HP LED Driver Shield

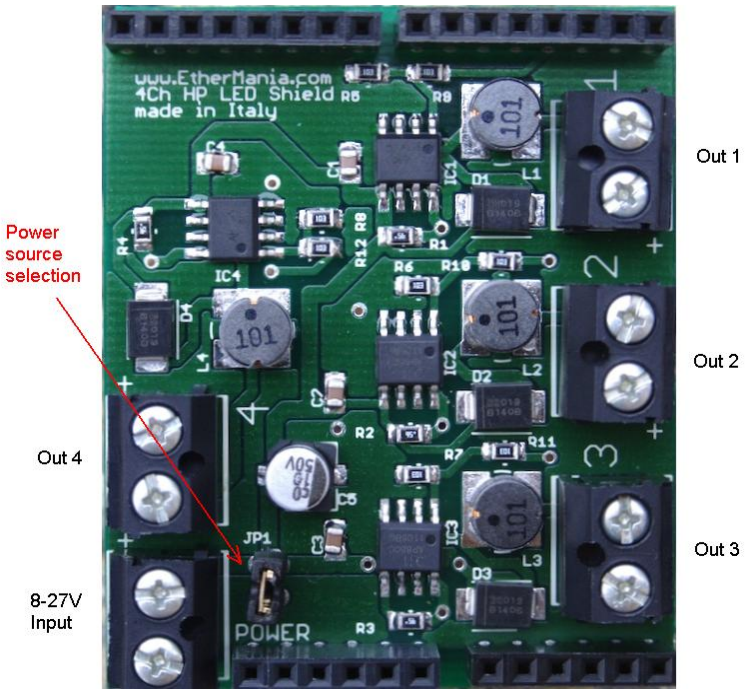
Introduction

HPLEDDriverShield is a shield for Arduino (UNO and Mega 2560) able to drive a maximum of 4 High Power LED strings. Each channel implements a precise 350mAmps constant current driver operating at supply voltages ranging from 8Volts to 27Volts DC.

The shield could drive, for example, a maximum of 4 LED strings made by 7x1Watt LEDs each channel, for a total of 28 LEDs. Not used channels could be left open.

The shield uses 4 PWMs generated by Arduino board to modulate the light intensity, independently on each channel (dimmering). The needed power supply could be derived by the Arduino board or directly feed to the shield through the specific connector.

Each channel is hardware short-circuit protected.



Installing and powering the shield

The shield is quite easy to install: just plug on top of an Arduino UNO or a Mega 2560 board and power it up. A jumper (JP1) set is required to select the power supply modality: with the jumper set, the power supply present on the 8-27V Input connector is feed to the shield and to the Arduino board (or, viceversa, the power supply present to the Arduino board is feed to the shield). This is useful for a single power supply setup but it's mandatory to not exceed the 12Volts (the maximum operating voltage for the Arduino board).

With the jumper left open, the shield should be externally powered through the 8-27V input power connector. In this situation a separated power supply is required for the Arduino board.

> **NOTE:** The minimum operating voltage for the shield is 8 Volts meaning that is not possible to power up Arduino and the shield through the USB connector that, normally, operates at 5Volts.

Four different LED strings could be connected to the 4 output sockets available on the shield with the proper polarity (a + sign is marked near the sockets). Please note that each string is not directly connected to the GND so is not possible to share a common conductor between 4 channels.

Eventually not used channels could be left open.

LED selection

This shield provides 350mAmps on each channel. This defines the High Power LED category compatibility. They could be found on the market as already soldered on an aluminum based PCB that should be mounted on a proper heat sink.

The shield is able to drive a variable number of High Power LEDs ranging from 1 to N where N is determined by the provided power supply voltage and the Forward Voltage associated to each LED.

A typical 1 Watt LED, when driven at 350mAmps, generates a Forward Voltage of 3.3Volts. With this in mind, and providing a 24V to the shield, the shield can drive a series of 7 LEDs for each channel, for a maximum of 27 LEDs on all channels. Lowering the power supply to 12Volts, the maximum LEDs for each channel will be equal to 3, for a maximum of 12 LEDs on all channels.

LEDs with different colors could be mixed on the same string. The only important constraint is related to the maximum Forward Voltage the string will generates when driven at 350mAmps. Please remember that the string Forward Voltage is the sum of Forward Voltages associated to each LED connected to the string (the string is a series connection).

The CREE Xlamp MCE Color (RGBW) is an example of compatible LED. It's combining 4 dies on a single package, providing Red, Green, Blue and White colors.

How to use it

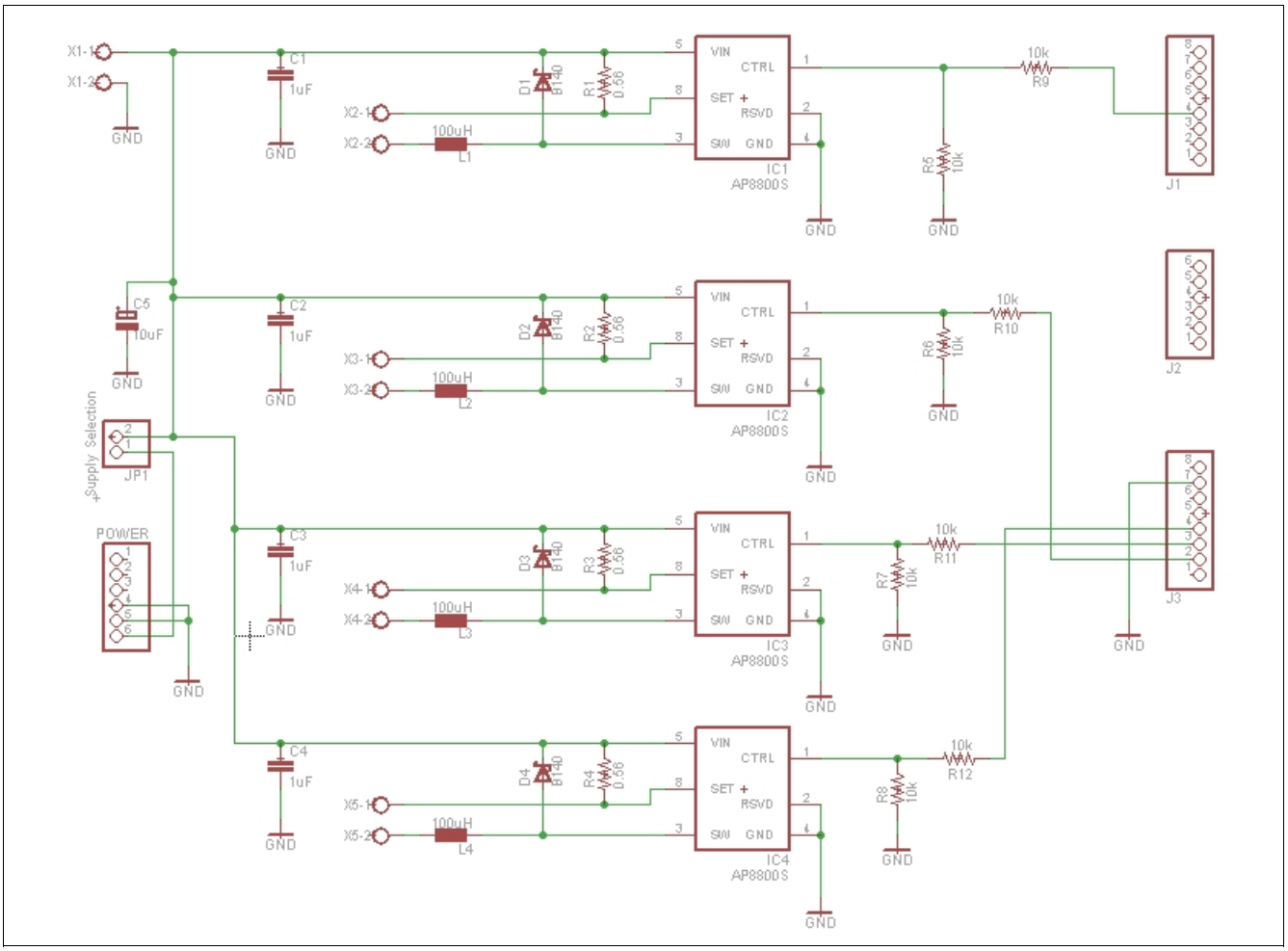
Each LED string is associated to an Arduino Pin. Writing a 1 logic to that pin triggers the constant current to power up the LEDs for that particular channel. If a PWM is applied (the max acceptable PWM frequency is 0.6MHz) the LED light will be reduced consequently.

The table below shows the associations between Arduino pins and the channels on the shield.

Please note that the shield could be used as a standalone 4 channel LED providing the required supply and applying 5Volts on all pins present in the connector headers.

PinOut Arduino	Channel
3	1
9	2
10	3
11	4

Schematics


X1-1
X1-2
GND
C1
1uF
X2-1
X2-2
100uH
L1
D1
B140
R1
0.56
IC1
AP8800S
VIN
CTRL
SET
RSVD
SW
GND
10k
R9
R5
J1
C5
10uF
C2
R2
D2
L2
X3-1
X3-2
R6
R10
J2
Supply Selection
JP1
POWER
C3
R3
D3
L3
X4-1
X4-2
IC3
R7
R11
J3
C4
R4
D4
L4
X5-1
X5-2
IC4
R8
R12

Sketch: an example

We reported here, as example, a simple sketch that generates a fixed color sequence. Each color fades smoothly on next color in the sequence.

The full sketch could be downloaded on the EtherMania website, at the product page.

```
#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);
}
```