Appunti IoT Systems

Capitolo 1

I Microcontrollori

Simone Coco

I microcontrollori

Formattato: Titolo, Rientro: Sinistro: 0 cm, Destro 0 cm, Interlinea: singola

Per introdurre i microcontrollori è necessario porci tre domande fondamentali:

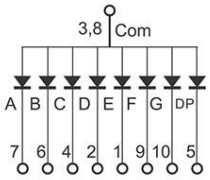
- Cosa è un microcontrollore?
- Qual è la differenza principale tra un microcontrollore e, ad esempio, un microprocessore?
- Perché abbiamo bisogno di un microcontrollore?

Un caso pratico: il microprocessore per il controllo della temperatura

Per rispondere a queste domande ed avere un quadro generale della definizione di microcontrollore analizziamo la differenza tra quest'ultimo ed un microprocessore; ci serviamo di un esempio: il microprocessore per il controllo della temperatura.

Iniziamo con l'osservare quali sono le informazioni che si devono tenere in considerazione e come esse possono essere rappresentate in modo digitale:

- **Lettura periodica della temperatura**: la temperatura essendo una grandezza fisica avrà i valori tipici di una grandezza analogica (valori che variano in un certo range, potendo assumere in modo continuo tutti i valori dell'intervallo), per questo motivo la definiamo **grandezza analogica**. Per rappresentarla in maniera digitale la dobbiamo **digitalizzare**, per digitalizzarla utilizziamo, ad esempio, **[4 BIT]** (16 combinazioni scelti dal progettista, ad esempio 16 intervalli da un grado che vanno da 0 a 15 o codifichiamo da 18 a 34 gradi (cioè sempre 16 bit in più rispetto il grado iniziale), etc…)
- **Sistema di accensione e spegnimento:** serve un bit che segnali l'accensione o lo spegnimento del dispositivo. **[1 BIT]**
- **Display per la temperatura attuale:** potrebbe essere necessario mostrare la temperatura in un display, si prenda il caso in figura: servirebbero 8 bit per accendere o spegnere i led del display più magari 3 bit per segnalare a quale cifra si fa riferimento in una ipotetica rappresentazione a 3 cifre (con la virgola) **[8+3 BIT]**



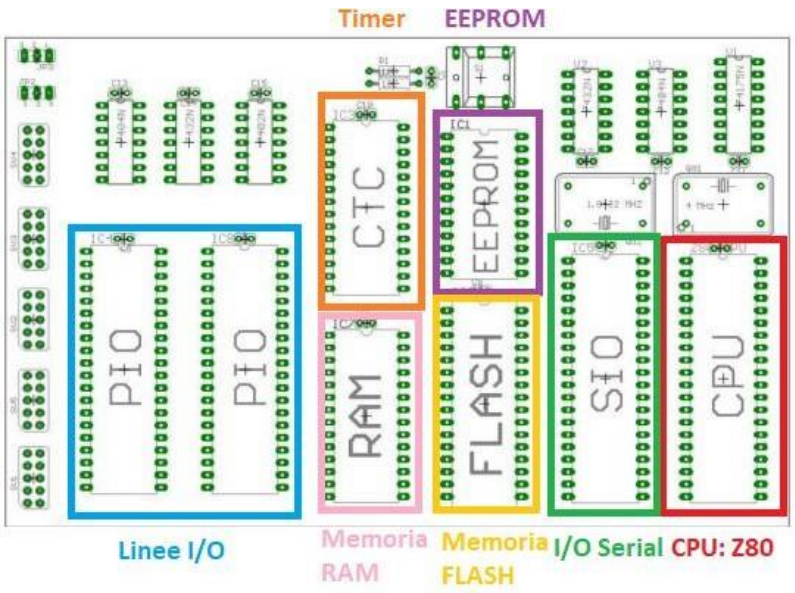
- **Pulsanti per alzare o abbassare il range delle temperature:** se si volessero implementare due pulsanti formati da "+" e "-" per aumentare o diminuire il range delle temperature avremmo bisogno di **[4 BIT]** per manovrare i pulsanti per l'aggiustamento delle due soglie
- **Sistema di comunicazione:** supponiamo inoltre l'esistenza di un sistema di comunicazione per comunicare o aggiornare i dati delle relative misurazioni

Abbiamo in totale bisogno di **20 BIT.**

Si presupponga di creare questo sistema di misurazione con il microprocessore **Z80**, **un processore general purpose** (processore di scopo generico) montato da vecchi calcolatori (come il Commodor 128, il Sinclair…) e per molti sistemi più semplici (vista la semplicità del microprocessore e visto il fatto che il suo progetto è stato reso open source).

Prendiamo questa board per creare il nostro sistema per il controllo della temperatura. In essa possiamo notare:

- Una CPU (o microprocessore), nel nostro caso è proprio lo Z80, chiaramente essa non è indipendente e ha bisogno di altri componenti per svolgere determinati compiti in cui viene impiegata, essa serve come unità computazionale.
- Due Chip aventi linee di Input/Output, punti di connessione dove possiamo connettere elementi del mondo esterno, ognuno dei chip contiene 16 linee di I/O ciascuno, noi necessitiamo di 20pin per i 20bit che abbiamo trovato precedentemente.
- Una componente per la comunicazione seriale: per gli aggiornamenti e la manutenzione del sistema
- Timer: è un chip che serve per effettuare delle operazioni periodicamente, come la letture della temperatura. (NB: non è possibile fare un ciclo while nella CPU, dobbiamo agire separatamente)
- Memoria RAM (Random access memory) per allocare le variabili con cui lavorare.
- La memoria Flash per i programmi da usare che farà il loop per la misurazione della temperature
- EEPROM, una rom riscrivibile per le costanti da mantenere

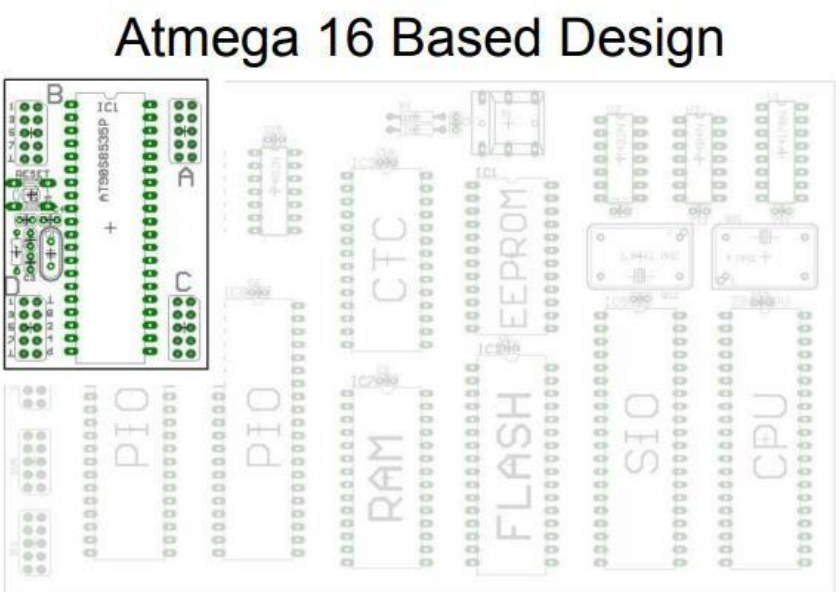


Quindi non abbiamo bisogno solo della CPU ma anche da un corredo di chip utili alla CPU per svolgere i compiti.

Come possiamo vedere ci sono tanti chip sulla board e molto spazio è sprecato perché ogni PIN occupa uno spazio e si necessita di un tot di spazio tra un pin e l'altro.

NB: il circuito stampato viene chiamato **PCB: Printed Circuit Board**

Questo spazio può essere risparmiato, riassumendo il funzionamento del microprocessore in un unico **microcontrollore** che funziona proprio come il microprocessore o addirittura meglio. Vogliamo vedere un esempio pratico di microcontrollore, l'AT Mega 16:

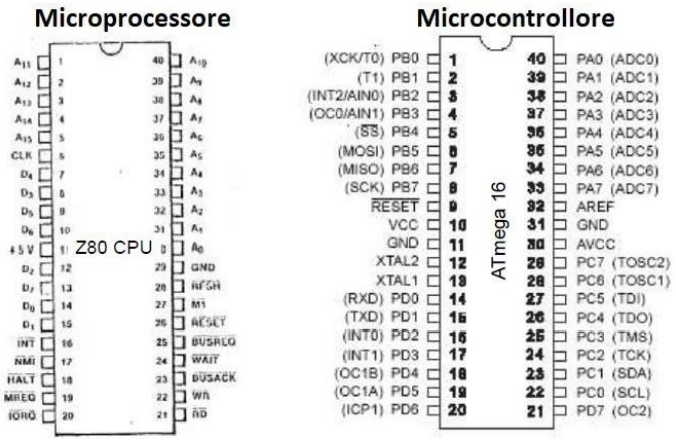


Come si può visibilmente notare, questo microcontrollore è molto più piccolo rispetto alla board.

Il microcontrollore è più compatto e in esso abbiamo integrato tutte le componenti utili allo svolgimento di un compito in un singolo chip (CPU, memorie, EEPROM…) e abbiamo un grande risparmio tra soldi e spazio.

Differenze tra microprocessore e microcontrollore

Formattato: Rientro: Sinistro: 0,83 cm, Sporgente 0,64 cm, SpazioPrima: 0 pt



Microprocessore: il microprocessore è una semplice CPU che deve essere inserita in una board per avere il supporto di altri elementi quali le memorie etc…
Il microcontrollore ha i primi pin che iniziano per "A" (Address), questi sono **pin di memoria** che la CPU utilizza in quanto essa è priva di memoria. Questi pin ovviamente non esistono nel microcontrollore poiché esso ha una memoria integrata al suo interno (ricorda: il microcontrollore è formato da tutto l'insieme di elementi che sussistono nella board). Il microprocessore ha poi un altro insieme di pin per vari funzioni per un totale di N pin.

Microcontrollore: il microcontrollore ha il numero equivalente di pin: N ma essi sono diversi da quelli del microprocessore; i pin sono **general purpose I/O:** essi hanno un nome e appartengono a delle determinate **classi: PA, PB, PC, PD tutti numerati da 0 a 7** (per un totale di 32 pin), possiamo inoltre vedere una sigla tra parentesi, essa si riferisce alla funzionalità accessoria possibile: **i pin possono essere utilizzati per trasmettere un bit generico (un dato), ma all'occorrenza possono essere utilizzati dai progettisti per varie funzioni e queste funzioni sono proprio le funzionalità accessorie possibili** (interrupt, etc…)**, ecco perché si chiamano "Pin general purpose I/O",** con questi pin possiamo interconnettere ciò che serve al microcontrollore che, in pratica, sono solo i sensori poiché il microcontrollore ha già tutto integrato al suo interno. Poi vi sono ovviamente altri pin che non hanno alcuna sigla tra parentesi poiché essi non sono general purpose (hanno funzionalità specifiche e ben definite), ad esempio GND, il pin di alimentazione o il pin RESET.

In generale, **un microcontrollore conviene utilizzarlo quando si hanno molte operazioni di input output**, grosse integrazioni da fonti eterogenee che devono essere processate con **operazioni computazionali semplici.**

Il microprocessore invece viene adoperato perlopiù quando si devono affrontare operazioni computazionali più complesse, che un microcontrollore non potrebbe fare, aventi magari un numero di **input finito e non tropo elevato.**

Scelta di un microcontrollore

Normalmente, quando si parla di processore (e quindi il discorso è analogo al microprocessore), si sceglie un componente piuttosto che un altro ponendo l'attenzione sulle prestazioni: più è performante e meglio è, talvolta anche comprando una CPU fin troppo performante per i casi in cui essa viene adoperata.

Per i microcontrollori questo non vale, non si necessita di scegliere un microcontrollore con "abbondanza di performance", il ragionamento è opposto: quando si progetta un sistema, non ci si ritrova a scegliere tra tante applicazioni poiché il sistema dovrà svolgere un determinato compito e quindi il microcontrollore da adoperare sarà orientato a risolvere una ben precisa computazione, sarebbe inutile prendere un microcontrollore che abbondi in performance, piuttosto si deve realizzare un sistema nella maniera meno costosa ed efficiente possibile: ad esempio, qualche sistema dovrebbe magari consumare meno energia possibile in quanto portatile, quindi si preferisce un microcontrollore a basso consumo energetico piuttosto che un altro in grado di gestire più input output (del necessario) ma ad alto consumo energetico.

Esistono varietà incredibili di microcontrollori, ognuno più o meno adatto alla risoluzione di un problema.

Famiglia di microcontrollori

La prima cosa da scegliere quando si deve prendere un microcontrollore è la sua **famiglia:** la famiglia è un insieme di microcontrollori diversi tra loro ma che condividono un core principale: ogni microcontrollore è formato, come detto, da vari piccoli componenti (memoria RAM, memoria Flash, EEPROM…) e tra queste vi è, chiaramente, un microprocessore; **tutti i microcontrollori appartenenti ad una famiglia hanno un core (microprocessore) compatibile (uguale), avente un determinato set di istruzioni, che possono fare una serie di operazioni.**

Stabilita la famiglia attraverso la tipologia di core in comune, ogni elemento avrà delle caratteristiche diverse l'uno dall'altro. Le caratteristiche che variano sono gli elementi integrati insieme alla CPU: la grandezza della memoria FLASH, il numero di linee I/O, l'eventuale presenza o assenza di conversione analogico/digitale (se serve o meno).

Queste caratteristiche ne determinano il prezzo, ecco perché si sceglie la soluzione più economica e non si tende a prendere componenti più complessi possibile per fare più operazioni diverse: è inutile prendere componenti costosi che fanno operazioni futili alla risoluzione del problema.

Il prezzo si replica per prodotti di massa quindi un incremento minimo del prezzo per un componente potrebbe portare ad un grande aumento dei costi quando si ha una produzione di massa.

Facciamo un esempio di famiglia di microcontrollori, la famiglia **Atmel's AVR family:**

Controller	Flash (KB)	SRAM (Byte)	EEPROM (Byte)	I/O-Pins	A/D (Channels)	Interfaces
AT90C8534	8	288	512	7	8	
AT90LS2323	2	128	128	3		
AT90LS2343	2	160	128	5		
AT90LS8535	8	512	512	32	8	UART, SPI
AT90S1200	1	64		15		
AT90S2313	2	160	128	15		
ATmega128	128	4096	4096	53	8	JTAG, SPI, IIC
ATmega162	16	1024	512	35		JTAG, SPI
ATmega169	16	1024	512	53	8	JTAG, SPI, IIC
ATmega16	16	1024	512	32	8	JTAG, SPI, IIC
ATtiny11	1		64	5+1 In		
ATtiny12	1		64	6		SPI
ATtiny15L	1		64	6	4	SPI
ATtiny26	2	128	128		16	SPI
ATtiny28L	2	128		11+8 In		

Analizziamo l'ordine delle dimensioni e del numero di componenti di questa famiglia di microprocessori:

Come possiamo vedere, le **memorie flash** sono nell'ordine dei Kilobyte, possono variare dalle unità alle centinaia di KB a seconda del compito che il microcontrollore è chiamato a svolgere.

Le **RAM** utilizzate (SRAM) sono nell'ordine dei byte poiché esse mantengono informazioni minime (i singoli bit), possono spaziare dai 60 ai 4000 byte, a seconda del microprocessore.

Anche le **EEPROM** sono nell'ordine dei byte e anche esse variano in grandezza dai 60 ai 4000 byte.

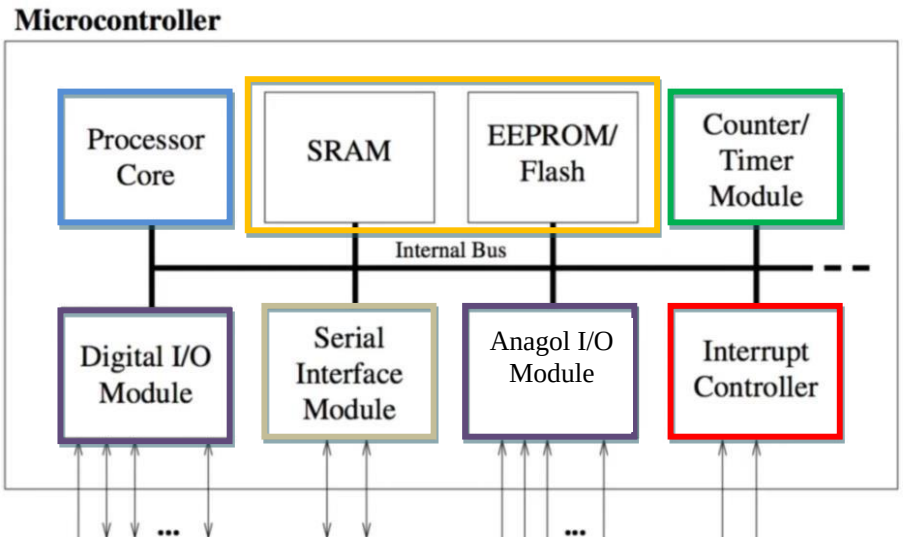
I **pin I/O** possono variare in numero nell'ordine delle decine (per questa famiglia ma magari potrebbero variare in ordini superiori-)

Ovviamente questi numeri non saranno negativi, non avrebbe senso parlare di memorie aventi numero di byte negativi o pin in numero negativi.

È importante notare come multiprocessori della stessa famiglia hanno delle pipeline molto semplici che cercano di eseguire le operazioni in un solo ciclo di clock. Questo significa che le istruzioni assembly per un determinato core sono identiche per tutti i microcontrollori della famiglia montante tale core, in altre parole: **i microcontrollori di una stessa famiglia condividono le stesse istruzioni assembly.**

Layout di un microcontrollore

Ogni microcontrollore ha tutte le componenti connesse da un bus interno e sono tutte integrate sul chip. I moduli del microcontrollore sono poi connessi al mondo esterno dai pin di Input/Output



Facciamo una panoramica delle componenti che andremo dettagliatamente a studiare più avanti:

- **Processor core (CPU)**
  Contiene l'unità aritmetico-logica [ALU]-, l'unità di controllo [CU] e i registi (è una CPU a tutti gli effetti). Ricordiamo che è uguale per elementi della stessa famiglia.

- **Memorie**
  Le memorie si suddividono in memoria dati e memoria di istruzioni ed alcuni microcontrollori hanno i DMA che collegano i componenti periferici e le memorie (lo vedremo in seguito)

- **Controller degli interrupt**
  Qualora si verificassero delle condizioni esterne, il microcontrollore deve ricevere un segnale per fermarsi e gestire la condizione che si è venuta a creare, questo meccanismo spetta agli interrupt (non basta lasciare la gestione delle condizioni con il timer)

- **Timer/Counter**
  Ci sono avvenimenti periodici che possono essere scanditi attraverso l'utilizzo di un timer

- **Pin di I/O**
  In numero variano dai 3-4 fino ai 90 pin e sono l'interfaccia fondamentale con il mondo esterno, qualsiasi collegamento con il mondo esterno è gestito con tali pin.
  Possono essere **digitali**: possono assumere grandezze aventi valori 1 o 0
  Possono anche essere **analogici**: possono assumere, in maniera continua, qualsiasi valore.
  Ovviamente, a prescindere dalla natura dei pin I/O, i dati all'interno del microcontrollore vengono gestiti in formato digitale (quindi qualora entrassero dati in forma analogica, questi devono essere convertiti in forma digitale.)

- **Interfacce**
  Le interfacce sono fondamentali sono necessarie per l'**aggiornamento** del programma, verificare cosa succede in debugging…
  Serve per la comunicazione tra pc e microcontrollore, attraverso i pin di I/O e un seriale.

Altri componenti possono essere:

- **Watchdog Timer**
  È un timer che scandisce il tempo fino ad arrivare a 0. Serve per resettare un microcontrollore: se il microcontrollore si blocca, il watchdog timer lo resetta. Per evitare che il microcontrollore venga resettato quando in realtà non si è bloccato, quest'ultimo incrementa il watchdog timer, impedendogli di arrivare a zero.

- **Debugging Unit**
  Non sono strettamente necessari al microcontrollore per il suo funzionamento ma sono utili ai fini del debug; alcuni controllori sono equipaggiati con una parte hardware addizionale che permette il controllo da remoto permettendo il debugging del chip da un computer, questo hardware è la debugging unit.

Confronto tra microcontrollori e microprocessori

Riprendiamo ancora il confronto tra microcontrollori e microprocessori, continuando ad evidenziarne le differenze:

- **I microcontrollori non hanno la MMU (Memory Management Unit)**: quando si ha un programma, questo necessita di indirizzi di memoria che gli devono essere allocati per la sua evoluzione, questi possono differire dagli indirizzi fisici (reali) in memoria e vengono mappati dalla MMU. Nel microcontrollore non è necessaria la MMU poiché non si ha un sistema operativo: non si devono gestire indirizzi da associare ai vari processi; il microcontrollore "vive" all'interno di un programma, fa partire un programma e non deve eseguire nessun altro processo (ecco perché non fa eccessive operazioni computazionali e non richiede l'utilizzo di core molto potenti). Non avendo la necessità di grandi prestazioni e quindi di grandi parallelismi, la **pipeline delle istruzioni è semplificata.**
- **I microcontrollori non hanno memoria cache** poiché esse servono per accedere a grandi memorie esterne ai fini di avere prestazioni migliori ma nei microcontrollori non abbiamo necessità di accedere a componenti esterni molto lenti. (inoltre, le memorie cache sono costose e ingombranti).

Sommario

Un microcontrollore è una versione semplificata di un microprocessore al quale viene integrata la memoria, i pin di I/O, i timer e le periferiche (= la periferica è tutto ciò che sta all'esterno della CPU) in un singolo chip.

I costi di produzione, di sviluppo ed energetici di un sistema a microcontrollore sono ridotti rispetto a quello di un sistema general purpose poiché in quest'ultimo non si sa a priori quale compito si andrà ad affrontare. Per un microcontrollore si sa a priori quale sia il compito che andrà a svolgere e quindi si può stabilire quale sia il microcontrollore più adatto alle specifiche del prodotto (sistema) da progettare, scegliendo un componente che si "ritaglia" apposta al sistema.

Ci sono però delle situazioni in cui l'utilizzo di un microcontrollore non è la soluzione migliore: ad esempio, le situazioni real time in cui la progettazione di un chip integrato (ovvero un hardware, un insieme di componenti) che deve svolgere un compito specifico è la migliore soluzione.
[Se un aereo avesse bisogno di un comando per l'apertura immediata di un portello sarebbe meglio usare un chip integrato piuttosto che un microcontrollore poiché serve una risposta in tempi brevissimi] Quindi a volte per soluzioni real time il microcontrollore non è il migliore dispositivo da scegliere conviene costruire un chip addetto ad un solo compito specifico.