



Les entrées, sorties numériques

[Mise à jour le 26/4/2021]



3. Programmation

3.5 Temporisations

- **Sources** : [delay](#), [delayMicroseconds](#), [micros](#) et [millis](#) sur le site arduino.

3.5.1 Temporisation bloquante

- de **n millisecondes** avec la fonction : `void delay(int millisecondes)`

*.cpp

```
// Le code met le programme en pause pendant une seconde avant de faire basculer l'état logique de la broche.
```

```
int ledPin = 13;           // LED connected to digital pin 13

void setup() {
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                // waits for a second
  digitalWrite(ledPin, LOW);  // sets the LED off
  delay(1000);                // waits for a second
}
```

- de **n microsecondes** avec la fonction : `void delayMicroseconds(int microsecondes)` (Erreur = +/-3µs)

*.cpp

```
// Le code configure la broche numéro 8 en sortie.
```

```
// Il envoie un train d'impulsions d'une période d'environ 100
microsecondes.
// L'approximation est due à l'exécution des autres instructions.

int outPin = 8;           // digital pin 8

void setup() {
  pinMode(outPin, OUTPUT); // sets the digital pin as output
}

void loop() {
  digitalWrite(outPin, HIGH); // sets the pin on
  delayMicroseconds(50);      // pauses for 50 microseconds
  digitalWrite(outPin, LOW);  // sets the pin off
  delayMicroseconds(50);      // pauses for 50 microseconds
}
```

3.5.2 Temporisation non bloquante

- **Temps écoulé**, en **millisecondes**, depuis le démarrage du programme avec la fonction :
unsigned long **millis()**

millis() renvoie le nombre de millisecondes écoulées depuis que la carte Arduino a commencé à exécuter le programme. Ce nombre va déborder (revenir à zéro), après environ 50 jours.

- *Structure d'une temporisation non bloquante*

*.cpp

```
// Répétition d'un bloc de code tous les intervalles de temps sans
bloquer le programme
unsigned long previousMillis = 0; // Mémoire le temps écoulé depuis
la dernière mesure
const long interval = 1000;      // Durée non bloquante souhaitée

void setup() {
  // Le contenu de setup dépend de l'application
}

void loop() {
  // Mesure du temps écoulé depuis que le programme s'exécute
  unsigned long currentMillis = millis();

  // Si l'attente souhaitée est atteinte on exécute le bloc de code
  if (currentMillis - previousMillis >= interval) {
    // la mesure de temps actuelle devient la dernière mesure
  }
}
```

```
previousMillis = currentMillis;
//
// Bloc de code à exécuter tous les intervalles de temps
//
}
//
// sinon d'autres actions sont possibles
//
}
```

- **Temps écoulé**, en **microsecondes**, depuis le démarrage du programme avec la fonction : *unsigned long **micros()***

micros() renvoie le nombre de microsecondes depuis que la carte Arduino a commencé à exécuter le programme. Ce nombre va déborder (revenir à zéro), après environ 70 minutes. Sur les cartes Arduino 16 MHz (par exemple Duemilanove et Nano), cette fonction a une résolution de quatre microsecondes (c'est-à-dire que la valeur renvoyée est toujours un multiple de quatre).

*.cpp

```
unsigned long time;

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

void loop() {
  Serial.print("Time: ");
  time = micros();

  Serial.println(time);
  delay(1000);
}
```

3.6 Interruption

3.6.1 Attacher une interruption

- **Source** : [attachInterrupt](#) sur le site Arduino.
- **Syntaxe**

*.cpp

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)
```

- **ISR** : gestionnaire d'interruption à appeler lorsque l'interruption se produit
- **pin** : broche de la carte (concernée par l'interruption)
- **mode** : définit le moment de déclenchement de l'interruption
 - **Paramètres**
 - **LOW** déclenche l'interruption chaque fois que la broche est à l'état bas,
 - **CHANGE** déclenche l'interruption chaque fois que la broche change de valeur,
 - **RISING** déclenche l'interruption sur un front montant,
 - **FALLING** déclenche l'interruption sur un front descendant.
- *Exemple*

*.cpp

```
// Le front montant du signal présent sur la broche 3 de la carte
// Arduino Uno
// déclenche l'exécution du sous-programme blink

const byte ledPin = 11; // 00 du shield Tinkerkit
const byte interruptPin = 3; // 05 du shield Tinkerkit
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```



Le projet **PlatformIO** pour l'IDE **VSCode** de l'exemple ci-dessus est téléchargeable [ici](#)

3.6.2 Détacher une interruption

La fonction `detachInterrupt()` désactive l'interruption donnée.

- **Source** : [detachInterrupt](#) sur le site Arduino.
- **Syntaxe**

*.cpp

```
detachInterrupt(digitalPinToInterrupt(pin))
```

From:

<http://webge.fr/dokuwiki/> - **WEBGE Wikis**

Permanent link:

<http://webge.fr/dokuwiki/doku.php?id=microc:arduino:esnumeriques>

Last update: **2023/08/20 11:03**

