# Projet 6: L'incrémentation

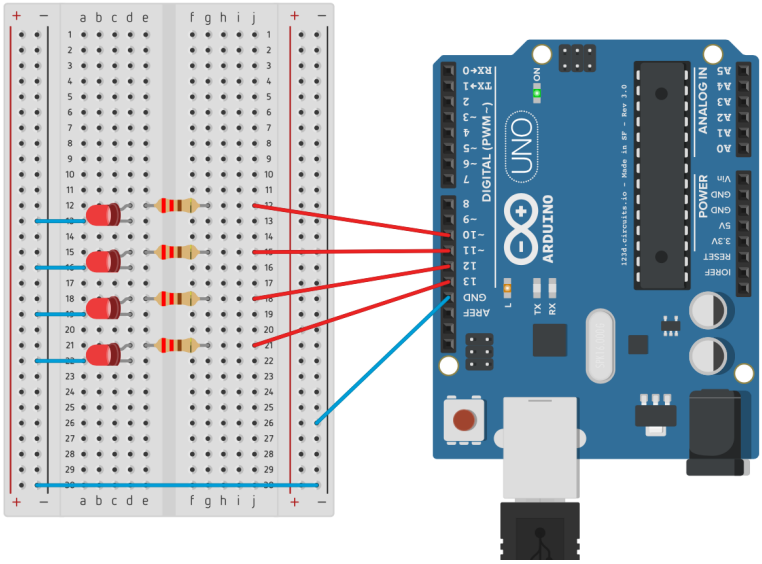
Il est possible d'appliquer aux valeurs des variables diverses opérations mathématiques. Commençons tout de suite par un petit exemple: *l'incrémentation*. Il s'agit simplement d'additionner 1 à la variable. A chaque fois qu'on répète le code, on prend la valeur de la variable et on y ajoute 1.

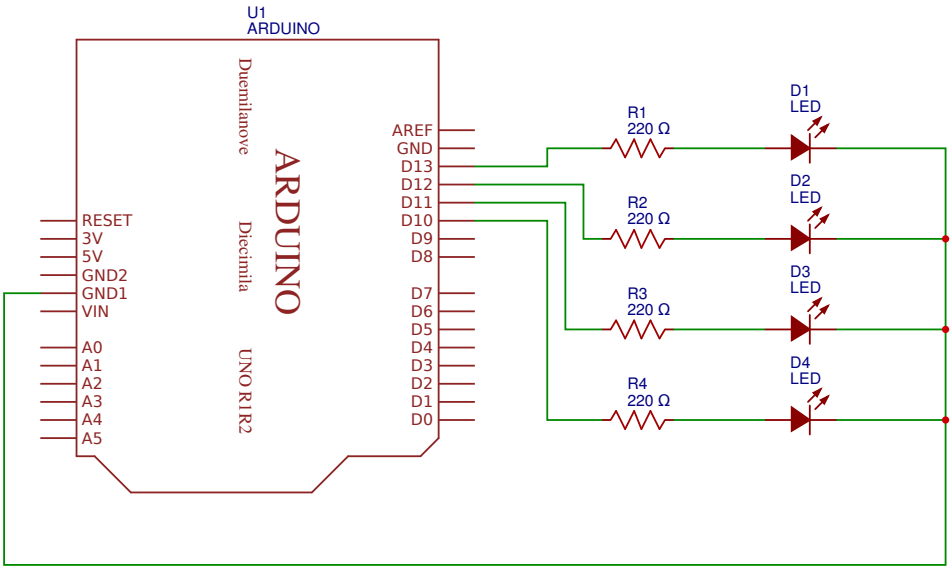
Cela se fait grâce à ce code (*var* étant une variable à choix): *var++;*

*var++;* revient à écrire : "var = var + 1;".

Le circuit sera des plus classiques: on va reprendre notre chenillard.

## Circuit 5






## Liste des composants:

- 4 LEDs rouges
- 4 résistances de 220 à 470Ω

# Code 5: faire clignoter 10 fois la LED 13

```
/*
Code 5 - Edurobot.ch, destiné à l'Arduino
Objectif: faire clignoter 10 fois la LED montée sur le port 13
*/

//***** EN-TETE DECLARATIVE *****
// On déclare les variables, les constantes...

byte compteur;                  //On définit la variable "compteur"
const int led1= 13;     //On renomme la broche 10 en "led1"


//***** FONCTION SETUP = Code d'initialisation *****
// La fonction setup() est exécutée en premier et une seule fois, au démarrage du programme

void setup()

{
  pinMode(led1, OUTPUT);       // Initialise la broche 13 comme sortie

  Serial.begin(9600);          // Ouvre le port série à 9600 bauds

// Exécute le programme entre accolades en partant de zéro et en incrémentant à chaque fois la
valeur de +1: 0+1/2+1/3+1... jusqu'à ce que la variable "compteur" soit égale à 9 (plus petit
que 10).

for(compteur=0 ; compteur<10 ; compteur++)

  {                             // début du programme exécuté 10 fois
    digitalWrite(led1, HIGH); // allume la LED
    delay(500);               // attend 500ms
    digitalWrite(led1, LOW);  // éteint la LED
    delay(500);               // attend 500ms
  }
}                       // fin du programme exécuté 10 fois

void loop() {
}                       // vide, car programme déjà exécuté dans setup
```

## Liens

**Télécharger le code**

http://arduino.education/codes/code5.zip

## Analyse du code[7]

Revenons à notre code et analysons-le.

La ligne *byte compteur;* permet de créer une variable appelée *compteur*. *Byte* indique le type de la variable, c'est-à-dire le type de données que l'on pourra stocker dans cette variable. Comme nous l'avons déjà vu, le type byte permet de stocker des valeurs comprises entre 0 et 255.

La ligne const int *led1= 13;* permet de créer une constante (une variable) nommée *led1* dans laquelle on stocke la valeur 13.

La ligne *for(compteur=0 ; compteur<10 ; compteur++)* sert à faire varier la variable compteur de 0 à 9 (en l'augmentant à chaque fois de 1: c'est ce que fait l'instruction *compteur++*)

Regardons cette ligne d'un peu plus près:

*for(compteur=0 ; compteur<10 ; compteur++)*

La déclaration *for* est habituellement utilisée pour répéter un bloc d'instructions entourées de parenthèses. On l'utilise souvent avec un compteur incrémentiel, qui permet de terminer une boucle après un certain nombre de fois.

La suite, à savoir *compteur=0 ; compteur<10 ; compteur++* doit être compris comme suit: «la valeur de la variable *compteur* est comprise entre 0 et 9 (<10 signifie "plus petit que 10") et ajoute un à la valeur de *compteur* (c'est le *compteur++)».*

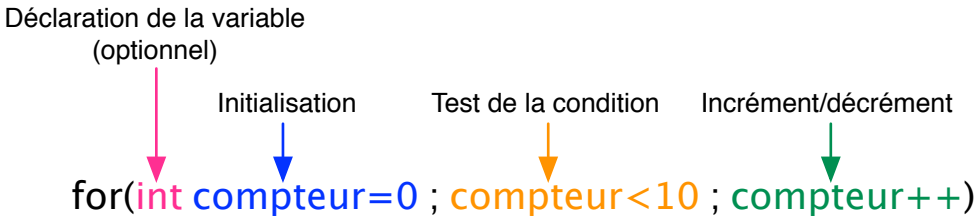
En conclusion, cette ligne signifie:

*«Au commencement, la variable compteur est égale à zéro. On va exécuter le code en boucle. A chaque fois, on ajoute +1 à ta variable compteur, jusqu'à ce qu'on arrive à 9. A ce moment, on s'arrête.»*

Le code qui est exécuté à chaque fois est celui qui est compris jusqu'à l'accolade }. Dans notre cas, c'est e code suivant qui est exécuté 10 fois:

```
digitalWrite(Led1, HIGH);   // allume la LED
delay(500);                 // attend 500ms
digitalWrite(Led1, LOW);    // éteint la LED
delay(500);                 // attend 500ms
```

Voici les parties d'une déclaration *for:*




L'initialisation est effectuée en premier et une seule fois. A chaque passage de la boucle, la condition est testée. Si elle est "vrai", le contenu de la boucle *for* est exécuté (par exemple faire clignoter une LED), et l'incrément de la variable est réalisé. Ensuite, la condition est testée à nouveau. Dès que le résultat du test de la condition est "faux", la boucle s'arrête

[7] Source: http://mediawiki.e-apprendre.net/index.php/Découverte_des_microcontrôleurs_avec_le_Diduino

# Code 6: Réaliser un chenillard sur les broches 10 à 13 avec un *for*

Nous pouvons sans problème utiliser une incrémentation et un *for* pour réaliser le *pinMode* de Leds qui se suivent, par exemple pour réaliser un chenillard:

```
/*
  Code 6 - Edurobot.ch, destiné à l'Arduino
  Objectif: faire un chenillard à 4 LEDs montées sur les ports 10 à 13
*/


//***** EN-TETE DECLARATIVE *****

// Définition de la variable "temps"

int timer = 100;           // Durée, en millisecondes



//***** FONCTION SETUP = Code d'initialisation *****
// La fonction setup() est exécutée en premier et une seule fois, au démarrage du programme


void setup() {

// Déclaration des broches 10 à 13 à l'aide d'un for et d'un incrément.
  for (int thisPin = 10; thisPin < 14; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}

void loop() {
  // boucle de la broche 10 à la broche 13:

for (int thisPin = 10; thisPin < 14; thisPin++) {   //Incrément faisant passer la variable
thisPin de 10 à 13
    digitalWrite(thisPin, HIGH);   //Allumer la LED
    delay(timer);                  //Durée
    digitalWrite(thisPin, LOW);    //Eteindre la LED
  }

  // boucle de la broche 13 à la broche 10:
  for (int thisPin = 13; thisPin >= 10; thisPin--) {  //Décrément faisant passer la variable
thisPin de 13 à 10
    digitalWrite(thisPin, HIGH);   //Allumer la LED
    delay(timer);                  //Durée
    digitalWrite(thisPin, LOW);    //Eteindre la LED;
  }
}
```

## Liens

**Télécharger le code**

http://arduino.education/codes/code6.zip

## Analyse du code

Regardons maintenant un peu ce code, en particulier:

```
for (int thisPin = 10; thisPin < 14; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
```

Sa première particularité est la manière de définir les broches. Au lieu d'accumuler les *pinMode (xxx, OUTPUT);*, on créé une variable de type *int* qu'on appelle *thisPin*[8] et donc la valeur initiale est de *10*. Un *for* avec un incrément (*++*) permet de permet d'incrémenter la valeur de la variable de 10 à 14. Il suffit ensuite de remplacer dans le *pinMode* le numéro de la proche par la variable *thisPin*. Ainsi, les broches 10, 11, 12, 13 et 14 sont définies en output. Naturellement, cela ne fonctionne que si les broches utilisées se suivent (par exemple: 10, 11, 12, 13. Cela ne fonctionnera pas pour des broches 3, 5, 7, 8, 10).

Passons maintenant à la seconde partie:

```
for (int thisPin = 10; thisPin < 14; thisPin++) {   //Incrément faisant passer la variable
thisPin de 10 à 13
    digitalWrite(thisPin, HIGH);   //Allumer la LED
    delay(timer);                  //Durée
    digitalWrite(thisPin, LOW);    //Eteindre la LED
  }
```

Comme pour la définition des broches en OUTPUT, on utilise la fonction *for* pour incrémenter la variable *thisPin* de 10 à 13. Ainsi, on allume les LEDs connectés sur les broches 10 à 13 l'une après l'autre.

Enfin dans la troisième partie du code, on va allumer les LEDs de la broche 13 à la broche 10 avec une fonction *for* et une décrémentation. Pour cela, on remplace le *++* par un *--*.

```
  // boucle de la broche 13 à la broche 10:
  for (int thisPin = 13; thisPin >= 10; thisPin--) {  //Décrément faisant passer la variable
thisPin de 13 à 10
    digitalWrite(thisPin, HIGH);   //Allumer la LED
    delay(timer);                  //Durée
    digitalWrite(thisPin, LOW);    //Eteindre la LED;
```

[8] Mais on aurait aussi pu l'appeler *petite_fleur*...