# HP LED Driver Shield

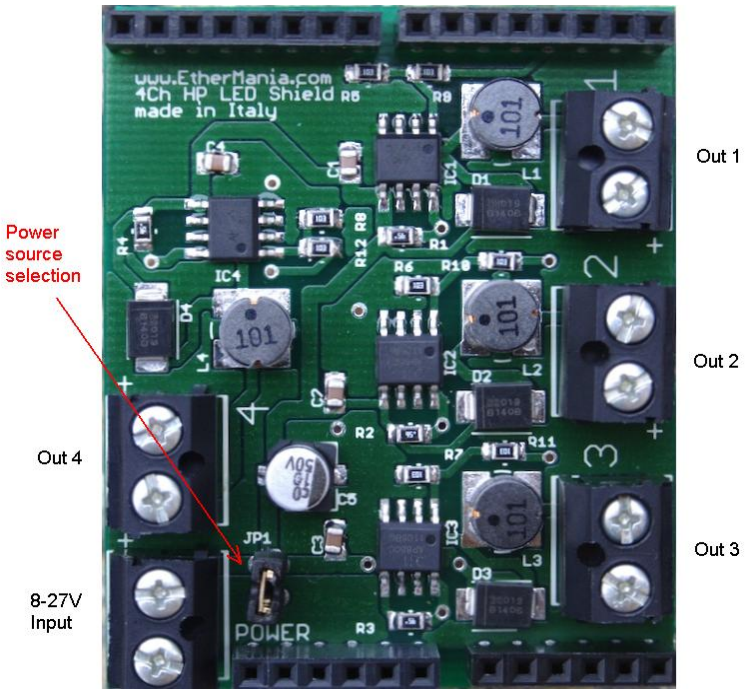
**Introduzione**

Lo HPLEDDriverShield e' uno shield per Arduino (UNO e Mega 2560) che permette di pilotare fino a 4 stringhe di LED High Power. Ogni canale e' dotato di un preciso generatore di corrente in grado di adattarsi a qualsiasi tipologia di carico operante con tensioni variabili da 8V a 27V, erogando 350mA.

Queste caratteristiche gli permettono di pilotare, ad esempio, quattro stringhe di LED, formate ognuna da sette LED da 1Watt, per un totale di 28 LED, come pure di lavorare correttamente con soli quattro LED da 1Watt senza alcuna modifica hardware. Eventuali canali non utilizzati possono essere liberamente lasciati scollegati.

Lo shield utilizza 4 pin PWM di cui Arduino e' dotato al fine di modulare la luminosita' dei LED collegati sui quattro canali (dimmering). L'alimentazione necessaria al funzionamento dei LED puo' essere prelevata dalla scheda Arduino o puo' essere fornita allo shield attraverso la morsettiera di cui e' provvista.

Ogni canale, inoltre, e' protetto contro i cortocircuiti.




**Installazione ed alimentazione dello shield**

L'installazione dello shield e' alquanto semplice. Basta inserirlo su un Arduino UNO o un arduino Mega e fornire alimentazione. La presenza di un jumper sullo shield permette di scegliere due modalita' di alimentazione. Nel caso in cui il jumper sia inserito a cavallo dei due terminali, la tensione presente al morsetto di alimentazione dello shield viene portata anche sull'Arduino tramite il piedino Vin (o viceversa, la tensione presente sul connettore DC di Arduino viene utilizzata per alimentare lo shield). Mantenere il jumper installato viene comodo per installazioni nelle quali la tensione di alimentazione richiesta e' inferiore ai 12 Volt, tensione massima accettabile da Arduino.

Nel caso in cui si renda necessario alimentare lo shield con tensioni superiori ai 12 Volt e' opportuno rimuovere il jumper dalla sua sede e fornire una alimentazione separata ad Arduino ed allo shield. In questo caso lo shield puo' essere alimentato tramite l'apposita morsettiera, mentre Arduino indifferentemente dallo USB o dalla presa DC.

> **NOTA:** L'alimentazione minima richiesta dallo shield e' pari ad 8Volt, di conseguenza, non e' possibile alimentare Arduino e lo shield contemporaneamente attraverso la presa USB che, normalmente, opera a 5Volt.

Lo shield e' dotato di 4 morsettiere a 2 poli per il collegamento di un massimo di 4 differenti stringhe di LED. Le stringhe di LED devono essere obbligatoriamente essere collegate a queste morsettiere rispettando la polarita'. Si fa presente che il “polo negativo” delle quattro stringhe non e' collegato alla massa e, quindi, non e' possibile condividere un solo conduttore tra le stringhe.

I canali non utilizzati dello shield possono essere lasciati parti.

**Quali LED scegliere**

La scelta dei LED ricade nella categoria dei LED operanti a 350mA. Essi sono del tipo High Power e, per un uso hobbistico, vengono tipicamente forniti gia' montati su un circuito stampato di alluminio che va necessariamente appoggiato su una opportuna aletta per dissipare il calore generato.

Lo shield e' in grado di pilotare un numero variabile di LED, da 1 a N, dove N e' determinato principalmente dalla caduta di tensione fornita dal particolare LED scelto e dalla tensione alla quale lo shield viene alimentato.

Un tipico LED da 1 Watt operante a 350mA, ad esempio, produce una caduta di tensione di circa 3.3V ai suoi capi. Ipotizzando di alimentare lo shield a 24V, si potra', per ogni canale, illuminare una serie di 7 LED per un totale complessivo di 28LED. Tale numero scende nel caso in cui si decida di alimentare lo shield ad una tensione differente. Ad esempio, alimentando lo shield a 12V si potranno illuminare un massimo di 3 LED da 1Watt per canale, portando il numero complessivo a 12.

LED con colori diversi possono essere “mischiati” all'interno della stessa serie. Il requisito fondamentale da rispettare e', comunque, che lavorino tutti con corrente di 350mA e che la somma totale della caduta di tensione della stringa sia inferiore a quella utilizzata per alimentare lo shield. Si ricorda che con il termine “stringa” si intende una connessione di tipo serie dei vari LED che la compongono ed, in questo caso, la caduta di tensione della stringa e' data dalla somma delle cadute di tensione dei singoli LED che la compongono.

Giusto per dare un esempio di LED commercialmente disponibile e perfettamente compatibile con questo shield, si puo' menzionare il CREE Xlamp MCE Color (RGBW) che integra 4 differenti High Power LED in un unico package.

**Utilizzo dello shield**

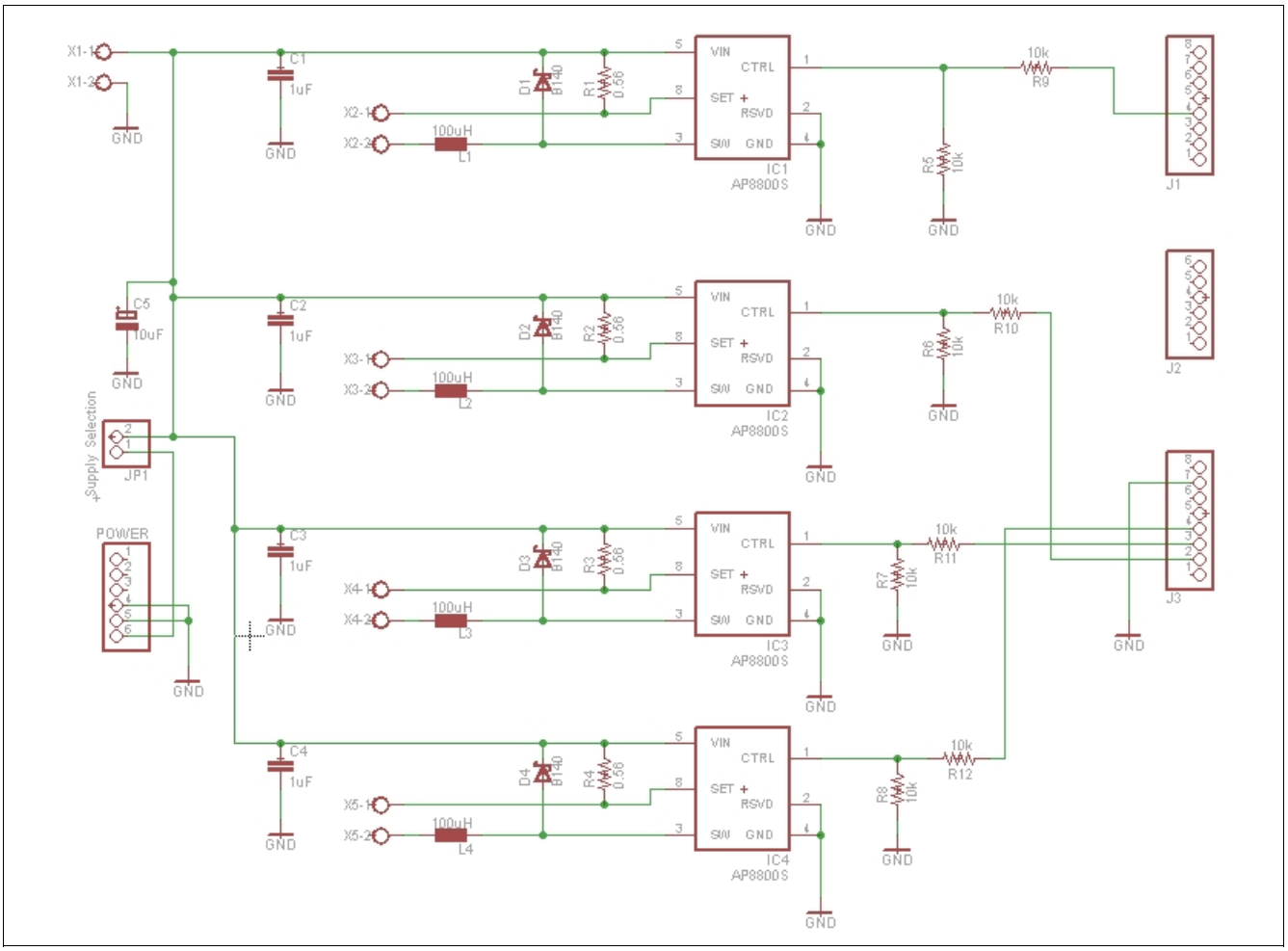
L'accensione dei LED collegati ad un canale avviene scrivendo un livello logico alto sul corrispondente piedino di Arduino. Applicando una serie di impulsi modulati (PWM) e' possibile variare la luminosita' di ogni singolo canale.

La tabella qui sotto riporta le associazioni tra i quattro canali ed i piedini di output digitale di Arduino. Volendo e' possibile utilizzare lo shield anche senza Arduino: in questo caso l'accensione dei singoli canali puo' essere “forzata” applicando una tensione di 5 volt ai rispettivi piedini dello shield.

| Piedino Out Arduino | Canale |
|---|---|
| 3 | 1 |
| 9 | 2 |
| 10 | 3 |
| 11 | 4 |

**Schema elettrico**

Si riporta, lo schema elettrico dello shield.



**Sketch d'esempio**

Lo shield utilizza 4 uscite PWM di Arduino. Per questo motivo il suo utilizzo e' di estrema semplicita'.

A titolo d'esempio si riporta uno sketch che permette di generare una sequenza di colori preimpostata. Il passaggio da un colore all'altro avviene in maniera graduale, ottenendo cosi' interessanti e morbidi effetti di luce.

Lo shetch completo puo' essere scaricato dal sito di EtherMania alla pagina del prodotto.

```
#define RED_PWMPIN    3
#define GREEN_PWMPIN  9
#define BLUE_PWMPIN  10
#define WHITE_PWMPIN 11

#define TRANSITION_STEPS 80

#define NUMSTEPS  16

unsigned char redSteps[NUMSTEPS] = { 0, 50, 255, 255, 100, 150, 200, 230, 0, 60, 20, 30, 30, 0, 150, 255 };
unsigned char greenSteps[NUMSTEPS] = { 100, 100, 0, 50, 200, 250, 255, 100, 255, 100, 150, 80, 0, 100, 0, 0 };
unsigned char blueSteps[NUMSTEPS] =  { 0, 255, 255, 0,  0, 0, 100, 150, 0, 100, 250, 255, 255, 180, 90, 0 };
unsigned char whiteSteps[NUMSTEPS] = { 0, 0, 50, 50, 100, 100, 150, 150, 200, 200, 150, 100, 50, 0, 0, 0 };

void setup()  {
}

void loop() {
  for (unsigned char step_i = 0; step_i < NUMSTEPS; step_i++) {

    morphToStep(step_i);
    delay(2000);
  }
}

void morphToStep(unsigned char newStep) {

  unsigned char prevStep = (newStep - 1) & 0x0F;

  unsigned char curRed = redSteps[prevStep];
  unsigned char curGreen = greenSteps[prevStep];
  unsigned char curBlue = blueSteps[prevStep];
  unsigned char curWhite = whiteSteps[prevStep];

  unsigned char nextRed = redSteps[newStep];
```

```
  unsigned char nextGreen = greenSteps[newStep];
  unsigned char nextBlue = blueSteps[newStep];
  unsigned char nextWhite = whiteSteps[newStep];

  while ((curRed != nextRed) || (curGreen != nextGreen) ||
         (curBlue != nextBlue) || (curWhite != nextWhite)) {

      curRed = updateBrightness(curRed, nextRed);
      curGreen = updateBrightness(curGreen, nextGreen);
      curBlue = updateBrightness(curBlue, nextBlue);
      curWhite = updateBrightness(curWhite, nextWhite);

      updatePWM(curRed, curGreen, curBlue, curWhite);
      delay(20);
  }
}

unsigned char updateBrightness(unsigned char current, unsigned char target) {
  if (current == target)
    return current;

  if (current > target)
    current = current - 1;
  else
    current = current + 1;

  return current;
}

void updatePWM(unsigned char red, unsigned char green, unsigned char blue, unsigned char white) {
  analogWrite(RED_PWMPIN, red);
  analogWrite(GREEN_PWMPIN, green);
  analogWrite(BLUE_PWMPIN, blue);
  analogWrite(WHITE_PWMPIN, white);
}
```