GESTIONE DI UN MENU' SU DISPLAY LCD

Nel realizzare programmi per microcontrollori, in genere ci si trova ad affrontare delle problematiche spesso simili in progetti differenti, come ad esempio la gestione dell'antirimbalzo software, necessaria ogni qualvolta vengono utilizzati pulsanti o switch meccanici.

Considerando la gestione di una possibile interfaccia utente, spesso composta da pulsanti e display, una problematica riguarda la gestione di piccoli display dove visualizzare una quantità a volte elevata di informazioni.

Anche se oggi ormai si trovano display grafici di vario tipo a costi davvero molto contenuti, il buon vecchio display alfanumerico a cristalli liquidi può essere utilizzato in tutte le applicazioni dove non è indispensabile la grafica.

Bisogna considerare infatti che un display alfanumerico, oltre che essere facilmente reperibile ed avere ormai uno standard sulla parte software, elettronica e meccanica (dimensioni uguali tra le varie marche) non richiede molte risorse da parte del microcontrollore, pertanto può ancora essere utilizzato in diverse applicazioni.

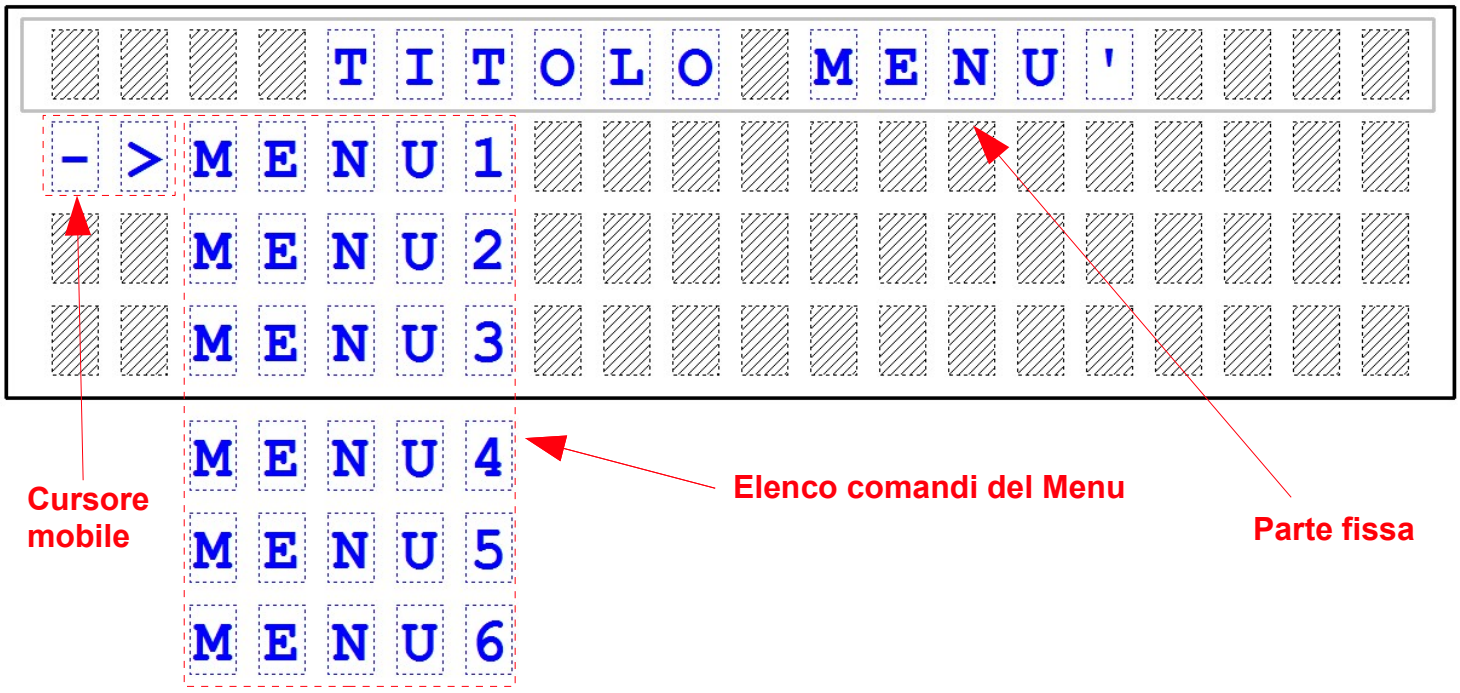
Il display alfanumerico però presenta lo svantaggio dei pochi caratteri che possiamo visualizzare, in quanto si trovano in commercio display, con 2/4 righe e 16/20/40 colonne.

Una delle configurazioni più usata oltre a quella 2 righe per 16 colonne, è quella da 4 righe con 20 colonne, come nel caso che vedremo nell'esempio.

Il problema che ho dovuto risolvere, è quello relativo alla gestione di un menù composto da più di 4 righe.

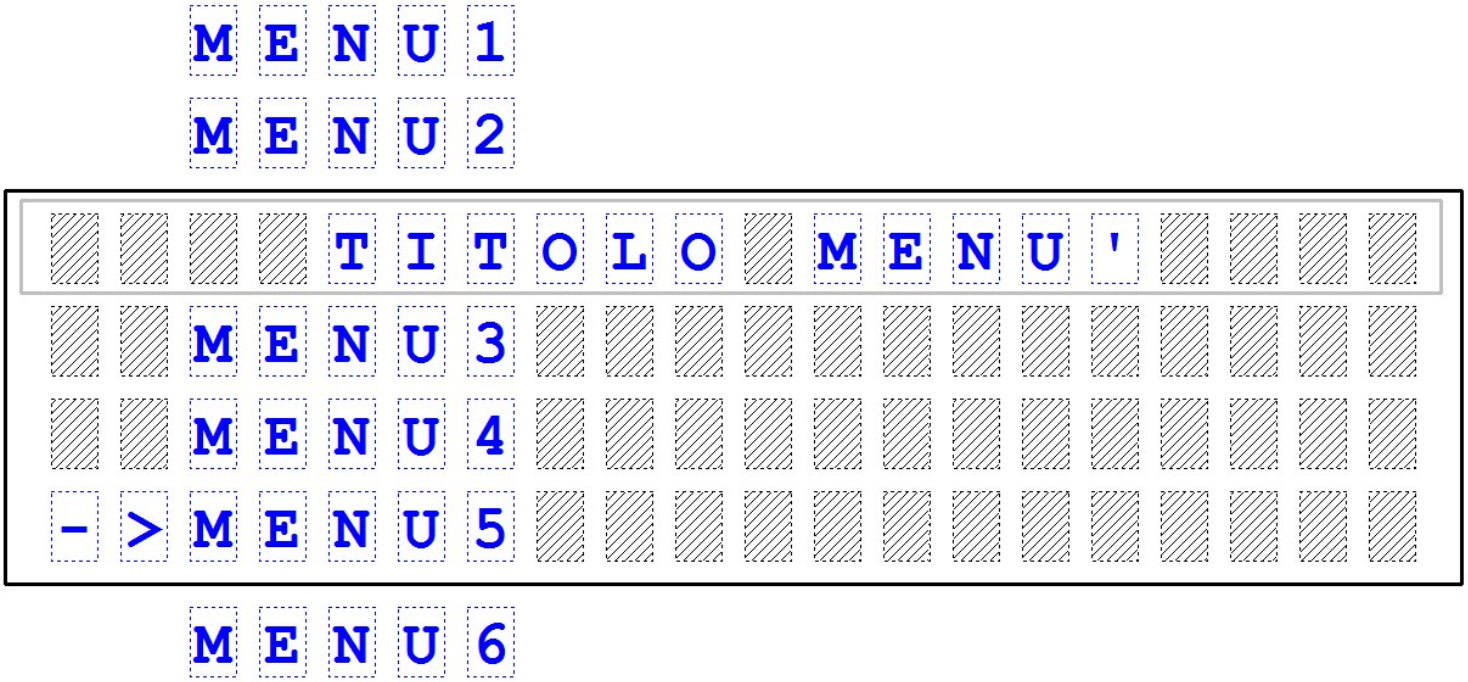
E se consideriamo che delle 4 righe disponibili sul display, la prima potrebbe essere occupata da un'eventuale intestazione o scritta fissa, rimarrebbero solo 3 righe oon cui possiamo rappresentare 3 voci di un menù.

Ma se avessimo pù di 3 voci come possiamo fare?



Nell'immagine vediamo ad esempio un display LCD 4 righe con 20 colonne, dove la prima riga viene utilizzata per visualizzare una scritta fissa, che rappresenta il titolo del menù, e le altre 3 righe invece visualizzano 3 righe del menù, lasciandone fuori altre 3 e cioè MENU 4, MENU 5 e MENU 6. Il cursore si potrà muovere in alto ed in basso tra la seconda e la quarta riga.

Il risultato che vorremmo ottenere è quello di poter spostare con dei pulsanti il cursore mobile in alto ed in basso, ed una volta che il cursore arriva tutto in basso si dovrà fermare non potendo andare oltre e dovrà essere l'elenco a spostarsi in alto, come nella seguente figura.

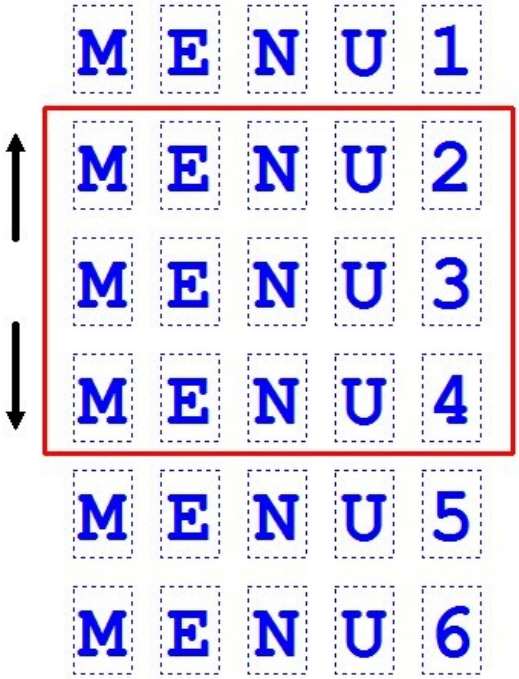


Analogamente se il cursore andrà in alto, una volta arrivato alla seconda riga, avremo la necessità anche in questo caso di spostare le voci del menù sul display.

E' un po' come se avessimo una finestra che scorre in alto ed in basso su un elenco di nomi.
Ma nel nostro caso non dobbiamo solo spostare la finestra, ma anche il cursore che ci serve per indicare quale delle 3 voci in elenco è selezionata.

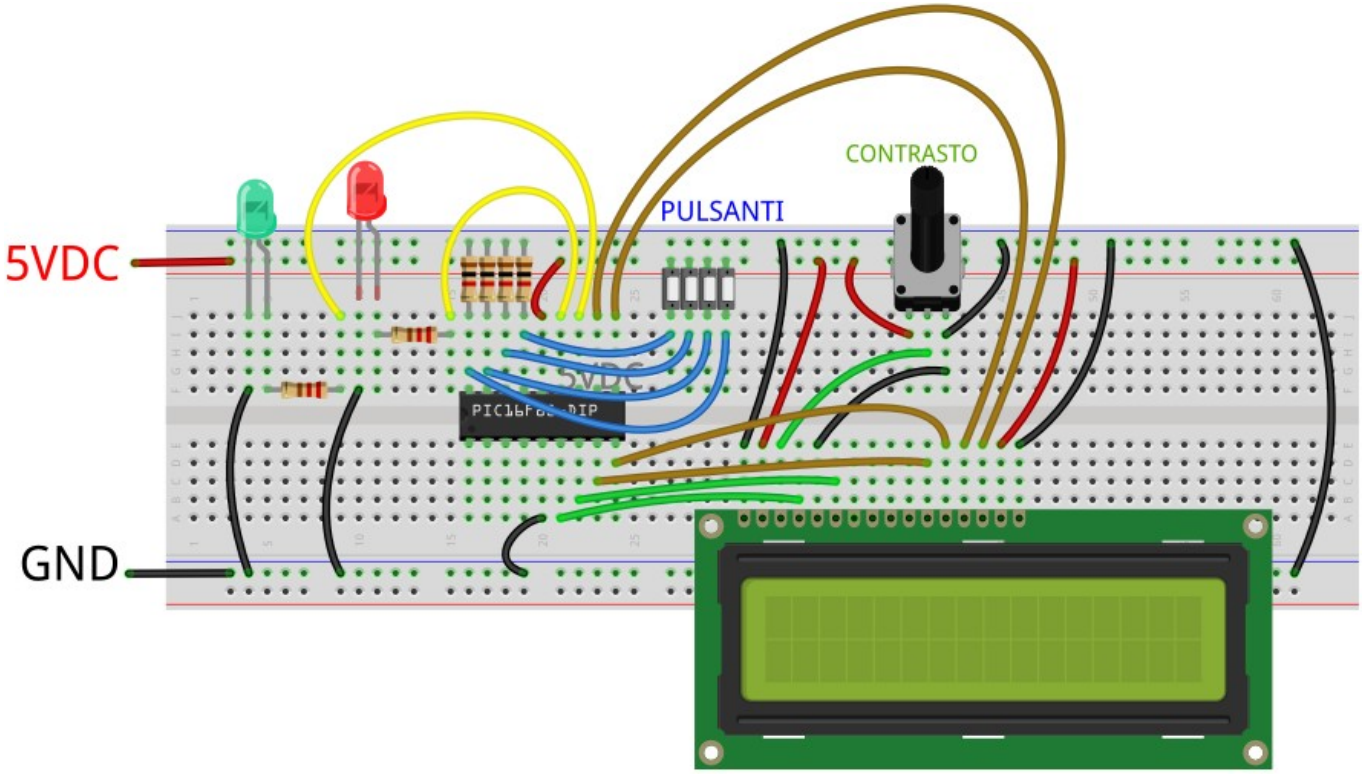
Oltre ai due tasti alto e basso, quando gestiamo un menù abbiamo bisogno di un tasto di conferma INVIO o ENTER, ed un tasto di uscita ESC.

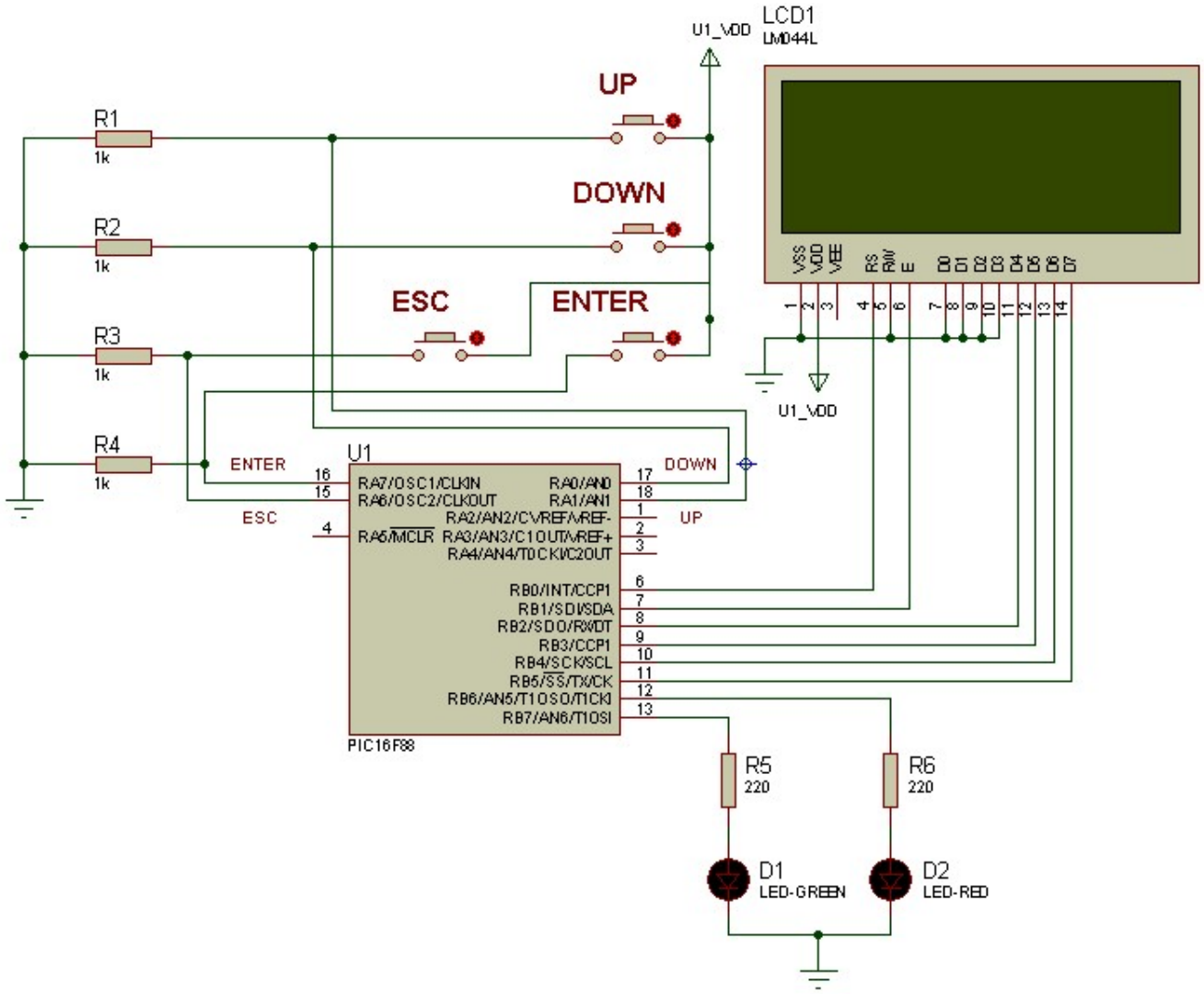
Perciò il minimo che possiamo avere sono 4 pulsanti, oltre ovviamente un display.



Innanzitutto realizziamo uno schema di esempio per provare questa gestione del display. Per comodità utilizzeremo il classico PIC16F88, anche se questo tipo di microcontrollore potrebbe essere limitato, in quanto la RAM interna è di soli 368 bytes.
Uno schema potrebbe essere il seguente, realizzato con Fritzing per quanto riguarda il montaggio su breadboard, e con Proteus Isis per quanto riguarda la simulazione, in questo secondo caso non è presente il trimmer per il controllo del contrasto del display LCD.

SCHEMA





Nello schema sopra abbiamo anche collegato due LED, uno rosso ed uno verde, per far eseguire delle funzioni ai vari comandi che inseriremmo nel menù.

Avendo due LED, possiamo immaginare di avere un menù composto da un elenco di 6 comandi, che possono ad esempio essere:

- ➔ ***Rosso ON***
- ➔ ***Verde ON***
- ➔ ***Tutti ON***
- ➔ ***Tutti OFF***
- ➔ ***Impulso Rosso***
- ➔ ***Impulso Verde***

Utilizziamo l'ambiente di sviluppo mikroC e la prima parte del codice sarà relativa alla definizione dei bit, assegniamo perciò un nome ad ogni PIN utilizzato dal microcontrollore.

```
// DEFINIZIONE BIT LCD
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D4 at RB2_bit;
sbit LCD_D5 at RB3_bit;
sbit LCD_D6 at RB4_bit;
sbit LCD_D7 at RB5_bit;

sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D7_Direction at TRISB5_bit;

sbit LED_ROSSO at RB6_bit;
sbit LED_VERDE at RB7_bit;

sbit PULS_UP    at RA1_bit;
sbit PULS_DOWN  at RA0_bit;
sbit PULS_ENTER at RA7_bit;
sbit PULS_ESC   at RA6_bit;
```

I pin del display LCD, dovranno essere chiamati come in figura, come richiesto dalla libreria che utilizzeremo.

I restanti pin potranno invece essere chiamati anche in maniera differente.

Nell'ambiente di sviluppo mikroC, le funzioni fornite dalla libreria per gestire il display sono le seguenti:

Lcd_Init()	*inizializza il microcontrollore a seconda dei piedini definiti con la direttiva* ***sbit***	
Lcd_Cmd(_LCD_CLEAR)	*cancella l'intero display*	
Lcd_Cmd(_LCD_CURSOR_OFF)	*spegne il cursore del display*	
Lcd_Chr(riga,colonna,carattere)	*visualizza dalla riga e colonna indicata un carattere ,*	***esempio 'A'***
Lcd_Chr_Cp(carattere)	*visualizza nell'attuale posizione del cursore un carattere,*	***esempio 'A'***
Lcd_Out(riga,colonna,stringa)	*visualizza dalla riga e colonna indicata una stringa,*	***esempio "hello"***
Lcd_Out(stringa)	*visualizza nell'attuale posizione del cursore una stringa,*	***esempio"hello"***

Altri comandi sono disponibili nell'help del software mikroC.

Dovremo costruirci anche un'array contenente l'elenco dei comandi del menù e dichiarare le variabili necessarie al programma.

```
char menu_imp[6][9] =   //Menu da gestire
{
{'R','o','s','s','o',' ','O','N',' '},
{'V','e','r','d','e',' ','O','N',' '},
{'T','u','t','t','i',' ','O','N',' '},
{'T','u','t','t','i',' ','O','F','F'},
{'I','M','P','.','R','o','s','s','o'},
{'I','M','P','.','V','e','r','d','e'},
};

char *txt_freccia="->";   //tipo cursore
char *txt_vuoto_2="  ";

char cont_rimbalzo=30;
char esito_menu;
```

Le voci dell'elenco del menù possono essere di lunghezze differenti, ma le dimensioni dell'array dovranno essere le stesse, in quanto risulta più comodo per effettuare la scrittura sullo schermo.
La funzione che utilizzeremo per scrivere queste voci, potrebbe essere strutturata nel seguente modo.

```
//-----------------------------------------------------------------------
//SCRITTURA VALORE NEL VETTORE MEMORIA_LCD E SU LCD
//scrittura eseguita un carattere alla volta partendo dalla riga/colonna
//occorre inviare anche la lunghezza e la stringa
void scrivi_display(char riga, char colonna, char lunghezza, char *testo)
{
  unsigned short cont_scrivi,cont_carattere=0;

  cont_scrivi=colonna;
  while(lunghezza)
   {
    Lcd_Chr(riga,cont_scrivi,testo[cont_carattere]);
    cont_scrivi++;
    cont_carattere++;
    lunghezza--;
   }//while(lunghezza)
}//scrivi_display()
```

Alla funzione, andrà passata la posizione dove cominciare a scrivere sottoforma di riga e colonna, la lunghezza del testo, e cioè la dimensione dell'array che nel nostro caso è di 9 caratteri, ed il testo da scrivere, nel nostro caso verrà inviata l'intera riga che si vuol scrivere.
Questa funzione scriverà nel display una riga per volta, dovrà essere chiamata pertanto 3 volte, essendo 3 le righe disponibili sul display (non dimentichiamoci che la prima riga viene utilizzata per rappresentare il titolo del menù ed è pertanto una scritta fissa).

Andiamo ora a vedere come può essere realizzata la gestione del menù, e questa parte di codice conviene sviulppara all'interno di una funzione.

La funzione, che chiameremo **"menu_principale**", dovrà consentire lo spostamento del cursore in alto ed in basso, ed la successiva modifica delle voci del menù visualizzate qualora il cursore arrivasse tutto in alto o tutto in basso. Oltre ai pulsanti per lo spostamento del cursore (PULS_UP e PULS_DOWN) dovremo gestire anche i pulsanti di invio (PULS_ENTER) e di uscita (PULS_ESC). Nello specifico premendo il pulsante invio, la funzione restituirà un numero da 1 a 6 a seconda della riga in quel momento evidenziata dal cursore, invece premendo il pulsante ESC, verrà restitutito il valore 0.

```
//MENU' PRINCIPALE
//ritorna 0 se viene premuto il pulsante ESC
//oppure il numero relativo alla riga scelta (1-6)
char menu_principale()
{
   char cont_down,cont_up,cont_enter,cont_esc;
   unsigned short flag_menu=1,flag_cursore=1,flag_cambio=1;
   unsigned short indice;
   //inizializzazione variabili per antirimbalzo pulsanti
   cont_down=cont_rimbalzo;
   cont_up=cont_rimbalzo;
   cont_enter=cont_rimbalzo;
   cont_esc=cont_rimbalzo;
   //inizializzazione display LCD
   Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out(1,4,"MENU PRINCIPALE");
```

Nella precedente immagine possiamo vedere la prima parte di questa funzione, dove troviamo la dichiarazione delle variabili locali, e la loro inizializzazione. Inoltre all'inizio di questa funzione dovremo cancellare il display, e visualizzare nella prima riga, la scritta fissa che rappresenta il titolo del menù e cioè "MENU PRINCIPALE".

Successivamente dovremo inserire un loop continuo, realizzato con un **while(1)**.

All'interno di questo ciclo, dovremo testare ogni intervallo di 1millisecondo, i 4 pulsanti. Per evitare il fenomeno del rimbalzo, inoltre consideriamo il pulsante attivo se rimane premuto per un numero di volte consecutive indicato nella variabile **cont_rimbalzo**.

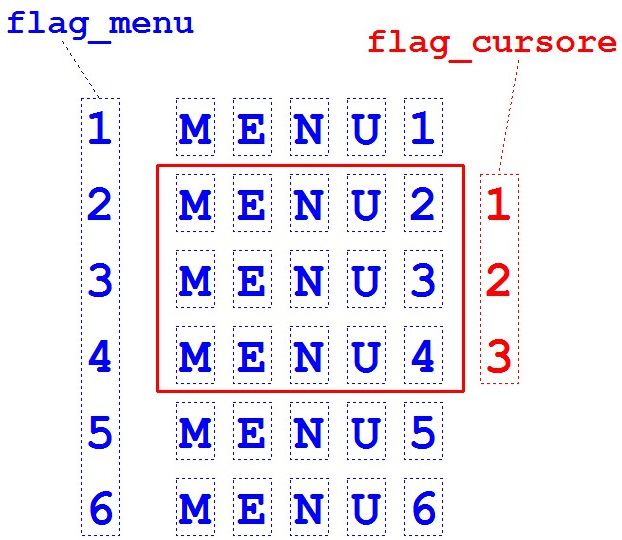
```
   while(1)  //loop continuo
    {
    //test pulsanti ----------------------------------------
    //se il pulsante è ad 1 si decremente il relativo contatore
    //se invece è a 0 il contatore viene inizializzato
    if(PULS_ENTER==1) cont_enter--;
    else              cont_enter=cont_rimbalzo;
    if(PULS_ESC==1)   cont_esc--;
    else              cont_esc=cont_rimbalzo;
    if(PULS_UP==1)    cont_up--;
    else              cont_up=cont_rimbalzo;
    if(PULS_DOWN==1)  cont_down--;
    else              cont_down=cont_rimbalzo;
    delay_ms(1);      //tempo per antirimbalzo
```

Successivamente controllando le variabili **cont** relative ad ogni tasto, abbiamo la certezza che il tasto sia stato rimasto effettivamente premuto per tutta la durata.
Ad esempio **cont_up** è associata a PULS_UP, se il valore letto è 1 per 30 volte consecutive (cont_rimblazo=30), allora **cont_up** diventerà 0 e potremo considerare il pulsante premuto.

```
//test dei contaori dei pulsanti, se il pulsante è attivo
//per un tempo pari a 30msec, il contatore del pulsante va a zero
//PULSANTE UP  ----------------------------------------
if(cont_up==0)
 {
  if(flag_menu>1)       flag_menu--;
  if(flag_cursore>1)    flag_cursore--;
  flag_cambio=1;
  cont_up=cont_rimbalzo;
 }//if(cont_up==0)
//PULSANTE DOWN  --------------------------------------
if(cont_down==0)
 {
  if(flag_menu<6)       flag_menu++;
  if(flag_cursore<3)    flag_cursore++;
  flag_cambio=1;
  cont_down=cont_rimbalzo;
 }//(cont_down==0)
//PULSANTE ENTER  -------------------------------------
```

Ora analizziamo cosa avviene quando viene premuto il pulsante UP o DOWN, guardando il codice sopra rappresentato. Alla pressione del pulsante UP, vengono decrementate due variabili, **flag_menu** e **flag_cursore**.

La prima rappresenta la riga del menu attiva, che nel nostro caso può andare da 1 a 6, la seconda variabile invece rappresenta la riga della finestra di visualizzazione vista prima.
Le due variabili possono essere decrementate solo se maggiori di 1, che rappresenta il loro valore minimo.



Se invece viene premuto il pulsante DOWN, le due variabili vengono incrementate, anche in questo caso fino al loro valore massimo.

In entrambi i casi, la variabile **cont**, associata ai due pulsanti viene inizializzata nuovamente al valore di **cont_rimbalzo** per la successiva pressione.

Alla pressione dei due pulsanti viene inoltre messa ad uno la variabile **flag_cambio**. Questa variabile segnala una modifica sul display, in quanto è stato premuto un pulsante, la modifica potrà essere lo spostamento del solo cursore, o la modifica delle voci dell'elenco del menu se il cursore è arrivato in alto o in basso, pertanto dovremo testare questa variabile per sapere se dobbiamo modificare la visualizzazione sul display.

Se viene premuto il tasto ENTER o ESC, usciremo dalla funzione con il **return**.

Se viene premuto il tasto ENTER, basterà restituire il valore della variabile **flag_menu**, che abbiamo detto prima, essa infatti contiene la riga del menù scelta, pertanto un valore che va da 1 a 6.

Invece di effettuare degli IF consecutivi, nel nostro caso è stata utilizzata la struttura SWITCH-CASE sulla variabile **flag_menu**. Se invece viene premuto il tasto ESC, viene restitutito il valore 0.

```
if(cont_enter==0)
 {
  cont_enter=cont_rimbalzo;
  switch(flag_menu)
   {
    case 1: return(1);
    case 2: return(2);
    case 3: return(3);
    case 4: return(4);
    case 5: return(5);
    case 6: return(6);
   }//switch(flag_menu)
 }//if(cont_enter==0)
//PULSANTE ESC  ----------------------------------------
if(cont_esc==0) return(0);
```

Come detto prima, dovremo controllare la variabile **flag_cambio**, per sapere se occorre modificare quello che viene visualizzato sul display, e questo avviene quando viene premuto il tasto UP o DOWN.

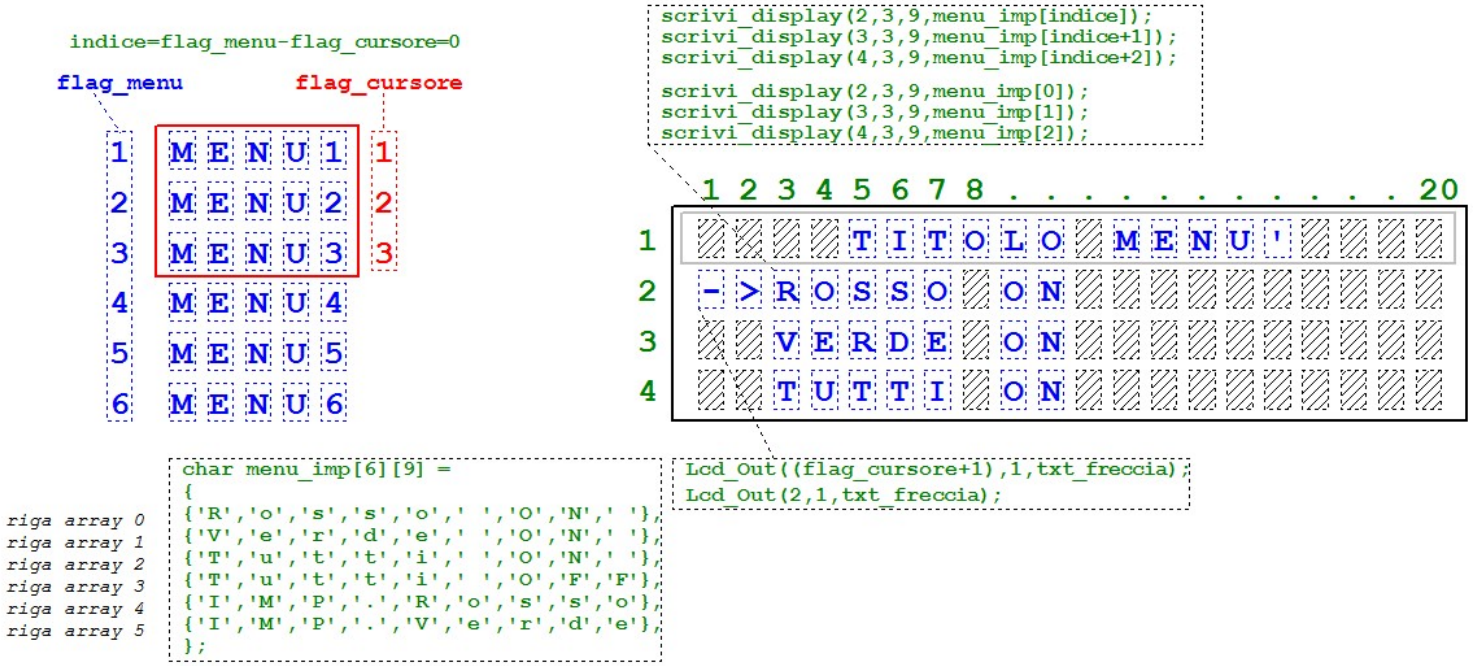
```
//modifica della visualizzazione LCD  --------------------
//se viene premuto PULS_UP o PULS_DOWN, flag_cambio=1
//viene pertanto modificata la vista delle voci sul display
if(flag_cambio==1)
 {
  indice=flag_menu-flag_cursore;
  Lcd_Out(2,1,txt_vuoto_2);
  Lcd_Out(3,1,txt_vuoto_2);
  Lcd_Out(4,1,txt_vuoto_2);
  Lcd_Out((flag_cursore+1),1,txt_freccia);
  scrivi_display(2,3,9,menu_imp[indice]);
  scrivi_display(3,3,9,menu_imp[indice+1]);
  scrivi_display(4,3,9,menu_imp[indice+2]);
  flag_cambio=0;
  //attesa rilascio pulsante
  while((PULS_UP==1)||(PULS_DOWN==1)) {}
 }//if(flag_cambio==1)
}//while(1)
}//menu_principale()
```

In questa parte di codice, viene innanzitutto calcolato il valore di una variabile chiamata **indice,** ottenuta come differenza tra **flag_menu** e **flag_cursore**. Poi viene cancellato il cursore, scrivendo due carattere vuoti nelle 3 posizioni possibili del cursore e subito dopo viene visualizzato il cursore nella posizione definita dalla variabile **flag_cursore**. Infine viene richiamata la funzione **scrivi_display** per 3 volte (una per ogni riga) utilizzando il valore della variabile **indice** calcolata prima, per definire quali voci del menù visualizzare.

Al termine della funzione viene reinizializzato il valore della variabile **flag_cambio** a 0 e con un ciclo **while** si attende il rilascio del pulsante premuto. Ma per capire bene il funzionamento di questa parte, che è il cuore della gestione dello scorrimento del menù, abbiamo bisogno di qualche immagine.

Cominciamo dalla condizione iniziale, dove **flag_cursore** e **flag_menu** valgono 1.

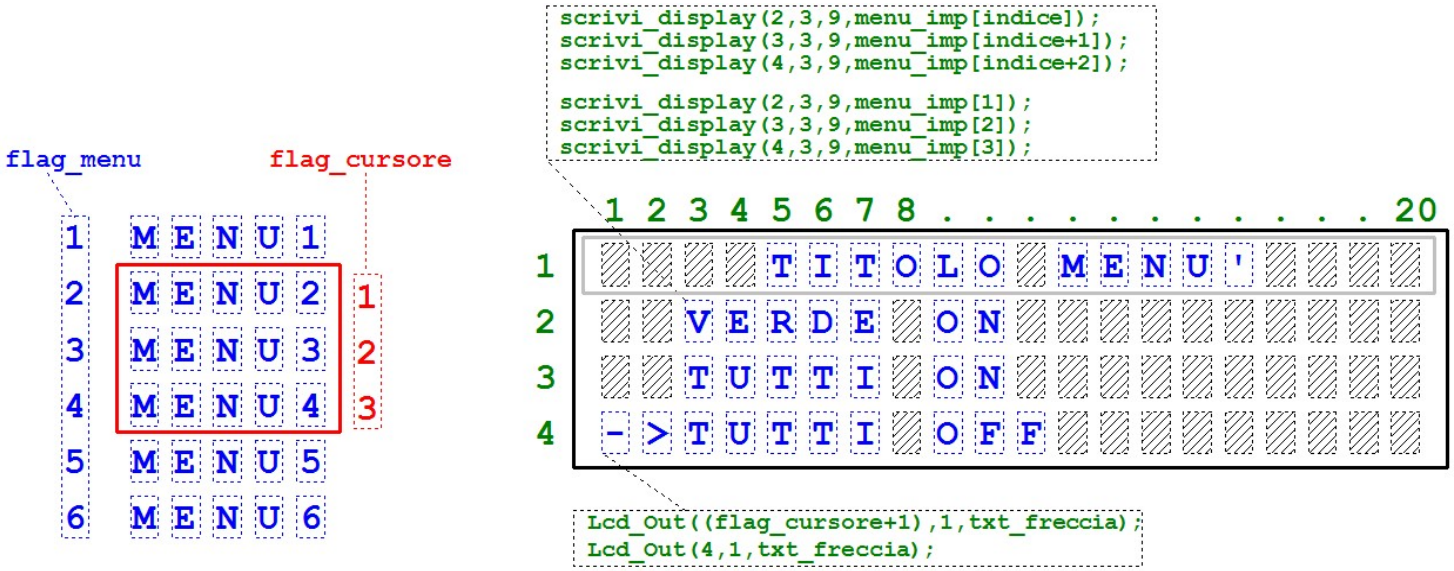
CONDIZIONE INIZIALE



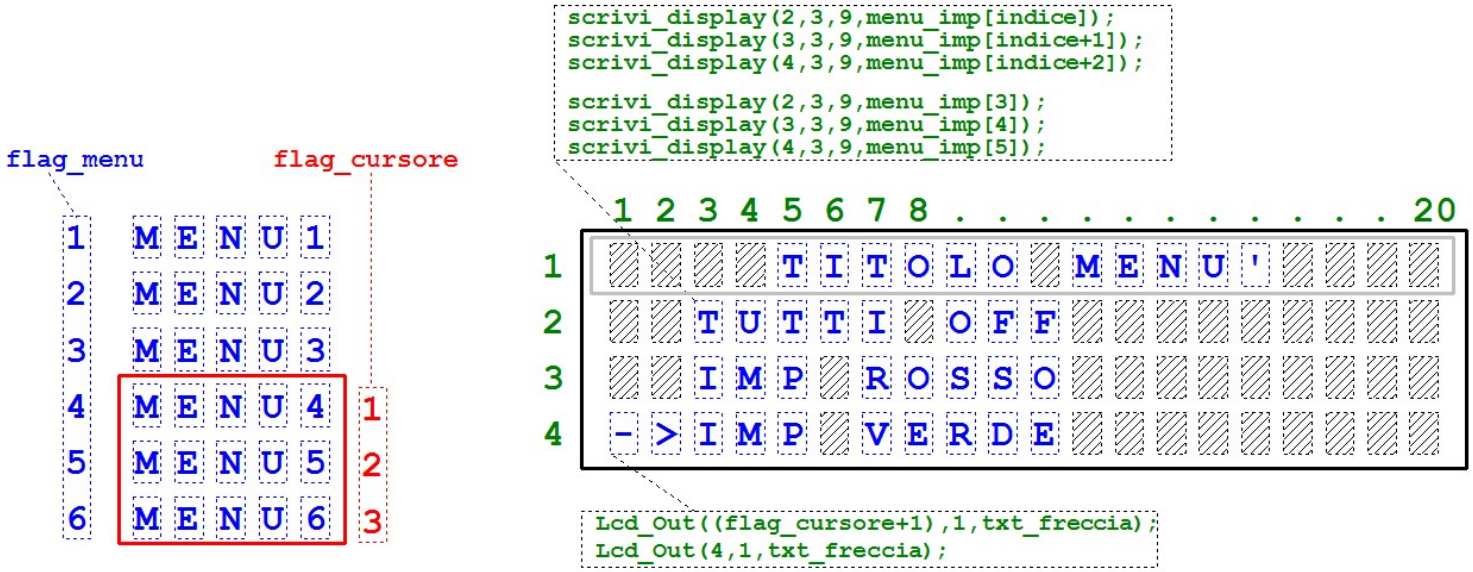
Come si può vedere nell'immagine sopra, la variabile **indice** vale 0, e verrà visualizzato il cursore nella riga 2 del display in quanto sull'istruzione **Lcd_Out** il primo valore è pari a **flag_cursore+1**, cioè 2.

Con la funzione scrivi_display invece visualizzeremo le prime 3 voci, in quanto l'array comincia con il valore 0 e la variabile **indice** rappresenta proprio l'indice dell'array.

Supponiamo ora di premere per 3 volte il tasto DOWN, avremo allora che **flag_cursore** vale 3 e **flag_menu** vale 4, perciò **indice** varrà 1, ciò significa che dovremo visualizzare l'array dalla riga 1 in poi, e che dovremo mettere il cursore sulla riga 4 dl display.



Supponiamo ora di premere più volte il tasto DOWN, fino a raggiungere il massimo valore di **flag_menu,** cioè 6, **flag_cursore** vale sempre 3, perché come abbiamo già detto è il suo valore massimo, perciò **indice** varrà 3, ciò significa che dovremo visualizzare l'array dalla riga 3 in poi, e che dovremo lasciare sempre il cursore sulla riga 4 del display **(flag_cursore+1)**.



Potremmo continuare, anche spostandoci in alto, il funzionamento sarebbe analogo, pertanto le 3 variabili assumono il seguente significato:

- **flag_menu** *rappresenta la voce del menù selezionata da 1 a 6*
- **flag_cursore** *rappresenta la riga nell'area di visualizzazione, che incrementata di 1 rappresenta la riga del display*
- **indice** *rappresenta l'indice della riga dell'array da visualizzare.*

Ovviamente il ragionamento può essere esteso ad un elenco di dimensioni maggiori, senza alcuna limitazione.

Nel main del programma, c'è un loop infinito che richiama la funzione menu_principale, e riceve come risposta un valore di 0 se viene premuto ESC o un valore da da 1 a 6 se viene premuto ENTER in base alla riga selezionata.

Nel ciclo main, inizialmente c'è la solita configurazione degli ingressi ed uscite, e succesivamente in base al ritorno del valore, c'è l'esecuzione del comando scelto nel menù, valutata in una struttura SWITCH-CASE.

```
void main()
{
 CMCON=0x07;       //disabilito i comparatori
 ANSEL=0x00;       //disabilito i convertitori
 OSCCON=0xFC;      //CLOCK interno 8MHz
 TRISA=0xFF;       //definizione in/out PORTA
 TRISB=0x00;       //definizione in/out PORTB

 //inizializzazione display LCD
 Lcd_Init();
 Lcd_Cmd(_LCD_CLEAR);
 Lcd_Cmd(_LCD_CURSOR_OFF);
 LED_ROSSO=0;
 LED_VERDE=0;
```

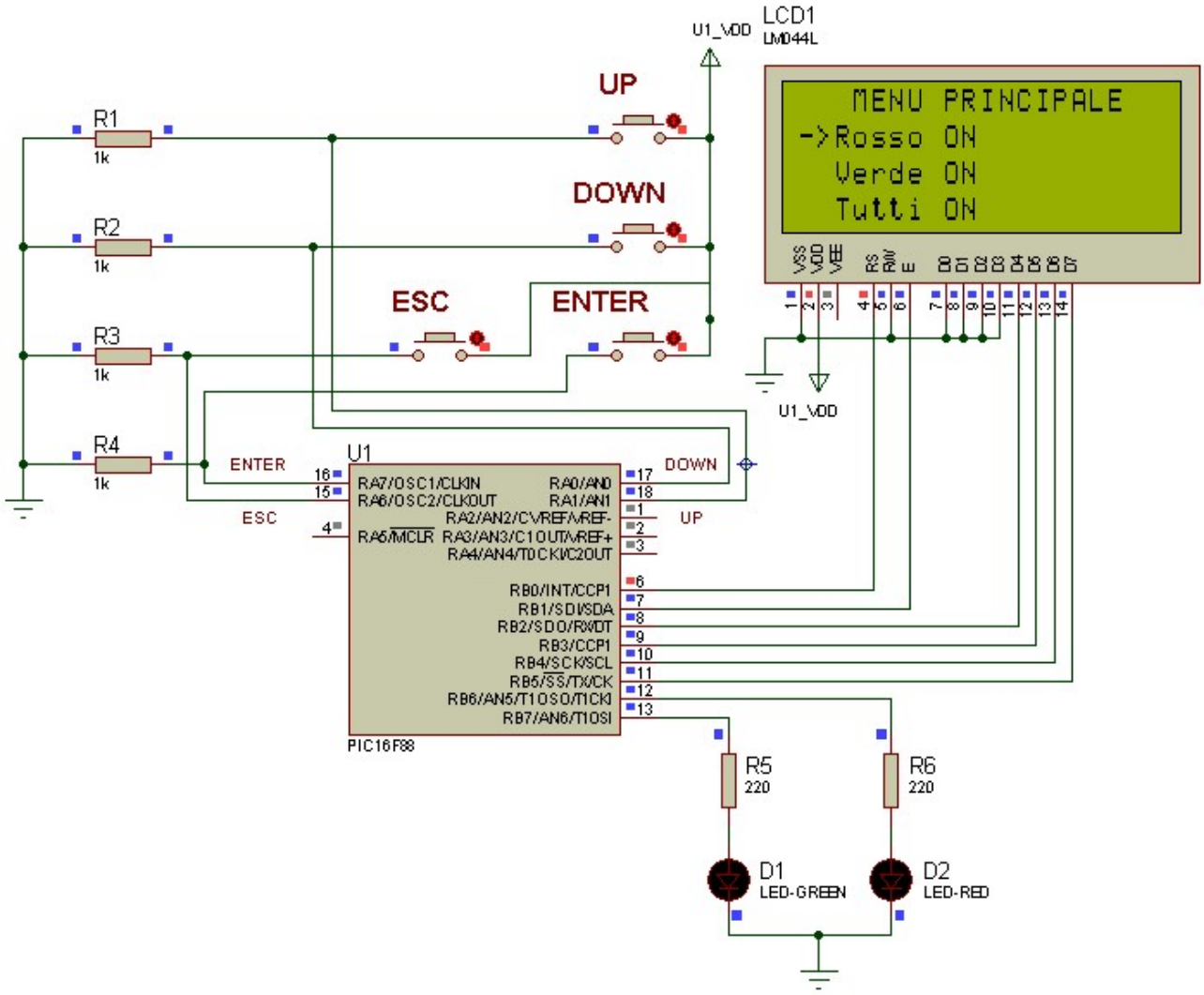
```
while(1)  //loop continuo ---------------------------------------
 {

  esito_menu=menu_principale();

  Lcd_Cmd(_LCD_CLEAR);
  switch(esito_menu)   //analisi della risposta
   {
     case 0:
      {
        Lcd_Out(2,3,"NESSUNA SCELTA");
        break;
      }//case 1:
     case 1:
      {
        Lcd_Out(2,3,"ROSSO ON");
        LED_ROSSO=1;
        break;
      }//case 1:
     case 2:
      {
        Lcd_Out(2,3,"VERDE ON");
        LED_VERDE=1;
        break;
      }//case 2:
     case 3:
      {
        Lcd_Out(2,3,"TUTTI ON");
        LED_ROSSO=1;
        LED_VERDE=1;
        break;
      }//case 3:
     case 4:
      {
        Lcd_Out(2,3,"TUTTI OFF");
        LED_ROSSO=0;
        LED_VERDE=0;
        break;
      }//case 4:
     case 5:
      {
        Lcd_Out(2,3,"IMPULSO ROSSO");
        LED_ROSSO=0;
        delay_ms(500);
        LED_ROSSO=1;
        delay_ms(500);
        LED_ROSSO=0;
        break;
      }//case 5:
```

```
        case 6:
         {
            Lcd_Out(2,3,"IMPULSO VERDE");
            LED_VERDE=0;
            delay_ms(500);
            LED_VERDE=1;
            delay_ms(500);
            LED_VERDE=0;
            break;
         }//case 6:
      }//switch(esito_menu)
     delay_ms(1000);
   }//while(1)
}//main
```

Questo è solo un esempio, poi ovviamente il codice potrà variare in base al tipo di problema da risolvere, ma la logica di funzionamento rimane identica.

Al seguente link, è possibile vedere la simulazione ed il funzionamento del programma con il software Proteus.

Video della simulazione: https://youtu.be/RBZjo8Abfec



```
// DEFINIZIONE BIT LCD
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D4 at RB2_bit;
sbit LCD_D5 at RB3_bit;
sbit LCD_D6 at RB4_bit;
sbit LCD_D7 at RB5_bit;

sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D7_Direction at TRISB5_bit;

sbit LED_ROSSO at RB6_bit;
sbit LED_VERDE at RB7_bit;

sbit PULS_UP    at RA1_bit;
sbit PULS_DOWN  at RA0_bit;
sbit PULS_ENTER at RA7_bit;
sbit PULS_ESC   at RA6_bit;

char menu_imp[6][9] =   //Menu da gestire
{
{'R','o','s','s','o',' ','O','N',' '},
{'V','e','r','d','e',' ','O','N',' '},
{'T','u','t','t','i',' ','O','N',' '},
{'T','u','t','t','i',' ','O','F','F'},
{'I','M','P','.','R','o','s','s','o'},
{'I','M','P','.','V','e','r','d','e'},
};

char *txt_freccia="->";   //tipo cursore
char *txt_vuoto_2="  ";

char cont_rimbalzo=30;
char esito_menu;


//-----------------------------------------------------------------------
//SCRITTURA VALORE NEL VETTORE MEMORIA_LCD E SU LCD
//scrittura eseguita un carattere alla volta partendo dalla riga/colonna
//occorre inviare anche la lunghezza e la stringa
void scrivi_display(char riga, char colonna, char lunghezza, char *testo)
{
  unsigned short cont_scrivi,cont_carattere=0;

  cont_scrivi=colonna;
  while(lunghezza)
   {
     Lcd_Chr(riga,cont_scrivi,testo[cont_carattere]);
     cont_scrivi++;
     cont_carattere++;
     lunghezza--;
   }//while(lunghezza)
}//scrivi_display()


//-----------------------------------------------------------------------
//MENU' PRINCIPALE
//ritorna 0 se viene premuto il pulsante ESC
```

```
63: //oppure il numero relativo alla riga scelta (1-6)
64: char menu_principale()
65: {
66:    char cont_down,cont_up,cont_enter,cont_esc;
67:    unsigned short flag_menu=1,flag_cursore=1,flag_cambio=1;
68:    unsigned short indice;
69:    //inizializzazione variabili per antirimbalzo pulsanti
70:    cont_down=cont_rimbalzo;
71:    cont_up=cont_rimbalzo;
72:    cont_enter=cont_rimbalzo;
73:    cont_esc=cont_rimbalzo;
74:    //inizializzazione display LCD
75:    Lcd_Cmd(_LCD_CLEAR);
76:    Lcd_Out(1,4,"MENU PRINCIPALE");
77:
78:    while(1)  //loop continuo
79:     {
80:      //test pulsanti ----------------------------------------
81:      //se il pulsante è ad 1 si decremente il relativo contatore
82:      //se invece è a 0 il contatore viene inizializzato
83:      if(PULS_ENTER==1) cont_enter--;
84:      else              cont_enter=cont_rimbalzo;
85:      if(PULS_ESC==1)   cont_esc--;
86:      else              cont_esc=cont_rimbalzo;
87:      if(PULS_UP==1)    cont_up--;
88:      else              cont_up=cont_rimbalzo;
89:      if(PULS_DOWN==1)  cont_down--;
90:      else              cont_down=cont_rimbalzo;
91:      delay_ms(1);      //tempo per antirimbalzo
92:
93:      //test dei contaori dei pulsanti, se il pulsante è attivo
94:      //per un tempo pari a 30msec, il contatore del pulsante va a zero
95:      //PULSANTE UP  -----------------------------------------
96:      if(cont_up==0)
97:       {
98:        if(flag_menu>1)      flag_menu--;
99:        if(flag_cursore>1)   flag_cursore--;
100:       flag_cambio=1;
101:       cont_up=cont_rimbalzo;
102:      }//if(cont_up==0)
103:     //PULSANTE DOWN  ---------------------------------------
104:     if(cont_down==0)
105:      {
106:       if(flag_menu<6)      flag_menu++;
107:       if(flag_cursore<3)   flag_cursore++;
108:       flag_cambio=1;
109:       cont_down=cont_rimbalzo;
110:      }//(cont_down==0)
111:     //PULSANTE ENTER  --------------------------------------
112:     if(cont_enter==0)
113:      {
114:        cont_enter=cont_rimbalzo;
115:        switch(flag_menu)
116:         {
117:          case 1: return(1);
118:          case 2: return(2);
119:          case 3: return(3);
120:          case 4: return(4);
121:          case 5: return(5);
122:          case 6: return(6);
123:         }//switch(flag_menu)
124:      }//if(cont_enter==0)
```

```
125:      //PULSANTE ESC  ----------------------------------------
126:      if(cont_esc==0) return(0);
127:
128:      //modifica della visualizzazione LCD  --------------------
129:      //se viene premuto PULS_UP o PULS_DOWN, flag_cambio=1
130:      //viene pertanto modificata la vista delle voci sul display
131:      if(flag_cambio==1)
132:       {
133:         indice=flag_menu-flag_cursore;
134:         Lcd_Out(2,1,txt_vuoto_2);
135:         Lcd_Out(3,1,txt_vuoto_2);
136:         Lcd_Out(4,1,txt_vuoto_2);
137:         Lcd_Out((flag_cursore+1),1,txt_freccia);
138:         scrivi_display(2,3,9,menu_imp[indice]);
139:         scrivi_display(3,3,9,menu_imp[indice+1]);
140:         scrivi_display(4,3,9,menu_imp[indice+2]);
141:         flag_cambio=0;
142:         //attesa rilascio pulsante
143:         while((PULS_UP==1)||(PULS_DOWN==1)) {}
144:       }//if(flag_cambio==1)
145:     }//while(1)
146: }//menu_principale()
147:
148:
149: void main()
150: {
151:  CMCON=0x07;       //disabilito i comparatori
152:  ANSEL=0x00;       //disabilito i convertitori
153:  OSCCON=0xFC;      //CLOCK interno 8MHz
154:  TRISA=0xFF;       //definizione in/out PORTA
155:  TRISB=0x00;       //definizione in/out PORTB
156:
157:  //inizializzazione display LCD
158:  Lcd_Init();
159:  Lcd_Cmd(_LCD_CLEAR);
160:  Lcd_Cmd(_LCD_CURSOR_OFF);
161:  LED_ROSSO=0;
162:  LED_VERDE=0;
163:
164:  while(1)  //loop continuo ----------------------------------------
165:   {
166:
167:     esito_menu=menu_principale();
168:
169:     Lcd_Cmd(_LCD_CLEAR);
170:     switch(esito_menu)   //analisi della risposta
171:      {
172:         case 0:
173:          {
174:             Lcd_Out(2,3,"NESSUNA SCELTA");
175:             break;
176:          }//case 1:
177:         case 1:
178:          {
179:             Lcd_Out(2,3,"ROSSO ON");
180:             LED_ROSSO=1;
181:             break;
182:          }//case 1:
183:         case 2:
184:          {
185:             Lcd_Out(2,3,"VERDE ON");
186:             LED_VERDE=1;
```

```
187:             break;
188:           }//case 2:
189:          case 3:
190:           {
191:             Lcd_Out(2,3,"TUTTI ON");
192:             LED_ROSSO=1;
193:             LED_VERDE=1;
194:             break;
195:           }//case 3:
196:          case 4:
197:           {
198:             Lcd_Out(2,3,"TUTTI OFF");
199:             LED_ROSSO=0;
200:             LED_VERDE=0;
201:             break;
202:           }//case 4:
203:          case 5:
204:           {
205:             Lcd_Out(2,3,"IMPULSO ROSSO");
206:             LED_ROSSO=0;
207:             delay_ms(500);
208:             LED_ROSSO=1;
209:             delay_ms(500);
210:             LED_ROSSO=0;
211:             break;
212:           }//case 5:
213:          case 6:
214:           {
215:             Lcd_Out(2,3,"IMPULSO VERDE");
216:             LED_VERDE=0;
217:             delay_ms(500);
218:             LED_VERDE=1;
219:             delay_ms(500);
220:             LED_VERDE=0;
221:             break;
222:           }//case 6:
223:       }//switch(esito_menu)
224:      delay_ms(1000);
225:    }//while(1)
226: }//main
```