Shield DMX512 per Arduino

Prezzo: 15.98 €

Tasse: 3.52 €

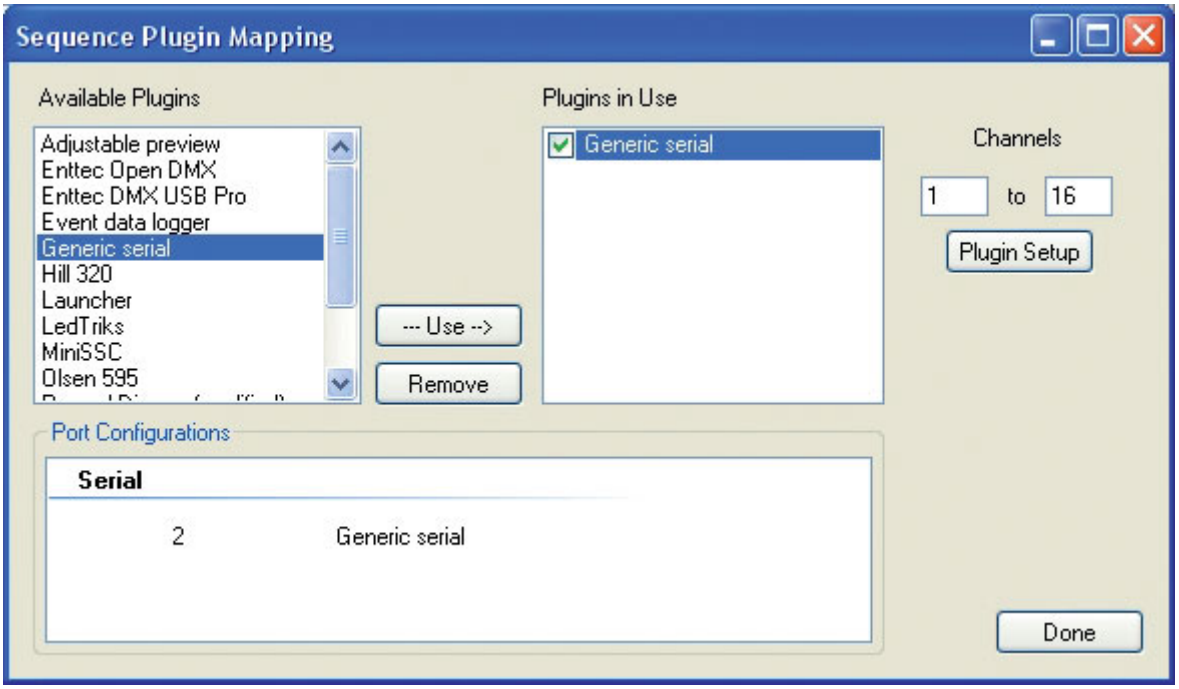
Prezzo totale (con tasse): 19.50 €

Shield dotata di interfaccia RS485, lettore per micro SD card e connettore XLR a 3 contatti. Abbinata ad Arduino e ad un software di libero utilizzo con cui possiamo costruire sequenze di attivazione (anche sincronizzate alla musica), permette di comandare i dispositivi ad interfaccia DMX512 da PC. Il lettore di micro SD card può essere gestito da Arduino allo scopo di realizzare molteplici funzioni correlate al controllo di dispositivi DMX512: ad esempio può memorizzare sequenze di attivazione di lampade e dispositivi per lo spettacolo, da riprodurre dietro comando impartito da computer o semplicemente dal pulsante P1 di cui lo shield è provvisto. La shield è predisposta per una connessione ad un display LCD seriale con il quale possiamo visualizzare informazioni riguardanti ad esempio la sequenza in fase di esecuzione, l'indirizzo assegnato al dispositivo DMX512 che si sta controllando, l'eventuale nome della sequenza memorizzata in micro SD card, ecc. **N.B.** la shield viene fornita con tutti i componenti SMD già montati, gli unici componenti da saldare sono i connettori strip, il connettore DMX e il pulsante.

La shield abbinata ad un PC in cui gira "Vixen" e ad una serie di tubi luminosi e lampade gestiti da un controller DMX512?

IL FIRMWARE

Il firmware permette ad Arduino di creare ed inviare stringhe di comando a standard DMX512 formate da un massimo di 512 byte, ognuno dei quali esprime 256 livelli di luminosità, oppure di movimento di un rotore sul quale vengono montati corpi illuminanti o altro, a seconda del dispositivo controllato. Ciascuna stringa inizia con un impulso header (vedi l'apposito riquadro) a cui seguono in sequenza tutti i bit; ogni decoder/controller monocanale è impostato in modo da interpretare solo il proprio byte, mentre i decoder con controller pluricanale interpretano un intervallo di byte. Perché ciò avvenga, ciascun decoder conta il numero di byte a partire dall'header. Ne deriva che per raggiungere il decoder/controller contraddistinto da un certo indirizzo, Arduino deve generare una stringa che contenga tanti byte quanti sono i canali fi no a quell'indirizzo: per esempio, volendo comandare la periferica che ha per address 128, il nostro sistema deve generare una stringa che contenga almeno i primi 128 byte (quelli dal 129 al 512 può anche non generarli, perché irrilevanti, contribuendo a mantenere una buona velocità di aggiornamento dei dispositivi DMX512). Alla generazione delle stringhe provvede una libreria chiamata DMXSimple (ma ne esistono altre...) scaricabile dal repository di Google http://code.google.com/p/tinkerit/wiki/DmxSimple; la versione attualmente online non è compatibile con l'IDE 1.0.1. Dal nostro sito potete invece scaricare una versione utilizzata per la realizzazione di questo progetto. Chiaramente dovete includere la libreria nel vostro sketch. All'interno della libreria vengono già forniti due sketch (vi accedete con il comando di menu Esempi): il primo permette di effettuare il fading (la dissolvenza...) di un canale DMX (vedi FadeUp, Listato 1) mentre il secondo consente di impostare il valore e il canale DMX tramite seriale (vedi SerialToDmx). Nell'impostare gli indirizzi delle periferiche da controllare ricordate quanto detto poco fa, quindi se dovete intervenire su un decoder/controller DMX512, possibilmente impostate per esso un indirizzo basso; ciò vi permetterà di ridurre il lavoro di Arduino e velocizzare la scansione, così da poter agevolmente eseguire giochi di luce molto rapidi. Oltre ai due sketch di esempio contenuti nella libreria, dal nostro sito potete scaricarne altri due chiamati FadeUpHSV, che esegue la dissolvenza tra i vari colori RGB, e DMX_LightSequencing, che permette di utilizzare il software Vixen, che gira su PC in ambiente Windows e vi consente di costruire sequenze di controllo luci definendo i canali da attivare ed eventualmente abbinando l'accensione e lo spegnimento di luci o l'attivazione di altri apparati DMX512, in determinati momenti della riproduzione di un brano musicale, che potete caricare dallo stesso programma. I due sketch d'esempio contenuti nella libreria e FadeUpHSV sono stati da noi modificati per essere compatibili con lampade RGB controllabili in DMX (per esempio la lampada Velleman VDPLP-64SB); dunque, in essi è sufficiente indicare l'indirizzo di partenza del DMX, poi Arduino fa il resto. Soffermiamoci ora sul quarto esempio di sketch (Listato 2), chiamato DMX_LightSequencing, perché è quello che ci permette di interagire con Arduino mediante un PC e di fare quindi le cose più interessanti; in questo sketch, oltre all'indirizzo DMX512 di partenza, va specificato anche il numero di canali da gestire, nel senso che bisogna dire ad Arduino qual è l'address del primo canale e quanti canali seguono. In questo modo Arduino potrà costruire la stringa necessaria, riempiendo con tanti byte a zero le posizioni corrispondenti agli indirizzi precedenti quello di partenza e terminando la stringa con il byte dell'indirizzo dell'ultimo canale. Ciò, come accennato, evita di sprecare risorse nella ricostruzione di quella parte di stringa che non serve. Lo sketch va utilizzato in abbinamento al programma Vixen, scaricabile gratuitamente dall'indirizzo http://www.vixenlights.com/; questo software è in grado di funzionare praticamente su qualsiasi computer (i requisiti sono un processore 486 o superiore, almeno 128 MB di RAM e 2 GB liberi su hard-disk, sistema operativo Windows 98 o successivo, purché vi sia installato Microsoft .NET Framework). Con questo software è possibile creare delle sequenze luminose (ma non solo) eventualmente legate alla musica; una delle applicazioni più suggestive è l'abbinamento alle illuminazioni natalizie, di cui vedete una piccola demo in questo filmato, all'indirizzo https://www.youtube.com/results?search_query=vixenlig hts&oq=vixenlights. Vixen ha diversi formati di output tra cui anche quello seriale (purtroppo è possibile selezionare solo le COM 1, 2, 3 o 4) che possiamo usare per una connessione USB da COM virtuale; selezionando la porta COM cui è collegato Arduino (Figura qui sotto "Sequence Plugin Mapping"), il software invia il valore che deve assumere ogni canale. Arduino interpreta questi dati e gestisce di conseguenza i propri I/O digitali. Normalmente, infatti, i progetti disponibili in "rete" prevedono



Listato 1 – LO SKETCH Fade Up

```
/* Welcome to DmxSimple. This library allows you to control DMX stage and
** architectural lighting and visual effects easily from Arduino. DmxSimple
** is compatible with the Tinker.it! DMX shield and all known DIY Arduino
** DMX control circuits.
**
** DmxSimple is available from: http://code.google.com/p/tinkerit/
** Help and support: http://groups.google.com/group/dmxsimple       */

/* To use DmxSimple, you will need the following line. Arduino will
** auto-insert it if you select Sketch > Import Library > DmxSimple. */

#include <DmxSimple.h>

void setup() {
  /* The most common pin for DMX output is pin 3, which DmxSimple
  ** uses by default. If you need to change that, do it here. */
  DmxSimple.usePin(3);

  /* DMX devices typically need to receive a complete set of channels
  ** even if you only need to adjust the first channel. You can
  ** easily change the number of channels sent here. If you don't
  ** do this, DmxSimple will set the maximum channel number to the
  ** highest channel you DmxSimple.write() to. */
  DmxSimple.maxChannel(4);
}

void loop() {
  int brightness;
  /* Simple loop to ramp up brightness */
  for (brightness = 0; brightness <= 255; brightness++) {

    /* Update DMX channel 1 to new brightness */
    DmxSimple.write(1, brightness);

    /* Small delay to slow down the ramping */
    delay(10);
  }
```

Nella nostra applicazione, invece, utilizziamo la libreria DMX, la quale tramite lo shield DMX permette di gestire 512 canali (tra l’altro anche analogici). In questo modo è possibile gestire svariate periferiche anche diverse dalle classiche luci, come ad esempio macchine del fumo e per la neve, strobo, teste rotanti ecc. Sul web si trovano tantissime sequenze già pronte; l’unica cosa da fare è impostare l’uscita della sequenza (ovvero la porta seriale) e programmare Arduino indicando il numero di canali utilizzati nella sequenza.

Listato 2 – DMX_LightSequencing

```
/*
The purpose of this code is to allow the Arduino to use the
generic serial output of vixen lights to control 5 channels of LEDs.
 */

#include <DmxSimple.h>

const int jrde =  2;
const int jdi  =  3;
const int jro  =  4;
const int lr   =  7;
const int ly   =  8;

#define DELAY    10

int i = 0;     // Loop counter
int incomingByte[25];   // array to store the 25 values from the serial port
int address=1;
int ch=16;
int k;

//setup the pins/ inputs & outputs
void setup()
{
  Serial.begin(9600);         // set up Serial at 9600 bps

  pinMode(jrde, OUTPUT);
  pinMode(jdi,  OUTPUT);
  pinMode(jro,  INPUT);
  pinMode(lr, OUTPUT);
  pinMode(ly, OUTPUT);

  digitalWrite(jrde, HIGH);

  /* The most common pin for DMX output is pin 3, which DmxSimple
  ** uses by default. If you need to change that, do it here. */
  DmxSimple.usePin(jdi);

  /* DMX devices typically need to receive a complete set of channels
  ** even if you only need to adjust the first channel. You can
  ** easily change the number of channels sent here. If you don't
  ** do this, DmxSimple will set the maximum channel number to the
  ** highest channel you DmxSimple.write() to. */
  DmxSimple.maxChannel(ch);

  for (k=0; k<ch+1; k++) {
    DmxSimple.write(k+address,0);
 }

}

void loop()
{  // 25 channels are coming in to the Arduino
   if (Serial.available() >= ch) {
    // read the oldest byte in the serial buffer:
      for (k=0; k<ch+1; k++) {
        incomingByte[k] = Serial.read();
      }
      //for (k=0; k<ch+1; k++) {
      //Serial.print ("ch "); Serial.print (k); Serial.print (" - value ");
Serial.println (incomingByte[k]);
      //}

      digitalWrite(lr, HIGH);

      for (k=0; k<ch+1; k++) { // for all three colours
        DmxSimple.write(k+address, incomingByte[k]);
      }
    }

  //delay(DELAY);
  digitalWrite(lr, LOW);
}
```

L’ultima versione della libreria è scaricabile direttamente da
https://github.com/PaulStoffregen/DmxSimple