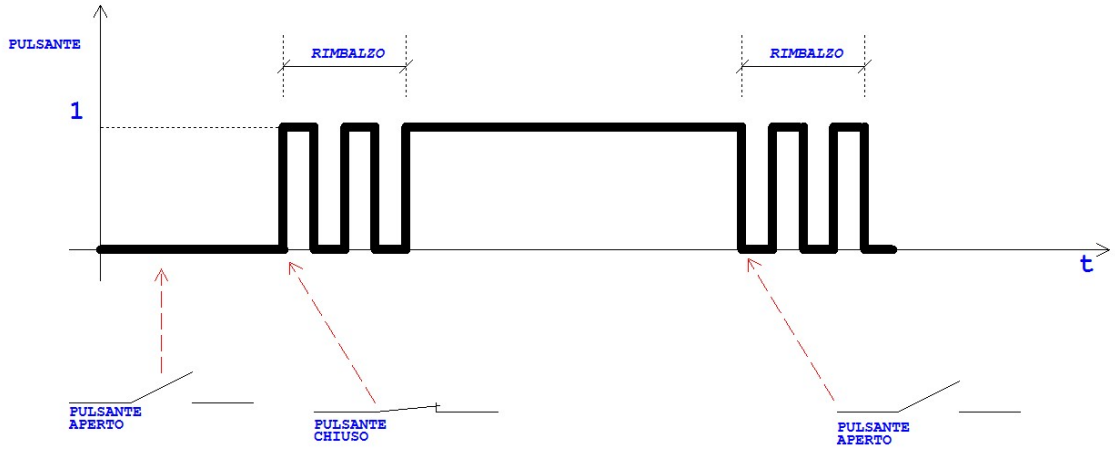
DEBOUNCING – ANTIRIMBALZO

Quando colleghiamo ad un sistema elettronico un contatto come ad esempio un pulsante, bisogna considerare un problema invisibile che invece può creare diversi problemi. Parliamo del cosiddetto FLICKERING, cioè la vibrazione del segnale elettrico nel passaggio dal livello 0 al livello 1 logico (o di tensione da 0 a 5 Volt ad esempio) e viceversa, come rappresentato in figura.



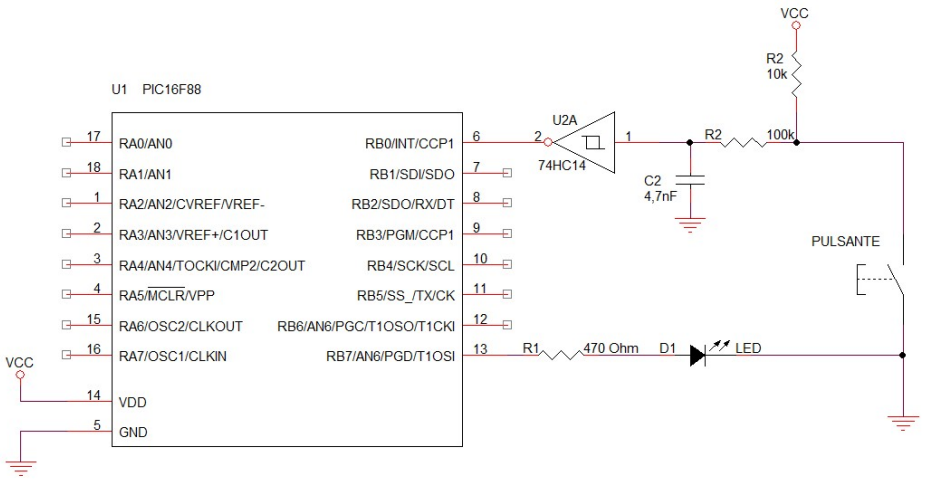
Questo problema è dovuto al fatto che nei contatti elettromeccanici (come nel caso di un pulsante) la lamina metallica del contatto, avvicinandosi per effettuare la chiusura, non garantisce un'immediato passaggio da 0 a 1, per motivi dovuti ai micro rimbalzi, o all'ossidazione della lamina che non garantisce un contatto perfettamente pulito.

Questo problema tradotto a livello elettrico, causa una variazione della tensione prima di stabilizzarsi al valore dato dal contatto aperto o chiuso. Questo effetto viene anche chiamato RIMBALZO, e può avvenire sia alla pressione che al rilascio del pulsante.

E' facilmente intuibile che questo rimbalzo del segnale potrebbe causare qualche problema, pensiamo ad esempio cosa accadrebbe se il nostro programma dovesse contare le pressioni di un pulsante.

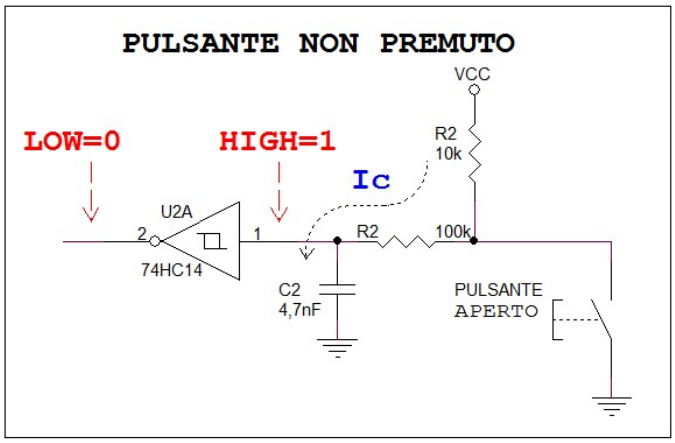
In un sistema a microcontrollore, data la rapidità con cui la CPU lavora e testa gli ingressi, il rimbalzo verrebbe letto interamente, acquisendo sia il valore 0 che 1 più volte.

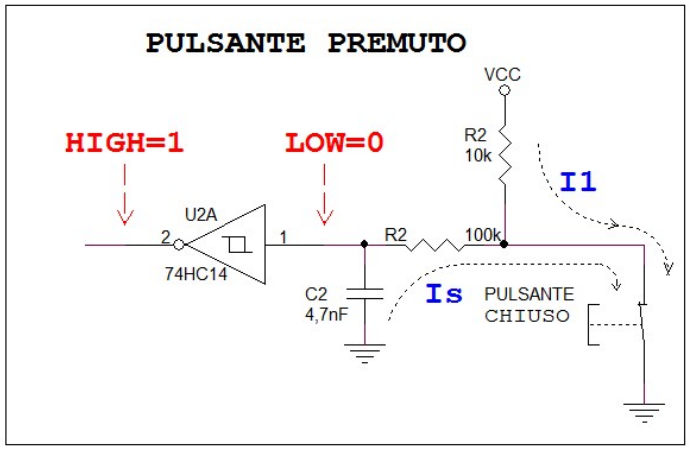
Il problema viene in genere risolto in maniera hardware con dei circuiti antirimbalzo, come ad esempio quello della figura seguente, dove abbiamo un microcontrollore collegato ad un pulsante ed ad un led.



ANTIRIMBALZO HARDWARE (HARDWARE DEBOUNCE)

Nel circuito della figura precedente, il pulsante viene collegato al microcontrollore, passando per un circuito con una porta logica NOT triggerata, ed una rete RC.

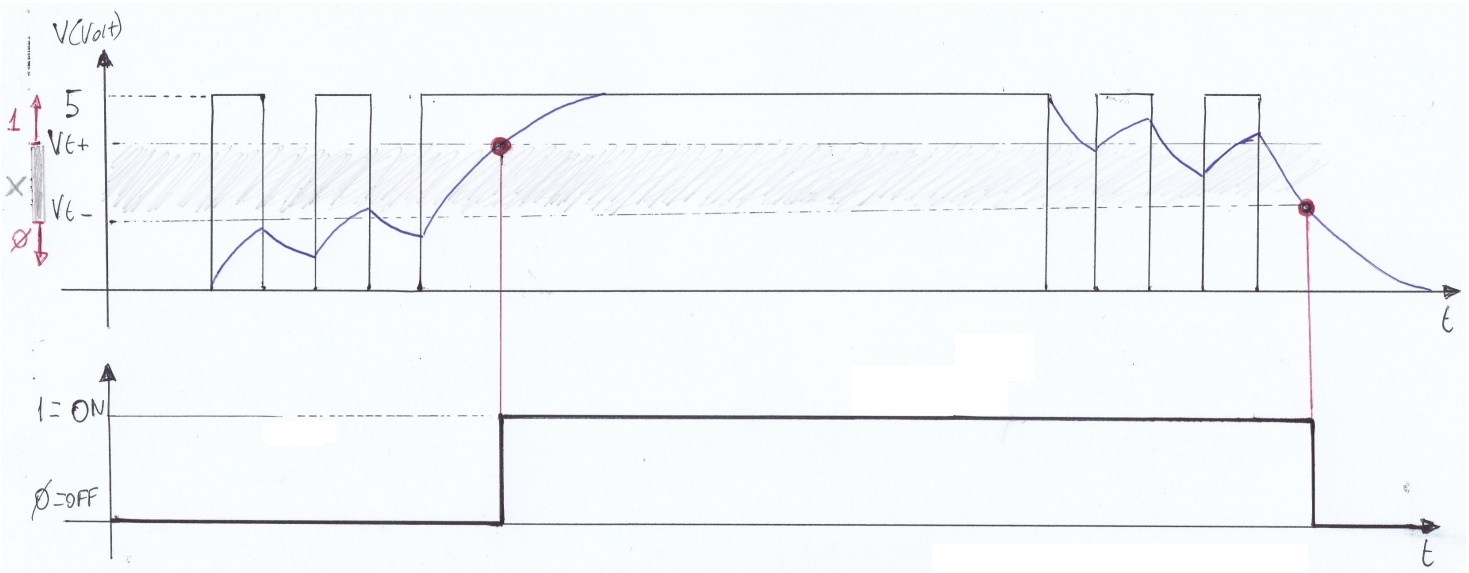




Il funzionamento di questo circuito è riassunto nelle due immagini sopra, in pratica quando il pulsante non è premuto, il condensatore si carica mediante le due resistenze, perciò con una costante di tempo dipendente dalla somma delle due resistene (110k) e dal valore della capacità 4,7nF. Alla pressione del pulsante, il condensatore invece si scarica con una costante di tempo dovuta alla sola resistenza R2 (100k) ed ovviamente allo stesso valore di capacità. Pertanto la carica e scarica sarà all'incirca uguale.

Comunque sia alla pressione ed al rilascio del pulsante sulla porta logica NOT, ci sarà sempre un andamento continuo del segnale elettrico in salita o in discesa che segue l'andamento di carica/scarica del condensatore.

Eventuali vibrazioni dovute al contatto, vengono "rallentate" dalla presenza della carica/scarica, inoltre essendoci una porta NOT con trigger (cioè una porta che ha due livelli logici differenti per considerare l'ingresso come alto o come basso) queste vibrazioni non verranno rilevate dal microcontrollore.



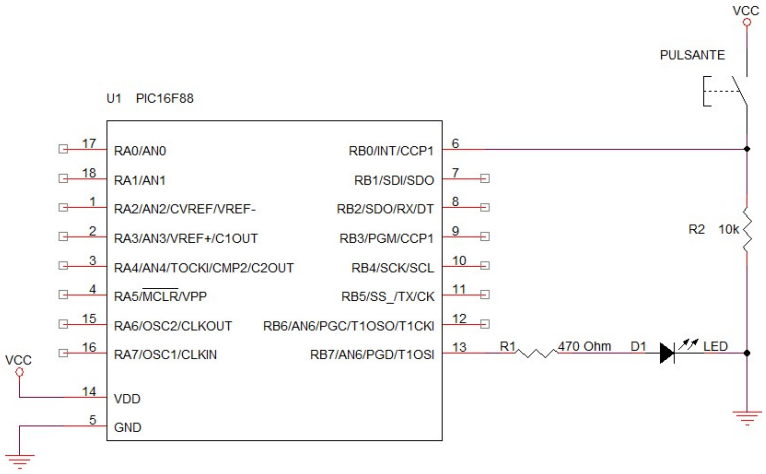
Come possiamo notare nell'immagine, il segnale fino a che non supera la soglia Vt+, viene considerato come un livello basso (0) e fino a quando non scende sotto alla soglia Vt- viene considerato come un livello alto (1).

Le oscillazioni del segnale in ingresso al circuito non influiranno sullo stato dell'uscita.

ANTIRIMBALZO SOFTWARE (SOFTWARE DEBOUNCE)

Il circuito hardware è sempre un ottimo metodo per ridurre l'effetto del rimbalzo di un contatto, ma presenta due problemi; la complessità del circuito elettronico, e la poca flessibilità dovuta ai valori fissi dei componenti. Sicuramente è sempre conveniente avere un antirimbalzo hardware, ma è sempre utile aggiungere ad esso un controllo software, anche se per economicizzare il circuito potremmo anche utilizzare solo il controllo software del segnale, realizzando comunque una valida soluzione antirimbalzo.

Consideriamo un circuito semplificato come quello in figura:
Supponiamo di voler realizzare un programma che ad ogni pressione del tasto venga eseguita una semplice funzione che è quella di invertire lo stato del led.



In questo caso per eliminare il problema del flickering del pulsante, potremmo adottare una soluzione diversa da quella hardware, ma una soluzione prettamente software, che è quella di verificare che il pulsante risulti premuto per un tempo consecutivo da definire in base alle necessità.
Per realizzare questa soluzione il microcontrollore, dovrebbe testare periodicamente l'ingresso dove è collegato il pulsante, ad esempio 30 volte come nell'esempio successivo.
Se il pulsante viene testato ogni millisecondo, dopo 30 test (cioè 30msec) se il pulsante risulta sempre attivo (ingresso ad 1) allora il microcontrollore identifica il pulsante come sicuramente premuto.
Ovviamente il tempo di scansione dell'ingresso non dovrebbe essere molto alto, in quanto dobbiamo sempre ricordare che durante il delay l'ingresso non viene testato, pertanto non ci accorgeremmo di eventuali variazioni.

Nei vari sistemi di sviluppo, esistono delle librerie che eseguono quanto detto sopra, come ad esempio nell'ambiente di sviluppo mikroC dove troviamo la libreria ***button*** con la relativa funzione.

Button

Prototype	`unsigned short Button(unsigned short *port, unsigned short pin, unsigned short time, unsigned short active_state);`
Returns	▪ 255 if the pin was in the active state for given period. ▪ 0 otherwise

Ad esempio scrivendo: `if (Button(&PORTB, 0, 1, 1))` *si testa il bit 0 di PORTB, che deve essere attivo per un tempo di 1msec, considerandolo attivo quando è ad 1. Il risultato della funzione è 255 se il bit è stato ad 1 per il tempo di 1msec.*

Un'altra soluzione valida invece in ogni caso, è quella proposta nel seguente codice dove possiamo realizzare un antirimbalzo software, che potrebbe anche essere sviluppato all'interno di una funzione.

ANTIRIMBALZO SOFTWARE

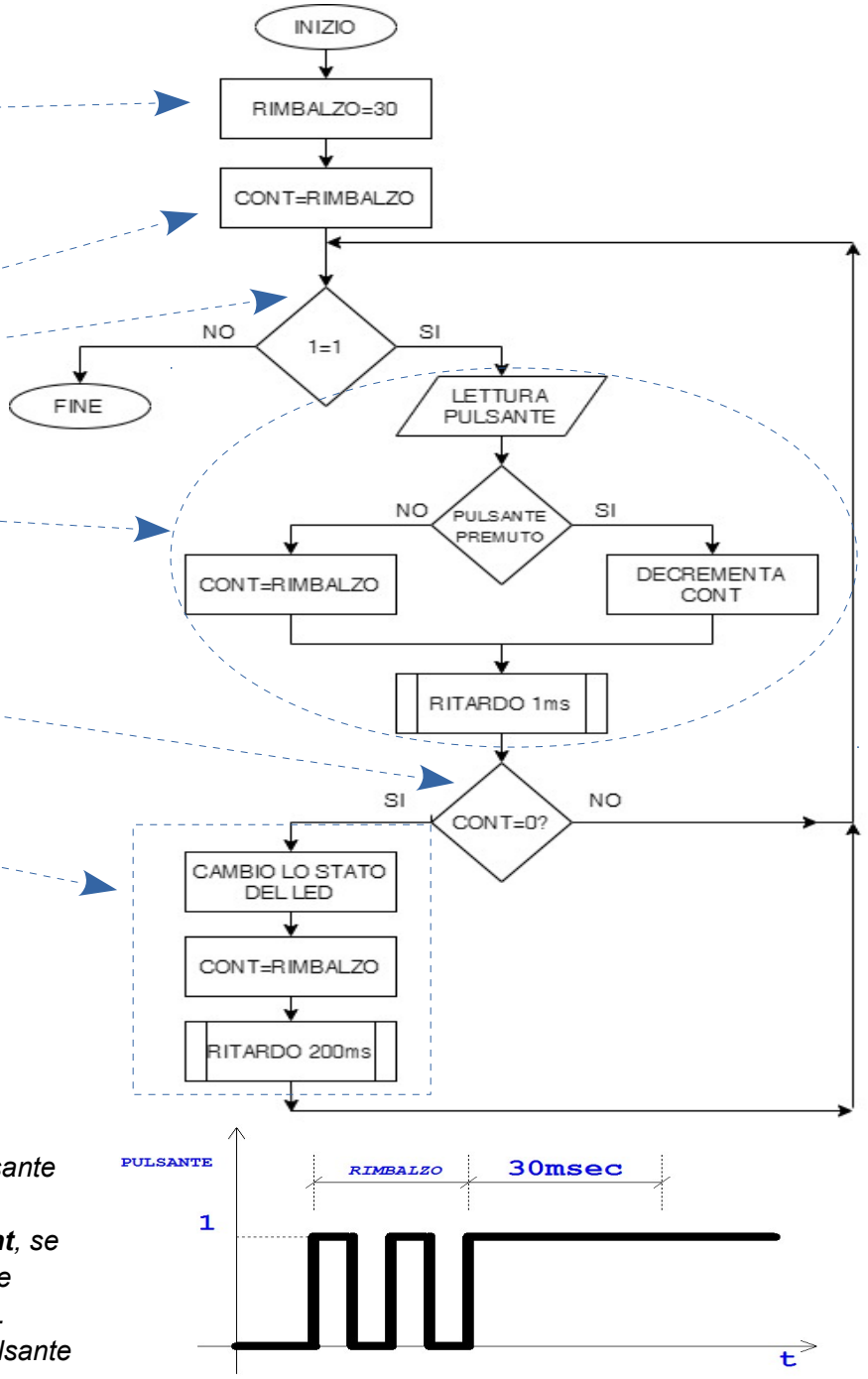
```
sbit PULSANTE at RB0_bit;    //assegno al bit 0 di PORTB l'etichetta PULSANTE
sbit LED      at RB7_bit;    //assegno al bit 0 di PORTB l'etichetta LED

const char rimbalzo=30;    //dichiarazione costante
char cont;                 //dichiarazione variabile

void main() {
 ANSEL=0;           //disattivo i convertitori
 CMCON=0x07;        //disattivo i comparatori
 OSCCON=0xFC;       //imposto un clock interno ad 8MHz
 TRISB.B0=1;        //imposto il bit 0 di PORTB come ingresso
 TRISB.B7=0;        //imposto il bit 7 di PORTB come uscita

 cont=rimbalzo;     //inizializzo il contatore
 while(1)           //loop infinito
  {
  if(PULSANTE==1)  cont--;         //decremento il contatore
  else             cont=rimbalzo; //inizializzo il contatore
  delay_ms(1);                    //ritardo di un millisecondo

  if(cont==0)
    {
     LED=!LED;        //se il cont=0 modifico lo stato del LED
     cont=rimbalzo;   //inizializzo il contatore
     delay_ms(200);   //ritardo di 200msec
    }//if(cont==0)
  }//while(1)
}//main
```



*Nell'esempio viene utilizzata una variabile di tipo char, **cont**.*
*La variabile viene inizializzata al valore memorizzato nella costante **rimbalzo**, che nell'esempio vale **30**.*

*Successivamente nel programma viene realizzato un loop infintio (ciclo while) dove al suo interno viene periodicamente testato l'ingresso del pulsante, ogni volta che l'ingresso vale **1**, viene decrementata la variabile **cont**, se invece l'ingresso vale **0**, la variabile viene inizializzata nuovamente al valore della costante rimbalzo.*
*Quando la variabile cont vale 0 significa che per **30** volte consecutive il pulsante è stato trovato al valore di **1**. Visto che il test avvene ogni msec, la variabile cont vale 0 se il pulsante rimane ad 1 per **30msec** consecutivi.*
*Per testare l'avvenuta pressione del pulsante pertanto basterà controllare il valore di **cont**, se essa vale 0, il pulsante è stato premuto in maniera consecutiva, senza perciò considerare eventuali rimbalzi. A questo punto testando il valore di cont, si cambierà lo stato del **LED**.*
*Successivamente **cont** va inizializzato nuovamente, e per dare il tempo di rilasciare il pulsante si può inserire un ritardo di **200msec**.*