```
1     .
2     .
3     .
4     .
5     //Init e Lcd Display============================
6     if (DisplayStatus)   // se = 0 sono disabilitate tutte le routine relative al display
7     {
8        if (InitFlag)
9        /* Inizializza l'LCD e le routine di movimento.
10          Il flag e' settato solo prima di entrare nel main,in questo modo l'inizializzazione
11          dell'LCD avviene una sola volta.
12          Rimane in loop ciclando i led colorati fino a quando non si tocca un pulsante
13       */
14
15       {
16          if  (BumperDx && BumperSx) // resta fermo finche' non si tocca  un baffo
17          {
18             if (TimerLcd <= 0)      // Per le temporizzazioni necessarie all'init dell'LCD
19             {
20                if (!LcdPutCharFlag) // e' terminata la trasmissione di un byte
21                {
22                   if (LcdStringPtr== 0xFF) // e' terminato l'invio della stringa
23                   {
24                      LcdInit();
25                   }
26                }
27             }
28          }
29          else
30          {
31             SequenzaStart();  // esce dallo stato init avviando le sequenze di movimento
32          }
33       }
34       else  // finita la fase di init, l'LCD viene usato per mostrare i parametri voluti
35       {
36         if (TimerLcd <= 0)    // e' tempo di aggiornare il display
37         {
38             TimerLcd=LcdCycle;// reset timer
39
40             if (DisplayStatus==99)
41             {
42                DisplayStatus=0;  // ha visualizzato la scritta fissa e disabilita LCD
43             }
44             else
45             {
46                if (DisplayStatus> MaxLcdStatusItem || DisplayStatus <= 0)
47                {
48                   LcdBL=0;          // LCD Backlight off
49                                     //   "0123456789012345  0123456789012345"
50                   LcdPutStringInitRom(0, "----- RUN ------\r----------------");
```

```
51                DisplayStatus=99;
52             }
53             else
54             {
55             LcdBL=1;          // LCD Backlight on
56             Display();
57             /* Visualizza valori diversi in funzione della variabile DisplayStatus
58
59                1: visualizza valori sensori di prossimita'
60                2: visualizza angolo bussola
61                3: visualizza valori gas e fotoresistenze
62                4: visualizza soglie Light1, Light2 e GasOn
63                5: visualizza costanti Kp, Ki, Kd, En
64
65               99: disabilita la routine dopo la scritta fissa
66
67                   impostando un qualsiasi valore diverso da quelli elencati
68                   si scrive prima una stringa fissa e poi (al giro successivo)
69                   si disabilita la scrittura su display.
70             */
71             }
72          }
73       }
74    }
75
76
77 //LcdPutChar ----------------------------------
78    if (LcdPutCharFlag)
79    {
80    /*
81       La routine a livello superiore ha abilitato la scrittura di un nuovo carattere
82       verso l'LCD tramite LcdPutChar()
83    */
84
85       if (!I2c[LcdPtr].Flag.Tx && !I2c[LcdPtr].Flag.Rx)
86       /*
87          se il buffer di trasmissione I2C e' libero si può passare un nuovo byte
88          all'I/O expander.
89          Per scrivere un carattere sull'LCD occorre inviare una sequenza di byte
90          tramite l'I/O expander
91       */
92       {
93          if (Lcd4_8BitFlag)   // modalita' 4 o 8 bit
94          {
95             Lcd4Bit();
96          }
97          else
98          {
99             Lcd8Bit();
100         }
```

```
101         }
102      }
103
104
105   //LcdPutString ------------------------------------------------
106      if (LcdStringPtr!=0xFF)     // c'e' una stringa da scrivere sull'LCD
107      {
108      /*
109         La routine a livello superiore ha abilitato la scrittura di una nuova stringa
110         verso l'LCD tramite LcdPutStringInit()
111      */
112         if (!LcdPutCharFlag) // e' terminata la trasmissione del byte  precedente
113         {
114            LcdPutString();
115         }
116      }
117
118   }//Init e Lcd Display
119   //============================================
120   .
121   .
122   .
123   .
124   /*Display ***************************************************************************
125   Visualizza valori sul display in funzione della variabile DisplayStatus
126   */
127
128   void Display (void)
129   {
130         switch (LcdStatus)
131         {
132         case 0:
133            TimerLcd=LcdCycle;                        // pausa all'avvio del ciclo
134            LcdPutChar (0x01, 0, 4);                  // clear display
135            LcdStatus ++;                             // passa allo stato successivo
136            break;
137
138         case 1:
139            LcdPutStringInitRom(0, &(LcdTable1[DisplayStatus][0]));// invia stringa all'LCD
140            TimerLcd=LcdCycle;                        // reset timer
141            LcdStatus ++;                             // passa allo stato successivo
142            break;
143
144         case 2:
145            LcdPutStringInitC2A(0x40,LcdVar[DisplayStatus][0]);// visualizza valore
146            TimerLcd=LcdCycle;                     // reset timer
147            LcdStatus ++;                          // passa allo stato successivo
148            break;
149
150         case 3:
```

```
151            LcdPutStringInitC2A(0x44,LcdVar[DisplayStatus][1]);// visualizza valore
152            TimerLcd=LcdCycle;                        // reset timer
153            LcdStatus ++;                             // passa allo stato successivo
154            break;
155
156
157         case 4:
158            if (FlagMenuMod)
159            {// se in modalita' modifica lampeggia il nome della variabile da modificare
160               LcdPutStringInitRom(0, &(LcdTable2[DisplayStatus+6+(CursorStatus*6)][0]));
161            }
162            else
163            {
164               LcdPutStringInitRom(0, &(LcdTable2[DisplayStatus][0]));// invia stringa all'LCD
165            }
166            TimerLcd=LcdCycle;                           // reset timer
167            LcdStatus ++;                                // passa allo stato successivo
168            break;
169
170
171         case 5:
172            LcdPutStringInitC2A(0x49,LcdVar[DisplayStatus][2]);// visualizza valore
173            TimerLcd=LcdCycle;                        // reset timer
174            LcdStatus ++;                             // passa allo stato successivo
175            break;
176
177         default:
178            LcdPutStringInitC2A(0x4D,LcdVar[DisplayStatus][3]);// visualizza valore
179            TimerLcd=LcdCycle;                        // reset timer
180            LcdStatus = 1;                            // ricomincia dal secondo passo
181            break;
182
183         } // end switch
184
185
186   } // Display
187   /*****************************************************************************/
188
189
190   /*LcdPutString ***************************************************************
191   Scrive sull'LCD, la stringa posta nel buffer
192   Un carattere "\r" all'interno della stringa significa "a capo" (Cursor = 0x40)
193   */
194
195   void LcdPutString(void)
196   {
197      if (LcdString[LcdStringPtr])
198      {
199         if (LcdString[LcdStringPtr] == '\r') // a capo
200         {
```

```
201             LcdPutChar(0x80+0x40, 0, 4);  // inizio seconda riga
202             LcdStringPtr++;               // al prossimo giro scrive il carattere successivo
203          }
204          else
205          {
206             LcdPutChar(LcdString[LcdStringPtr], 1, 4);   // scrive nesimo carattere
207             LcdStringPtr ++;              // al prossimo giro scrive il carattere successivo
208          }
209       }
210       else
211       {
212          LcdStringPtr=0xFF;   // ultimo passaggio, disabilita questa  routine
213       }
214
215    } // LcdPutString
216    /*****************************************************************************/
217
218
219    /*LcdPutStringInitC2A ***********************************************************
220    Inizializza la scrittura di una stringa di caratteri sull'LCD alla posizione "Cursor".
221    Converte una variabile unsigned char in tre caratteri ASCII
222    grazie a Rocco Iannacci
223    */
224
225    void LcdPutStringInitC2A(unsigned char Cursor,unsigned char Val)
226    {
227       unsigned char u,d,c,tmp;
228
229       LcdPutChar(0x80+Cursor, 0, 4);   // posizione di partenza del cursore
230
231       LcdStringPtr=0;      // Puntatore al carattere della stringa in stampa
232                            // se diverso da 0xFF avvia la stampa della stringa
233       c=Val/100;
234       tmp=(Val-100*c);
235       d=tmp/10;
236       u=tmp-10*d;
237
238       LcdString[3]=0;                    // EOL
239       LcdString[2]="0123456789"[u];      // unita'
240       LcdString[1]="0123456789"[d];      // decine
241       LcdString[0]=" 123456789"[c];      // centinaia
242
243    } // LcdPutStringInitC2A
244    /*****************************************************************************/
245
246
247    /*LcdPutStringInitRom ***********************************************************
248    Inizializza la scrittura di una stringa di caratteri sull'LCD alla posizione "Cursor".
249    Dal momento che l'allocazione della memoria e' diversa tra memoria programma e memoria
250    ram, questa routine e' valida solo per la visualizzazione di stringhe fisse.
```

```
251  */
252
253  void LcdPutStringInitRom(unsigned char Cursor,const char rom *LcdStr)
254  {
255     char count=0;         // contatore
256     LcdStringPtr=0;       // Puntatore al carattere della stringa in stampa
257                           // se diverso da 0xFF avvia la stampa della stringa
258
259     while (*(LcdStr+count))
260     {
261        LcdString[count] = *(LcdStr+count);
262        count++;
263     }
264     LcdString[count]=0;     // nul character
265
266
267     LcdPutChar(0x80+Cursor, 0, 4);   // posizione di partenza del cursore
268
269  } // LcdPutStringInitRom
270  /*****************************************************************************/
271
272
273  /*LcdPutChar ***************************************************************************
274  inizializza flag e variabili per inviare un nuovo carattere verso l'LCD.
275  Parametri:
276  LChar    = carattere da scrivere (dato o comando)
277  LRS      =1 se dato, =0 se comando
278  L4_8Bit  =4 se modalita' 4 bit, 8 se modalita' 8 bit
279  */
280
281  void LcdPutChar (char LChar, unsigned char LRS, unsigned char L4_8Bit)
282  {
283     LcdChar = LChar;  // carattere (dato o comando)
284
285     if (LRS)          // = 1 se Dato, =0 se Comando
286     {
287        LcdRS=1;
288     }
289     else
290     {
291        LcdRS=0;
292     }
293
294     if (L4_8Bit==4)
295     {
296        Lcd4_8BitFlag=1;  // = 1 se modalita' 4 bit, =0 se 8 bit
297     }
298     else
299     {
300        Lcd4_8BitFlag=0;
```

```
301     }
302
303     LcdRW=0;           // scrittura
304     LcdPutCharFlag=1; // abilita la scrittura del byte
305     LcdByteStatus=0;  // azzera il contatore di stato della routine LcdXBit
306
307  } // LcdPutChar
308  /*****************************************************************************/
309
310
311  /*LcdByteSet*****************************************************************
312  ricostruisce il byte da inviare all'LCD tramite I2C dai singoli bit
313  */
314
315  unsigned char LcdByteSet (void)
316  {
317
318  return (((((((0x00|LcdData)<<1)|LcdNA)<<1)|LcdRW)<<1)|LcdRS)<<1)|LcdBL;
319
320  } // LcdByteSet
321  /*****************************************************************************/
322
323
324  /*Lcd4Bit ********************************************************************
325  Invia un byte all'LCD nella modalita' a 4 bit tramite I2C
326  L'I/O expander 1 usato per pilotare l'LCD e' il device I2c numero 3
327  Il driver LCD HD44780 con il clock interno a 270KHz richiede una pausa di 37uSec
328  tra un carattere e l'altro. Per inviare un byte tramite l'I/O expander PCF8574 con il
329  clock a 100KHz ci vogliono almeno 90uSec (8 bit + ACK * 10uSec a bit), piu' che
330  sufficienti quindi per le temporizzazioni richieste.
331  */
332
333  void Lcd4Bit (void)
334  {
335     switch (LcdByteStatus)
336     {
337        case (0):
338           LcdEN=0;
339           LcdData=LcdChar >> 4;                  // upper nibble
340           I2c[LcdPtr].Flag.Tx = 1;               // un byte da trasmettere
341           I2c[LcdPtr].TxBuff[0] = LcdByteSet() ; // compone il byte da inviare e lo passa
342                                                  // alle routine I2C. Il buffer e' stato
343                                                  // controllato prima e quindi e'
344                                                  // sicuramente vuoto
345           LcdByteStatus ++;                         // passa allo stato successivo
346        break;
347
348        case (1):
349           LcdEN=1;                               // strobe
350           LcdByteStatus ++;                      // passa allo stato successivo
```

```
351            LcdEN=0;
352         break;
353
354         case (2):
355            LcdData=LcdChar;                          // lower nibble
356            I2c[LcdPtr].Flag.Tx = 1;                  // in modalita' 4 bit trasmette  un
357            I2c[LcdPtr].TxBuff[0] = LcdByteSet() ;    // nibble per volta
358            LcdByteStatus ++;                         // passa allo stato successivo
359         break;
360
361         case (3):
362            LcdEN=1;                                  // strobe
363            LcdPutCharFlag = 0;  // l'invio del carattere e' finito, se la routine a
364                                 // livello superiore ha altri byte da inviare
365                                 // inizia una nuova sequenza con LcdPutChar
366            LcdEN=0;
367         break;
368
369      } // end switch
370
371   } // Lcd4Bit
372   /*****************************************************************************/
373
374
375   /*Lcd8Bit ********************************************************************
376   Invia un byte all'LCD nella modalita' a 8 bit tramite I2C, usata solo per l'init
377   L'I/O expander 1 usato per pilotare l'LCD e' il device I2c numero 3
378   Il driver LCD HD44780 con il clock interno a 270KHz richiede una pausa di 37uSec
379   tra un carattere e l'altro. Per inviare un byte tramite l'I/O expander PCF8574 con il
380   clock a 100KHz ci vogliono almeno 90uSec (8 bit + ACK * 10uSec a bit), piu' che
381   sufficienti quindi per le temporizzazioni richieste.
382   */
383
384   void Lcd8Bit (void)
385   {
386      switch (LcdByteStatus)
387      {
388         case (0):
389            LcdEN=0;
390            LcdData=LcdChar;
391            I2c[LcdPtr].Flag.Tx = 1;                  // un byte da trasmettere
392            I2c[LcdPtr].TxBuff[0] = LcdByteSet() ;    // compone il byte da inviare e lo passa
393                                                      // alle routine I2C. Il buffer e' stato
                                                  // controllato prima e quindi e'
394                                                      // sicuramente vuoto
395            LcdByteStatus ++;                         // passa allo stato successivo
396         break;
397
398         case (1):
399            LcdEN=1;                                  // strobe
```

```
400           LcdPutCharFlag = 0;  // l'invio del carattere e' finito, se la routine a
401                                // livello superiore ha altri byte da inviare
402                                // inizia una nuova sequenza con LcdPutChar
403           LcdEN=0;;
404        break;
405
406     } // end switch
407
408  } // Lcd8Bit
409  /*****************************************************************************/
410
411
412  /*LcdInit ********************************************************************
413  Per l'inizializzazione del display LCD
414  Interfaccia a 4 bit, 2 linee di visualizzazione,e caratteri di 5 * 7 punti.
415  */
416  void LcdInit (void)
417  {
418
419     switch (LcdInitStatus)
420     {
421        case 0:
422           LcdPutChar (0x03, 0, 8);              // invia carattere 0x30 all'LCD
423           LcdInitStatus ++;                     // passa allo stato successivo
424           break;
425
426        case 1:
427           TimerLcd=5;                           // inizializza timer 5mS
428           LcdInitStatus ++;                     // passa allo stato successivo
429           break;
430
431        case 2:
432
433           LcdPutChar (0x03, 0, 8);              // invia carattere 0x30 all'LCD
434           LcdInitStatus ++;                     // passa allo stato successivo
435           break;
436
437        case 3:
438           TimerLcd=5;                           // inizializza timer 5mS
439           LcdInitStatus ++;                     // passa allo stato successivo
440           break;
441
442        case 4:
443           LcdPutChar (0x03, 0, 8);              // invia carattere 0x30 all'LCD
444           LcdInitStatus ++;                     // passa allo stato successivo
445           break;
446
447        case 5:
448           LcdPutChar (0x02, 0, 8);                 // Modalita' 4 bit
449           LcdInitStatus ++;                     // passa allo stato successivo
```

```
450             break;
451
452         case 6:
453             TimerLcd=5;                             // inizializza timer 5mS
454             LcdInitStatus ++;                       // passa allo stato successivo
455             break;
456
457         case 7:
458             LcdPutChar (0x28, 0, 4);                // 1/16 duty, 5x7 font
459             LcdInitStatus ++;                       // passa allo stato successivo
460             break;
461
462         case 8:
463             LcdPutChar (0x0C, 0, 4);                // display on, blink cursor off
464             LcdInitStatus ++;                       // passa allo stato successivo
465             break;
466
467         case 9:
468             LcdPutChar (0x06, 0, 4);                // entry mode=increment cursor position
469             LcdInitStatus ++;                       // passa allo stato successivo
470             break;
471
472         case 10:
473             LcdPutChar (0x01, 0, 4);                // clear display
474             LcdInitStatus ++;                       // passa allo stato successivo
475             break;
476
477         case 11:
478             TimerLcd=5;                             // inizializza timer 5mS
479             LcdInitStatus ++;                       // passa allo stato successivo
480             break;
481
482         case 12:
483             LcdBL=1;                                // Backlight ON
484             LcdPutStringInitRom(0,Ver);             // scrive versione
485             LcdInitStatus ++;                       // passa allo stato successivo
486             break;
487
488
489
490         case 13:
491             LcdBL=0;                                // Backlight OFF
492             LedRossoON;                             // rimane in loop ciclando il led
493             LedGialloOFF;                           // ogni 250mS
494             LedVerdeOFF;
495             LcdInitStatus ++;                       // passa allo stato successivo
496             LcdPutStringInitC2A(0x40,GasVal);       // visualizza valore sensore gas
497             break;
498
499         case 14:
```

```
500            TimerLcd=250;                       // inizializza timer 250mS
501            LcdInitStatus ++;                   // passa allo stato successivo
502            break;
503
504         case 15:
505            LedRossoOFF;
506            LedGialloON;
507            LcdInitStatus ++;                   // passa allo stato successivo
508            break;
509
510         case 16:
511            TimerLcd=250;                       // inizializza timer 250mS
512            LcdInitStatus ++;                   // passa allo stato successivo
513            break;
514
515         case 17:
516            LedGialloOFF;
517            LedVerdeON;
518            LcdInitStatus ++;                   // passa allo stato successivo
519            break;
520
521         case 18:
522            TimerLcd=250;                       // inizializza timer 250mS
523            LcdInitStatus = 13;                 // ricomincia il ciclo dei led
524            break;
525
526
527         default:
528            break;
529
530      } // end switch
531
532   } // LcdInit
533   /*****************************************************************************/
534   .
535   .
536   .
537   .
538
539
```