16x32 RGB LED MATRIX

(Adafruit)

PREMESSA:

La documentazione utile è stata presa totalmente dal sito di Adafruit.

PANORAMICA:

Possiede 512 led RGB organizzati in una matrice rettangolare 16x32.

Sul retro è presente un PCB con 2 connettori IDC uno per input ed uno per output (che possono essere collegati insieme) e 12 latch a 16 bit che consentono di pilotare il display con una scala di ingrandimento 1:8.

Questi pannelli richiedono 12 o 13 pin digitali (6 bit per dati, 6 o 7 bit per controllo) e una solida alimentazione a 5V, almeno un paio di Ampere per pannello.

Si suggerisce un alimentatore da 2A (o più) e 5V and un terminale jack DC.

Teniamo in mente che questi display sono progettati per essere gestiti da FPGA o altri controllori ad alta velocità; non possiedono alcun tipo di controllo PWM (Pulse Width Modulation).

Dovresti ridisegnare lo schermo più e più volte su PWM "manualmente".

Su un Arduino Uno a 16 MHz siamo riusciti ad ottenere colori a 12 bit ($2^{12} = 4096$ colori) ma questi display splenderebbero davvero se guidati da FPGA, CPLD, Propeller, XMOS o altri controllori ad alta velocità.

Su un Arduino Uno, avrai bisogno di 12 pin digitali e circa 800 byte di RAM per poter ottenere tutti 4096 colori.

La libreria supporta un numero limitato di schede: Arduino Uno, Mega, Zero, Adafruit Metro M0 and Metro M4. Altre schede (come Arduino Leonardo) NON SONO SUPPORTATE.

ALIMENTAZIONE:

Sebbene siano efficienti risorse di luce, metterne molte in uno stesso punto comporta valori molto alti di corrente.
Una singola matrice RGB 16x32 con tutti I led impostati su bianco potrebbe richiedere fino a 4A di corrente.
Tuttavia per disegnare simboli e animazioni, questi pannelli useranno fino ad un massimo di 2A sufficienti per matrici 16x32.
Non vi sono problemi se si decidesse di usare un alimentatore da più Ampere basta che la tensione rimanga a 5V.
Su questi pannelli gli ingressi dell'alimentazione e dei dati sono separati.
Avendo scelto la tipologia con connettori stile Molex è sufficiente collegare l'omonimo terminale del cavo di alimentazione nell'apposito connettore sul retro della matrice, osservandone la polarità mentre l'altra estremità composta da 2 cavi con fissaggio a vite andrà collegata all'apposito connettore sulla SHIELD, anche qui occorre osservare la polarità prima di effettuare qualsiasi collegamento.

COLLEGAMENTO:

Questi pannelli sono normalmente progettati per la concatenazione (che collega end-to-end a formare display più grandi) l'output di un pannello si collega all'ingresso del successivo, e così via.

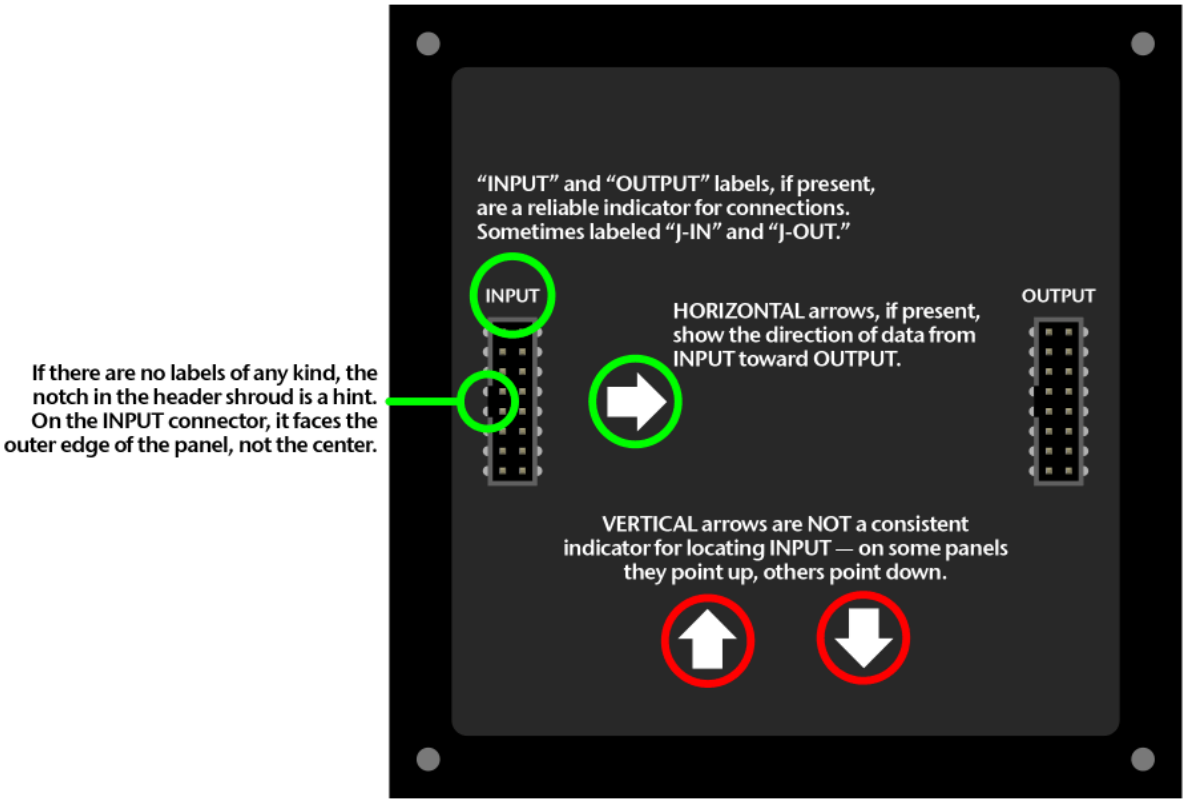
Con la RAM limitata in un Arduino, la è raramente pratico. Tuttavia, **è necessario distinguere le connessioni di ingresso e uscita** sul pannello ... non risponderà se siamo collegati alla presa sbagliata.

La libreria per Arduino non supporta la concatenazione!

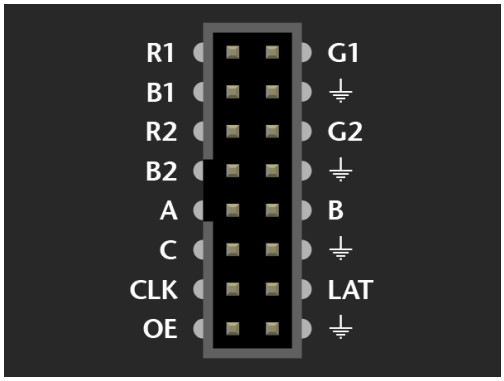
Su alcuni pannelli le prese sono etichettate INPUT e OUTPUT (a volte IN e OUT o simili), quindi è facile capire quale sia la presa di ingresso.

Se INPUT e OUTPUT non sono etichettati, cercare una o più frecce **orizzontali** (ignorare le frecce verticali) che mostrano la direzione in cui i dati si spostano da INPUT a OUTPUT.

Se tali etichette non sono presenti, un'ultima opzione è quella di esaminare la copertura di plastica attorno ai pin del connettore. La tacca sul connettore INPUT sarà rivolta verso il bordo esterno del pannello.



La disposizione dei pin sul connettore INPUT varia in base alle dimensioni della matrice e al luogo in cui è stato prodotto in particolare i connettori per matrici di dimensione 16x32 avranno la seguente piedinatura:



Un pannello **32x16** utilizza questa disposizione dei pin. Le etichette potrebbero essere leggermente diverse, oppure i pin potrebbero non essere etichettati affatto; in entrambi i casi utilizzare questa immagine come riferimento.

Questi pannelli sono normalmente gestiti da processori o FPGA molto veloci, non da un Arduino a 16 MHz. Per ottenere prestazioni ragionevoli in questo ambiente limitato, il nostro software è ottimizzato collegando segnali specifici a specifici pin Arduino**.** Qualche linea di controllo può essere riconfigurata, ma altre sono molto specifiche, non è possibile riconfigurare tutto.

COLLEGAMENTO TRAMITE RGB MATRIX SHIELD:

Questo è il metodo migliore per far interagire queste matrici con una scheda tipo Arduino, poiché è veloce e senza problemi.
L'Adafruit RGB Matrix Shield funziona con Arduino Uno e compatibili schede ATmega328-based.
La shield arriva spopolata bisogna effettuare un po' di saldatura per poterla utilizzare.
I pin sono installati dalla parte inferiore e saldati nella parte superiore.
Tre componenti:

- un pulsante,
- un terminale di alimentazione
- un connettore Molex a 16 pin

Sono invece inseriti dall'alto e saldati sotto.

Lo slot a 16 pin (8x2) deve essere installato con l'orientamento corretto, la tacca di polarità è indicata sulla shield.
Mettendolo al contrario, la matrice non funzionerà.

L'alimentazione della matrice LED può essere collegata ai terminali a vite sulla shield

- filo rosso a + 5Vout
- filo nero a GND

L'intero circuito viene quindi alimentato dalla presa DC di Arduino o da un cavo USB a 5 Volt sicuri e regolati.

CODICE DI ESEMPIO:

È necessario scaricare e installare *due* librerie:

- **libreria RGB Matrix Panel** (contiene il codice di basso livello specifico per questo dispositivo).
- **libreria GFX di Adafruit** (che gestisce le operazioni grafiche comuni a molti display che portiamo).

Aprire l'IDE e caricare il codice di esempio:

File → Esempi → RGBmatrixPanel → testcolors_16x32

Questo è un modello di prova che mostra 512 colori (su 4096) sui 512 pixel.

La riga di codice più utile è:

```
matrix.drawPixel(x, y, matrix.Color333(r, g, b));
```

Disegna un solo pixel alla volta.

I parametri **x(colonna)** ed **y(riga)** sono le coordinate che si riferiscono ai singoli pixel del display.

L'origine (0,0) è nell'angolo in alto a sinistra, il punto da essa più lontano **(31, 15)** è in basso a destra.

Per creare un colore, utilizzare la funzione **Color333** che prende tre numeri interi a 3 bit e li combina in un unico numero intero.

Una funzione simile, **Color444** , accetta tre numeri interi a 4 bit per un massimo di 4096 colori.

LIBRERIA:

1) **Disegnare un singolo pixel:**

```
matrix.drawPixel(0, 0, matrix.Color333(7, 7, 7));
```

Accende di bianco il led situato nell'anglo in alto a sinistra (0,0).

2) **Disegnare un rettangolo pieno:**

```
matrix.fillRect(0, 0, 32, 16, matrix.Color333(0, 7, 0));
```

Disegna un rettangolo verde pieno di base 32 e altezza 16 a partire dal punto in alto a sinistra (0,0).

3) **Disegnare il perimetro di un rettangolo:**

```
matrix.drawRect(0, 0, 32, 16, matrix.Color333(7, 7, 0));
```

Disegna la forma di un rettangolo di colore giallo di base 32 e altezza 16 a partire dal punto in alto a sinistra (0,0).

4) **Disegnare una linea:**

```
matrix.drawLine(0, 0, 31, 15, matrix.Color333(7, 0, 0));
```

Disegna una linea rossa a partire dal punto in alto a sinistra fino al punto in basso a destra, disegna quindi la diagonale discendente.

```
matrix.drawLine(0, 0, 31, 15, matrix.Color333(7, 0, 0));
matrix.drawLine(0, 15, 31, 0, matrix.Color333(7, 0, 0));
```

Disegna una X rossa composta dalle due diagonali.

5) **Disegnare una circonferenza:**

```
matrix.drawCircle(7, 7, 7, matrix.Color333(0, 0, 7));
```

Disegna una circonferenza blu con centro nel punto (7,7) e di raggio 7.

6) **Disegnare un cerchio:**

```
matrix.fillCircle(23, 7, 7, matrix.Color333(7, 0, 7));
```

Disegna un cerchio magenta nel punto (23,7) di raggio 7.

7) **Riempire lo schermo di un unico colore:**

```
matrix.fillScreen(matrix.Color333(0, 0, 0));
```

Tutti i led si accendono di nero, sono tutti spenti.

8) **Disegnare del testo:**

```
matrix.setCursor(x, y);
```

Posiziona il cursore nel punto (x,y).

```
matrix.setTextSize(dimensione); // size 1 == 8 pixels high
```

Setta la dimensione del font in scala di ingrandimento 1:8.

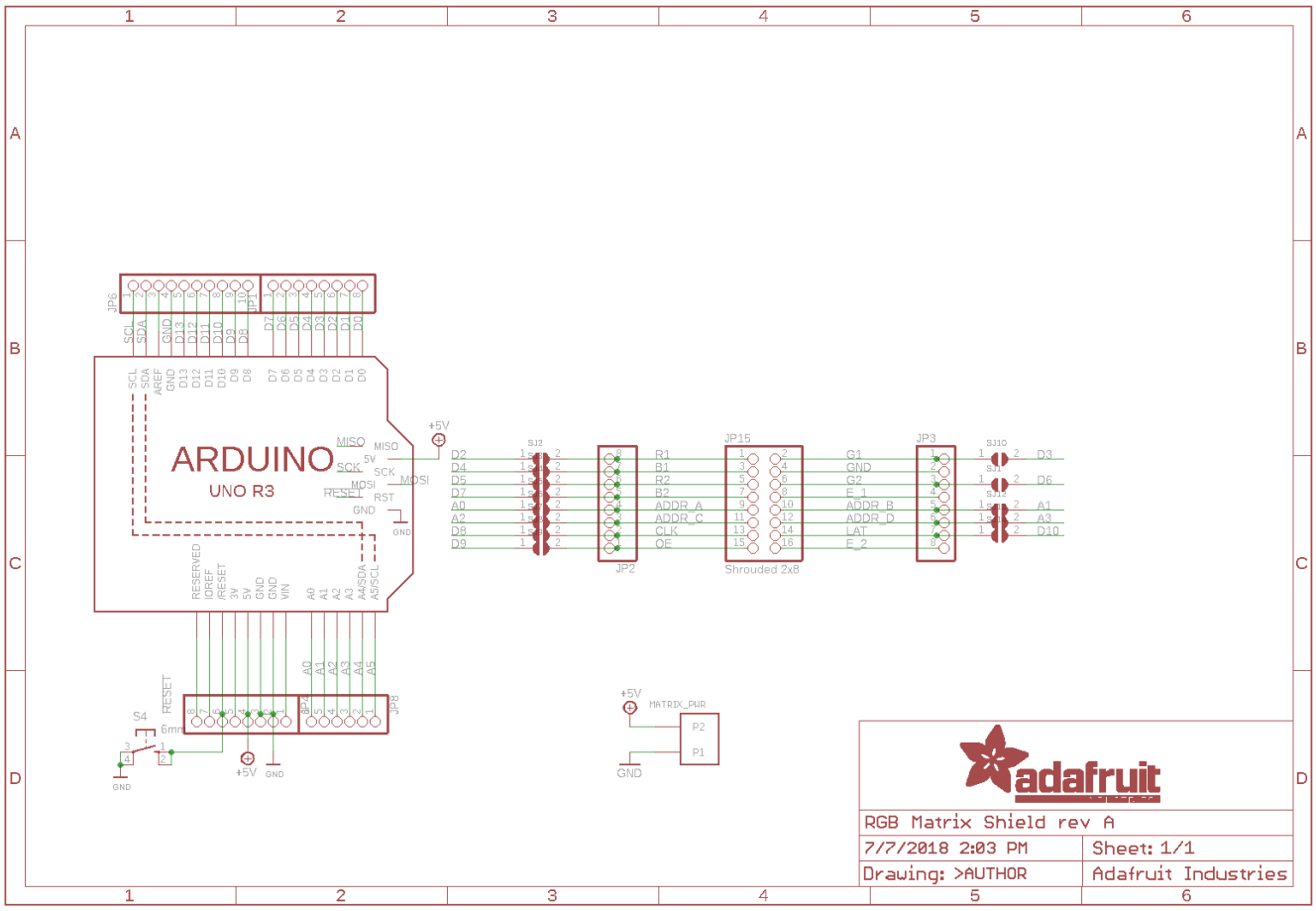
```
matrix.setTextColor(matrix.Color333(r,g,b));
```

Setta il colore del testo.

```
matrix.print(carattere);
```

Stampa un carattere sullo schermo.

SCHEMATICA:



PCB: