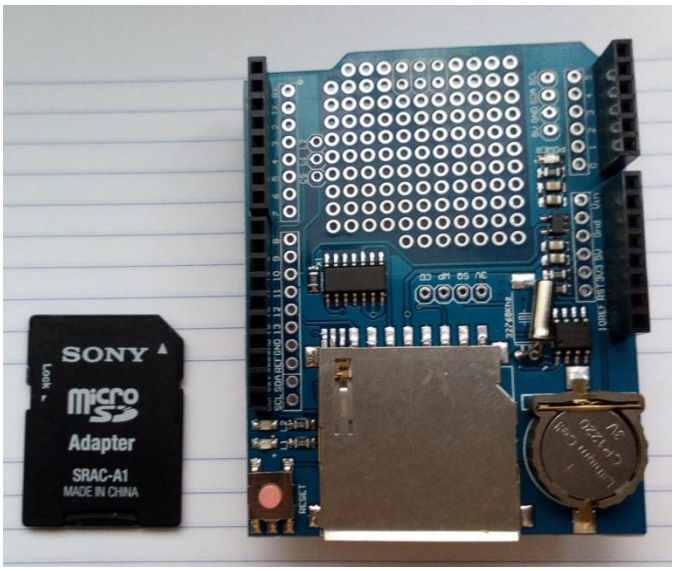
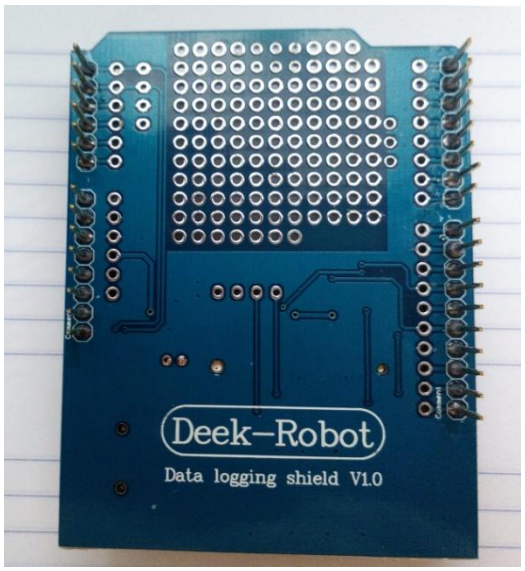
43- Secure Digital (some notes at end of this section)





Le schede secure digital sono un efficiente ed economico supporto per la memorizzazione di informazioni. In arduino trovano normalmente impiego nella registrazione di dati relativi ai processi che si stanno gestendo (in pratica in un sistema di “data logging”). Tipica la registrazione dei dati ambientali (ad esempio temperatura, umidita’, pressione atmosferica, velocita’ del vento e millimetri di pioggia) ad intervalli di tempo predefiniti. I dati, una volta registrati su SD possono poi essere importati in un programma di gestione residente su pc (ad esempio un programma excell) ed elaborati a piacimento.

Esistono svariati tipi di stazioni di lettura e scrittura ed in questo esercizio ne e’ stata usata una inserita in una shield (vedi figura) dotata anche di un RTC (real time clock) in grado di sincronizzare l’orologio interno di Arduino con data ed ora effettiva (Arduino, quando non e’ alimentato, perde la cognizione del tempo).

L’utilizzo della stazione di lettura/scrittura, proposto in questo esercizio e’ abbastanza inusuale. Sulla scheda sd viene memorizzato un testo formato da un numero indefinito di righe di 20 caratteri ognuna, che viene poi letto ed esposto sul monitor seriale e, se disponibile, su di un display 2004 (20 caratteri su quattro righe).

Il testo puo’ essere indifferentemente caricato sulla secure digital da PC, tramite un programma in grado di scrivere file di tipo .txt, oppure direttamente da Arduino, tramite la tastiera seriale (la tastiera del pc, al quale e’ collegato tramite il cavo usb).

A questo scopo il programma e’ stato arricchito da un embrionale sistema operativo, in grado di interpretare ed eseguire i seguenti quattro comandi:

- ^lgg Legge il file “testo.txt” presente su SD ed espone il contenuto sul display 2004 e sul monitor seriale;
- ^scr Scrive su SD quanto digitato da tastiera (in realta’ registra solo la parte finale del testo poiche’ la parte iniziale, divisa in blocchi da 20 caratteri, viene registrata mano a mano che viene ricevuta da Arduino);
- ^cnc Cancella il file “testo.txt” presente su scheda sd
- ^clr Pulisce il buffer in cui e’ memorizzato il testo digitato ma non ancora riportato su scheda SD

Sotto l'aspetto operativo il programma, appena attivato, cerca su scheda SD un file di nome "testo.txt" e se lo trova lo legge e lo espone, ripetitivamente, su monitor seriale e sul display 2004 (l'esposizione e' volutamente rallentata per consentire la lettura del testo).

In questo modo il programma puo' funzionare anche in assenza di tastiera e monitor seriale, limitandosi ad esporre ciclicamente quanto presente sulla scheda.

Alla fine di ogni ciclo espositivo, arduino verifica se e' arrivato qualche segnale dalla tastiera seriale e, in caso positivo, interrompe la visualizzazione, elabora il segnale pervenuto (memorizza il testo oppure esegue il comando) e si pone in attesa di ulteriori azioni da parte dell'operatore.

Se vengono digitati testi piu' lunghi di 20 caratteri, Arduino li spezza in blocchi da 20 carattere che registra immediatamente su scheda SD. Per registrare la parte finale del testo (la coda eventualmente minore di 20 caratteri), e' necessario ricorrere al comando "^scr".

Il film di questo progetto e' reperibile qui (click).

Prima di procedere alla compilazione del programma e' necessario installare, se non gia' fatto, l'ultima versione della libreria:

- LiquidCrystal_I2C.h - reperibile qui'.

Per le modalita' di installazione della libreria si rimanda all'iter gia' illustrato nei precedenti esercizi e comunque riassumibile in:

- download della libreria in forma compressa (normalmente ogni sito che tratta una libreria ha un punto dal quale si puo' attivare il download);
- installazione via IDE->sketch->include library->add .zip library
- verifica di avvenuta installazione (la libreria deve essere nell'elenco presente in IDE->sketch->include library->contributed library)

Nota: Questo esercizio e questa nota sono parte di una serie che vede protagonisti Arduino ed alcuni dei componenti ad esso collegabili. Per la maggior parte degli esercizi e' anche disponibile un filmato su youtube.

- Esercizi facenti parte della raccolta
- Filmati presenti su youtube
- Informazioni su arduino e sui componenti collegabili (PDF scaricato nell'area di download)
- Breve manuale di programmazione (PDF scaricato nell'area di download)

Per eventuali chiarimenti o suggerimenti sul contenuto di questa scheda scrivere a giocarduino@libero.it

Here some note about this project, translated by google translator 

The secure digital cards are a cheap and efficient support for information storage. In Arduino they are normally used to register some job-related data (basically a "data logging system"). For instance to record environmental data (temperature, humidity, atmospheric pressure, wind speed and rain millimeters) at predefined time intervals. Data recorded on SD can then be imported into a management program resident on a PC (such as an excell program) and processed at will.

There are several devices for read and write SD cards and the one used in this project is inserted in a board (see figure) also equipped with an RTC (real time clock) able to synchronize the Arduino internal clock with actual date and time (Arduino, when not feeded, lose track of time).

The use of the read/write station, proposed in this project, is quite unusual. Arduino writes on sd a text composed of rows of 20 characters (no limits on numbers of rows), which is then read and displayed on serial monitor and, if available, on a 2004 display (20 characters on four rows).

For this purpose the program was enriched from an embryonic operating system, able to interpret and execute the following four commands:

- **^ lgg** Read the "text.txt" file on SD, and displays the content on lcd display and on serial monitor;
- **^ scr** Writes on SD card, words typed from keyboard. Actually only records the final part of the text because the initial part is recorded, in rows of 20 characters, when received on Arduino);
- **^ cnc** Delete the "text.txt" file on sd card
- **^ clr** Clears buffer where was stored text, typed but not yet writed on SD card

Under operational aspect, the program, just activated, tries to find on SD a file named "text.txt" and, if found, reads it and exposes it, repetitively, on the serial monitor and on lcd display. The exposure is deliberately slowed to allow reading.

At the end of each reading cycle, Arduino checks if is arrived some signal from keyboard and, if so, stops displaying, processes the received signal (stores the text or executes the command), and waits for further action by operator.

If you type more of 20 characters, Arduino breaks them into blocks of 20 character and immediately records each block on SD. To record the final part of the text (the tail, if less than 20 characters), you need to use the command "^ scr".

The movie of this project is available here (click).

Before proceeding to program compilation, must be installed, if not already done, the library:

- LiquidCrystal_I2C.h found here

For library installation, see process shown in previous examples, and summarized in:

- library download in compressed form;
- Installation via IDE-> sketch-> includes Library-> add .zip library
- After installation need verification: the library must be present in IDE-> sketch-> includes Library-> Contributed library
- delete existing library of LCD display management (the new library will incorporate the functions) going to c: -> Programs (x86) -> arduino-> libraries, selecting the old library (should be called liquidcrystal) and pressing the "delete key ".

Note: This project and this note is part of a series that sees, as main characters, Arduino and some of connectable components. For most projects there is also a video on youtube.

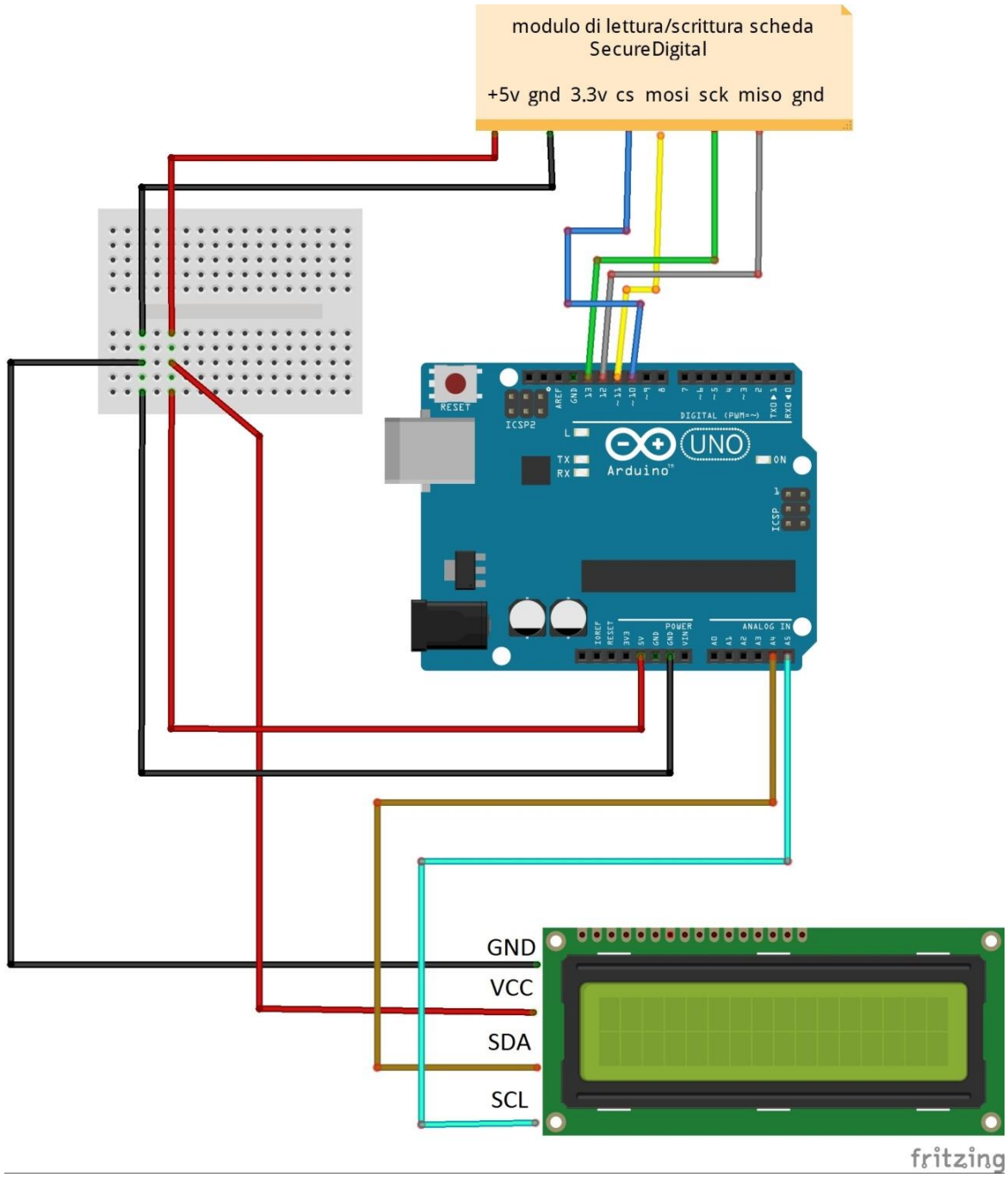
- Projects collection
- Movies on youtube
- About Arduino and components (italian; pdf will be downloaded in your download area
- Quick programming guide (almost english; pdf will be downloaded in your download area)

For any questions or suggestions about this note (and on its english translation), please write to giocarduino@libero.it (simple words and short sentences, please)

Materiali

- Una stazione di lettura/scrittura di schede secure digital
- Una scheda secure digital, formattata FAT32
- Un display 2004 (20 caratteri su quattro righe) – opzionale

Schema



Programma

```
/* Attenzione: facendo il copia/incolla dal PDF all'IDE si perde la formattazione del testo.
 * Per rendere piu' facilmente leggibile il programma e' opportuno formattarlo subito dopo il
 * trasferimento nell'IDE, premendo CTRL+T.
 *
 *  questo programma utilizza un lettore/registratore di secure digital per
 *  visualizzare un testo presente su di una scheda SD e/o per
 *  memorizzare un testo battuto sulla tastiera del monitor seriale.
 *
 *  collegamento del lettore/registratore SD
 *
 *  ..........arduino uno.....Arduino mega
```

Arduino: Secure Digital

```
*  GND ......non usato........non usato
*  MISO .........12 ................50
*  SCK  .........13 ................52
*  MOSI .........11 ................51
*  CS.............. 10 ................53
*  3.3v .....non usato ........non usato
*  GND ..........GND ..............GND
*  +5V ..........5v ................5v
*
*  collegamento del display lcd 2004 con driver I2C
*
*  GND..........GND ..............GND
*  VCC .........5v ................5v
*  SDA .........A4 ................20
*  SCL .........A5 ................21
*
*  REGOLE DI UTILIZZO DEL PROGRAMMA
*
*  Arduino legge il testo presente su secure digital e lo espone
*  sul display LCD. Se intercetta un segnale proveniente dalla
*  tastiera del computer interrompe l'esposizione del testo
*  e gestisce cio' che arriva da tastiera (testo oppure comandi).
*  Se non arrivano comandi si limita a memorizzare quanto digitato
*  ed a registralo su secure digital, a blocchi di venti caratteri.
*
*  nota #1: la battitura del carattere "^" segnala l'arrivo di un comando
*  per cui i tre caratteri che  seguono sono interpretati come un comando
*  e non vengono memorizzati su secure digital
*
*  nota #2: su secure digital il nuovo testo viene accodato al
*  preesistente, per cui se si vuole digitare un nuovo testo bisogna
*  prima cancellare il vecchio (tramite il comando ^cnc).
*
*  nota #3:  alla fine della digitazione di un testo e' necessario
*  inviare il comando di scrittura (^scr) per essere certi che tutto
*  quanto digitato sia stato effettivamente memorizzato su secure
*  digital
*
*  Questo l'elenco dei comandi
*
*  ^scr -  scrive su sd cio che e' memorizzato nel buffer di ricezione
*  ^lgg -  legge il file su sd e lo visualizza sul display lcd
*  ^cnc -  cancella il file su sd
*  ^clr -  cancella quanto memorizzato ma non ancora scritto su sd
*
*
*-----------------------------------------------------------------------------------------------
*
* Warning: cut&paste from PDF to IDE loses formatting. to restore it press CTRL + T.
*
* This program uses a "secure digital" device to display a text stored on an SD card and/or to
* store a text incoming from serial keyboard.
*
* Connection on SD appliance:
*  ..........arduino uno.....Arduino mega
*  GND .......not used.........not used
*  MISO .........12 ................50
*  SCK  .........13 ................52
*  MOSI .........11 ................51
*  CS.............. 10 ................53
*  3.3v .....not used .........not used
*  GND ..........GND ..............GND
*  +5V ..........5v ................5v
*
* Connection on display 2004 with I2C driver
*
*  GND..........GND ..............GND
*  VCC .........5v ................5v
*  SDA .........A4 ................20
*  SCL .........A5 ................21
*
* RULES
*
* Arduino reads text on "secure digital" and exposes it on LCD. If intercept a signal from
* keyboard, interrupts the text display and manages what comes from keyboard (text or commands)
* If don't arrive commands, it stores text, on SD card, in blocks of twenty  characters.
*
* note # 1: The "^" character signals the arrival of a command. The three following characters are
```

```
 * interpreted as a command, and are not stored on secure digital
 *
 * note # 2: new inputted text is appended to the existing one on SD card, so if you want to type a
 * new text, first must be deleted the old one (through "^cnc" command).
 *
 * note # 3: At the end of text you must send the write command (^scr) to make sure that everything
 * typed was stored on secure digital
 *
 * COMMANDS:
 *  ^scr -  write on sd
 *  ^lgg -  read from sd and display it on lcd
 *  ^cnc -  delete file on sd
 *  ^clr -  clear memory
 * ---------------------------------------------------------------------------------------------------
 *
 */
#include <SPI.h>                // libreria spi, presente di default nell'IDE
#include <SD.h>                 // libreria SD, presente di default nell'IDE
#include <Wire.h>               // libreria wire presente, di fault, nell'IDE
#include <LiquidCrystal_I2C.h> // libreria di gestione del display lcd
//--------------------addr  en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // definisce la tipologia del display
File filesd;
char tabdescr [6] [16]      // tabella descrizioni da esporre
{
  {"cmd invalido"},         // 0  "invalid command"
  {"mem. cancellata"},      // 1  "erased memory"
  {"apertura ok"},          // 2  "open OK"
  {"apertura ko"},          // 3  "open KO"
  {"scritt.completa"},      // 4  "write complete"
  {"file cancellato"},      // 5  "file erased"
};
int codescr        = 0;    // codice della descrizione da esporre
char tabellacomandi [16]  // in tabella sono presenti i comandi alfabetici. un comando ogni
// tre caratteri. l'indice che individua la posizione del comando e' anche l'indice per la
// codifica (la trasformazione in un numero) del comando stesso tramite la tabellacodicecomando
{"scrlggcncclr   "};  // command table
int tabellacodicecomando [6]     // tabella parallela alla tabellacomandi; si utilizza il medesimo
{1, 2, 3, 4, 0, };               // indice per trasformare il comando alfabetico in un numero
char bufferricezione [21];       // tabella di memorizzazione dei caratteri ricevuti
int indicebuffer  = 0;           // indice di scorrimento del buffer di ricezione
int i             = 0;           // indice utilizzato nei cicli di for
char ch;                         // zona di ricezione dei caratteri da tastiera
char mex [4];                    // zona di memorizzazione del comando ricevuto
int icom          = 0;           // indice di scorrimento della tabella comandi
int comando       = 0;           // zona di memorizzazione del codice del comando ricevuto
int semaforo2     = 0;           // semaforo utilizzato nella verifica del comando ricevuto
char nomefile [10] = {"testo.txt"}; // nome del file su sd
int indicedisplay = 0;           // indice per la gestione della tabella del display lcd;
int rig           = 0;           // numero di riga nella gestione del diplay lcd
int primavolta    = 0;           // interruttore per l'interruzione della visualizzazione ripetitiva
// del testo su sd
//
//  ***routine di esposizione su display lcd e su monitor seriale del testo battuto o letto ***
//  **** display on lcd and on serial monitor, text incoming from keyboard or readed on sd *****
//
void visualizzacarattere (void)
{
  if (indicedisplay == 80)     // se lo schermo e' pieno lo pulisce
  {
    delay (1000);              // attende un secondo per consentire il completamento della lettura
    lcd.clear();
    indicedisplay = 0;
    rig = 0;
    Serial.println (" ");
  }
  if ((indicedisplay == 20) || (indicedisplay == 40) || (indicedisplay == 60)) // se fine riga
  {
    rig++;
    lcd.setCursor (0, rig); // si sposta su di una nuova riga
    Serial.println (" ");
  }
  lcd.print (bufferricezione [indicebuffer]);       // espone su lcd il carattere in esame
  Serial.print (bufferricezione [indicebuffer]);    // espone sul monitor il carattere in esame
  delay (100);               // rallenta l'esposizione per consentirne la lettura
  indicedisplay ++;          //incrementa l'indice di visualizzazione sul display
}
//
```

```
//  ****** routine di esposizione dei messaggi di sistema su display lcd e su monitor seriale
//  *************  send system message on lcd display and on serial monitor *********
//
void descrizione (void)
{
  lcd.clear ();
  lcd.print (tabdescr [codescr]);
  Serial.println (tabdescr [codescr]);
  delay (3000); // attende tre secondi per consentire la lettura del messaggio
  lcd.clear ();
}
//
// ****** routine di ricezione e controllo dei comandi (tre caratteri preceduti da un "^") *********
// *********** receive and check commands (three characters preceeded by an "^") *****
//
void ricevicomando (void)
{
  for (i = 0; i < 3; i = i)          //for di ricezione di tre caratteri da tastiera Si forza i=i
    // per mantenere "vivo" il ciclo sino a quando non sono pervenuti tre caratteri
  {
    if (Serial.available() > 0)      //se e' in arrivo qualcosa dalla tastiera
    {
      ch = Serial.read();     //legge (inserisce in ch) cio' che e' stato battuto sulla tastiera
      mex [i] = ch;           // memorizza il carattere ricevuto
      i++;
    }
  }
  //  a questo punto sono pervenuti tre caratteri, ora se ne verifica validita' e li si
  //  trasforma in codici di comando
  //
  //---------------------  controllo di validita' del comando-----------------------
  //--------------------------check command validity -------------------------------
  //
  lcd.clear();
  Serial.println (" ");
  lcd.print (mex);
  Serial.println (mex);
  semaforo2 = 0;          // azzera preventivamente il semaforo di controllo del codice comando
  comando   = 0;          // azzera preventivamente il codice comando
  for (icom = 0; icom < 5; icom++)     // for di scorrimento della tabella comandi
  {
    if ((tabellacomandi [icom * 3] == mex [0]) && (tabellacomandi [icom * 3 + 1] == mex[1]) &&
(tabellacomandi[icom * 3 + 2] == mex[2]))
    { // se trova corrispondenza in tabella
      comando = tabellacodicecomando [icom]; // preleva dalla tabellacodicecomando il codice
                                             // numerico del comando appena digitato
      semaforo2 = 1;    // mette a 1 il semaforo2 per uscire dalla routine di ricezione e controllo
    }
    if (semaforo2 == 1)
      break;
    // se il comando e' presente in tabella comandi, esce dal for di scorrimento
    // della tabella comandi
  }
  if (comando == 0)
  {
    // se arriva qua' significa che e' stato inserito un comando sbagliato ed il campo comando,
    // inizializzato a zero, e' rimasto a zero
    codescr = 0;         // segnala comando invalido
    descrizione ();
  }
}
//
// **** routine di pulizia del buffer di ricezione/memorizzazione/scrittura/lettura *****
// ***************************** clear buffer routine ***********************************
//
void pulisci (void)
{
  for (indicebuffer = 0; indicebuffer <= 20; indicebuffer ++)
    bufferricezione [indicebuffer] = ' ';    // pulisce la tabella di ricezione
  indicebuffer  = 0;       // inizializza l'indice della tabella di ricezione
}
//
// ******************** routine di scrittura su secure digital ************************
// ********************** write on sd card routine ******************************
//
void scrivi (void)
{
  codescr = 0;         // pone preventivamente a zero il codice descrizione errore
```

```
  filesd = SD.open("testo.txt", FILE_WRITE); //File in scrittura
  if (filesd)          // Se il file è stato aperto correttamente
  {
    filesd.println(bufferricezione); // Scrive su file i dati ricevuti
    filesd.close();  // Chiude il file su sd
    codescr = 4;     // segnala scrittura completata
    pulisci ();      // pulisce il buffer e l'indice di gestione
  }
  else
  {
    codescr = 3;     // segnala errore apertura file
    descrizione ();
  }
}
//
//  ****routine di lettura ed esposizione del contenuto del file memorizzato su sd ***
//  ********************** read sd card and display contents ************************
//
void leggi (void)
{
  pulisci ();               // pulisce preventivamente il buffer di ricezione
  lcd.clear();
  rig = 0;
  indicedisplay = 0;
  Serial.println (" ");
  filesd = SD.open(nomefile);  //apro il file
  if (filesd)                  //Se il file è stato aperto correttamente
  {
    while (filesd.available()) //Continua fino a che c'è qualcosa
    {
      ch = (filesd.read());   // memorizza in ch il carattere appena letto
      if ((ch >= 32) && (ch <= 125))// ignora i caratteri speciali (line feed, carriage return et
// similia)
      {
        bufferricezione [indicebuffer] = ch;
        if (!((ch == 32) && ((indicedisplay == 20) || (indicedisplay == 40) || (indicedisplay == 60)
|| (indicedisplay == 80)))) //ignora gli spazi ad inizio riga
        {
          visualizzacarattere ();  // visualizza il carattere letto
        }
      }
    }
    filesd.close(); //Chiudo file
  }
  else
  {
    codescr = 3;                 // segnala errore apertura file
    descrizione ();
  }
}
//
// ***********************routine di cancellazione del file su sd  *****************
// ********************************** delete file on SD ******************************
//
void cancellasd (void)
{
  SD.remove(nomefile);         //cancella il file da sd
  delay (100);
  if (SD.exists(nomefile))
    codescr = 3;               // segnala cancellazione ko
  else
    codescr = 5;               // segnala cancellazione completata
  descrizione ();
}
//
//
void setup()
{
  Serial.begin(9600);          // inizializza il monitor seriale
  lcd.begin (20, 4);           // inizializza il  dispaly lcd
  lcd.backlight();             // illumina il display lcd
  pinMode (10, OUTPUT);        // il pin CS e' collegato alla porta 10
  if (!SD.begin(10)) // verifica la presenza della secure digital
  {
    codescr = 3;     // segnala aperture KO
  }
  else
  {
```

```
    codescr = 2;      // segnala apertura OK
  }
  descrizione ();
}
//
//
void loop()
{
  if (Serial.available() > 0)  //se e' in arrivo qualcosa dalla tastiera
  {
    primavolta = 1;            // blocca la esposizione ripetitiva del testo memorizzato su sd
    comando = 99;              // inserisce un valore non valido in comando
    ch = Serial.read();        //legge (inserisce in ch) cio' che e' stato battuto sulla tastiera
    if (!(ch == '^'))          // se non e' stato ricevuto un "^" (un segnale di inizio comando)
    {
      bufferricezione [indicebuffer] = ch;  // memorizza il carattere ricevuto
      visualizzacarattere ();  // visualizza sul lcd e monitor seriale il carattere appena ricevuto
      indicebuffer++;          // predispone l'indice alla memorizzazione del prossimo carattere
      if (indicebuffer == 20)  // se il buffer di memorizzazione e' pieno
        scrivi();              // registra su sd il messaggio memorizzato
    }
    else                       // se invece e' stato ricevuto un "^"
    {
      ricevicomando ();        // attiva la routine di ricezione di un comando
      if (comando == 0)        // se e' arrivato un comando sconosciuto
        pulisci ();            // pulisce il buffer
      if (comando == 1)        // se e' arrivato il comando di scrittura
      {
        scrivi ();             // trascrive su sd il messaggio memorizzato
        descrizione ();        // segnala esito scrittura (ok o non ok)
      }
      if (comando == 2)        // se e' arrivato il comando di lettura
        leggi ();              // legge e visualizza il testo presente su sd
      if (comando == 3)        // se e' arrivato un comando di cancellazione file su sd
        cancellasd ();
      if (comando == 4)        // se e' pervenuto un comando di cancellazione del buffer
      {
        pulisci ();            // pulisce il buffer
        codescr    = 1;        // segnala buffer cancellato
        descrizione ();
      }
    }
  }
  else                         // se non e' in arrivo nulla da tastiera
  {
    if (primavolta == 0)
      leggi ();               // legge ed espone il file presente su sd
      delay (1000);           // attende un secondo prima di leggere ed esporre nuovamente il file
  }
}
```