La funzione millis()

Il comando **delay** è molto comodo ma ha un grandissimo difetto: quello di fermare completamente il programma. Quando viene impartito un comando **delay(1000)**, ad esempio, il programma si fermerà per 1 sec arrestandolo completamente. Solo quando sarà terminato questo intervallo di tempo verrà eseguito il comando seguente.

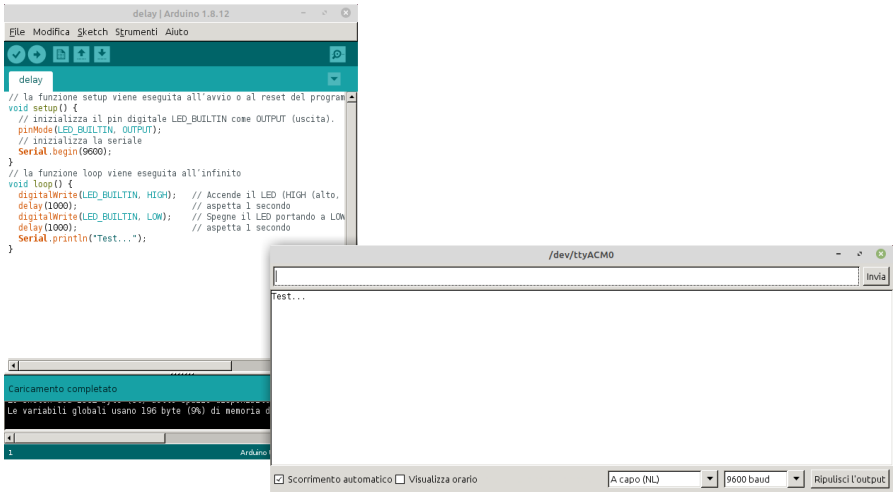
Prendiamo l'esempio **01Basic > Blink**, che fa lampeggiare un led con un intervallo di 1 secondo:

```
// la funzione setup viene eseguita all'avvio o al reset del programma
void setup() {
  // inizializza il pin digitale LED_BUILTIN come OUTPUT (uscita).
  pinMode(LED_BUILTIN, OUTPUT);
}
// la funzione loop viene eseguita all'infinito
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // Accende il LED (HIGH (alto, 5V) è la
tensione del pin)
  delay(1000);                       // aspetta 1 secondo
  digitalWrite(LED_BUILTIN, LOW);    // Spegne il LED portando a LOW (basso, 0V)
la tensione del pin
  delay(1000);                       // aspetta 1 secondo
}
```

Basterà aggiungere un semplice comando di stampa verso la seriale per renderci conto di quanto è stato affermato.

```
// la funzione setup viene eseguita all'avvio o al reset del programma
void setup() {
  // inizializza il pin digitale LED_BUILTIN come OUTPUT (uscita).
  pinMode(LED_BUILTIN, OUTPUT);
  // inizializza la seriale
  Serial.begin(9600);
}
// la funzione loop viene eseguita all'infinito
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // Accende il LED (HIGH (alto, 5V) è la
tensione del pin)
  delay(1000);                       // aspetta 1 secondo
  digitalWrite(LED_BUILTIN, LOW);    // Spegne il LED portando a LOW (basso, 0V)
la tensione del pin
  delay(1000);                       // aspetta 1 secondo
    Serial.println("Test..."); // Invia il messaggio sulla seriale
}
```

Il monitor seriale mostrerà il messaggio ogni 2 secondi esatti.



Se volessimo aggiungere un pulsante per accendere e spegnere il led non potremo farlo con il codice scritto in questo modo perché Arduino potrà leggere la condizione dello stesso solo prima o dopo i comandi delay. Se

non premessimo il pulsante durante il secondo in cui è in esecuzione il comando **delay** sarà come se non lo avessimo fatto.
Come risolvere il problema? Non ha senso usare gli interrupt che sono utili in caso estremo: ad esempio se usassimo un pulsante per fermare istantaneamente una macchina CNC. Poi Arduino Uno ha solo due pin usabili per tale scopo.
La soluzione più ovvia è quella di usare la funzione **millis()** che restituisce il numero di millisecondi trascorsi da quando la scheda Arduino ha iniziato a eseguire il programma corrente. Per memorizzare il tempo del timer viene usata la variabile **`previousMillis`** di tipo unsigned long che avendo come valore massimo 4.294.967.295, cioé 4294967295 millisecondi / (24 ore x 3600 secondi x 1000 millisecondi) = 49,7 giorni, si azzererà dopo questo periodo di tempo.
`millis() - previousMillis` calcolerà il tempo trascorso dal precedente cambio di stato del LED.

Vediamo l'esempio 02Digital > BlinkWithoutDelay, da me semplificato e tradotto:

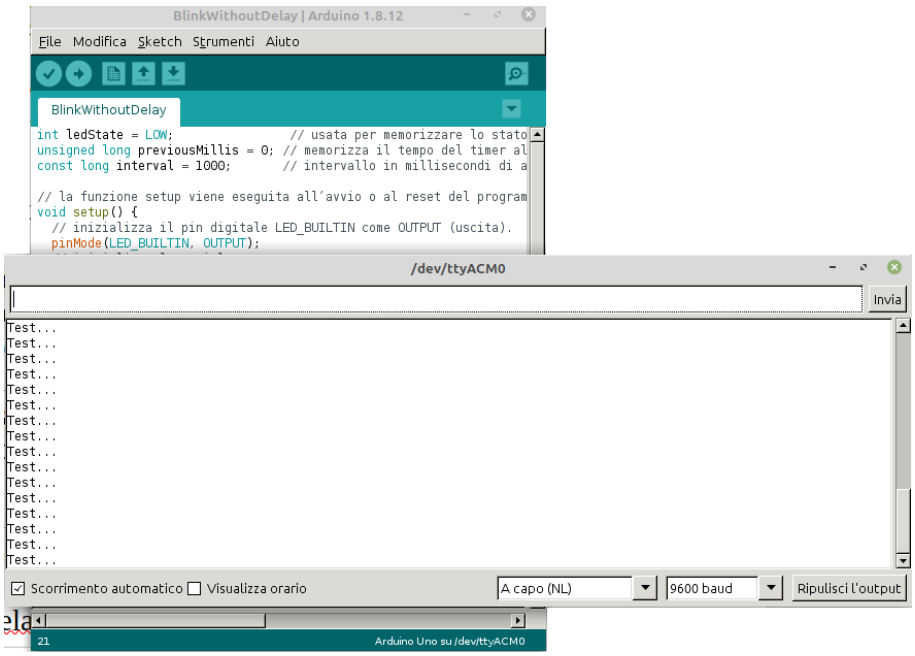
```
int ledState = LOW;                 // usata per memorizzare lo stato del LED
unsigned long previousMillis = 0;  // memorizza il tempo del timer al cambio di stato del LED
const long interval = 1000;        // intervallo in millisecondi di accensione o spegnimento

// la funzione setup viene eseguita all'avvio o al reset del programma
void setup() {
  // inizializza il pin digitale LED_BUILTIN come OUTPUT (uscita)
  pinMode(LED_BUILTIN, OUTPUT);
  previousMillis = millis(); // memorizza il tempo corrente
}
// la funzione loop viene eseguita all'infinito
void loop() {
  if (millis() - previousMillis >= interval) { // se è passato più di 1 sec
    previousMillis = millis(); // salva il tempo in cui è cambiato lo stato del led
    // inverte lo stato del LED
    ledState = !ledState;  // accende o spegne il LED in base al valore di ledState
    digitalWrite(LED_BUILTIN, ledState);
  }
}
```

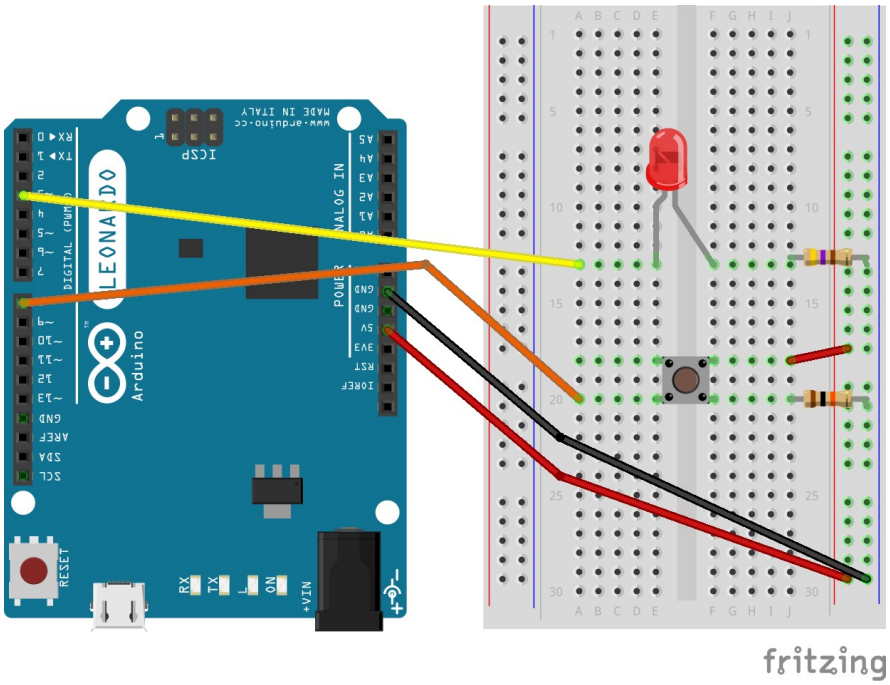
Ora se aggiungiamo, anche a questo esempio, il comando di stampa vedremo che i messaggi arriveranno al monitor molto velocemente.

```
int ledState = LOW;                 // usata per memorizzare lo stato del LED
unsigned long previousMillis = 0; // memorizza il tempo del timer al cambio di stato del LED
const long interval = 1000;        // intervallo in millisecondi di accensione o spegnimento

// la funzione setup viene eseguita all'avvio o al reset del programma
void setup() {
  // inizializza il pin digitale LED_BUILTIN come OUTPUT (uscita).
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);         // inizializza la seriale
  previousMillis = millis(); // memorizza il tempo corrente
}
// la funzione loop viene eseguita all'infinito
void loop() {
  if (millis() - previousMillis >= interval) { // se è passato più di 1 sec
    previousMillis = millis(); // salva il tempo in cui è cambiato lo stato del led
    ledState = !ledState;  // inverte il valore di ledState
     digitalWrite(LED_BUILTIN, ledState); // accende o spegne il LED in base al valore di
ledState
  }
    Serial.println("Test..."); // Invia il messaggio sulla seriale
}
```



Un esempio pratico è quello che aggiunge un pulsante per accendere il lampeggio del led. Il lampeggio ci sarà solo se teniamo premuto il pulsante.
Al posto del led sull'Arduino usiamo un led collegato al pin 3 e un pulsante collegato al pin 8.
Usiamo lo stesso schema dell'Esercizio 2.




```
const int pinLED = 8;     // imposta il pin a cui è collegato il LED
const int pinBUTTON = 3; // imposta il pin a cui è collegato il pulsante
int ledState = LOW;       // memorizza lo stato del LED
unsigned long previousMillis = 0;  // memorizza il tempo del timer al cambio di stato del LED
unsigned long previous1Millis = 0; // memorizza il tempo del timer alla premuta del pulsante
const long interval = 1000;        // intervallo in millisecondi di accensione o spegnimento
boolean buttonState = false;       // memorizza lo stato del pulsante

// la funzione setup viene eseguita all'avvio o al reset del programma
void setup() {
```

```
  // inizializza i pin digitali pinLED come OUTPUT (uscita) e pinBUTTON come INPUT (entrata)
  pinMode(pinLED, OUTPUT);
  pinMode(pinBUTTON, INPUT);
  Serial.begin(9600);         // inizializza la seriale
  previousMillis = millis(); // memorizza il tempo corrente
}
// la funzione loop viene eseguita all'infinito
void loop() {
  buttonState = digitalRead(pinBUTTON); // legge e memorizza lo stato del pulsante
  // se il pulsante  premuto ed è passato più di 1 sec
  if (buttonState & millis() - previousMillis >= interval)
    previousMillis = millis(); // salva il tempo in cui è cambiato lo stato del led
    ledState = !ledState;  // inverte il valore di ledState
    digitalWrite(pinLED, ledState); // accende o spegne il LED in base al valore di ledState
  }
}
```

Un'altra variante può essere quella di usare il pulsante per accendere e spegnere il lampeggio del led: se premo una volta il led lampeggia, la seconda il led si spegne.
Per risolvere il problema ho bisogno di un'altra variabile **`unsigned long`** che ho chiamato **`previous1Millis`** per salvare il tempo del timer alla premuta del pulsante.
Ho aggiunto altre tre variabili: **`oldbuttonState`** che memorizza lo stato precedente del pulsante, **`pulseState`** e **`oldpulseState`** che memorizzano lo stato del lampeggio e quello precedente.

```
const int pinLED = 8;     // imposta il pin a cui è collegato il LED
const int pinBUTTON = 3; // imposta il pin a cui è collegato il pulsante
int ledState = LOW;       // memorizza lo stato del LED
unsigned long previousMillis = 0; // memorizza il tempo del timer al cambio di stato del LED
unsigned long previous1Millis = 0;  // memorizza il tempo del timer alla premuta del pulsante
const long interval = 1000;        // intervallo in millisecondi di accensione o spegnimento
boolean buttonState = false;       // memorizza lo stato del pulsante
boolean oldbuttonState = false;    // memorizza lo stato precedente del pulsante
boolean pulseState = false;    // memorizza lo stato del lampeggio
boolean oldpulseState = false;    // memorizza lo stato precedente del lampeggio
// la funzione setup viene eseguita all'avvio o al reset del programma
void setup() {
  // inizializza i pin digitali pinLED come OUTPUT (uscita) e pinBUTTON come INPUT (entrata)
  pinMode(pinLED, OUTPUT);
  pinMode(pinBUTTON, INPUT);
  Serial.begin(9600);         // inizializza la seriale
  previousMillis = millis(); // memorizza il tempo corrente
}

// la funzione loop viene eseguita all'infinito
void loop() {
  buttonState = digitalRead(pinBUTTON); // memorizza lo stato del pulsante
  // se il pulsante è premuto ma non lo era prima
  if (buttonState & !oldbuttonState) {
    previous1Millis = millis(); // memorizza il tempo corrente
    oldbuttonState = true; // pone il vecchio stato del pulsante a vero
  // se il pulsante è stato rilasciato ma era premuto in precedenza
  } else if (buttonState == false && oldbuttonState) {
    oldbuttonState = false;  // pone il vecchio stato del pulsante a falso
    oldpulseState = false; // pone la variabile oldpulseState a falso
  }
  //se il pulsante è premuto, era premuto in precedenza dopo essere stato rilasciato ed è stato premuto per almeno 50 millisecondi
```

```
   if (buttonState & oldbuttonState && oldpulseState == false && millis() -
previous1Millis > 50) {
    pulseState = !pulseState; // inverte il valore di state (stato del led)
    oldpulseState = true;  // lo stato del led è cambiato
  }
  if (pulseState & millis() - previousMillis >= interval) {
    previousMillis = millis(); // salva il tempo in cui è cambiato lo stato del led
    ledState = !ledState;  // inverte il valore di ledState
      digitalWrite(pinLED, ledState); // accende o spegne il LED in base al valore di
ledState
  }
  if (!pulseState) {
    digitalWrite(pinLED, LOW); // spegne il LED
    ledState = false;
  }
}
```