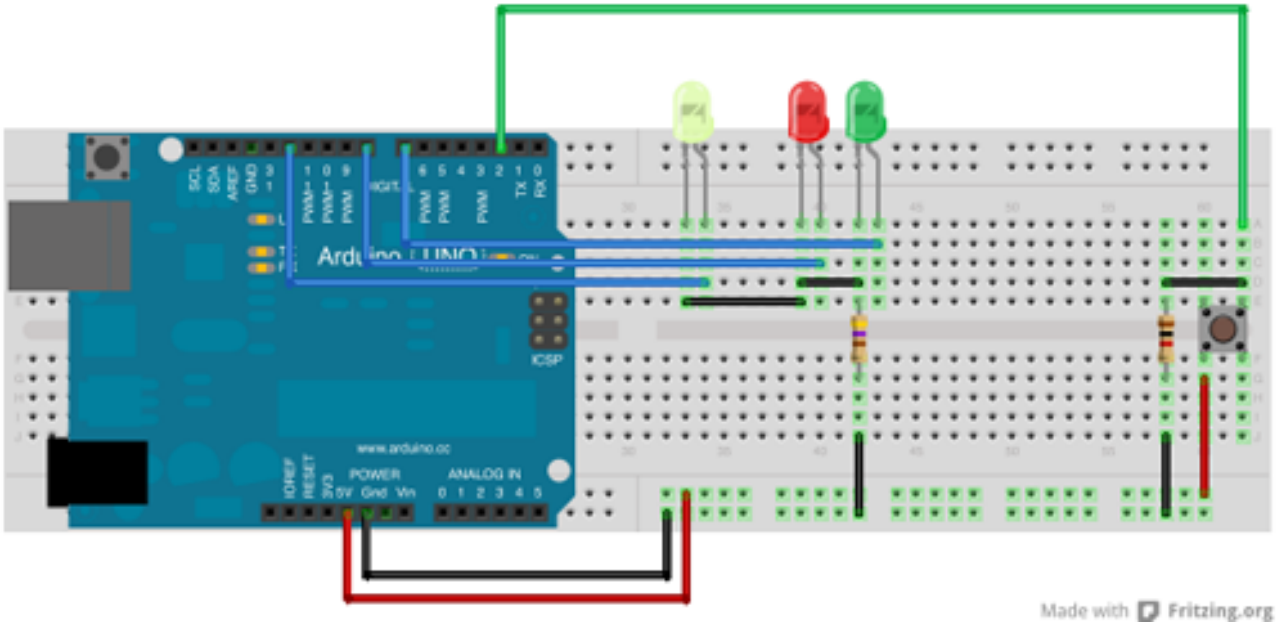
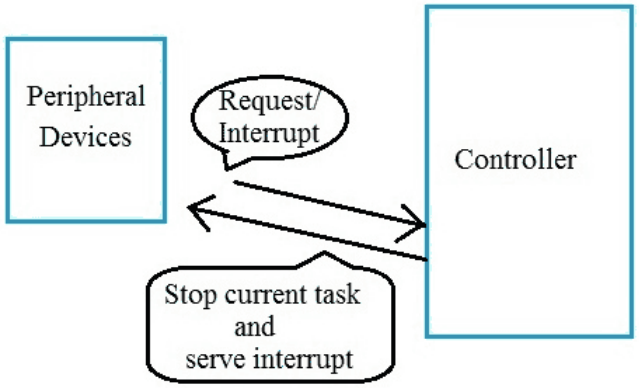
Gestione delle Interruzioni (INTERRUPT)



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 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.***

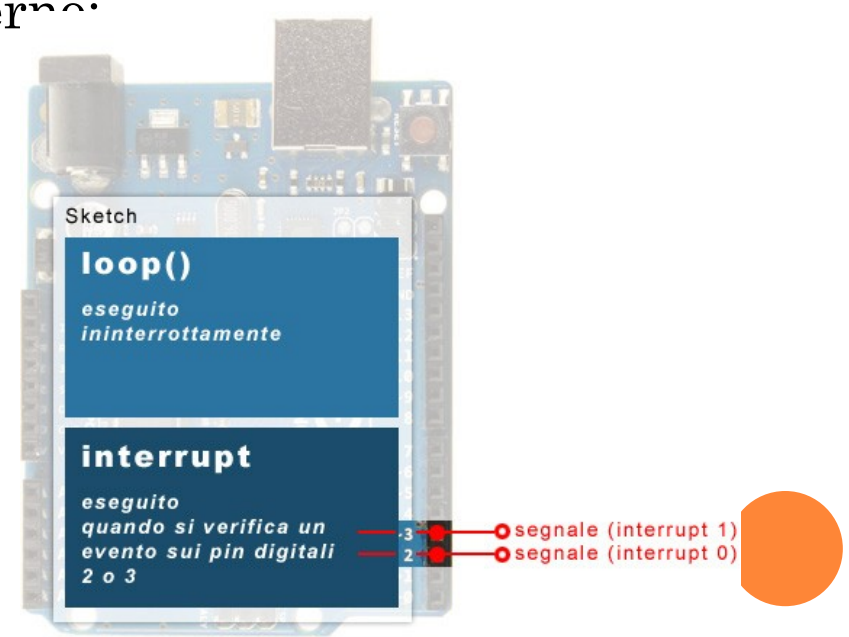
Possiamo immaginare dunque un interrupt come qualcosa sempre in allerta da parte di arduino, quando si verifica la condizione per cui è stato impostato il programma [le istruzioni della funzione loop()] si sgancia dall'esecuzione in corso e va ad eseguire una routine di risposta all'interruzione terminata la quale ritorna al punto esatto, all'interno del loop(), in cui si è verificata l'interruzione.

Ci sono due tipi di interrupt:
Interrupt hardware generati da dispositivi esterni alla CPU (periferiche);
Interrupt software: sono delle istruzioni che possono essere assimilate alle chiamate di sottoprogrammi, ma che sfruttano il meccanismo delle interruzioni per passare il controllo dal programma chiamante a quello chiamato, e viceversa; vengono utilizzati per accedere direttamente alle risorse del sistema operativo.
L'interrupt viene generata quando si verifica una variazione di stato su un particolare pin della scheda Arduino.
La scheda Arduino UNO possiede 2 piedini abilitati alla ricezione di interrupt generati da un segnale esterno:

pin digitale 2: interrupt 0
pin digitale 3: interrupt 1

L'Arduino mega dispone di 4 piedini abilitati alla ricezione di interrupt:

pin digitale 21: **interrupt 2**
pin digitale 20: **interrupt 3**
pin digitale 19: **interrupt 4**
pin digitale 18: **interrupt 5**



L'istruzione
attachInterrupt(interrupt,funzione,modo)
specifica la routine da chiamare quando avviene l' interrupt

interrupt : indica il numero dell'interrupt
funzione: è la funzione che viene chiamata quando si verifica l'interrupt; questa funzione non deve avere parametri e non deve restituire alcun valore;
Il parametro ***modo*** può assumere quattro valori
LOW l'interrupt viene eseguito quando il pin è allo stato 0
CHANGE l'interrupt viene eseguito quando il pin cambia di stato
RISING l'interrupt viene eseguito quando il pin passa da 0 a 1 (fronte di salita)
FALLING l'interrupt viene eseguito quando il pin passa da 1 a 0 (fronte di discesa)

Esistono pure:
La funzione ***interrupts()*** serve ad abilitare l'interrupt globalmente
La funzione ***noInterrupts()*** serve a disabilitare l'interrupt globale
La funzione ***detachInterrupt(interrupt)*** interrompe l'interrupt.

ATTENZIONE: all'interno della funzione utilizzata in **attachInterrupt**:

- il delay() non funziona;
- il valore restituito dalla funzione millis() non verrà incrementato.
- i dati seriali ricevuti durante l'esecuzione della funzione di interrupt possono essere sbagliati.
- qualsiasi variabile globale modificabile all'interno della funzione attached (chiamata all'interno attachInterrupt) deve essere dichiarata come volatile.
- Le funzioni attached non possono avere parametri

Quando gli interrupts sono disabilitati, alcune funzioni non lavorano e le comunicazioni in arrivo (eventi esterni) potrebbero essere ignorate.
Gli interrupt possono variare leggermente il timing di esecuzione del codice.

Esempio 1: LED COLLEGATO A UN PULSANTE

```
int pin = 13;
volatile int state = LOW;
 /* dichiariamo volatile la variabile
state usata nella funzione usata all'interno di attachInterrupt */

void setup()
{
pinMode(pin, OUTPUT); // definiamo pin 13 output
attachInterrupt(0, blink, CHANGE);
 // usiamo l'interrupt 0 che è associato al pin digitale 2
// attachInterrupt chiamerà la funzione collegata blink
// il modo per la rilevazione del cambiamento di stato sarà di tipo: CHANGE
// cioè l'interrupt viene eseguito quando avviene un qualsiasi cambiamento di
// stato sul pin
 }
```

Esempio 1: LED COLLEGATO A UN PULSANTE

```
void loop()
{
digitalWrite(pin, state);
// il pin digitale 13 viene impostato a "state" che può essere LOW o HIGH
delay(300);
}

void blink()
// la funzione blink() esegue la funzione NOT di "state" cioè
// se state = LOW viene cambiato in HIGH, se state = HIGH viene cambiato
// in LOW
 {
state = !state;
}
```

Riportare :

- **Flow chart**
- **Schema di collegamento (inserire foto)**
- **Provare con le altre modalità di interrupt**

Esercitazioni:

- **Aggiungere un display e generare un numero casuale alla pressione del tasto**
- **Realizzare un contatore di presenze (ad ogni pressione del pulsante visualizzare un numero progressivo**
- **Realizzare un cronometro con display LCD (utilizzare 2 pulsanti: uno per far partire il Timer e l'altro per fermarlo)**