Faire clignoter l'Arduino (manipulation explicite du temps)

Michel Billaud (michel.billaud@laposte.net)

11 juin 2023

Table des matières

1 Motivation

Dans les forums consacrés à la programmation, on trouve souvent des appels à l'aide de débutants qui n'arrivent pas à réaliser des programmes qui leur paraissent pourtant simple comme

> quand on appuie sur le bouton, la LED doit clignoter pendant 5 secondes

Pourtant ils ont vu des tutoriels leur expliquant

- comment faire clignoter une LED avec `delay()` et `digitalWrite()` ;
- comment tester l'état du bouton avec `digitalRead()` ;

mais ces exemples ne peuvent pas être combinés si, par exemple

- on veut gèrer plusieurs boutons et plusieurs LEDs en parallèle
- on veut que clignotement dure un certain temps **après** le dernier appui du bouton (l'appui repousse la fin du clignotement).

La raison est simple : pendant que l'arduino fait un `delay()`, il ne peut pas exécuter du code qui regarde l'état du bouton.

Donc il faudra s'y prendre autrement.

2 Le clignotement avec `delay()`

Pour mémoire, voila le genre de code qui est montré aux débutants pour faire clignoter indéfiniment une diode.

Ici on utilise la LED intégrée d'un Arduino UNO, reliée à la broche 13.

```
/*
 * Blinking using delay
 */

const byte LED_PIN = 13;

const unsigned long BLINK_DELAY = 200; // ms

void setup()
{
  pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_PIN, HIGH);
  delay(BLINK_DELAY);
  digitalWrite(LED_PIN, LOW);
  delay(BLINK_DELAY);
}
```

La constante indique la demi-période du clignotement : la LED est allumée 200 ms, puis éteinte 200 ms, etc.

Rappel : la fonction `loop()` est rappelée en boucle automatiquement. On s'occupe donc de gérer une seule période.

3 Tester le bouton et faire clignoter 10 fois (mauvaise solution)

Si on demande de prendre en compte un bouton (ici on a utilisé un simple interrupteur entre GND et la broche 2 configurée en "pull-up"), et que ça

déclenche le clignotement pendant un certaine durée, l'exemple précédent conduit généralement à ce genre de "solution" :

- Dans `loop()` on teste l'état du bouton
- si il est appuyé (état `LOW` dans notre montage), on se lance dans une boucle de clignotement.

```
/*
 * Blinking using delay
 * and button
 * The LED blinks 10 times after the button is pressed
 * Warning: the button is not checked during when the LED blinks
 */

const byte LED_PIN    = 13;
const byte BUTTON_PIN = 2;

const unsigned long BLINK_DELAY   = 200; // ms
const int           BLINK_PERIODS = 10;

void setup()
{
  pinMode(LED_PIN,    OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);

  digitalWrite(LED_PIN, LOW);
}

void loop()
{
  if (digitalRead(BUTTON_PIN) == LOW) {
    for (int i = 0; i < BLINK_PERIODS; i++) {
      digitalWrite(LED_PIN, HIGH);
      delay(BLINK_DELAY);
      digitalWrite(LED_PIN, LOW);
      delay(BLINK_DELAY);
    }
  }
}
```

Le hic, c'est que pendant la boucle `for`, l'état du bouton n'est pas testé. C'est comme une minuterie d'escalier avec laquelle il faudrait attendre que la lumière soit éteinte pour pouvoir rallumer.

Ce n'est pas une bonne solution.

4 Clignotement simple avec gestion du temps

Ici on regarde une autre façon de faire, qui se base sur la fonction `millis()`, qui indique l'heure qu'il est, ou plus exactement le nombre de millisecondes écoulées depuis le démarrage de l'arduino.

Le principe

- dans `loop()`, on regarde combien de temps s'est écoulé,
- à partir de cette durée, on calcule dans quelle demi-période de clignotement on est,
- si c'est la première (numero 0) : la LED doit être allumée, si 1 éteinte, si 2 allumée, etc : on se base sur la parité.

```
const byte LED_PIN = 13;

const unsigned long BLINK_DELAY = 200; // ms

const unsigned  START_MILLIS = millis();

void setup() {
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
}

void loop() {
  unsigned long elapsed_millis = millis() - START_MILLIS;
  unsigned long elapsed_half_periods   = elapsed_millis / BLINK_DELAY;
  if (elapsed_half_periods % 2 == 0) {
    digitalWrite(LED_PIN, HIGH);  // even => Led ON
  } else {
    digitalWrite(LED_PIN, LOW);   // odd =>  Led OFF
  }

}
```

On peut préférer la variante ci-dessous, avec une variable qui mémorise l'état de la LED, et ne fait de `digitalWrite` que quand il y a besoin de changer réellement l'état

```
/*
 * Blinking using millis()
 */
const byte LED_PIN = 13;

const unsigned long BLINK_DELAY = 200; // ms

const unsigned  START_MILLIS = millis();


int led_state = LOW;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, led_state);
}

 void loop() {
```

```
  unsigned long elapsed_millis       = millis() - START_MILLIS;
  unsigned long elapsed_half_periods = elapsed_millis / BLINK_DELAY;

  int new_led_state = elapsed_half_periods % 2 == 0
                      ? HIGH
                      : LOW;

  if (led_state != new_led_state) {
    led_state = new_led_state;
    digitalWrite(LED_PIN, led_state);
  }

}
```

5 Clignoter pendant un temps limité

Pour limiter le temps de clignotement à 5 secondes par exemple, on peut définir une constante

```
const unsigned long BLINK_LIMIT = 5000;
```

et modifier très légèrement l'expression conditionnelle qui, dans `loop()`, établit le nouvel état de la LED

```
  int new_led_state =   elapsed_millis > BLINK_LIMIT  ? LOW
                      : elapsed_half_periods % 2 == 0 ? HIGH
                      : LOW;
```

6 Prise en compte du bouton

Si on veut maintenant que

> la led clignote si et seulement si le dernier bouton a été appuyé il y a moins de 5 secondes

une solution simple est de dire que chaque appui sur le bouton définit "l'heure de fin du clignotement", variable `blink_end_millis` dans le code qui suit.

Pour avoir un clignotement sans "sauts" quand on on réappuie, une variable `blink_start_millis` remplace `START_MILLIS`, elle contient l'heure de démarrage (premier appui) de la séquence de clignotement en cours (et 0 si le clignotement n'est pas en route).

```
/*
 * Blinking using millis()
 */
const byte LED_PIN = 13;
const byte BUTTON_PIN = 2;     // used in pullup mode

const unsigned long BLINK_DELAY = 200; // ms
const unsigned long BLINK_LIMIT = 5000;
```

```
unsigned long  blink_start_millis = 0;
unsigned long  blink_end_millis = 0;


int led_state = LOW;

void setup() {
  pinMode(LED_PIN,    OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);

  digitalWrite(LED_PIN, led_state);
}

void loop() {
  unsigned long current_millis = millis();
  if (digitalRead(BUTTON_PIN) == LOW) {  // pressed
    if (blink_start_millis == 0) {
      blink_start_millis = current_millis;
    }
    blink_end_millis = current_millis + BLINK_LIMIT;
  }

  unsigned long elapsed_millis = current_millis - blink_start_millis;

  int new_led_state = LOW;
  if (current_millis > blink_end_millis) {
    blink_start_millis = 0;
  } else {
      unsigned long elapsed_half_periods = elapsed_millis / BLINK_DELAY;
      new_led_state = elapsed_half_periods % 2 == 0 ? HIGH
                    : LOW;
  }
  if (led_state != new_led_state) {
    led_state = new_led_state;
    digitalWrite(LED_PIN, led_state);
  }

}
```