Fonctions millis

Reference arduino : millis() EN

Description

Renvoie le nombre de millisecondes écoulées depuis que la carte Arduino a commencé à exécuter le programme en cours. Ce nombre débordera (reviendra à zéro) après environ 50 jours.

Notes et avertissements

- La valeur de retour de millis() est de type unsigned long, des erreurs logiques peuvent se produire si un programmeur essaie d'effectuer des opérations arithmétiques avec des types de données plus petits tels que int. Même les valeurs signées longpeuvent rencontrer des erreurs car leur valeur maximale est la moitié de celle de leur équivalent non signé.
- millis() est incrémenté (pour les puces AVR 16 MHz et certaines autres) toutes les 1,024 millisecondes, puis incrémenté de 2 (au lieu de 1) tous les 41 ou 42 ticks, pour le remettre en synchronisation ; ainsi, certaines valeurs millis() sont ignorées. Pour un timing précis sur de courts intervalles, pensez à utiliser micros().
- **millis() reviendra à 0 après environ 49 jours (micros en environ 71 minutes).**
- La reconfiguration des temporisateurs du microcontrôleur peut entraîner millis()des lectures inexactes. Les cœurs « Arduino AVR Boards » et « Arduino megaAVR Boards » utilisent Timer0 pour générer des millis(). Les cœurs « Arduino ARM (32 bits) Boards » et « Arduino SAMD (32 bits ARM Cortex-M0+) Boards » utilisent le temporisateur SysTick.

Gérer le temps avec la fonction millis()

1. Introduction :

Lors de la découverte de la programmation de l'Arduino, la principale fonction que nous apprenons pour créer des temporisations est la fonction delay().

Hors celle-ci a ses limites car elle met en pause la suite du code et cela peut vite devenir contraignant lorsque l'on souhaite exécuter plusieurs tâches à la fois.

Pour palier ce problème, une solution est possible : utiliser la fonction millis().

Dans ce tutorial nous allons découvrir comment utiliser celle-ci pour remplacer la fonction delay() dans différentes applications avec différents exemples pour mieux comprendre la fonction millis().

2. Description de la fonction millis()

millis() est un compteur qui est mis à jour en permanence et qui renvoie une valeur qui va

représenter le nombre de millisecondes écoulées depuis la mise sous tension de la carte.

Ce compteur millis() est automatiquement incrémenté par une interruption attachée au temporisateur 0. La valeur qu'il renvoie augmente constamment sans être influencée par le code de l'utilisateur.

La fonction millis() renvoie une variable de type non signée « unsigned long ».

Vu que la valeur maximale de ce type de variable est de 4,294,967,295, elle se réinitialisera aux bouts de 49 jours.

Voici un petit morceau de code pour afficher ce compteur dans la console de l'IDE Arduino.

millis020.ino

```
// Déclaration variable ValeurMillis qui va servir à stocker une valeur au format unsigned long
unsigned long ValeurMillis;

void setup() {
  // ouvre le port série à 9600 bps
  Serial.begin(9600);
}

void loop() {
  // La variable ValeurMillis prend la valeur de millis()
  ValeurMillis = millis();
  // imprimer sous forme de nombre décimal codé en ASCII - identique à "DEC" la valeur de ValeurMillis.
  // puis ajoute le retour à la ligne avec "println"
  Serial.println(ValeurMillis);
}
```

3. Exemples 3.1 Définir différents intervalles de temps

Imaginons que nous avons plutôt besoin d'allumer la LED que 100 ms mais avec un intervalle de 1 seconde.

Voici un petit exemple de code que vous pouvez utiliser :

millis031.ino

```
// La broche numérique 2 est reliée à la led rouge. On lui donne le nom Ledrouge.
const int Ledrouge = 2;
// Déclaration variable ledState qui va servir à stocker une valeur au format bool soit LOW ou HIGH.
bool ledState;
// Déclaration variable previousMillis qui va servir à stocker une valeur au format unsigned long.
```

```
unsigned long previousMillis = 0;
// Déclaration variable interval qui va servir à stocker une valeur au format unsigned int.
unsigned int interval;
// Déclaration variable interval1 qui va servir à stocker une valeur au format unsigned int.
// On lui donne la valeur de 1000 qui correspondra à intervalle 1 de clignotement (millisecondes)
unsigned int interval1 = 1000;
// Déclaration variable interval2qui va servir à stocker une valeur au format unsigned int.
// On lui donne la valeur de 100 qui correspondra à intervalle 2 de clignotement (millisecondes)
unsigned int interval2 = 100;

void setup() {
  // Définit Ledrouge comme sortie.
  pinMode(Ledrouge, OUTPUT);
}

void loop() {
   // Lit la valeur millis() et stock ça valeur dans currentMillis au format unsigned long
  unsigned long currentMillis = millis();
  // Si currentMillis - previousMillis >= interval on exécute les actions entre {}
  if (currentMillis - previousMillis >= interval)
  {
    // Stock la valeur de currentMillis dans la variable previousMillis
    previousMillis = currentMillis;
    // Si ledState == LOW on exécute les actions entre {}
    if (ledState == LOW) {
      // La variable ledState prend la valeur de HIGH
      ledState = HIGH;
      // Stock la valeur de interval2 dans interval
      interval = interval2;
    }
    //Sinon  on exécute les actions entre {}
    else {
      // La variable ledState prend la valeur de LOW
      ledState = LOW;
      // Stock la valeur de interval1 dans interval
      interval = interval1;
    }
    // Met la broche numérique stockée dans Ledrouge soit 2 à la valeur de ledState
    digitalWrite(Ledrouge, ledState);
  }
}
```

3.2 Définir plusieurs intervalles de temps

Imaginons que nous souhaitons faire clignoter 2 LED mais à différents moment avec un intervalle de temps différents.

Voici un petit exemple de code que vous pouvez utiliser :

millis032.ino

```
//////////////////Led rouge //////////////////////
// La broche numérique 2 est reliée à la led rouge. On lui donne le nom Ledrouge.
const int Ledrouge = 2;
// Déclaration variable ledState qui va servir à stocker une valeur au format bool soit LOW ou HIGH.
bool LedrougeState;
// Déclaration variable LedrougepreviousMillis qui va servir à stocker une valeur au format unsigned long.
unsigned long LedrougepreviousMillis = 0;
// Déclaration variable Ledrougeinterval qui va servir à stocker une valeur au format unsigned int.
unsigned int Ledrougeinterval;
// Déclaration variable Ledrougeinterval1 qui va servir à stocker une valeur au format unsigned int.
// On lui donne la valeur de 1000 qui correspondra à intervalle 1 de clignotement (millisecondes)
unsigned int Ledrougeinterval1 = 1000;
// Déclaration variable Ledrougeinterval2 qui va servir à stocker une valeur au format unsigned int.
// On lui donne la valeur de 100 qui correspondra à intervalle 2 de clignotement (millisecondes)
unsigned int Ledrougeinterval2 = 100;

//////////////////Led verte //////////////////////
const int Ledverte = 3;
bool LedverteState;
unsigned long LedvertepreviousMillis = 0;
unsigned int Ledverteinterval;
unsigned int Ledverteinterval1 = 2000;
unsigned int Ledverteinterval2 = 500;

void setup() {
  // Définit Ledrouge comme sortie.
  pinMode(Ledrouge, OUTPUT);
  // Définit Ledverte comme sortie.
  pinMode(Ledverte, OUTPUT);
}

void loop() {
   // Lit la valeur millis() et stock ça valeur dans currentMillis au
```

```
format unsigned long
  unsigned long currentMillis = millis();

  ////////////////Led rouge ////////////////////
  // Si currentMillis - LedrougepreviousMillis >= Ledrougeinterval on
exécute les actions entre {}
  if (currentMillis - LedrougepreviousMillis >= Ledrougeinterval)
  {
    // Stock la valeur de currentMillis dans la variable
LedrougepreviousMillis
    LedrougepreviousMillis = currentMillis;
    // Si LedrougeState == LOW on exécute les actions entre {}
    if (LedrougeState == LOW) {
      // La variable LedrougeledState prend la valeur de HIGH
      LedrougeState = HIGH;
      // Stock la valeur de Ledrougeinterval2 dans interval
      Ledrougeinterval = Ledrougeinterval2;
    }
    //Sinon  on exécute les actions entre {}
    else {
      // La variable LedrougeState prend la valeur de LOW
      LedrougeState = LOW;
      // Stock la valeur de Ledrougeinterval1 dans interval
      Ledrougeinterval = Ledrougeinterval1;
    }
    // Met la broche numérique stockée dans Ledrouge soit 2 à la valeur
de LedrougeState
    digitalWrite(Ledrouge, LedrougeState);
  }

  ////////////////Led verte ////////////////////
  if (currentMillis - LedvertepreviousMillis >= Ledverteinterval)
  {
    LedvertepreviousMillis = currentMillis;
    if (LedverteState == LOW) {
      LedverteState = HIGH;
       Ledverteinterval = Ledverteinterval2;
    }
    else {
      LedverteState = LOW;
      Ledverteinterval = Ledverteinterval1;
    }
    digitalWrite(Ledverte, LedverteState);
  }
}
```

autre exemple

millisautre001.ino

```
int ledPin = 2;
unsigned long previousMillis = 0; // Store the time of the last LED
toggle

void setup() {
  pinMode(ledPin, OUTPUT); // Set the LED pin as an output
}

void loop() {
  unsigned long currentMillis = millis(); // Get the current time
  if (currentMillis - previousMillis < 500) {
    digitalWrite(ledPin, HIGH);
    Serial.println("LED is ON");
    Serial.println("LED is ON");
    Serial.println("LED is ON");
  } else if (currentMillis - previousMillis < 1000) {
    digitalWrite(ledPin, LOW);
    Serial.println("LED is OFF");
    Serial.println("LED is OFF");
    Serial.println("LED is OFF");
} else{
    previousMillis = currentMillis;
}
}
```