

Sorties « analogiques » et impulsions : produire des sons, générer des impulsions « analogiques » (PWM), contrôler une LED multicolore (RGB).



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 d'apprendre pleins de petites choses sympa avec les broches numériques en sortie utilisées en "sortie analogique" :

- produire des sons simples de fréquences variées,
- créer des petites mélodies,
- créer des impulsions de largeur variable,
- utiliser une LED multicolore RGB,

... afin de connaître les utilisations possibles des impulsions et de leur utilisation « analogique ».

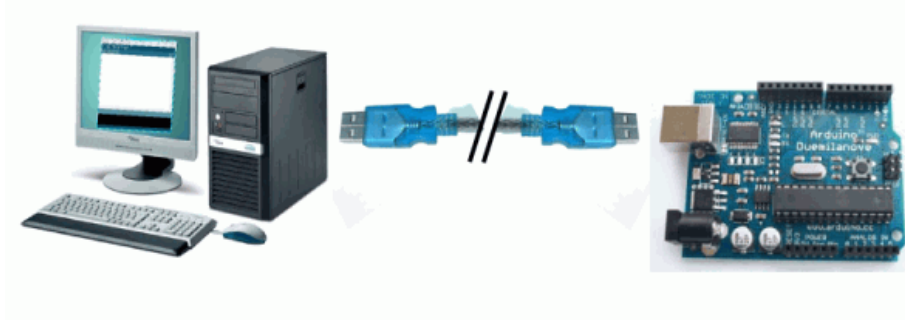


Prêt ? C'est parti !

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

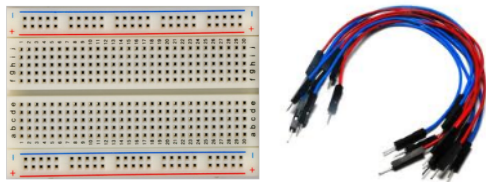


L'espace de développement Arduino associe :

- 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

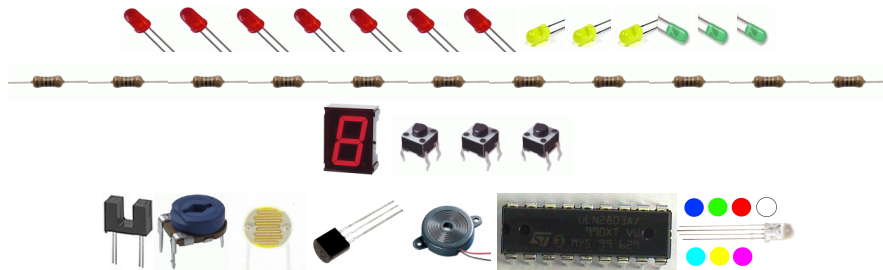


Pour réaliser des montages sans soudure, vous aurez besoin :

- 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

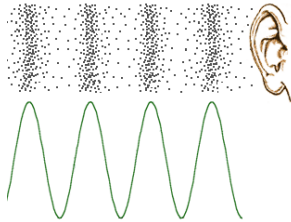
Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

- 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. Rappel de physique : le principe des sons

C'est quoi un son ?

Un son, c'est, rappelons-le, une vibration de l'air périodique qui est perçue par le tympan, une membrane dans l'oreille qui vibre sous l'action de la vibration de l'air.



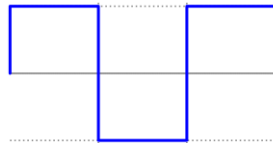
Un son c'est donc une onde : on parle d'onde sonore. Un son nécessite un support matériel : ici, l'air. Dans le vide, pas de sons !

Les différents types d'ondes

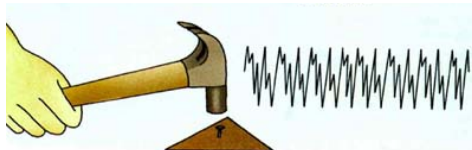
L'onde type est une onde dite « sinusoidale », en vague :



Une autre onde simple est l'onde dite carrée, celle qu'Arduino est capable de produire sur chacune des broches en sorties :

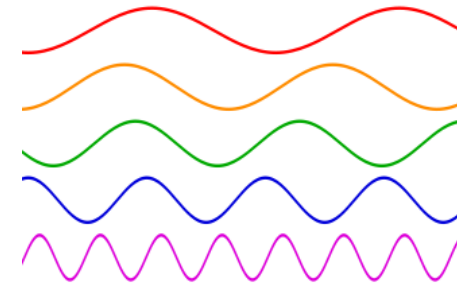


La voix ou les sons courants, sont des ondes complexes qui mélangent des ondes simples.

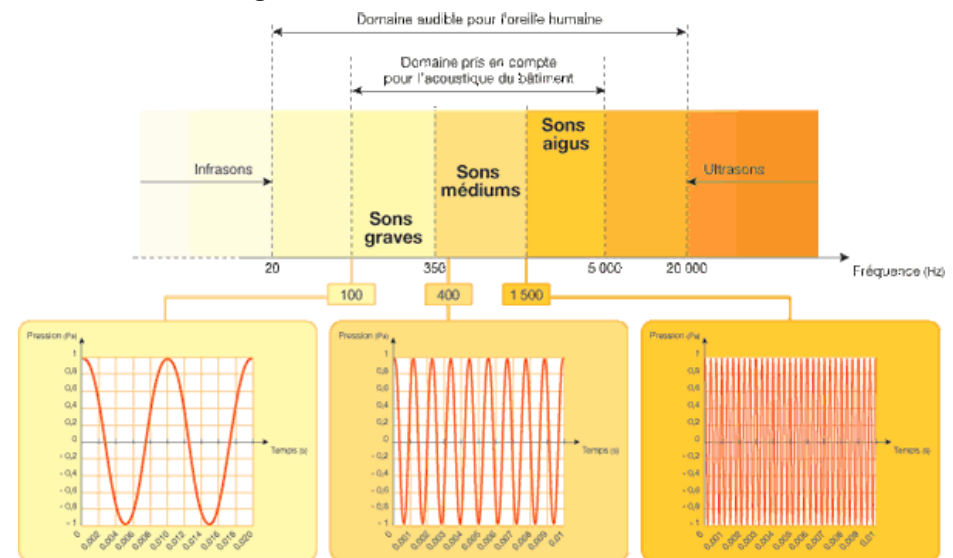


Fréquence d'un son et tonalité du son

La fréquence est le nombre de « vagues » par secondes et se mesure en Hertz. Plus la fréquence est élevée, et plus le son est aigu :

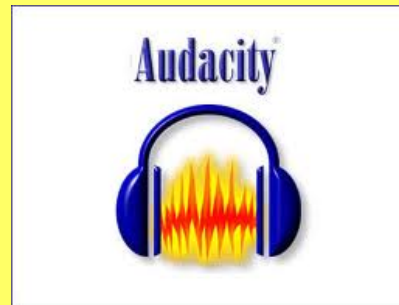


L'oreille humaine est capable d'entendre des sons allant de 20 Hz (très grave) à 20 000 Hz (très aigu).

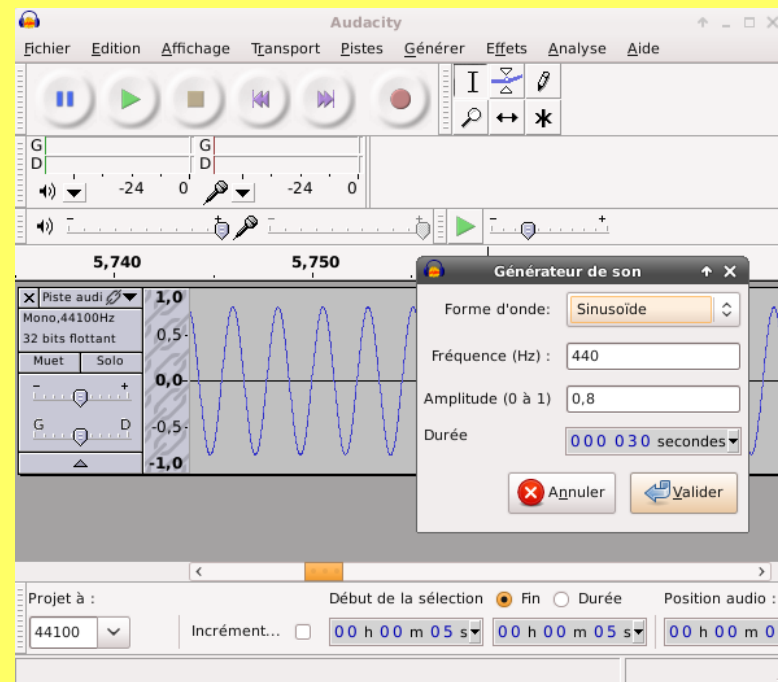


Les notes de musique correspondent à une fréquence précise : le La du 3ème octave correspond à la fréquence 440 Hz, par exemple.

4. Bon à savoir



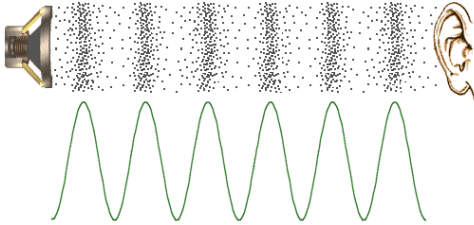
Avec le logiciel libre Audacity, vous pouvez vous amuser par vous-même à créer des sons ayant des formes d'ondes et de fréquences variées !
(Aller dans Menu Générer > Son)



5. Fiche composant : Découvrir le buzzer piézo-électrique et son utilisation avec Arduino

Comment transformer un signal électrique en onde sonore

Pour transformer une onde électrique en onde sonore, on utilise un haut-parleur (l'onde électrique fait vibrer une membrane) :



Arduino va permettre de créer des sons simples de ce type.

Présentation du buzzer piézo-électrique

Pour produire des sons avec Arduino, on va utiliser une sorte de haut-parleur miniature appelé capsule (ou buzzer) piézo-electrique (coût = 1€). Ce type de buzzer dispose d'une membrane métallique qui va vibrer sous l'action d'un signal électrique et va fournir un son de fréquence correspondante.

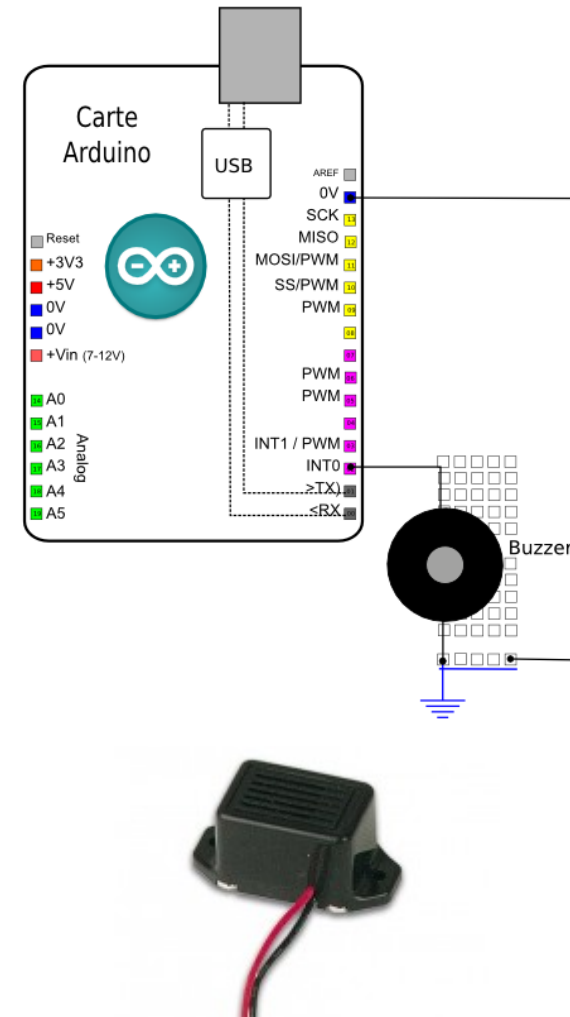


Ce type de buzzer piézo-electrique dispose d'un fil noir et d'un fil rouge. Le principe d'utilisation est très simple :

- on connecte le fil noir au 0V
- et le fil rouge sur la broche utilisée pour générer la fréquence du son :

Ce type de capsule piézo-électrique a les caractéristiques suivantes :

- intensité = 2mA
- tension = 3 à 30V
- l'utilisation directe sur une broche Arduino est donc possible !

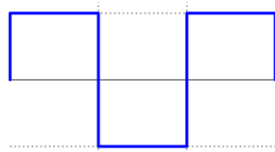


Attention : distinguer le buzzer piézo-électrique du buzzer « simple » qui produit un son ou une mélodie lorsqu'il est mis sous tension : ce type de buzzer n'est pas adapté pour produire des sons de fréquence voulue.

6. Produire un simple son avec une carte Arduino

Pour comprendre

- Pour obtenir un son avec une capsule piézo-électrique, il faut produire une onde électrique de la fréquence voulue au buzzer.
- Si on met une broche E/S de la carte Arduino au niveau HAUT puis au niveau BAS et que l'on recommence rapidement avec un délai court entre chaque niveau, on va produire une onde carrée... et le tour est joué !



- C'est un peu comme si on faisait « clignoter » le buzzer très vite... ce qui produit un son. Le problème à présent va être de fixer la fréquence du son, ce qui va se faire très simplement à l'aide d'une pause en microsecondes (instruction `delayMicroseconds()`).

Fonction setup()

On commence par mettre en sortie la broche que l'on utilise.

Fonction loop()

De la même façon que pour faire clignoter une LED, on va :

- mettre la broche au niveau HAUT
- faire une courte pause à l'aide de l'instruction `delayMicroseconds()` (pour 500 Hz, utiliser 1000 μ s = 1ms)
- mettre la broche au niveau BAS
- faire à nouveau une courte pause de même durée pour obtenir une onde carrée symétrique.

La fréquence obtenue vaudra $f=1/T=10^6/(2 \times \text{délai})$ (délai en μ s).

A l'inverse, pour obtenir une fréquence donnée, on utilisera une pause de $\text{délai} = 10^6/(2 \times f) \mu$ s, soit pour 500Hz, $\text{délai} = 10^6/(2 \times 500) = 1000 \mu$ s

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

const int PIEZO=2; // constante pour la broche du piézo

int demiPeriode=1000; // variable de demi période en micro-secondes

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

  pinMode(PIEZO, OUTPUT); // met la broche en sortie

} // fin de la fonction setup()

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

  digitalWrite(PIEZO,HIGH); // allume la LED
  delayMicroseconds(demiPeriode); // pause en microsecondes
  digitalWrite(PIEZO,LOW); // éteint la LED
  delayMicroseconds(demiPeriode); // pause en microsecondes

} // fin de la fonction loop()

// NB : les lignes précédées de // sont des commentaires
```

7. Produire un son de fréquence voulue avec une carte Arduino

Devoir calculer la demi-période, c'est un petit peu compliqué... Allez, on va simplifier ça...

Les instructions **tone()** et **noTone()**

Le langage Arduino propose une instruction dédiée qui permet de générer simplement une impulsion carrée de fréquence voulue sur n'importe quelle broche : l'instruction **tone**(broche, fréquence)

Le son durera tant que l'instruction **noTone**(broche) n'aura pas été appelée.

Fonction **setup()**

On peut mettre la broche en sortie, mais on n'est pas obligé de la faire, car la fonction **tone** met automatiquement la broche en sortie.

Fonction **loop()**

On appelle tout simplement la fonction **tone** :

- en précisant la broche à utiliser
- la fréquence voulue
- ... et c'est tout.
- ensuite mettre une pause de durée voulue,
- et stopper le son avec **noTone()**,
- faire une nouvelle pause,
- etc...

Difficile de faire plus simple !

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

const int PIEZO=2; // constante pour la broche du piézo

int LA3=440; // variable de la fréquence

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

  pinMode(PIEZO, OUTPUT); // met la broche en sortie

} // fin de la fonction setup()

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

  tone(PIEZO, LA3); // lance la génération du son
  delay(1000);
  noTone(PIEZO); // stoppe le son
  delay(1000);

} // fin de la fonction loop()

// NB : les lignes précédées de // sont des commentaires
```


8. Produire une simple mélodie avec une carte Arduino

A ce stade, vous devez commencer probablement à avoir quelques idées intéressantes : il devient facile de créer de petites alarmes sonores, ou même des mélodies simples.

Variables de notes

L'idée qui vient à l'esprit est de créer des variables de notes dont le nom correspond à la note. Il ne reste plus qu'à trouver un listing des notes et à créer toutes les variables de notes...

Par exemple, ici, nous allons créer les 3 notes DO, RE, MI :

- `int DO=262;`
- `int RE=294;`
- `int MI=330;`

Entete déclarative

On va déclarer à ce niveau :

- une constante désignant la broche utilisée
- des constantes de notes
- 1 tableau de note et 1 tableau de durée des notes

Fonction `setup()`

On peut mettre la broche en sortie, mais on n'est pas obligé de la faire, car la fonction `tone` met automatiquement la broche en sortie.

Fonction `loop()`

- On appelle tout simplement la fonction `tone` pour chaque note au sein d'une boucle `for`, en précisant la broche à utiliser et la fréquence voulue,
- ensuite mettre une pause de durée voulue,
- et stopper les sons avec `noTone()`,
- faire une nouvelle pause, etc...

A vous de jouer !

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

const int PIEZO=2; // constante pour la broche du piézo

int DO=262; // variable de la fréquence note
int RE=294; // variable de la fréquence note
int MI=330; // variable de la fréquence note

const int nombreNotes=11; // nombre de notes - doit etre une constante

int note[nombreNotes]= {DO,DO,DO,RE,MI,RE,DO,MI,RE,RE,DO}; // liste des notes
int duree[nombreNotes]={1 ,1 ,1 ,1 ,1 ,2 ,1 ,1 ,1 ,1 ,1 }; // liste des notes

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

  pinMode(PIEZO, OUTPUT); // met la broche en sortie

} // fin de la fonction setup()

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

  for (int i=0; i<nombreNotes; i++) {

    //-- joue la note de rang i
    tone(PIEZO, note[i]); // lance la génération du son
    delay(500*duree[i]); // pause pendant durée de la note

    //-- stoppe brièvement la note
    noTone(PIEZO); // stoppe le son
    delay(100);

  }

  noTone(PIEZO); // stoppe le son
  delay(1000); // pause

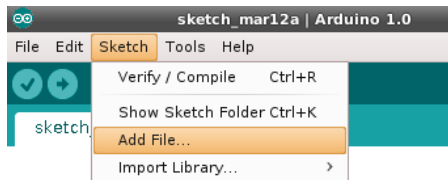
} // fin de la fonction loop()
```

9. Un truc bon à connaître : inclure un fichier dans votre code !

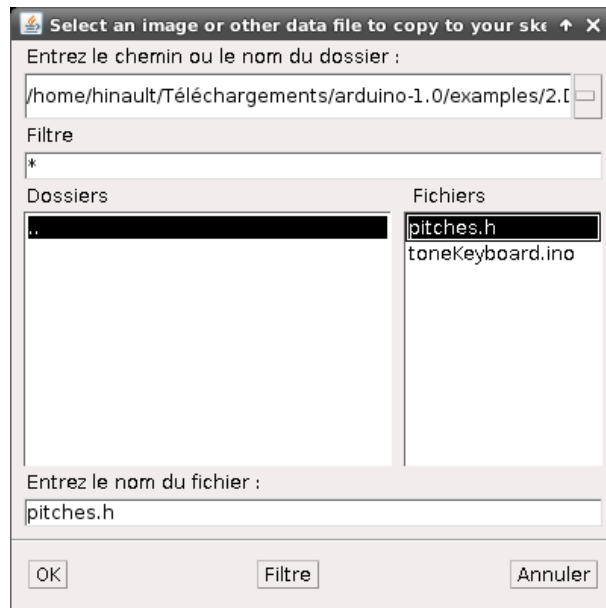
Le logiciel Arduino est décidément bien pensé : vous pouvez inclure un fichier existant dans votre code... Suivez mon regard : en définissant toutes les notes une fois pour toutes dans un fichier, vous n'avez plus qu'à l'inclure à votre programme utilisant les sons.

Comment faire ?

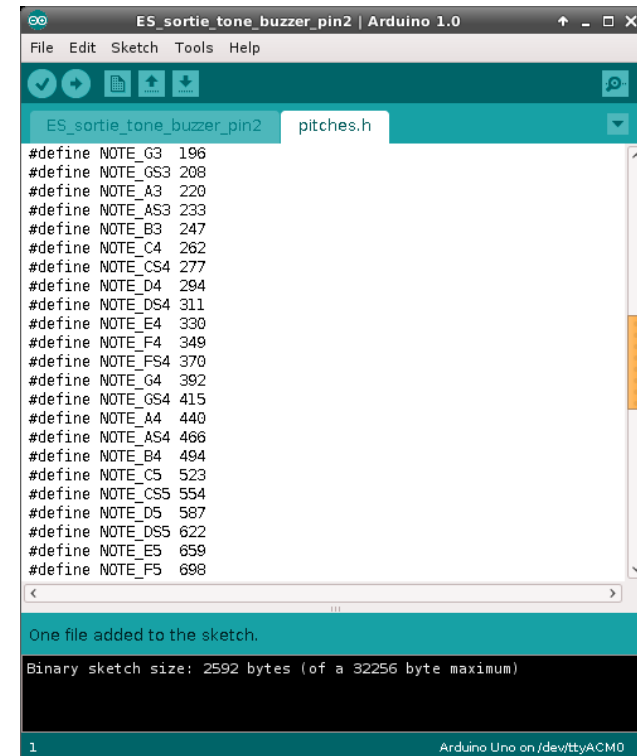
Allez dans le menu sketch > Add file



Sélectionner le fichier puis valider :



Une fois fait, un nouvel onglet est créé et le contenu du fichier y est placé :



On pourra de cette façon inclure le fichier de notes pitches.h fournit avec Arduino et qui se trouve dans exemples > digital > toneMelody
Ce fichier définit toutes les notes courantes au format anglo-saxon.

10. Générer un son de sirène de police avec une carte Arduino

Continuons nos explorations autour des sons et créons un son de sirène de police. Le son de sirène de police correspond à une ascension croissante de la fréquence suivi d'une diminution progressive de la fréquence dans la même proportion. Voyons comment réaliser cela avec l'instruction `tone()`

Entete déclarative

On va déclarer à ce niveau :

- une constante désignant la broche utilisée.
- une variable de mémorisation de la valeur de `millis()`

Fonction `setup()`

- On peut mettre la broche en sortie, mais on n'est pas obligé de la faire, car la fonction `tone()` met automatiquement la broche en sortie.
- On mémorise également la valeur de `millis()`

Fonction `loop()`

- On commence par utiliser une boucle `for` pour faire varier la fréquence dans un sens croissant. Pour chaque valeur de la fréquence, on exécute la fonction `tone()`.
- De la même façon, on utilise ensuite une boucle `for` pour faire décroître la fréquence. Là encore, pour chaque valeur de la fréquence, on appelle la fonction `tone()`.
- la séquence se répète, créant un son de sirène de police !
- on teste si un délai de 10 secondes s'est écoulé : si c'est le cas, on stoppe le programme. Un simple reset relancera la sirène pour 10 secondes.



On a ici un bel exemple simple de la souplesse d'utilisation d'Arduino !

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

const int PIEZO=2; // constante pour la broche du piézo

int millis0=0; // variable de mémorisation millis()

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

  pinMode(PIEZO, OUTPUT); // met la broche en sortie

  millis0=millis(); // mémorise millis()

} // fin de la fonction setup()

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

  //---- boucle défilement croissant de la fréquence ---
  for (int frequence=500; frequence<=1000; frequence++) {

    tone(PIEZO, frequence); // génère le son de fréquence voulue
    delay(1); // courte pause

  } // fin boucle for

  //---- boucle de défilement décroissant de la fréquence ---
  for (int frequence=1000; frequence>=500; frequence--) {

    tone(PIEZO, frequence); // génère le son de fréquence voulue
    delay(1); // courte pause

  } // fin boucle for

  if (millis()-millis0>10000) { // après 10 secondes
    noTone(PIEZO); // stoppe le son
    while(1); // stoppe loop
  }

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

11. Le concept de modulation de largeur d'impulsion (MLI)

Nous allons à présent voir comment « simuler » des comportements analogique à partir d'une broche d'E/S numérique en sortie. Rappelez-vous :

- numérique (ou ON/OFF) : tout ou rien, pas de niveau intermédiaire.
- analogique : variation progressive avec tous les niveaux intermédiaires.

Par exemple, dans le cas d'une lampe :

- en fonctionnement ON/OFF : la lampe est allumée ou éteinte
- en fonctionnement analogique : la lampe peut voir sa luminosité varier d'aucune lumière à luminosité maximale.

Simuler de l'analogique avec une broche numérique ...

Il serait pratique de pouvoir réaliser un comportement « analogique » avec une E/S broche numérique... mais comment faire ? La broche numérique ne peut en effet prendre que 2 niveaux : HAUT ou BAS....

Imaginez à présent que l'on définisse une période de temps donnée assez courte et que l'on allume la LED que pendant une partie de cette durée, par exemple 25% puis qu'on la laisse éteinte les 75% restant et que l'on recommence ensuite très rapidement. Que va-t-il se passer ? Vous verrez la LED allumée au quart de sa luminosité maximale ! Le tour est joué...

En effet, la tension moyenne au fil du temps vaudra 25% de 5V soit 1,25V !

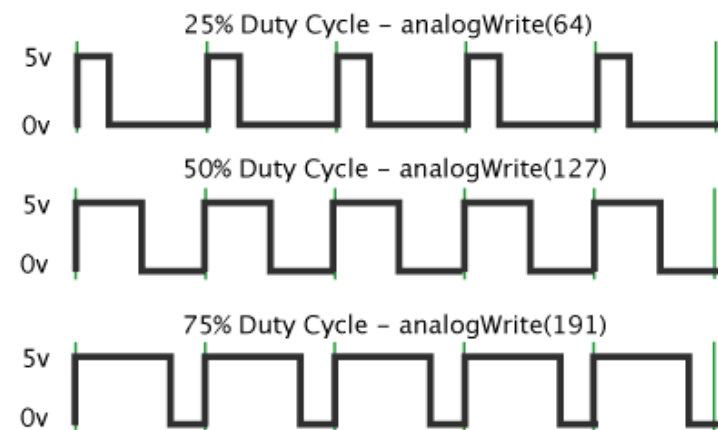
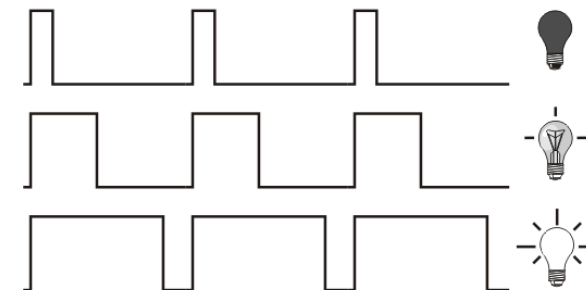
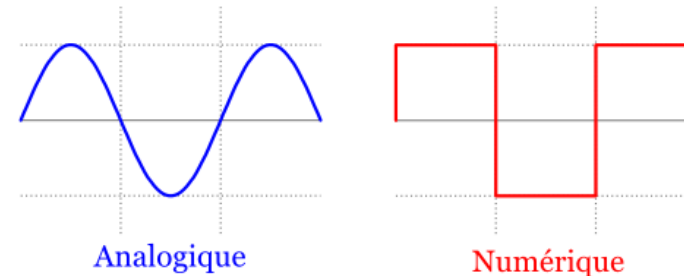
Le concept de Modulation de Largeur d'Impulsion (ou PWM)

Vous avez compris ? Alors, vous venez de saisir ce qui se cache derrière le concept de « Modulation de Largeur d'Impulsion » ou MLI (Pulse Width Modulation en anglais, ou PWM) !

En pratique :

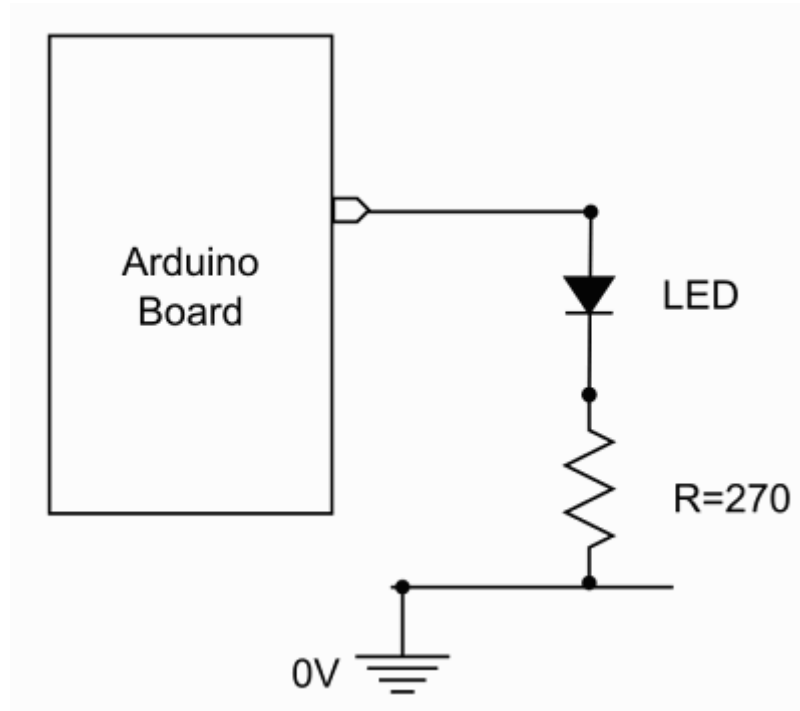
- on appelle « duty-cycle » la période de temps de base, qui correspond à 100%
- on fixera la largeur de l'impulsion (ou « pulse width ») de 0 à 100%,
- ce qui en binaire s'exprimera par un octet dont la valeur sera entre 0(0%) et 255 (=100%).

Avec le langage Arduino, on générera une telle impulsion grâce à l'instruction `analogWrite()` **disponible seulement sur les broches dites PWM** (sigle ~ à côté soit les broches 3,5,6,9,10,11).

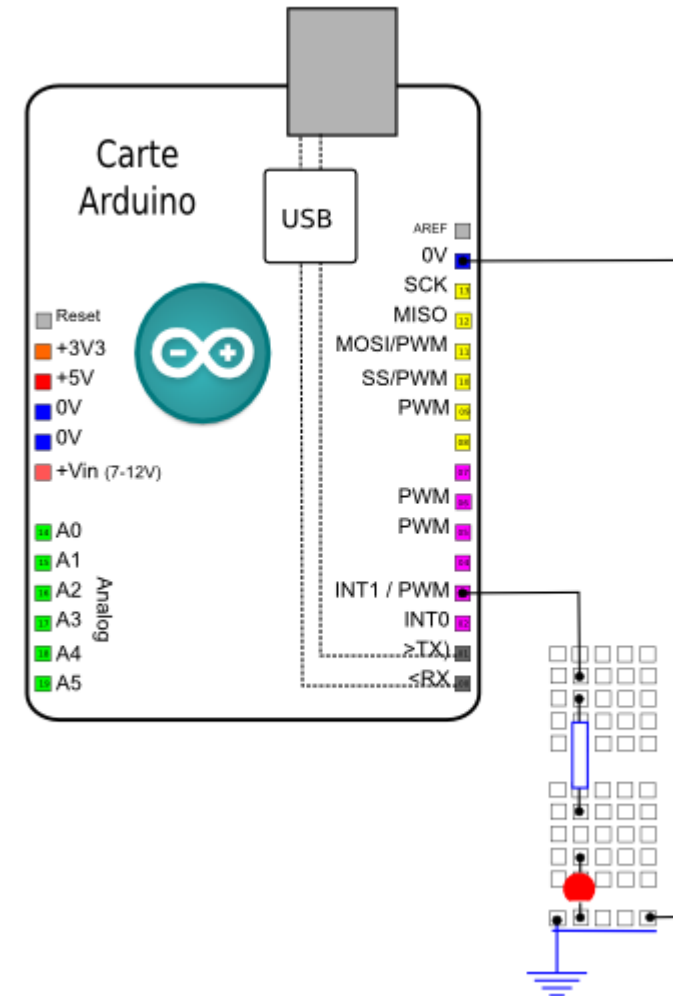


12. Faire varier la luminosité d'une LED : le montage

On va réutiliser le premier montage réalisé avec la carte Arduino, ce qui ne devrait pas vous poser de problème, mais ici sur la broche 3 qui dispose de l'impulsion PWM.



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



13. Faire varier la luminosité d'une LED : le programme

Les possibilités d'utilisation de la modulation de largeur d'impulsion sont nombreuses, notamment :

- faire varier la luminosité d'une LED
- faire varier la vitesse d'un moteur
- générer des variations de couleur avec une LED multicolore RVB

Ici, nous allons faire quelque chose de très simple : faire varier la luminosité de d'une LED de 0% à 100% puis de 100% à 0%.

Entete déclarative

- On va déclarer à ce niveau la constante désignant la broche utilisée.

Fonction `setup()`

- on configure la broche en sortie avec l'instruction `pinMode()`

Fonction `loop()`

- La modulation de largeur d'impulsion est déclenchée par l'instruction `digitalWrite(broche, largeur)` où :
 - broche : est la broche utilisée uniquement parmi celles ayant cette fonction, à savoir les broches avec le symbole ~, soit les broches 3,5,6,9,10,11 sur la carte Arduino standard.
 - largeur : % de temps du cycle où la broche est au niveau haut, avec 0 pour 0% et 255 pour 100%
- Au sein d'une première boucle, on fera varier la largeur d'impulsion de 0% à 100%.
- puis dans une 2ème boucle de 100% à 0%

Truc : noter l'instruction

`map(valeur,minSource,maxSource,minDestination, maxDestination)` qui permet de ré-échelonner une valeur simplement.

Utile ici pour convertir la largeur d'impulsion comprise entre 0% et 100% dans la valeur correspondante comprise entre 0 et 255 !

Autrement dit, cette fonction réalise une « règle de 3 »...

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

const int LED=3; // constante désignant la broche de la LED
int vitesse=20; // variable fixant la durée de la pause en ms

int largeur=0; // variable pour la largeur d'impulsion

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

  pinMode(LED, OUTPUT); // met la broche en sortie

} // fin de la fonction setup()

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

  for (int i=0; i<=100; i++) { // boucle croissante

    largeur=map(i,0,100, 0,255); // ré-échelonne i (0-100%) vers (0-255)
    analogWrite(LED,largeur); // génère impulsion PWM voulue sur la broche
    delay(vitesse); // pause

  } // fin for i

  for (int i=100; i>=0; i--) { // boucle décroissante

    largeur=map(i,0,100, 0,255); // ré-échelonne i (0-100%) vers (0-255)
    analogWrite(LED,largeur); // génère impulsion PWM voulue sur la broche
    delay(vitesse); // pause

  } // fin for i

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

14. Fiche composant : Découvrir la LED multicolore RGB

La LED

Rappelez-vous la LED (Light Emitting Diode) : c'est un composant électronique qui comme tout le monde le sait sert à produire de la lumière :

- la LED standard 5mm est de couleur rouge verte ou jaune
- la **tension à ses bornes est constante à 1,5V** environ lorsqu'elle est en circuit
- son intensité de fonctionnement est de l'ordre de 20mA
- la LED est une diode : elle a donc un sens de connexion.

Le montage type d'une LED standard en 5V consiste à :

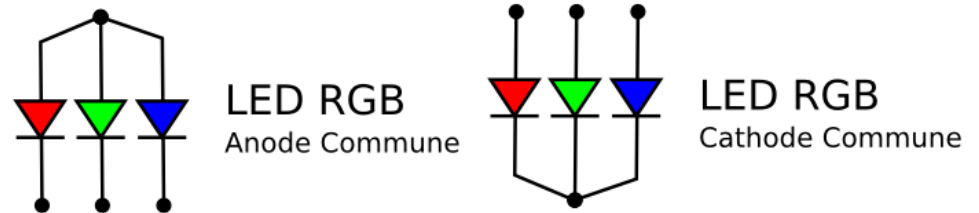
- connecter la LED en série avec une résistance (La patte – de la LED devra être tournée vers le -)
- la résistance a pour but de limiter l'intensité. Utiliser une valeur entre 200 et 300 Ohms. En pratique, j'utilise 270 Ohms / 1/4 de watt.

La LED multicolore dite « RGB » ou « RVB » :

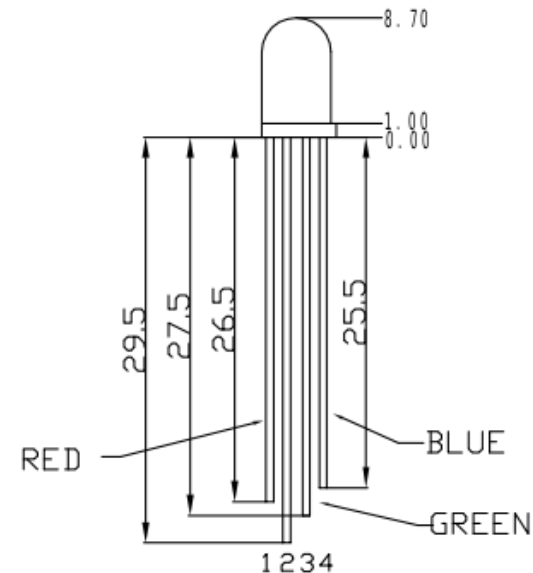
- Une **LED RVB** pour Rouge-Vert-Bleu (ou RGB pour Red Green Blue), n'est autre qu'un **ensemble de 3 LEDs couleurs rassemblées dans un même boîtier** : si vous savez utiliser une LED, alors vous saurez utiliser une LED RVB !
- Une LED RGB aura donc **4 broches** : une commune à l'ensemble des LEDs et une pour chaque LED de couleur. La broche commune pourra, selon les modèles, être le + (anode commune) ou le - (cathode commune)
- Pour obtenir les couleurs unitaires, il suffira d'allumer la LED de la couleur voulue.

Pour les matheux (on est pas du tout obligé de savoir faire ce calcul !) :

- aux bornes de la LED, la tension vaut 1,5V environ (fixe)
- la tension aux bornes de la résistance en série avec la LED, dans le cas d'une alimentation en 5V, vaudra donc $5V - 1,5V = 3,5V$
- si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R = U/I = 3,5V / 0,013A = 270 \text{ Ohms}$.

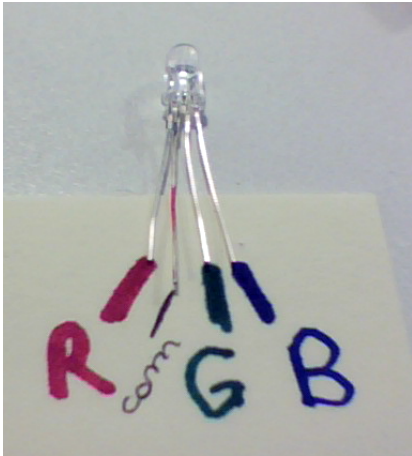


Le brochage type d'une LED RGB est le suivant (la "patte" la plus longue est la broche commune, l'anode ou la cathode selon le modèle) :

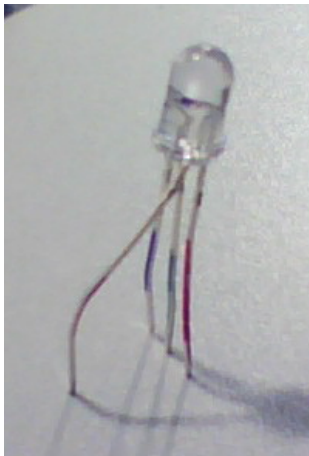


15. Truc technique : préparer la LED multicolore RGB pour une utilisation simplifiée sur plaque d'essai

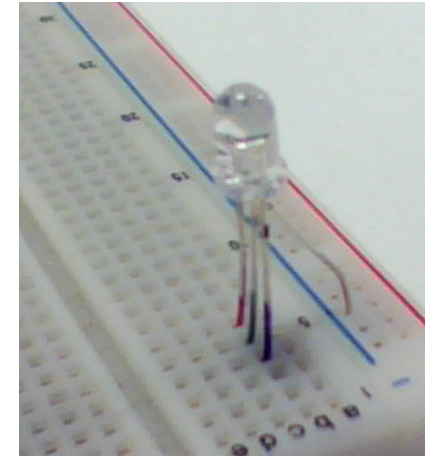
Repérer les broches, notamment la plus longue, commune :



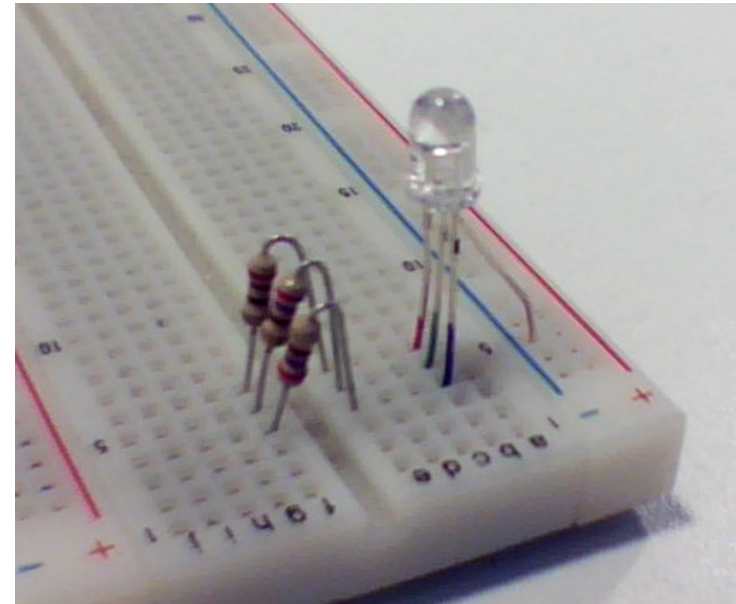
Courber une patte longue (broche commune) en la mettant en face de la broche G et rapprocher les 3 broches RGB entre-elles, notamment la broche R :



Ainsi préparée, la LED RGB est très facile à utilisée sur la plaque d'essai :



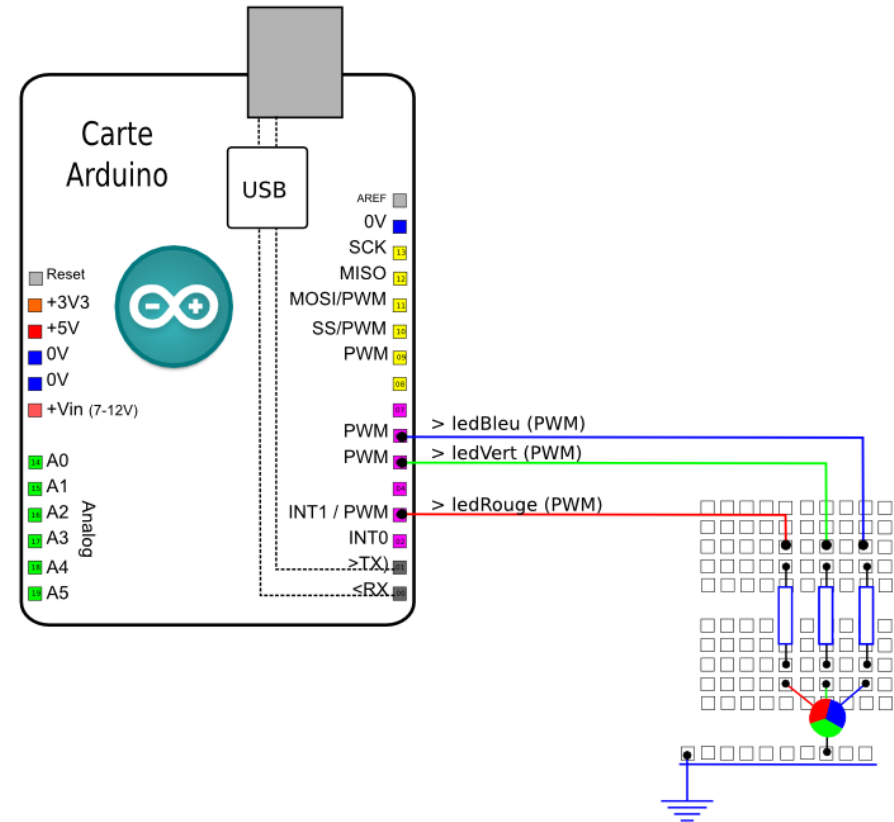
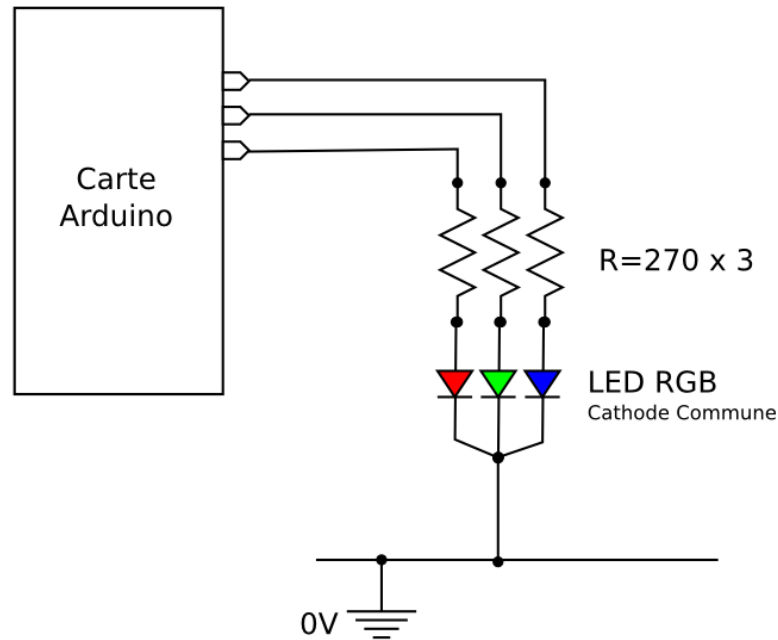
La LED multicolore RGB avec ses 3 résistances en place sur la plaque d'essai :



16. Utiliser la LED multicolore RVB avec une carte Arduino : le montage.

Le principe d'utilisation de la LED multicolore est le même que celui d'une LED, ou plutôt de 3 LEDs avec une broche commune :

- la broche commune sera connectée au 0V (LED RGB à cathode commune) ou au +5V (LED RGB à anode commune)
- chaque LED sera connectée sur une broche de E/S de la carte Arduino configurée en sortie via une résistance pour fixer l'intensité circulant dans la LED.



Montage de la LED multicolore RGB à cathode commune (0V commun)

Pour les matheux (on est pas du tout obligé de savoir faire ce calcul !) :

- aux bornes de la LED, la tension vaut 1,5V environ (fixe)
- la tension aux bornes de la résistance en série avec la LED, dans le cas d'une alimentation en 5V, vaudra donc $5V - 1,5V = 3,5V$
- si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R = U/I = 3,5V / 0,013A = 270 \text{ Ohms}$.

ATTENTION :

On va utiliser 3 broches E/S particulières ayant le signe ~ devant le numéro sur la carte Arduino : ces broches ont une fonction spéciale que nous allons utiliser, la modulation de largeur d'impulsion (MLI ou PWM en anglais).

Connecter la broche **R** sur la broche **3**, la broche **V** sur la broche **5** et la broche **B** sur la broche **6**.

17. Utiliser la LED multicolore RVB avec une carte Arduino : allumer les couleurs unitaires

On va commencer par utiliser la LED RGB comme on sait déjà le faire, c'est à dire comme 3 LEDs connectées sur 3 broches E/S de la carte Arduino configurées en sortie.

Entete déclarative

On va déclarer à ce niveau :

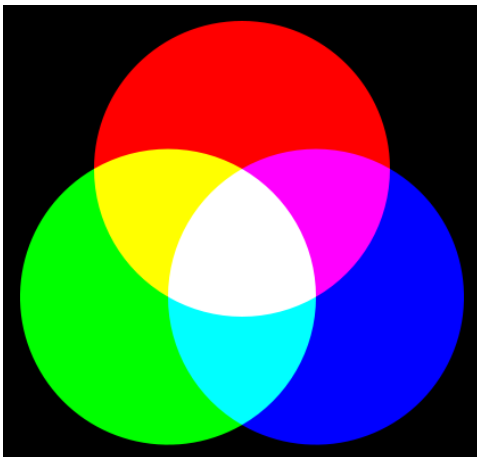
- 3 constantes désignant les 3 broches utilisées.

Fonction **setup()**

- on configure les 3 broches en sortie avec l'instruction `pinMode()`

Fonction **loop()**

- On commence par allumer la première LED pendant 1 seconde puis on l'éteint.
- On fait de même pour les autres LEDs.
- Puis on allume les LEDs 2 à 2, ce qui permet de visualiser des couleurs par combinaison des LEDs de couleurs.
- Enfin, on termine en allumant les 3 LEDs ce qui donne le blanc



On remarque ici que **les 3 LEDs de la LEDs RGB sont en quelque sorte 3 pinceaux de lumière** que l'on peut mélanger entre eux afin d'obtenir d'autre couleurs.

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

const int ledR=3; // constante désignant la broche de la LED
const int ledV=5; // constante désignant la broche de la LED
const int ledB=6; // constante désignant la broche de la LED

const boolean R=true; //--- constante binaire couleur R
const boolean V=true; //--- constante binaire couleur G
const boolean B=true; //--- constante binaire couleur B

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

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

  pinMode(ledR, OUTPUT); // met la broche en sortie
  pinMode(ledV, OUTPUT); // met la broche en sortie
  pinMode(ledB, OUTPUT); // met la broche en sortie

} // fin de la fonction setup()

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

  //--- couleur de base ---
  ledRVB(R,0,0),ledRVB(0,0,0); // rouge puis éteint
  ledRVB(0,V,0),ledRVB(0,0,0); // vert puis éteint
  ledRVB(0,0,B),ledRVB(0,0,0); // bleu puis éteint

  //---- mix ----
  ledRVB(R,V,0),ledRVB(0,0,0); // rouge + vert = jaune puis éteint
  ledRVB(R,0,B),ledRVB(0,0,0); // rouge + bleu = violet puis éteint
  ledRVB(0,V,B),ledRVB(0,0,0); // vert + bleu = bleu clair puis éteint
  ledRVB(R,V,B),ledRVB(0,0,0); // rouge + vert + bleu = blanc puis éteint

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

18. Utiliser la LED multicolore RVB avec une carte Arduino : allumer les couleurs unitaires (suite)

Fonction de gestion des LEDs RGB

Pour simplifier le code, on crée une fonction pour gérer l'affichage des LEDs :

- la fonction ne renvoie rien = type void
- reçoit 3 valeurs binaires d'activation des couleurs RVB
- à l'aide de 3 valeurs, on teste les paramètres reçu en entrée :
 - si true, on allume la LED
 - si false, on éteint la LED
- la pause est intégrée dans cette fonction

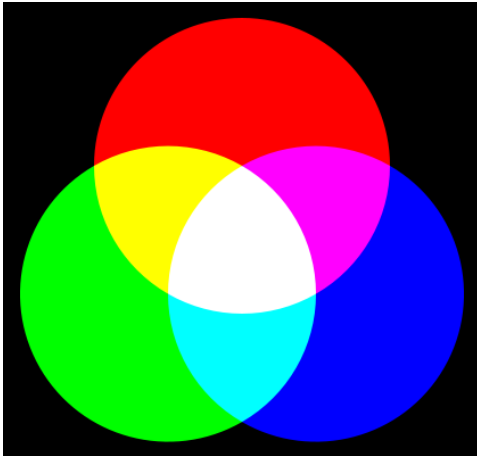
Truc : si vous utilisez une LED à anode commune, il suffit de mettre !HIGH à la place de HIGH et !LOW à la place de LOW dans le code de la fonction.

```
//----fonction pour combiner couleurs ON/OFF----  
  
void ledRVB(boolean Rouge, boolean Vert, boolean Bleu) {  
  
  if (Rouge==true) digitalWrite(ledR,HIGH); // allume couleur  
  else digitalWrite(ledR,LOW); // éteint couleur  
  
  if (Vert==true) digitalWrite(ledV,HIGH); // allume couleur  
  else digitalWrite(ledV,LOW); // éteint couleur  
  
  if (Bleu==true) digitalWrite(ledB,HIGH); // allume couleur  
  else digitalWrite(ledB,LOW); // éteint couleur  
  
  delay(vitesse); // pause de n millisecondes  
  
}
```

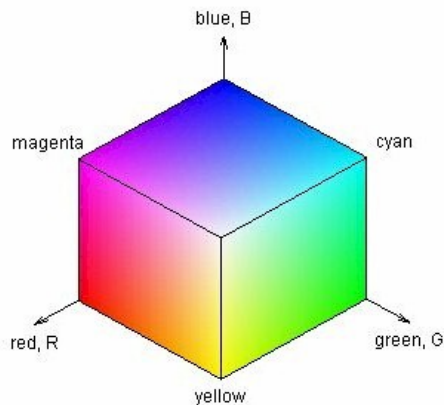


19. Le concept de couleur RGB et le cube RGB

A présent, nous allons voir comment créer toutes les variantes de couleurs avec une LED multicolore RGB. Rappelez-vous, nous avons vu qu'il était possible d'obtenir plusieurs couleurs différentes en combinant les LEDs individuelles Rouge Vert Bleu d'une LED multicolore RGB :

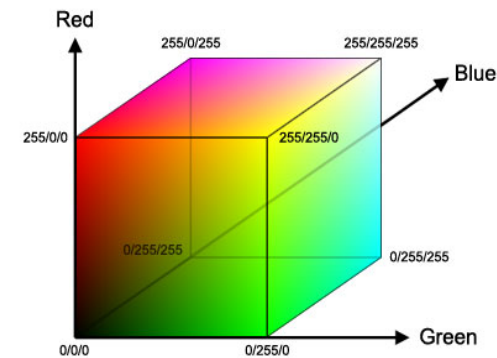


Le résultat obtenu correspond à une utilisation de chaque couleur à 100% ou à 0% (LED allumée ou éteinte). Imaginez à présent que l'on puisse doser le pourcentage de chaque couleur utilisée : nous allons alors obtenir toutes les variantes des couleurs intermédiaires, ce qui peut se représenter sous la forme d'un cube dit RGB :



Avec une LED multicolore RGB, il va être possible de doser chaque couleur à l'aide.... d'une impulsion de largeur voulue tout simplement ! Ainsi :

- en faisant varier la largeur d'impulsion de la LED rouge on fera varier la proportion de Rouge de 0% à 100% (ou de 0 à 255)
- en faisant varier la largeur d'impulsion de la LED verte on fera varier la proportion de vert de 0% à 100% (ou de 0 à 255)
- en faisant varier la largeur d'impulsion de la LED bleue on fera varier la proportion de bleu de 0% à 100% (ou de 0 à 255)



De cette façon, on dispose potentiellement de $256 \times 256 \times 256 = 16777216$: soit plus de 16 millions de nuances de couleurs différentes potentielles ! En théorie...

20. Bon à savoir : Tester le mélange des couleurs RGB simplement sur votre ordinateur

Il existe sur la plupart des systèmes d'exploitation des petits utilitaires appelés « palettes des couleurs » qui permettent de se familiariser simplement avec le système de codage des couleurs RGB. Un exemple : sous Ubuntu, le logiciel **GColor2** (accessible depuis Synaptic ou dans les dépôts).

Ce petit logiciel permet très simplement de tester l'effet des valeurs pour le rouge, le vert et le bleu. Une manière très simple de comprendre le principe du codage RGB. Le même logiciel offre également la possibilité de coder en mode HSV. Facile et efficace pour apprendre !



Dans cet exemple, le canal Vert est mis à 255 (=maximum) et les canaux rouge et bleu laissés à 0, ce qui donne du vert !

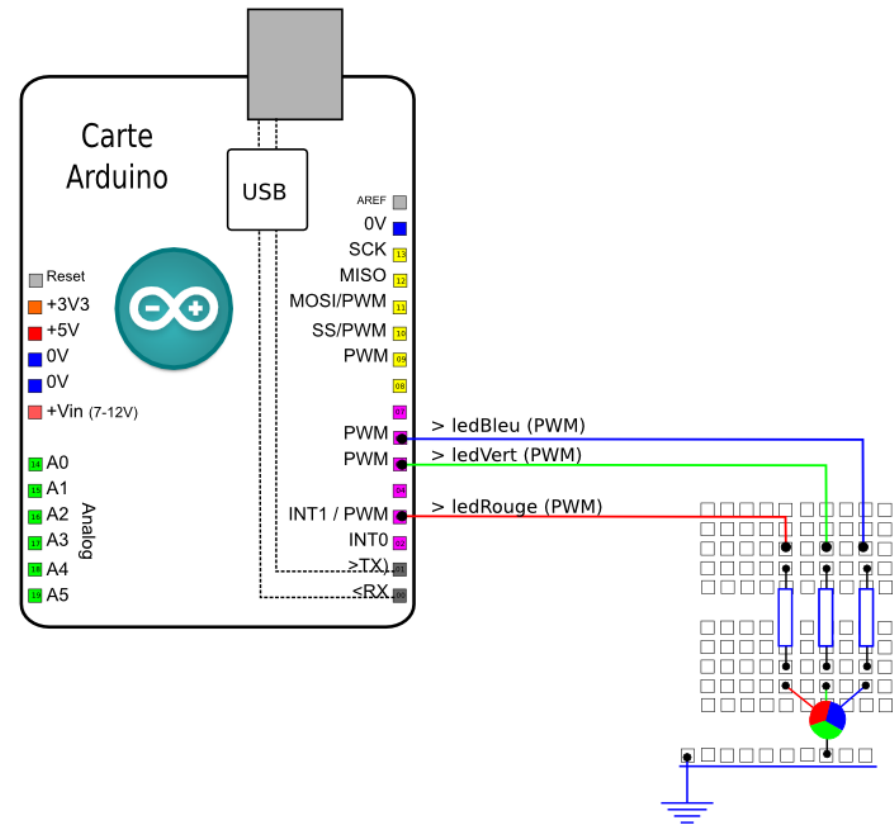
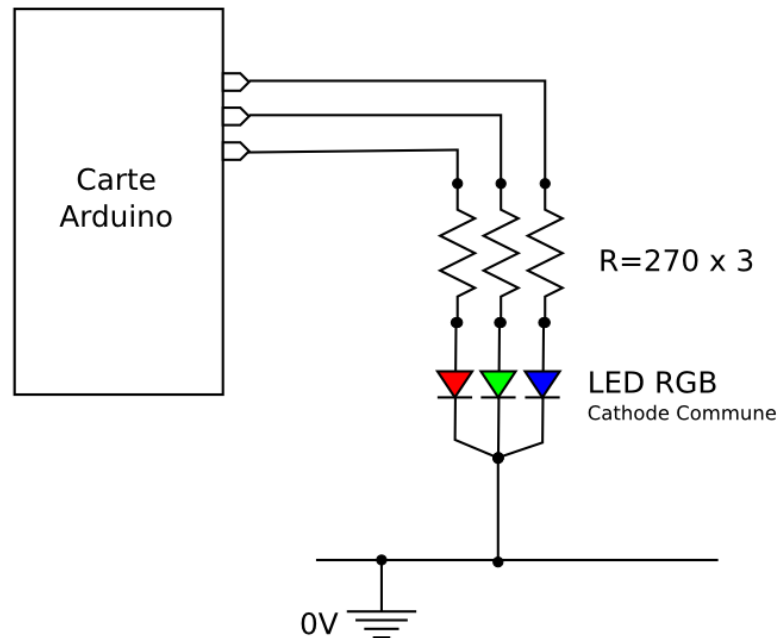
Des utilitaires équivalents existent sous Windows et Mac Os X : à vous de choisir celui que vous préférez.

Il existe même un plugin Firefox (et donc multi-Os) qui fait cela : **ColorZilla**.

21. Utiliser la LED multicolore RVB pour créer des couleurs variées avec une LED RGB : le montage

On reprend ici le montage de la LED multicolore RGB déjà vu :

- la broche commune sera connectée au 0V (LED RGB à cathode commune) ou au +5V (LED RGB à anode commune)
- chaque LED sera connectée sur une broche de E/S de la carte Arduino configurée en sortie via une résistance pour fixer l'intensité circulant dans la LED.



Montage de la LED multicolore RGB à cathode commune (0V commun)

Pour les matheux (on est pas du tout obligé de savoir faire ce calcul !) :

- aux bornes de la LED, la tension vaut 1,5V environ (fixe)
- la tension aux bornes de la résistance en série avec la LED, dans le cas d'une alimentation en 5V, vaudra donc $5V - 1,5V = 3,5V$
- si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R = U/I = 3,5V / 0,013A = 270 \text{ Ohms}$.

ATTENTION :

On va utiliser 3 broches E/S particulières ayant le signe ~ devant le numéro sur la carte Arduino : ces broches ont une fonction spéciale que nous allons utiliser, la modulation de largeur d'impulsion (MLI ou PWM en anglais).

Connecter la broche **R** sur la broche **3**, la broche **V** sur la broche **5** et la broche **B** sur la broche **6**.

22. Utiliser la LED multicolore RVB pour créer des couleurs variées avec une LED RGB : le programme.

A présent, nous allons créer des variations de couleur avec une LED multicolore RGB en se utilisant la modulation de largeur d'impulsion (ou PWM) pour doser la proportion de chaque couleur unitaire.

Entete déclarative

On va déclarer à ce niveau :

- 3 constantes désignant les 3 broches utilisées.

Fonction **setup()**

- on configure les 3 broches en sortie avec l'instruction `pinMode()`

Fonction **loop()**

- On commence par faire varier la LED rouge, puis la Bleue puis la Verte
- Ensuite, on réalise des variations basées sur le mélange de 2 couleurs
- et enfin, un mélange aléatoire des 3 couleurs

Une nouvelle fois, on remarque ici que **les 3 LEDs de la LEDs RGB sont en quelque sorte 3 pinceaux de lumière** que l'on peut mélanger entre eux afin d'obtenir d'autre couleurs.



```
const int ledR=3; // constante désignant la broche de la LED
const int ledV=5; // constante désignant la broche de la LED
const int ledB=6; // constante désignant la broche de la LED

const int R=0; //--- constante binaire couleur R
const int V=0; //--- constante binaire couleur G
const int B=0; //--- constante binaire couleur B

int vitesse=10; // variable fixant la durée de la pause

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

  pinMode(ledR, OUTPUT); // met la broche en sortie
  pinMode(ledV, OUTPUT); // met la broche en sortie
  pinMode(ledB, OUTPUT); // met la broche en sortie

} // fin de la fonction setup()

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

  //--- couleur de base ---
  for (int i=0; i<=255; i++) ledRVB(i,0,0); // variation de rouge
  for (int i=0; i<=255; i++) ledRVB(0,i,0); // variation de vert
  for (int i=0; i<=255; i++) ledRVB(0,0,i); // variation de bleu

  //---- mix ----
  for (int i=0; i<=255; i++) ledRVB(i,0,255-i); // variation de rouge +
  bleu
  for (int i=0; i<=255; i++) ledRVB(255-i,i,0); // variation de rouge +
  vert
  for (int i=0; i<=255; i++) ledRVB(0,255-i,i); // variation de vert +
  bleu

  //---- aléatoire ----
  for (int i=0; i<=50; i++) {
    ledRVB(random(0,255),random(0,255),random(0,255)); // variation au
    hasard des 3 couleurs
    delay(200);
  }

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


23. suite...

Fonction de gestion des LEDs RGB

Pour simplifier le code, on crée une fonction pour gérer l'affichage des LEDs :

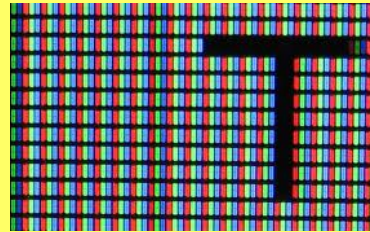
- la fonction ne renvoie rien = type void
- reçoit 3 valeurs entières (0-255) fixant la proportion des couleurs RVB
- à l'aide de 3 valeurs, on crée les impulsions PWM correspondantes
- la pause est intégrée dans cette fonction

Truc : si vous utilisez une LED à anode commune, il suffit de mettre 255-valeur à la place de valeur

```
//---autresfonctions
//----fonction pour combiner couleurs----
void ledRVB(int Rouge, int Vert, int Bleu) {
  analogWrite(ledR,Rouge); // proportion de rouge
  analogWrite(ledV,Vert); // proportion de vert
  analogWrite(ledB,Bleu); // proportion de bleu

  delay(vitesse); // pause de n millisecondes
}
```

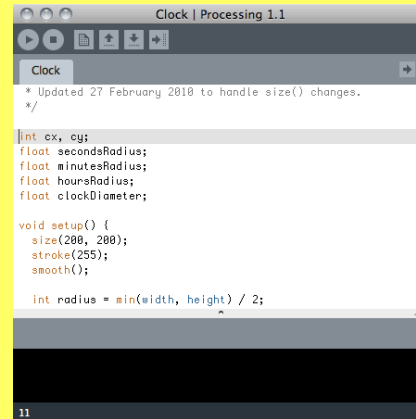
Si vous avez compris le codage des couleurs RGB à l'aide de 3 valeurs, alors vous avez les connaissances nécessaires pour travailler avec des images numériques sur ordinateur : chaque pixel d'une image couleur fonctionne exactement de la même façon, par combinaison du Rouge, du Vert et du Bleu !



Zoom sur un écran d'ordinateur : remarquer les pixels RGB !



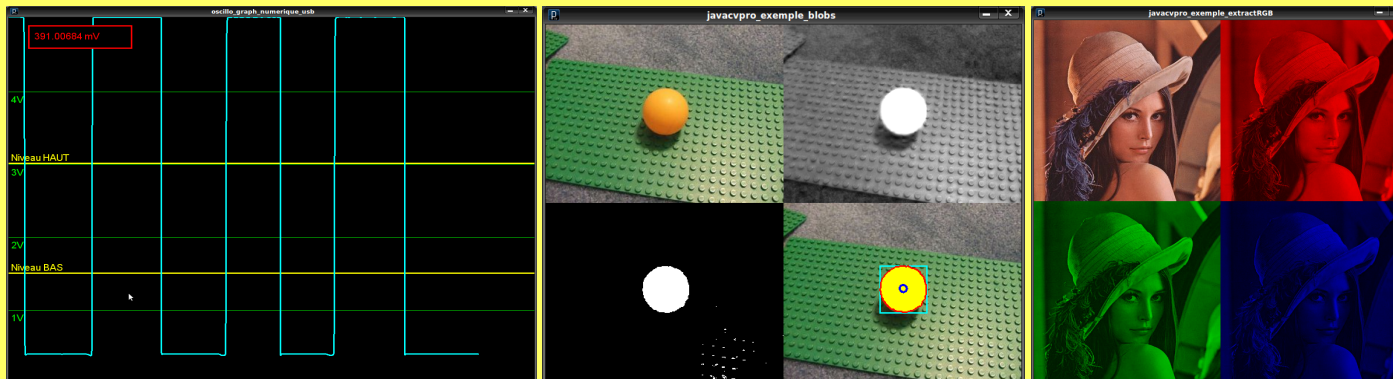
24. Bon à savoir : manipuler et modifier des images couleurs par programmation avec Processing !



Le logiciel Processing, libre et opensource, créé par le MIT, est une interface écrite en Java qui est un véritable couteau suisse ! Grâce à ce logiciel, vous allez pouvoir simplement, par programmation, créer des dessins 2D statiques ou animés, faire du traitement d'image en gérant les images numériques au niveau du pixel, **mais aussi communiquer avec la carte Arduino !!**

(En fait, le logiciel Arduino est basé sur une interface Processing au départ)

Si cela vous intéresse, vous allez pouvoir réaliser des interfaces graphique pour Arduino, du traitement d'image de base ou même de la vision par ordinateur !!



Voir ici : Processing.org et www.mon-club-elec.fr

25. Les éléments du langage Arduino étudiés dans cet atelier

Structure

Variables et constantes

Fonctions

Sortie analogique (impulsion)

- [analogWrite](#)(broche, valeur)

Temps

- [delay](#)(ms)
- unsigned long [millis](#)()
- unsigned long [micros](#)()
- [delayMicroseconds](#)(us)

Sons

- [tone](#)()
- [noTone](#)()

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 :*

- produire des sons simples de fréquences variées ou un buzzer piézo-électrique.
- créer des impulsions de largeur variable
- utiliser une LED multicolore RGB

Table des matières

Sorties « analogiques » et impulsions : produire des sons, générer des impulsions « analogiques » (PWM), contrôler une LED multicolore (RGB).

Intro |

Matériel nécessaire pour les ateliers Arduino |

Rappel de physique : le principe des sons |

Bon à savoir |

Fiche composant : Découvrir le buzzer piézo-électrique et son utilisation avec Arduino |

Produire un simple son avec une carte Arduino |

Produire un son de fréquence voulue avec une carte Arduino |

Produire une simple mélodie avec une carte Arduino |

Un truc bon à connaître : inclure un fichier dans votre code ! |

Générer un son de sirène de police avec une carte Arduino |

Le concept de modulation de largeur d'impulsion (MLI) |

Faire varier la luminosité d'une LED : le montage |

Faire varier la luminosité d'une LED : le programme |

Fiche composant : Découvrir la LED multicolore RGB |

Truc technique : préparer la LED multicolore RGB pour une utilisation simplifiée sur plaque d'essai |

Utiliser la LED multicolore RVB avec une carte Arduino : le montage. |

Utiliser la LED multicolore RVB avec une carte Arduino : allumer les couleurs unitaires |

Utiliser la LED multicolore RVB avec une carte Arduino : allumer les couleurs unitaires (suite) |

|

Le concept de couleur RGB et le cube RGB |

Bon à savoir : Tester le mélange des couleurs RGB simplement sur votre ordinateur |

Utiliser la LED multicolore RVB pour créer des couleurs variées avec une LED RGB : le montage |

Utiliser la LED multicolore RVB pour créer des couleurs variées avec une LED RGB : le programme. |

suite... |

Bon à savoir : manipuler et modifier des images couleurs par programmation avec Processing ! |

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