

Sorties numériques : contrôler 8 LEDs et réaliser des « jeux de lumière », simuler les feux de circulation, apprendre la boucle for, la condition if, utiliser un digit.



Ateliers Arduino

par X. HINAULT

www.mon-club-elec.fr



Tous droits réservés – 2012.

Ce document légèrement payant est soumis au droit d'auteur et est réservé à l'usage personnel.

Afin d'encourager la production de supports didactiques de qualité, ce document est légèrement payant.

La licence d'utilisation est attribuée pour un usage personnel uniquement, dans le cercle familial. Mise en ligne et diffusion non autorisées.

Si vous n'avez pas payé pour l'usage de ce document, soyez sympa, merci d'acheter votre exemplaire personnel ici : <https://monclubelec.dpdcart.com/>

Pour tout problème lié à l'utilisation de ce document, veuillez envoyer une copie ici : support@mon-club-elec.fr

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : support@mon-club-elec.fr

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : support@mon-club-elec.fr

Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !

1. *Intro*

L'objectif ici est :

- de contrôler 8 LEDs à l'aide de 8 broches numériques en sortie
- d'apprendre à utiliser la boucle for
- d'apprendre à utiliser la condition if.. else..
- de simuler les feux de circulation sur 6 LEDs,
- de découvrir le principe d'utilisation d'un digit
- afficher les segments d'un digit ainsi que des chiffres

... afin d'être en mesure de réaliser par vous-même des montages/programmes permettant de réaliser des « jeux de lumières » à LEDs variés et d'utiliser le digit.

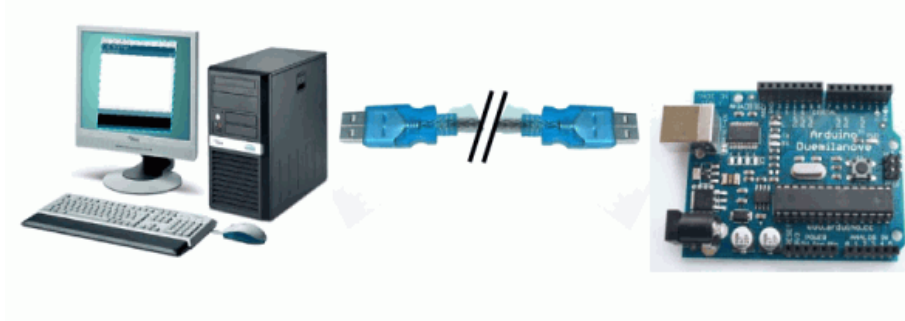


Prêt ? C'est parti !

2. Matériel nécessaire pour les ateliers Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

De l'espace de développement Arduino

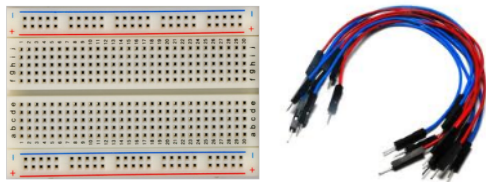


L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : <http://www.arduino.cc/>)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : <http://shop.snootlab.com/> ou <http://www.gotronic.fr/>

Du nécessaire pour réaliser des montages sans soudure

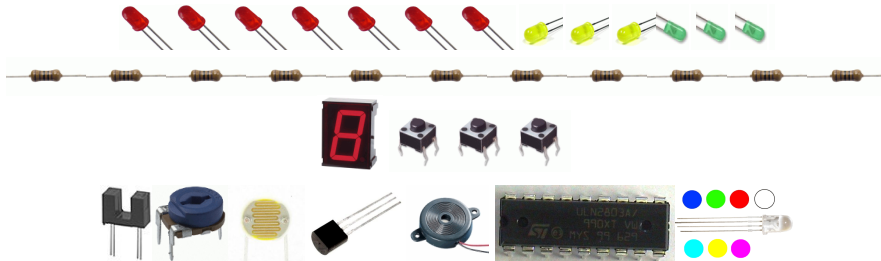


Pour réaliser des montages sans soudure, vous aurez besoin :

- d'une plaque d'essai ou breadboard moyenne (450 points)
- de quelques câbles souples (ou jumpers) mâle/mâle

disponible chez : <http://www.gotronic.fr/>

De quelques composants de base



Pour vous simplifier la vie, nous avons négocié ce kit pour vous !

Vous pouvez commander ce kit complet directement en 1 clic chez notre partenaire
<http://www.gotronic.fr/> avec le code express **701710**

GO TRONIC
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

Pour plus de détails, voir : http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS

Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

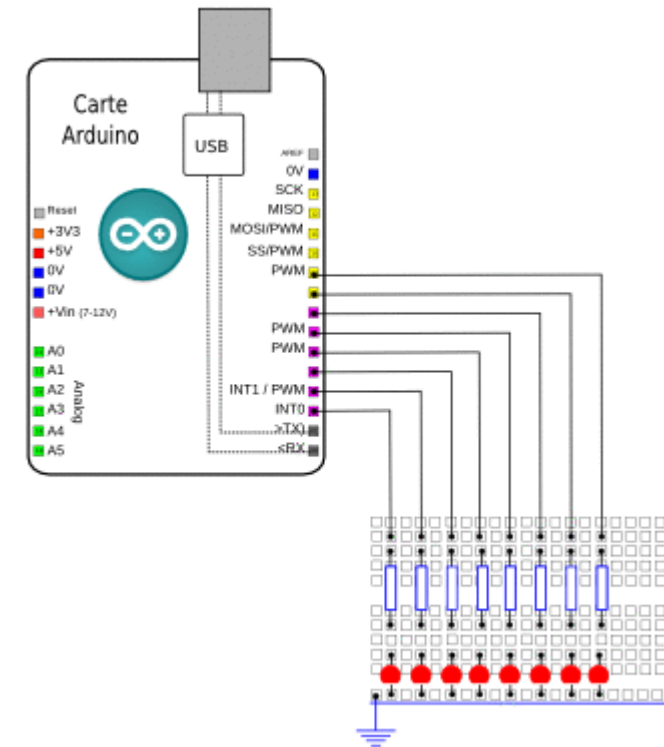
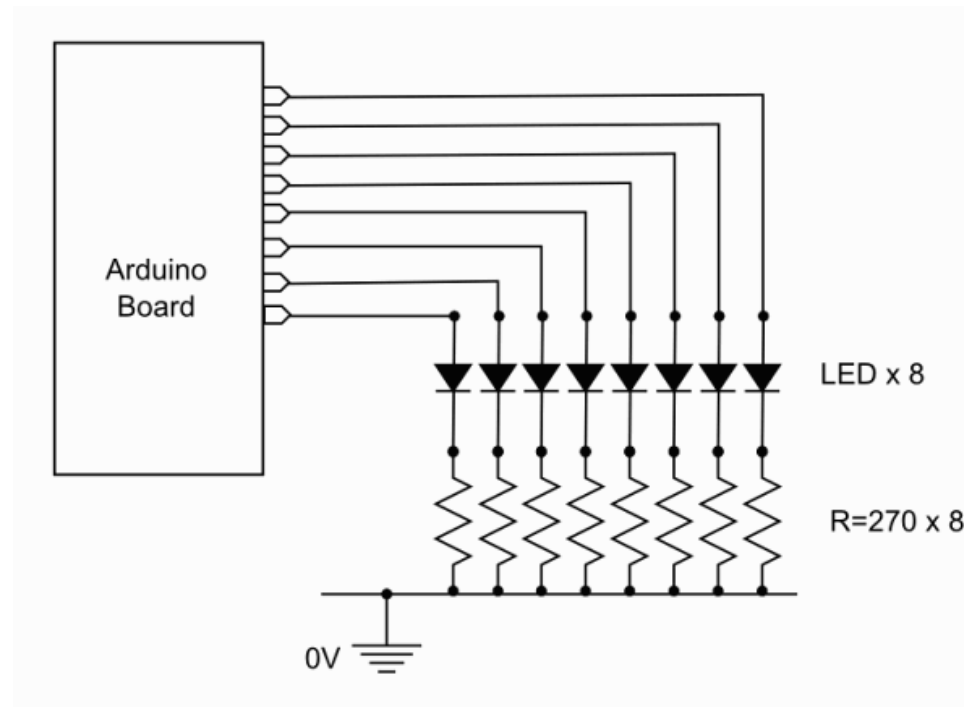
- des LEDs 5mm Rouges(x20), Vertes (x5) et 3 Jaunes (x5)
- digit à cathode commune rouge 13mm (x1)
- Résistances (1/4w - 5%) de 270 Ohms (x20), 4,7K Ohms (x1), 1K Ohms (x1)
- mini bouton-poussoir (x3)
- Opto-fourche (x 1)
- Résistance variable linéaire 10K (x 1)
- Photo-résistance 7mm (x 1)
- Capteur de température LM35DZ (-55/+150°C - 10mV/°C) (x 1)
- Capsule son piézoélectrique (x 1)
- ULN 2803A (CI amplificateur 8 voies, 500mA/ voie) (x 1)
- LED 5mm multicolore RVB cathode commune (x 1)

3. Faire clignoter 8 LEDs : le montage

Continuons, sur notre lancée avec un montage utilisant maintenant 8 LEDs simultanément ! Pour faire clignoter 8 LEDs, on va connecter 8 LEDs chacune en série avec une résistance sur 8 broche E/S de la carte Arduino configurées en sortie.

Le principe sera le suivant :

- lorsque la broche sera au niveau HAUT, la LED sera allumée,
- lorsque la broche sera au niveau BAS, la LED sera éteinte.



Comme vu précédemment, si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R=U/I = 3,5V/0,013A = 270 \text{ Ohms}$.

4. Faire clignoter 8 LEDs en utilisant un tableau de constantes : 1ère version

Cette fois on va reprendre le même programme en utilisant cette fois un tableau de 8 constantes pour désigner les broches :

- un tableau de 8 constantes pour désigner les 8 broches
- et une variable pour fixer la vitesse de clignotement.

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type int pour désigner les broches 2,3,4,5,6,7,8 et 9.
- une variable de type int pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va :

- initialiser les broches utilisées en sortie (avec les constantes LED[0], LED[1], LED[2], LED[3], LED[4],LED[5],LED[6],LED[7])

```
//--- entete déclarative
// = déclarer ici variables et constantes globales

const int LED[8] = {2,3,4,5,6,7,8,9}; // Tableau de constantes

int vitesse=1000; // variable fixant la durée de la pause

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    pinMode(LED[0], OUTPUT); // met la broche en sortie
    pinMode(LED[1], OUTPUT); // met la broche en sortie
    pinMode(LED[2], OUTPUT); // met la broche en sortie
    pinMode(LED[3], OUTPUT); // met la broche en sortie
    pinMode(LED[4], OUTPUT); // met la broche en sortie
    pinMode(LED[5], OUTPUT); // met la broche en sortie
    pinMode(LED[6], OUTPUT); // met la broche en sortie
    pinMode(LED[7], OUTPUT); // met la broche en sortie

} // fin de la fonction setup()
```

(suite) Fonction loop()

A ce niveau on va :

- Allumer les LEDs = mettre les broches au niveau HAUT (5V)
- Attendre le temps voulu (= la valeur de la variable vitesse en millisecondes)
- Eteindre les LEDs = mettre ea broches au niveau BAS (0V)
- Attendre le temps voulu (= la valeur de la variable vitesse en millisecondes)
- le code de la fonction loop se répète sans fin...

Bon, là vous avez le droit de trouver ça bof... C'est un peu « bourrin » cette manière de programmer, mais ça marche !

Allez, on va voir comment améliorer ça !

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    digitalWrite(LED[0],HIGH); // allume la LED
    digitalWrite(LED[1],HIGH); // allume la LED
    digitalWrite(LED[2],HIGH); // allume la LED
    digitalWrite(LED[3],HIGH); // allume la LED
    digitalWrite(LED[4],HIGH); // allume la LED
    digitalWrite(LED[5],HIGH); // allume la LED
    digitalWrite(LED[6],HIGH); // allume la LED
    digitalWrite(LED[7],HIGH); // allume la LED

    delay(vitesse); // pause de n millisecondes

    digitalWrite(LED[0],LOW); // éteint la LED
    digitalWrite(LED[1],LOW); // éteint la LED
    digitalWrite(LED[2],LOW); // éteint la LED
    digitalWrite(LED[3],LOW); // éteint la LED
    digitalWrite(LED[4],LOW); // éteint la LED
    digitalWrite(LED[5],LOW); // éteint la LED
    digitalWrite(LED[6],LOW); // éteint la LED
    digitalWrite(LED[7],LOW); // éteint la LED

    delay(vitesse); // pause de n millisecondes

} // fin de la fonction loop()
```

5. La boucle For

En programmation, comme vous venez de le constater, **on rencontre fréquemment la situation où l'on doit répéter plusieurs fois l'exécution d'un même code** : **pour pouvoir faire cela simplement, on utilise ce que l'on appelle une boucle** !

Imaginons par exemple que l'on veuille faire clignoter très exactement 10 fois une LED : une boucle va nous permettre de faire cela très simplement ! Au lieu de 10, on pourra le faire 23 fois, 252 fois, 1000 fois : c'est très pratique et précis !

Il existe **plusieurs types de boucles** : **la plus fréquente et la plus pratique est la boucle dite for**. Cette boucle dit au microprocesseur la chose suivante :

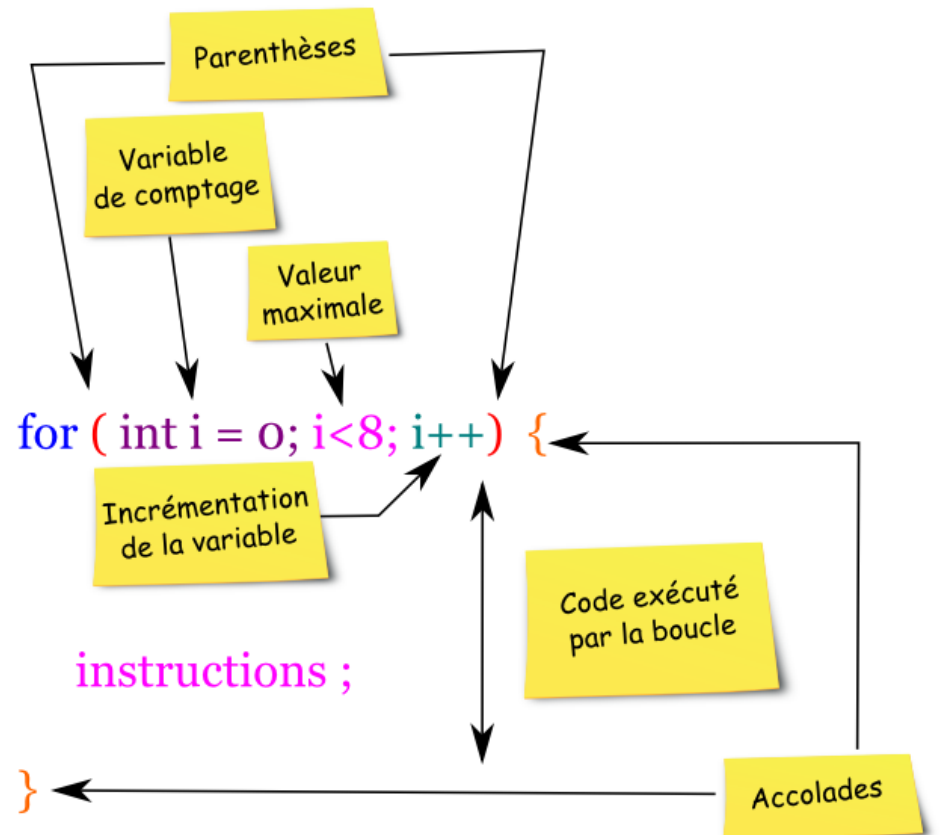
- commencer à compter à partir de telle valeur (0 le plus souvent)
- tant que la valeur n'a pas atteint la valeur maximale
- continuer de compter en ajoutant 1 (ou plus) à la valeur
- à chaque « passage », exécuter les instructions voulues

La structure d'une boucle for s'apparente à celle d'une fonction avec cependant quelques ajouts :

- on commence par le mot clé **for** (pour en anglais) suivi de la parenthèse (
- ensuite on déclare la variable de comptage en déclarant son type (**int** par exemple) et on fixe la valeur de départ. **La variable de comptage est locale limitée à la boucle**. On fait suivre d'un ;
- ensuite, on fixe la valeur maximale à ne pas dépasser. On fait suivre à nouveau d'un ;
- enfin, on indique l'incrémentation à utiliser et on ferme la parenthèse)
- ensuite comme pour une fonction, on met le code entre {}

Explication avancée : En fait le contenu entre les parenthèses d'une boucle correspond à 3 instructions, d'où les points-virgule de séparation :

- la première instruction est la déclaration/initialisation d'une variable typiquement
- la seconde instruction est une condition : tant qu'elle est vraie, la boucle exécute la 3ème instructions
- la troisième instruction correspond à l'opération à effectuer à chaque fois que la condition est vraie. Noter que l'on peut utiliser toute sorte d'opération : $i=i+10$ par exemple.



L'opérateur d'incrément ++ équivaut à l'opération +1

i++ équivaut à i=i+1

L'opérateur de décrémentation - - équivaut à l'opération -1

i-- équivaut à i=i-1

6. Faire clignoter 8 LEDs : 2ème version en utilisant une boucle FOR

Le plus simple pour comprendre l'intérêt d'une boucle est de l'utiliser dans un exemple. Ici, on va reprendre tout simplement le programme précédent en utilisant des boucles pour alléger le programme !

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type int pour désigner les broches 2,3,4,5,6,7,8 et 9.
- une variable de type int pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va :

- initialiser les broches utilisées en sortie en utilisant une boucle, sous la forme LED[i] où i est la variable de comptage de la boucle.

Fonction loop()

A ce niveau on va :

- Allumer les LEDs = mettre les broches au niveau HAUT (5V) en utilisant une boucle pour défiler les 8 broches,
- Attendre le temps voulu (= la valeur de la variable vitesse en millisecondes)
- Eteindre les LEDs = mettre les broches au niveau BAS (0V) en utilisant une boucle pour défiler les 8 broches,
- Attendre le temps voulu (= la valeur de la variable vitesse en millisecondes)
- le code de la fonction loop se répète sans fin...

Remarquer comment le programme s'allège ! (il tient sur 1 seule page...)

Remarquer également que si on utilisait 20 broches, le programme n'est quasiment pas modifié !!

En pratique, utiliser des boucles à chaque fois que vous répétez le même code à plusieurs reprises !

```
//--- entete déclarative
// = déclarer ici variables et constantes globales

const int LED[8] = {2,3,4,5,6,7,8,9}; // Tableau de constantes
int vitesse=1000; // variable fixant la durée de la pause

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //--- met les 8 broches en sortie ---
    for (int i=0; i<8; i++) { // répète 8 fois le code
        pinMode(LED[i], OUTPUT); // met la broche en sortie
    } // fin for

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    // allume les 8 LEDs
    for (int i=0; i<8; i++) { // répète 8 fois le code
        digitalWrite(LED[i],HIGH); // allume la LED
    } // fin for

    delay(vitesse); // pause de n millisecondes

    //--- éteint les 8 LEDs
    for (int i=0; i<8; i++) { // boucle for répète 8 fois
        digitalWrite(LED[i],LOW); // éteint la LED
    } // fin for

    delay(vitesse); // pause de n millisecondes

} // fin de la fonction loop()
```


7. La condition if... else...

Imaginons, à présent que l'on veuille exécuter une instruction que dans certains cas : par exemple, au lieu de faire clignoter toutes les LEDs, on ne va en allumer que une et défiler les LEDs une à une de cette façon.

La solution passe par ce que l'on appelle une condition qui dit au microprocesseur :

- si telle condition est vraie, alors faire ceci
- sinon faire cela.

L'instruction utilisée pour une condition est l'instruction **if ... else...** (si... sinon...) que l'on écrit de la façon suivante :

- on commence par le mot clé **if** suivi de la condition entre ()
- suivi des { } qui contiennent les instructions à exécuter si la condition est vraie, chaque instructions devant être suivie d'un ;
- en option, on peut compléter du mot clé **else** suivi des { } qui contiennent les instructions à exécuter si la condition est fausse.

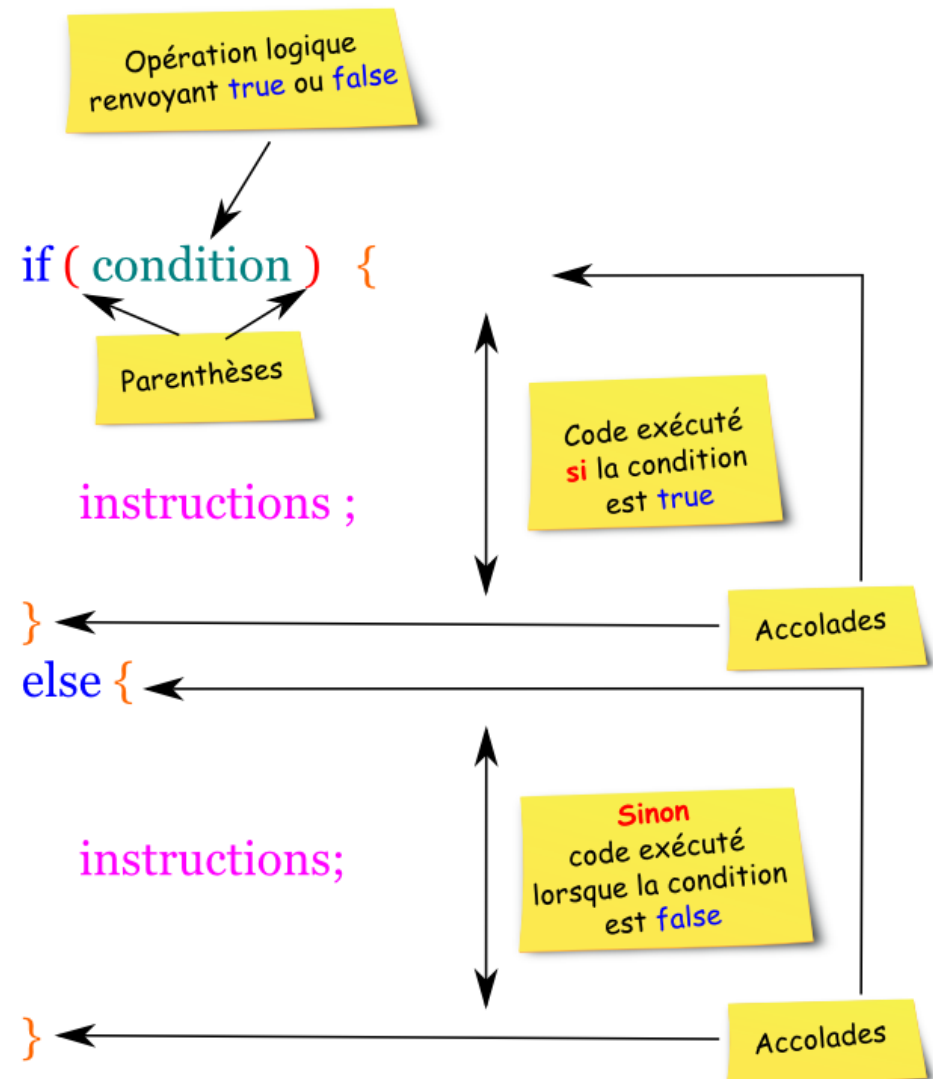
La condition devra être une opération logique :

- qui renverra une valeur de type boolean soit **true** (vrai) soit **false** (faux)
- on utilisera pour cela les **opérateurs logiques de comparaison**
- pour info, 0 est considéré comme false et toute valeur différente de 0 est considérée comme true (ainsi, if(1) est toujours vrai !)

La condition de base utilisable est de tester si une variable vaut une certaine valeur :

- on écrira la condition sous la forme **variable==valeur** (2 signes ==)
- Par exemple on écrira if (variable==2) { instructions ; } ce qui veut dire « si la condition « variable vaut 2 » est vraie alors exécuter les instructions »

Ne pas confondre l'opération logique de test d'égalité == avec le signe = d'affectation : c'est une erreur fréquente de débutant et même de programmeur expérimenté... !!



8. Pour info : les différentes syntaxes (=manières d'écrire) de la condition if.. else..

La forme if simple

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120){ digitalWrite(LEDpin, HIGH); } // toutes ces formes sont correctes
```

Voir également : http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.If

La forme if .. else...

```
if (brocheCinqEntree < 500)
{
    // action A
}
else
{
    // action B
}
```

La forme if ... else if... else...

```
if (brocheCinqEntree < 500)
{
    // faire l'action A
}
else if (brocheCinqEntree >= 1000)
{
    // faire l'action B
}
else
{
    // faire l'action C
}
```

Voir également : http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.IfElse

9. Pour info : les opérateurs logiques et leur utilisation

Les opérateurs logiques de comparaison

- $x == y$: VRAI si x est **égal** à y
- $x != y$: VRAI si x est **différent de** y
- $x < y$: VRAI si x est **inférieur** à y
- $x > y$: VRAI si x est **supérieur** à y
- $x <= y$: VRAI si x est **inférieur ou égal** à y
- $x >= y$: VRAI si x est **supérieur ou égal** à y

Les opérateurs booléens (permettent d'enchaîner des conditions entre-elles)

- **&&** (ET logique) : VRAI seulement si les deux conditions sont VRAI

```
if (digitalRead(2) == HIGH && digitalRead(3) == HIGH) { // lit l'état de 2 boutons poussoirs
// ...
}
```

- **||** (OU logique) : VRAI si l'une des deux conditions est VRAI

```
if (x > 0 || y > 0) { // si x supérieur à 0 ou si y supérieur à 0
// ...
}
```

- **!** (NON logique) : VRAI si l'opérande est FAUX – cas particulier : !x est VRAI chaque fois que x=0 (« tordu » mais pratique !)

```
if (!x) {
// ...
}
```

Pas de panique !

Ce n'est pas grave si vous ne comprenez pas tout ou si vous ne retenez pas tout ! Vous pourrez y revenir tranquillement ultérieurement...
Le but ici est de vous informer de l'existence de ces opérateurs : vous vous familiariserez progressivement avec !

10. Un jeu de lumière à 8 LEDs en utilisant un tableau de variables et une boucle FOR

Allez, on continue... Ici, on va allumer une seule LED à la fois, on va faire « défiler » et recommencer en boucle. On va donc se baser sur :

- un tableau de constantes pour désigner les broches,
- une boucle for de défilement pour parcourir les broches,
- une condition if pour choisir la LED à allumer

Prêt ? C'est parti !

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type **int** pour désigner les broches 2,3,4,5,6,7,8 et 9.
- une variable de type **int** pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va :

- initialiser les broches utilisées en sortie en utilisant une boucle, sous la forme LED[i] où i est la variable de comptage de la boucle.

Fonction loop()

A ce niveau on va :

- Dans une boucle principale, répéter 8 fois le code suivant :
 - Parcourir 1 à 1 les LEDs à l'aide d'une boucle **for**
 - A l'aide d'une condition **if** :
 - Allumer la LED courante de la boucle principale = mettre les broches au niveau HAUT (5V)
 - sinon éteindre les autres LEDs
 - Attendre le temps voulu (= la valeur de la variable vitesse en millisecondes)
- le code de la fonction **loop** se répète sans fin...

Une fois encore, « pensez » le déroulement de votre code :

Ici, les 8 LEDs sont parcourues successivement mais c'est tellement rapide que c'est instantané, quasi-simultané et vous ne voyez que la LED qui reste allumée. Retenez que le microprocesseur va très vite (plusieurs millions d'opérations par secondes !).

```
//--- entete déclarative
// = déclarer ici variables et constantes globales

const int LED[8] = {2,3,4,5,6,7,8,9}; // Tableau de constantes

int vitesse=50; // variable fixant la durée de la pause

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //--- met les 8 broches en sortie ---
    for (int i=0; i<8; i++) { // répète 8 fois le code
        pinMode(LED[i], OUTPUT); // met la broche en sortie
    } // fin for

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    for (int numLED=0; numLED<8; numLED++) { // défile les 8 LEDs

        // allume la LED voulue
        for (int i=0; i<8; i++) { // répète 8 fois le code

            if (i==numLED) { // si la LED est la LED courante
                digitalWrite(LED[i],HIGH); // allume la LED
            }
            else { // sinon
                digitalWrite(LED[i],LOW); // éteint la LED
            }

        } // fin for i

        delay(vitesse); // pause de n millisecondes

    } // fin for numLED

} // fin de la fonction loop()
```

11. IMPORTANT !



Une fois que vous avez compris les boucles (boucle For) et les conditions (if.. else...) vous êtes en mesure de faire face à de très nombreuses situations de programmation !

C'est pour ça que je vous embête avec tout ça d'ailleurs...

Vous êtes à présent capables de trouver une solution dans la plupart des situations auxquelles vous allez être confrontés !

A présent, vous allez surtout apprendre de nouvelles fonctions ou à optimiser votre code, mais vous avez les bases pour coder par vous mêmes !

12. Encore plus fort ! Jeu de lumière à 8 LEDs : une variante à la K2000... avec une fonction !

Poursuivons nos explorations du codage... A présent, on peut compléter un peu ce programme, en réalisant un « aller-retour » ce qui va passer par une boucle de « décrementation ». Essayer de la coder par vous-mêmes et si vous n'y arrivez pas, regardez la solution que je vous propose qui utilise aussi une fonction dédiée...

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type **int** pour désigner les broches 2,3,4,5,6,7,8 et 9.
- une variable de type **int** pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va initialiser les broches utilisées en sortie en utilisant une boucle, sous la forme LED[i] où i est la variable de comptage de la boucle.

Fonction loop()

A ce niveau on va :

- Dans une première boucle principale, répéter 8 fois :
 - l'appel de la fonction chargée d'allumer la LED voulue
 - Attendre le temps voulu (= la valeur de la variable vitesse en millisecondes)
- Ensuite, on écrit le même code dans une 2ème boucle, mais cette fois en parcourant les broches en sens inverse ! Pour ce faire, on peut : soit inverser la logique de la boucle, ou bien utiliser l'indice selon (7-i) (le plus simple) !
- le code de la fonction **loop** se répète sans fin...

Autre fonction : fonction allumeLED(broche)

Afin de ne pas répéter 2 fois le code pour allumer la LED voulue, on peut le mettre dans une fonction séparée qui va « renvoyer rien », va recevoir le numéro de la LED et qui aura pour fonction (trop facile celle-là !...) :

- Parcourir 1 à 1 les LEDs à l'aide d'une boucle **for**
- A l'aide d'une condition **if** :
 - Allumer la LED courante de la boucle principale = mettre les broches au niveau HAUT (5V)
 - sinon éteindre les autres LEDs

Il est déjà pas mal ce petit code ! Vous avez tout compris ? Bravo !

```
//--- entete déclarative
// = déclarer ici variables et constantes globales

const int LED[8] = {2,3,4,5,6,7,8,9}; // Tableau de constantes
int vitesse=20; // variable fixant la durée de la pause

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //--- met les 8 broches en sortie ---
    for (int i=0; i<8; i++) { // répète 8 fois le code
        pinMode(LED[i], OUTPUT); // met la broche en sortie
    } // fin for

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    for (int numLED=0; numLED<8; numLED++) { // défile les 8 LEDs
        crescendo
        allumeLED(numLED); // appelle la fonction pour allumer LED voulue
        delay(vitesse); // pause de n millisecondes
    } // fin for numLED

    for (int numLED=0; numLED<8; numLED++) { // défile les 8 LEDs
        decrescendo
        allumeLED(7-numLED); // appelle la fonction pour allumer LED
        voulu
        delay(vitesse); // pause de n millisecondes
    } // fin for numLED

} // fin de la fonction loop()

//---- fonction qui allume la LED voulue
void allumeLED (int numLEDIn) { // la fonction reçoit le numéro de le
    LED à allumer

    // allume la LED voulue
    for (int i=0; i<8; i++) { // répète 8 fois le code
        if (i==numLEDIn) { // si la LED est la LED courante
            digitalWrite(LED[i],HIGH); // allume la LED
        } // fin if
        else { // sinon
            digitalWrite(LED[i],LOW); // éteint la LED
        } // fin else
    } // fin for i

} // fin allumeLED
```

13. Jeux de lumière à LEDs : une solution plus visuelle pour coder des séquences variées !

A présent, pour ceux qui voudraient aller encore plus loin (si vous avez un peu de mal, vous êtes pas obligés...!), je propose ici une solution pour créer des « jeux de lumière » sous une forme plus visuelle. Cette solution passe par la possibilité, en langage Arduino, d'exprimer les valeurs numériques au format binaire, sous la forme 10010101. En faisant correspondre une LED à chaque valeur 1/0, on obtient une façon simple de « visualiser » l'état des 8 LEDs... Les séquences d'effets deviennent ainsi plus facile à coder. Allez, on se lance...

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type **int** pour désigner les broches 2,3,4,5,6,7,8 et 9.
- une variable de type **int** pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va :

- initialiser les broches utilisées en sortie en utilisant une boucle, sous la forme LED[i] où i est la variable de comptage de la boucle.

Fonction loop()

A ce niveau on va :

- appelle ensuite la fonction de gestion des LEDs en passant une valeur sous une forme binaire, en faisant B11111111 par exemple pour allumer toutes les LEDs. Il faut simplement bien utiliser 8 chiffres !
- répéter autant de fois que voulu pour créer des séquences
- le code de la fonction **loop** se répète sans fin...

Autre fonction : fonction de gestion des LEDs

Nous allons ici à nouveau séparer le code de gestion des LEDs dans une fonction séparée :

- cette fonction va recevoir la valeur binaire
- on défile les 8 broches et à chaque fois on met la LEDs dans l'état du 0/1 correspondant. On utilise pour cela une fonction du langage Arduino qui permet de lire la valeur d'un bit d'une variable, **bitRead**(valeur, bit).
- puis faire une pause entre 2 appels de la fonction.

Ce code est un petit peu plus « avancé » mais il rend la programmation d'effets lumineux très simple !

```
//--- entete déclarative
// = déclarer ici variables et constantes globales
const int LED[8] = {2,3,4,5,6,7,8,9}; // Tableau de constantes
int vitesse=100; // variable fixant la durée de la pause
```

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {
  //--- met les 8 broches en sortie ---
  for (int i=0; i<8; i++) { // répète 8 fois le code
    pinMode(LED[i], OUTPUT); // met la broche en sortie
  } // fin for
} // fin de la fonction setup()
```

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {
```

```
  for (int repet=0; repet<10; repet++ ) { // repete 10 fois la séquence
    allumeLED(B10101010); // appelle la fonction pour allumer LED à
    partir valeur binaire
    // bien utiliser 8 chiffres - B devant pour indiquer format binaire
    allumeLED(B01010101); // appelle la fonction pour allumer LED à
    partir valeur binaire
  } // fin for repet
```

```
  for (int repet=0; repet<10; repet++ ) { // repete 10 fois la séquence
    allumeLED(B00000000); // appelle la fonction pour allumer LED à
    partir valeur binaire
    allumeLED(B10000001);
    allumeLED(B11000011);
    allumeLED(B11100111);
    allumeLED(B11111111);
    allumeLED(B11100111);
    allumeLED(B11000011);
    allumeLED(B10000001);
  } // fin for repet
```

```
} // fin de la fonction loop()
```

```
//fonction qui reçoit une valeur de type int non signé et ne renvoie auc
une valeur
```

```
void allumeLED(unsigned int valeur) { // fonction pour allumer/éteindre
les LEDs voulue en fonction valeur 8 bits reçue
```

```
  for (int i=0; i<=7; i++) {
    digitalWrite(LED[i],bitRead(valeur,7-i)); // met la broche LED[i]
    dans l'état du bit de rang i de la variable
  }
  delay(vitesse); //pause
```

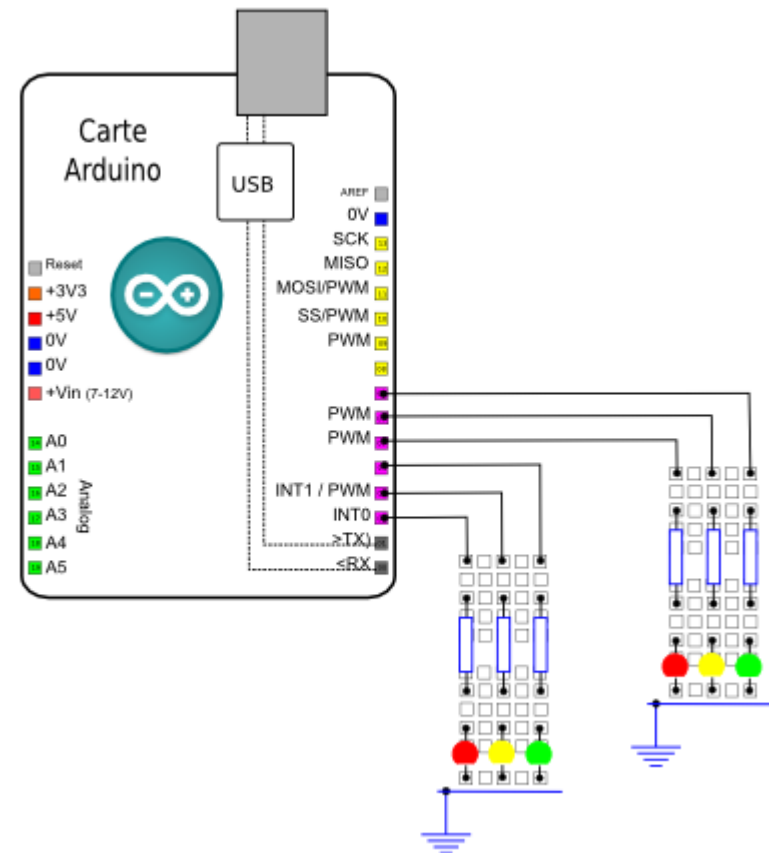
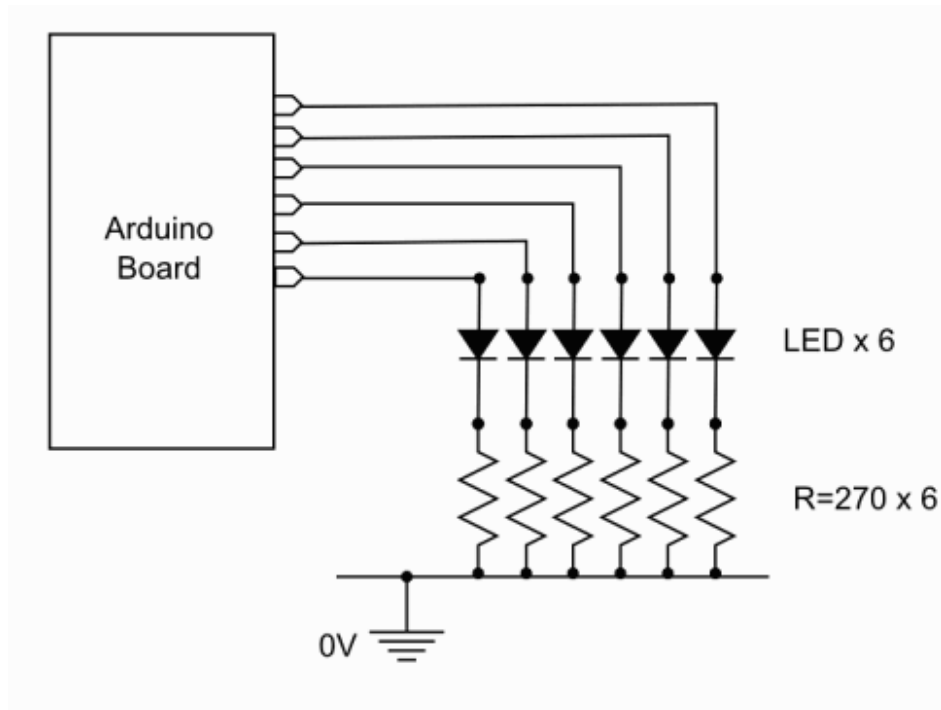
```
} // fin de la fonction allumeLED
```

14. Simuler les feux de circulation avec 6 LEDs : Le montage



Allez, on change un peu de sujet, histoire de varier les plaisirs... Cette fois, je vous propose un petit montage didactique assez parlant : la simulation des feux de circulation ! Une façon sympa de s'entraîner à utiliser les broches en sortie !

On va ici utiliser 6 LEDs : 2 groupes de 3 LEDs associant chacun 1 rouge, 1 jaune et 1 verte. Chaque LED sera connectée en série avec une résistance.



Comme vu précédemment, si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R = U/I = 3,5V/0,013A = 270 \text{ Ohms}$.

Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Sorties numériques : contrôler 8 LEDs et réaliser des « jeux de lumière », apprendre la boucle for, la condition if, utiliser un digit.

15. Simuler les feux de circulation avec 6 LEDs : Le programme.

Bon, là, c'est à vous de jouer... Réfléchissez bien à la séquence des feux de circulation :

- le premier feu est au vert et le 2ème est au rouge
- puis le premier passe à l'orange puis au rouge
- Les 2 restent un peu au rouge
- puis le second passe au vert, le premier restant au rouge
- puis le second passe à l'orange puis au rouge
- les 2 restent un peu au rouge puis...
- le premier feu est au vert et le 2ème est au rouge, etc...

Entête déclarative

On déclare :

- 6 constantes de broches désignant les feux en utilisant des noms significatifs

Fonction setup()

A ce niveau, on va initialiser les broches utilisées en sortie.

Fonction loop()

A ce niveau on va :

- combiner des pauses et des conditions de façon à réaliser la séquence voulue.
- Il y a plusieurs solutions : la mienne est ici : http://www.mon-club-elec.fr//pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ArduinoInitiationLedsFeuxCirculation

Essayez de coder la séquence par vous-même avant de regarder la solution : vous serez content d'y arriver par vous-même !

Vous avez là un bel exemple de séquence élaborée programmable simplement.

Vous ne vous arrêterez plus au feu rouge tout à fait de la même façon...

```
//--- le feu 2 passe à l'orange
digitalWrite(VERT_2,LOW); // éteint vert feu 2
digitalWrite(ORANGE_2,HIGH); // allume orange feu 2

delay (2000); // pause 2 secondes

//--- le feu 2 passe au rouge
digitalWrite(ORANGE_2,LOW); // éteint orange feu 2
digitalWrite(ROUGE_2,HIGH); // allume rouge feu 2

delay (1000); // pause courte 1 seconde

//--- le feu 1 passe au vert
digitalWrite(ROUGE_1,LOW); // éteint rouge feu 1
digitalWrite(VERT_1,HIGH); // allume vert feu 1

delay (5000); // pause longue 5 secondes

//--- le feu 1 passe à l'orange
digitalWrite(VERT_1,LOW); // éteint vert feu 1
digitalWrite(ORANGE_1,HIGH); // allume orange feu 1

delay (2000); // pause 2 secondes

//--- le feu 1 passe au rouge
digitalWrite(ORANGE_1,LOW); // éteint orange feu 1
digitalWrite(ROUGE_1,HIGH); // allume rouge feu 1

delay (1000); // pause courte 1 seconde

//--- le feu 2 passe au vert
digitalWrite(ROUGE_2,LOW); // éteint rouge feu 2
digitalWrite(VERT_2,HIGH); // allume vert feu 2

delay (5000); // pause longue 5 secondes
```

16. Présentation du Digit

Nous allons à présent découvrir le « digit » à LEDs : vous savez, c'est cet afficheur à chiffre rouge ou vert que l'on retrouve sur les réveils ou autres dispositifs de mesure.

Eh bien, vous avez déjà toutes les connaissances nécessaires pour utiliser un digit à LEDs ! Si, si, je vous assure !

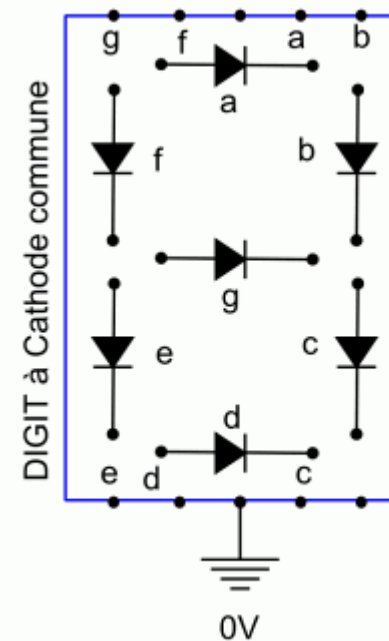
En fait, un digit à LEDs n'est rien d'autre en fait que 8 LEDs de forme allongée qui ont été mises dans un boîtier plastique et disposées en 7 « segments » + 1 point pour pouvoir afficher des chiffres : donc, **si vous savez utiliser 8 LEDs, vous savez utiliser un digit !**

Ces 8 LEDs vont avoir une broche commune. Un digit aura logiquement :

- une broche par LED (7 segments et 1 point), soit 8 broches
- une broche commune, qui va en fait être dédoublée, soit 2 broches.
- Un digit aura donc 10 broches en tout...

On distingue 2 types de Digits en fonction de la broche qui est commune :

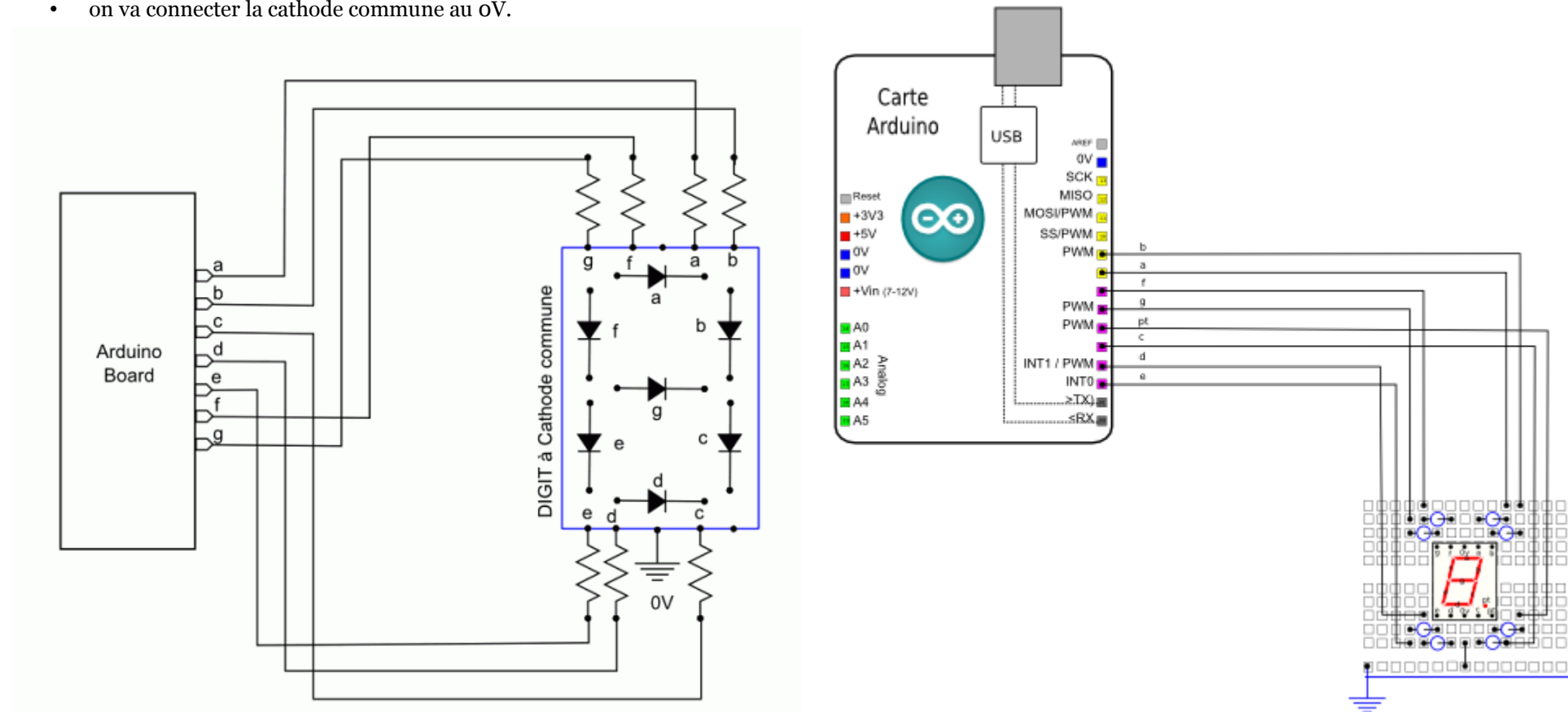
- à cathode commune si – commun
- à anode commune si + commun



17. Afficher les segments d'un digit : le montage

A ce stade, vous devriez être capable de faire le montage vous-mêmes :

- on va connecter chaque LED-segment du digit à une broche Arduino via une résistance de série pour chaque,
- on va connecter la cathode commune au 0V.



Comme vu précédemment, si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R = U/I = 3,5V/0,013A = 270 \text{ Ohms}$.

Ce montage nécessite un peu de rigueur. Utiliser les broches que vous voulez pour les segments : le programme pourra être adapté en conséquence !

Notez bien sur un papier pendant quelle broche est utilisée pour chaque segment.

18. Afficher les segments d'un digit : le programme 1ère version

Donc, à présent, on va allumer puis éteindre successivement tous les segments et le point du Digit. Nous allons reprendre une structure de programme que nous avons déjà utilisée pour réaliser un jeu de lumière à 8 LEDs.

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type **int** pour désigner les 7 broches de segments ainsi que la broche du point.
- une variable de type **int** pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va :

- initialiser les broches utilisées en sortie en utilisant une boucle, sous la forme LED[i] où i est la variable de comptage de la boucle.

Fonction loop()

A ce niveau on va :

- à l'aide d'une boucle, allumer successivement la LED voulue puis l'éteindre après une pause avant de passer à la LED suivante.
- le code de la fonction **loop** se répète sans fin...

Remarquer tout l'intérêt d'utiliser un tableau de constantes pour désigner les broches : si votre montage est différent de celui proposé ici, vous n'avez qu'à modifier la liste des broches au niveau de l'initialisation et tout le reste du programme est inchangé !

Une bonne habitude à prendre pour des programmes faciles à maintenir !

```
//--- entete déclarative
// = déclarer ici variables et constantes globales
const int LED[8] = {8,9,4,3,2,7,6,5}; // Tableau de constantes
// --- ordre des broches : a,b,c,d,e,f,g,pt

int vitesse=200; // variable fixant la durée de la pause

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //--- met les 8 broches en sortie ---
    for (int i=0; i<8; i++) { // répète 8 fois le code
        pinMode(LED[i], OUTPUT); // met la broche en sortie
    } // fin for

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    //--- met les 8 broches en sortie ---
    for (int i=0; i<8; i++) { // répète 8 fois le code

        digitalWrite(LED[i], HIGH); // allume la LED
        delay(vitesse); // pause
        digitalWrite(LED[i], LOW); // éteint la LED

    } // fin for

} // fin de la fonction loop()
```

19. Afficher les segments d'un digit : le programme version améliorée

Comme on l'a déjà vu, on peut rendre un tel programme plus visuel en utilisant des expressions binaires pour fixer l'état des LEDs, technique que nous avons déjà utilisée pour réaliser un jeu de lumière à 8 LEDs. « Y'a plus qu'à... » :

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type **int** pour désigner les 7 broches de segments ainsi que la broche du point.
- une variable de type **int** pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va :

- initialiser les broches utilisées en sortie en utilisant une boucle, sous la forme LED[i] où i est la variable de comptage de la boucle.

Fonction loop()

A ce niveau on va :

- à l'aide d'une boucle, allumer successivement la LED voulue.
- le code de la fonction loop se répète sans fin...

Autre fonction : fonction de gestion des LEDs

Nous allons ici encore une fois séparer le code de gestion des LEDs dans une fonction séparée :

- cette fonction va recevoir la valeur binaire
- on défile les 8 broches et à chaque fois on met la LEDs dans l'état du 0/1 correspondant. On utilise pour cela une fonction du langage Arduino qui permet de lire la valeur d'un bit d'une variable, **bitRead**(valeur, bit).
- puis faire une pause entre 2 appels de la fonction.

Un petit défi, ça vous tente ?

A présent, vous êtes en mesure d'afficher tous les chiffres successivement...

Réfléchissez un peu avant de passer à la suite, et à vous de jouer !

Vous pensez avoir trouvé la solution...? Regardez la diapo suivante !

```
//--- entete déclarative
// = déclarer ici variables et constantes globales
const int LED[8] = {8,9,4,3,2,7,6,5}; // Tableau de constantes
// --- ordre des broches : a,b,c,d,e,f,g,pt

int vitesse=1000; // variable fixant la durée de la pause

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //--- met les 8 broches en sortie ---
    for (int i=0; i<8; i++) { // répète 8 fois le code
        pinMode(LED[i], OUTPUT); // met la broche en sortie
    } // fin for

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    // --- ordre des broches : a,b,c,d,e,f,g,pt
    allumeLED(B10000000); // segt a
    allumeLED(B01000000); // segt b
    allumeLED(B00100000); // segt c
    allumeLED(B00010000); // segt d
    allumeLED(B00001000); // segt e
    allumeLED(B00000100); // segt f
    allumeLED(B00000010); // segt g
    allumeLED(B00000001); // segt pt
    allumeLED(B00000000); // éteint tout

} // fin de la fonction loop()

//fonction qui reçoit une valeur de type int non signé et ne renvoie auc
une valeur
void allumeLED(unsigned int valeur) { // fonction pour allumer/éteindre
les LEDs voulue en fonction valeur 8 bits reçue

    for (int i=0; i<=7; i++) {
        digitalWrite(LED[i],bitRead(valeur,7-i)); // met la broche LED[i]
dans l'état du bit de rang i de la variable
    }
    delay(vitesse); //pause

} // fin de la fonction allumeLED
```

20. Principe d'affichage des chiffres sur un digit

A ce stade, vous devez avoir une petite idée du principe à suivre pour afficher des chiffres sur votre digit... Voici en image le principe à utiliser :

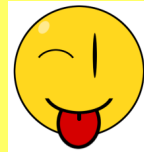


Ainsi :

- pour afficher le 1, on allumera les segments b et c, les autres restant éteints,
- pour afficher le 2, on allumera les segments a,b,d,e et g, les autres restant éteints,
- et ainsi de suite...

Avant de lire la suite, prenez un crayon et écrivez vous-mêmes toutes les séquences.... !

Faut bien que vous bossiez un peu quand même...



21. Afficher des chiffres sur un digit : 1ère version

Vous êtes prêts ? Allez, on se lance pour afficher nos chiffres sur le digit :

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type **int** pour désigner les 7 broches de segments ainsi que la broche du point.
- une variable de type **int** pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va :

- initialiser les broches utilisées en sortie en utilisant une boucle, sous la forme LED[i] où i est la variable de comptage de la boucle.

Fonction loop()

A ce niveau on va :

- à l'aide d'une boucle, allumer successivement les LEDs voulues :
 - B01100000 pour le 1
 - B11011010 pour le 2
 - B11110010 pour le 3
 - etc...
- le code de la fonction loop se répète sans fin...

Autre fonction : fonction de gestion des LEDs

Nous allons ici séparer le code de gestion des LEDs dans une fonction séparée :

- cette fonction va recevoir la valeur binaire
- on défile les 8 broches et à chaque fois on met la LEDs dans l'état du 0/1 correspondant. On utilise pour cela une fonction du langage Arduino qui permet de lire la valeur d'un bit d'une variable, **bitRead**(valeur, bit).
- puis faire une pause entre 2 appels de la fonction.

Sympa ce petit code, non ?

Bravo, vous savez afficher des chiffres sur un digit !

Vous ne regarderez plus votre radio-réveil tout à fait de la même façon !

```
//--- entete déclarative
// = déclarer ici variables et constantes globales
const int LED[8] = {8,9,4,3,2,7,6,5}; // Tableau de constantes
// --- ordre des broches : a,b,c,d,e,f,g,pt

int vitesse=1000; // variable fixant la durée de la pause

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //--- met les 8 broches en sortie ---
    for (int i=0; i<8; i++) { // répète 8 fois le code
        pinMode(LED[i], OUTPUT); // met la broche en sortie
    } // fin for

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    // --- ordre des broches : a,b,c,d,e,f,g,pt
    allumeLED(B11111100); // nombre 0
    allumeLED(B01100000); // nombre 1
    allumeLED(B11011010); // nombre 2
    allumeLED(B11110010); // nombre 3
    allumeLED(B01100110); // nombre 4
    allumeLED(B10110110); // nombre 5
    allumeLED(B10111110); // nombre 6
    allumeLED(B11100000); // nombre 7
    allumeLED(B11111110); // nombre 8
    allumeLED(B11110110); // nombre 9
    allumeLED(B00000000); // éteint tout

} // fin de la fonction loop()

//fonction qui reçoit une valeur de type int non signé et ne renvoie auc
une valeur
void allumeLED(unsigned int valeur) { // fonction pour allumer/éteindre
les LEDs voulue en fonction valeur 8 bits reçue

    for (int i=0; i<=7; i++) {
        digitalWrite(LED[i],bitRead(valeur,7-i)); // met la broche LED[i]
dans l'état du bit de rang i de la variable
    }
    delay(vitesse); //pause

} // fin de la fonction allumeLED
```

22. Pour info : Afficher des chiffres sur un digit en utilisant une fonction dédiée

A présent, on va essayer d'améliorer un petit peu ça (ben oui... on allait tout de même pas s'arrêter en si bon chemin...) On va écrire une fonction de la forme `digit(chiffre,point)` pour afficher le chiffre voulu sur le digit +/- le point !

Entête déclarative

On déclare :

- un tableau de 8 constantes appelé LED de type int pour désigner les 7 broches de segments ainsi que la broche du point.
- une variable de type int pour fixer la vitesse, appelée vitesse

Fonction `setup()`

A ce niveau, on va :

- initialiser les broches utilisées en sortie en utilisant une boucle, sous la forme `LED[i]` où `i` est la variable de comptage de la boucle.

Fonction `loop()`

A ce niveau on va :

- à l'aide d'une boucle, on appelle successivement la fonction d'affichage des chiffres et du point avec les valeurs de 0 à 9
- le code de la fonction `loop` se répète sans fin...

Autre fonction 1: fonction de gestion des LEDs

Nous allons ici séparer le code de gestion des LEDs dans une fonction séparée qui va recevoir la valeur binaire :

- on défile les 8 broches et à chaque fois on met la LEDs dans l'état du 0/1 correspondant. On utilise pour cela une fonction du langage Arduino qui permet de lire la valeur d'un bit d'une variable, `bitRead(valeur, bit)`.
- puis faire une pause entre 2 appels de la fonction.

Remarquer comment l'usage d'une fonction simplifie le code :
il suffit de passer la valeur et les segments voulus seront allumés.

La plupart des instructions du langage Arduino sont des fonctions qui simplifient la programmation et exécutent un code sous-jacent parfois complexe (dont vous n'avez pas à vous soucier d'ailleurs...). Tout ce qui compte pour vous, c'est qu'une fonction Arduino fasse ce qu'on attend elle sans erreur.

« Bandes de petits veinards » je vous dis !

La fonction `digit()` page suivante...

```
//--- entete déclarative
// = déclarer ici variables et constantes globales
const int LED[8] = {8,9,4,3,2,7,6,5}; // Tableau de constantes
// --- ordre des broches : a,b,c,d,e,f,g,pt

int vitesse=1000; // variable fixant la durée de la pause

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //---défile les 8 broches 0 à 7 ---
    for (int i=0; i<8; i++) { // répète 8 fois le code
        pinMode(LED[i], OUTPUT); // met la broche en sortie
    } // fin for

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    //---défile les chiffres 0 à 9 ---
    for (int chiffre=0; chiffre<=9; chiffre++) {
        //digit(chiffre, true); // chiffre + point
        //digit(chiffre, false); // chiffre sans le point
        digit(chiffre, chiffre % 2 ); // chiffre + le point allumé si
        impair (reste=1)
        delay(vitesse); //pause
    }

    digit(-1, false); // éteint segments + point
    delay(vitesse); //pause

} // fin de la fonction loop()

//fonction qui reçoit une valeur de type int non signé et ne renvoie auc
une valeur
void allumeLED(unsigned int valeur) { // fonction pour allumer/éteindre
les LEDs voulue en fonction valeur 8 bits reçue

    for (int i=0; i<=7; i++) {
        digitalWrite(LED[i],bitRead(valeur,7-i)); // met la broche LED[i]
dans l'état du bit de rang i de la variable
    }

} // fin de la fonction allumeLED
```


23. Pour info : Afficher des chiffres sur un digit en utilisant une fonction dédiée (suite)

A présent, voyons le détail de la fonction `digit()` utilisée pour l'affichage des segments et du point sur le digit :

Autre fonction 2: fonction d'affichage du chiffre et du point

Cette fonction appelée `digit` :

- ne renvoie rien (=type **void**)
- reçoit 2 paramètres :
 - la valeur **int** à afficher
 - et la valeur **boolean** pour l'affichage du point
- utilise une série de conditions pour tester la valeur du chiffre reçu et appelle la fonction de gestion des LEDs pour allumer les segments voulus,
- si une valeur négative est reçue, cela éteint tous les segments.
- si le 2ème paramètre :
 - vaut **true**, le point est allumé,
 - si il vaut **false**, le point est éteint.

Ce code vous est présenté à titre informatif pour vous montrer comment écrire une fonction qui reçoit 2 paramètres. Ce n'est pas grave si vous n'arrivez pas à l'écrire par vous-mêmes : c'est un programme déjà un peu avancé. Mais c'est une bonne façon pour vous d'apprendre !



```
//--- fonction qui reçoit chiffre à afficher et boolean pour afficher le point
void digit(int chiffreIn, boolean pointIn) {

    // --- ordre des broches : a,b,c,d,e,f,g,pt
    if (chiffreIn==0) allumeLED(B11111100); // nombre 0
    if (chiffreIn==1) allumeLED(B01100000); // nombre 1
    if (chiffreIn==2) allumeLED(B11011010); // nombre 2
    if (chiffreIn==3) allumeLED(B11110010); // nombre 3
    if (chiffreIn==4) allumeLED(B01100110); // nombre 4
    if (chiffreIn==5) allumeLED(B10110110); // nombre 5
    if (chiffreIn==6) allumeLED(B10111110); // nombre 6
    if (chiffreIn==7) allumeLED(B11100000); // nombre 7
    if (chiffreIn==8) allumeLED(B11111110); // nombre 8
    if (chiffreIn==9) allumeLED(B11110110); // nombre 9

    if (chiffreIn<0) allumeLED(B00000000); // éteint tout si chiffre négatif

    if (pointIn==true) digitalWrite(LED[7], HIGH); // allume point sans modifier les autres
    else digitalWrite(LED[7], LOW); // sinon éteint le point

} // fin fonction digit
```

24. Les digits en pratique

L'utilisation des digits à LEDs en pratique, c'est pas très pratique... :

Avec une carte Arduino, on pourra réaliser un montage avec 2 digits mais guère plus de façon simple : au-delà de 2 digits, la mise en oeuvre des Digits peut être assez complexe et on utilisera alors des modules dédiés qui réalisent le multiplexage de l'affichage.

L'affichage des chiffres sur un Digit vous a été présenté à titre didactique, mais en pratique, pour afficher des valeurs numériques et même des lettres, il vaudra mieux utiliser un afficheur LCD comme nous le verrons : le langage Arduino dispose de fonctions qui rendent très facile l'affichage des nombres et des lettres !

25. Les éléments du langage Arduino étudiés dans cet atelier

Structure

Structures de contrôle

- [if](#)
- [if...else](#)
- [for](#)

Opérateurs de comparaison

- [==](#) (égal à)
- [!=](#) (différent de)
- [<](#) (inférieur à)
- [>](#) (supérieur à)
- [<=](#) (inférieur ou égal à)
- [>=](#) (supérieur ou égal à)

Opérateurs composés

- [++](#) (incréméntation)
- [--](#) (décréméntation)

Variables et constantes

Constante prédéfinies

- [true](#) | [false](#)

Types de données

- [void](#)

Fonctions

Bits et octets

- [bitRead\(\)](#)

La documentation complète du langage Arduino en français est disponible ici :

http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi

26. *A présent, vous devriez être capable :*

- D'utiliser les broches E/S en sortie avec plusieurs LEDs et de réaliser des « jeux de lumière ».
- D'utiliser des digits

Table des matières

Sorties numériques : contrôler 8 LEDs et réaliser des « jeux de lumière », apprendre la boucle for, la condition if, utiliser un digit.

Intro |

Matériel nécessaire pour les ateliers Arduino |

Faire clignoter 8 LEDs : le montage |

Faire clignoter 8 LEDs en utilisant un tableau de constantes : 1ère version |

La boucle For |

Faire clignoter 8 LEDs : 2ème version en utilisant une boucle FOR |

La condition if... else... |

Pour info : les différentes syntaxes (=manières d'écrire) de la condition if.. else.. |

Pour info : les opérateurs logiques et leur utilisation |

Un jeu de lumière à 8 LEDs en utilisant un tableau de variables et une boucle FOR |

IMPORTANT ! |

Encore plus fort ! Jeu de lumière à 8 LEDs : une variante à la K2000... avec une fonction ! |

Jeux de lumière à LEDs : une solution plus visuelle pour coder des séquences variées ! |

Simuler les feux de circulation avec 6 LEDs : Le montage |

Simuler les feux de circulation avec 6 LEDs : Le programme. |

Présentation du Digit |

Afficher les segments d'un digit : le montage |

Afficher les segments d'un digit : le programme 1ère version |

Afficher les segments d'un digit : le programme version améliorée |

Principe d'affichage des chiffres sur un digit |

Afficher des chiffres sur un digit : 1ère version |

Pour info : Afficher des chiffres sur un digit en utilisant une fonction dédiée |

Pour info : Afficher des chiffres sur un digit en utilisant une fonction dédiée (suite) |

Les digits en pratique |

Les éléments du langage Arduino étudiés dans cet atelier |

A présent, vous devriez être capable : |

Bravo !
vous avez terminé cet atelier Arduino !



Prêt pour la suite ? Retrouvez de nombreux autres thèmes d'ateliers Arduino ici :

http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS