Problématique de la fonction Delay

Ce que dit la documentation arduino :

delay()

Description : Pauses the program for the amount of time (in miliseconds) specified as parameter. (There are 1000 milliseconds in a second.)
Syntax : delay(ms)
Parameters : ms: the number of milliseconds to pause (unsigned long)
Returns : nothing

Un exemple pour comprendre le problème

Objectif du programme : il doit lire et fournir l'information de 2 capteurs. Le premier capteur sera lu toutes les minutes, le deuxième toutes les 3 minutes

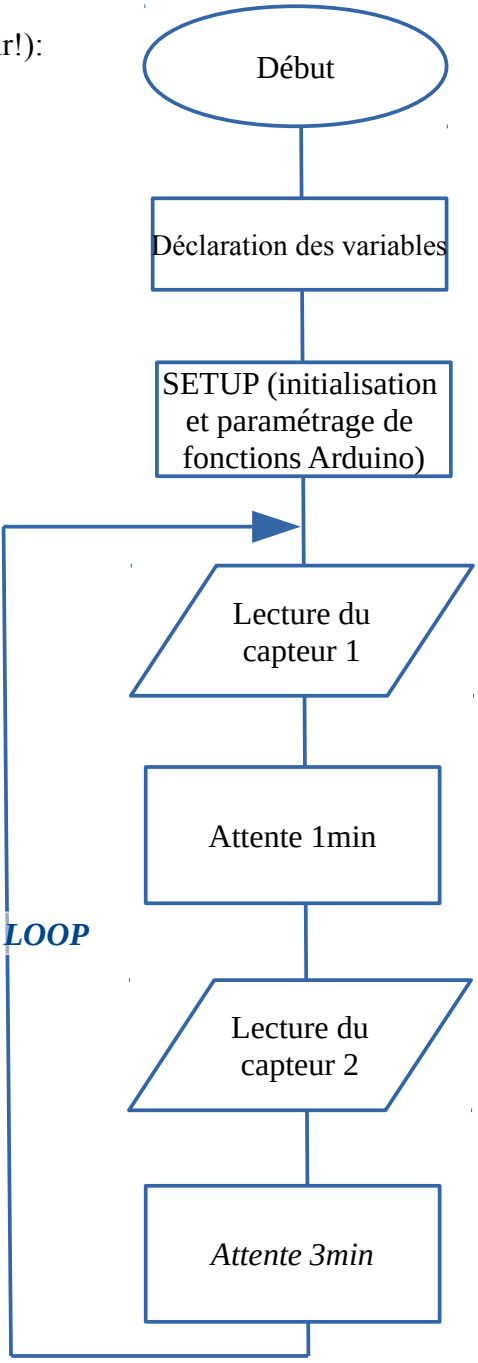
Une version d'essai du programme (programmée un peu vite, sans réfléchir!):

```
//**** debut du programme *********//
// adressage des capteurs (PIN) //
int CAPT1_PIN = A0;
int CAPT2_PIN = A1;
int lecture-capt1=0;
int lecture-capt1=0 ;

void setup()
{
 analogRead(CAPT1_PIN);
 analogRead(CAPT2_PIN);
 }

void loop()
{
 lecture-capt1 = analogRead(SLIDER1_PIN); // lecture capteur 1//
delay(60000) ;  // attente en ms
lecture-capt2 = analogRead(SLIDER1_PIN); // lecture capteur 2 //
delay(180000) ;  // attente en ms
}
```

On voit très bien que le programme ne répond pas au cahier des charges. En effet, on ne peut lire le capteur 1 toutes les minutes car à la suite du capteur 2 on est bloqué 3 minutes.

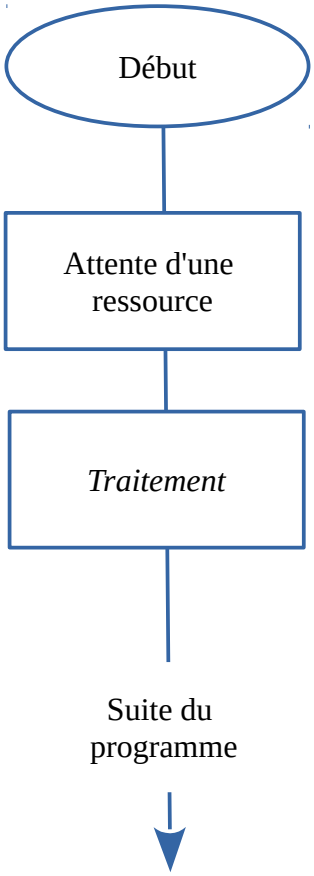
Imaginez que le programme gère la sécurité (incendie, fuite de gaz, …) ou la santé d'une personne après une opération (rythme cardiaque, …) et que pendant les 3 minutes du 2ème Delay on ne fait rien et qu'il se passe quelque chose !!!!!

Le problème devient sérieux !

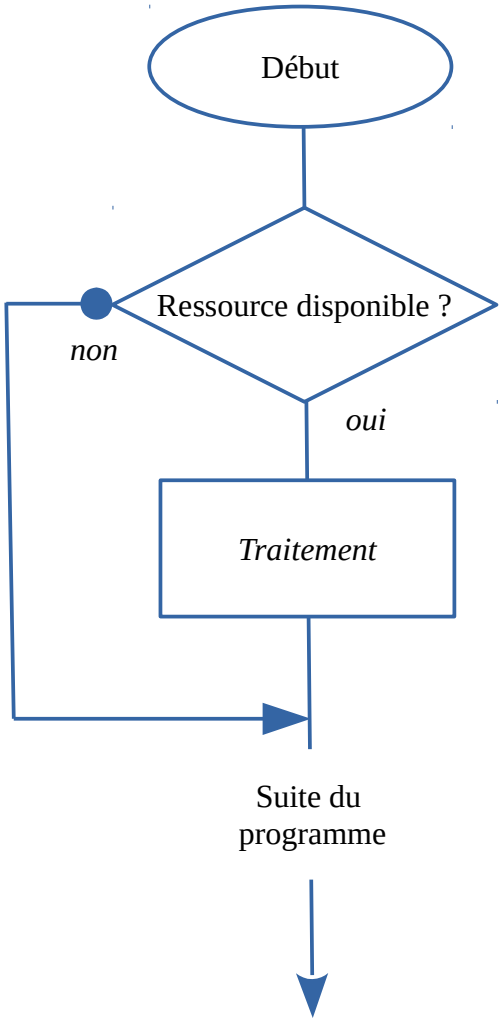


Delay est donc une fonction bloquante alors qu'il nous faudrait une fonction non bloquante :

fonction bloquante :



fonction non bloquante :



Une solution possible : la fonction "millis()"

Ce que dit la documentation arduino :

millis()

Description : Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.
Parameters : None
Returns : Number of milliseconds since the program started (unsigned long)

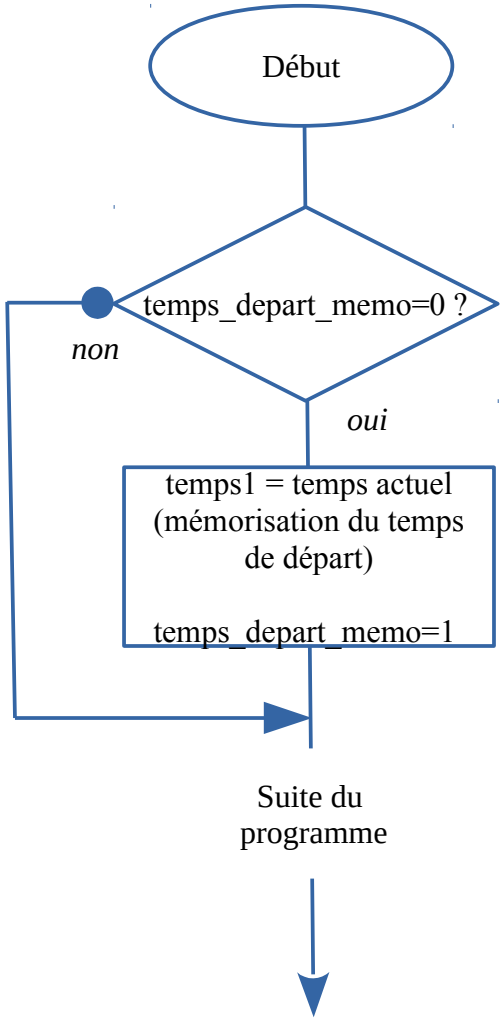
Note: please note that the return value for millis() is an unsigned long, logic errors may occur if a programmer tries to do arithmetic with smaller data types such as int's. Even signed long may encounter errors as its maximum value is half that of its unsigned counterpart.

Ce que l'on comprend :

A l'intérieur de la carte Arduino se trouve un chronomètre. Ce chronomètre mesure l'écoulement du temps depuis le lancement de l'application. Sa précision est la milliseconde. La fonction millis() nous sert à savoir quelle est la valeur courante de ce compteur. Ce compteur est capable de mesurer une durée allant jusqu'à 50 jours. La valeur retournée doit être stockée dans une variable de type « unsigned long ».

Dans la pratique, l'utilisation de la fonction millis() va nécessiter 3 étapes :

- ***étape 1*** : on mémorise le temps de référence (le temps de départ). Ce temps est le moment où l'on dit : on commence à mesurer le temps "à partir de maintenant". En général cette mémorisation se fait une fois (à chaque début de lancer du chronomètre), à un certain moment. A chaque cycle du programme, on ne va donc pas ré-écrire ce temps de départ.



Pour faire une action une seule fois, il suffit de prendre une variable (par exemple "temps_depart_memo") dont on modifiera la valeur pour dire si on a mémorisé ou non le temps de départ (par exemple 0 signifie que le temps n'est pas mémorisé et 1 signifie que le temps est mémorisé)

Remarque : pour lancer une nouvelle mesure du temps il faudra remettre à 0 la variable "temps_depart_memo"

Cela peut donner :

```
//**** extrait du programme *********//

void loop()
{

if (temps_depart_memo==0)
{
   temps1=millis() ; // mémorisation du temps de départ
   temps_depart_memo=1 ;  // variable mise à 1 =le temps est mémorisé
}
…..
…..
```

- ***étape 2***: on gère notre action en fonction du temps.

Exemple : je veux faire une action après un certain temps (exemple ici 3 minutes) :

```
//**** programme *********//

…..
…..
if (temps_depart_memo==1) // test si l'action de la temporisation est lancée
{
        temps_ecoule=milli()-temps1 ;    // calcul du temps écoulé en millisecondes
        if (temps_ecoule>180000)  // test si le temps écoulé dépasse 180000ms (3min*60*1000)
        {
                // ici mon action à exécuter
        }
}
…..
…..
```

- ***étape 3***: pour relancer une nouvelle temporisation, il faudra remettre à zéro la variable qui nous sert à faire une seule fois la mémorisation du temps de départ ("temps_depart_memo").

Une règle à retenir

En dehors des phases de test (notamment des capteurs), il ne faut pas utiliser la fonction Delay.

Cet astuce est-elle fiable ?

La fonction millis() retourne un nombre entier de taille fixe (32 bits), cela signifie qu'après un certain temps la valeur retournée pas millis() va déborder et revenir à zéro.

En effet après 11111111111111111111111111111111 (32 bits à 1 = FFFFFFFF en hexa)
il devrait y avoir 100000000000000000000000000000000
mais comme le type "unsigne byte" est sur 32 bits, le 33ème bit (à 1) n'existe pas et on obtient :
00000000000000000000000000000000 (32 zéro)

Ce ***débordement*** de millis() se produit environ 50 jours après le démarrage du programme. Pour être précis, le débordement se produit après 4 294 967 296 millisecondes, soit 49 jours, 17 heures, 2 minutes, 47 secondes et 296 millisecondes.
Si votre code est susceptible de rester en activité plus de 49 jours et que vous utilisez millis(), il va falloir trouver une solution !

Remarque : Il n'est pas nécessaire d'attendre 50 jours pour tester votre programme et mettre en évidence le bug. Il suffit de rajouter lors de la mémorisation du temps de départ une valeur proche de la valeur maximum.
Remarque 2 : le fait d'avoir utiliser dans notre exemple une soustraction temps_ecoule=milli()-temps1 peut nous amener vers une solution. Vous pouvez faire des essais avec le programme suivant en changeant les valeur de temps_actuel et temps_départ

Exemple de programme de test à faire (ici fonctionnement normal car le temps_actuel > temps_depart):

```
unsigned long temps_actuel=0xFFFFFFFF;  // valeur en hexadecimal
unsigned long temps_depart=0xFFFFFFFA;    // valeur en hexadecimal
unsigned long resultat;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
   resultat=temps_actuel-temps_depart;
   Serial.println(resultat);
}
```

Faire d'autres tests en simulant un temps supérieur à 50 heures (temps_actuel repasse à 00000000h) pour voir si ça fonctionne et si ce n'est pas le cas il faut trouver une solution.