GLI INTERRUPT con ARDUINO

Con il termine interrupt (interruzione) intendiamo un segnale asincrono che indica la necessità di "attenzione" da parte di una periferica collegata ad Arduino.

L'interrupt viene generato quando si verifica una variazione di stato su uno dei piedini di Arduino. Normalmente il microcontrollore esegue all'interno del loop() in modo sequenziale e ripetitivo le istruzioni in esso inserite, ma quando si verifica un interrupt viene interrotto il flusso delle istruzioni all'interno del loop() ed invocate altre routine (create dall'utente). Quando le routine terminano il flusso del programma prosegue normalmente.

L'utilizzo dell'interrupt è particolarmente utile quando abbiamo la necessità di gestitire in background alcune routine ovvero eseguire istantaneamente un'operazione nel caso si manifesti un evento asincrono esterno.

Capita spesso di avere la necessità di controllare lo stato di diversi pin di input, in questa situazione potrebbe capitare che il programma non si accorga del cambiamento di stato su uno dei pin, utilizzando gli interrupt evitiamo questo tipo di errore.

Esempio:

Supponete di dover rilevare lo stato di alcuni sensori esterni. All'interno del loop() tutte le istruzioni sono eseguite in modo sequenziale e quindi anche la rilevazione del cambiamento di stato dei sensori collegati ad Arduino avviene in modo sequenziale. Supponete di dover eseguire il controllo della variazione di stato di 3 sensori, il vostro sketch eseguirà il controllo sul primo sensore, sul secondo e poi sul terzo.

Supponete che tutti i sensori si trovino al medesimo stato iniziale che chiameremo S1 e che il controllo di Arduino sia in un determinato istante sul secondo sensore, potrebbe capitare nello stesso istante una variazione repentina di stato sul primo sensore che passa da S1 a S2 e poi a S1, Arduino non si accorgerà di nulla perché la sua attenzione è sul secondo sensore; ecco che in questa situazione potrebbe essere utile l'utilizzo degli interrupt.

La maggior parte delle schede Arduino hanno due piedini abilitati alla ricezione di interrupt:

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

Funzioni utilizzate per gli interrupt

```
attachInterrupt(interrupt,funzione,modo)
```

istruzione che specifica la funzione da chiamare quando avviene interrupt (interruzione esterna).
Sostituisce ogni altra precedente funzione che è stata collegata all'interrupt.

- ***interrupt*** indica il numero dell'interrupt, cioè indicheremo 0 per il pin 2, e 1 per il pin 3;
- ***funzione***: è la funzione che viene richiamata quando si verifica l'interrupt; questa **funzione non deve avere parametri e non deve restituire alcun valori;** questa funzione a volte viene rinviata alle routine di gestione degli interrupt;
- La rilevazione della variazione di stato sui pin di Arduino può avvenire in modi diversi è può essere configurata in funzione di come è impostato il parametro ***modo*** utilizzato per scatenare l'interrupt, questo parametro può assumere quattro valori
  - **LOW** l'interrupt viene eseguito quando il livello del segnale è basso
  - **CHANGE** l'interrupt viene eseguito quando avviene un cambiamento di stato sul pin
  - **RISING** l'interrupt viene eseguito quando si passa da un livello LOW ad un livello HIGH
  - **FALLING** l'interrupt viene eseguito quando si passa da un livello HIGH ad un livello LOW

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

- **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 modificabile** all'interno della funzione attached (chiamata all'interno attachInterrupt) devono essere dichiarare come **volatili**.

```
detachInterrupt(interrupt)
```

funzione che interrompe l'interrupt.

```
interrupts()
```

Abilita nuovamente gli interrupt (dopo che questi sono stati disabilitati da noInterrupts()).

Gli interrupts, abilitati per defalut, consentono ad eventi esterni di essere eseguiti in background. 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 per particolari sezioni critiche del codice possono essere disabilitati.

```
void setup( ) { }
void loop( ){
noInterrupts( );
// ... sezione critica, codice dipendente dal tempo
interrupts( );
// altro codice
}
```

noInterrupts()
disabilita interrupts (che sono stati abilitati con interrupts( )).

```
void setup() {}
void loop()
{
noInterrupts();
// ... sezione critica, codice dipendente dal tempo
interrupts();
// altro codice
}
```

Esempio 1

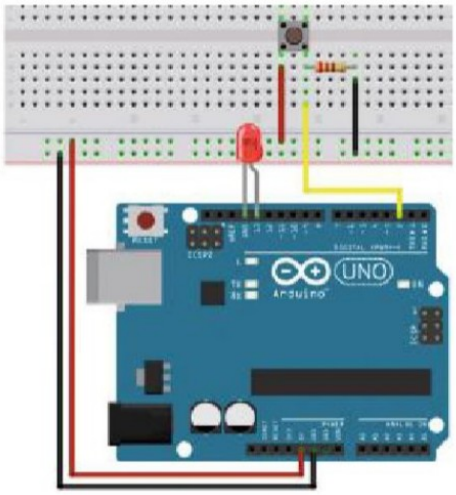
```
int pin = 13;
volatile int state = LOW;
/* dichiariamo volatile la variabile state usata nella funzione usata all'interno
di attachInterrupt */

void setup()
{
pinMode(pin, OUTPUT); // definiamo pin 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
}

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

Esempio 2

Al pigiare del pulsante sul pin2 avremo un fronte di salita (da 0 a 5 V) e di conseguenza il led cambia stato (da spento si accende e da acceso si spegne)



Per evitare il rimbalzo del pulsante poniamo una resistenza di 220 ohm tra il capo collegato al piedino 2 della scheda Arduino e la massa (GND).

interrupt

```
//Dichiarazione variabile di tipo volatile
//Modificata nella funzione attached
volatile int stato = 0;
void setup()
{
   //Associamo int 0 (cioè pin 2) alla funzione attached accendi
   //che verrà richiamata ad ogni cambiamento di stato del pin 2
   //Al pin2 abbiamo invece collegato il pulsante
   attachInterrupt(0, accendi, RISING);
   //Definiamo il pin 13 per collegare il LED in output
   pinMode(13, OUTPUT);
}
void loop()
{
   //Il pin 13 viene impostato a quanto contenuto nella
   //variabile globale stato, che può essere 1 oppure 0
   digitalWrite(13, stato);
   delay(300);
}
void accendi()
{
   //La funzione effettua il NOT logico sulla variabile stato
   stato = !stato;
}
```