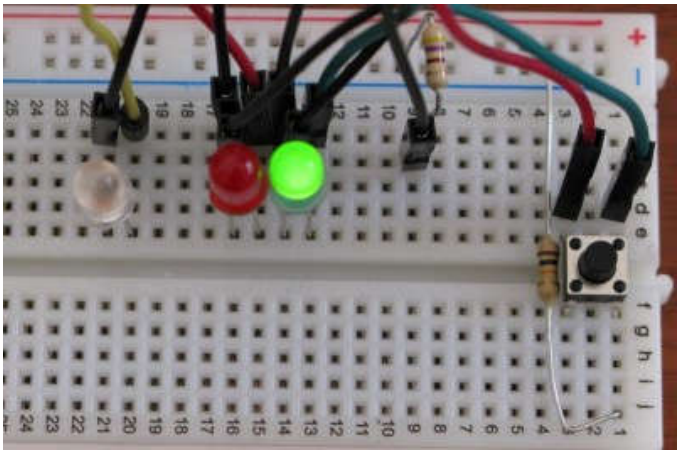
Interrupt con Arduino

In questi giorni mi sono imbattuto in alcuni progetti che fanno uso di *interrupt* e incuriosito ho deciso di approfondire l'argomento, realizzando un tutorial semplice per comprendere meglio come funzionano in Arduino.

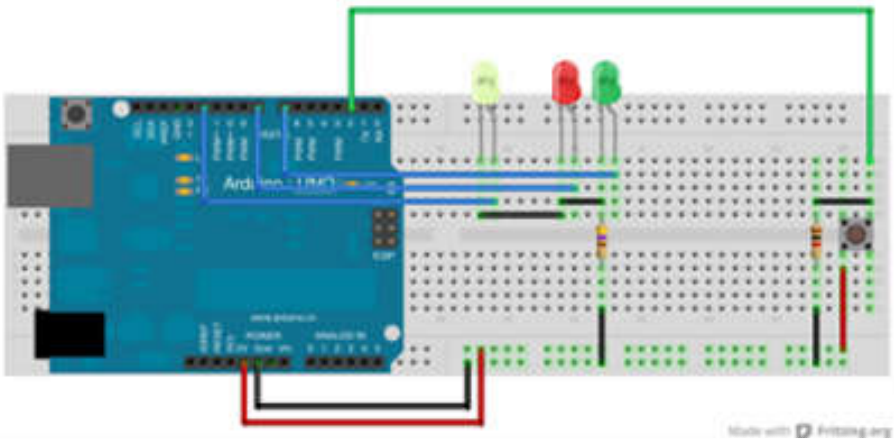


*In informatica, un **interrupt** o **interruzione** è un segnale asincrono che indica il 'bisogno di attenzione' da parte di una periferica finalizzata ad una particolare richiesta di servizio, un evento sincrono che consente l'interruzione di un processo qualora si verifichino determinate condizioni (gestione dei processi) oppure più in generale una particolare richiesta al sistema operativo da parte di un processo in esecuzione.*

Cosa ti occorre

Per questo esperimento ti occorre: 1 Arduino Uno, 1 led verde, 1 led rosso, 1 led giallo, 1 pulsante N/A, 1 resistenza da 470Ω, 1 resistenza da 1KΩ, 1 breadboard e relativi cavetti.

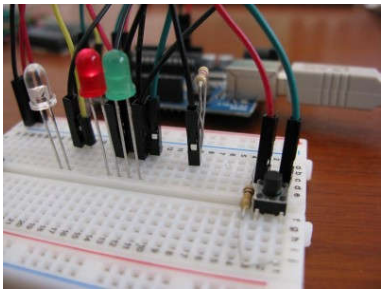
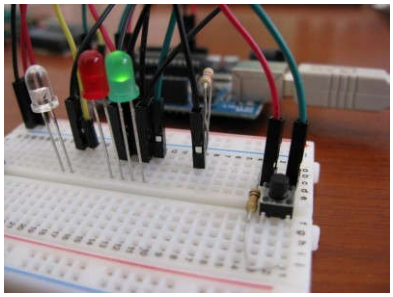
Collega il pulsante al pin 2 che utilizzerai come interrupt, alla pressione del pulsante il valore letto sul pin 2 sarà HIGH in quanto la pressione dl pulsante mette in contatto il pin 2 di Arduino con il polo positivo +5V.



in pratica

Parti dallo sketch che ho scritto e ti sarà più chiaro come funziona un interrupt, premetto solo che in Arduino Uno i pin che puoi legare a interrupt sono il pin 2 ed il pin 3 rispettivamente come interrupt 0 (il pin 2) e 1 (il pin 3).

```
/**
 * Interrupt Test - Alfieri Mauro
 *
 * Web: www.mauroalfieri.it
 * Tw:  @mauroalfieri
 */

int ledRed=8;
int ledGreen=7;
int ledYellow=12;
int interrCount=0;

void setup()
{
  pinMode(ledRed, OUTPUT);
  pinMode(ledGreen, OUTPUT);
  pinMode(ledYellow, OUTPUT);

  digitalWrite(ledRed, LOW);
  digitalWrite(ledGreen, LOW);
  digitalWrite(ledYellow, LOW);

  attachInterrupt(0, interruptGiallo, RISING);
}

void loop() {
  interrCount++;

  digitalWrite(ledRed, HIGH);
  digitalWrite(ledGreen, LOW);
  delay(300);
  digitalWrite(ledRed, LOW);
  digitalWrite(ledGreen, HIGH);
  delay(300);

  if ( interrCount == 10 )
  {
    interrCount = 0;
    digitalWrite(ledYellow, LOW);
  }

}

void interruptGiallo()
{
  digitalWrite(ledYellow, HIGH);
}
```

Lo sketch ha lo scopo di far lampeggiare alternatamente i due led rosso e verde con un periodo di 300 millisecondi tra l'uno e l'altro; alla pressione del pulsante si accende il led giallo e resta accesso fino a quando la funzione loop() non ha eseguito i suoi 10 cicli.

leggendo linea per linea:
linea 08: definisci una variabile di tipo *integer* denominata ledRed il cui valore è quello del pin a cui è collegato il pin rosso;
linea 09: definisci una variabile simile alla precedente per il led verde;
linea 10: definisci una variabile simile alle precedenti per il led giallo;
linea 11: definisci una variabile per contare i cicli di loop() dopo cui spegnere il led giallo;
linee 15-17: per ciascun pin legato ad un led imposta la modalità di OUTPUT;
linee 19-21: per ciascun led spegni il led corrispondente portando a LOW il valore digitale del pin;
linea 23: è il cuore di questo esempio, il comando *attachInterrupt(interrupt, function, mode)* i cui parametri indicano
INTERRUPT: il pin di interrupt (0 o 1) cui attaccare la funzione specificata come secondo parametro
FUNCTION: la funzione da richiamare quando si verifica un evento sul pin specificato
MODE: la modalità in cui l'interrupt è attivato, per questo punto sul sito Arduino sono riportate le 4 modalità di funzionamento

- **LOW** fa scattare l'evento di interrupt ogni volta che il pin (2 o 3) ha valore logico basso
- **CHANGE** fa scattare l'evento di interrupt quando il pin cambia stato
- **RISING** fa scattare l'evento di interrupt quando il pin passa da LOW a HIGH
- **FALLING** fa scattare l'evento di interrupt quando il pin passa da HIGH a LOW

nel tuo caso potresti utilizzare sia CHANGE sia RISING io consiglio RISING in modo che il l'interrupt sia verificato solo alla pressione del pulsante e non anche quando rilasci il pulsante.
linea 27: subito dopo la definizione della funzione loop() incrementa il valore della variabile interrCount ad ogni loop();
linea 29-34: si occupano di accendere e spegnere in nodo alternato i due led rosso e verde, queste semplici righe servono anche a dimostrarti come funziona l'interrupt in quanto si alternano indipendentemente dalla funzione di interrupt o dal verificarsi dell'evento;
linea 36: verifica che il numero di cicli di loop eseguiti sia 10;
linea 38: reimposta a 0 il valore di interrCount, in modo che il conteggio possa ricominciare al loop() successivo;
linea 39: spegne il led giallo, indipendentemente se sia mai stato acceso o meno, ha il solo scopo di far continuare l'esperimento.
linea 44: definisci la funzione *interruptGiallo* che hai indicato, alla linea 23, come funzione da richiamare in caso di evento di interrupt rilevato.
linea 46: accendi il led giallo utilizzando il comando digitalWrite e portando a HIGH il valore del pin a cui è collegato il led.
Alcune note sulla funzione richiamata dal comando attachInterrupt: il comando delay() non funziona, così come il valore restituito dalla funzione millis() non si incrementa e non puoi usarlo come contatore del tempo.