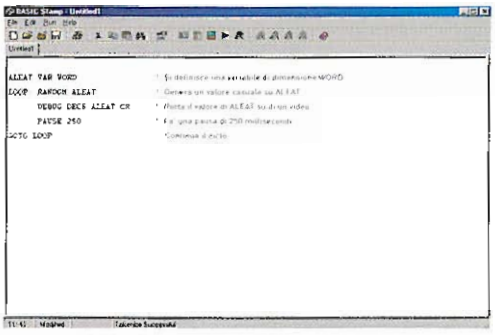
Programma che genera numeri casuali

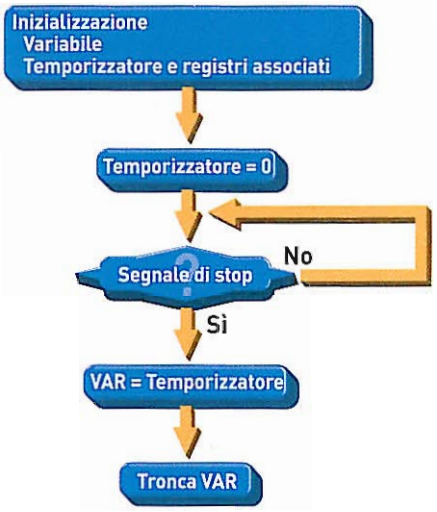
Non tutti i compiti che un microrobot esegue hanno bisogno di una temporizzazione esatta, cioè di un inizio e di una fine con lo stesso intervallo di tempo. Questo tempo a volte è dato dal compimento di una missione, in altre può interessare che sia completamente casuale. In alcuni microrobot questa caratteristica può addirittura essere determinante.

Presentazione del problema

In molti linguaggi di programmazione esiste un'operazione denominata RANDOM, che genera numeri casuali da utilizzarsi secondo necessità. La dimensione di questi numeri dipende dal linguaggio e dovrà essere adattata per delimitarla o ampliarla. Ci sono casi in cui questa istruzione non esiste come succede nell'assembler dei PIC e quindi deve essere "costruita". Vediamo questa volta come generare un numero casuale partendo dai dispositivi che ci fornisce il PIC e come adattarlo per il suo utilizzo in differenti situazioni in cui il range è importante.



Linguaggi come il Basic Stamp di Parallax dispongono di una istruzione RANDOM.



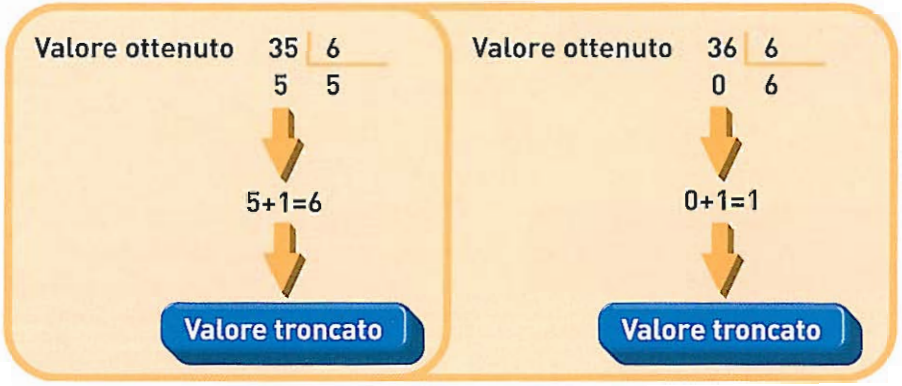
Organigramma della generazione di un numero casuale.

Idea dell'algoritmo

Per la generazione di un numero casuale utilizzeremo i temporizzatori. A seconda del range che vorremo ottenere utilizzeremo quelli da 8 bit (TMR0 e TMR2) o quello da 16 (TMR1).

In entrambi i casi, i passi da realizzare sono simili. Come sempre, per prima cosa inizializzeremo le variabili da utilizzare. Per questo algoritmo bisogna tener presente che i temporizzatori dovranno evolvere nel modo più rapido possibile, per fare diminuire la possibilità di generare risultati uguali a cicli di programma brevi. Al momento di inizializzare i registri associati ai temporizzatori, quindi, si annulleranno i predivisori e i postdivisori di frequenza, in modo che si incrementino a ogni ciclo di istruzione facendoli lavorare con il clock interno del sistema, nel caso in cui si disponga di più di una possibilità. Anche se il TMR2 funziona in maniera diversa dal TMR0 e dal TMR1, in questo caso per noi è uguale. Inizializzando con valore 0 il TMR0 e il TMR1, inizieranno a contare a partire da questo punto, andranno in overflow arrivando a FF (o FFFF), e torneranno nuovamente a 0 per poi ricominciare a contare.

Tuttavia il TMR2 ha una particolarità, cioè quando il suo valore equivale al valore inserito in PR2, torna a 0; in questo modo non esegue il ciclo completo. Per farlo lavorare in modo uguale agli altri, bisognerà inserire in PR2 il valore 255. Questo particolare oltre a renderlo idoneo per il nostro problema, ci permette di generare un numero casuale per poter simulare un dado elettronico, in questo caso l'opzione sarebbe di inserire in PR2 il valore 5, e permettere così che si possano generare numeri fra 0 e 5, a cui in seguito sommare il numero 1 per adattare il range (1-6). Potrebbe succedere inoltre che il TMR2 sia già occupato da un'altra risorsa o che ci interessi utilizzare



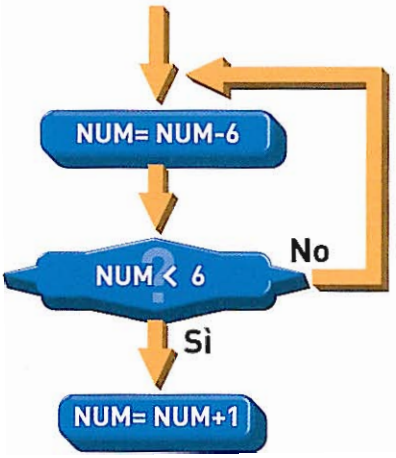
Troncamento di un valore tramite divisione.

i tre temporizzatori contemporaneamente. Pensate ad esempio a una macchina slot-machine da tre ruote, ognuna con un pulsante per fermare il conteggio. In questi casi potrebbe essere necessario utilizzare uno degli altri temporizzatori e aver bisogno di troncare il valore ottenuto per adattarlo al range. In ogni caso lo stop al conteggio verrà segnalato da un fattore esterno, come ad esempio un pulsante. Verificando continuamente il valore di questo pulsante, al momento in cui cambierà di valore si acquisirà il valore istantaneo del temporizzatore. Uno dei modi per troncare il valore ottenuto è applicare una divisione, in modo tale che se volessimo ottenere un valore fra 1 e 6 potremmo dividere per 6 e tenere il resto, a cui dovremo sommare 1, dato che il resto può variare tra 0 e 5.

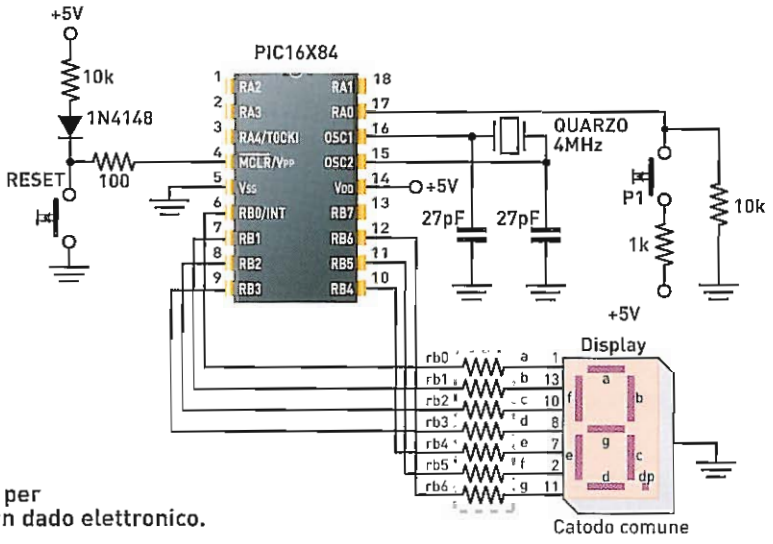
In questo caso, però, ci troveremo nuovamente di fronte a un problema, perché come abbiamo visto nel capitolo corrispondente, nell'assembler del PIC non esiste l'espressione di divisione, quindi la stessa operazione la dovremo risolvere tramite sottrazioni successive seguendo l'organigramma riportato nella figura.

Sensori e attuatori

Questo algoritmo necessita solamente di un segnale digitale di ingresso per fermare il conteggio e generare il numero casuale, se si desidera avere più di un numero si potrebbero utilizzare tutti e tre i temporizzatori ognuno con il suo pulsante di arresto; oppure utilizzando lo stesso pulsante per tutti e tre seguendo poi un ordine di arresto, in modo che lo stesso temporizzatore generi tutti i numeri casuali che sono necessari. Secondo le applicazioni sceglieremo gli elementi di uscita. Potrebbe essere un display a 7 segmenti per visualizzare il numero fra 1 e 6 (o qualsiasi range fra 0 e 9) o potrebbe essere una matrice di LED, nel caso in cui stessimo giocando a tombola, anche se in questo caso bisognerebbe predisporre una routine ausiliaria per verificare che il numero non sia già uscito per non ripeterlo.



Organigramma per il troncamento di un valore per operazione successiva.



Dispositivi per simulare un dado elettronico.