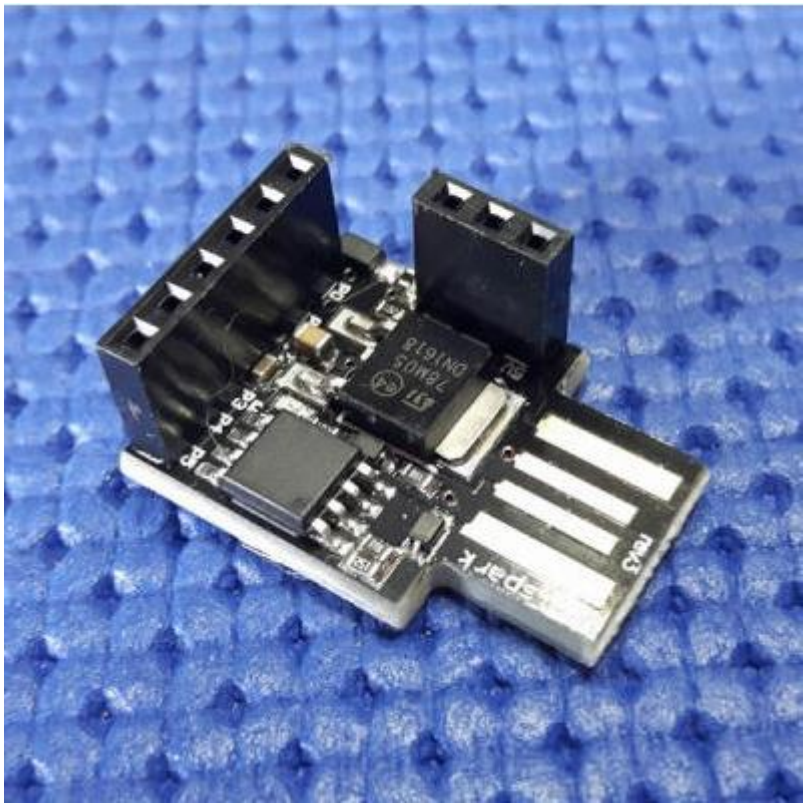
Guida per l’utilizzo delle librerie

Questa guida serve a spiegare i passi della programmazione con Arduino.

Driver Digispark

Per questo progetto viene utilizzata la scheda Digispark:



I due connettori si devono saldare sulla scheda perché all’acquisto ciò non è presente per il semplice fatto che vengono scelti in base alle proprie esigenze.

I pin della scheda sono i seguenti:

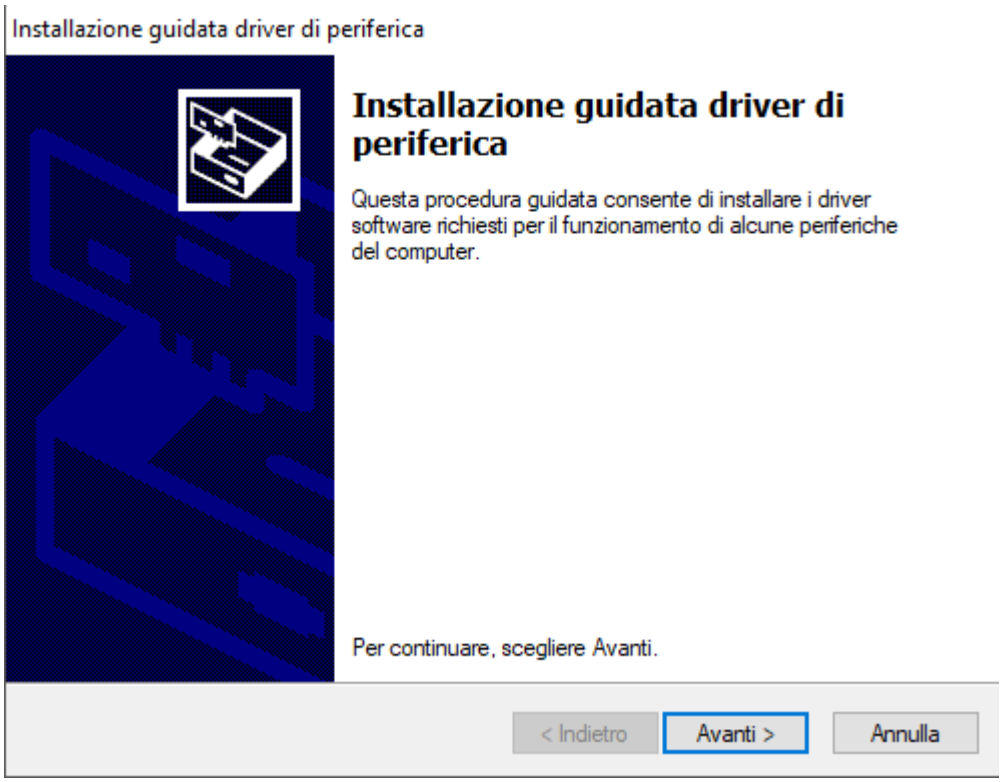
Pin 0 → I2C SDA, PWM
Pin 1 → PWM
Pin 2 → I2C SCK(PWM), Analogico(1)
Pin 3 → Analogic(3)
Pin 4 → PWM, Analogico(2)
Pin 5 → Analogico(0)

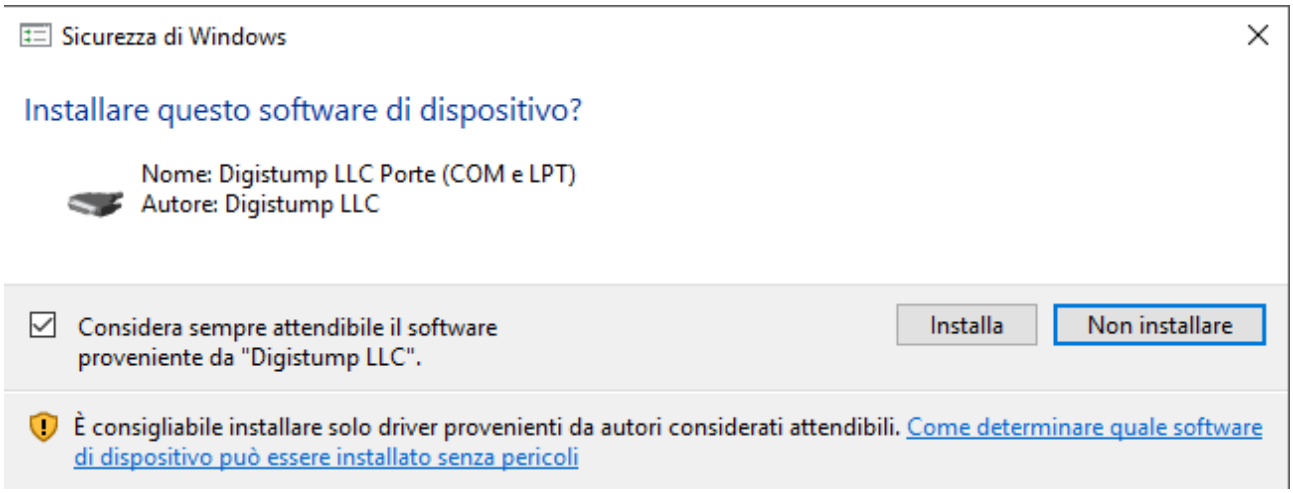
Per utilizzare questa scheda bisogna scaricare i drivers e ciò viene fatto da internet.

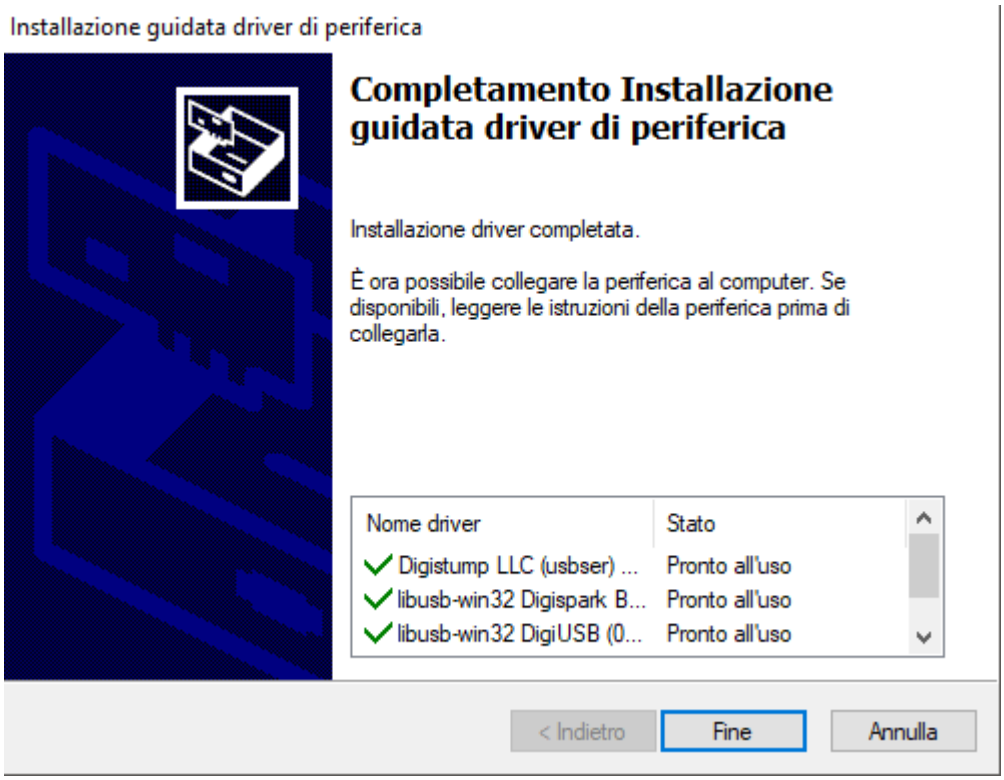
È necessario scaricare e installare manualmente i driver per la scheda Digispark. Bisgona scaricare, decomprimere ed eseguire "Install Drivers" (sui sistemi a 32 bit) o "DPInst64" (sui sistemi a 64 bit). Il link

dal quale scaricare i Drivers è:
(https://github.com/digistump/DigistumpArduino/releases/download/1.6.7/Digistump.Drivers.zip).

Una volta eseguito il programma appare la finestra dell’installazione guidata, seguire i passi mostrati di seguito:

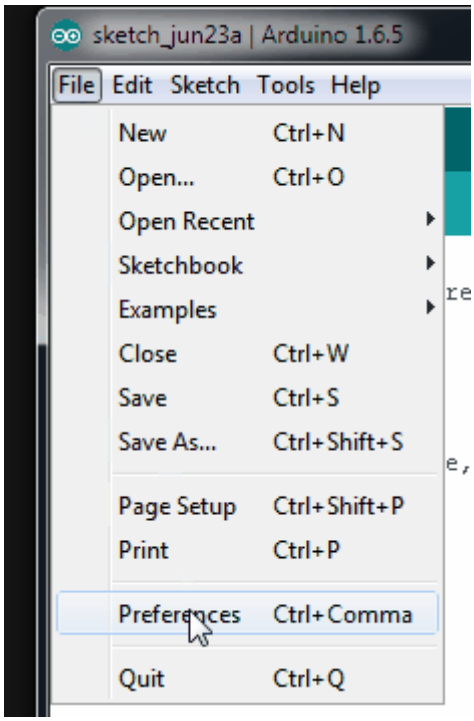


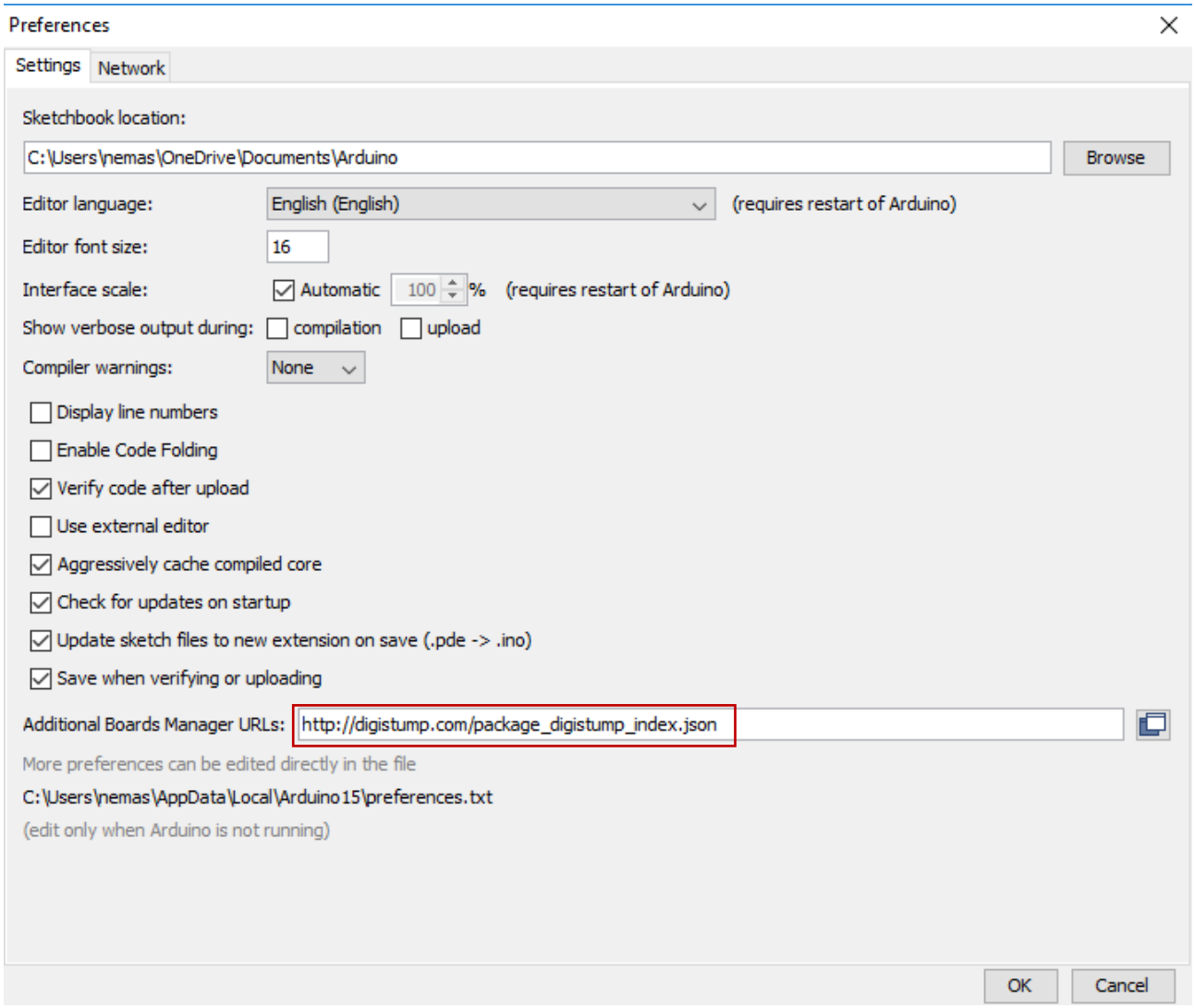




Quando la scheda verrà inserita in una porta USB senza che ciò venga richiesto dall’IDE è possibile che la scheda non venga riconosciuta ma è normale.

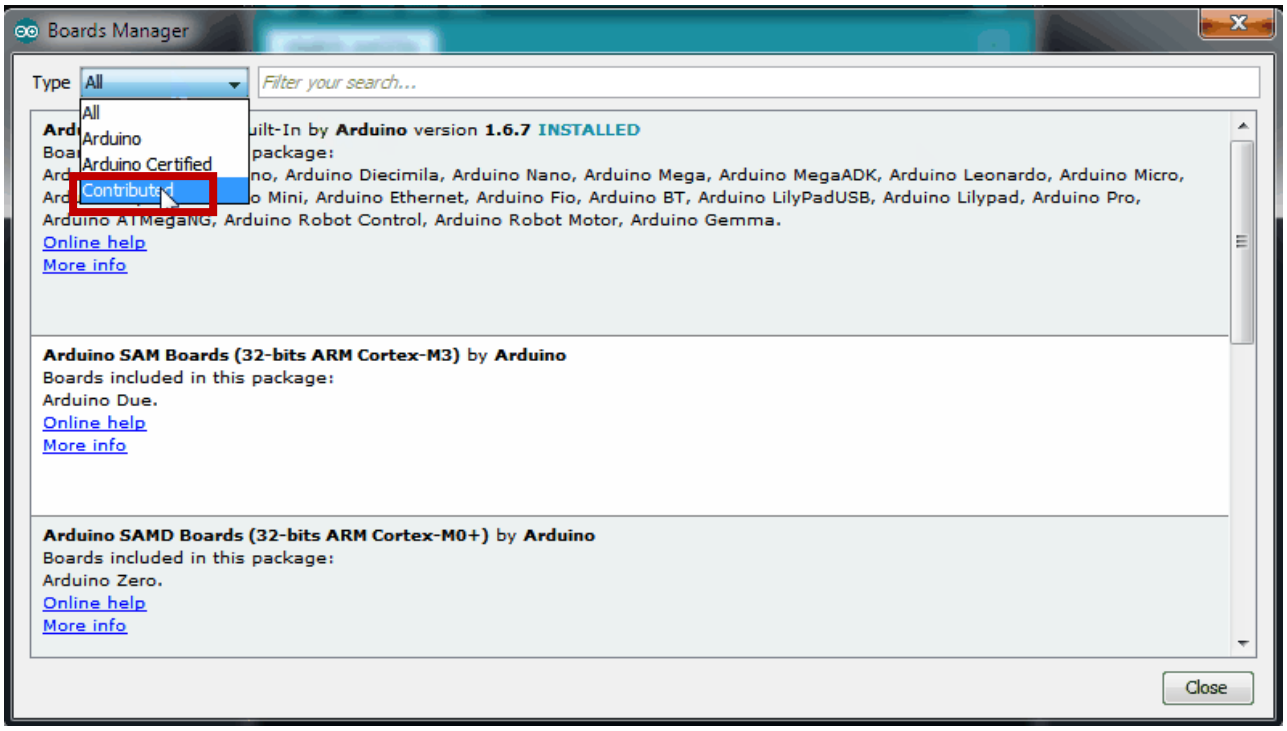
Una volta finita l’installazione, aprire il programma di Arduino e nel menu cliccare sulla voce “File” poiché scegliere “Preferences” oppure “Impostazioni”.



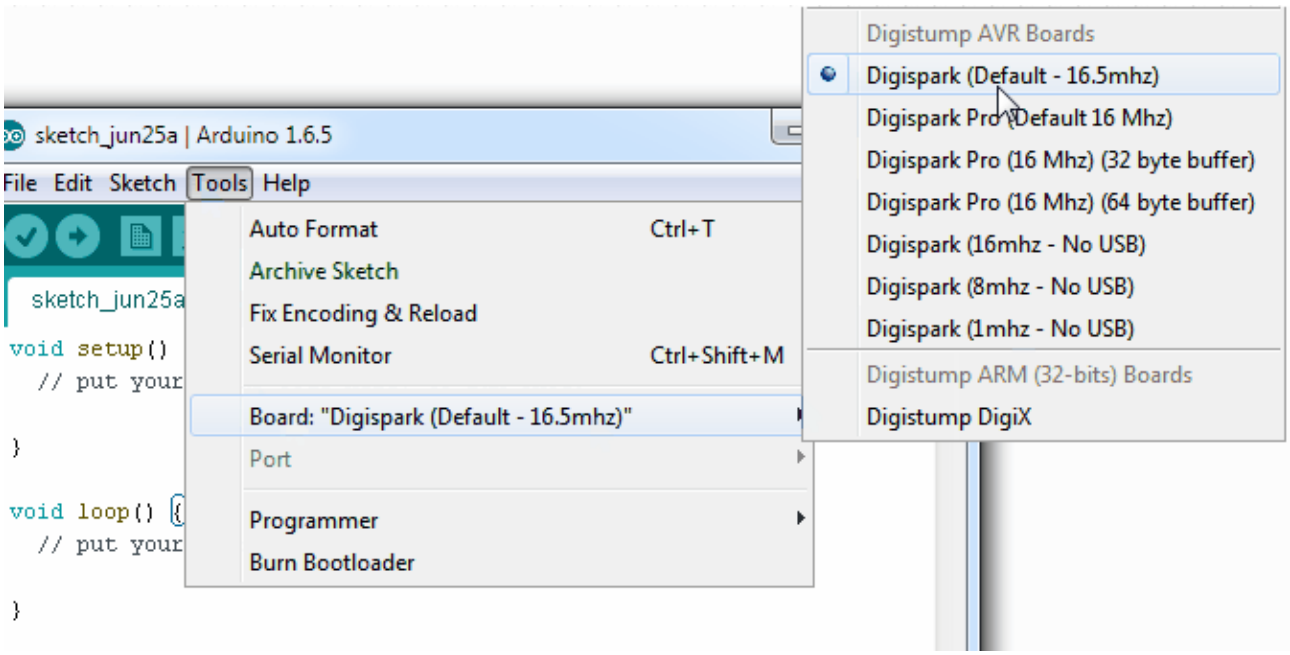


Nella casella con l’indicazione “URL ….” Inserire il seguente
http://digistump.com/package_digistump_index.json e cliccare su OK.

Quando questo viene terminato, nel menu scegliere “Strumenti” e poi in “Gestore schede”, quindi scegliere il tipo, il quale deve essere:



Una volta finito bisogna scegliere questa scheda per poterla utilizzare:

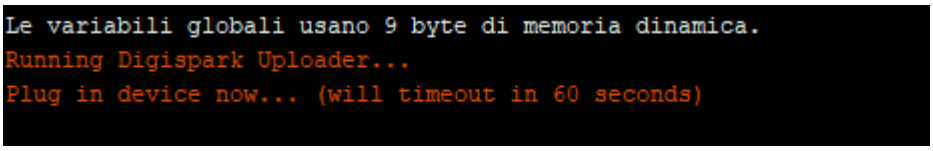


A questo punto l’installazione è completa.

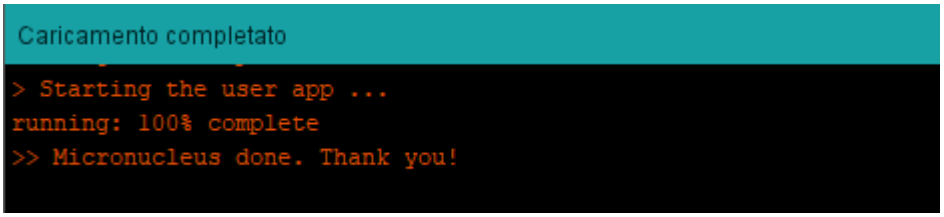
Uso della scheda Digispark con l’IDE

Questa scheda funziona in un modo diverso rispetto agli altri prodotti Arduino. La programmazione segue una procedura diversa.

- Come prima cosa bisogna assicurarsi che il sia selezionata la scheda Digispark.
- Utilizzare un codice (scriverlo o aprire uno già creato in precedenza)
- Caricare il programma/codice. Dopo la compilazione sarà richiesto di inserire il vostro Digispark, a questo punto collegarlo oppure scollegarlo e ricollegarlo.

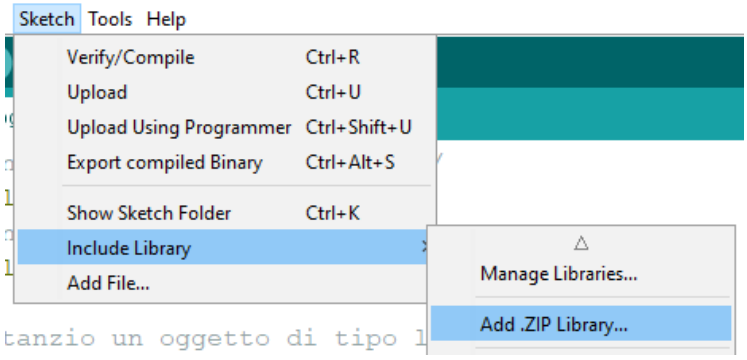


Una volta inserita la scheda e se tutto andrà a buon fine apparirà la scritta “Caricamento terminato” e il codice sarà eseguito sul Digispark.



Come integrare le librerie:

Per integrare una libreria bisogna:



In questo modo si aggiunge la libreria desiderata.

Bisogna assicurarsi che all’inizio del programma sia inclusa la libreria nel modo seguente:

#include <Button.h> (#include <”NomeLibreria”.h>)

Il prossimo passo è istanziare un oggetto della libreria che si sta utilizzato, per esempio:

Button button(0); *(Libreria “nomeogetto”(pin))*

Per utilizzare un metodo della libreria bisogna utilizzare l’oggetto istanziato precedentemente, quindi ***oggetto.metodo()*** oppure come nel esempio di seguito:

bool state = button.getButtonValue();

Questo pezzo di codice ritorna il valore del bottone.

Attuatori:

- Potenziometro
- Buzzer
- Bottone
- Led
- Led RGB

Potenziometro

Il ***potenziometro*** è un dispositivo elettrico equivalente ad un partitore di tensione resistivo variabile (cioè a due resistori collegati in serie, aventi la somma dei due valori di resistenza costante, ma di cui può variare il valore relativo), difatti una sua parte viene disposta in parallelo al carico utilizzatore.

Buzzer

Un ***buzzer*** è un dispositivo di segnalazione audio, che può essere meccanico, elettromeccanico, o piezoelettrico(abbreviato anche come piezo). I tipici utilizzi del buzzer includono dispositivi di allarme, timer, e PC speaker per i feedback sugli input dell'utente, come pressione dei tasti o click del mouse, nei vecchi personal computer.

Bottone

Il bottone è un dispositivo elettrico con una sola posizione di riposo (monostabile), una volta azionato una molla lo riporta alla posizione di partenza appena viene rilasciato.Una volta premuto esso aziona qualcosa (Accensione della luce, suonare il campanello).

Led

In elettronica il LED (sigla inglese di Light Emitting Diode[1]) o diodo a emissione di luce è un dispositivo optoelettronico che sfrutta la capacità di alcuni materiali semiconduttori di produrre fotoni attraverso un fenomeno di emissione spontanea.

Led RGB

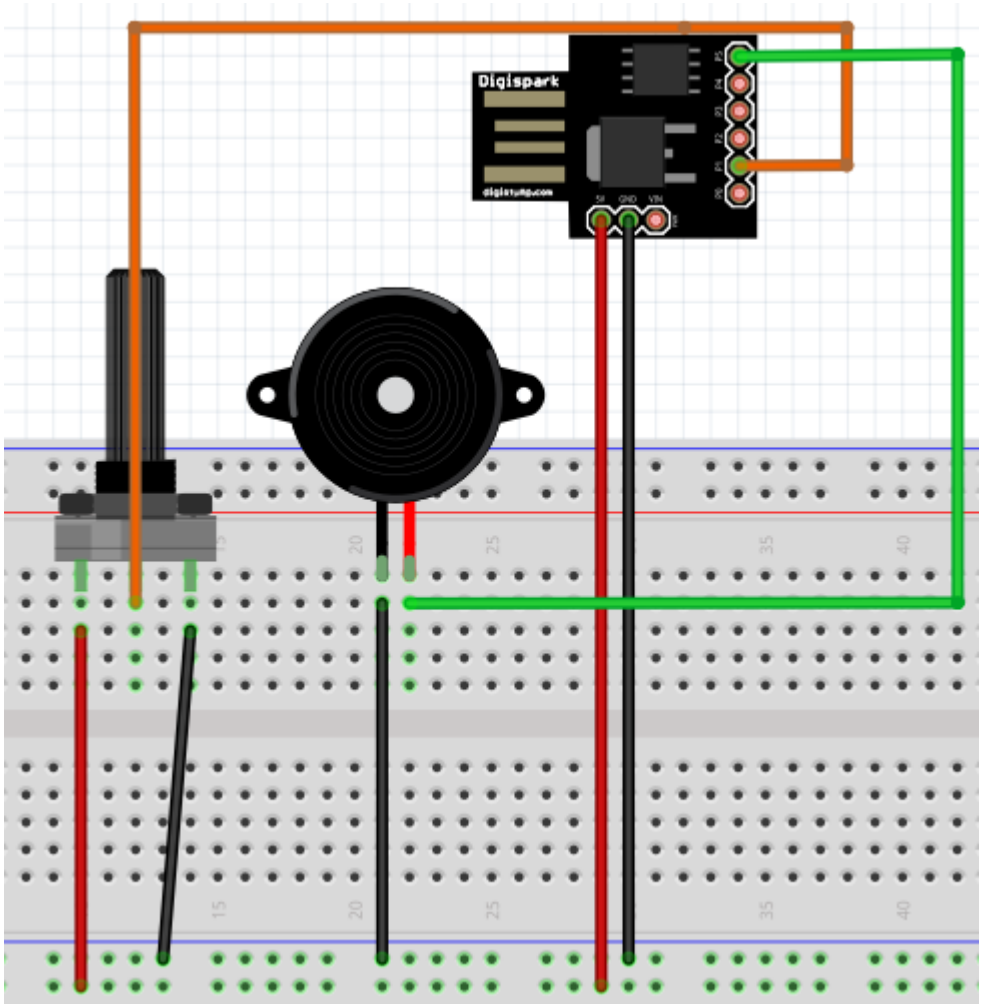
LED RGB indica i **LED** rossi, blu e verdi. I **LED RGB** combinano questi tre colori per produrre oltre 16 milioni di tonalità di luce. Non è possibile riprodurre tutti i colori. Alcuni di questi non rientrano nel triangolo formato dai LED RGB. Inoltre, i colori pigmentati come il marrone o il rosa sono difficili, se non impossibili, da ottenere.

Combinazioni di librerie

Buzzer e Potenziometro

Prima di iniziare bisogna assicurarsi che i driver del Digispark siano installati sul PC (Vedi primo capitolo “Driver Digispark”).

Prima di scrivere il codice bisogna seguire i passi del capito sull’integrazione delle librerie (Vedi secondo capitolo “Come integrare le librerie”).



Per questo circuito si necessità: 1 buzzer, 1 potenziometro, scheda Digispark.

Il potenziometro è collegato alla massa, al “5v” e al pin “P1” della scheda Digispark. Il Buzzer è collegato alla massa e al pin “P5” di Digispark.

Esempi:

BuzzerOnOff

```
/* Includere la libreria del Buzzer */
#include <Buzzer.h>
/* Includo la libreria del Buzzer */
#include <potentiometer.h>

/* Istanziare un oggetto di tipo Potentiometer */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo Buzzer */
Buzzer buzzer(2);

/*Metodo setup che viene avviato solo all'inizio*/
void setup() {
}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {

  /*Ritorna il valore del potenziometro*/
  int stato = pot.getValue();

  /*Se il valore del potenziometro è minore di 500 spegne
   *il Buzzer e se invece è maggiore lo accende
   */
  if(stato < 500){
    buzzer.setOffBuzzer();
  }else{
    buzzer.setOnBuzzer();
  }
}
```

Questo esempio accende e spegne il buzzer in base al valore del potenziometro.

BuzzerPotentiometerFrequence

```
/* Includere la libreria del Buzzer */
#include <Buzzer.h>
/* Includere la libreria del Buzzer */
#include <potentiometer.h>

/* Includo la libreria del Buzzer */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo Buzzer */
Buzzer buzzer(2);

/*Setta il range*/
int range= 16000;

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {
}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
  /* Prendo lo stato del potenziometro */
  int stato = pot.getValue();

  /*Setto il range e lo stato del potenziometro che poi mi setta la frequenza al buzzer*/
  buzzer.frequence(range, stato);
}
```

Questo esempio riceve come parametro il range e lo stato del potenziometro e poi calcola la frequenza nel seguente modo:

```
/**
* Ritorna la frequenza che verrà applicata al buzzer
*/
void Buzzer::frequence(int range, int potValue)
{
    int frequence = range/1024*potValue;
    tone(_pin, frequence);
}
```

Riceve come parametro il range di frequenza e il valore del potenziometro dopodiché fa il calcolo che viene salvato nella variabile “frequence”, la quale poi viene applicata come frequenza al buzzer tramite il rispettivo pin.

BuzzerPotentiometerFrequenceReverse

```
/* Includere la libreria del Buzzer */
#include <Buzzer.h>
/* Includere la libreria del Buzzer */
#include <potentiometer.h>

/* Includo la libreria del Buzzer */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo Buzzer */
Buzzer buzzer(2);

/*Setta il range*/
int range= 16000;

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {
}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
  /* Prendo lo stato del potenziometro */
  int stato = pot.getValue();

  /*Setto il range e lo stato del potenziometro che
  *poi mi setta la frequenza al buzzer all'inverso*/
  buzzer.setOnReverseBuzzerFrequenze(range, stato);
}
```

Questo esempio si comporta quasi nello stesso modo di quello precedente ma la frequenza sarà diversa poiché viene calcolata in un altro modo.

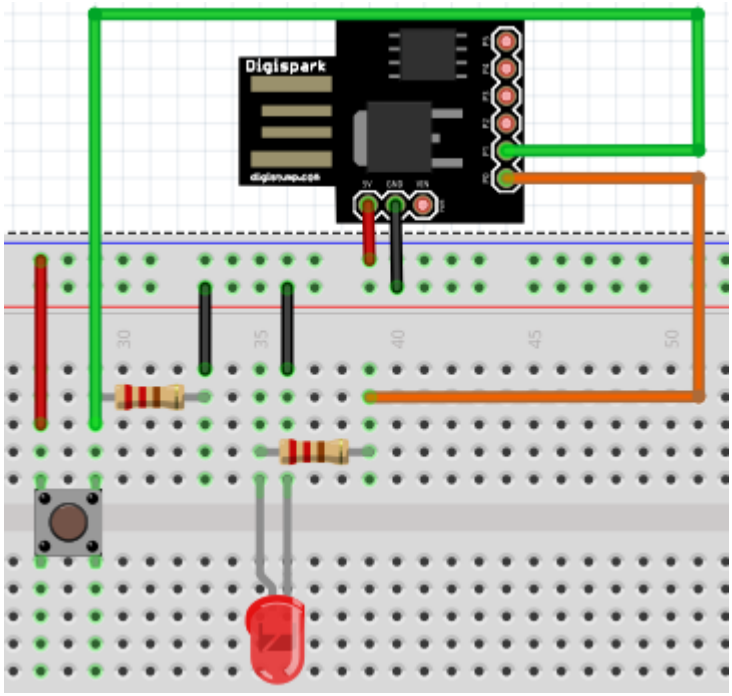
```
/**
* Setta la frequenza al buzzer all'inverso
*/
void Buzzer::setOnReverseBuzzerFrequenze(int range, int potValue){
    int frequence = range / 1024 * potValue;
    int reverse = 1024-frequence;
    tone(_pin, reverse);
}
```

Riceve come parametro il range di frequenza e il valore del potenziometro dopodiché fa il calcolo che viene salvato nella variabile “frequence”, dove poi verrà fatta la differenza tra il massimo valore del potenziometro e la “perquence”, questa differenza viene applicata come frequenza al buzzer tramite il rispettivo pin.

Led e Bottone

Prima di iniziare bisogna assicurarsi che i driver del Digispark siano installati sul PC (Vedi primo capito “Driver Digispark”).

Prima di scrivere il codice bisogna seguire i passi del capito sull’integrazione delle librerie (Vedi secondo capitolo “Come integrare le librerie”).



Per questo circuito si necessità: 1 led, 1 bottone, 2 resistenze, scheda Digispark.

Il bottone è collegato alla massa tramite la resistenza da 10 k Ω, al “5v” e al pin “P1” della scheda Digispark.
Il led è collegato alla massa e al pin “P0” attraverso una resistenza da 330Ω.

Esempi:

LedOnOff

```
/* Include la libreria del buttone */
#include <Button.h>
/* Include la libreria del led */
#include <Led.h>

/*Istanzia un oggetto di tipo led*/
Led led(1);
/*Istanzia un oggetto di tipo button*/
Button button(0);

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
  /* Prendo lo stato del bottone */
  bool state = button.getButtonValue();

  /* Se il bottone è premuto viene acceso, sennò viene spento. */
  if(state) {
    led.ledOn();
  } else {
    led.ledOff();
  }
}
```

Questo esempio accende il led se il bottone è premuto altrimenti il led rimane spento.

LedBlink

```
/* Includo la led button */
#include <Led.h>

/*Istanzio un oggetto di tipo button*/
Led led(1);

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {
}

/*Variabile millisecondi che specifica ogni
 *quanto bisogna fare il blink del led
 */
int milliseconds = 1000;

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
    /*Ritorna il valroe del led*/
    bool ledstate = led.getState();

    /*Se il led è acceso allora lo fa blinkare
     *ogni (milliseconds), sennò spegne il led
     */
    if(ledstate == HIGH){
      led.blink(milliseconds);
    }else{
      led.ledOff();
    }
}
```

Se il led è accesso allora esso viene fatto lampeggiare ogni tot millisecondi, i quali vengono specificati nella variabile “milliseconds”. Se il led non è acceso, in quel caso rimane spento.

LedToogle

```
/* Includo la libreria button */
#include <Button.h>
/* Includo la led button */
#include <Led.h>

/*Istanzio un oggetto di tipo led*/
Led led(1);
/*Istanzio un oggetto di tipo button*/
Button button(0);

/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
  /*Ritorna il valore del bottone*/
  int buttonState = button.getButtonValue();

  /*Setta il led con il pin del buttonState*/
  led.toggle(buttonState);
}
```

Viene ricavato il pin del pulsante e poi viene dato come parametro al metodo toggle() della libreria del Led.

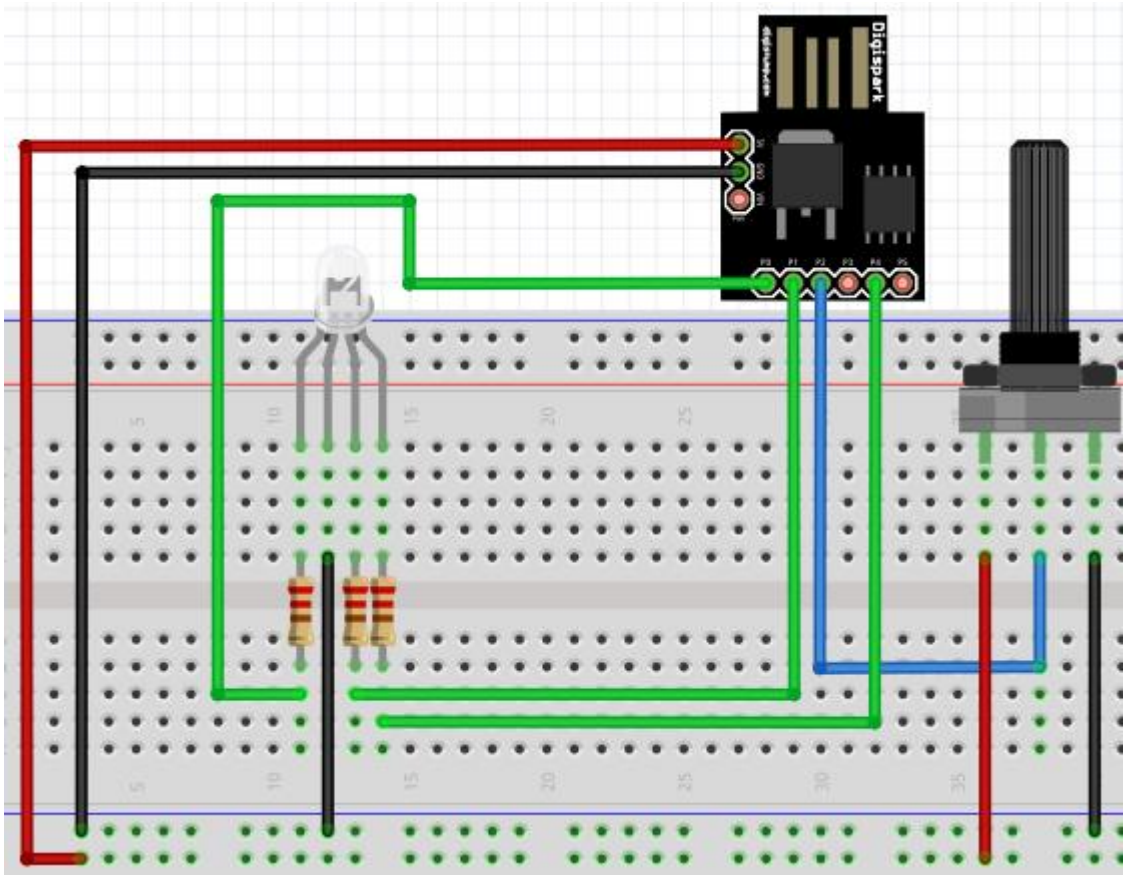
```
void Led::toggle(int buttonPin)
{
    bool mode = 0;
    bool buttonState = 0;

    if (!digitalRead(buttonPin)){
        if (!buttonState) {
          buttonState = true;
          Mode = !Mode;
        }else{
          buttonState = false;
        }

        digitalWrite(_pin, Mode);
        delay(5);
    }
}
```

Il metodo toogle() si occupa di cambiare lo stato del pulsante quando viene cliccato.

Led RGB e Potenziometro

Prima di iniziare bisogna assicurarsi che i driver del Digispark siano installati sul PC (Vedi primo capito “Driver Digispark”).

Prima di scrivere il codice bisogna seguire i passi del capito sull’integrazione delle librerie (Vedi secondo capitolo “Come integrare le librerie”).



Per questo circuito si necessità: 1 led RGB, 1 potenziometro, 3 resistenze, scheda Digispark.

Il potenziometro è collegato alla massa, al “5v” e al pin “P2” della scheda Digispark. Il led è collegato alla massa e al pin “P0”, al “P1” e al “P4” attraverso 3 resistenze da 330Ω.

Esempi:

AnalogRGB

```
/* Includere la libreria del Buzzer */
#include <RGB.h>
/* Includere la libreria del potenziometro */
#include <potentiometer.h>

/* Includo la libreria del potenziometro */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo rgb */
RGB rgb(0,1,4);


/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
  /* Prendo lo stato del potenziometro */
  int stato = pot.getValue();

  /*Setto il valore del potenziometro al metodo analogRGB della libreria RGB*/
  rgb.analogRGB(stato);
}
```

Si occupa di ricavare lo stato del potenziometro e poi di settare questo valore al metodo analogRGB()

```
void RGB::analogRGB(int val)
{
    if(val > 854){
        intensity = map(val-854,0,170,255,0);
        analogWrite(redPin, 0);
        analogWrite(greenPin, 255-intensity);
        analogWrite(bluePin, 255);
    }else if(val > 680){
        intensity = map(val-680,0,170,255,0);
        analogWrite(redPin, intensity);
        analogWrite(greenPin, 0);
        analogWrite(bluePin, 255);
    }else if(val > 510){
        intensity = map(val-510,0,170,255,0);
        analogWrite(redPin, 255);
        analogWrite(greenPin, 0);
        analogWrite(bluePin, 255-intensity);
    }else if(val > 340){
        intensity = map(val-340,0,170,255,0);
        analogWrite(redPin, 255);
        analogWrite(greenPin, intensity);
        analogWrite(bluePin, 0);
    }else if(val > 170){
        intensity = map(val-170,0,170,255,0);
        analogWrite(redPin, 255-intensity);
        analogWrite(greenPin, 255);
        analogWrite(bluePin, 0);
    }else{
        intensity = map(val,0,170,255,0);
        analogWrite(redPin, 0);
        analogWrite(greenPin, 255-intensity);
        analogWrite(bluePin, 0);
    }
}
```

In base al valore del potenziometro val (stato del potenziometro passato come parametro),
il range di quel valore (quello dell'if) viene mappato, calcolando l'intensità che deve avere ogni led ad una determinata fase.

DigitalRGB

```
/* Includere la libreria del Buzzer */
#include <RGB.h>
/* Includere la libreria del potenziometro */
#include <potentiometer.h>

/* Includo la libreria del potenziometro */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo rgb */
RGB rgb(0,1,4);


/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
  /* Prendo lo stato del potenziometro */
  int stato = pot.getValue();

  /*Setto il valore del potenziometro al metodo digitalRGB della libreria RGB*/
  rgb.digitalRGB(stato);
}
```

Si occupa di ricavare lo stato del potenziometro e poi di settare questo valore al metodo digitalRGB()

```
void RGB::digitalRGB(int val)
{
    if(val > 850){
        digitalWrite(_pinR, LOW);
        digitalWrite(_pinG, HIGH);
        digitalWrite(_pinB, HIGH);
    }else if(val > 680){
        digitalWrite(_pinR, LOW);
        digitalWrite(_pinG, LOW);
        digitalWrite(_pinB, HIGH);
    }else if(val > 510){
        digitalWrite(_pinR, HIGH);
        digitalWrite(_pinG, LOW);
        digitalWrite(_pinB, HIGH);
    }else if(val > 340){
        digitalWrite(_pinR, HIGH);
        digitalWrite(_pinG, LOW);
        digitalWrite(_pinB, LOW);
    }else if(val > 170){
        digitalWrite(_pinR, HIGH);
        digitalWrite(_pinG, HIGH);
        digitalWrite(_pinB, LOW);
    }else{
        digitalWrite(_pinR, LOW);
        digitalWrite(_pinG, LOW);
        digitalWrite(_pinB, LOW);
    }
}
```

In base al valore del potenziometro val (stato del potenziometro passato come parametro), vengono accesi e spenti i led.

ResetterRGB

```
/* Includere la libreria del Buzzer */
#include <RGB.h>
/* Includere la libreria del potenziometro */
#include <potentiometer.h>

/* Includo la libreria del potenziometro */
Potentiometer pot(1);
/* Istanziare un oggetto di tipo rgb */
RGB rgb(0,1,4);


/*Metodo setup che viene avviato una volta sola all'inizio*/
void setup() {

}

/*Metodo loop che viene richiamato all'infinito*/
void loop() {
  /* Prendo lo stato del potenziometro */
  int stato = pot.getValue();

  /*Setto il valore del potenziometro al metodo resetterRGB della libreria RGB*/
  rgb.resetterRGB(stato);
}
```

Si occupa di ricavare lo stato del potenziometro e poi di settare questo valore al metodo resetterRGB()

```
void RGB::resetterRGB(int val)
{
    bool maxed = false;
    int reset = 0;
    int intensity = 0;

    if(val > 1000){
        maxed = true;
    }

    if(maxed && val < 2){
        reset++;
        maxed = false;
    }
    if(reset%4 == 0){
        intensity = map(val,0,1023,255,0);
        analogWrite(redPin, intensity);
        analogWrite(greenPin, intensity);
        analogWrite(bluePin, intensity);
    }else if(reset%4 == 1){
        intensity = map(val,0,1023,255,0);
        analogWrite(redPin, intensity);
        analogWrite(greenPin, 255);
        analogWrite(bluePin, 255);
    }else if(reset%4 == 2){
        intensity = map(val,0,1023,255,0);
        analogWrite(redPin, 255);
        analogWrite(greenPin, intensity);
        analogWrite(bluePin, 255);
    }else if(reset%4 == 3){
        intensity = map(val,0,1023,255,0);
        analogWrite(redPin, 255);
        analogWrite(greenPin, 255);
        analogWrite(bluePin, intensity);
    }

}
```

Quanodo il potenziometro ha raggiunto almeno una voltà l'intensità massima,e sucessivamente quella minima, viene incrementato il valore reset, che col modulo 4 definirà in che fase di colore si trova.