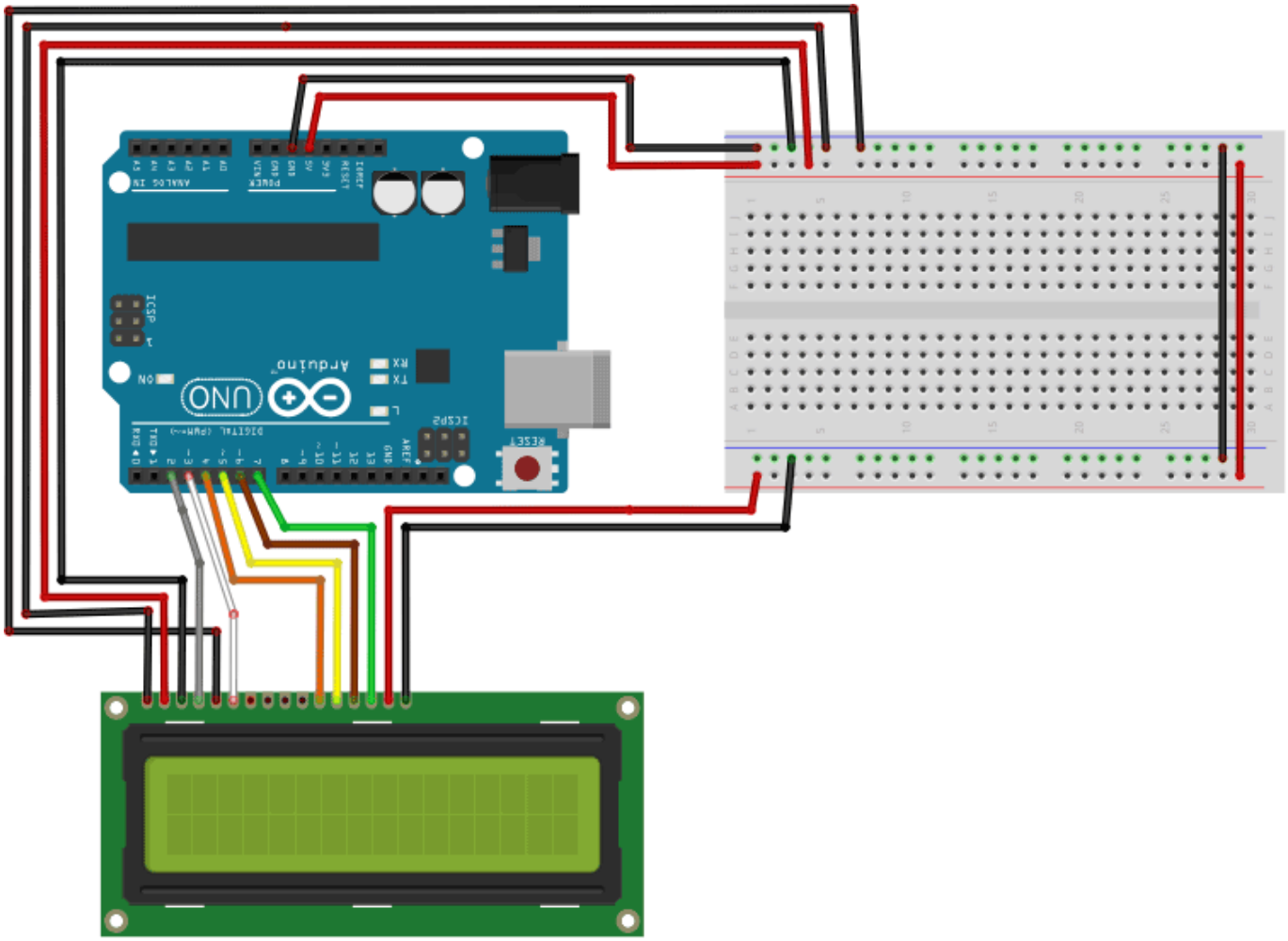
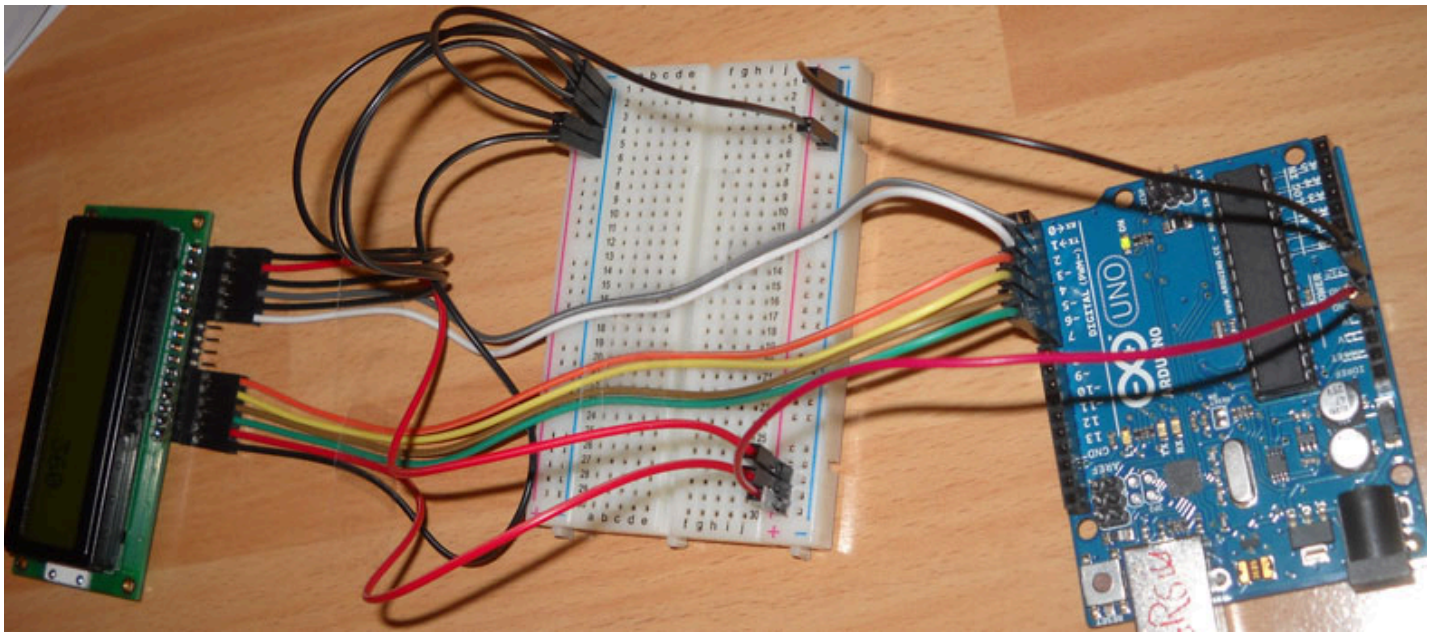
INPUT ANALOGICO: DISPLAY ANALOGICO

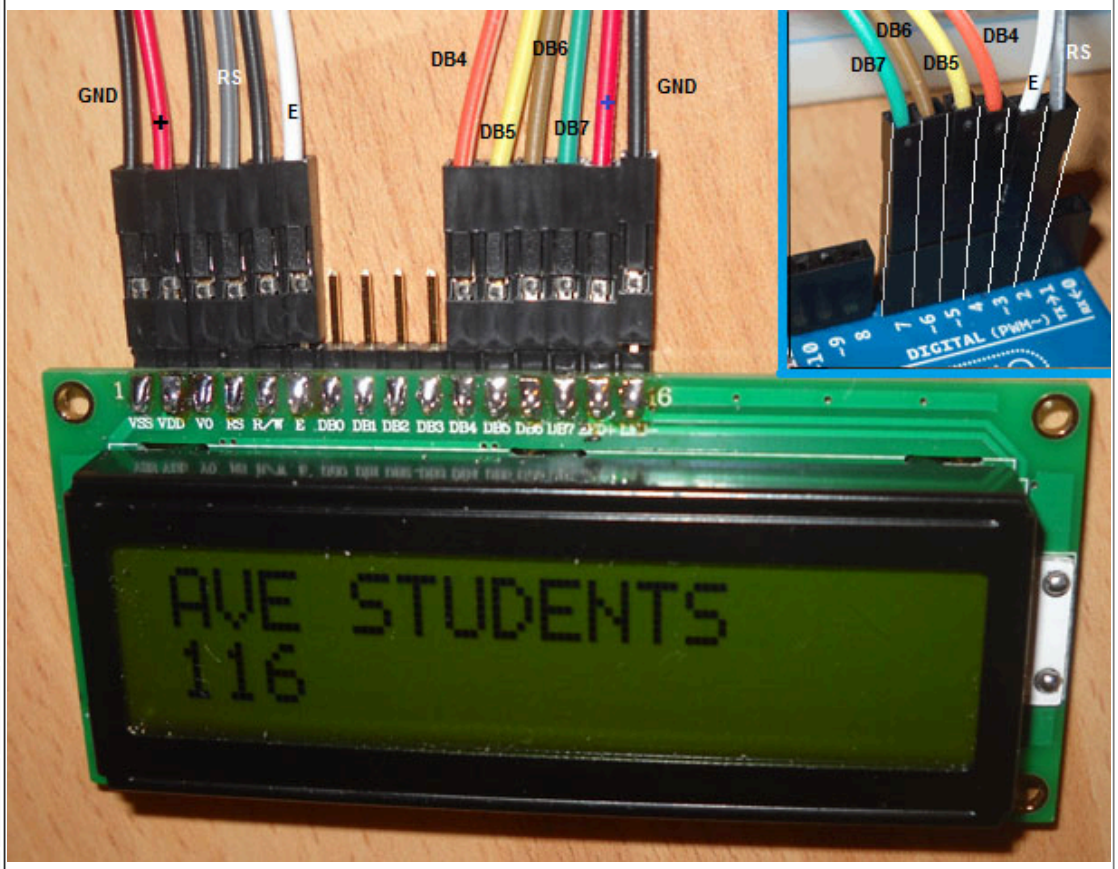
Costruire il seguente circuito (serve un Display LCD)





Esistono 2 categorie di display: quelli a 14 contatti e quelli a 16 contatti. Questi ultimi sono retroilluminati e la luminosità viene regolata agendo sul pin 15 e 16. Ecco la tabella dei PIN e le connessioni implementate nel nostro circuito:

PIN	Significato
1	VSS - GND – Massa
2	Vdd - Vcc: +5v power (4.5 – 5.5 V)
3	V0 - Contrasto
4	RS: Register Select indica il tipo di comando che inviamo
5	R/W indica all'lcd se deve scrivere o leggere – 0 scrivi 1 leggi
6	E: Enabled per abilitare la scrittura nei registri
7	DB0: linea dati 0, non usato
8	DB1: linea dati 1, non usato
9	DB2: linea dati 2, non usato
10	DB3: linea dati 3, non usato
11	DB4: linea dati 4 collegato al pin 4 di Arduino
12	DB5: linea dati 5 collegato al pin 5 di Arduino
13	DB6: linea dati 6 collegato al pin 6 di Arduino
14	DB7: linea dati 7 collegato al pin 7 di Arduino
15	Led+: terminale positivo per la retroilluminazione (Anodo)
16	Led-: terminale negativo per la retroilluminazione (Catodo)



I Pin 7,8,9 e 10, nel nostro schema, risultano come non collegati. In realtà i display di questo tipo possono essere pilotati sia con 4 che 8 bit (con 4 possiamo risparmiare i pin di Arduino!). La differenza tra una modalità e l'altra consiste nella differente velocità di trasferimento dell'informazione sul display. Non avendo particolari necessità i 4 bit sono sufficienti. Dal Pin 7 al Pin 14 abbiamo quindi le linee dati su cui transitano i dati che si inviano o si ricevono dai registri del display. Un valore HIGH (H) indica scrittura (WRITE) del bit nel registro del display, un valore LOW (L) indica un valore letto (READ) da un registro.

Al pin V0 In genere viene collegato ad un potenziometro o trimmer in configurazione partitore di tensione in modo che possiate applicare sul Pin 3 una tensione che varia da 0 a +5V. Al variare della tensione varia il contrasto.

Gli esempi proposti utilizzano la libreria di sistema "**LiquidCrystal**". Tale libreria è descritta in modo egregio nella pagina ufficiale presente sul sito (http://arduino.cc/en/Reference/LiquidCrystal). I metodi disponibili della classe **LiquidCrystal** sono: **begin(), clear(), home(), setCursor(), write(), print(), cursor(), noCursor(), blink(), noBlink(), display(), noDisplay(), scrollDisplayLeft(), scrollDisplayRight(), autoscroll(), noAutoscroll(), leftToRight(), rightToLeft(), createChar()**

Codice Sorgente A

Obiettivo progetto: Scrivere sul display LCD la scritta "AVE STUDENTS" seguita dal numero di secondi trascorsi dall'accensione di Arduino.

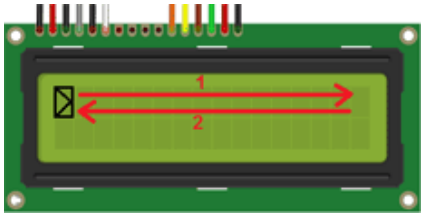
soluzione:

In questo esempio i metodi dell'oggetto **LiquidCrystal()** utilizzati sono: **begin()**, **clear()**, **print(), setCursor()**

```
/* ---------------------------------------------------------------------------
 Esempio che mostra l'utilizzo del display LCD senza uso del potenziometro.
 La libreria LiquidCrystal lavora con tutti i display LCD compatibili con il
 driver  Hitachi HD44780 driver (interfaccia a 16/14 pin)

 Nel mio circuito i pin dell'LCD sono connessi a quelli di Arduino in questo modo:
 * Pin R/W dell LCD => pin GND
 * Pin V0 dell LCD => pin GND
 * Pin RS dell LCD => pin digitale 2
 * Pin E  dell LCD => pin digitale 3
 * Pin D4 dell LCD => pin digitale 4
 * Pin D5 dell LCD => pin digitale 5
 * Pin D6 dell LCD => pin digitale 6
 * Pin D7 dell LCD => pin digitale 7
--------------------------------------------------------------------------------- */
#include <LiquidCrystal.h> // Libreria per gestire il Display LCD

/* ------------------------------------------------------------
Inizializzo la libreria con il comando LiquidCrystal impostando i
contatti secondo il circuito che ho costruito:

   LiquidCrystal(rs, enable, d4, d5, d6, d7)
   LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
   LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
   LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
------------------------------------------------------------- */
LiquidCrystal lcd(2, 3, 4, 5, 6, 7); //Ho usato la 1^ sintassi

void setup()
{
  // Imposto il nr di colonne e righe del Display
  lcd.begin(16, 2);
  lcd.clear(); // Pulisce l'LCD e pone il cursore all'inizio
  lcd.print("AVE ");     // Stampo a video il AVE ...
  lcd.print("STUDENTS"); // e prosegue sulla stessa riga
}

void loop()
{
   // ISTRUZIONE: setCursor(col, row) - La numerazione parte da 0
   lcd.setCursor(0, 1); // 1° colonna - 2° riga
   // stampa il numero di secondi trascorsi dal reset di Arduino
   lcd.print(millis()/1000);
}
```

Codice Sorgente B

Obiettivo progetto: Creare un simbolo a forma di busta chiusa (disposto in verticale) che oscilla da destra a sinistra sul display LCD



soluzione:

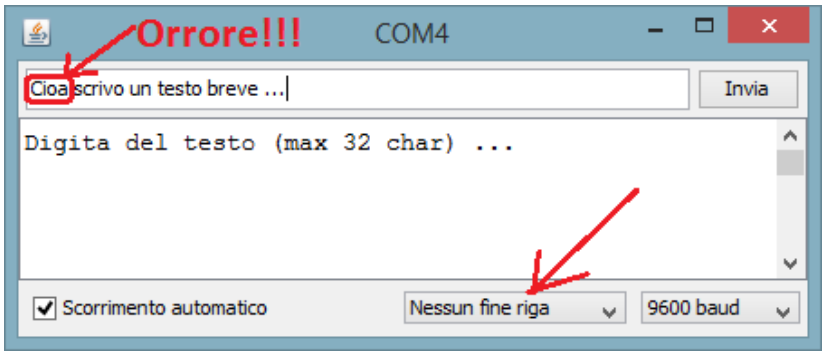
In questo esempio i metodi dell'oggetto **LiquidCrystal()** utilizzati sono: **begin()**, **clear()**, **scrollDisplayRight(), scrollDisplayLeft(), setCursor(), createChar(), write()**

```
/* ---------------------------------------------------------------------------
 Esempio che mostra l'utilizzo del display LCD senza uso del potenziometro.
 La libreria LiquidCrystal lavora con tutti i display LCD compatibili con il
 driver  Hitachi HD44780 driver (interfaccia a 16/14 pin)
```

```
 Nel mio circuito i pin dell'LCD sono connessi a quelli di Arduino in questo modo:
 * Pin R/W dell LCD => pin GND
 * Pin V0 dell LCD => pin GND
 * Pin RS dell LCD => pin digitale 2
 * Pin E  dell LCD => pin digitale 3
 * Pin D4 dell LCD => pin digitale 4
 * Pin D5 dell LCD => pin digitale 5
 * Pin D6 dell LCD => pin digitale 6
 * Pin D7 dell LCD => pin digitale 7
---------------------------------------------------------------------------------- */
#include <LiquidCrystal.h> // Libreria per gestire il Display LCD
#define nCOL 15
byte Segno[8] =
{
  B11111,
  B11001,
  B10101,
  B10011,
  B10101,
  B11001,
  B11111,
  B00000
}; // Array 8x5 bit per la definizione di un singolo carattere
/* ------------------------------------------------------------
Inizializzo la libreria con il comando LiquidCrystal impostando i
contatti secondo il circuito che ho costruito:

   LiquidCrystal(rs, enable, d4, d5, d6, d7)
   LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
   LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
   LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
------------------------------------------------------------ */
LiquidCrystal lcd(2, 3, 4, 5, 6, 7); //Ho usato la 1^ sintassi

void setup()
{
  // Creo un carattere personalizzato. Sono gestibili al massimo 8 caratteri personalizzati 8x5 pixels
  // (numerati da 0 a 7). L'aspetto del carattere è specificato mediante un array di 8 byte (uno per ogni
  // riga. I primi 5 bit determinano i pixel accesi. Deve stare all'inizio del setup()
  lcd.createChar(0, Segno); // Creo il carattere che mi sono inventato
  // Imposto il nr di colonne e righe del Display
  lcd.begin(16, 2); //
  lcd.clear();
  // Per mostrare il carattere occorre poi usare la funzione
  // write con in numero abbinato nel createChar(n,Array) - write(byte(n))
  lcd.write(byte(0));
}

void loop()
{
  int i;
  // Questo ciclo sposta il testo a destra
  for (i = 0; i < nCOL; i++)
  {
    // Sposta il contenuto del display di una colonna a destra
    lcd.scrollDisplayRight();
    delay(150); // aspetta un attimo
  }
  // Questo ciclo sposta il testo a sinistra
  for (i = 0; i < nCOL; i++)
  {
    // Sposta il contenuto del display di una colonna a sinistra
    lcd.scrollDisplayLeft();
    delay(150); // aspetta un attimo
  }
}
```

Codice Sorgente C

Obiettivo progetto: Creare uno programma che sia in grado di leggere il testo digitato sul Serial Monitor e lo mostri sul display LCD. Poichè la linea di invio del serial monitor accetta al massimo 69 caratteri mentre il display utilizzato può mostrare solamente 32 caratteri inserire i necessari controlli che evitino di mostrare la sequenza digitata oltre il trentaduesimo carattere.



soluzione:

In questo esempio i metodi dell'oggetto **LiquidCrystal()** utilizzati sono: **begin()**, **clear()**, **cursor()**, **leftToRight()**, **blink()**, **setCursor()**, **createChar()**, **write()**, **home()**, **print()**

```
// La libreria WIRE.H va aggiunta se ho effettuato l'aggiornamento
// della libreria LiquidCrystal alla versione che supporta I2C
#include <Wire.h>  // Libreria di sistema - E' richiesta da I2CIO.cpp
#include <LiquidCrystal.h> // Libreria LCD I2C
/* --------------------------------------------------------------------------------
 Esempio che mostra sul display LCD quello che viene digitato sul serial monitor
 La libreria LiquidCrystal lavora con tutti i display LCD compatibili con il
 driver  Hitachi HD44780 driver (interfaccia a 16/14 pin)

 Nel mio circuito i pin dell'LCD sono connessi a quelli di Arduino in questo modo:
 * Pin R/W dell LCD => pin GND
 * Pin V0 dell LCD => pin GND
 * Pin RS dell LCD => pin digitale 2
 * Pin E  dell LCD => pin digitale 3
 * Pin D4 dell LCD => pin digitale 4
 * Pin D5 dell LCD => pin digitale 5
 * Pin D6 dell LCD => pin digitale 6
 * Pin D7 dell LCD => pin digitale 7
-------------------------------------------------------------------------------- */

/* ------------------------------------------------------------
Inizializzo la libreria con il comando LiquidCrystal impostando i
contatti secondo il circuito che ho costruito:

   LiquidCrystal(rs, enable, d4, d5, d6, d7)
   LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
   LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
   LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
------------------------------------------------------------ */
LiquidCrystal lcd(2, 3, 4, 5, 6, 7); //Ho usato la 1^ sintassi

void setup()
{
   Serial.begin(9600);
   lcd.begin(16, 2);   // inizializzo LCD
   lcd.clear(); // pulisce lo schermo
   lcd.setCursor(0,0); // Si posiziona sulla prima riga, prima colonna
   lcd.print("Digita sul S.M.");
   lcd.setCursor(0,1);
```

```
   lcd.print("del testo ...");
   Serial.println("Digita del testo (max 32 char) ...");
}

void loop()
{
  char ch;
  int r=0, c=0;
    if (Serial.available()) // Se ci sono caratteri sul Serial Monitor
    {
      // aspetta un attimo affinchè arrivi l'intero messaggio
      delay(100);
      lcd.clear(); // pulisce lo schermo
      while (Serial.available() > 0)
      {
          if (r!=2)
          {
             lcd.setCursor(c++,r);
             lcd.write(Serial.read());
             if (c==16) {c=0; r++;}
          }
          else
            Serial.read(); // butto via i caratteri in +
      }
    }
}
```

Si osservi, nonostante la tipologia del circuito utilizzata sia completamente diversa, l'estrema somiglianza di questo programma con quello proposto nell'esempio (*analogo come funzionalità richieste*) al **codice sorgente A** proposto per LCD con bus I2C.