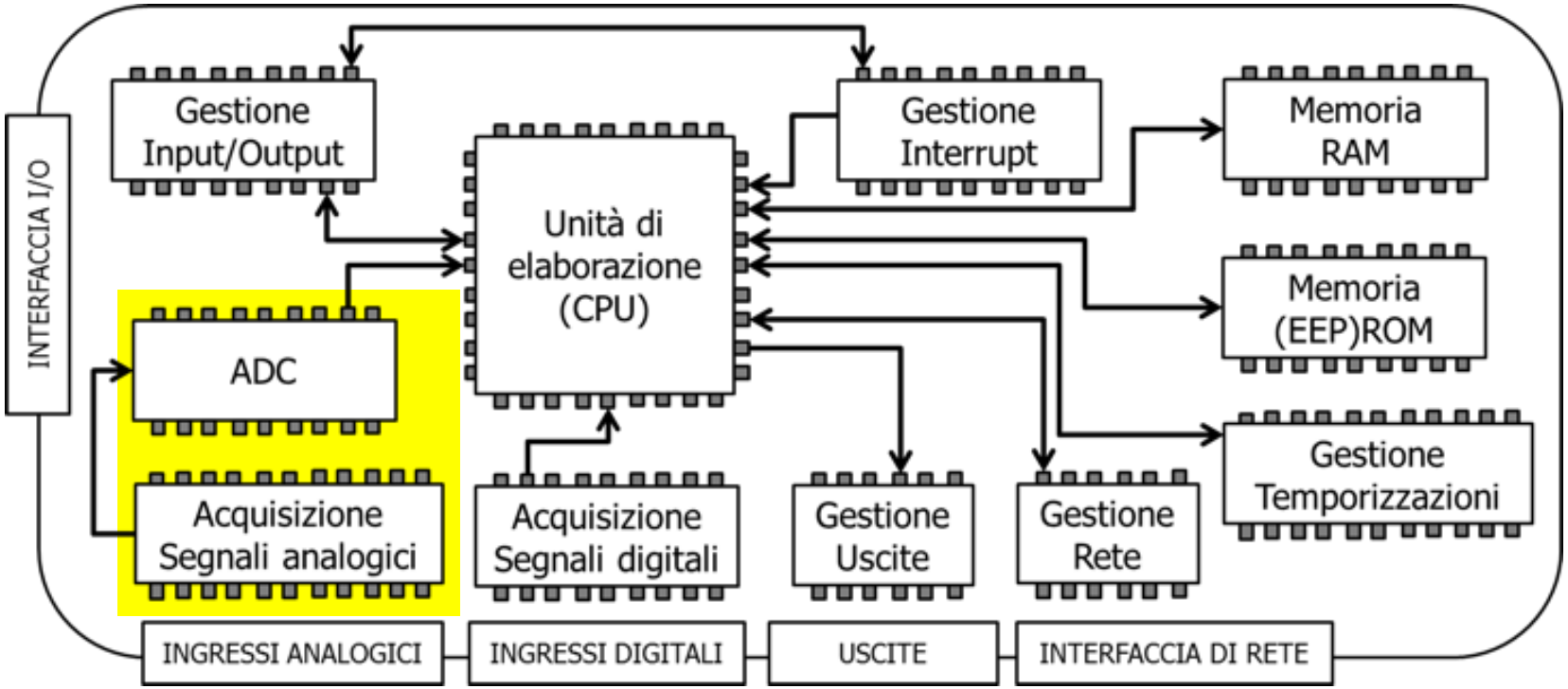
Circuiti per **l'accoppiamento con i segnali digitali in ingresso** (ad es. TTL 0-5V, CMOS 0-3/15V, OPTO)



Architettura Hardware degli Embedded Controller

ACQUISIZIONE SEGNALI ANALOGICI + ADC

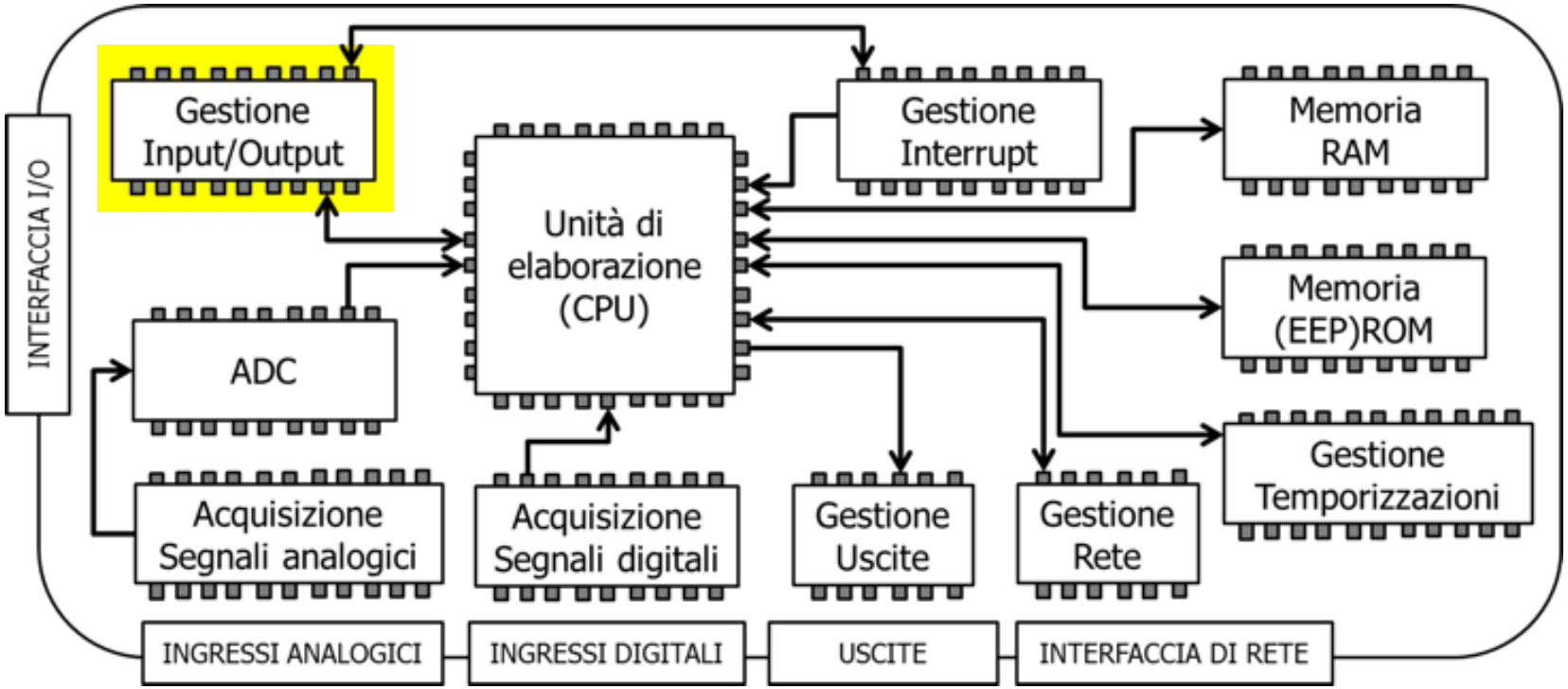
Circuiti per **l'accoppiamento con i segnali analogici in ingresso**, l'eventuale **multiplexer** e il **converitore A/D** (ADC) per il **campionamento** e la **quantizzazione**.



Architettura Hardware degli Embedded Controller

GESTIONE I/O

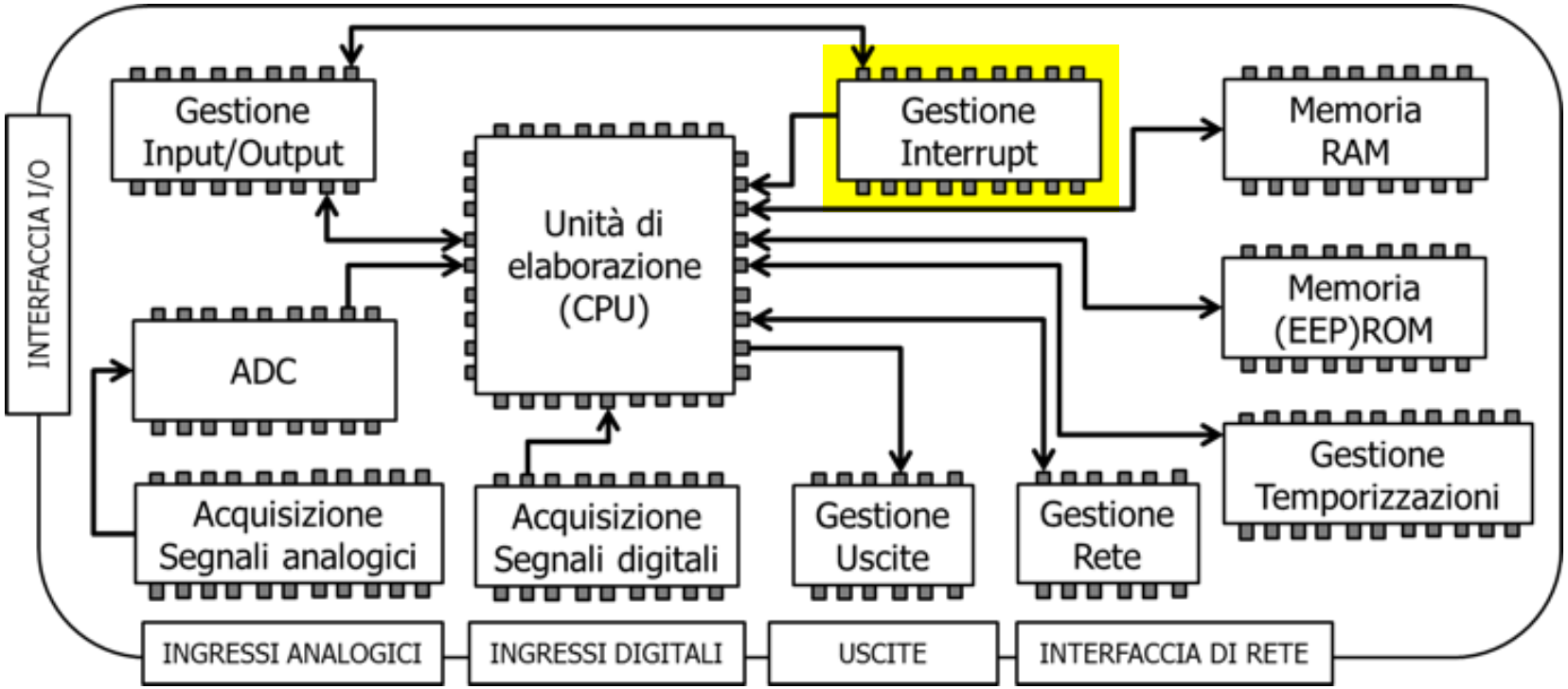
Circuiti per **l'accoppiamento con bus dedicati** o **schede di espansione**.



Architettura Hardware degli Embedded Controller

GESTIONE INTERRUPT

Circuiti per la **rilevazione degli eventi**.



Microcontrollori

DEFINIZIONE

Un **embedded controller** realizzato tramite **un singolo circuito integrato** viene chiamato **microcontrollore**.

OSSERVAZIONE

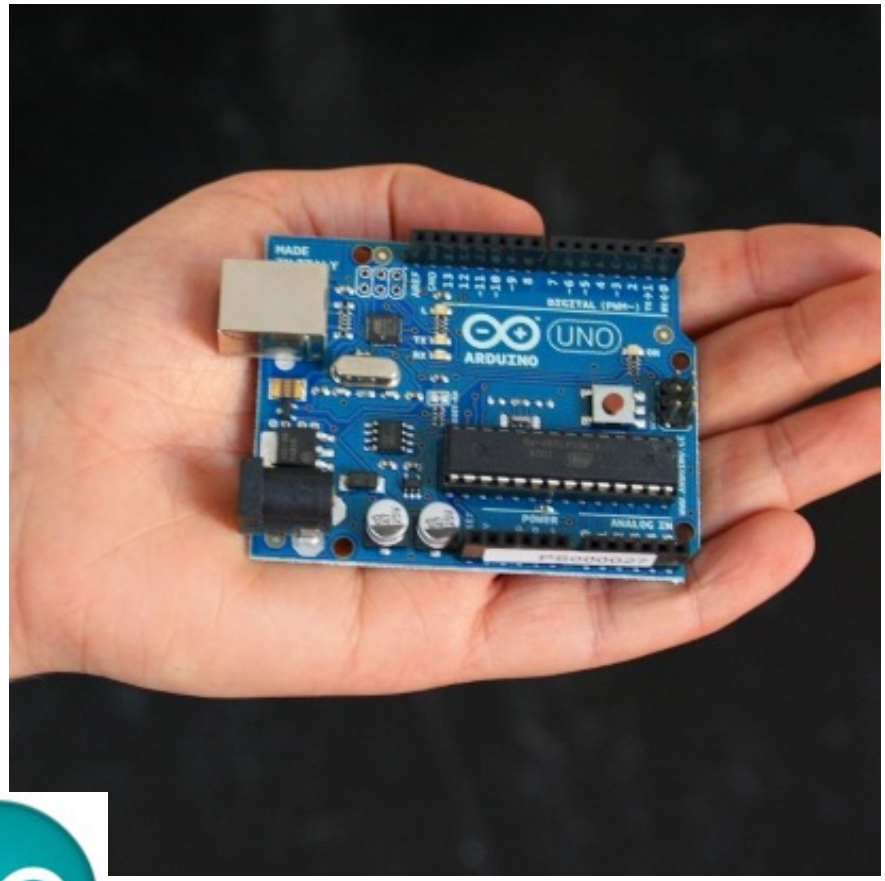
Ha senso prendere in considerazione l'utilizzo di microcontrollori esclusivamente in quei sistemi di **Automazione** che richiedano:

- notevole **riduzione degli ingombri** (cellulari, elettrodomestici, servizi di rete, centraline elettroniche, wearable devices, etc.);
- **numero limitato di segnali di ingresso / uscita** (digitali o analogici);
- **basso consumo energetico** (alimentazione a batteria);
- **HMI minimale** (ad es. touchscreen, tastiera, display LCD, LED) o nulla;
- **interoperabilità** e **integrazione** con altri dispositivi **limitata** o **nulla**.

Scenari applicativi appetibili per i **μcontrollori** sono: **Building Automation, Smart Grid, Smart Cities, Wireless Sensor and Actuator Networks, Mobilità, Health**.

ARDUINO

COSA È ARDUINO?



Arduino is an **open-source** electronics prototyping platform based on flexible, easy-to-use hardware and software.

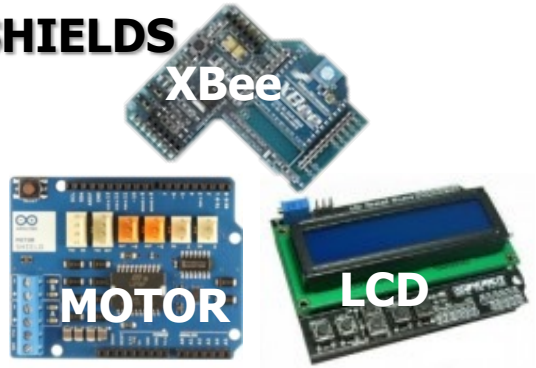
It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

COSA SI PUO' FARE CON ARDUINO?

- Arduino è una iniziativa finalizzata alla definizione di requisiti hw/sw per la costruzione open-source di **controllori embedded operanti a livello di campo.**
- Il successo dell'iniziativa si basa su:
  - Possibilità di scaricare **gratuitamente** gli schemi hardware da www.arduino.cc di tutti i modelli delle **BOARD** Arduino;
  - Possibilità di costruire **gratuitamente** (no royalties) delle **SHIELD Arduino** da collegare meccanicamente con le BOARD al fine di estendere le capacità e le funzionalità della stessa;
  - Possibilità di scaricare **gratuitamente l'SDK** per iniziare da subito a **programmare** le **BOARD Arduino** e le sue **SHIELD ufficiali**.

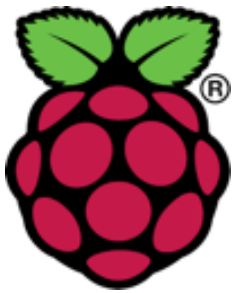
BOARDS

SHIELDS



IDE (Integrated Development Environments)

RASPBERRY PI

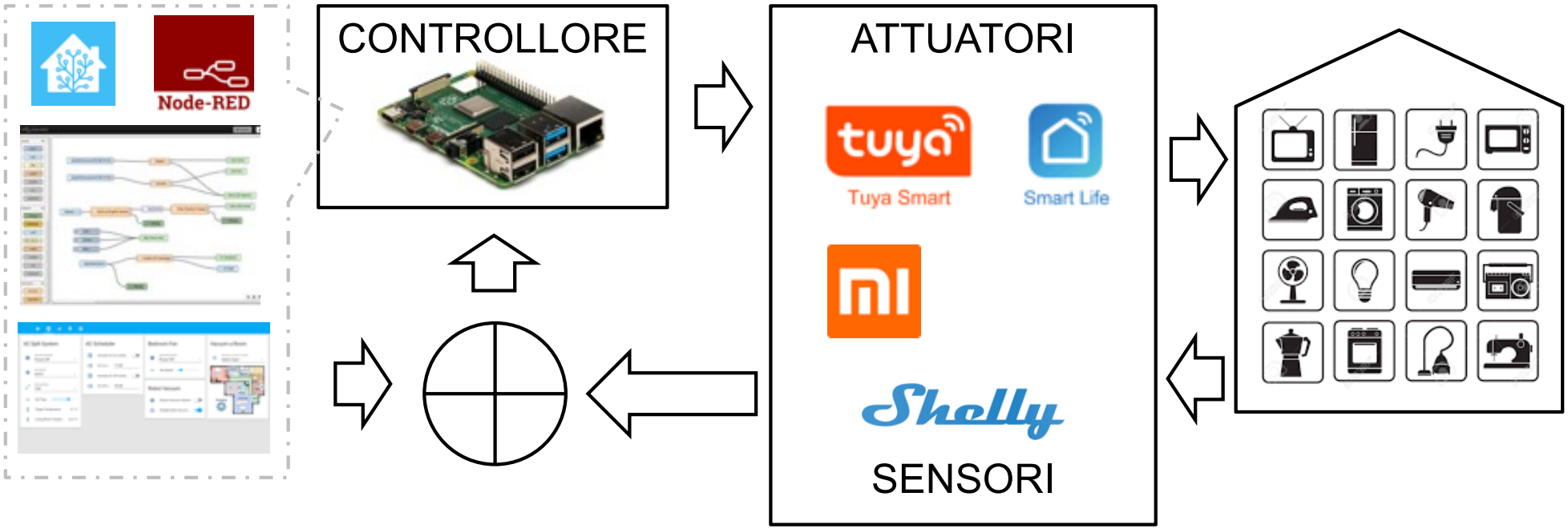


COSA È RASPBERRY PI?

is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse.

COSA SI PUO' FARE CON ARDUINO?

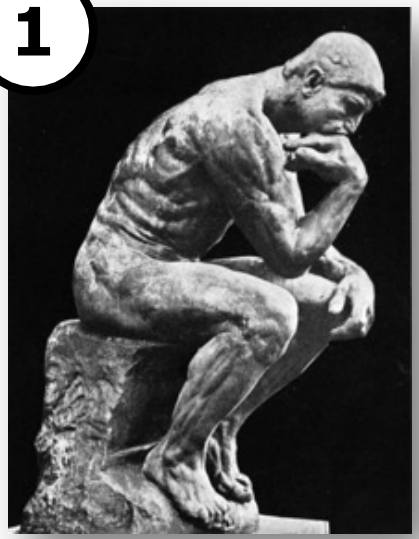
- Raspberry Pi è un mini-PC perfetto per sviluppare sistemi di controllo a livello di coordinamento o di supervisione.
- Il successo dell'iniziativa si basa sulla possibilità di costruire sistemi di automazione low-cost (ad es. domotica)



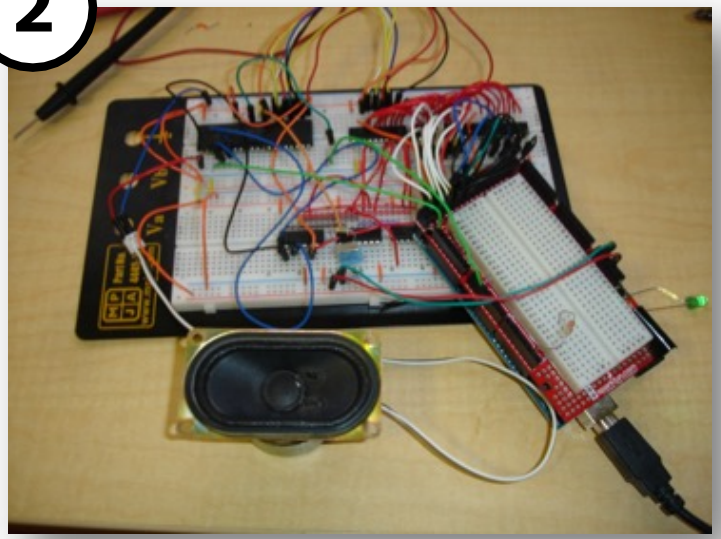
COME USARE UN EMBEDDED SYSTEM IN AUTOMAZIONE ?

1. Trovate un **caso d'uso** applicabile al settore dell'**automazione** (non necessariamente industriale);
2. **Progettate l'idea** e **realizzatela** (spendendo il meno possibile);
3. **Presentate l'idea** come **prova scritta di questo nucleo didattico.**

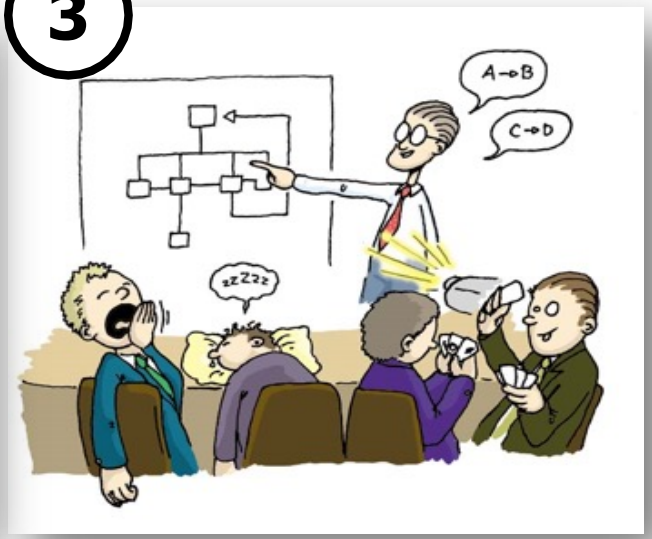
1



2



3



REALIZZAZIONE SU PLC

Controllori a BUS

Quando si deve realizzare un sistema di controllo logico sequenziale caratterizzato da:

- **complessità** di calcolo degli algoritmi molto elevata;
- **elevato numero** di ingressi e/o uscite analogiche/digitali;
- **HMI** complesse ed articolate;
- **interoperabilità** con altri sistemi di controllo ed informativi

Si preferisce sostituire i **controllori embedded** con i **CONTROLLORI A BUS**.

DEFINIZIONE

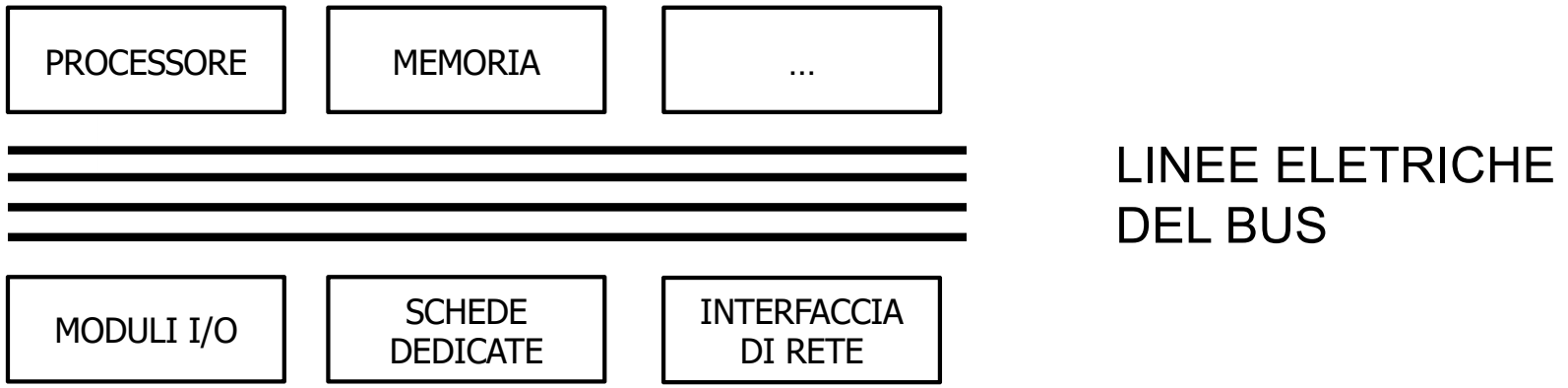
Un **BUS** è un insieme di **linee di trasporto di energia e di informazione** (in generale di **tipo elettrico**) che permettono la **comunicazione tra più dispositivi**.

OSSERVAZIONE

Per **identificare un BUS** è pertanto necessario fissare il **numero di linee**, definire le **funzionalità offerte** da ciascuna linea, i **protocolli di comunicazione** usati dai dispositivi interconnessi e le **interfacce meccaniche**.

Architettura a BUS di un sistema di controllo

In una **architettura a bus**, ad un modulo principale ospitante il **processore**, vengono **connessi tutti gli altri moduli** necessari a comporre il controllore (memoria, moduli I/O, periferiche di HMI, interfacce di rete, schede dedicate, etc.).

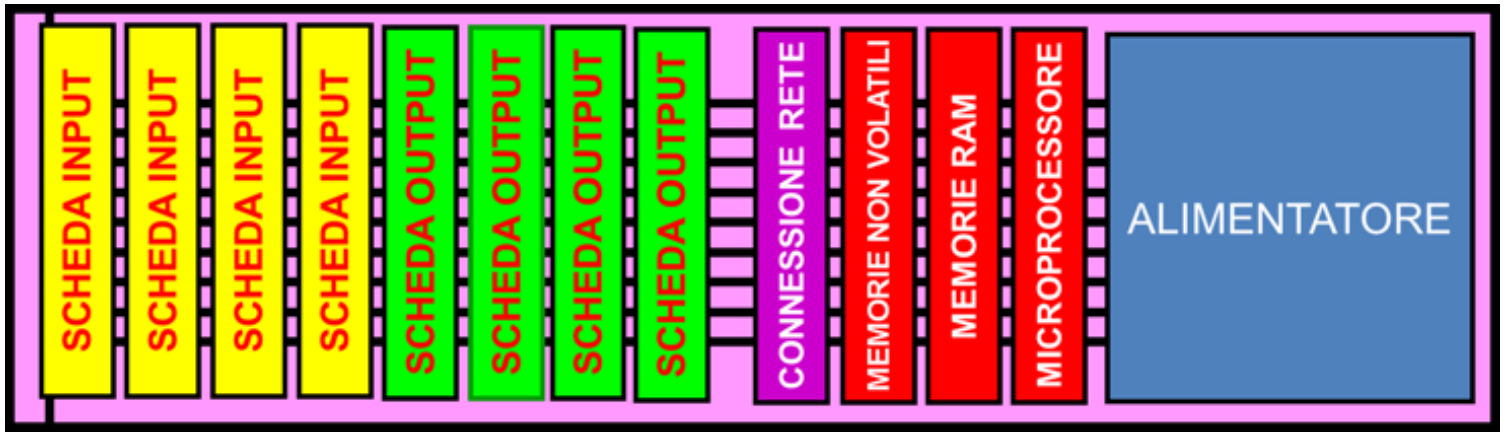


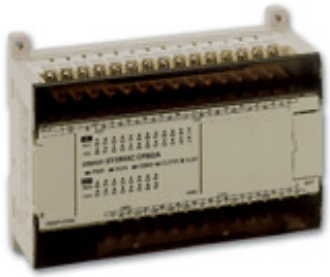
Le **linee elettriche** del BUS vengono raggruppate e differenziate in base alle **funzioni**:

- **linee di INDIRIZZO**
- **linee DATI**
- **linee di ALIMENTAZIONE**
- **linee di CONTROLLO**

Programmable Logic Controller (PLC)

I **controllori a logica programmabile** sono controllori basati su **architettura a bus** e hanno riscosso ampio successo in ambito della **automazione industriale**.





μPLC

PLC MEDI

PLC GRANDI

PROGRAMMAZIONE

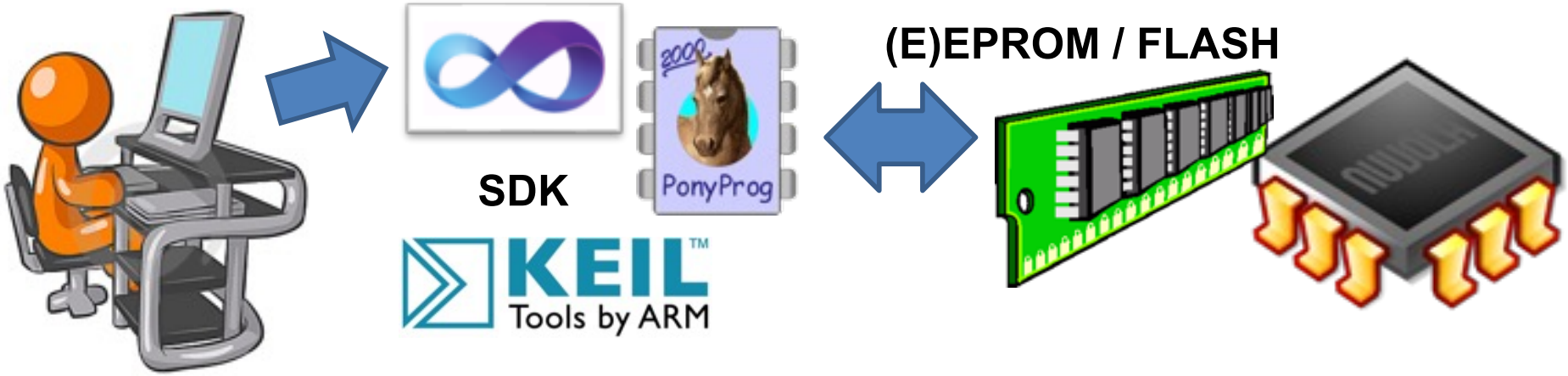
SISTEMA DI SVILUPPO DEL SOFTWARE

OSSERVAZIONE

Ogni **microcontrollore** esegue un **set di istruzioni** (codice macchina) definito dall'utente. È pertanto necessario utilizzare opportuni **sistemi di sviluppo** per caricare il software nei microcontrollori.

DEFINIZIONE

Per **SISTEMA DI SVILUPPO** s'intende l'insieme di strumenti **(kit) software e hardware** necessari alla **generazione del codice macchina** che deve essere eseguito dal processore (implementazione del software), al suo collaudo e messa a punto (debug).



SOFTWARE DI PROGRAMMAZIONE DEI P L C

SECONDO LE NORME IEC 61131-3

LINGUAGGI GRAFICI

LD - LADDER DIAGRAM
FBD - FUNCTION BLOCK DIAGRAM
SFC - SEQUENTIAL FUNCTIONAL CHART

LINGUAGGI TESTUALI

IL - INSTRUCTION LIST
ST - STRUCTURED TEXT

LINGUAGGI DI PROGRAMMAZIONE

SECONDO LE NORME IEC 61131

LINGUAGGI DI PROGRAMMAZIONE GRAFICI

LADDER DIAGRAM
SCHEMA A CONTATTI

ottenuto come **trasposizione** informatica dei **quadri a relè**.

FUNCTIONAL BLOCK DIAGRAM

ottenuto come **trasposizione** dei **diagrammi circuitali** in cui le interconnessioni rappresentano i **percorsi dei segnali** che collegano i vari componenti. I blocchi rappresentano le singole operazioni logiche.

SEQUENTIAL FUNCTIONAL CHART

ottenuto applicando un formalismo grafico per la descrizione di **operazioni logiche sequenziali** e formalismi grafici propri di altri linguaggi di programmazione. utilizzato per descrivere in maniera orientata alla **progettazione sistemi complessi di automazione**.

LINGUAGGI DI PROGRAMMAZIONE

SECONDO LE NORME IEC 61131

LINGUAGGI DI PROGRAMMAZIONE TESTUALI

INSTRUCTION LIST

linguaggio di programmazione di **basso livello** molto simile all'**ASSEMBLER**. Le istruzioni sono costituite da un operatore e da un solo operando e fanno riferimento ad un registro di memoria. I formalismi adottati possono essere molto differenti in quando fissati dal produttore dell'hardware per il PLC.

STRUCTURED TEXT

linguaggio di programmazione strutturato ad **alto livello** con un formalismo che si ispira al **BASIC** e al **PASCAL**. È adatto alla rappresentazione di procedure complesse che non potrebbero essere descritte con i linguaggi grafici.