```
/*
  Rotore_2012

 Uso ingressi
 -----------------------------------------------------------------------------
 I0  non utilizzato
 I1  non utilizzato
 I2  LCD D7
 I3  LCD D6
 I4  LCD D5
 I5  LCD D4
 I6  pulsante senso orario
 I7  pulsante senso antiorario
 I8  uscita controllo motore senso orario
 I9  non utilizzato
 I10  uscita controllo motere senso antiorario
 I11  LCD Enable
 I12  LCD RS
 I13  uscita cicalino

 A0  ingresso potenziometro puntamento manuale
 A1  ingresso potenziometro posizione antenna
 A2  non utilizzato
 A3  non utilizzato
 A4  non utilizzato
 A5  non utilizzato
 -----------------------------------------------------------------------------
 */

// include the library code:

#include <LiquidCrystal.h>
#define buzzer 13  //   uscita per buzzer

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// creo il carattere "grado"

byte grado[8] = {
  B11100,
  B10100,
  B11100,
  B00000,
  B00000,
  B00000,
  B00000,
  B00000,
};

byte tacca[8] = {

  B00000,
  B00000,
  B00000,
  B00100,
  B00100,
  B10101,
  B10101,
};

byte ago1[8] = {

  B10000,
  B10000,
  B10000,
  B10000,
  B10000,
  B10000,
  B10000,
};
```

```
byte ago2[8] = {

  B01000,
  B01000,
  B01000,
  B01000,
  B01000,
  B01000,
  B01000,
};

byte ago3[8] = {

  B00100,
  B00100,
  B00100,
  B00100,
  B00100,
  B00100,
  B00100,
};

byte ago4[8] = {

  B00010,
  B00010,
  B00010,
  B00010,
  B00010,
  B00010,
  B00010,
};

byte ago5[8] = {

  B00001,
  B00001,
  B00001,
  B00001,
  B00001,
  B00001,
  B00001,
};

int potPin = 0;           // select the input pin for the potentiometer
int potValue = 0;         // valore posizione potenziometro
int potValueVecchio=512;  // Valore "vecchio" del potenziomentro per determinare
int DirezObbiettivo=0;    // variabile che contiene la direzione in gradi verso cui puntare l'antenna
int AdcObbiettivo=0;      // variabile che contiene il valore a cui deve arrivare l'ADC che legge la
posizione
// dell'antenna

int differenza=0;         // variabile contenente la differenza tra il valore dell'ADC e la variabile
AdcObbiettivo
int isteresi=4;           // valore minimo di differenza tra ADC e Adc obbiettivo per poter comandare
il rotore

int incomingByte = 0;   // for incoming serial data
int dato;
int pippo;
char stringa[9];    // vettore per memorizzazione dati della seriale
int count = 0;    // contatore

// se c'è stata una veriazione di puntamento
// che faccia commutare il controllo al potenziometro


int rotPin = 1;  // ingresso potenziometro del rotore
int rotValue =0;

int dirAnt;  // appoggio per la direzione attuale di puntamento dell'antenna
```

```
int dirPot;  // appoggio per la direzione attuale di puntamento del potenziometro

const int tastoOrario=6;
const int tastoAntiorario=7;
const int cicalino=13;


int giraOr=0;
int giraAnt=0;

int nTacche=0;  // varibile per contenere la posizione in cui visualizzare la tacche che rappresenta
la direzione corrente dell'antenna.
int nCaratteri=0;  // varibile per contenere il numero di caratteri bianchidopo cui visualizzare la
tacche che rappresenta
//  la direzione corrente dell'antenna

int nPosTacca=0;    //  variabile che stabilisca la posizione della tacca nel carattere

// variabili che indicano la richiesta di controllo da uno dei dispositivi

boolean richTasti=false;
boolean richPot=false;
boolean richPc=false;

boolean Uscita=false;  // per uscire dai loop ctr.pot e pc


int Controllo=0;    // variabile che determina chi controlla il rotore
// 0 - Rotore fermo
// 1 - tasti orario/antiorario/memoria
// 2 - p.c.
// 3 - potenziometro

int Controllo_Old=0;    // variabile che memorizza chi controllava il rotore al ciclo loop precedente
// 0 - Rotore fermo
// 1 - tasti orario/antiorario/memoria
// 2 - p.c.
// 3 - potenziometro


void setup() {

  // Carico il carattere personalizzato ed imposto il display come 20 caratteri per 4 righe


  lcd.createChar(1, ago1);
  lcd.createChar(2, ago2);
  lcd.createChar(3, ago3);
  lcd.createChar(4, ago4);
  lcd.createChar(5, ago5);

  lcd.createChar(6, grado);
  lcd.createChar(7, tacca);

  // inizializzo il display
  lcd.begin(20, 4);

  // inizializzo la seriale
  Serial.begin(9600);

  // inizializzo i pin
  pinMode(tastoOrario, INPUT);
  pinMode(tastoAntiorario, INPUT);

  // salvo la posizione iniziale del potenziometro

  potValueVecchio=analogRead(potPin);

  digitalWrite(cicalino,HIGH);
  delay(150);
  digitalWrite(cicalino,LOW);
  delay(50);
```

```
  digitalWrite(cicalino,HIGH);
  delay(50);
  digitalWrite(cicalino,LOW);
  delay(50);
  digitalWrite(cicalino,HIGH);
  delay(150);
  digitalWrite(cicalino,LOW);
  delay(50);

  lcd.setCursor(0,0);  // pulisco la riga
  lcd.print("                    ");
  lcd.setCursor(0,0);
  lcd.print(" Rotore controllato ");
  lcd.setCursor(0,1);  // pulisco la riga
  lcd.print("                    ");
  lcd.setCursor(0,1);
  lcd.print("  con Arduino 2009  ");
  lcd.setCursor(0,2);  // pulisco la riga
  lcd.print("                    ");
  lcd.setCursor(0,2);
  lcd.print("      Ver. 0.1      ");
  lcd.setCursor(0,3);  // pulisco la riga
  lcd.print("                    ");
  lcd.setCursor(0,3);
  lcd.print("  Copyleft IK5PWS   ");







  delay(4000);

  lcd.setCursor(0,0);  // pulisco la riga
  lcd.print("                    ");
  lcd.setCursor(0,1);  // pulisco la riga
  lcd.print("                    ");
  lcd.setCursor(0,2);  // pulisco la riga
  lcd.print("                    ");
  lcd.setCursor(0,3);  // pulisco la riga
  lcd.print("                    ");

}
//
################################################################################################################
//
################################################################################################################
//
################################################################################################################

void loop() {


  // azzero le variabili con le richieste di controllo del puntamento

  richTasti=false;
  richPot=false;
  richPc=false;
  Uscita=false;

  Controllo_Old=Controllo;
  Controllo=0;


  ScriviOrientamento();  // richiamo la funzione che visualizza l'orientamento corrente
                         // dell'antenna disegnando la riga della bussola e il relativo ago

  //--------------------------------------------------------------------
  // verifica se i tasti direzione sono stati premuti

  if ((digitalRead(tastoOrario)) | ( digitalRead(tastoAntiorario)) == true)
  {
```

```
    richTasti=true;
  }
  else
  {
    // spengo tutte le uscite
    digitalWrite(8,LOW);
    digitalWrite(10,LOW);

    richTasti=false;
  }

  //-------------------------------------------------------------------------
  // verifico se la seriale contiene dei dati di puntamento

  if (Serial.available()>0 ){
    // ci sono dati nella seriale
    richPc=true;
  }
  else
  {
    richPc=false;
  }

  //-------------------------------------------------------------------------
  // valuto se il potenziometro di puntamento manuale è stato spostato
  // inserendo anche una "soglia" minima di differenza di lettura dell'ADC

  if (abs(int(potValueVecchio-analogRead(potPin)))>5)  {
    // si è avuta una variazione del puntamento del potenziometro
    richPot=true;
  }
  else
  {
    richPot=false;
  }

  //-------------------------------------------------------------------------------------------------------
  // stabilisco chi controlla il rotore
  //-------------------------------------------------------------------------------------------------------

  if (richTasti==true){

    lcd.setCursor(0,1);  // pulisco la riga
    lcd.print("                    ");

    Ctrl_Tasti();

  }
  else
  {
    if (richPot==true)
    {

      Ctrl_Pot();

      richPot==false;
    }
    else
    {
      if (richPc==true){

        Ctrl_Pc();
      }
    }
  }
}
//
#########################################################################################################
//
#########################################################################################################
//
#########################################################################################################
```

```
void Ctrl_Tasti() {

  //#############################################################################################
  // gestione tasti direzione.
  // devo evitare che premendo contemporaneamente i tasti direzione vengano comandati i due rami
  // del ponte ad H del motore generando un corto circuito
  // !-----------------------------------------------------!
  // ! tasto SX ! tasto DX ! ramo antiorario ! ramo orario !
  // !-----------------------------------------------------!
  // !     0    !     0    !         0       !       0     !
  // !     0    !     1    !         0       !       1     !
  // !     1    !     0    !         1       !       0     !
  // !     1    !     1    !         0       !       0     !
  // !-----------------------------------------------------!

  // avendo comandato il puntamento con i tasti devo
  // copiare la posizione corrente del potenziomentro
  // nella variabile potVaule Vecchio altrimenti
  // al successivo ciclo se il potenziometro era stato spostato torna a lui il controllo

  potValueVecchio=analogRead(potPin);

  // carico le variabili con lo stato dei tasti direzione

  giraOr=digitalRead(tastoOrario);
  giraAnt=digitalRead(tastoAntiorario);

  // prima spengo tutte le uscite

  digitalWrite(8,LOW);
  digitalWrite(10,LOW);

  if(giraOr==HIGH)
  { // faccio girare il rotore in senso orario
    // adesso imposto le uscite per girera in senso orario
    digitalWrite(10,HIGH);
    digitalWrite(8,LOW);
  }

  if(giraAnt==HIGH)
  { // faccio girare il rotore in senso antiorario
    digitalWrite(10,LOW);
    digitalWrite(8,HIGH);
  }

}

void Ctrl_Pot() {
  //#############################################################################################
  // controllo del puntamento con il poteziomentro
  // in base alla sua posizione faccio girare il motore per ottenere un valore di ADC
  // dipendente dai gradi impostati.
  // ad ogni ciclo verifico se sono stati premuti i tasti direzione o è pervenuto un dato
  // nella seriale

  while ((digitalRead(tastoOrario)==false)&&(digitalRead(tastoAntiorario)==false)&&(Uscita==false)){

    ScriviOrientamento();

    if (Serial.available()>0){
      // se durante il
      richPot=false;
      //carico il valore corrente del potenziometro nella variabile potValueVecchio altrimenti al
successivo loop principale
      // sembra che sia di nuovo spostato il pot
      potValueVecchio=analogRead(potPin);
      return;

    }

    // se non ci sono dati nella seriale
```

```
  //Calcolo la direzione con il potenziometro in gradi e la salvo nella variabile DirzObbiettivo

  if (dirPot>=512)
  {
    DirezObbiettivo=int(dirPot*0.351)-180;
  }
  else
  {
    DirezObbiettivo= int(dirPot*0.351)+180;
  }

  // scrivo la direzione voluta sul display

  lcd.setCursor(0,1);  // pulisco la riga
  lcd.print("                    ");
  lcd.setCursor(0,1);
  lcd.print("Ctrl pot. -> ");
  lcd.print(DirezObbiettivo);
  lcd.write(6);  // scrivo il simbolo del grado

  // valuto la posizione del potenziomentro e calcolo il valore dell'ADC da raggiungere

  if (dirPot>=512) //tra 0° e 180°
  {
    AdcObbiettivo=(DirezObbiettivo*(2.275))+512;
  }
  else // >180°
  {
    AdcObbiettivo=((DirezObbiettivo-180)*(2.275))+102;
  }

  // leggo la posizione del rotore e calcolo la direzione in cui
  // devo far ruotare il motore

  rotValue=analogRead(rotPin);
  differenza=(AdcObbiettivo-rotValue);

  if (differenza >= isteresi)
  {
    // adesso imposto le uscite per girera in senso orario
    digitalWrite(10,HIGH);
    digitalWrite(8,LOW);
  }
  else if (differenza <= -isteresi)
  {
    // faccio girare il rotore in senso antiorario

    digitalWrite(10,LOW);
    digitalWrite(8,HIGH);
  }
  else
  {
    // FERMO IL ROTORE

    digitalWrite(10,LOW);
    digitalWrite(8,LOW);

    // azzero le variabili

    Uscita=true;
    richPot=false;


// devo copiare la posizione corrente del potenziomentro
// nella variabile potVauleVecchio altrimenti
// al successivo ciclo se il potenziometro era stato spostato torna a lui il controllo

  potValueVecchio=analogRead(potPin);

  lcd.setCursor(0,1);  // pulisco la riga
  lcd.print("                    ");
```

```
    }

  }
}


void Ctrl_Pc() {
  //#############################################################################################
  // avendo comandato il puntamento con i tasti devo
  // copiare la posizione corrente del potenziomentro
  // nella variabile potVauleVecchio altrimenti
  // al successivo ciclo se il potenziometro era stato spostato torna a lui il controllo

  potValueVecchio=analogRead(potPin);

  // read the incoming byte:
  incomingByte = Serial.read();

  // Store it in a character array
  stringa[count] = incomingByte;

  count++;

  if ( incomingByte == 13)  // è arrivato un ritorno carrello che indica fine della stringa
  {
    count = 0;

    if (((stringa[0])==2)&&((stringa[1])==65)&&((stringa[2])==71)&&((stringa[6])==13)){
      pippo=3;
      for (pippo =3; pippo<6;pippo++){
        dato=((stringa[pippo]));
        CvDecimale();
        switch(pippo){
        case 3:
          DirezObbiettivo=(100*(dato));
          break;
        case 4:
          DirezObbiettivo=DirezObbiettivo+(10*(dato));
          break;
        case 5:
          DirezObbiettivo=DirezObbiettivo+((dato));
          break;
        }

        Controllo=2;

      }
      digitalWrite(cicalino, HIGH);
      delay(150);
      digitalWrite(cicalino, LOW);

      lcd.setCursor(0,1);  // pulisco la riga
      lcd.print("                    ");
      lcd.setCursor(0,1);
      lcd.print("Ctrl PC ->");
      lcd.print(DirezObbiettivo);
      lcd.write(6);  // scrivo il simbolo del grado

      VaiAdAngolo();
    }

    memset(stringa, 0, sizeof(stringa)); // clear out our previous buffer
  }

}


void VaiAdAngolo(){
  //#############################################################################################

  // calcolo il valore di ADC che corrisponde all'angolo impostato con i potenziometro
```

```
  while ((digitalRead(tastoOrario)==false)&&(digitalRead(tastoAntiorario)==false)&&(Uscita==false)){
    ScriviOrientamento();

    if (Serial.available()>0 ){

      break;
    }

    if (DirezObbiettivo<=180)
    {
      AdcObbiettivo=(DirezObbiettivo*(2.275))+512;
    }
    else
    {
      AdcObbiettivo=((DirezObbiettivo-180)*(2.275))+102;
    }

    rotValue=analogRead(rotPin);
    differenza=(AdcObbiettivo-rotValue);

    if (differenza >= isteresi)
    {
      // adesso imposto le uscite per girera in senso orario
      digitalWrite(10,HIGH);
      digitalWrite(8,LOW);
    }
    else if (differenza <= -isteresi)
    {
      // faccio girare il rotore in senso antiorario

      digitalWrite(10,LOW);
      digitalWrite(8,HIGH);
    }
    else
    {
      // FERMO IL ROTORE

      digitalWrite(10,LOW);
      digitalWrite(8,LOW);


      // azzero le variabili

      Uscita=true;
      richPc=false;


  // devo copiare la posizione corrente del potenziomentro
  // nella variabile potVauleVecchio altrimenti
  // al successivo ciclo se il potenziometro era stato spostato torna a lui il controllo

    potValueVecchio=analogRead(potPin);

    lcd.setCursor(0,1);  // pulisco la riga
    lcd.print("                    ");

      break;
    }
  }
}
void ScriviOrientamento() {
  //################################################################################
  //--------------------------------------------------------------------------------
  // Inizio blocco lettura dell'orientamento corrente dell'antenna

  lcd.setCursor(0,0);

  dirPot=(analogRead(potPin));

  // converto la lettura grezza in un lettura diretta in gradi del puntamento dell'antenna

  if(analogRead(rotPin)>=512)
```

```
{
  dirAnt=(int(analogRead(rotPin)*0.4398)-225);
}
else
{
  dirAnt=(int(analogRead(rotPin)*0.4398)+135);
}

lcd.setCursor(0,0);  // pulisco la riga
lcd.print("                    ");
lcd.setCursor(0,0);

lcd.print("Dir. antenna = ");
lcd.print(dirAnt);


//  Serial.print(dirAnt);
// Serial.println("");


lcd.write(6);  // scrivo il simbolo del grado

// se sono nell aparte di sormonto del rotore metto un asterisco o cancelletto

if(analogRead(rotPin)<102){
  lcd.setCursor(19,0);  // pulisco la riga
  lcd.print("*");
}
if(analogRead(rotPin)>921){
  lcd.setCursor(19,0);  // pulisco la riga
  lcd.print("#");
}

// Fine blocco lettura puntamento corrente dell'antenna
//--------------------------------------------------------------------
// Disegno nella terza riga la scala di puntamento

lcd.setCursor(0,2);
//  lcd.write(7);
lcd.print("_");
lcd.print("S");
lcd.write(7);
lcd.write(7);
lcd.write(7);
lcd.print("O");
lcd.write(7);
lcd.write(7);
lcd.write(7);
lcd.print("N");
lcd.write(7);
lcd.write(7);
lcd.write(7);
lcd.print("E");
lcd.write(7);
lcd.write(7);
lcd.write(7);
lcd.print("S");
//  lcd.write(7);
lcd.print("_");
lcd.setCursor(0,3);

// devo separare l'indicazione da Nord in senso orario (passando da EST verso SUD)

if ((dirAnt >= 0) && (dirAnt <=180))   {

  nTacche=int(dirAnt/4.44)+2;

  nCaratteri=int(nTacche/5);

  lcd.setCursor(0,3);  // pulisco la riga
  lcd.print("                    ");
```

```
    lcd.setCursor((nCaratteri+9),3);

    nPosTacca=nTacche-nCaratteri*5;

    switch(nPosTacca){

    case 0:
      lcd.write(1);
      break;
    case 1:
      lcd.write(2);
      break;
    case 2:
      lcd.write(3);
      break;
    case 3:
      lcd.write(4);
      break;
    case 4:
      lcd.write(5);
      break;
    }
  }
  else //-------------------------------------------------
  {
    nTacche=int((dirAnt-180)/4.44)-3;

    nTacche=nTacche+10;  //   aggiungo i due caratteri di offset da sx del sud


    nCaratteri=int(nTacche/5); // il sud (di sx) antiorario parte da un offset di due caratteri


    lcd.setCursor(0,3);  // pulisco la riga
    lcd.print("                    ");

    lcd.setCursor((nCaratteri),3);

    nPosTacca=nTacche-nCaratteri*5;

    switch(nPosTacca){

    case 0:
      lcd.write(1);
      break;
    case 1:
      lcd.write(2);
      break;
    case 2:
      lcd.write(3);
      break;
    case 3:
      lcd.write(4);
      break;
    case 4:
      lcd.write(5);
      break;

    }
  }

}

int CvDecimale(){
  //################################################################################
  switch (dato){
  case 48:
    dato=0;
    break;
  case 49:
    dato=1;
    break;
```

```
  case 50:
    dato=2;
    break;
  case 51:
    dato=3;
    break;
  case 52:
    dato=4;
    break;
  case 53:
    dato=5;
    break;
  case 54:
    dato=6;
    break;
  case 55:
    dato=7;
    break;
  case 56:
    dato=8;
    break;
  case 57:
    dato=9;
    break;
  }
  return dato;
}
```