/*** CONTROLLO BRANDEGGIO Motore303)
per Antenna SELFSAT, con ARDUINO-UNO ***
********** realizzazione windom46 **********

Pilotaggio display LCD 162B Displaytech e controllo brandeggio con bussola CMPS03
(in modalita' PWM), accelerometro ADXL355 (analogico a 3 assi) per l'elevazione e
SatFinder per rilevamento livello segnale antenna.
Ogni lettura dell'accelerometro verra ripetuta diverse volte(buffer) e poi verra fatta
la loro media. Questo in quanto la lettura singola si è rivelata molto instabile.
Si potrebbe pure collegare un GPS sulla seriale e fare il calcolo dei valori di
orientamento, elevazione e skew in base a latitudine e longitudine del posto ma ho visto
che le ultime genarazioni di cellulari hanno già programmi dedicati(SatfinderPro/Android).
Vengono usati tutti i Pin, 6 per il display, 4 per i 2 motori, 1 per il fine corsa
orientamento, 1 per il rilevamento motore camper acceso, 1 per il rilevamento segnale
bussola, 1 per il rilevamento segnale accelerometro, 1 per il livello segnale SatFinder
e 1 per memorizzare i valori di direz./elevaz. nella EEPROM.
In totale 12 digitali e 6 analogici.(0/1 della seriale,liberi per eventuale GPS).

Circuito collegamento LCD :
 * Pin 1-4 al positivo 5V
 * Pin 2-3 a massa
 * RW pin 7 collegato a massa
 * RS pin 6 su digital pin 7
 * Enable pin 8 su digital pin 6
 * D4 pin 13 su digital pin 5
 * D5 pin 14 su digital pin 4
 * D6 pin 15 su digital pin 3
 * D7 pin 16 su digital pin 2
 * VO pin 5 su cursore potenziom. 10K
 * tra +5V e massa (regola contrasto)

 Library originally added 18 Apr 2008
 by David A. Mellis
 library modified 5 Jul 2009
 by Limor Fried (http://www.ladyada.net)
 example added 9 Jul 2009
 by Tom Igoe
 modified 25 July 2009
 by David A. Mellis
 http://www.arduino.cc/en/Tutorial/LiquidCrystal
 */

```
#include <EEPROM.h>                          // include codice librerie
#include <LiquidCrystal.h>
#include <avr/wdt.h>
#define Reset_AVR() wdt_enable(WDTO_30MS)    // softReset PIC
LiquidCrystal lcd(7, 6, 5, 4, 3, 2); // initializza libreria LCD
int levelSF;                                 // livello segnale del SatFinder
int buttonState = LOW;                       // imposta stato pulsanti man./auto
int oldState = LOW;                          // imposta stato bistabile
float dirIn = 0;                             // lettura bussola in gradi
char level = -1;                             // barra segnale SatFinder
int dirHB = 162;                             // imposta direzione HotBird
int elevHB = 38;                             // impostazione elevazione HotBird
int elevRif = elevHB+5;                      // elevazione di sicurezza
int range = 2;                               // tolleranza su angolazioni
int buffer = 16;                             // buffer letture accelerometro
int dirLock = 0;                             // direzione OK
int elevLock = 0;                            // elevazione OK
float elevIn = 0;                            // valore elevazione(ADXL355)
float compass = 0;                           // valore orientamento(CMPS03)

void setup(){
pinMode(A0, INPUT);                          // ingresso accelerome.(elevazione)
pinMode(A1, INPUT);                          // pulsante man./auto
pinMode(A2, OUTPUT);                         // LED man./auto
pinMode(A3, INPUT);                          // sensore motore acceso
pinMode(A4, INPUT);                          // ingresso livello SatFinder
pinMode(A5, INPUT);                          // pulsante salva valori elev./direz.
pinMode(8, INPUT);                           // ingresso bussola (direzione)
pinMode(9, OUTPUT);                          // motore elevazione 9/10
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);                         // motore orientamento 11/12
pinMode(12, OUTPUT);
pinMode(13, INPUT);                          // micro fine corsa rotazione

lcd.begin(16, 2);                            // setup LCD (num.colonne/righe)
//Serial.begin(9600);                        // inizializza porta seriale

}
void loop() {
int dirHB = EEPROM.read(0);                  // legge direziona dalla cella 0
int elevHB = (EEPROM.read(1)-2);             // legge elevazione dalla cella 1
```

```
int elevRif = elevHB+5;                     // elevazione di sicurezza
compass = pulseIn(8, HIGH);                 // lettura bussola
dirIn = (compass/1000-1)*10.15;             // transforma in gradi
buttonState = digitalRead(A1);              // rileva pressione pulsante man./auto

// elevazione, fa la media delle letture dell'accelerometro impostate in buffer
    elevIn = 0;
    for (int i=0; i<buffer; i++) {
    elevIn += analogRead(A0);
    delay(3);
}
elevIn = (398-(elevIn/buffer))*0.9;         // valore medio di elevazione, formula
                                            // da adattare al tipo di accelerometro

//  ********* BISTABILE COMMUTAZIONE MAN./AUTO ******************
  if (buttonState == HIGH) {
    if (oldState == LOW) {
      oldState = HIGH;                      // attiva automatismi e
      digitalWrite(A2, HIGH);               // accende LED
      } else {
      Reset_AVR();                          // resetta PIC
      }
    }
//  *********** CHIUDE ANTENNA A MOTORE ACCESO ******************
   if (digitalRead(A3) == HIGH) {           // motore camper acceso
   oldState = LOW;                          // disattiva automatismi e
   digitalWrite(A2, LOW);                   // spegne LED
     if (elevIn < elevRif) {                // chiude antenna se elevIn < elevRif
     digitalWrite(11, LOW);
     digitalWrite(12, HIGH);
     digitalWrite(9, LOW);                  // abbassa antenna
     digitalWrite(10, HIGH);
     } else {
    digitalWrite(11, LOW);                  // chiude antenna (orario)
    digitalWrite(12, HIGH);
    digitalWrite(9, LOW);                   // ferma abbassamento sino a
    digitalWrite(10, LOW);                  // chiusura completa
     }
     if (digitalRead(13) == LOW) {          // abbassa completamente antenna
       digitalWrite(9, LOW);
       digitalWrite(10, HIGH);
       }
   } else {
if (oldState == HIGH) {                     // modo auto, abilitato dal pin A3
// ****** ORIENTAMENTO PARABOLA (direzine/elevazione) AI VALORI IMPOSTATI ******
if (elevIn > elevRif) {                     // alza antenna
   digitalWrite(9, HIGH);
   digitalWrite(10, LOW);
   } else {                                 // orienta ant.se elevIn < elevRif
   if ((dirLock == 0) || (dirIn > dirHB + range)) {
     digitalWrite(11, HIGH);                // apre antenna (antiorario)
     digitalWrite(12, LOW);
     }
   if ((dirLock == 1) && (dirIn < dirHB - range)) {
     digitalWrite(11, LOW);                 // chiude antenna (orario)
     digitalWrite(12, HIGH);
     }
   if ((dirIn > dirHB - range) && (dirIn < dirHB + range))  {
     digitalWrite(11, LOW);                 // ferma rotazione antenna
     digitalWrite(12, LOW);
     dirLock = 1;                           // direzione OK
     }
   if (elevIn > elevHB + range) {           // alza antenna
     digitalWrite(9, HIGH);
     digitalWrite(10, LOW);
     }
   if (elevIn < elevHB - range) {           // abbassa antenna
     digitalWrite(9, LOW);
     digitalWrite(10, HIGH);
     }
   // mantiene l'antenna entro il valore di elevazione e range impostato
   if ((elevIn < elevHB + range) && (elevIn > elevHB - range)) {
     digitalWrite(9, LOW);
     digitalWrite(10, LOW);
     elevLock = 1;                          // elevazione OK
     }
   }
 }
}
// se raggiunge elevazione/direzione impostati o se aggancia segnale SAT, resetta PIC
if ((elevLock == 1) && (dirLock == 1) || (analogRead(A4) > 200) && (elevLock+dirLock > 0)) {
  Reset_AVR();                              // resetta PIC e passa a modo manuale
  }
if (digitalRead(A5) == HIGH) {              // salva valori dir./elev in memoria (EEPROM)
EEPROM.write(0, dirIn);                     // memorizza direzione nella cella 0
EEPROM.write(1, elevIn);                    // memorizza elevazione nella cella 1
```

```
}
/* ********************************************************************************
Qui bisognerebbe sviluppare una routine per il posizionamento fine, con spostamenti di
qualche grado di orientamento ed elevazione e abbinare i valori massimi rilevati
dal SatFinder ai dati forniti da accelerometro e bussola, per poi posizionarsi su
queste angolazioni. Usando per il movimento brandeggio, motori stepper con encoder,
il centraggio automatico del satellite, si potrebbe fare in modo perfetto.
Sfruttando 2 assi dell' accelerometro (x/y) si potrebbe controllare anche lo Skew.
IMPORTANTE !!! E' fondamentale fare un'ottima schermatura tra i vari componenti in
modo che specialmente l'accelerometro non sia influenzato da cause esterne. Aggiungere
condensatori da 100Kpf verso massa sui 4 comandi motori e sull' uscita accelerometro.
Durande i posizionamenti, in presenza di disturbi, e' facile avere letture sfalsate
anche di parecchi gradi con conseguenti continui aggiustamenti della ricerca automatica.
La bussola in modalita' PWM va abbastanza bene e meno male, questa non si puo' schermare.
Fare attenzione ai campi magnetici circostanti, altoparlanti, masse metalliche,ecc...
Gli assi x/y, ad antenna chiusa, si potrebbero usare per la messa in bolla del camper.
**********************************************************************************/

// ***** Scrive su LCD 16x2 con livello segnale *****
lcd.setCursor(0,0);
lcd.print("Elev.");
lcd.print (elevIn, 0);
lcd.print("  ");
lcd.setCursor(9,0);
lcd.print("Dir.");
lcd.print(dirIn, DEC);
lcd.print("  ");
lcd.setCursor(0,1);
lcd.print("Level ");                          // formazione barra livello segnale
  for (int lev=0; lev<=analogRead(A4)/9; lev++) {
lcd.print(level); }
lcd.print("          ");

/*
Serial.print("Direzione  ");
Serial.println(dirIn,1);
Serial.print("Elevazione ");
Serial.println(elevIn,1);
Serial.print("Level_SAT  ");
Serial.println(analogRead(A4));
Serial.println("");
*/

delay(200);                                  // attesa per prossima lettura
}                                            // *** FINE PROGRAMMA ****
```