Come funzionano i timer hardware?
Per "perdere tempo" in un programma potremmo pensare di eseguire delle operazioni "nop" per far eseguire al processore alcuni cicli di clock "a vuoto".

Supponendo che ogni "nop" prenda 2 cicli di clock, e che il nostro processore sia da 1 MHz, ogni "nop" farebbe "attendere" il processo per soli 2 ms.

Un approccio leggermente più intelligente potrebbe essere creare un loop dove si decrementa una variabile per 255 volte finché essa non arriva da 255 a 0.
Il problema di questo approccio, però, è sempre lo stesso: in primo luogo non sappiamo quanto delay stiamo introducendo. Inoltre, il problema è che mentre la CPU è in questo loop, non può fare nient'altro.

La soluzione è quella di usare un TIMER HARDWARE.

Per comprendere meglio il funzionamento, possiamo usare un Arduino, che monta il microcontroller ATmega328P. Quando si usa la funzione delay() in arduino, viene invocata una funzione che ha come unico compito quello di far entrare il programma in un loop, e quindi, non affidabile.

ATmega328P Timer : normal mode

Per usare il timer del processore, dobbiamo impostare alcuni suoi registri da codice.

Quello che accade nella pratica, è un COUNTER che conta da 0 a 2^{16} bit; quando il counter arriva a 65535 viene azzerato, ed inizia a contare nuovamente.

Quando il counter arriva a 65535, però, avviene un OVERFLOW, che possiamo sfruttare.

Possiamo infatti scrivere una funzione che faccia qualcosa ogni volta che incontra un'interruzione

```
ISR(TIMER1_OVF_vect) {...}
```

Per creare delle interruzioni ad esattamente 1 s, ci basta aumentare il valore da cui il counter inizia a contare.