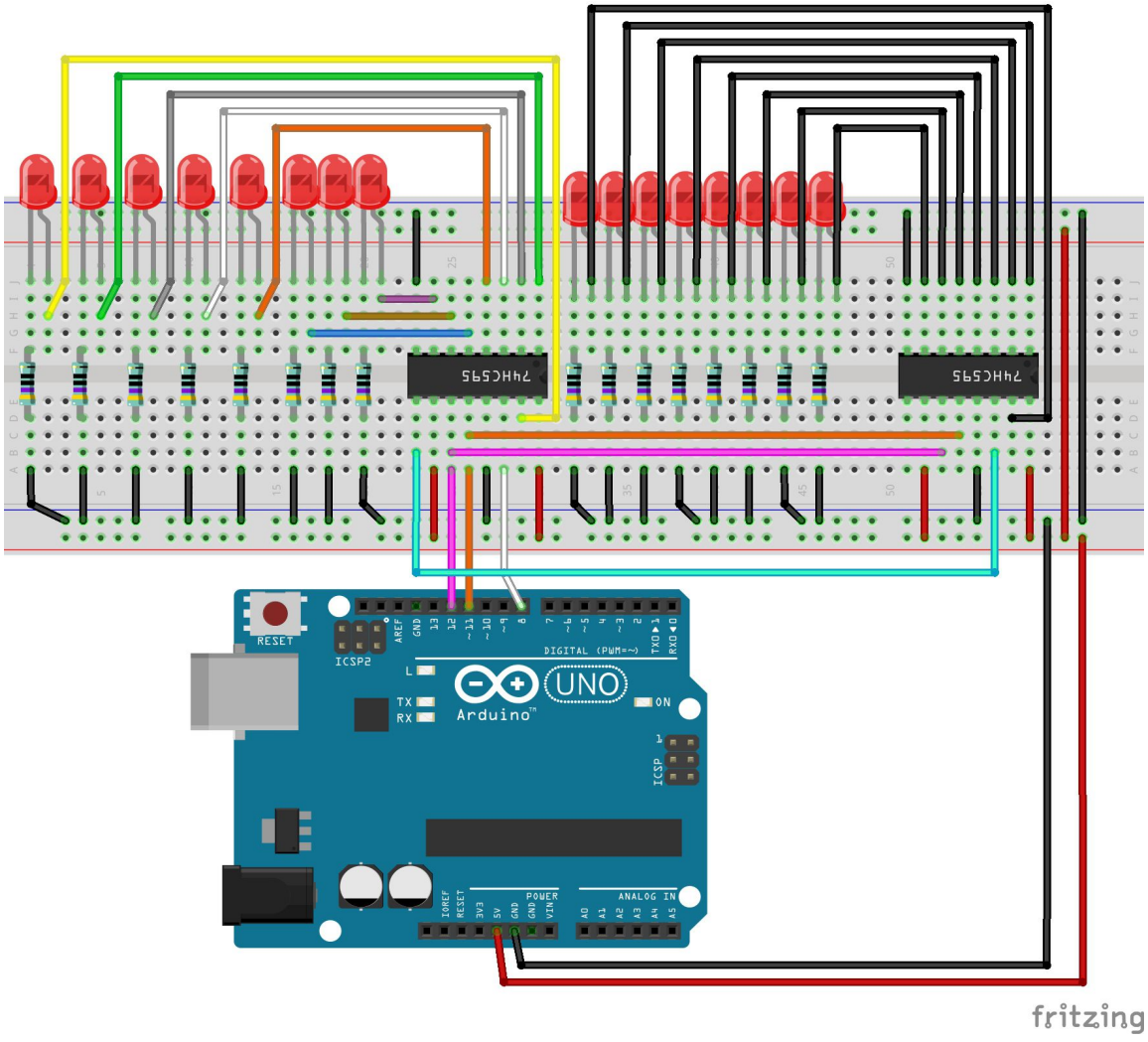
---------- **Il programma seguente non è stato eseguito durante il corso** ----------

Aggiungiamo una piccola modifica al software per ottenere il controllo di questi led e già che ci siamo creiamo una funzione con le tre istruzioni per la gestione dell'integrato 74HC595, in questo modo:

```
/*
 * 74HC595 comandato da serial monitor
 */

int dataPin = 8;
int latchPin = 11;
int clockPin = 12;

byte leds = 0;

void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, MSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}
void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);

  updateShiftRegister();  // chiamo la funzione di aggiornamento con la variabile leds = 0 per spegnere tutto

  Serial.begin(9600);
  delay(1000);
  Serial.println("Inserire il numero del led da accendere");
  Serial.println("da 0 a 7 oppure 'x' per spegnere tutto");
}

void loop()
{
  if (Serial.available()){
    char ch = Serial.read();              // ch contiene il valore ASCII del carattere inserito da tastiera
    if (ch >= '0' && ch <= '7'){
      int led = ch - '0';                 // sottraggo da ch il valore ASCII di zero e trasformo il risultato in numero intero
      bitSet(leds, led);                  // setto a 1 il bit = led nel byte leds
      updateShiftRegister();              //aggiorno i led
      Serial.print("Ho acceso il led: ");
      Serial.println(led);
    }
    if (ch == 'x'){
      leds = 0;
      updateShiftRegister();
      Serial.println("Spengo tutto");
    }
  }
}
```

Vediamo l'uso di un nuovo tipo di variabile con ***byte leds = 0***, questa variabile può contenere un numero intero che va da 0 (zero) a 255 decimale ovvero da 00000000 a 11111111 in binario cioè un numero a 8 bit. Essendo 8 le uscite del chip che utilizziamo è immediato capire quali led vogliamo accendere e quali spegnere se usiamo questa notazione binaria dove 1 vuol dire led acceso e 0 led spento. L'altra istruzione nuova che vediamo è ***bitSet(variabile_byte, numero_del_bit_da_settare)***, se ad esempio scriviamo ***bitSet(leds, 3) leds*** che è = ***00000000*** diventerà ***00001000*** cioè il bit numero 4 da destra (si conta da 0) verrà cambiato ad 1, se questo era già uguale ad 1 l'operazione lo lascerà inalterato come anche non verranno modificati altri bit eventualmente già posti ad 1. Useremo poi questa variabile ***leds*** per inviare gli 8 bit seriali ed accendere il led corrispondente. Per spegnere tutti led dovremo inviare il carattere x, con il quale la variabile ***leds*** verrà azzerata nuovamente. Fino ad ora eravamo abituati a visualizzare le

operazioni compiute da Arduino sul monitor seriale, mentre qui ne vediamo l'uso come periferica di input verso Arduino. Nel loop infatti controlliamo se nella memoria della porta seriale (buffer) sono presenti dei caratteri con l'istruzione ***Serial.available()***, che appunto restituisce il numero di caratteri presenti nel buffer, se l'if da come risultato "vero" leggiamo un carattere e lo mettiamo nella variabile (che definiamo essere di tipo carattere) ch con l'istruzione ***char ch = Serial.read()***. A questo punto controlliamo se abbiamo ottenuto un carattere tra zero e sette o la lettera "x". Da notare che il confronto non lo facciamo con i valori numerici 0 e 7 (confronto numerico di tipo matematico), bensì con i caratteri di tipo testo infatti questi sono racchiusi tra virgolette. Il confronto verrà effettuato come dicevamo, non con il valore numerico rappresentato ma con il valore del codice ASCII del carattere (per il carattere 0 vale 48). questo modo di procedere è dovuto al fatto che dal monitor seriale con l'istruzione read() otteniamo un carattere in formato byte, saremo noi che dovremo poi interpretarlo nel modo corretto. Ritornando, se il confronto è positivo per il primo caso, troviamo l'istruzione ***int led = ch – '0'*** che tradotta significa: al byte in ch sottraiamo il valore del carattere 0 e il risultato lo memorizziamo come valore numerico intero nella variabile led in questo modo applichiamo una variazione di tipo (da testo a numerico) della variabile. Ora abbiamo "l'indirizzo" del bit da settare a 1. Se invece dal confronto iniziale risulta che abbiamo premuto la x vengono azzerati direttamente gli 8 bit della variabile leds, in tutti gli altri casi il programma non fa nulla e resta in attesa di un carattere valido.
Fino ad ora non abbiamo mai usato il pin 9 del chip 74HC595, quello contrassegnato come Q7S (o Q7*) che è il serial out (uscita seriale). Collegando questo pin al pin data serial di un altro chip 74HC595 e collegando in parallelo i pin ST_CP e SH_CP dei due integrati possiamo raddoppiare le uscite cioè 16, sempre usando le solite tre uscite di Arduino.
Vediamo lo schema:



e ora il codice

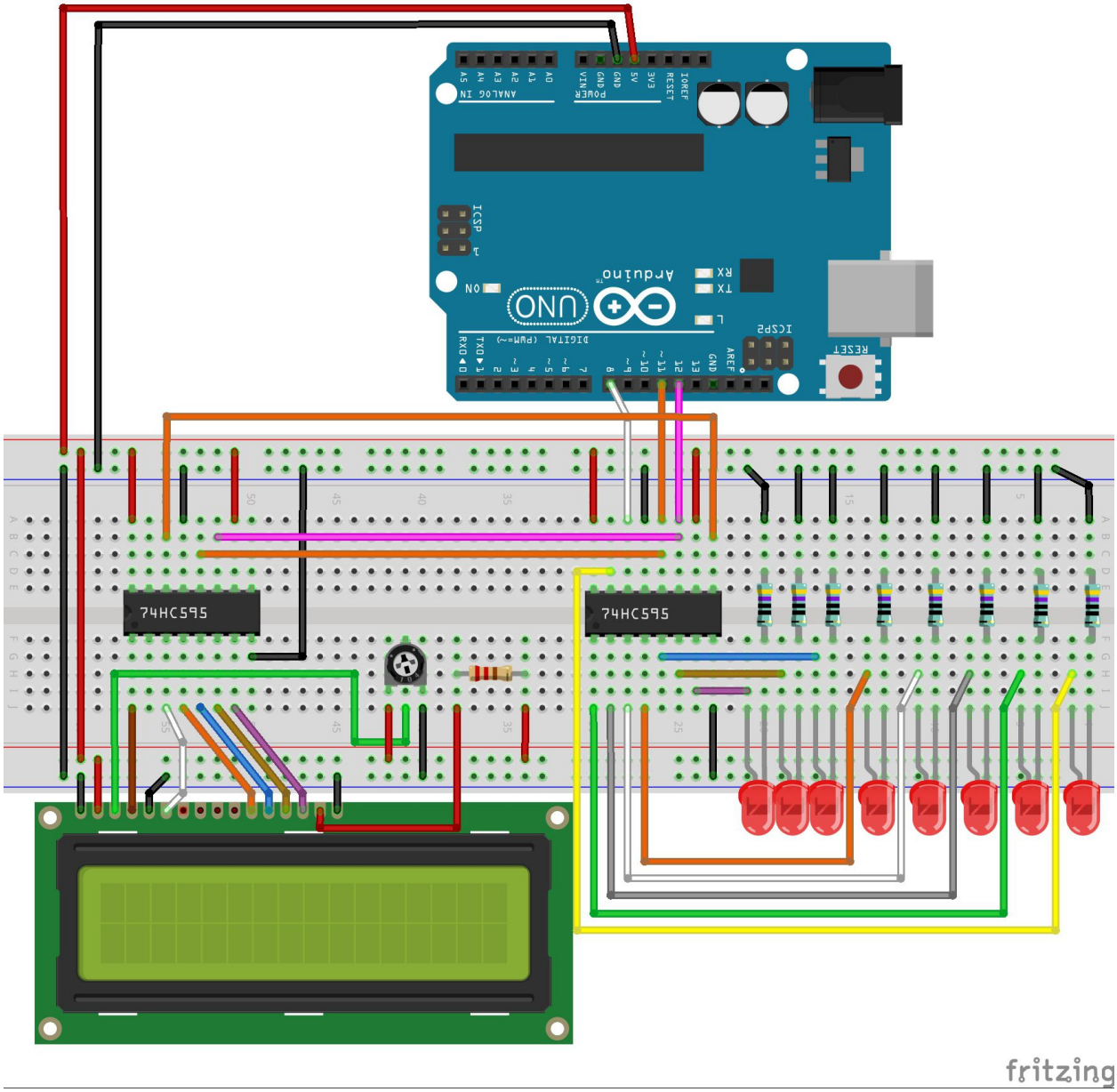
```
//************************************************************//
//  Name    : shiftOut12,
//  Notes   : Codice per usare 2 x 74HC595 Shift Register          //
//          : to count from 0 to 65535
//************************************************************

////Pin connected to DS of 74HC595
int dataPin = 8;
//Pin connected to ST_CP of 74HC595
int latchPin = 11;
//Pin connected to SH_CP of 74HC595
int clockPin = 12;

unsigned int numberToDisplay = 0;


void setup() {
  //set pins to output so you can control the shift register
  Serial.begin(9600);
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {
  // count from 0 to 65535 and display the number
  // on the LEDs
  for (numberToDisplay = 0; numberToDisplay < 65536; numberToDisplay++) {
    // take the latchPin low so
    // the LEDs don't change while you're sending in bits:
    digitalWrite(latchPin, LOW);
    // shift out the bits:
    shiftOut(dataPin, clockPin, MSBFIRST, numberToDisplay);
    shiftOut(dataPin, clockPin, MSBFIRST, (numberToDisplay >> 8));



    //take the latch pin high so the LEDs will light up:
    digitalWrite(latchPin, HIGH);
    // pause before next value:
    Serial.println(numberToDisplay);
    delay(50);
  }
}
```

Ho usato lo stesso codice del primo esempio con modifiche minime che sono:

1. ***numberToDisplay*** :deve contenere un numero a 16 bit (abbiamo 16 led) quindi non può più essere di tipo int (intero con segno) che al massimo possono memorizzare 32767 (2^15 – 1) il bit più significativo è riservato al segno (se = 1 numero negativo, se = 0 numero positivo). Per questo motivo diventa unsigned int (intero senza segno), in questo modo possiamo usare tutti e 16 i bit (2^16 – 1 = 65.535)
2. Il ciclo for andrà da 0 a 65.535
3. Dobbiamo inviare il dato seriale 2 volte, la prima volta così come lo otteniamo (vengono inviati solamente i primi 8 bit o gli ultimi 8 a seconda che sia MSBFIRST o LSBFIRST), per poter inviare gli altri 8 bit (da 8 a 15), dobbiamo usare un'operazione particolare che si chiama "***shift***" (scorrimento). Lo ***shift*** può avvenire verso destra **(>>)** o verso sinistra **(<<)** e l'entità dello

spostamento è indicata dal numero esempio:
A = 00000101 (= 5 decimale) se eseguiamo ***A << 3*** ora ***A = 00101000*** (= 40 decimale)
A = 00101000 (=40 decimale) se eseguiamo ***A >> 3*** ora ***A= 00000101*** (=5 decimale)
Notare che l'istruzione ***shiftOut(...)*** fa esattamente la stessa cosa quando invia i dati al chip. Con il secondo invio i dati precedenti vengono a loro volta fatti scorrere, escono dal chip attraverso il pin ***Q7S*** e vanno al registro del secondo integrato.

Si possono concatenare altri integrati ottenendo 24, 32, 40, … uscite, usando sempre lo stesso metodo e inviando i dati a gruppi di 8 uno per ogni chip usato.
Ovviamente potremo usare uno di questi integrati anche per controllare un display lcd che come abbiamo già visto necessita di sei collegamenti al nostro Arduino per poter funzionare, quindi saranno necessari sei pin anche per il 74HC595 e, per ragioni di compatibilità con diverse librerie, conviene lasciare vuoti i pin numero 2 (Q2) e 15 (Q0). Effettuiamo i collegamenti in questo modo:



si tratta del circuito precedente capovolto per visualizzare dalla parte giusta l'LCD
Per quanto riguarda lo sketch, per renderlo più semplice utilizzerò una libreria che a fronte di un po di studio, permette di gestire agevolmente anche un numero elevato di 74HC595 concatenati e sui quali

possiamo connettere lcd, led e display a segmenti. La libreria in questione di chiama McMajan e usa un buon numero di funzioni per gestire i chip. Vediamo ora lo sketch poi commenteremo la libreria.

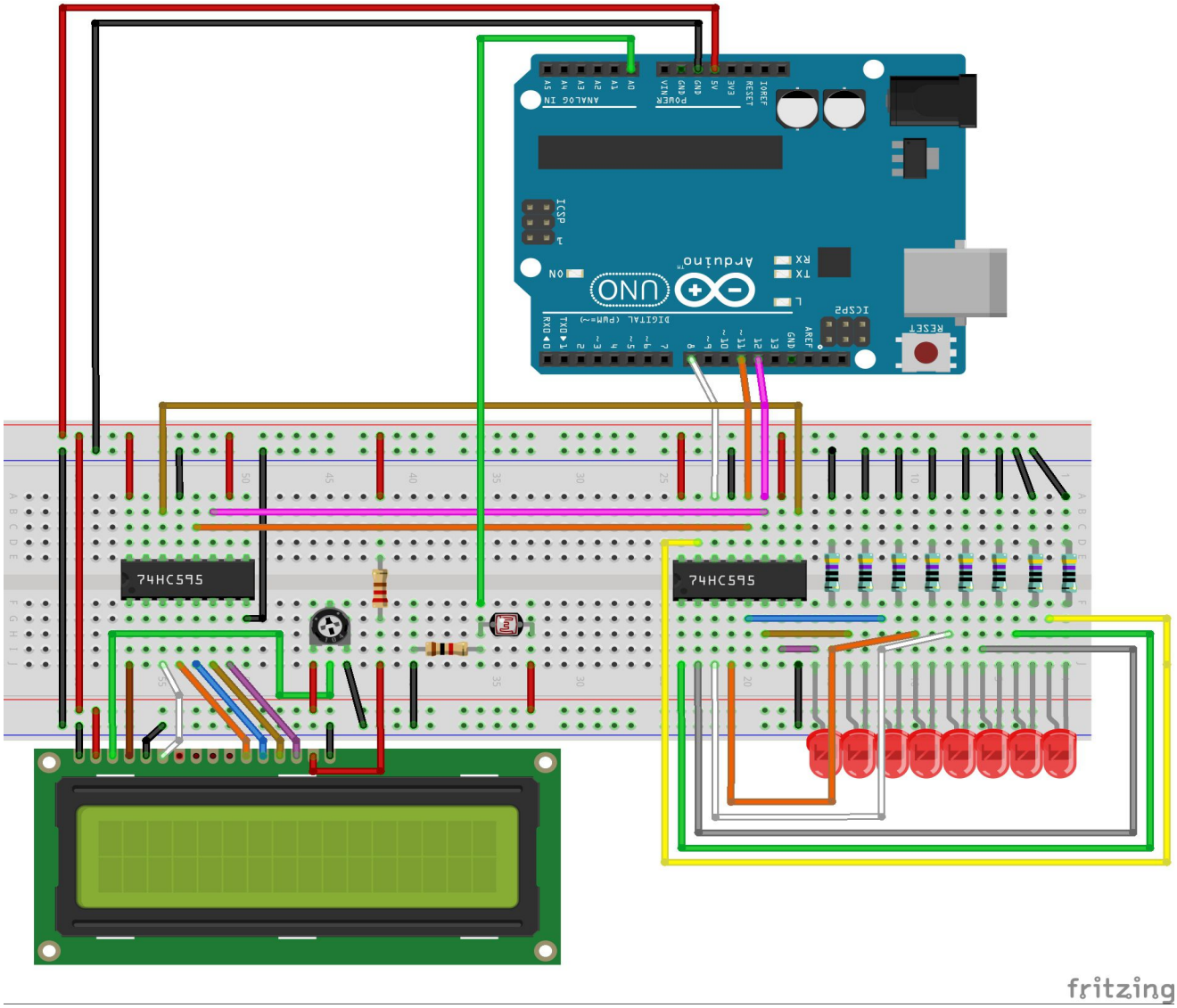
```
/*
 * _74Hc595LedLcd
 * utilizzo la libreria McMajan per gestire gli integrati
 * a cui ho connesso 8 led e un lcd
 */

#include <Ss_McMajan_Config.h>

// creo l'oggetto My595 per gestire i 2 integrati
hc595 My595(12,11,8,2); // latch12,clock11,data14,numero di 74hc595

unsigned char Leds;
unsigned char ledByte = 0;

byte effectId = 0;
byte prevEffect = 0;
byte effectRepeat = 0;
byte effectSpeed = 30;
int tempoLcd = 800;
unsigned long mioTempo = 0;

void setup(){
   Leds=0;
   // definisco il tipo di lcd usato
   My595.DisplayReset(LCD595_BASIC_DISPLAY_INIT | LCD595_MORELINES ,1); // display multilinea
   My595.SetCursor(0,0,1,1);                                            // cursore a 0,0, display tipo 1 sul secondo 595
   My595.DisplayWrite("Setup is OK...",1);                              // stringa da scrivere sul secondo 595
   delay(3000);                                                         // attesa...
   My595.DisplayClean(1);                                               // cancello display
}

void loop(){
   do{
      effectId = random(6);               // seleziono random l'effetto del gioco luci che voglio
   } while (effectId == prevEffect);      // tra 0 e 6, se è uguale al precedente ritento
   prevEffect = effectId;

   switch (effectId){                     // a seconda dell'effetto scelto, sempre in modo random
   case 0:                                // decido quante volte devo ripeterlo
      effectRepeat = random(1, 2);
       break;
   case 1:
      effectRepeat = random(1, 2);
      break;
   case 3:
      effectRepeat = random(1, 2);
      break;
   case 4:
      effectRepeat = random(1, 4);
      break;
   case 5:
      effectRepeat = random(1, 3);
      break;
   }

   for (int i = 0; i < effectRepeat; i++) {  // per ogni ripetizione dell'effetto scelgo la velocità
      effectSpeed = random(15, 90);          // di lampeggio dei led ed eseguo l'effetto
       switch (effectId){
      case 0:
```

```
60          effectA(effectSpeed);
61          break;
62       case 1:
63          effectB(effectSpeed);
64          break;
65       case 3:
66          effectC(effectSpeed);
67          break;
68       case 4:
69          effectD(effectSpeed);
70          break;
71       case 5:
72          effectE(effectSpeed);
73          break;
74       }
75    }
76 }
77
78 void effectA(int speed){
79    for (int i = 0; i < 8; i++){
80       for (int k = i; k < 8; k++){
81          regWrite(k, HIGH);
82          delay(speed);
83          regWrite(k, LOW);
84       }
85       regWrite(i, HIGH);
86    }
87    for (int i = 0; i < 8; i++){
88       for (int k = i; k < 8; k++){
89          regWrite(k, LOW);
90          delay(speed);
91          regWrite(k, HIGH);
92       }
93       regWrite(i, LOW);
94    }
95 }
96
97 void effectB(int speed){
98    for (int i = 7; i >= 0; i--){
99       for (int k = 0; k <= i; k++){
100         regWrite(k, HIGH);
101         delay(speed);
102         regWrite(k, LOW);
103      }
104      regWrite(i, HIGH);
105   }
106   for (int i = 7; i >= 0; i--){
107      for (int k = 0; k <= i; k++){
108         regWrite(k, LOW);
109         delay(speed);
110         regWrite(k, HIGH);
111      }
112      regWrite(i, LOW);
113   }
114 }
115
116 void effectC(int speed){
117    int prevI = 0;
118    for (int i = 0; i < 8; i++){
```

```
119         regWrite(prevI, LOW);
120         regWrite(i, HIGH);
121         prevI = i;
122         delay(speed);
123      }
124      for (int i = 7; i >= 0; i--){
125         regWrite(prevI, LOW);
126         regWrite(i, HIGH);
127         prevI = i;
128         delay(speed);
129      }
130   }
131
132   void effectD(int speed){
133      for (int i = 0; i < 4; i++){
134         for (int k = i; k < 8; k++){
135            regWrite(k, HIGH);
136            regWrite(7 - k, HIGH);
137            delay(speed);
138            regWrite(k, LOW);
139            regWrite(7 - k, LOW);
140         }
141         regWrite(i, HIGH);
142         regWrite(7 - i, HIGH);
143      }
144      for (int i = 0; i < 4; i++){
145         for (int k = i; k < 8; k++){
146            regWrite(k, LOW);
147            regWrite(7 - k, LOW);
148            delay(speed);
149            regWrite(k, HIGH);
150            regWrite(7 - k, HIGH);
151         }
152         regWrite(i, LOW);
153         regWrite(7 - i, LOW);
154      }
155   }
156
157   void effectE(int speed){
158     for (int i = 7; i >= 0; i--){
159         for (int k = 0; k <= i; k++){
160            regWrite(k, HIGH);
161            regWrite(7 - k, HIGH);
162            delay(speed);
163            regWrite(k, LOW);
164            regWrite(7 - k, LOW);
165         }
166         regWrite(i, HIGH);
167         regWrite(7 - i, HIGH);
168      }
169    }
170
171   void regWrite(int pin, bool state){
172      bitWrite(ledByte, pin, state);
173      My595.Set595Pin(ledByte,0);
174      My595.Send595();
175      //in questa che è la funzione richiamata più spesso
176      //aggiungo la scrittura sul display lcd
177      if (millis() - mioTempo > tempoLcd) {
178         My595.SetCursor(Leds%13,Leds%2,1,1);            // posizione cursore (leds mod 13, leds mod 2)
179         My595.DisplayChar(32+Leds%96,1);                // scrivo 1 carattere alla volta partendo da ASCII 32
180         Leds++;
181         mioTempo = millis();
182      }
183   }
```

Questo Sketch è sensibilmente più lungo perché si occupa di definire diverse condizioni come durata, numero di ripetizioni e tipo di effetto luminoso per ottenere un gioco di luci da applicare ai led connessi al primo chip 74HC595, poi in ultimo scrive ad uno ad uno tutti i caratteri stampabili sull'lcd.
Il programma non è parametrizzato per cui aggiungendo un altro 74HC595 con 8 led non verrebbe gestito dagli effetti luminosi, per fare questo sono necessarie delle modifiche che però rischiavano di rendere meno chiaro lo sketch.
La riga 10 crea l'oggetto che è incaricato di gestire i due registri di scorrimento. Nel setup viene definito il tipo di display utilizzato e viene resettato .
All'inizio del loop vediamo una nuova istruzione: ***do {…} while …*** Questa istruzione ripete quello che c'è tra le parentesi graffe fino a che non è soddisfatta la condizione scritta dopo il while. Nel nostro caso viene assegnato a caso un numero tra 0 a 6 alla variabile effectId finché non risulta diverso dal numero assegnato la volta precedente. Subito dopo troviamo l'istruzione ***switch (variabile) { case valore_variabile_1: case valore_variabile_2: default: ... }*** a seconda del valore della variabile eseguo le istruzioni all'interno del ***case*** corrispondente se il valore è diverso da quelli considerati eseguo le istruzioni scritte in ***default***, se all'interno di un ***case*** devo definire delle variabili allora è obbligatorio aggiungere le parentesi graffe. Dopo aver eseguito le istruzioni previste, per velocizzare il programma evitando di controllare anche tutti gli altri case, si aggiunge di solito l'istruzione ***break*** che interrompe l'esecuzione di tutto lo ***switch*** e continua con il comando successivo. Notare che le linee di programma con ***case*** e ***default*** terminano con due punti (:) e non punto e virgola (;)
A seguire troviamo le funzioni che creano i giochi di luce e che per accendere o spegnere i led richiamano la funzione ***regWrite(int pin, bool state)*** che richiede 2 parametri, il primo è l'uscita (da Q0 a Q7) da impostare e come secondo parametro lo stato dell'uscita (alto o basso). Questi 2 valori vengono passati all'istruzione ***bitWrite(byte_var, bit_num, stato)*** con la quale siamo in grado di impostare a 1 o a 0 il valore di un bit di una variabile di tipo byte, nel nostro caso ***ledByte***. Le due istruzioni successive che fanno parte della libreria McMajan sono: ***Set595Pin(byte, numero_chip)*** che esegue l'invio degli 8 bit seriali al chip indicato e ***Send595()*** che trasferisce i valori alle uscite collegate.
Le ultime quattro righe si occupano di gestire l'lcd scrivendo ogni 800 millisecondi uno dopo l'altro i caratteri stampabili sul display utilizzando altre due funzioni della libreria che sono: ***SetCursor(X_pos, Y_pos, tipo_lcd, numero_chip)*** che ci permette di posizionare il cursore del display alla riga X e colonna Y, tipo_lcd normalmente è uguale ad 1 e numero chip è come sempre il numero di integrato al quale è collegato il display (partendo da zero), infine ***DisplayChar(carattere, numero_chip)*** scrive il carattere indicato nella posizione corrente del cursore del display. L'elenco di tutte le funzioni di questa libreria si trova on-line ma per comodità le ho riportate in un documento pdf di veloce consultazione.

Manteniamo ora lo stesso circuito a cui aggiungiamo una resistenza da 1K ohm e una fotoresistenza collegate in serie tra positivo e negativo e il centro lo colleghiamo con il pin analogico A0 di Arduino come nel seguente schema



e carichiamo lo sketch seguente:

```
/*
 * _74Hc595LedLcdFtc
 * utilizzo la libreria McMajan per gestire gli integrati
 * a cui ho connesso 8 led e un lcd
 * per visualizzare in forma scritta e visuale la luminosità rilevata da
 * una fotocellula connessa al pin A0
 */

#include <Ss_McMajan_Config.h>

// creo l'oggetto My595 per gestire i 2 integrati
hc595 My595(11,12,8,2); // latch12,clock11,data14,numero di 74hc595

int ftc = 0;
int luce = 0;
unsigned char leds;

void setup(){
   leds=0;
   // definisco il tipo di lcd usato
   My595.DisplayReset(LCD595_BASIC_DISPLAY_INIT | LCD595_MORELINES ,1); // display multilinea
   My595.SetCursor(0,0,1,1);                                           // cursore a 0,0, display tipo 1 sul secondo 595
   My595.DisplayWrite("Setup is OK...",1);                             // stringa da scrivere sul secondo 595
   delay(3000);                                                        // attesa...
   My595.DisplayClean(1);                                              // cancello display
}

void loop(){
   int leggiFtc = analogRead(ftc);              // essendo un pin analogico non devo dichiarare nulla nel setup
   int numLedAccesi = leggiFtc / 101;            // max valore ftc= 910 / 9 combinazioni = 101
   if (numLedAccesi > 8) numLedAccesi = 8;
   leds=0;
   for (int i = 0; i< numLedAccesi; i++) {      // per ogni led acceso
      bitSet(leds,i);                           // metto a uno il bit corrispondente in leds
   }
   luce= map(leggiFtc,0,910,0,100);             // trasformo in valore da 0 a 100 la lettura analogica
   regWrite();
   delay(500);
}

void regWrite(){
   My595.Set595Pin(leds,0);                     // accendo i led
   char myBuf[12];                              // creo una variabile temporanea
   My595.DisplayClean(1);                       // pulisco il display collegato al chip 1
   My595.SetCursor(1,0,1,1);                    // posizione cursore carattere 2, riga 1, tipo 1, chip 1
   My595.DisplayWrite("Luminosita'= ",1);       // scrivo sul chip 1
   My595.SetCursor(6,1,1,1);                    // posiziono cursore carattere 6, riga 2, tipo 1, chip 1
   My595.DisplayWrite(itoa(luce,myBuf,10), 1);  // scrivo il numero contenuto in luce, dopo averlo trasformato in strings
   //My595.DisplayWrite(luce,1);                // cosi non funziona!!
   My595.SetCursor(9,1,1,1);                    // mi posiziono dopo il numero appena scritto
   My595.DisplayWrite("%",1);                   // aggiungo il carattere percento
   My595.Send595();                             // invio tutto quello che ho fatto prima alle uscite dei 74HC595
}
```

Il setup è lo stesso di prima, alla riga 29 leggo il valore di tensione fornito dal partitore composta dalla resistenza fissa da 1000 Ohm e la fotocellula che in piena luce ha anch'essa una resistenza di circa 500 Ohm e al buio circa 100K Ohm che varierà da 0 (al buio = 0 Volt) a 1023 (piena luce = 5 Volt). Verificando sperimentalmente il valore massimo di tensione che il circuito mi fornisce è di 4,76V e infatti il valore massimo che ottengo è circa 910-915 sul pin A0, per cui nella riga 30 in cui eseguo il calcolo di quanti led accendere (da 0 a 8 quindi 9 passi) divido il valore ottenuto dal la lettura per 101 (910 / 9 = 101). Alla linea 36 trasformo il valore in un numero da 0 a 100 per poterlo scrivere sul display come percentuale. Alla riga 41 inizia la funzione di invio dati ai due 74HC595, prima scrivo nel registro del primo integrato il valore corrispondente ai led da accendere poi dopo aver pulito lo schermo scrivo un testo sulla prima linea e sulla seconda linea, siccome la libreria che usiamo non gestisce la scrittura di un dato numerico, devo per forza convertirlo in alfanumerico (o stringa). Per fare questo devo creare una variabile di tipo matrice di caratteri (***char myBuf[]***) di lunghezza adeguata e poi usando la funzione (integer to ascii) ***itoa(numIntero, char variabile_array [], baseNumIntero)*** propria del linguaggio C

effettuare la trasformazione, in questo caso il risultato viene inviato direttamente al registro del secondo integrato aggiungendo il carattere % al termine del valore. Con l'ultima istruzione invio tutti i dati memorizzati alle rispettive uscite (led e lcd).