

Les interruptions

Une interruption, comme son nom l'indique, consiste à interrompre momentanément le programme que l'Arduino exécute pour qu'il effectue un autre travail. Quand cet autre travail est terminé, l'Arduino retourne à l'exécution du programme et reprend à l'endroit exact où il l'avait laissé.

Les broches d'interruptions

Sur un Arduino Uno basé sur microcontrôleur AVR 328P

INT0 Interruption externe sur la broche 2

INT1 Interruption externe sur la broche 3

0 ou 1 sur un Arduino Uno, ce qui correspond respectivement aux broches 2 et 3.

Sur un Arduino Mega basé sur microcontrôleur AVR 2560

0 à 5 sur un Arduino Mega ce qui correspond, dans l'ordre, aux broches 21, 20, 19, 18, 2 et 3.

Sur un ESP32

L'ESP32 dispose de 26 broches numériques qui peuvent être utilisées pour déclencher l'exécution d'une fonction à l'aide d'une interruption externe

[Interruptions sur ESP32](#)

Les interruptions : Syntaxe

[methodespourlesinterruptions](#)

```
digitalPinToInterrupt (broche); // convertit un identifiant de broche
en identifiant d'interruption,
                                // à utiliser avec attachInterrupt() et
                                detachInterrupt() .

attachInterrupt (digitalPinToInterrupt (pin), ISR, mode); // conseillé
attachInterrupt (interruption, ISR, mode); // non recommandé
detachInterrupt (digitalPinToInterrupt (pin));
detachInterrupt (interruption);
```

```
noInterrupts (); // désactive les interruptions

interrupts (); // réactiver les interruptions après l'
noInterrupts() de noInterrupts() .
```

Paramètres	remarques
interruption	Identifiant de l'interruption. Ne doit pas être confondu avec le numéro d'identification.
ISR	Routine de service d'interruption. C'est la méthode qui sera exécutée lorsque l'interruption se produira.
mode	Qu'est-ce qui devrait provoquer le déclenchement de l'interruption? ? Etat LOW, CHANGE, RISING, ou FALLING. Les cartes DUE permettent aussi HIGH

Remarques

Les routines de service d'interruption (ISR) **doivent être aussi courtes que possible**, car elles mettent en pause l'exécution du programme principal et peuvent donc vider le code en fonction du temps. Généralement, cela signifie que dans l'ISR, vous définissez un drapeau et sortez, et dans la boucle du programme principal, vous vérifiez le drapeau et faites ce que ce drapeau est censé faire.

Vous ne pouvez pas utiliser `delay()` ou `millis()` dans un ISR car ces méthodes elles-mêmes reposent sur des interruptions. Interruption sur le bouton presse

Programmes d'exemples

Exemple 1 BP avec Rebonds

Cet exemple utilise un bouton-poussoir (commutateur tactile) connecté à la broche numérique 2 et à la masse, en utilisant une résistance de rappel interne pour que la broche 2 soit haute lorsque le bouton n'est pas enfoncé.

[interrupBP.ino](#)

```
const int LED_PIN = 13;
const int INTERRUPT_PIN = 2;
volatile bool ledState = LOW;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(INTERRUPT_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), myISR,
  FALLING); // trigger when button pressed, but not when released.
}

void loop() {
  digitalWrite(LED_PIN, ledState);
}
```

```
void myISR() {
    ledState = !ledState;
    // note: LOW == false == 0, HIGH == true == 1, so inverting the
    // boolean is the same as switching between LOW and HIGH.
}
```

Avec cet exemple simple, les boutons-poussoirs ont tendance à rebondir, ce qui signifie que le circuit s'ouvre et se ferme plus d'une fois avant de s'établir dans l'état final fermé ou ouvert.

Cet exemple ne prend pas cela en compte. Par conséquent, il suffit parfois d'appuyer sur le bouton pour basculer le voyant plusieurs fois, au lieu d'une fois.

exemple 2 sur les broches A1 A2 et A3

[brochesA1A2A3eninterruption.ino](#)

```
//ce code configure les broches A1,A2 et A3 en interruption.
//ont utilise la bibliotheque PinChangeInt.h
//ont peut egalement le faire avec les broches digitale.
#include <PinChangeInt.h>//appelle de la fonction
const byte B1 = A1;
const byte B2 = A2;//Déclaration des boutons
const byte B3 = A3;

volatile int led1=2;
volatile int led2=3;// Déclaration des LED
volatile int led3=4;

volatile bool bascule = false;//Cette variable enregistre l'état d'un
bouton
volatile int i;// Cette variable permet de savoir si un bouton à été
actionner

int f1(){
    bascule = !bascule;
    i = 1;
    if(bascule){
        digitalWrite(led1,HIGH);
    }
    else{
        digitalWrite(led1,LOW);
    }
}

int f2(){
    bascule = !bascule;
    i = 1;
```

```
    if(bascule){
        digitalWrite(led2,HIGH);
    }
    else{
        digitalWrite(led2,LOW);
    }
}

int f3(){
    bascule = !bascule;
    i = 1;
    if(bascule){
        digitalWrite(led3,HIGH);
    }
    else{
        digitalWrite(led3,LOW);
    }
}

void setup() {// fonction d'initialisation des variables
    Serial.begin(9600);
    pinMode(B1,INPUT);
    pinMode(B2,INPUT);
    pinMode(B3,INPUT);

    PCintPort::attachInterrupt(B1, f1, FALLING);
    PCintPort::attachInterrupt(B2, f2, FALLING);
    PCintPort::attachInterrupt(B3, f3, FALLING);
}

void loop() {
    if(i==1){
        delay(5);
        i=0;
    }else{i=0}
}
```

Exemple 3 avec un bouton poussoir pour passage piétons

[FeuxBP.ino](#)

```
const int vert = 10; // Broche 10 pour le voyant vert
const int orange = 11;// Broche 11 pour le voyant orange
const int rouge = 12;// Broche 12 pour le voyant rouge
const int BP1_Pietons = 2;// Broche 2 pour le Bouton piétons
const int RougePietons = 9;// Broche 9 pour le voyant Rouge Piétons
```

```
const int VertePietons = 8; // Broche 8 pour le voyant Vert Pietons
const int delais1s = 1000; // Defini un delai de 1 s ==> 1000
millisecondes
const int delais3s = 3000; // Defini un delai de 3 s ==> 3000
millisecondes
const int delais5s = 5000; // Defini un delai de 5 s ==> 5000
millisecondes
volatile int etatBP1 = LOW;

void setup() {
  pinMode(vert, OUTPUT); // Definit les broches des Voyants en sortie
  pinMode(orange, OUTPUT);
  pinMode(rouge, OUTPUT);
  pinMode(RougePietons, OUTPUT);
  pinMode(VertePietons, OUTPUT);
  pinMode(BP1_Pietons, INPUT); // Definit la broche Bouton pietons en
  entrée
  digitalWrite(vert, LOW); // Initialise tous les voyants eteint pour
  demarrer
  digitalWrite(orange, LOW);
  digitalWrite(rouge, LOW);
  digitalWrite(VertePietons, LOW);
  digitalWrite(RougePietons, LOW);
  attachInterrupt(0, memoireBP, CHANGE);
}

void memoireBP(){
  etatBP1 = 1;
}

void loop() {
  digitalWrite(VertePietons, LOW); // eteint le voyant vert pietons
  digitalWrite(RougePietons, HIGH); // allume le voyant rouge pietons
  digitalWrite(vert, HIGH); // allume le voyant vert voiture
  delay(delais3s); // pendant 3 secondes
  digitalWrite(vert, LOW); // eteint le voyant vert voiture
  // if (digitalRead(BP1_Pietons) == 1 ){ // test si le bouton pietons
  est appuyé pendant que le voyant vert voiture est allumé
  if (etatBP1 == 1){
    digitalWrite(orange, HIGH); // si OUI fait la sequence pietons
    --> orange voiture allumé
    delay(delais1s); // pendant 1 seconde
    digitalWrite(orange, LOW); // voyant orange voiture eteint
    digitalWrite(rouge, HIGH); // voyant rouge voiture allumé
```

```
digitalWrite(RougePietons,LOW);// voyant rouge pietons eteint
digitalWrite(VertePietons,HIGH);// voyant vert pietons allumé
delay(delais5s); // pendant 5 secondes
digitalWrite(VertePietons, LOW); // eteint voyant vert pietons
digitalWrite(RougePietons, HIGH);// allume voyant rouge
pietons
digitalWrite(rouge, LOW); // voyant rouge voiture eteint
etatBP1 = 0;
} // et l'on recommence la boucle au debut

else { // si NON = Bouton pietons non appuyé pendant le voyant vert
voiture allumé
digitalWrite(orange, HIGH);// sequence normale des feux voiture -->
Orange voiture allumé
delay(delais1s);// pendant 1 seconde
digitalWrite(orange, LOW);// voyant orange voiture eteint
digitalWrite(rouge, HIGH);// voyant rouge voiture allumé
delay(delais3s);// pendant 3 secondes
digitalWrite(rouge, LOW); // voyant rouge voiture eteint
}
// et l'on recommence la boucle au debut
}
```

Liens Web avec les interruptions

[Sur Locoduino et ILS](#)

[Autre Exemple](#)

From:
<http://chanterie37.fr/fablab37110/> - Castel'Lab le Fablab MJC de Château-Renault

Permanent link:
<http://chanterie37.fr/fablab37110/doku.php?id=start:arduino:interruptions&rev=1606926484>

Last update: **2023/01/27 16:08**

