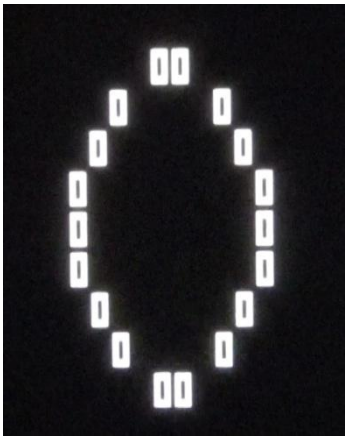
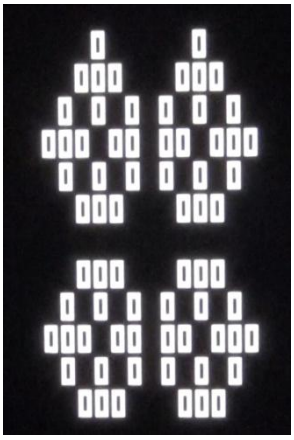
# 51- Conway: il gioco della vita – game of life (some notes at section end)





Il "**gioco della vita**" e' una programma di gestione di un universo cellulare che vive, si riproduce e muore seguendo le regole enunciate da John Conway e cioe:
- una cella vuota si riempie (nasce una cellula) se e' adiacente a tre cellule vive
- una una cellula viva muore se e' adiacente a meno di tre o a piu' di quattro celle piene (muore perche' il mondo circostante e' sotto o sovra popolato).

Qui' il film del progetto.
In questo progetto si utilizza la libreria TVout.h per gestire uno schermo TV sul quale viene esposta la vita delle cellule. L'utilizzo della libreria e delle matrici dell'universo delle cellule (due matrici, una con la situazione corrente ed una nella quale il programma costruisce la generazione successiva) ha comportato un ingente occupazione di memoria ed ha percio' richiesto l'impiego di un Arduino mega, piu' capiente e performante del classico Arduino UNO.

Nella costruzione del circuito e' stato inserito un potenziometro per rallentare o velocizzare la generazione delle cellule, ed un pulsante con il quale provocare la generazione di un nuovo schema iniziale (quatto gruppi di cellule adiacenti, sparsi a caso nella matrice) o, a seguito di una pressione prolungata, l'avvio di una serie di schemi predefiniti.

L'utilizzo della libreria TVout.h sembra indurre svariati problemi tra i quali una sensibile modifica delle funzionalita' dell'orologio interno di Arduino, l'impossibilita' di utilizzare al meglio le funzioni seriali, l'impossibilita' di utilizzare le porte analogiche (eccetto la A1) ed altri problemi che provocano, saltuariamente, il blocco casuale ed ingiustificato del programma.

La libreria TV.out sembra inoltre consentire l'esecuzione di un limitato numero di comandi di gestione del video per cui sono state create delle routine ognuna delle quali esegue un diverso comando. In questo modo i comandi sono stati scritti una sola volta per poter poi essere piu' volte richiamati all'interno del programma, limitando quindi la loro ripetizione fisica, ma non il loro utilizzo logico.

In pratica la libreria TVout ha imposto la scrittura di un programma complesso, con passaggi apparentemente inutili e quasi incomprensibili.

Prima di procedere alla compilazione del programma deve ovviamente essere installata, se non gia' presente, la libreria:
- TVout.h, reperibile gui.

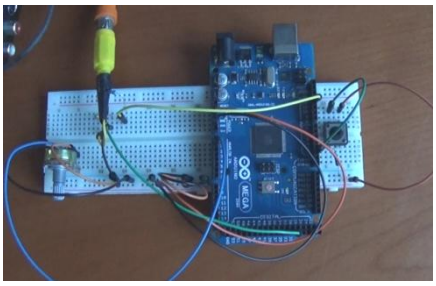
Per installare la libreria e' necessario seguire la seguente procedura:
- download della libreria in formato compresso
- installare la nuova libreria andando in IDE-> sketch-> includes Library-> add .zip library
- verificare l'avvenuta installazione (andando in IDE-> sketch-> includes Library-> Contributed 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 notes about this project, translated by google translator



The "**game of life**" is a program that simulates a cellular universe that lives, reproduces and dies following some rules set forth by John Conway:
- an empty position fills up (borns a cell) if is adjacent to three living cells
- a living cell dies if is adjacent to less than three or more than four living cells (dies because the the surrounding world is under or over populated)

Here the project movie. In this project we use the TVout.h library to manage a TV screen on which cells life is exposed. The library use and the cells universe management, has involved a large amount of memory and therefore required the use of an Arduino Mega, which is considerably more powerful than the classic Arduino UNO.

In the circuit construction, a potentiometer has been inserted to slow or speed cells life, and a button to trigger a new initial pattern (four adjacent cell groups, randomly scattered in the array ) or, after a long press, to starting some predefined universes.

The TVout.h library seems causes a number of problems, including a significant change in the functionality on Arduino internal clock, the inability to use some serial comands, the inability to use analogue ports (except A1) and other problems that occasionally cause a, random and unjustified, program blockage.

The TV.out library also seems to allow a limited numbers of video commands (in term of occurencies) , for which some routines have been created each of which executes a command. In this way, the commands have been written only one time, to be recalled more times, if needed, within the program, thus limiting their physical repetition, but not their logical use.

In practice, the TVout library imposed a complex program, with seemingly unnecessary and almost incomprehensible passages.

Before proceeding to program compilation must be obviously installed, if not already done, the library:
- TVout.h found here

For library installation:
- download library in compressed form;
- Install it via IDE-> sketch-> includes Library-> add .zip library
- After installation please verify the library. It must be present in IDE-> sketch-> includes Library-> Contributed library

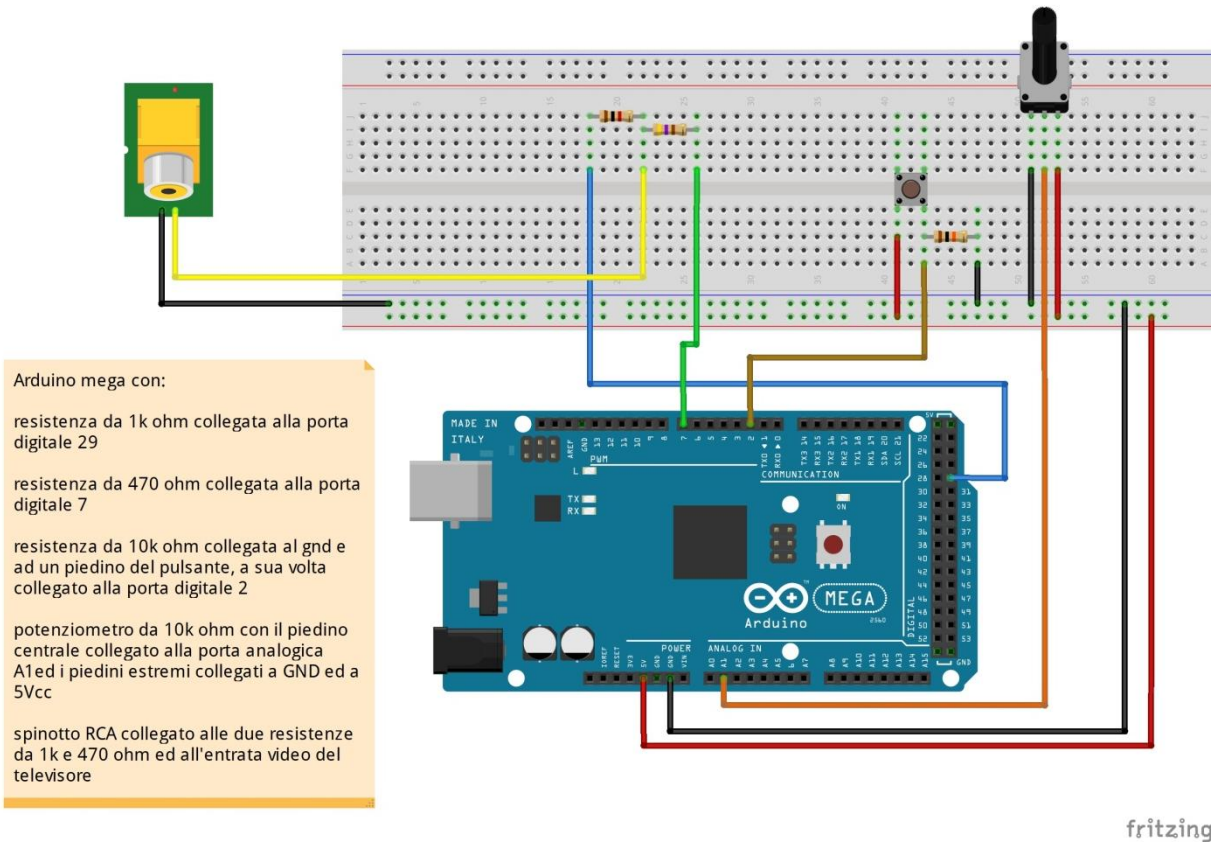
**Note:** This project and this note is part of a series that sees, as main characters, Arduino and some of connectable components. For most projects there is also a video on youtube.
- Projects collection
- Movies on youtube
- About Arduino and components (italian; pdf will be downloaded in your download 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

- Uno spinotto RCA
- Un potenziometro da 10k ohm
- Una resistenza da 470 ohm
- Una resistenza da 1k ohm
- Una resistenza da 10k ohm
- Un pulsante
- Un Arduino mega
- Un televisore

## 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.

   Il "gioco della vita" e' una programma di gestione di una matrice che simula un universo
   cellulare che vive, si riproduce e muore seguendo le regole enunciate da John Conway e cioe:
   - una cella vuota si riempie (nasce una cellula) se e' adiacente a tre celle piene (tre cellule
     vive)
   - una cella piena (una cellula viva) si svuota (muore) se e' adiacente a meno di tre o a piu' di
     quattro celle piene (muore perche' il mondo circostante e' sotto o sovra popolato).

   Il potenziometro consente di accelerare o decelerare la nascita di una nuova generazione mentre
   il pulsante consente di interrompere l'universo corrente e di ripartire con un uovo universo,
```

```
   contenente quattro gruppi di cellule (ogni gruppo e' composto da tre o quattro cellule) disposte
   a caso sulla matrice.  Una pressione prolungata del pulsante (piu' di due secondi) innesca la
   generazione di alcuni schemi predefiniti.

 ---------------------------------------------------------------------------------------------------
   Warning: cut&paste from PDF to IDE loses formatting. To restore it press CTRL + T.

  The "game of life" is a program that simulates a cellular universe that lives, reproduces and dies
  following the rules set forth by John Conway and that is:
  - an empty cell fills up (borns a cell) if is adjacent to three full cells (three living cells)
  - a full cell (a living cell) empties (dies) if is adjacent to less than three or more than four
    full cells (dies because the the surrounding world is under or over populated)

  The potentiometer allows to accelerate or decelerate the birth of a new generation while the
  button allows you to interrupt the current universe and to start with a new universe,  containing
  four groups of cells (each group is composed of three or four cells) arranged randomly. A long
  press (more than two seconds) triggers some predefined universes

*/
#define pulsante 2              // il pulsante e' collegato alla porta 2
#include <TVout.h>              // libreria di gestione dell'output su tv
#include <fontALL.h>            // libreria di gestione dei font
TVout TV;
int  statopulsante       = 0;   // stato del pulsante (1 = premuto; 0 = non premuto)
byte schemacorrente [541];      // matrice generazione corrente 0 = cella libera,
// 1 = cella occupata, 2 = cella non usabile (bordi esterni),
byte schemafuturo   [541];      // matrice della generazione successiva
int indice               = 0;   // indice di scorrimento delle matrici delle generazioni
int posizione            = 0;   // posizione della cella in via di analisi
int semaforopulsante     = 0;   // semaforo che indica se il pulsante e' stato premuto
float tempoinizio        = 0;   // momento di inizio della pressione del pulsante
float tempocorrente      = 0;   // momento corrente
int seme                 = 0;   // seme per l'inizializzazione del generatore di numeri a caso
int indgen               = 0;   // indice utilizzato nella generazione casuale delle celle iniziali
int indgen1              = 0;   // indice per la generazione iniziale di triplette o quadriplette
int posizione            = 0;   // posizione della cellula in via di analisi
int resto                = 0;   // zona utilizzata nella valutazione della posizione iniziale
int numerocellule        = 0;   // numero di cellule da mettere vicino alla cellula iniziale
// durante la fase di creazione
int poscellula           = 0;   // cellula intorno alla quale devono essere generate altre cellule
int posizionecontigua    = 0;   // posizione della cellula adiacente, in via di creazione
int posizionepapabile    = 0;   // posizione, da sottoporre a verifica, della nuova cellula
int posizionenonusabile  = 0;   // semaforo posizione : 0 = usabile, 1 = non usabile
int celluleadiacenti     = 0;   // numero delle cellule adiacenti alla cellula in esame
float iniziogenerazione  = 0;   // momento di inizio delle generazione corrente
float tempo              = 0;   // tempo, di intervallo tra una generazione e l'altra
int potenziometro        = 0;   // valore fornito dal potenziometro
int tempogenerazione     = 0;   // tempo della generazione corrente
int tabpredefinita [50] =       // matrice schemi predefiniti
{
  6, 44, 78, 79, 80,                                  // glider
  230, 265, 266, 267, 301, 303, 338,                  // piccola esplosione
  196, 198, 200, 232, 236, 268, 272, 304, 308, 340, 342, 344, // esplosione
  229, 230, 231, 232, 233, 234, 235, 236, 237, 238, //  riga

};
int indtabpred           = 0;       // indice di scorrimento della tabella dei predefiniti
int numgen               = 0;       // numero delle generazioni passate
int numgentot            = 999999;  // numero max di generazioni (inizialmente 999999)
int semaforoseqprog      = 0;
int indicesegnalazione   = 0;       // indice utilizzato nella esposizione delle descrizioni
char tabsegnalazione [12] [36] =    // tabella delle descrzioni
{
  {"Il gioco della vita,"},               //0  the game of life
  {"basato sulle regole di Conway      "}, //1  based on Conway's rules
  {"Conway's Game of life              "}, //2
  {"          REGOLE                   "}, //3  Rules
  {"una cellula nasce se e' adiacente  "}, //4  a cell born if is near
  {"a 3 cellule                        "}, //5  to 3 cells
  {"una cellula vive se adiacente a    "}, //6  a cell lives i is near to
  {"2 o 3 cellule                      "}, //7  2 or 3 cells
  {"una cellula muore se e' adiacente a"}, //8  a cell dies if is near to
  {"meno di 2 o piu' di 4 cellule      "}, //9  less than 2 or more than 4 cells
  {"premi il pulsante...               "}, //10 Push the button…
  {"                                   "}, //11
};

/*
```

# Arduino: Conway – il gioco della vita – the game of life

```
// ********************************** Attenzione - Warning************************************
//sembra che la libreria TVout consenta di eseguire un limitato numero di comandi con il prefisso
"TV" (in termini di numero di ripetizioni all'interno di un programma) per cui le prossime routine
hanno il solo scopo di eseguire alcuni comandi con il prefisso "TV", che vengono poi piu' volte
richiamati all'interno del programma. In questo modo si limita il numero fisico di detti comandi,
senza pero' limitarne l'utilizzo logico.
-------------------------------------------------------------------------------------------------
The TV.out library seems to allow a limited numbers of video commands (in term of occurencies), for
which the following routines have been created each of which executes a different command. In this
way, the commands have been written only one time, to be recalled more times if needed, within the
program, thus limiting their physical repetition, but not their logical use.*/
//
// ******************  routine per la esecuzione del comando TV.println **********
//
void TVprintln ()
{
  TV.println ("");
}
//
// **************  routine di pausa di 3 secondi ***************************
//
void TVdelay3000 ()
{
  TV.delay (3000);
}
//
// ******************** routine di pulizia dello schermo ********************
//
void TVclearscreen()
{
  TV.clear_screen ();
}
//
// ********** routine di selezione del set di caratteri con font 6x8 ***********
//
void TVselectfont6x8 ()
{
  TV.select_font(font6x8);      // seleziona la dimensione dei caratteri (6X8)
}
//
// ********** routine di selezione del set di caratteri con font 4x6 ***********
//
void TVselectfont4x6 ()
{
  TV.select_font(font4x6);      // seleziona la dimensione dei caratteri ( 4x6)
}
//
// ************* routine di esposizione delle descrizioni ******************
// ******************* description exposure routine  ************************
//
void segnala ()
{
  TV.println (tabsegnalazione [indicesegnalazione]);
}
//
// ********* routine di conteggio delle cellule adiacenti *********************
// **************** Counting adjacent cells  ***********************************
//
void contaadiacenti ()
{
  celluleadiacenti = 0;
  if (schemacorrente [indice - 37] == 1)
    celluleadiacenti ++;
  if (schemacorrente [indice - 36] == 1)
    celluleadiacenti ++;
  if (schemacorrente [indice - 35] == 1)
    celluleadiacenti ++;
  if (schemacorrente [indice - 1] == 1)
    celluleadiacenti ++;
  if (schemacorrente [indice + 1] == 1)
    celluleadiacenti ++;
  if (schemacorrente [indice + 37] == 1)
    celluleadiacenti ++;
  if (schemacorrente [indice + 36] == 1)
    celluleadiacenti ++;
  if (schemacorrente [indice + 35] == 1)
    celluleadiacenti ++;
}
```

```
//
// ****************** routine di esposizione dello schema predefinito glider **************
// ********************** predefined schema "glider" ***********************************
//
void glider ()
{
  schemainiziale ();      // pulisce la matrice
  TVclearscreen();
  TV.set_cursor (25, 30);
  TVselectfont6x8 ();
  TV.println ("glider");
  TVdelay3000 ();        // pausa di 3 secondi
  TVselectfont4x6 ();
  for (indtabpred = 0; indtabpred < 5; indtabpred++)
  {
    indice = tabpredefinita [indtabpred];
    schemacorrente [indice + 36] = 1;
  }
  numgen = 0;
  numgentot = 50;
  semaforoseqprog++;
  esponischemacorrente ();
}
//
//  ******** routine di esposizione dello schema predefinito "piccola esplosione" *************
//   *************** "little explosion" predefined schema ******************************
//
void piccolaesplosione ()
{
  schemainiziale ();            // pulisce la matrice corrente
  TVclearscreen();
  TV.set_cursor (18, 30);
  TVselectfont6x8 ();
  TV.println ("piccola esplosione");
  TVdelay3000 ();               // pausa di 3 secondi
  TVselectfont4x6 ();
  for (indtabpred = 5; indtabpred < 12; indtabpred++)
  {
    indice = tabpredefinita [indtabpred];
    schemacorrente [indice + 36] = 1;
  }
  numgen = 0;
  numgentot = 35;
  semaforoseqprog++;
  esponischemacorrente ();
}
//
//  ********* routine di esposizione dello schema predefinito "esplosione" ***************
//  ***************** "explosion" predefined schema ***************************************
//
void esplosione ()
{
  schemainiziale ();            // pulisce la matrice corrente
  TVclearscreen();
  TV.set_cursor (30, 30);
  TVselectfont6x8 ();
  TV.println ("esplosione");
  TVdelay3000 ();               // attende tre secondi
  TVselectfont4x6 ();
  for (indtabpred = 12; indtabpred < 24; indtabpred++)
  {
    indice = tabpredefinita [indtabpred];
    schemacorrente [indice + 36] = 1;
  }
  numgen = 0;
  numgentot = 40;
  semaforoseqprog++;
  esponischemacorrente ();
}
//
//  ***************** routine di esposizione dello schema predefinito "riga" *****************
// ************************ "line" predefined schema ***************************************
//
void diecicelle ()
{
  schemainiziale ();            // pulisce la matrice corrente
  TVclearscreen();
  TV.set_cursor (35, 30);
```

```
  TVselectfont6x8 ();
  TV.println ("riga");
  TVdelay3000();                // lancia la routine di pausa di 3 secondi
  TVselectfont4x6 ();
  for (indtabpred = 24; indtabpred < 34; indtabpred++)
  {
    indice = tabpredefinita [indtabpred];
    schemacorrente [indice + 36] = 1;
  }
  numgen = 0;
  numgentot = 50;
  semaforoseqprog++;
  esponischemacorrente ();
}
//
// **** routine di ritorno alla sequenza random dopo la esecuzione delle sequenze programmate ****
// ********* return to a new random schema after the execution of the predefined schemas *****
//
void riprendirandom ()
{
  schemainiziale ();            // pulisce la matrice corrente
  TVclearscreen();
  TV.set_cursor (0, 30);
  TVselectfont6x8 ();
  TV.println ("fine seq. programmata");
  TVprintln ();
  TV.println ("inizia sequenza a caso");
  TVdelay3000();                // lancia la routine di pausa di 3 secondi
  TVselectfont4x6 ();
  semaforoseqprog = 0;
  numgentot = 999999;
  numgen = 0;
  sequenzarandom ();
}
//
// ***************** routine di esposizione dello schema corrente ***************************
// ********************** shows te current universe ************************
//
void esponischemacorrente ()
{
  TV.set_cursor (0, 0);
  for (indice = 0; indice < 540; indice++)
  {
    if (!( schemacorrente [indice] == 2))
    {
      resto = indice % 36;
      if (resto == 0)
        TVprintln ();
      if (schemacorrente [indice] == 1)
        TV.print ("0");
      if (schemacorrente [indice] == 0)
        TV.print (" ");
    }
  }
  iniziogenerazione = millis ();  // memorizza il momento di inizio della generazione corrente
}
//
//  ** routine di controllo di una posizione papabile (fase di creazione dello schema iniziale)**
//  *** check a possibile position for a new cell (in the initial schema creation) **********
//
void controllaposizione ()
{
  posizionenonusabile = 1;   // pone preventivamente a 1 il semaforo di posizione non usabile
  if (schemacorrente [posizionepapabile] == 0)
    posizionenonusabile = 0;   // pone a zero il semaforo di posizione non usabile
}
//
// **** routine di generazione delle tre triplette (o quadriplette) iniziali di cellule contigue **
//  *** creating  three or four adjacent cells (during initial cration) ***************************
//
void sequenzarandom ()
{
  schemainiziale ();                        // prepara la matrice vuota generazione
  schemacorrente [71]  = 2;                 // delimita l'angolo superiore destro
  schemacorrente [37]  = 2;                 // delimita l'angolo superiore sinistro
  schemacorrente [469] = 2;                 // delimita l'angolo inferiore sinistro
  schemacorrente [503] = 2;                 // delimita l'angolo inferiore destro
  for (indgen = 0; indgen < 4; indgen++)    // crea quattro gruppi di cellule
```

```
  {
    //
    // ******** posiziona la prima cellula della tripletta o quadripletta ******************
    // ************ positioning the first cell of a group *********************
    //
    posizionenonusabile = 1;                // pone ad  1 il semaforo di posizione non usabile
    while (posizionenonusabile == 1)        // cerca, nella matrice corrente, una cellula vuota
    {
      posizione = random (468);             // genera un numero a caso per  il possibile
// posizionamento di una cellula
      posizione = posizione + 36;           // non considera la prima riga non occupabile
      posizionepapabile = posizione;
      controllaposizione ();                // verifica l'usabilita' della posizione trovata
    }                                       // esce dal ciclo di while quando il semaforo
//  posizionenonusabile = 0
    schemacorrente [posizione] = 1;         // occupa la posizione
    //
    // ***** decide quante cellule (3 o 4)devono comporre l'iniziale gruppo di cellule ****
    // *****how many cels (3 or 4) in the initial cells groups ********************
    //
    numerocellule = random (1);            // genera un numero a caso da 0 a 1
    numerocellule = numerocellule + 2;     // numero delle cellule da creare
    //
    // ******** posiziona le cellule vicino a quella in esame *****************************
    // ******* positions a new cell neara the current cell *********************************
    //
    poscellula = posizione;
    for (indgen1 = 0; indgen1 <= numerocellule; indgen1 ++)   // loop di creazione cellule contigue
    {
      posizionenonusabile = 1;               // setta a 1 il semaforo posizione non usabile
      while (posizionenonusabile == 1)
      {
        posizionecontigua = random (7);      // genera la possibile posizione della nuova cellula
        // (numerazione in senso orario: la posizione 0 e' la posizione in alto a sinistra).
        //
        //  ******** calcola e verifica la disponibilita' della posizione della cellula da creare
        // ***************  check the position availability for a new cell **********************
        //
        switch (posizionecontigua)
        {
          case 0:
            posizionepapabile = poscellula - 37;
            if (posizionepapabile < 0)
              posizionepapabile = 0;
            controllaposizione ();           // verifica la validita della posizione papabile
            break;
          case 1:
            posizionepapabile = poscellula - 36;
            controllaposizione ();           // verifica la validita della posizione papabile
            break;
          case 2:
            posizionepapabile = poscellula - 35;
            controllaposizione ();           // verifica la validita della posizione papabile
            break;
          case 3:
            posizionepapabile = poscellula + 1;
            controllaposizione ();           // verifica la validita della posizione papabile
            break;
          case 4:
            posizionepapabile = poscellula + 37;
            if (posizionepapabile > 541)
              posizionepapabile = 541;
            controllaposizione ();           // verifica la validita della posizione papabile
            break;
          case 5:
            posizionepapabile = poscellula + 36;
            controllaposizione ();           // verifica la validita della posizione papabile
            break;
          case 6:
            posizionepapabile = poscellula + 35;
            controllaposizione ();           // verifica la validita della posizione papabile
            break;
          case 7:
            posizionepapabile = poscellula - 1;
            controllaposizione ();           // verifica la validita della posizione papabile
            break;
          default:
            posizionenonusabile = 1;
```

```
        }
      }
      schemacorrente [posizionepapabile] = 1;// occupa la posizione trovata libera
      poscellula = posizionepapabile;
    }
    schemacorrente [71]  = 0;                // libera l'angolo superiore destro
    schemacorrente [37]  = 0;                // libera l'angolo superiore sinistro
    schemacorrente [469] = 0;                // libera l'angolo inferiore sinistro
    schemacorrente [503] = 0;                // libera l'angolo inferiore destro
  }
  esponischemacorrente ();
}
//
// ***********************  routine di creazione della nuova generazione ***************
// ***********************************  new generation routine ************************
//
void prossimagenerazione ()
{
  tempogenerazione = millis();
  potenziometro = analogRead (1);
  tempo = map (potenziometro, 0, 1023, 0, 800);
  if ((tempogenerazione - iniziogenerazione) > tempo)  // se e' trascorso il tempo di generazione
  {
    //******************* prepara la matrice vuota per la nuova generazione *****************
    for (indice = 0; indice < 541; indice++)
      schemafuturo [indice] = 0;
    for (indice = 0; indice < 37; indice++)
      schemafuturo [indice] = 2;             // delimita il bordo superiore dello schema
    for (indice = 504; indice < 541; indice++)
      schemafuturo [indice] = 2;             // delimita il bordo inferiore dello schema
    for (indice = 0; indice < 15; indice++)
      schemafuturo [indice * 36] = 2;        // delimita il bordo laterale dello schema
    numgen++;                                // incrementa di 1 il numero delle generazioni
    //
    //************* analizza la generazione corrente e crea la nuova generazione
    // ******* check the current generation and prepare the next  *************
    //
    for (indice = 36; indice < 504; indice++)
    {
      contaadiacenti ();          // lancia la routine di conteggio del numero di cellule adiacenti
      //
      // ********* verifica se si tratta di una cellula che deve vivere o morire ******************
      //  ******************** check il cell must die or alive *******************************
      //
      if (schemacorrente[indice] == 1)
      {
        if ((celluleadiacenti >= 2) && (celluleadiacenti <= 3 ))  // se 2 o 3 cellule vive adiacenti
          schemafuturo [indice] = 1;  // la cellula vive nella prossima generazione
      }
      if (schemacorrente[indice] == 0)
      {
        if (celluleadiacenti == 3)   // se ci sono tre cellule vive adiacenti
          schemafuturo [indice] = 1; // nasce una nuova cellula
      }
    }
    // trasferisce la nuova generazione nella generazione corrente
    for (indice = 0; indice < 541; indice++)
      schemacorrente [indice] = schemafuturo [indice];
    esponischemacorrente ();   // visualizza la generazione corrente
  }
}
//
//   ********** predispone lo schema vuoto per accogliere la prima generazione
//   ********************* prepare array for the first generation *************
//
void schemainiziale ()
{
  for (indice = 0; indice < 541; indice++)
    schemacorrente [indice] = 0;
  for (indice = 0; indice < 37; indice++)
    schemacorrente [indice] = 2;             // delimita il bordo superiore dello schema
  for (indice = 504; indice < 541; indice++)
    schemacorrente [indice] = 2;             // delimita il bordo inferiore dello schema
  for (indice = 0; indice < 15; indice++)
    schemacorrente [indice * 36] = 2;        // delimita il bordo laterale dello schema
}
//
//
```

# Arduino: Conway – il gioco della vita – the game of life

```
void setup()
{
  Serial.begin  (9600);
  pinMode (pulsante, INPUT);
  TV.begin(PAL, 150, 84);        // definisce il tipo di video e la risoluzione (150 pixel
// orizzontali e 84 verticali
  TVselectfont6x8 ();
  TV.set_cursor (0, 20);         // posiziona verticalmente il cursore a circa' meta' schermo
  indicesegnalazione = 0;        // espone "il Gioco della vita,"
  segnala ();
  indicesegnalazione = 1;
  segnala ();                    // espone "basato sulle regole di Conway"
  TVprintln ();
  indicesegnalazione = 2;
  segnala ();                    // espone: "Conway's game of life"
  TVdelay3000 ();                // pausa di 3 secondi
  TVdelay3000 ();
  TVclearscreen ();
  TVselectfont4x6 ();
  TVprintln ();
  indicesegnalazione = 3;
  segnala ();                    // espone: "REGOLE ";
  TVprintln ();
  indicesegnalazione = 4;
  segnala ();                    // espone le regole di conway
  indicesegnalazione = 5;
  segnala ();                    // espone le regole di conway
  TVprintln ();
  indicesegnalazione = 6;
  segnala ();                    // espone le regole di conway
  indicesegnalazione = 7;
  segnala ();                    // espone le regole di conway
  TVprintln();
  indicesegnalazione = 8;
  segnala ();                    // espone le regole di conway
  indicesegnalazione = 9;
  segnala ();                    // espone le regole di conway
  TVprintln ();
  indicesegnalazione = 10;
  segnala ();                    // espone "premi il pulsante.."
  while (statopulsante == 0)
    statopulsante = digitalRead (pulsante);
  TVclearscreen ();
  seme = millis ();              // utilizza il tempo da inizio attivita' come seme per la
//  generazione di numeri a caso
  randomSeed (seme);             // inzializza il generatore di numeri a caso
}
//
//
void loop()
{
  statopulsante = digitalRead (pulsante);
  if (statopulsante == 1)
  {
    if (semaforopulsante == 0)
    {
      semaforopulsante = 1;
      tempoinizio = millis ();
    }
  }
  if ((statopulsante == 0) && (semaforopulsante == 1))    // se e' stato premuto il pulsante
  {
    semaforopulsante = 0;                       // azzera il semaforo per rendere il pulsante
// fruibile alla prossima pressione
    tempocorrente = millis ();
    if ((tempocorrente - tempoinizio) > 500)    // se il pulsante e' stato premuto per un lungo
//  periodo
    {
      semaforoseqprog = 1;                      // pone ad 1 il semaforo della sequenza programmata
      TVclearscreen ();
      TV.set_cursor (10, 30);
      TVselectfont6x8 ();
      TV.println ("sequenze programmate");
      TVdelay3000 ();
      numgentot = 0;                            // azzera il numero di generazioni da eseguire, per
// forzare l'avvio della sequenza programmata
    }
    else
```

```
    {
      numgen = 0;                              // azzera il contatore di generazioni
      numgentot = 999999;                      // predispone il lancio di 999999 generazioni
      sequenzarandom ();                       // lancia la routine di preparazione della sequenza
// randomizzata
    }
  }
  if (numgen < numgentot)                      // se non sono state completate le generazioni
    prossimagenerazione ();
  else                                         // se il numero delle generazioni e' completo, avvia
// la sequenza programmata
  {
    switch (semaforoseqprog)
    {
      case 1:
        glider(); break;
      case 2:
        piccolaesplosione (); break;
      case 3:
        esplosione (); break;
      case 4:
        diecicelle (); break;
      default:
        riprendirandom (); break;
    }
  }
}
```