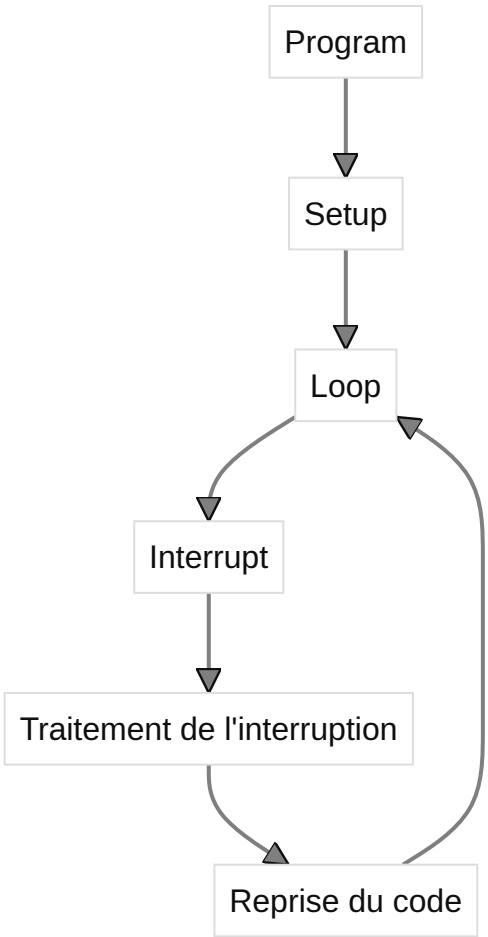
Arduino - Les interruptions

Les interruptions

Les interruptions sur Arduino sont des événements qui interrompent momentanément le programme principal exécuté pour effectuer un autre travail. Quand cet autre travail est terminé, on retourne à l'exécution du programme à l'endroit exact où il l'avait été laissé.

Lorsqu'une interruption est déclenchée, le programme principal est temporairement interrompu et une fonction d'interruption est exécutée. Cette fonction est appelée une routine d'interruption et elle est programmée par l'utilisateur. Une fois que la routine d'interruption est terminée, le programme principal reprend là où il s'était arrêté



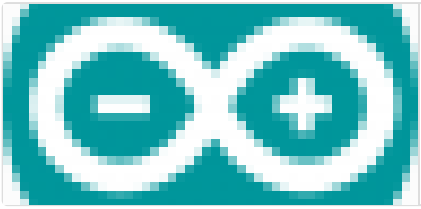
Les déclencheurs

Un déclencheur d'interruption en Arduino est un événement spécifique qui provoque l'interruption du flux d'exécution normal du programme. Lorsque le déclencheur d'interruption se produit, le microcontrôleur arrête temporairement l'exécution du code principal et saute vers une routine d'interruption spécifique pour gérer cet événement.

Une fois la routine d'interruption terminée, le microcontrôleur revient à l'exécution du code principal à l'endroit où il s'était arrêté.

les types de déclencheur:

- **Interruptions externes** (External Interrupts) : Ces interruptions sont déclenchées par des changements d'état sur des broches spécifiques. Les microcontrôleurs Arduino tels que l'Arduino Uno disposent généralement de deux ou trois broches dédiées aux interruptions externes.
- **Interruptions de minuterie** (Timer Interrupts) : Les interruptions de minuterie sont déclenchées par les compteurs internes du microcontrôleur. Les minuteries peuvent être configurées pour déclencher des interruptions après un certain délai ou à intervalles réguliers.
- **Interruptions de transmission série** (Serial Interrupts) : Certaines bibliothèques de communication série permettent de configurer des interruptions lors de la réception ou de l'envoi de données série. Cela permet de traiter les données en temps réel sans attendre dans la boucle principale.
- **Interruptions analogiques** (Analog Comparator Interrupts) : Certains microcontrôleurs Arduino disposent d'un comparateur analogique qui peut générer des interruptions lorsqu'une valeur analogique dépasse un seuil spécifié.



attachInterrupt() - Arduino Reference

The Arduino programming language Reference, organized into Functions, Variable and Constant, and Structure keywords.
https://www.arduino.cc/reference/en/language/functions/external-…

les paramètres pour les interruptions

Lors de la configuration d'un déclencheur, il faudra déclare sous quelle condition cela est valide, par exemple, le passage de 0 à 1 sur la broche d'un bouton.

Mode	Description
LOW	Interruption déclenchée lorsque le signal est bas
CHANGE	Interruption déclenchée lorsque le signal change
RISING	Interruption déclenchée lorsque le signal passe de bas à haut
FALLING	Interruption déclenchée lorsque le signal passe de haut à bas

Création d'une interruption

Pour configurer une interruption sur Arduino, voici les étapes générales à suivre :

1. Sélectionnez la broche appropriée : Choisissez la broche sur laquelle vous souhaitez activer l'interruption. Assurez-vous de sélectionner une broche

supportant les interruptions, car toutes les broches ne sont pas capables de le faire.

2. Choisissez le type d'interruption : Il existe différents types d'interruptions, tels que les interruptions de changement d'état (rising, falling, ou change), les interruptions de minuterie, les interruptions de transmission série, etc.

3. Définissez la routine d'interruption : Écrivez une fonction spéciale appelée "ISR" (Interrupt Service Routine) qui sera exécutée lorsque l'interruption se produit. Cette fonction doit être définie avec le type de données `void` et ne doit pas prendre d'arguments.

4. Activez l'interruption : Dans la fonction `setup()`, utilisez les fonctions appropriées pour activer l'interruption sur la broche choisie et spécifiez le type d'interruption (par exemple `attachInterrupt()`).

5. Écrivez la logique de votre programme : Dans la routine d'interruption, définissez le code qui sera exécuté lorsque l'interruption se produit. Ce code doit être concis et rapide, car il est exécuté au milieu de l'exécution du code principal.

Voici un exemple de code pour activer une interruption de changement d'état sur la broche 2 de l'Arduino Uno :

C++

```
void setup() {
  pinMode(2, INPUT);  // Configure la broche 2 en entrée
  attachInterrupt(digitalPinToInterrupt(2), isrFunction,
CHANGE);  // Active l'interruption sur la broche 2 avec le
type CHANGE
}

void loop() {
  // Code principal
}

void isrFunction() {
  // Code à exécuter lors de l'interruption
}
```

Dans cet exemple, la fonction `isrFunction()` sera appelée chaque fois qu'il y aura un changement d'état (*rising* ou *falling*) sur la broche 2. Vous pouvez remplacer `CHANGE` par `RISING` pour détecter uniquement les fronts montants, ou par `FALLING` pour détecter uniquement les fronts descendants.

Compatibilité des broches

Toutes les pins ne supporte pas les interruptions externe voici un récapitulatif provenant du site officiel arduino:

Board	Digital Pins Usable For Interrupts
Uno, Nano, Mini, other 328-based	2, 3
Uno WiFi Rev.2, Nano Every	all digital pins
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21 (**pins 20 & 21** are not available to use for interrupts while they are used for I2C communication)
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	all digital pins, except 4
MKR Family boards	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Nano 33 IoT	2, 3, 9, 10, 11, 13, A1, A5, A7
Nano 33 BLE, Nano 33 BLE Sense	all pins
Due	all digital pins
101	all digital pins (Only pins 2, 5, 7, 8, 10, 11, 12, 13 work with **CHANGE**)

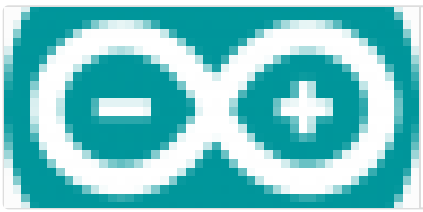
Exemple

Changement d'état d'une DEL à chaque appui sur un bouton:

```c
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  //Configuration de l'interruption sur un changement d'état
de la pin 2
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink,
CHANGE);
}

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

//Déclaration de la fonction d'interruption
//celle-ci sera éxécuté à chaque fois que la condition est
remplie
void blink() {
  state = !state;
}
```

Mise en pause des interruptions

L'exécution des interruptions peut être suspendu avec les fonctions: **interrupts()** et **noInterrupts()**



interrupts() - Arduino Reference

The Arduino programming language Reference, organized into Functions, Variable and Constant, and Structure keywords.
https://www.arduino.cc/reference/en/language/functions/interrupts/in…

```c
void setup() {}

void loop() {
  noInterrupts();
  // critical, time-sensitive code here
  interrupts();
  // other code here
}
```