```
1     /*
2     viene incluso in "dino.c", contiene:
3        definizioni
4        prototipi funzioni
5        settings
6        descrizioni
7     */
8
9
10    /* Definizioni globali+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
11
12    #define PLL 1
13
14    //----- Status bits
15    #ifdef PLL  // 40MHz
16       #pragma config OSC = HSPLL       //  10 x 4 = 40MHZ
17    #else
18       #pragma config OSC = HS          //  20Mhz
19    #endif
20
21    #pragma config WDT = OFF
22    #pragma config LVP = OFF
23    // #pragma config PWRT = ON   non compatibile con ICD2
24    #pragma config XINST = ON
25
26    //----- Definizione porte
27    #define EncoderRint INTCON3bits.INT1IF    // interrupt flag encoder SX
28    #define EncoderLint INTCON3bits.INT2IF    // interrupt flag encoder DX
29    #define KeypadInt INTCONbits.INT0IF       // interrupt flag tasto premuto
30
31                                      // B0 Interrupt I/O exp      I INT0
32    #define EncoderRpulse PORTBbits.RB1 // B1 Encoder SX,ticks     I INT1
33    #define EncoderLpulse PORTBbits.RB2 // B2 Encoder DX,ticks     I INT2
34                                      // B3
35    #define EncoderRdir PORTBbits.RB4   // B4 Encoder SX,direzione  I
36    #define EncoderLdir PORTBbits.RB5   // B5 Encoder DX,direzione  I
37                                      // B6 PGC (ICD2)
38                                      // B7 PGD (ICD2)
39                                      // C0
40                                      // C1 PWMmotR               O PWM
41                                      // C2 PWMmotL               O PWM
42                                      // C3 SCL
43                                      // C4 SDA
44    #define Buzzer PORTCbits.RC5        // C5 Buzzer               O
45    #define MotEnable PORTDbits.RD0     // D0 Enable H-bridge      O
46    #define SoundIn PORTDbits.RD1       // D1 Tone  decoder 4KHz    I
47    #define BumperDx PORTDbits.RD2      // D2 Bumper destro        I
48    #define BumperSx PORTDbits.RD3      // D3 Bumper sinistro      I
49    #define LedRosso PORTDbits.RD4      // D4 LedRosso (temp)      O
50    #define LedVerde PORTDbits.RD5      // D5 LedVerde (temp)      O
```

```
51    #define LedGiallo PORTDbits.RD6     // D6 LedGiallo (temp)      O
52    #define LcdEN PORTDbits.RD7         // D7 Lcd Enable            O
53
54    struct Bits{
55       unsigned bit0:1;
56       unsigned bit1:1;
57       unsigned bit2:1;
58       unsigned bit3:1;
59       unsigned bit4:1;
60       unsigned bit5:1;
61       unsigned bit6:1;
62       unsigned bit7:1;
63    };
64
65    struct Bits VARbits1;
66
67    struct Bits VARbits2;
68
69    struct Bits VARbits3;
70
71    struct Bits VARbits4;
72
73    //----- Flags
74    #define  PidTick VARbits1.bit0          // timer routine PID, impostato dalla ISR
75    #define  Space2RunFlag VARbits1.bit1    // Abilita la routine di controllo dello spazio da percorrere
76    #define  FlagBumpers VARbits1.bit2      // Abilita la routine controllo collisioni
77    #define  FlagStop VARbits1.bit3         // Flag per stop motori
78    #define  FlagAD VARbits1.bit4           // Flag per controllo stato routine conversione AD
79    #define  FlagEye VARbits1.bit5          // Flag abilitazione routine Eye
80    #define  FlagLight VARbits1.bit6        // Flag abilitazione routine Light
81    #define  FlagSound VARbits1.bit7        // Flag abilitazione routine Sound
82    #define  FlagTargetLight VARbits2.bit0 // Flag abilitazione sensori luce dopo "obiettivo raggiunto"
83    #define  FlagTargetSound VARbits2.bit1 // Flag abilitazione sensori suono dopo "obiettivo raggiunto"
84    #define  FlagTargetGas VARbits2.bit2   // Flag abilitazione sensore gas dopo "obiettivo raggiunto"
85    #define  RandomBit VARbits2.bit3        // Cambia da 1 o 0 ogni millisecondo
86    #define  FlagGas VARbits3.bit2          // Flag abilitazione routine gas
87    #define  FlagTime VARbits3.bit4         // Flag abilitazione routine Temporizzazioni
88
89
90    //----- Variabili
91    int i = 0; // contatore generico
92    int j = 0; // contatore generico
93
94    // le variabili long seguenti contengono impulsi dell'encoder
95    long EncoderRcount = 0;       // impulsi encoder destro
96    long EncoderLcount = 0;       // impulsi encoder sinistro
97    long EncoderRcountPrev =0;    // misura precedente per il computo della velocita'
98    long EncoderLcountPrev =0;    // spazio percorso = valore attuale - valore precedente
99
100   long Space2RunR = 0;          // spazio da percorrere
```

```
101   long Space2RunL = 0;          //
102   long Space2RunRstart = 0;     // valore di partenza per il calcolo della strada da percorrere
103   long Space2RunLstart = 0;     //
104
105   int SpeedR = 0;               // velocita' attuale ruota destra
106   int SpeedL = 0;               // velocita' attuale ruota sinistra
107   int DesSpeedR = 0;            // velocita' desiderata ruota destra
108   int DesSpeedL = 0;            // velocita' desiderata ruota sinistra
109   int IntR = 0;                 // accumulatore errore integrale del PID (motore destro)
110   int IntL = 0;                 // accumulatore errore integrale del PID (motore sinistro)
111   int ErroreRprev = 0; // errore precedente, serve per il calcolo dell'errore differenziale del PID (motore destro)
112   int ErroreLprev = 0; // errore precedente, serve per il calcolo dell'errore differenziale del PID (motore sinistro)
113   int TimerStop;                // timer fino a 30 Sec per routine Stop motori
114   int TimerBumpers;             // timer per routine Bumpers
115   int TimerBeep = 0;            // timer per routine Beep
116   int TimerSensor = 0;          // timer per disabilitazione sensori
117   int TimerDeadCorner = 0;      // timer per routine angoli morti
118
119   unsigned char DeltaT;          // tempo trascorso, per il computo della velocita'
120   char DeadCorner = 0;           // conta le collisioni sullo stesso lato per uscire da un angolo morto
121   #define EyeLlVal LcdVar[1][0] // valore attuale del rivelatore IR laterale sinistro
122   #define EyeLfVal LcdVar[1][1] // valore attuale del rivelatore IR frontale sinistro
123   #define EyeRfVal LcdVar[1][2] // valore attuale del rivelatore IR frontale destro
124   #define EyeRlVal LcdVar[1][3] // valore attuale del rivelatore IR laterale destro
125
126   #define FrLVal LcdVar[3][0]   // valore attuale della Fotoresistenza Sinistra
127   #define FrCVal LcdVar[3][1]   // valore attuale della Fotoresistenza Centrale
128   #define FrRVal LcdVar[3][2]   // valore attuale della Fotoresistenza Destra
129   #define GasVal LcdVar[3][3]   // valore attuale del Sensore di gas
130   char DebounceLight =0 ;       // contatore per debounce target ok luce
131   char DebounceSound =0 ;       // contatore per debounce target ok suono
132   char BeepCount = 0;           // indica quanti beep verranno suonati
133
134   char TimerAD = 0;             // timer per routine AD
135   char PathSeq [11]; // contiene la sequenza dei movimenti da fare (n-1 + EOL)
136   char PathSeqPointer; // puntatore dell'array sequenza
137
138
139   //----- Definizione parametri movimento
140   /* diametro ruota = 58mm -> circonferenza = 182,212mm
141      l'encoder e' su uno degli assi del gruppo riduttore e gira 20 volte piu' della ruota
142      questo ha 40 finestrelle che danno un interrupt ognuna
143      quindi 40 x 20 = 800 tick per giro -> risoluzione 0,227765 mm
144   */
145   #define stepR  228   //  in millesimi di millimetro
146   #define stepL  228   //  i valori leggermente diversi correggono le piccole differenze tra le due ruote
147   #define constSpeed 350 // velocita' normale, da verificare se il PID ha ancora sufficiente controllo
148   #define lowSpeed   200 // velocita' ridotta per approccio ostacoli
149   #define manSpeedFwd   200 // velocita' delle ruote durante le manovre
150   #define manSpeedRew  -200 // tenendola bassa la precisione e' maggiore
```

```
151
152  #define kp LcdVar[5][0]    // parametro memorizzato in EEPROM interna
153  #define ki LcdVar[5][1]    // parametro memorizzato in EEPROM interna
154  #define kd LcdVar[5][2]    // parametro memorizzato in EEPROM interna
155  #define En LcdVar[5][3]    // parametro memorizzato in EEPROM interna enable motori
156
157
158  /*
159     parametri per il moto in vicinaza degli ostacoli
160     sono prese in considerazione due soglie di allarme per la misura dei sensori IR
161     sotto LowDist il bot comincia a rallentare
162     sotto MinDist gira in senso opposto al lato dell'ostacolo rivelato
163  */
164
165  // luce
166  #define Light1 LcdVar[4][0]    // parametro memorizzato in EEPROM interna
167  #define Light2 LcdVar[4][1]    // parametro memorizzato in EEPROM interna
168  // queste soglie sono valide per il campo di gara del Von Neumann
169  #define DebounceLightCount 20 // soglia conteggi target ok luce
170  #define TimeDisSensLight TimeMotStopTarget + 6000 //tempo sensori off oltre stop motori (in mS)
171
172  // gas
173  #define GasOn LcdVar[4][3]     // parametro memorizzato in EEPROM interna
174  #define TimeDisSensGas TimeMotStopTarget + 15000 //tempo sensori off oltre stop motori (in mS)
175
176  // suono
177  #define DebounceSoundCount 10 // soglia conteggi target ok suono
178  #define TimeDisSensSound TimeMotStopTarget + 6000 //tempo sensori off oltre stop motori (in mS)
179
180  #define BeepTime 200          // durata/2 in ms del periodo di beep
181  #define TimeMotStopTarget 1500 // motori fermi per 1,5 sec raggiunto l'obiettivo
182  // secondo il regolamento dovrebbe fermarsi per 4 sec ma... non lo rispetta nessuno

183
184  #define MinDist 0x60 // 6cm distanza di guardia dei sensori, sotto questa distanza gira
185  #define LowDist 0x26 // 15cm distanza di guardia dei sensori, sotto questa distanza rallenta
186  #define TargetDist 0x14 // 22 cm distanza di "bersaglio individuato"
187  // il valore esadecimale fa riferimento alla tabella sotto
188
189  // ATTENZIONE: a tensione minore corrisponde distanza maggiore
190
191  /* sotto i 5cm gira di 10° nella direzione opposta a quella del sensore che lo ha rilevato.
192  ------------------------------------------------------------------------------------
193  lettura sensori  IR Sharp GP2D120
194  distanza dal paraurti
195
196  distanza    cm:25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02
197  lettura AD hex:10 11 13 14 16 18 1a 1e 20 24 26 2b 30 33 36 3b 3f 46 4e 5a 66 77 8d a2
198
199  sotto i 2cm la  tensione ricomincia a  scendere.
```

```
200  ------------------------------------------------------------------------------------------*/
201
202  /*Porte analogiche, ADCON0
203  bit 5-2 CHS2:CHS0: Analog Channel Select bits
204  0000 = Channel 0 (AN0)   A0
205  0001 = Channel 1 (AN1)   A1
206  0010 = Channel 2 (AN2)   A2
207  0011 = Channel 3 (AN3)   A3
208  0100 = Channel 4 (AN4)   A5
209  0101 = Channel 5 (AN5)   E0
210  0110 = Channel 6 (AN6)   E1
211  0111 = Channel 7 (AN7)   E2
212  */
213  #define FrR     0b00000000       // AN0 FotoResistenza destra         I analog
214  #define FrL     0b00000100       // AN1 FotoResistenza sinistra       I analog
215  #define EyeRf   0b00001000       // AN2 IR detect frontale destro     I analog
216  #define EyeLf   0b00001100       // AN3 IR detect frontale sinistro   I analog
217  #define EyeLl   0b00010100       // AN5 IR detect laterale destro     I analog
218  #define EyeRl   0b00011100       // AN7 IR detect laterale sinistro   I analog
219  #define FrC     0b00011000       // AN6 FotoResistenza centrale       I analog
220  #define GasIn   0b00010000       // AN4 Gas                            I analog
221
222
223  #define LedVerdeON LedVerde=0
224  #define LedVerdeOFF LedVerde=1
225  #define LedRossoON LedRosso=0
226  #define LedRossoOFF LedRosso=1
227  #define LedGialloON LedGiallo=0
228  #define LedGialloOFF LedGiallo=1
229  #define DeadCornerTime 5000   // finestra di tempo di analisi delle collisioni
230  #define MaxDeadCorner  10 // limite di urti su bumpers o sensori IR
231
232
233  //----- Macro
234  #define abs(x) ((x) > 0 ? (x) : -(x)) // funzione abs() valida per int, long, float...
235
236  /* Signed divide by power of two
237     mantiene i bit di segno in quanto il compilatore non li propaga nel caso di shift
238     di numeri negativi
239  */
240  #define DivIntS64(x) ((x) >= 0 ? (x>>6) : ((x>>6) | 0xFC00))   // div 64 signed int
241  #define DivLongS16(x)  ((x) >= 0 ? (x>>4) : ((x>>4) | 0xF0000000))   // div 16 signed long
242  #define DivLongS128(x) ((x) >= 0 ? (x>>7) : ((x>>7) | 0xFE000000))   // div 128 signed long
243  #define DivLongS256(x) ((x) >= 0 ? (x>>8) : ((x>>8) | 0xFF000000))   // div 256 signed long
244
245  //----- Definizioni variabili I2C
246  #define  I2cEventFlag VARbits2.bit4   // la porta SSP ha comunicato un evento tramite ISR
247  #define  I2cBusCollFlag VARbits2.bit5 // la porta SSP ha comunicato un bus collision tramite ISR
248  #define  I2cBusyFlag VARbits2.bit6    // e' in corso una comunicazione I2c (N byte Tx e/o Rx)
249
```

```
250  #define  I2cMaxDev 6          // n di device I2C da controllare
251
252  unsigned char I2cDevPtr = 0;  // puntatore al device I2C in uso
253
254  const unsigned char I2cAdr[] = {0x00, 0xC0, 0x50, 0x40, 0x42, 0xE0}; //indirizzi dei device I2C
255
256  struct Nibble {
257     unsigned char Rx:4;        // semibyte da usare dove bastano 16 stati
258     unsigned char Tx:4;
259     };
260
261  struct I2cS {
262     struct Nibble Flag;        // Rx 0=idle, 1=1 Byte da ricevere, 2=2 Byte da ricevere
263                                // Tx 0=idle, n=n Byte da trasmettere(con n!=0)
264     unsigned char RxBuff[2];   // buffer Byte(s) da ricevere
265     unsigned char TxBuff[4];   // buffer Byte(s) da trasmettere
266     };
267
268  struct I2cS I2c[I2cMaxDev];
269  struct Nibble Ptr;            // puntatore per buffer byte di scambio I2c (Rx e TX)
270  char I2cStat;                 // variabile di stato sequenza I2c
271
272     // codici stato eseguito
273
274     enum I2cBusStates{START, WRITE, ADRR, READ, ACK, NACK, RSTART, STOP, FINE};
275
276  // Indirizzi puntatori devices I2c
277  #define ExtEepromPtr 0
278  #define CmpPtr 1
279  #define AdExpanderPtr 2
280  #define LcdPtr 3
281  #define KeypadPtr 4
282  #define SonarPtr 5
283
284
285
286  //----- Definizioni variabili LCD
287  struct {
288    unsigned LcdBL:1;  // Back Light-> 1 = ON
289    unsigned LcdRS:1;  // Instruction/Data register selection-> 0=instruction 1=data
290    unsigned LcdRW:1;  // Read/Write selection-> 0=reg.write, 1=reg.read
291    unsigned LcdNA:1;  // Non Assegnato
292    unsigned LcdData:4;// Data I/O  lines (4 bit mode)
293    } LcdBits;
294
295  #define LcdBL LcdBits.LcdBL
296  #define LcdRS LcdBits.LcdRS
297  #define LcdRW LcdBits.LcdRW
298  #define LcdNA LcdBits.LcdNA
299  // LcdEN = Port D7
```

```
300  #define LcdData LcdBits.LcdData
301
302  #define LcdPutCharFlag VARbits2.bit7 // Flag abilitazione routine LcdPutByte
303  #define Lcd4 8BitFlag VARbits3.bit0  // Flag scelta  modalita' 4 o 8 bit
304  #define InitFlag VARbits3.bit1       // Avvia la  routine di inizializzazione
305  #define LcdCycle 100                 // Tempo di visualizzazione delle variabili(mS)
306
307  unsigned char LcdByteStatus;// Contatore stati per la scrittura sull'LCD
308  unsigned char LcdInitStatus;// Contatore stati per la routine di inizializzazione
309  unsigned char LcdStatus;    // Contatore stati per visualizzazione normale
310  unsigned char DisplayStatus;// Contatore stati per diversi modi di visualizzazione
311  char LcdChar;               // Il carattere (dato o comando) da inviare all'LCD
312  unsigned char LcdStringPtr; // Puntatore al carattere della stringa in stampa
313  unsigned char LcdString[34]="----------------\r----------------";// 32 caratteri + CR + EOL
314  char  *LcdBtoa="-999";      // stringa per la conversione da char a ascii
315  int TimerLcd;               // per le temporizzazioni dell'LCD
316
317  /* tabella stringhe su LCD
318     0 = Eye
319     1 = Compass
320     2 = GasLight
321     3 = Threshold
322     4 = PID
323  */
324
325  #define MaxLcdStatusItem 5  // numero max item da visualizzare
326
327  rom const char LcdTable1[][17] = {  "                ",
328                                      "ELl ELf. ERf ERl",
329                                      "Compass. Bearing",
330                                      "FrL FrC. FrR Gas",
331                                      "L1On L2On- GasOn",
332                                      " kP  kI-  kD  En" };
333  rom const char LcdTable2[][17] = {  "                ",
334                                      "ELl ELf .ERf ERl",
335                                      "Compass .Bearing",
336                                      "FrL FrC .FrR Gas",
337                                      "L1On L2On -GasOn",
338                                      " kP  kI - kD  En",
339                                      "                ",
340                                      "ELl ELf .ERf ERl",
341                                      "Compass .Bearing",
342                                      "FrL FrC .FrR Gas",
343                                      "     L2On  GasOn",
344                                      "     kI   kD  En",
345                                      "                ",
346                                      "ELl ELf .ERf ERl",
347                                      "Compass .Bearing",
348                                      "FrL FrC .FrR Gas",
349                                      "L1On       GasOn",
```

```
350                                     " kP       kD  En",
351                                     "              ",
352                                     "ELl ELf .ERf ERl",
353                                     "Compass .Bearing",
354                                     "FrL FrC .FrR Gas",
355                                     "L1On L2On  GasOn",
356                                     " kP  kI       En",
357                                     "              ",
358                                     "ELl ELf .ERf ERl",
359                                     "Compass .Bearing",
360                                     "FrL FrC .FrR Gas",
361                                     "L1On L2On       ",
362                                     " kP  kI   kD    " };
363
364   unsigned char LcdVar[MaxLcdStatusItem+1][5];// variabili da mostrare sul display
365
366   //----- Definizioni variabili Keypad
367   #define  FlagKbdIntr VARbits3.bit3        // Flag abilitazione routine Keypad
368   #define  FlagKbdStartRead VARbits3.bit5   // Flag avvio lettura Keypad
369   #define  FlagKbdStartI2c VARbits4.bit0    // Flag per attesa lettura I2c
370
371   unsigned char CursorStatus;         // Posizione cursore nei menu con modifica
372   unsigned char TimerKeypad;          // Timer per debounce tasti
373   unsigned char EepromStatus;         // Contatore stati per scrittura su EEPROM
374
375   #define  UserInterfaceFlag VARbits3.bit6  // Flag di esecuzione routine interfaccia utente
376   #define  FlagMenuMod VARbits4.bit1        // Flag Menu Modifica
377   #define  EepromFlag VARbits3.bit7         // Flag di esecuzione routine EEPROM
378
379
380   //----- Definizioni variabili bussola
381   #define CmpBearing LcdVar[2][0] // angolo letto dalla bussola
382   int TimerCmp = 0;               // timer lettura bussola
383   #define CmpCycle 100;           // periodo lettura bussola in ms
384   #define  FlagCmp VARbits4.bit2  // Flag esecuzione routine bussola
385   #define  FlagCmpCal VARbits4.bit3  // Flag esecuzione routine calibrazione bussola
386   #define  FlagCmpReg15 VARbits4.bit4  // Flag per scrittura in registro di calibrazione
387
388
389   /* Fine definizioni++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
390
391   /* Prototipi funzioni ************************************************************/
392
393   void CmpCal (void);
394
395   void CmpRead (void);
396
397   void Display (void);
398
399   void WriteEeprom(unsigned char Data, unsigned char Adr);
```

```
400
401   unsigned char ReadEeprom (unsigned char Adr);
402
403   void UserInterface(void);
404
405   void TimeKeeping (void);
406
407   void SequenzaStart (void);
408
409   void LcdPutString(void);
410
411   void LcdPutStringInitC2A(unsigned char Cursor,unsigned char Val);
412
413   void LcdPutStringInitRom(unsigned char Cursor, const char rom *LcdStr);
414
415   void LcdPutChar (char LChar, unsigned char LRS, unsigned char L4_8Bit);
416
417   unsigned char LcdByteSet (void);
418
419   void Lcd4Bit (void);
420
421   void Lcd8Bit (void);
422
423   void LcdInit (void);
424
425   void I2cHighService (void);
426
427   void I2cLowService (void);
428
429   void Target (char Target);
430
431   void Beep (void);
432
433   void SegueLuce (void);
434
435   void Light (void);
436
437   void Sound (void);
438
439   void Gas (void);
440
441   void Eye (void);
442
443   void Settings(void);
444
445   void ReadAD (void);
446
447   void MotSpeed (void);
448
449   void Turn(long Gradi);
```

```
450
451   void Walk(long Space);
452
453   void Space2Run (void);
454
455   void Stop(void);
456
457   void Path (void);
458
459   void Bumpers(void);
460
461   void InterruptHandlerHigh (void);
462
463   void InterruptHandlerLow (void);
464
465   /* Fine prototipi funzioni ************************************************************/
466
467   /*Settings*****************************************************************************
468                                                              setting di porte e registri
469   */
470   void Settings(void)
471   {
472
473   //-------Port A
474
475   TRISA    =    0b11111111;    // tutti input (anche quelli non usati)
476
477   //-------Port B
478
479   TRISB    =    0b00111111;    // tutti input
480   INTCON2bits.RBPU;            // PORTB Pull-up enabled
481
482
483   //-------Port C
484
485   TRISC = 0b10011001;
486   /*
487   RX
488   TX
489   Buzzer
490   SDA
491   SCL
492   PWMR
493   PWML
494   don't care
495   */
496
497
498   //-------Port D
499
```

```
500   TRISD = 0b00001110;
501   /*
502   bit 0, 4, 5, 6, 7 come output
503   gli altri come input
504
505
506   //-------Port E
507
508   TRISE = 0b00000111;
509   /*
510   don't care
511   don't care
512   don't care
513   PORTD in I/O mode
514   don't care
515   all PORTE as input
516   */
517
518
519   //-------A/D converter
520   ADCON1bits.VCFG1=0;  // VCFG1:VCFG0 Voltage Reference Configuration bit (VREF- source) 0=Vss
521   ADCON1bits.VCFG0=0;  //             Voltage Reference Configuration bit (VREF+ source) 0=Vdd
522   ADCON1bits.PCFG3=0;  // PCFG3:PCFG0 A/D Port Configuration Control bits
523   ADCON1bits.PCFG2=1;  //             From AN0 to AN7 Analog ports.
524   ADCON1bits.PCFG1=1;  //             From A8 to AN12 Digital ports.
525   ADCON1bits.PCFG0=1;
526   ADCON2bits.ADFM=0;   // la lettura e' giustificata a sinistra, gli 8 bit piu' significativi sono
527                        // in ADRESH gli ultimi 2 bit (che possono non essere utilizzati se sono
528                        // sufficienti 256 livelli) sono i bit 7 e 6 di ADRESL
529   ADCON2bits.ADCS2=1;  // ADCS2:ADCS0 A/D Conversion Clock Select bits
530   ADCON2bits.ADCS1=1;
531   ADCON2bits.ADCS0=1;
532   ADCON2bits.ACQT2=0;  // ACQT2:ACQT0 A/D Acquisition Time
533   ADCON2bits.ACQT1=0;  // ACQT2:ACQT0 = 000 Manual acquisition Time
534   ADCON2bits.ACQT0=0;  // compatible with device without acquisition time.
535   ADCON0bits.ADON=1;   // modulo A/D acceso
536
537
538
539   //-------Interrupts
540
541   RCONbits.IPEN=1;        // abilitata priorita' interrupt, non compatibile con 16fxx
542   INTCONbits.GIEH=0;      // disabilitati tutti gli interrupt high
543   INTCONbits.GIEL=0;      // disabilitati tutti gli interrupt low
544
545
546   //-------Interrupt su PortB
547
548   INTCONbits.INT0IE=1;    // INT0 abilitato, Keypad
549   INTCON3bits.INT1IE=1;   // INT1 abilitato, PortB1 encoder ticks
```

```
550   INTCON3bits.INT2IE=1;   // INT2 abilitato, PortB2 encoder ticks
551
552                           // INT0 sempre ad alta priorita'
553   INTCON3bits.INT1IP=0;   // INT1 bassa priorita'
554   INTCON3bits.INT2IP=0;   // INT2 bassa priorita'
555
556   INTCON2bits.INTEDG0=0;  // INT2 sul fronte di discesa
557   INTCON2bits.INTEDG1=1;  // INT1 sul fronte di salita (se verso errato cambiare in 0)
558   INTCON2bits.INTEDG2=1;  // INT2 sul fronte di salita
559
560   INTCONbits.RBIE=0;      // interrupt on change PortB disabled
561
562
563   //-------Timer0
564
565   TMR0L -= 156;
566   /*interrupt ogni 1mSec
567   periodo = CLKOUT * prescaler * TMR0
568     1mSec =~ 1/(20.000.000/4) * 32 * 156 @ 20MHz
569     1mSec =~ 1/(40.000.000/4) * 64 * 156 @ 40MHz
570   */
571
572   #ifdef PLL // 40MhZ
573      T0CON  = 0b11000101;
574   #else
575      T0CON  = 0b11000100;
576   #endif
577
578   /*
579   TMR0ON   TMR0 acceso
580   T08BIT   Timer a 8 bit
581   T0CS     Timer on internal clock
582   T0SE     increment on low-to-high transition
583   PSA      Prescaler assegnato
584   prescaler 1:32 per TMR0 = 1ms con clock a 20MHz
585   prescaler 1:64 per TMR0 = 1ms con clock a 40MHz
586   */
587
588   INTCONbits.TMR0IE=1;       // interrupt on TMR0 overflow enabled
589   INTCON2bits.TMR0IP=1;      // TMR0 interrupt high priority
590
591
592   //-------USART
593
594   #ifdef PLL // 40MhZ
595      SPBRG=129;              //19200 @ 40MHz
596   #else
597      SPBRG=64;               //19200 @ 20MHz
598   #endif
599
```

```
600   TXSTAbits.BRGH=1;               //High baud rate
601   TXSTAbits.SYNC=0;               //asynchronous
602   RCSTAbits.SPEN=1;               //enable serial port pins
603   RCSTAbits.CREN=1;               //enable reception
604   RCSTAbits.SREN=0;               //no effect
605   PIE1bits.TXIE=0;                //disable tx interrupts
606   PIE1bits.RCIE=0;                //disable rx interrupts
607   TXSTAbits.TX9=0;                //8-bit transmission
608   RCSTAbits.RX9=0;                //8-bit reception
609   TXSTAbits.TXEN=0;               //reset transmitter
610   TXSTAbits.TXEN=1;               //enable the transmitter
611
612
613   //-------PWM
614
615   CCP1CON=0b00001100;  //ccpxm3:ccpxm0 11xx=pwm mode (compatible mode)
616   CCP2CON =0b00001100; //ccpxm3:ccpxm0 11xx=pwm mode
617   T2CON = 0X01;        //STOP TIMER2 registers to POR state + prescaler = 4
618   PR2 = 0x7F;          // PWM freq = 19KHz @ 40 MHz
619
620   T2CONbits.TMR2ON = 1;// Turn on PWM
621
622   //-------I2C
623
624   OpenI2C(MASTER, SLEW OFF); //Master I2C, slew rate off per clock=100KHz
625   SSPADD=99;                 //100KHz baud clock @ 40MHz
626                              //SSPAD=((Fosc/BitRate)/4)-1=(400/4)-1=99
627   PIE1bits.SSPIE=1;          // SSP (I2C events) int enabled
628   PIE2bits.BCLIE=1;          // BUS COLLISION int enabled
629   IPR1bits.SSPIP=0;          // SSP int = low priority
630   IPR2bits.BCLIP=0;          // BUS COLLISION int = low priority
631
632
633   //-------Interrupts non usati
634
635   PIE1bits.TMR2IE=0;         // interrupt on TMR2 overflow disabled
636   PIE1bits.TMR1IE=0;         // interrupt on TMR1 overflow disabled
637   PIE2bits.TMR3IE=0;         // interrupt on TMR3 overflow disabled
638   PIE1bits.PSPIE=0;          // PSP int disabled
639   PIE1bits.ADIE=0;           // AD int disabled
640   PIE1bits.RCIE=0;           // USART RX int disabled
641   PIE1bits.TXIE=0;           // USART TX int disabled
642   PIE1bits.CCP1IE=0;         // CCP1 int disabled
643   PIE2bits.CCP2IE=0;         // CCP2 int disabled
644   PIE2bits.EEIE=0;           // EEPROM int disabled
645   PIE2bits.LVDIE=0;          // LOW VOLTAGE int disabled
646
647
648   //-------Interrupts
649         //INTCON3bits.INT1IF       // interrupt flag encoder SX
```

```
650         //INTCON3bits.INT2IF       // interrupt flag encoder DX
651         //INTCONbits.INT0IF        // interrupt on INT0
652         //PIR1bits.TMR2IF          // interrupt on TMR2 overflow
653         //PIR1bits.TMR1IF          // interrupt on TMR1 overflow
654         //PIR1bits.PSPIF           // PSP int
655         //PIR1bits.ADIF            // AD int
656         //PIR1bits.RCIF            // USART RX int
657         //PIR1bits.TXIF            // USART TX int
658         //PIR1bits.SSPIF           // SSP int
659         //PIR1bits.CCP1IF          // CCP1 int
660         //PIR2bits.CCP2IF          // CCP2 int
661         //PIR2bits.EEIF            // EEPROM int
662         //PIR2bits.BCLIF           // BUS COLLISION int
663         //PIR2bits.LVDIF           // LOW VOLTAGE int
664         //PIR2bits.TMR3IF          // interrupt on TMR3 overflow
665
666   }
667   /*fine settings*************************************************************************/
668
669   /* ////////////////////////////////////////////////////////////////////////////////////
670   ** Continua descrizione:
671   **
672   ** Processore PIC 18F452, controller board Mark III
673   **
674   **   e' previsto l'utilizzo dei seguenti sensori:
675   **      4 sensori di prossimita' IR sharp GP2D120        4 I   (EyeRf,EyeLf,EyeRl,EyeLl)
676   **                                                             AN2-AN3-AN5-AN7
677   **      2 sensori d'urto a microswitch (bumper)          2 I   (BumperR, L)       D2-D3
678   **      2 shaft encoder in quadratura                    4 I   (EncoderRdir,L)    B1-B2
679   **                                                             (EncoderRpulse,L) B3-B4
680   **      3 Sensori di luce                                3 I   (FrL,FrC,FrR)
681   **                                                             AN0-AN1-AN6
682   **      Sensore di suono                                 1 I   (Sound)             D1
683   **      Sensore di gas                                   1 I   (Gas)               AN4
684   **
685   **   e dei seguenti attuatori:
686   **    H-bridge motor driver                             3 O   (PWMmotR, L)       C1-C2
687   **                                                             (MotEnable)       D0
688   **      Buzzer                                          1 O                     C5
689   **   delle porte Rx-Tx                                  2 I/O
690   **
691   **   oltre ai seguenti dispositivi I2c (2 I/O SCL-SDA)
692   **      Ultrasonic range finder Devantech SRF08         Adr =   0xE0 (224)
693   **      256Kbit I2C EEprom                              Adr =   0x00 (000)
694   **      I2C 8bit I/O expander PCF8574                   Adr =   0x40 (064)
695   **      I2C 8bit I/O expander PCF8574                   Adr =   0x42 (066)
696   **      Analog input expander                           Adr =   0x50 (080)
697   **      Digital compass Devantech CMPS03                Adr =   0xC0 (192)
698   **
699   **
```

```
700   ** A0(AN0) FrR                      I   analog
701   ** A1(AN1) FrL                      I   analog
702   ** A2(AN2) EyeRf                    I   analog
703   ** A3(AN3) EyeLf                    I   analog
704   ** A4      ------------------
705   ** A5(AN4) Gas                       I   analog
706   ** B0      I/O expander interrupt    I
707   ** B1      EncoderRpulse             I
708   ** B2      EncoderLpulse             I
709   ** B3      ------------------
710   ** B4      EncoderRdir               I
711   ** B5      EncoderLdir               I
712   ** B6      PGC (ICD2)
713   ** B7      PGD (ICD2)
714   ** C0      ------------------
715   ** C1      PWMmotR                   O   PWM
716   ** C2      PWMmotL                   O   PWM
717   ** C3      SCL                       I
718   ** C4      SDA                       I
719   ** C5      Buzzer                    O
720   ** C6      TX                        O
721   ** C7      RX                        I
722   ** D0      MotEnable                 O
723   ** D1      Sound                      I
724   ** D2      BumperR                   I
725   ** D3      BumperL                   I
726   ** D4      LedRosso----------        O  temp
727   ** D5      LedVerde----------        O  temp
728   ** D6      LedGiallo---------        O  temp
729   ** D7      LCD enable                O
730   ** E0(AN5) EyeRl                     I   analog
731   ** E1(AN6) FrC                       I   analog
732   ** E2(AN7) EyeLl                     I   analog
733
734   ------------------------------------------------------------------------------------
735   lettura sensori  IR Sharp GP2D120
736   distanza dal paraurti
737
738   distanza    cm:25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02
739   lettura AD hex:10 11 13 14 16 18 1a 1e 20 24 26 2b 30 33 36 3b 3f 46 4e 5a 66 77 8d a2
740
741   sotto i 2cm la  tensione ricomincia a  scendere.
742   ------------------------------------------------------------------------------------
743
744   ------------------------------------------------------------------------------------
745   Descrizione delle metodologie generali di programmazione
746
747   Il programma lavora in una versione semplificata di "Real Time Operating System",
748   sono state seguite quindi le seguenti regole per lo sviluppo di un
749   "Sistema Operativo Multitasking Cooperativo":
```

```
750
751   -Ogni routine  collabora con tutte le altre, ed in particolare con il main,
752      per il buon funzionamento del sistema.
753
754   -Ogni routine occupa il sistema per il minor tempo possibile
755
756   -Non ci sono loop lunghi o ritardi a SW
757
758   -Il main si preoccupa di schedulare i vari task, le routine si chiamano raramente tra loro
759
760   -Lo scambio di informazioni tra task avviene tramite flag e variabili globali
761
762   -Le temporizzazioni sono realizzate usando come "Real Time Clock" l'interrupt overflow
763      del TIMER0 che e' il vero "hearth-beat" del sistema
764
765   -L'esecuzione dei vari task e' condizionata dai relativi semafori, il main o un altro task
766      possono disabilitare una certa funzione
767
768   ------------------------------------------------------------------------------------
769   Descrizione delle procedure di movimento:
770
771   Il cuore di tutto e' il timer (TMR0) che ogni 1mSec genera un interrupt.
772   Nella ISR vengono incrementati i timer (a 8, 16, 24 bit secondo la durata necessaria)
773   che servono per le temporizzazioni delle varie routine.
774
775   Anche il computo dello spazio percorso tramite encoder ottici in quadratura, funziona ad
776   interrupt,ognuno dei due encoder genera il proprio interrupt che, analizzato e confrontato
777   con il flag di direzione, incrementa o decrementa le variabili di conteggio dello spazio.
778
779   Nella stessa ISR viene calcolata, ogni 50mSec, la velocita' di avanzamento attuale,
780   dividendo lo spazio percorso per il tempo.
781
782   Il main e' semplicemente uno scheduler che lancia le diverse routine con le giuste priorita'.
783
784   Una di queste e' la MotSpeed che lavora in background e, ogni 50 mSec (dopo la misura della
785   velocita'), imposta il PWM (calcolato tramite un algoritmo PID) per fare in modo che la
786   velocita' attuale corrisponda con quella desiderata, .
787
788   A questo punto abbiamo un sistema a loop chiuso nel quale sappiamo la velocita', lo spazio
789   percorso ed il tempo trascorso.
790
791   Anche la routine Path lavora in background e, se abilitata, controlla la sequenza di
792   operazioni da effetuare per seguire il percorso impostato da qualche altra routine che
793   decide il comportamento del bot.
794   La routine che decida di eseguire un percorso diverso dal normale, deve:
795   - impostare il flag di attivazione della routine Path
796   - scrivere nell'array PathSeq la sequenza di codici della manovra da effettuare
797      (0=fine sequenza)
798   - azzerare il puntatore della sequenza
799   Path comincera' allora a scandire l'array chiamando le routine in funzione del codice
```

```
800   impostato (cammina X cm, ruota Y gradi, cammina W mm/sec).
801   La routine chiamata (Walk, Turn) calcola lo spazio che deve percorrere ogni ruota prima
802   di ripassare il comando a Path.
803   A questo punto si e' attivata Space2Run (schedulata da main) che si disattiva, ripassando
804   il controllo a Path, solo quando le ruote si sono mosse dello spazio desiderato.
805   Path controlla la sequenza successiva ripetendo le operazioni o restituendo il controllo
806   quando il codice e' = 0.
807
808   Il Main e' sempre attivo e, visto che le operazioni di movimento sono molto lente rispetto
809   ai tempi di esecuzione del PIC, per la maggior parte del tempo sara' occupato dall'interrupt
810   degli encoder:
811   a 36cm/s le ruote fanno 2 giri/s, l'encoder fa 40 giri/s, con 40 finestrelle genera
812   1600 impulsi/s (solo fronte di salita) -> 3200 tick/s da entrambi gli encoder
813   -> un impulso ogni 312,5us.
814   Con il clock a 40MHz (100ns a istruzione) -> 3125 istruzioni tra un interrupt e l'altro.
815
816
817   ------------------------------------------------------------------------------------
818   Descrizione delle procedure di comunicazione I2C
819
820   Indirizzi devices I2c
821   // #  device   descrizione    |fix   | |free| |R/W|   |adr|
822   // 0  24LC256  32KB eeprom    0 0 0 0  0 0 0   x      0x00
823   // 1  CMPS03   compass        1 1 0 0  0 0 0   x      0xC0
824   // 2  MAX127   A/D expander   0 1 0 1  0 0 0   x      0x50
825   // 3  PCF8574  I/O expander   0 1 0 0  0 0 0   x      0x40
826   // 4  PCF8574  I/O expander   0 1 0 0  0 0 1   x      0x42
827   // 5  SRF08    Devantech US   1 1 1 0  0 0 0   x      0xE0
828
829   Analizzando i sorgenti delle librerie I2c del C18 Microchip si nota che alcune contengono
830   dei waiting loop in attesa dell'esecuzione dell'operazione da parte della SSP e non sono
831   quindi compatibili con il sistema pseudo-RTOS
832
833   ReadI2C     wait until byte received
834   getcI2C     idem
835   getsI2C     perform getcI2C() for 'length' number of bytes
836               check that receive sequence is over
837               wait until ACK sequence is over
838   IdleI2C     Test and wait until I2C module is idle
839   WriteI2C    wait until write cycle is complete
840   putI2C      idem
841   putsI2C     transmit data until null character
842               test for idle condition
843               wait until ninth clock pulse received
844
845   altre invece possono essere usate
846   AckI2C      ok
847   CloseI2C    ok
848   DataRdyI2C  ok
849   NotAckI2C   ok
```

```
850   OpenI2C     ok
851   RestartI2C  ok
852   StopI2C     ok
853   StartI2C    ok
854
855   Tutte le funzioni relative alla eeprom contengono molti wait on idle
856
857   Tutte le funzioni relative agli LCD contengono molti delay
858
859   Le procedure non utilizzabili sono quindi state riscritte in modo compatibile  usando gli
860   interrupt generati dall'SSP, con flag di stato e variabili globali di appoggio.
861
862   Un'ottima traccia di partenza e' stata l'application note AN736 di Microchip che descrive
863   in modo molto dettagliato questo tipo di tecnica.
864
865   La comunicazione tramite I2c avviene a diversi livelli.
866
867   -Strato applicativo: ogni device I2c ha le sue caratteristiche,
868      la routine relativa al device ha a disposizione 4byte in TX e 2byte in RX per ogni turno.
869      Ad esempio: verso la EEPROM si trasmetteranno due byte per l'indirizzo della locazione
870      di memoria e si riceveranno o trasmetteranno uno o due byte (variabile char o int).
871      E' questo il device che richiede piu' byte in trasmissione e determina quindi la dimensione
872      del buffer.
873      La funzione I2cDeviceXX prepara i byte da scambiare e alza i flag relativi alle azioni
874      da svolgere:
875      I2c[I2cDevPtr].Flag.Rx
876      I2c[I2cDevPtr].Flag.Tx .
877
878
879   -Strato I2cHighLevel:
880      E' eseguito se il deviceXX ha qualcosa da scambiare
881      I2c[I2cDevPtr].Flag.Rx !=0 || I2c[I2cDevPtr].Flag.Tx!=0
882      e se la routine a livello piu' basso e' disponibile:
883      I2cBusyFlag = 0
884
885      scambia i byte con il device attraverso la routine a livello piu' basso alza il I2cBusyFlag
886
887
888   -Strato I2cLowLevel:
889      E' eseguito se la ISR ha alzato I2cEventFlag
890
891      START->ADRW->                          se ha qualcosa da trasmettere, altrimenti Rx
892      WRITE byte[0]->...WRITE byte[n]->
893      RSTART->ADRR->
894      READ byte[0]->ACK...READ byte[n]NACK->
895      STOP
896      FINE
897
898
899   Nel  ciclo main sono controllati i flag a partire da quelli a livello piu' basso a salire.
```

```
L'intero ciclo e' ripetuto  per ogni device I2c scandendo il buffer

1-Evento I2c ? YES -> esegui I2cLowService -> EXIT
   |
   V
   NO
2-I2cBusyFlag ? YES -> EXIT
   |
   V
   NO
3-C'e' qualche byte da trasmettere e/o da ricevere ? -> YES -> I2cHighService -> EXIT
   |
   V
   NO
4-Ci sono ancora record da controllare nel buffer I2c ? -> YES -> Incrementa contatore I2cDevPtr  -> EXIT
   |
   V
   NO
5-Azzera contatore I2cDevPtr -> EXIT

descrizione
1-L'evento I2c e' comunicato da SSP tramite ISR
   I2cLowService esegue sequenza I2c per scambiare singolo byte
      azzera flag evento o collisione I2c
      scambiato l'intero byte azzera flag I2cBusyFlag per abilitare  la routinee high

2-I2cBusyFlag e' alzato da I2cHighService e azzerato da I2cLowService

3-La routine relativa al singolo  device I2c ha riempito il buffer ed alzato i flag
   I2cHighService inizializza sequenza I2cLowService, questa scambia ogni singolo byte
      ad ogni byte scambiato e' decrementato il contatore relativo
      quando contatore TX e' a zero si inizia la ricezione
      quando entrambi i contatori sono a zero si passa al device successivo
   solo se entrambi i contatori sono a zero la routine del singolo device, quella a
      livello piu' alto, può cominciare a scambiare altri byte

4-E' stato scandito tutto il buffer, al prossimo giro si ricomincia dal primo

5-Si prepara a controllare il device successivo al prossimo giro


//////////////////////////////////////////////////////////////////////////////// */

/* Version History ////////////////////////////////////////////////////////////


0.1 - inizio lavori 19 luglio 2003

0.2 - 16 agosto 2003
prime prove di movimento, prime implementazioni dell'algoritmo PID
```

```
950
951   0.3 - 14 aprile 2004
952   prima versione dimostrabile e pubblicata sul sito. Implementate le funzioni di movimento di base.
953   Il PID funziona bene, riesco a far muovere il bot ad una precisa velocita' e farlo girare dell'angolo
954   che voglio come dimostrato col programma"quadrato.c" e col video pubblicato sul sito.
955   Implementato riconoscimento ostacoli con bumper ed algoritmo per uscire dagli angoli morti
956
957   0.4 - inizio 30 novembre 2004
958   Cambiato boot loader con versione di Peter Huemer, Shane Tolmie e Pic Bootloader+ di Herman Aartsen.
959   Questo ha un ritardo di soli 0.2 secondi all'accensione che non provocano movimento inutile del bot
960   all'avvio (5 sec nella versione inizialmente caricata nel MarkIII).
961   Ho modificato la versione originale per fare in modo che i motori non girassero durante la programmazione:
962   Motenable = 0
963   Implementazione riconoscimento ostacoli tramite sensori infrarossi Sharp GPxxx.
964   Cambiati sensori originali a lungo raggio perche' iniziano la misura da 20cm. Vanno meglio i GP2D120 che
965   partono da 4cm. Piu' che sufficiente la distanza massima di 40cm.
966   Implementate le funzioni di lettura A/D.
967
968   0.41 - inizio 17 gennaio 2005
969   Dopo il cambio di motori e alcune prove sul campo a Pisa, ho cambiato il rapporto di riduzione degli
970   ingranaggi per aumentare la velocita' di esplorazione del campo. Da 269:1 sono passato a 101:1,
971   passando da una velocita' teorica massima di 260cm/s ad una di 690cm/s, i motori hanno ancora sufficiente
972   potenza e questo mi dovrebbe permettere un buon aumento di velocita' con ancora un alto margine di controllo.
973   28-01-2005
974   Ho tolto i sensori "LineX" sul paraurti, inutili per un robot explorer e ho aggiunto altri 2 sensori IR Sharp
975   GP2D120 per rivelare ostacoli frontali usando le porte A/D precedentemente assegnate ai sensori di linea.
976   Ho aggiunto anche tre fotoresistenze al CdS per rivelare le sorgenti di luce
977
978   0.42 - inizio 21 marzo 2005
979   modificato algoritmo per uscire dagli angoli morti per adattarlo ad un diverso numero e tipo di sensori
980
981   0.5 - 28 marzo 2005
982   Trovati falsi allarmi dovuti ai sensori IR che restituiscono impulsi quando sono investiti dalla sorgente di luce.
983   Inserito debounce su rivelazione luce per diminuire falsi allarmi.
984   Quando comincia a rivelare la luce diminuisce la velocita', altrimenti i tempi di intervento sono troppo ristretti
985   e si rischia di perdere la sorgente o di sbatterci contro
986
987   0.6 - 1 maggio 2005
988   Aggiunto circuito per la rivelazione del tono a 4KHz e relativa routine di gestione. Le logiche di rivelazione,
989   debounce e segnalazione sono simili a quelle usate per la rivelazione delle sorgenti luminose
990
991   1.0 - 7 maggio 2005
992   Con questa versione Dino ha partecipato alla prima gara all'ITIS Von Neumann con un discreto piazzamento.
993   Modificato tra la prima e la seconda manche per disabilitare la ricezione del suono mentre e' in pausa dopo aver
994   rivelato la luce, altrimenti c'e' il rischio che entri in un loop suono-luce se rivela il tono insieme alla luce.
995   Tarate soglie di luce per il campo di gara. Aumentato delay dopo "TargetOK".
996
997   1.1 - 10 maggio 2005
998   Disabilitato sensore suono anche durante beep per evitare che riveli il suo tono come segnale a 4KHz
999   Regolati ritardi, soglie e velocita' min e MAX
```

```
1000
1001  Ha partecipato anche alla gara al Pacinotti, problema sull'HW del suono nella seconda manche, risolto.
1002
1003  2.0 - 1 giugno 2005
1004  Installato bootloader "tiny boot loader" adatto per Pic18, occupa solo 100byte ricompilabile per la frequenza
1005  di clock voluta.
1006  Iniziato porting su PIC18F452 con compilatore Microchip C18 e clock a 40MHz
1007
1008  18 1.0 - 8 giugno 2005
1009  Porting finito in imitazione 16f877 anche con clock a 40MHz
1010
1011  18 2.0  - 13 giugno 2005
1012  Modificata gestione interrupt sfruttando le caratteristiche del 18f452
1013  TMR0 e' gestito come interrupt ad alta priorita'.
1014  Encoder gestiti come interrupt a bassa priorita' su INT1 e INT2, solo il fronte di salita genera interrupt,
1015  sono quindi dimezzati gli interrupt generati rispetto alla versione precedente.
1016  Non cambia il cablaggio rispetto alle versioni precedenti, sono solo invertiti EncoderXPulse con EncoderXDir.
1017
1018  18_2.1 - 17 luglio 2005
1019  Iniziato sviluppo comunicazione I2C
1020  Nelle procedure di movimento sono state ricalcolate le costanti ragionando in base 2, in  questo modo si sono
1021  sostituite le divisioni con gli shift, risparmiando centinaia di microsecondi a procedura.
1022
1023  18 2.2 - 20 agosto 2005
1024  Sviluppate procedure comunicazione I2C
1025  Sviluppato HW e SW per I/O expander per pilotare LCD e tastiera
1026
1027  18 2.3 - 3 settembre 2005
1028  Sviluppate procedure per la scrittura su display LCD
1029
1030  18 2.4 - 5 settembre 2005
1031  Cambiata procedura inizializzazione, il ciclo idle prima della partenza e' gia
1032  nell'RTOS.
1033  Iniziato sviluppo routine rivelazione gas
1034
1035  18 2.41 - 12 settembre 2005
1036  Terminate procedure rivelazione gas
1037  Lcd enable collegato a porta RD7 e non piu' a I/O expander per problema temporizzazioni:
1038  lo strobe all'LCD (En=1, En=0) deve essere eseguito quando il dato e' stabile sul bus,
1039  il PCF8574 garantisce la stabilita' dei segnali sulla porta d'uscita solo dopo 4uS
1040  dall'ultimo colpo di clock dopo l'ACK.
1041  Capitava spesso che il dato sull'LCD non fosse corretto, facendolo impazzire.
1042
1043  18 2.42 - 1 novembre  2005
1044  Ottimizzate procedure LCD.
1045  Cambiata Interrupt Service Routine ad alta priorita':
1046     Time keeping esterno all'ISR
1047     Interrupt dell'I/O  expander 2 (Keypad) su INT0
1048  Sviluppate procedure di base per keypad.
1049  Tolto bootloader. programmazione e debug con ICD2
```

```
1050
1051  18 2.43 - 16 novembre  2005
1052  Ottimizzata procedura interpretazione e debounce tasti.
1053
1054  18 2.44 - 17 novembre 2005
1055  Suddiviso codice in piu' file per facilitarne la scrittura
1056
1057  18 2.45 - 3 dicembre 2005
1058  Ottimizzato main raggruppando le user interfaces
1059     (LCD, keypad, ecc...) sotto un unico semaforo.
1060  Finita gestione display e tastiera con menu visualizzazione delle variabili
1061     e modifica e salvataggio in EEPROM delle costanti
1062
1063  18 2.50 - 01 gennaio 2006
1064  Porting su 18F4525
1065     il grosso del cambiamento e' sulla configurazione dell'A/D converter,
1066     nell'utilizzo non e' cambiato niente
1067  Finite procedure bussola
1068     lettura bussola (1 byte -> 0-360° = 0-255) ogni 100 ms
1069     calibrazione automatica bussola sui quattro punti cardinali
1070
1071  18 2.51 - 01 gennaio 2006
1072  Inizio procedure odometria
1073
1074  ////////////////////////////////////////////////////////////////////////////// */
1075
1076  /*fine descrizioni ////////////////////////////////////////////////////////////// */
1077
1078
1079
```