

NIVEAU DEBUTANT

Sorties numériques : découvrir et apprendre à utiliser les broches de la carte Arduino en sorties numériques.



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 comprendre la notion d'électronique numérique
- de découvrir le fonctionnement d'une broche numérique
- de connaître les broches numériques de la carte Arduino
- d'apprendre les instructions du langage Arduino permettant de contrôler les broches numériques
- de savoir écrire un programme utilisant une broche numérique
- de rappeler quelques bases fondamentales d'électricité
- d'apprendre les caractéristiques électriques de la carte Arduino
- d'apprendre à faire un montage sans soudure avec une plaque d'essai
- d'apprendre à utiliser les broches E/S en sortie
- d'apprendre à contrôler 1 à 4 LEDs avec des broches numériques en sortie
- d'apprendre à utiliser les tableaux de variables

... afin d'être en mesure de pouvoir utiliser les broches numériques en sortie et de réaliser par vous-même des montages/programmes utilisant de 1 à 4 LEDs.

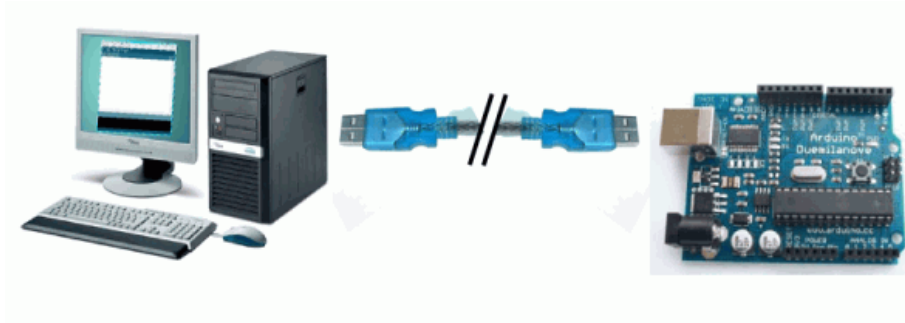


Prêt ? C'est parti !

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

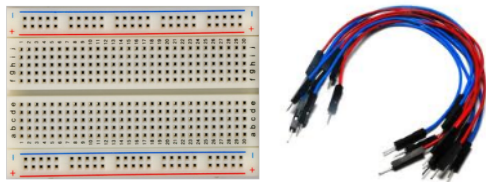


L'espace de développement Arduino associe :

- 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

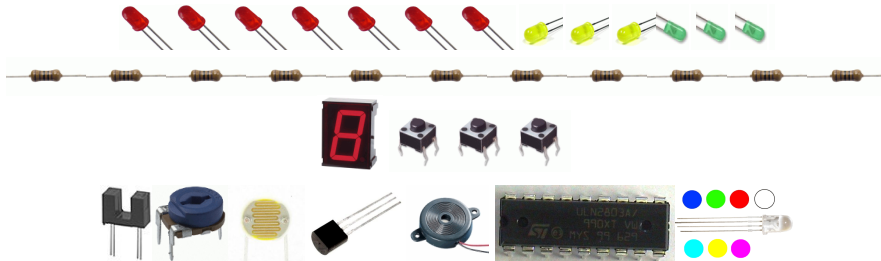


Pour réaliser des montages sans soudure, vous aurez besoin :

- 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. Notion d'électronique numérique

L'électronique est une technique qui manipule les « électrons » sous forme de tension ou d'intensité. On distingue 2 types d'électronique :

- L'électronique **analogique** qui utilise des **variations continues** de la tension qui peut prendre toutes les valeurs intermédiaires (potentiomètre = variation du minimum au maximum).
- L'électronique **numérique** qui utilise des **variations « abruptes »** de la tension qui va prendre 2 niveaux (interrupteur = allumé ou éteint) : l'un dit HAUT (5V), l'un dit BAS (0V).

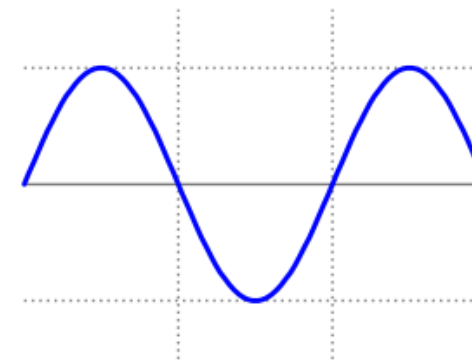
Un microprocesseur, tel que celui de la carte Arduino, est un circuit numérique qui va permettre de manipuler des niveaux HAUT/BAS.

Pour prendre une image de tuyauterie :

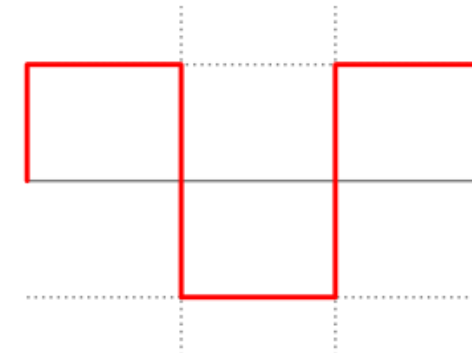
- L'électronique **analogique** est comparable à un **robinet** dont l'écoulement va pouvoir varier entre l'absence d'eau et le débit maximum.
- L'électronique **numérique** est comparable à une **vanne « tout ou rien »** qui va soit stopper l'écoulement, soit donner le débit maximum, les écoulements intermédiaires n'étant pas possibles.

L'intérêt majeur d'utiliser 2 niveaux HAUT/BAS est de permettre :

- De **compter et calculer** en combinant les niveaux HAUT et BAS entre-eux : c'est le comptage binaire, qui utilise les 0 et les 1,
- De **commander / contrôler** des dispositifs à partir de plusieurs niveaux HAUT / BAS combinés entre eux,
- De **communiquer** des informations entre 2 circuits numériques en envoyant des niveaux successifs de HAUT/BAS
- De **numériser** des signaux analogiques (conversion analogique-numérique) !
- De **coder des instructions** à exécuter par un microprocesseur.



Analogique



Numérique

4. Une broche numérique ne peut avoir que 2 états : HAUT ou BAS, « y'a ou y'a pas » !

Une broche numérique, dans un circuit numérique est un point du circuit matérialisé par une broche métallique dans le cas d'un circuit intégré ou d'une carte électronique.

Une broche numérique va être caractérisée par son **état** ou niveau de tension : elle va pouvoir se trouver dans **2 états possibles seulement** :

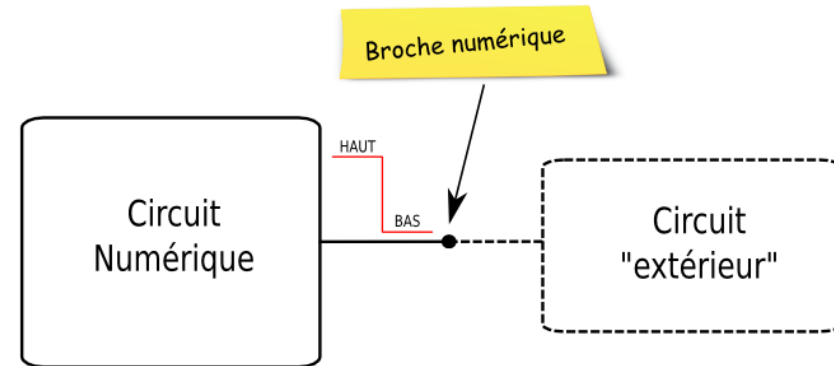
- soit au niveau **HAUT** (=5V), symbolisé par **1** ou **HIGH**
- soit au niveau **BAS** (=0V), symbolisé par **0** ou **LOW**
- noter qu'à **un instant quelconque, la broche se trouve obligatoirement dans l'un de ces 2 états.**

Pour reprendre l'image d'une vanne « tout ou rien » :

- soit il y a de l'eau qui circule
- soit il n'y en n'a pas

Pour reprendre l'image d'un « interrupteur » :

- soit il y a de la lumière,
- soit il n'y en n'a pas...



5. Une broche numérique est caractérisée par son SENS : en SORTIE ou en ENTREE !

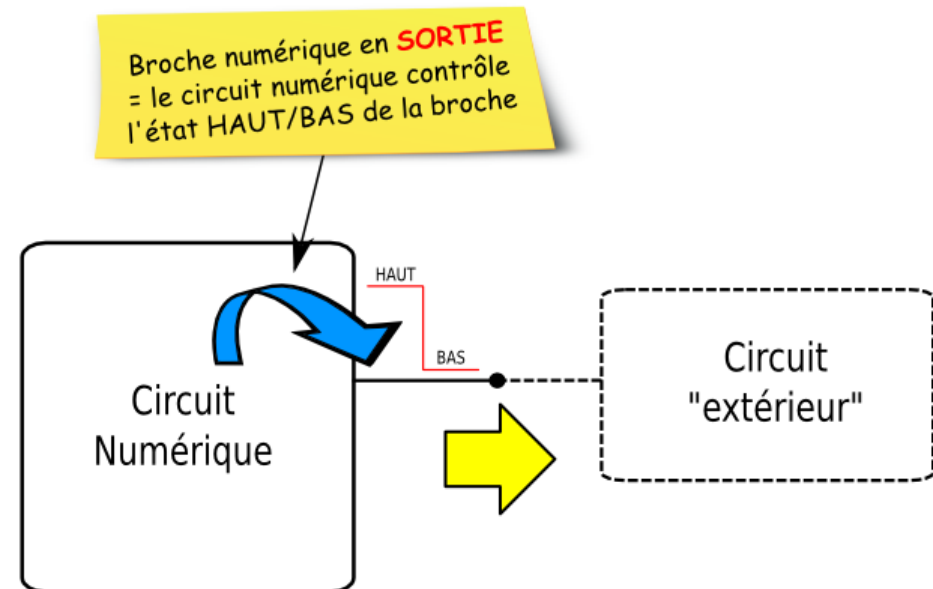
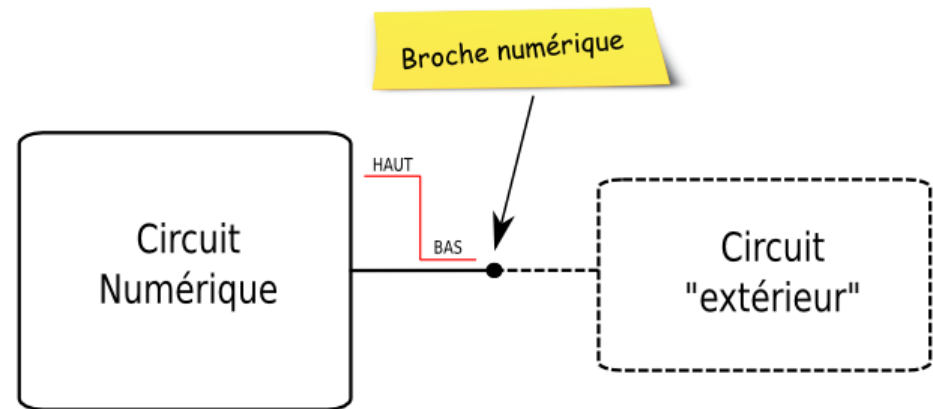
Une broche numérique est également caractérisée par son **sens** :

- la broche est dite en **sortie** d'un circuit numérique lorsque c'est **le circuit numérique qui contrôle l'état Haut/Bas de la broche**. On pourra symboliser le sens de la broche numérique en sortie par une flèche sortante.
- la broche est dite en **entrée** d'un circuit numérique lorsque le circuit numérique « reçoit » (ou « subit ») son état. **L'état Haut/Bas de la broche numérique est contrôlé par « l'extérieur »**. On pourra symboliser le sens de la broche numérique en entrée par une flèche entrante.
- noter qu'une broche numérique qui est en sortie d'un circuit numérique est en entrée du circuit extérieur auquel elle est connectée... et inversement... !

Usage avancé : en interne, dans un microprocesseur, une broche numérique est associée :

- à un bit de donnée (case unitaire mémoire) qui va permettre de fixer/lire son état
- à un bit de sens qui va définir son mode de fonctionnement

Techniquement, il existe par ailleurs plusieurs technologies de broches numériques (TTL, CMOS,..) qui ont des définitions différentes des niveaux de tension HAUT et BAS. Seules des broches compatibles entre-elles pourront être utilisées/connectées ensemble. Arduino est très souple de ce point de vue et est compatible avec la plupart des technologies E/S !



6. Les broches numériques de la carte Arduino

La carte Arduino de base (la UNO, la Duemilanove, etc..) est une carte numérique qui possède **20 broches d'Entrée/Sortie numérique** (notées E/S) numérotées **de 0 à 19** !

Les broches 0 à 19 sont potentiellement utilisables en broches E/S !

Cependant, certaines broches ne doivent pas, dans la mesure du possible être utilisées en broches E/S :

- les **broches 0 et 1** sont utilisées par la communication USB donc, les utiliser pourrait perturber cette communication. En pratique, ne pas les utiliser.
- les **broches 14 à 19** ont un double rôle : elles peuvent également être utilisées en tant que broches analogiques pour réaliser des mesures. Donc, si possible, ne pas les utiliser en broches numériques... mais si on est obligé, on peut le faire !

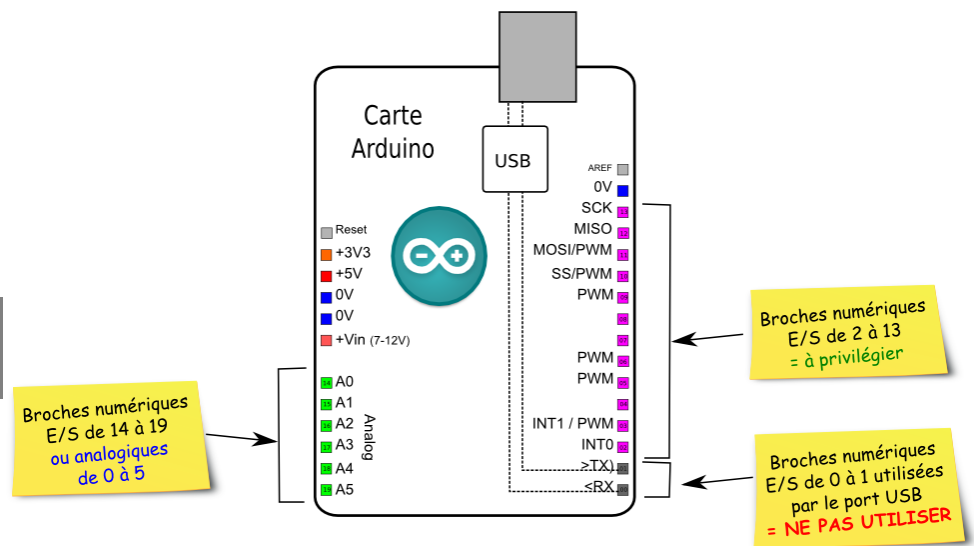
A savoir : les broches numériques 14 à 19 sont numérotées de 0 à 5 (ou désignées par A0, A1, A2, A3 et A5) lorsqu'elles sont utilisées en tant que broches analogiques.

Remarquer également que la plupart des broches numériques ont des fonctions particulières potentielles qui seront présentées au fur et à mesure de leur utilisation. *A titre indicatif, les fonctions disponibles sont la génération d'impulsion, la communication SPI, la communication I2C, les interruptions externes...*

D'un point de vue électrique, retenir que :

- chaque broche numérique E/S peut supporter 40 mA d'intensité en sortie ou en entrée
- L'ensemble des broches numériques E/S ne doit pas dépasser 200mA en entrée ou en sortie !

Usage avancé : pour des projets nécessitant de nombreuses broches E/S (= mal conçu?), la carte Arduino Mega dispose de plus d'une 50aine de broches E/S !

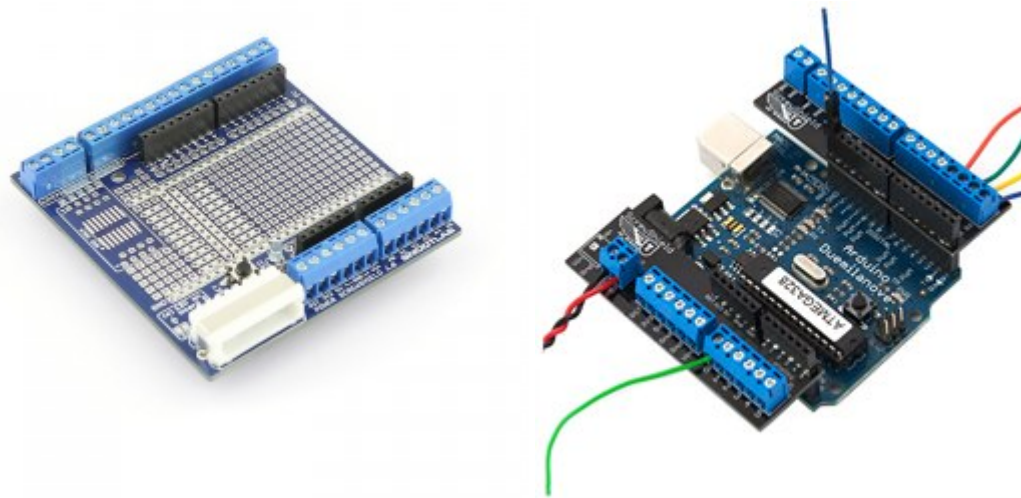


7. *Truc technique : les broches E/S de la carte Arduino sur borniers à vis avec un screwshield.... !*

Bon à savoir :

Il existe des shields (carte d'extension) de la carte Arduino qui permettent de dédoubler les broches de la carte Arduino sur des borniers à vis : **les « screwshields » !**

Très pratique pour finaliser des montages en « dur ».



2 exemples de « screwshields » : le Powerscrewshield de Snootlab et le un screwshield en 2 parties.

8. Les instructions du langage Arduino pour la gestion des broches numériques

On a donc vu qu'une broche numérique E/S est caractérisée par :

- son sens : **ENTREE** ou **SORTIE**
- son état : HAUT ou BAS

Fixer le sens E/S d'une broche numérique

La première instruction à connaître est l'instruction `pinMode(broche, mode)` qui sert à fixer le sens (ou mode de fonctionnement) d'une broche numérique E/S avec :

- broche : le numéro de la broche de 0 à 19
- mode : une des constantes prédéfinies suivantes :
 - **OUTPUT** : pour un fonctionnement en **sortie**
 - **INPUT** : pour un fonctionnement en **entrée**
- Cette fonction est à utiliser dans la fonction `setup()`

Fixer le niveau HAUT/BAS d'une broche numérique

Si la broche est configurée en **SORTIE**, on pourra contrôler son état (ou « écrire » sur la broche) à l'aide de la fonction `digitalWrite(broche, etat)` avec :

- broche : le numéro de broche de 0 à 19
- valeur : une des constantes prédéfinies suivantes :
 - **HIGH** : pour mettre la broche au niveau HAUT (5V)
 - **LOW** : pour mettre la broche au niveau BAS (0V)

Si la broche est configurée en **ENTREE**, on pourra « lire » son état à l'aide de la fonction `int digitalWrite(broche)` avec :

- broche : le numéro de broche de 0 à 19
- int : la valeur renvoyée par la fonction de type int

	ETAT HAUT	ETAT BAS
BROCHE EN SORTIE <code>pinMode(broche, OUTPUT);</code>	<code>digitalWrite(broche, HIGH);</code>	<code>digitalWrite(broche, LOW);</code>
BROCHE EN ENTREE <code>pinMode(broche, INPUT);</code>	<code>digitalRead(broche);</code>	<code>digitalRead(broche);</code>

9. *Ecrire un programme qui met une broche en sortie au niveau HAUT*

Maintenant que l'on connaît les instructions de gestion des broches E/S, on est capable d'écrire un programme pour mettre en sortie une broche et pour la mettre au niveau HAUT. Dans notre exemple, nous utiliserons la broche 13 qui dispose d'une LED intégrée sur la carte Arduino. Reprenons la structure type d'un programme :

Entête déclarative

Aucune variable à déclarer ici.

Fonction setup()

A ce niveau, on va :

- initialiser la broche en sortie avec l'instruction `pinMode()`
- mettre la broche à HAUT avec l'instruction `digitalWrite()`. A noter que la LED connectée à la broche s'allumera si le niveau est HAUT.

Fonction loop()

Laissée vide.

Voici le code complet :

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

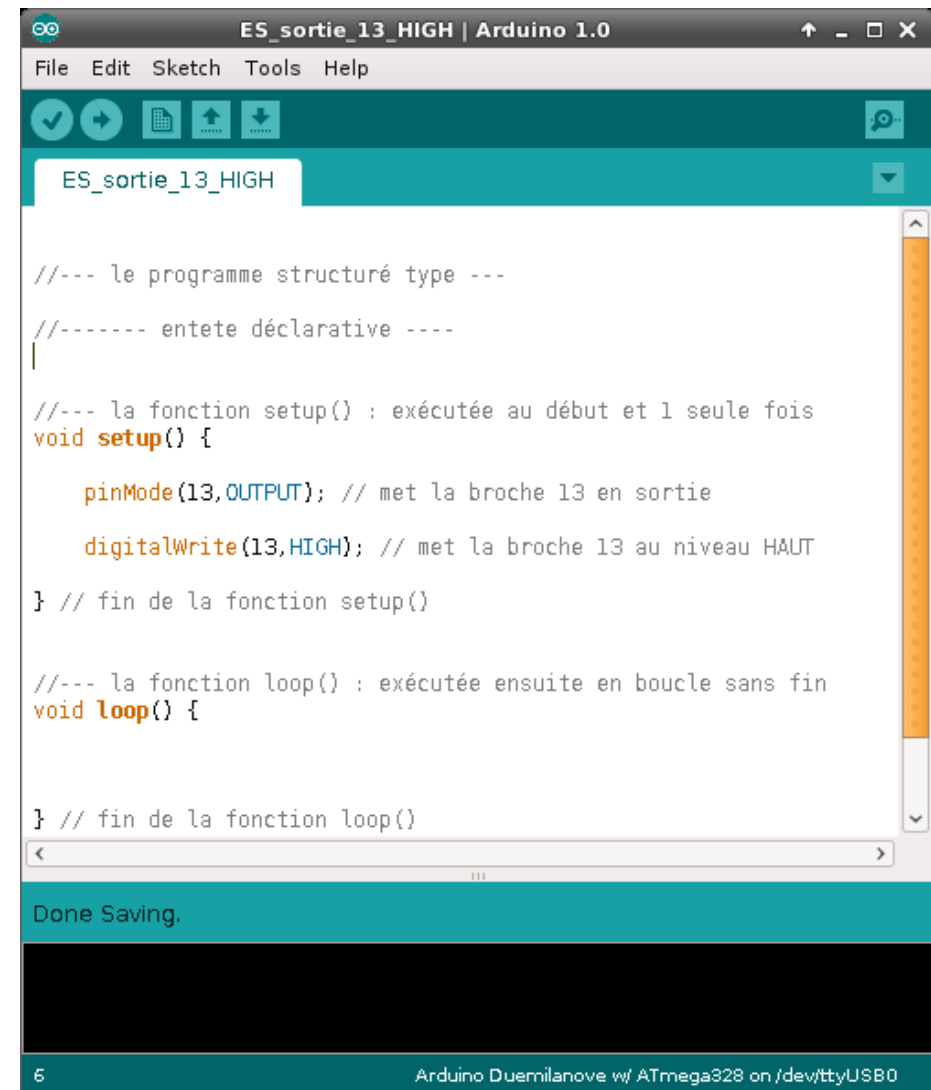
    pinMode(13,OUTPUT); // met la broche 13 en sortie

    digitalWrite(13,HIGH); // met la broche 13 au niveau HAUT

} // fin de la fonction setup()

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

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



10. Notion d'électricité élémentaire à bien connaître

Pour faire de l'électronique numérique avec Arduino, il n'y a pas beaucoup de règles d'électricité à connaître, mais il y en a une qui est très importante à connaître (elle vous servira tout le temps !!) :

TOUT « DISPOSITIF » ELECTRIQUE SE CARACTERISE PAR 2 CHOSES :

UNE TENSION D'ALIMENTATION mesurée en VOLTS

UNE INTENSITE DE FONCTIONNEMENT mesurée en AMPERES

La règle qui lui est associée et qui est tout aussi importante :

TOUTE « ALIMENTATION » ELECTRIQUE SE CARACTERISE PAR 2 CHOSES :

UNE TENSION FOURNIE mesurée en VOLTS

UNE INTENSITE MAXIMALE mesurée en AMPERES

Ainsi, avant de connecter un dispositif sur une alimentation, il va falloir systématiquement se poser 2 questions :

- la tension fournie par l'alimentation correspond-elle à la tension d'alimentation du dispositif ?
- l'intensité de fonctionnement du dispositif est-elle bien inférieure à l'intensité maximale possible ?

Il faudra toujours répondre OUI à ces 2 questions. Exemples :

- puis-je connecter un appareil fonctionnant en 12V et consommant 1A sur une prise de courant de 220V / 10A ? => **NON (danger!)**
- puis-je connecter un moteur fonctionnant en 6V et consommant 2A sur un bloc secteur 6V/500mA ? => **NON**
- puis-je connecter une lampe consommant 5V/20mA sur une broche pouvant fournir 5V/40mA ? => **OUI**

Exemple de fiche technique d'un moteur.



Moteur: RE280/1
Alimentation: 12 à 24 Vcc
Consommation à vide: 0,10 A à 12 Vcc
Consommation en charge: 0,30 A à 12 Vcc

Exemple de fiche technique d'un bloc secteur

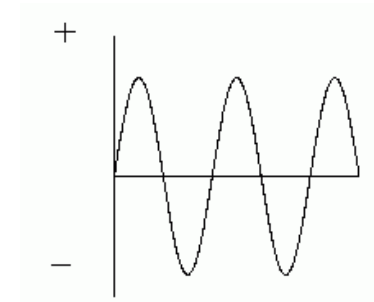


Tension d'entrée: 230 Vac - 50 Hz
Tension de sortie régulée: 12 Vcc
Courant maxi : 1000 mA

11. Notion de tension alternative, continue, régulée..

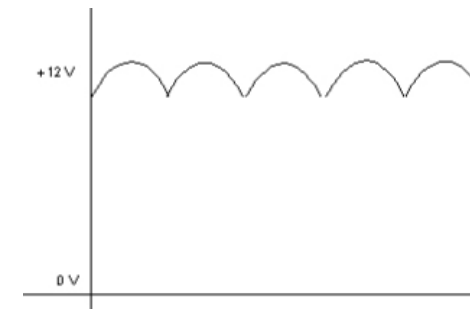
Tension alternative = celle du secteur 220V

La tension du secteur est une tension qui varie tout le temps en faisant des « vagues » : la tension est dite sinusoïdale ou alternative.



Tension continue non régulée = celle d'un bloc secteur 12V

La plupart des dispositifs « basse tension » tels que les moteurs et autres petits appareils alimentés par une alimentation externe (bloc secteur) utilise une alimentation continue mais qui n'est pas parfaitement « plane » avec quelques variations possibles autour de la tension théorique prévue.



Tension continue régulée = celle des circuits numériques

Les circuits numériques tels qu'une carte Arduino, ou la carte-mère (interne) d'un ordinateur, utilisent et nécessitent des tensions continues **PARFAITEMENT STABLES** qui sont obtenues à l'aide d'un composant appelé régulateur : ce sont des **alimentations dites régulées**.

Les tensions régulées les plus utilisées en électronique numérique sont le **5V**, le **12V** et le 3.3V. La carte Arduino standard fonctionne en 5V régulé.



La tension Vin d'une carte Arduino pourra être de type continu régulé ou non, la carte Arduino disposant d'un régulateur 5V intégré.

12. Truc technique... : une alimentation régulée de « labo » à pas cher !

BON à savoir :

une alimentation de PC de récupération, dite alimentation ATX, est une alimentation régulée peu coûteuse (<10€) et qui fournit du 5V régulé sous plusieurs ampères, du 12V régulé sous plusieurs ampères, et aussi du 3.3V régulé, du -12V régulé, etc.. Pratique !

A comparer à un bloc secteur 220V AC / 6-12V DC qui ne fournira que 0.5 à 1A dans le meilleur des cas, pour le même prix...

Avec ça, vous êtes sur de pouvoir alimenter tous vos projets, y compris (et surtout) si vous utilisez des moteurs ou de nombreux servomoteurs !



Pour activer l'alimentation, au niveau du connecteur ATX, il suffit de mettre un strap entre la broche verte (16) et la masse (0V - noir)

+3,3 VDC	13	1	+3,3 VDC
-12 VDC	14	2	+3,3 VDC
Masse	15	3	Masse
PS_ON	16	4	+5 VDC
Masse	17	5	Masse
Masse	18	6	+5 VDC
Masse	19	7	Masse
-5 VDC	20	8	Power OK
+5 VDC	21	9	+5 VSB
+5 VDC	22	10	+12 V1DC
+5 VDC	23	11	+12 V1DC
Masse	24	12	+3,3 VDC

13. Caractéristiques électriques globales de la carte Arduino

La carte Arduino standard peut être alimentée de 2 façons :

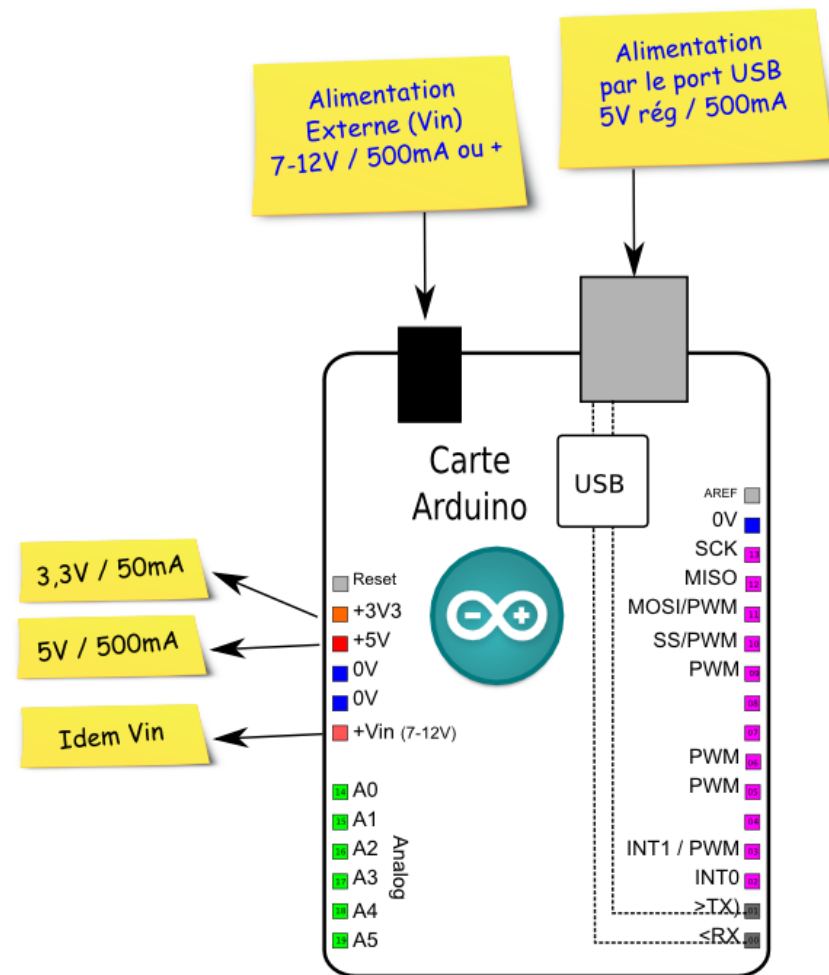
- soit directement par le port USB (qui fournit 5V/500mA régulé)
- soit par une alimentation continue externe via le connecteur d'alimentation (cette source s'appelle Vin) :
 - tension continue, régulée ou non, entre 7 et 12V conseillé (jusqu'à 20V maxi) (**La carte Arduino dispose d'un régulateur 5V intégré**)
 - pouvant fournir au moins 500mA, ou plus...

La carte Arduino dispose de d'un connecteur d'alimentation qui fournit plusieurs tensions :

- **Vin** qui est une reprise de la tension Vin connectée au connecteur d'alimentation externe et a les mêmes caractéristiques que l'alimentation utilisée. A noter que Vin est régulé si l'alimentation utilisée est régulée (Alimentation de PC)
- **5V** qui est du 5V régulé jusqu'à 500mA maximum, utilisable pour alimenter les capteurs, modules utilisés avec la carte Arduino,
- **3V3** régulé jusqu'à 50mA (attention faible intensité max) et utilisable pour alimenter certains composants nécessitant du 3V3. Peu utilisé en pratique... mais il est là si besoin...

La tension Vin d'une carte Arduino pourra être de type continu régulé ou non, la carte Arduino disposant d'un régulateur 5V intégré.

En pratique, pour alimenter votre carte Arduino avec une alimentation externe, utiliser un bloc 220V => 6-12V cc /500mA ou mieux 6-12V cc / 1A.



14. Caractéristiques électriques d'une broche Numérique Arduino en sortie

Une broche numérique Arduino individuelle en sortie peut-être considérée **comme une « mini »-alimentation** :

- fournissant une tension de **5V régulé** au niveau HAUT et 0V au niveau BAS.
- pouvant fournir **au maximum 40mA** pour une seule broche en sortie.

Pour l'ensemble des broches numériques :

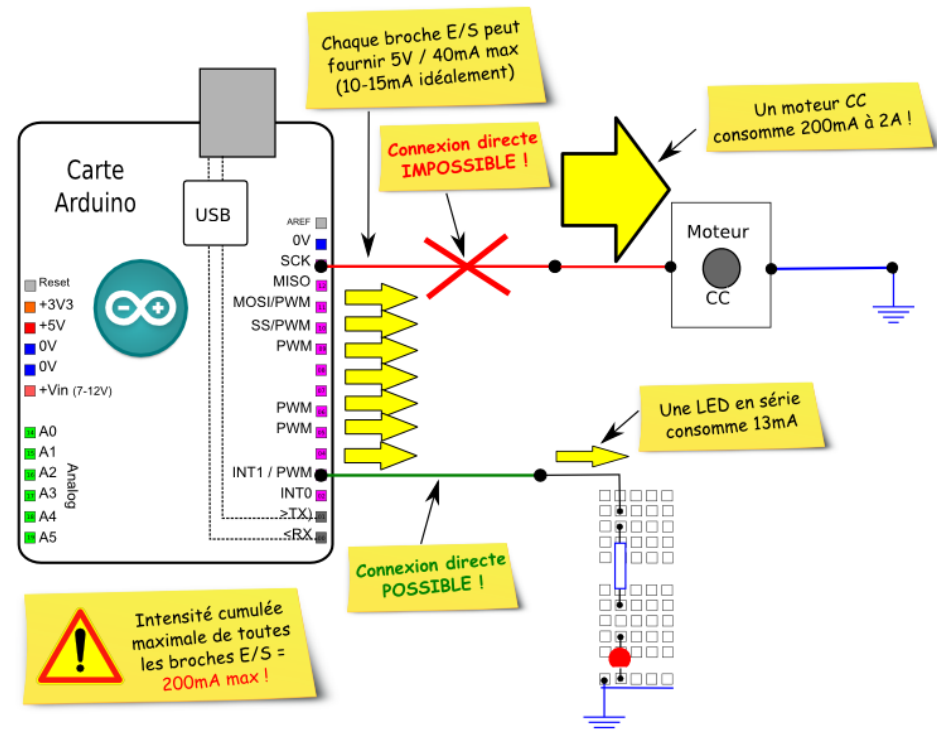
- **l'intensité maximale cumulée ne doit pas dépasser les 200mA.**
- Ainsi, pour éviter les soucis, dans l'hypothèse où l'on utilisera au maximum une 15aine de broches en sortie simultanément, considérer que l'on peut utiliser sans danger $200\text{mA}/15 = 13\text{mA}$ / broche soit en pratique 10 à 15mA par broche.

En pratique : considérer qu'une broche Arduino fournit 5V / 10-15mA

Voici quelques exemples, pour se faire une idée de ce que l'on peut connecter directement sur une broche numérique d'une carte Arduino en sortie :

- une broche d'un autre CI numérique consommera dans les 1 mA, => **connexion directe POSSIBLE !**
- une LED en série avec sa résistance consommera dans les 10-20mA selon la résistance utilisée, => **connexion directe POSSIBLE !**
- la broche de commande d'un servomoteur consommera dans les 5mA, => **connexion directe POSSIBLE !**
- un moteur CC consommera dans les 250mA => **connexion directe IMPOSSIBLE !** (Dans ce cas, on devra utiliser une interface de puissance comme nous le verrons)

En pratique : TOUJOURS se demander « quelle intensité va être utilisée ? »



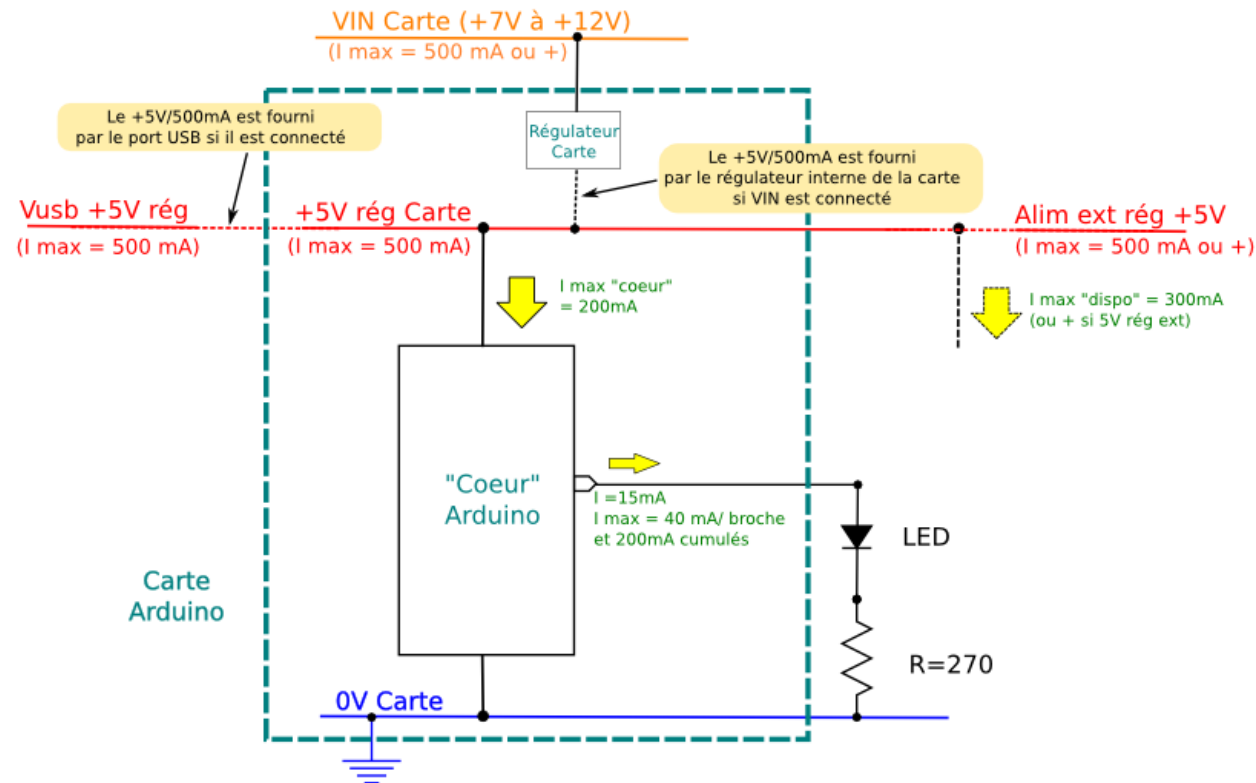
15. Technique : Le schéma électrique interne de l'alimentation de la carte Arduino

Afin de comprendre ce que l'on fait, il est important d'avoir toujours à l'esprit les notions d'intensité et de tension de la carte Arduino. Comme on l'a dit, la carte Arduino intègre une alimentation interne régulée de **5V rég / 500mA** :

- soit en provenance du port USB
- soit à partir de l'alimentation Vin 7-12V / 500mA (ou +) (connecteur Jack)

Il est essentiel de distinguer :

- **l'intensité maximale (200mA) que peut fournir le « coeur »** de la carte Arduino et limité à **40mA par broche** en sortie mais **200mA maximum** pour l'ensemble des broches réunies. **Une LED en série avec une résistance utilisera par exemple 15mA fournis par le « coeur » Arduino.**
- **l'intensité maximale (500mA) que peut fournir l'alimentation +5V régulé/500mA.** Cette alimentation de +5V/500mA de la carte Arduino peut également être mise en parallèle avec une alimentation externe +5V régulée au besoin.
- **l'intensité résiduelle disponible de 300mA supplémentaires maxi pour alimenter des dispositifs 5V directement à partir de l'alimentation de la carte Arduino (mais pas à partir des broches !!)**



16. Découvrir les composants et accessoires de base pour faire des montages avec la carte Arduino

La LED = Light Emitting Diode (ou DEL : diode électroluminescente)

Ce composant produit de la lumière. C'est une diode et donc a un sens de connexion.



La résistance

Ce composant sert à limiter l'intensité, c'est à dire le nombre d'électrons qui circulent dans le circuit. Sens de connexion indifférent. Se mesure en Ohms. Les bandes colorées indiquent la valeur de la résistance.



Jumper ou Straps

Câble souple ou semi-rigide permettant de réaliser des connexions entre 2 composants. Existents dans différentes tailles et couleurs. Existents en Mâle/Mâle, Femelle/Femelle, Mâle/Femelle.



Plaque d'essai ou « breadboard »

Plaque permettant de réaliser simplement des montages électroniques sans soudure.



Il existe de quelques autres composants que vous découvrirez au fur et à mesure.

A la différence de l'électronique analogique, en électronique numérique, l'utilisation des composants peut être simplifiée au maximum.

17. Découvrir la résistance et son principe d'utilisation

La résistance est un composant qui réalise une sorte de « goulot d'étranglement » pour les électrons : elle sert à **limiter l'intensité qui circule dans le circuit**.

La résistance :

- se mesure en Ohms , le symbole du Ohm est le Ω (Omega),
- cette valeur est indiquée par des bandes de couleurs présentes sur la résistance et qui correspondent à un code de couleur,
- peut aussi être mesurée avec un Ohmètre (ou multimètre)
- n'a pas de sens de connexion particulier
- existe avec plusieurs degrés de précision (5%, 1%..) et de puissance (1/4w, 1W, etc..) : en pratique on utilise des résistances carbone 5% en 1/4w

Pas besoin de retenir le code des couleurs !!

En pratique, on peut faire de l'électronique avec Arduino en n'utilisant que 3 ou 4 valeurs de résistances que l'on finit par connaître par cœur.

Nous utiliserons souvent la résistance 270 Ohms (rouge – violet – marron)

On peut monter une résistance avec un autre composant :

- en série = à la queue leu leu
- en parallèle = côte à côte

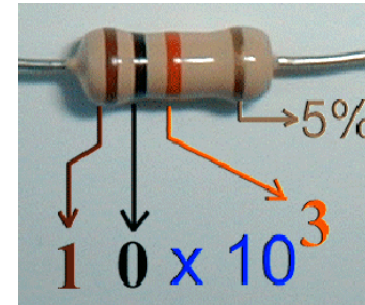
Symbolisée par :



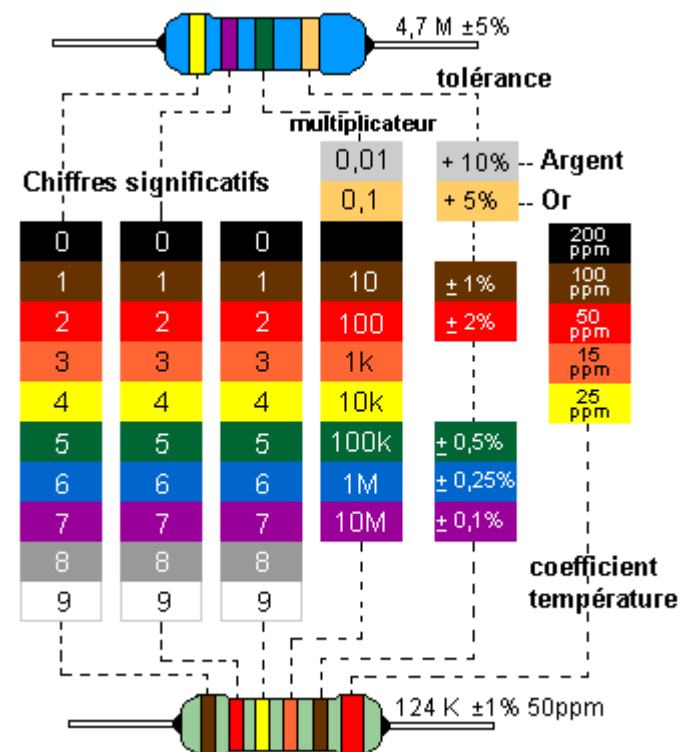
La relation mathématique fondamentale pour la résistance est la loi d'Ohm qui dit que :

- la tension aux bornes de la résistance est égale à la résistance multipliée par l'intensité,
- soit $U = R \times I$

En pratique, vous n'avez pas besoin de savoir calculer la valeur d'une résistance à utiliser dans un circuit, mais c'est bien si vous savez le faire...



Résistance de $10 \times 1000 = 10 \text{ k}\Omega$ (dire « dix kilo-ohms »)



18. Découvrir la LED et son principe d'utilisation

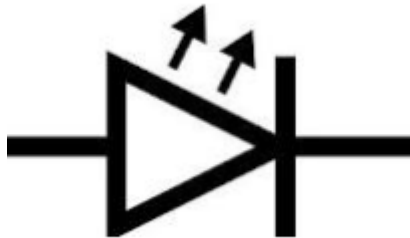
La LED (Light Emitting Diode) est un composant électronique qui comme tout le monde le sait sert à produire de la lumière :

- 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 facile à repérer :
 - sa patte courte doit être connectée vers le MOINS (à noter le méplat en regard)
 - sa patte longue doit être connectée vers le PLUS

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

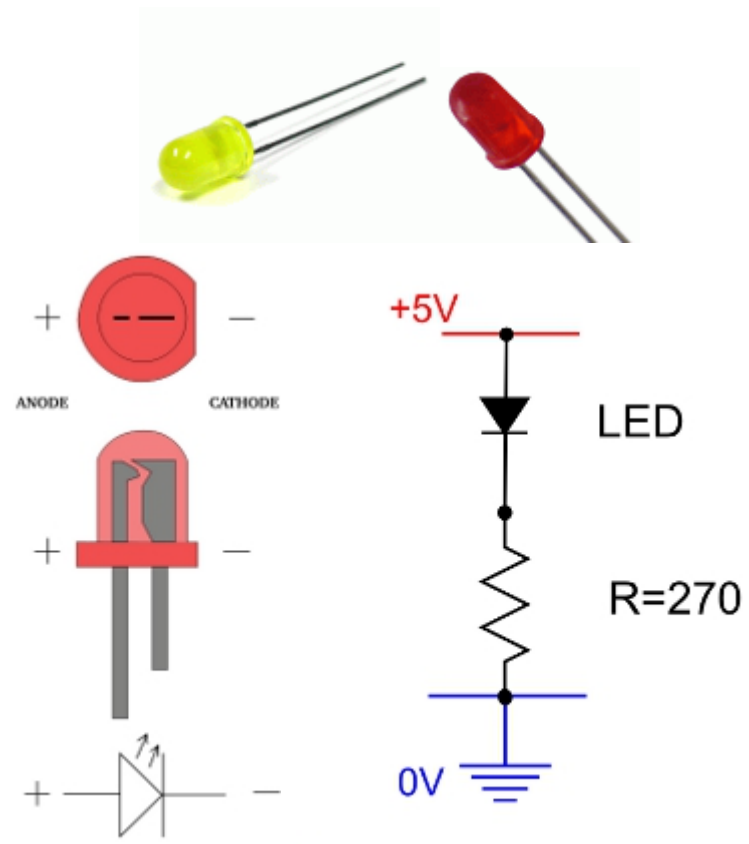
- connecter la LED en série avec une résistance (La patte - de la LED devra être tournée vers le -)
- la résistance a pour but de limiter l'intensité. Utiliser une valeur entre 200 et 300 Ohms. En pratique, j'utilise 270 Ohms / ¼ de watt.

Symbolisée par :



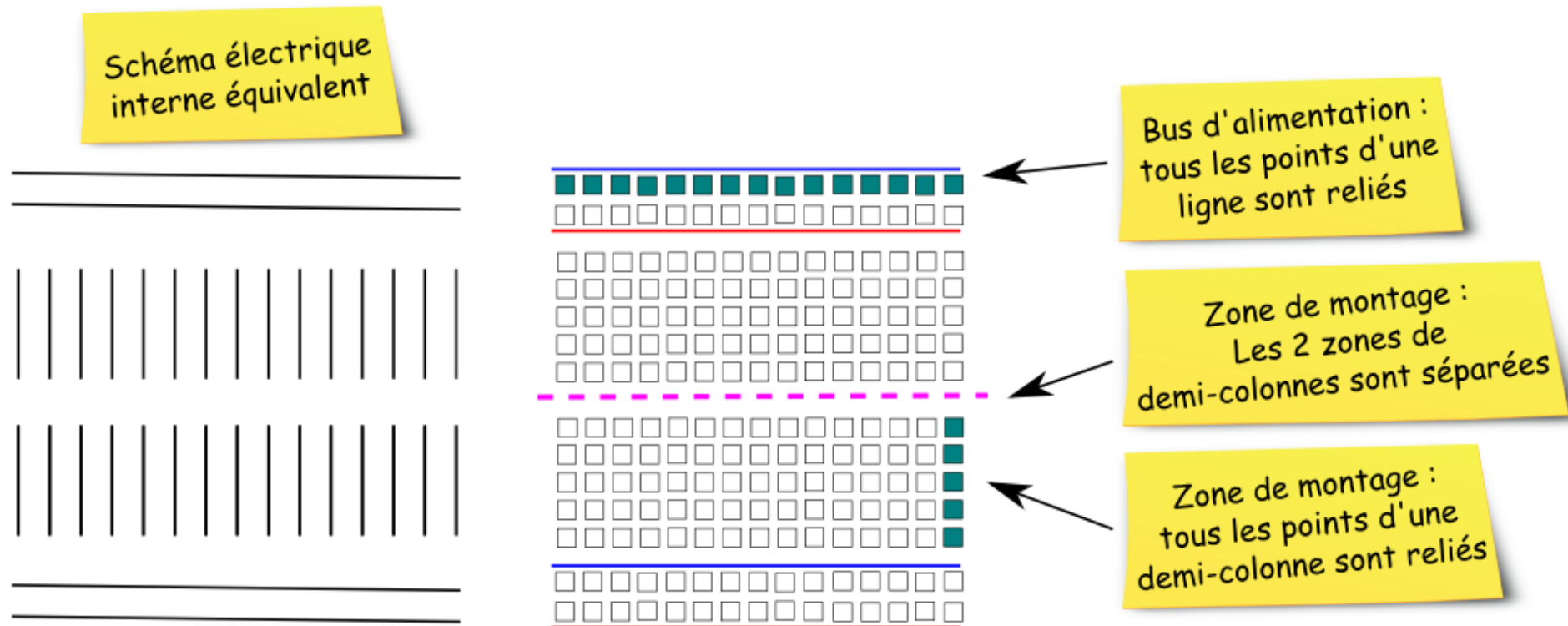
Pour les matheux (on est pas du tout obligé de savoir faire ce calcul !) :

- 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 $R = U/I = 3,5V / 0,013A = 270 \text{ Ohms}$.



19. Principe d'utilisation d'une plaque d'essai (ou « breadboard »)

Pour réaliser des montages électroniques sans soudure, on va utiliser ce que l'on appelle une « plaque d'essai » ou « breadboard » en anglais. Une fois que l'on a compris comment utiliser cette plaque, on pourra réaliser facilement toutes sortes de montages associés ou non à la carte Arduino.



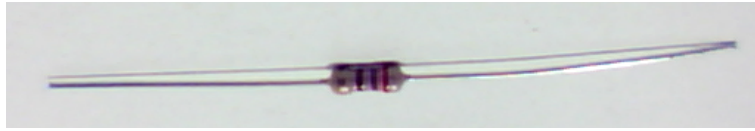
Une bonne habitude :

Mettez le bus **bleu** adjacent à la zone de montage en **bas** et le bus **rouge** adjacent à la zone de montage en **haut**.

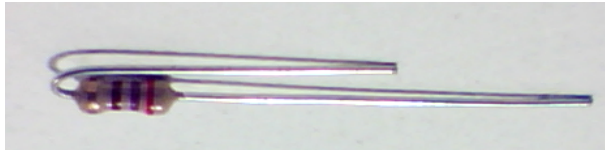
De cette façon, la transposition du schéma théorique en montage réel sera plus évidente.

20. *Truc pratique : préparer ses composants pour une utilisation facilitée avec la plaque d'essai*

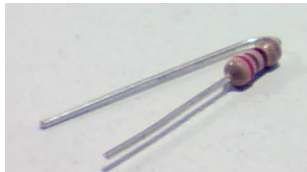
Préparer une résistance



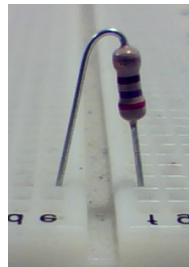
Courber une patte :



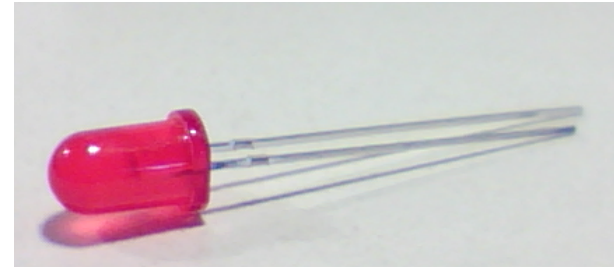
Couper les 2 pattes à la même longueur :



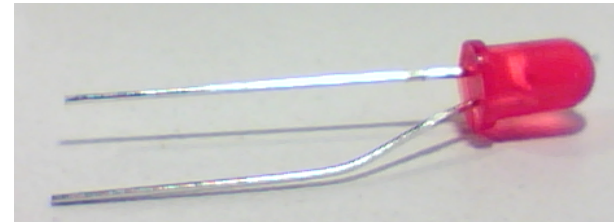
La résistance s'utilisera très simplement sur la plaque d'essai :



Préparer une LED



Courber légèrement la patte longue de façon à obtenir la même longueur pour les 2 pattes :

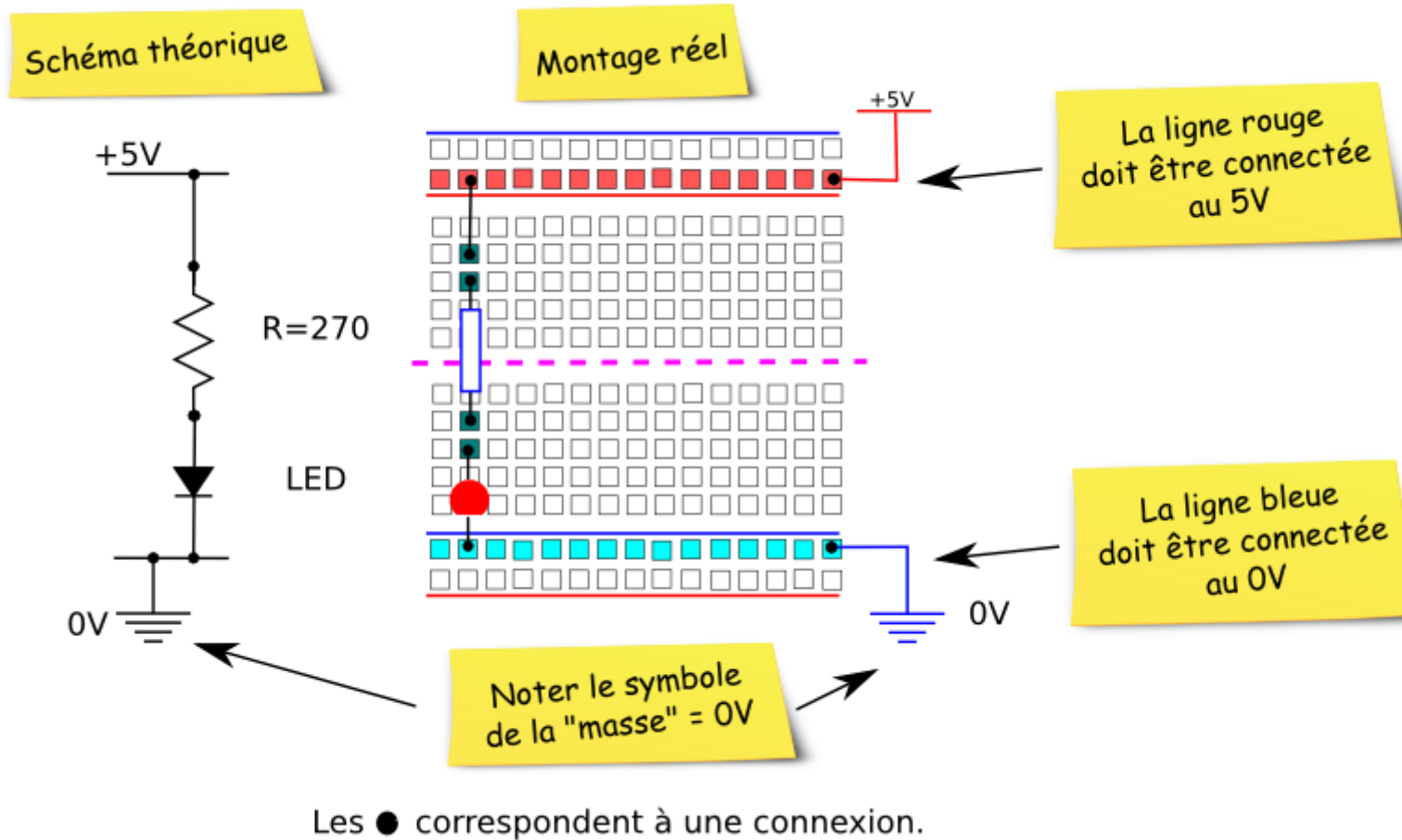


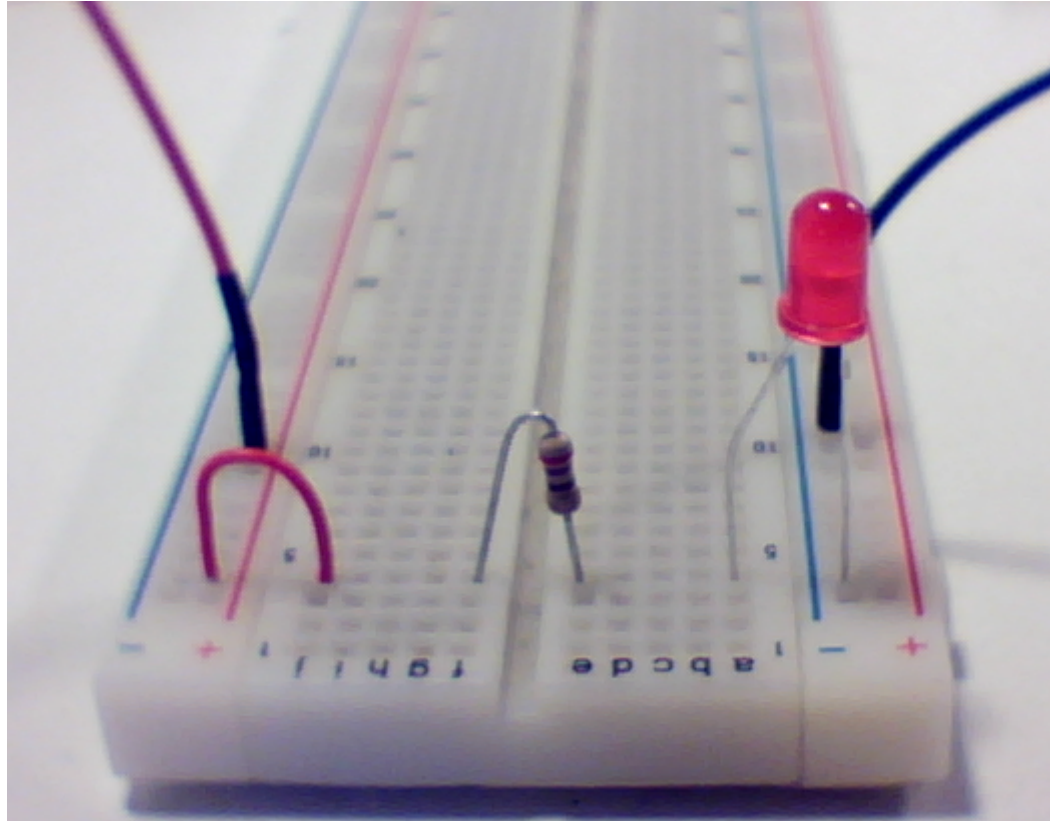
Ainsi, la LED sera très facile à utiliser sur une plaque d'essai :



21. Un exemple de montage simple sur une plaque d'essai : à vous de jouer !

Dans ce montage très simple, on va monter une LED en série avec une résistance entre le 0 et le 5V. On utilisera ici le 0V et le 5V de la carte Arduino pour alimenter la plaque d'essai. Une fois fait, la LED doit être allumée.





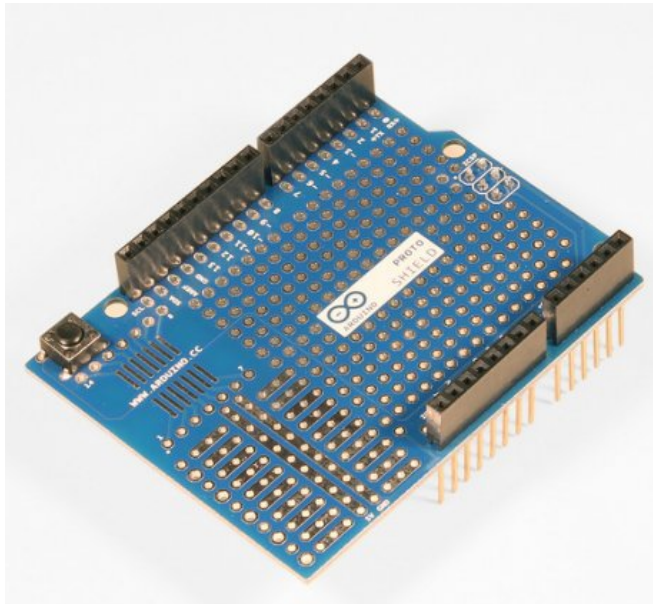
Votre premier montage sur plaque d'essai : la LED s'allume ? Bravo !

22. Truc technique : rendre définitif un montage sur plaque d'essai avec un proto-shield.... !

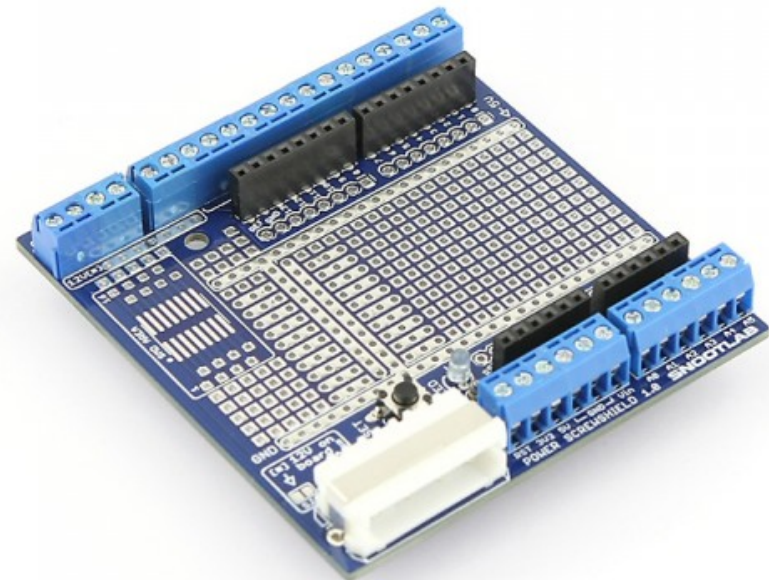
Bon à savoir :

Tous les montages que vous réaliserez sur plaque d'essai (et donc transitoires) peuvent être facilement rendus définitifs au besoin grâce au « proto-shield » ou shield de prototypage. Ce shield (ou carte d'extension) s'insérera broche à broche sur la carte Arduino et dispose d'une zone de pastilles à souder équivalentes à une plaque d'essai.

Un montage que vous avez envie de rendre définitif : soudez-le sur un protoshield (de 10 à 15€ selon modèle) !



Protoshield Arduino



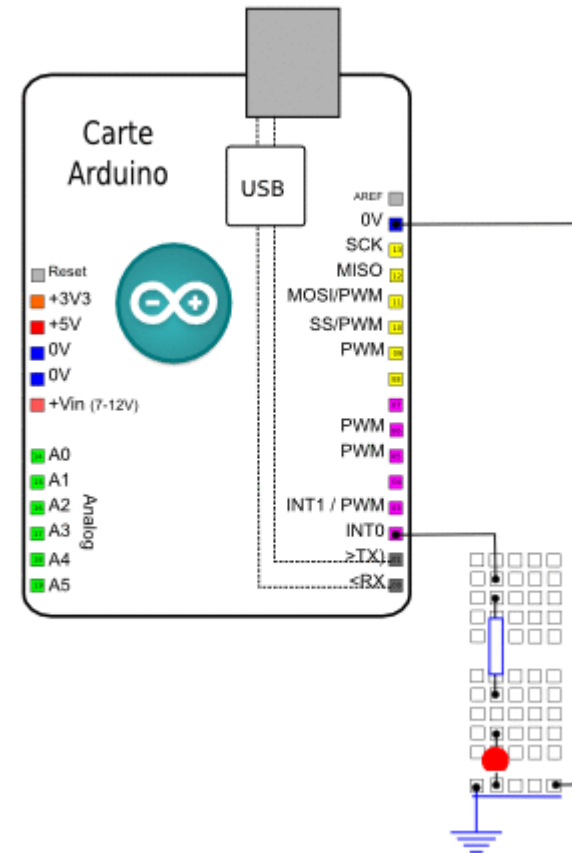
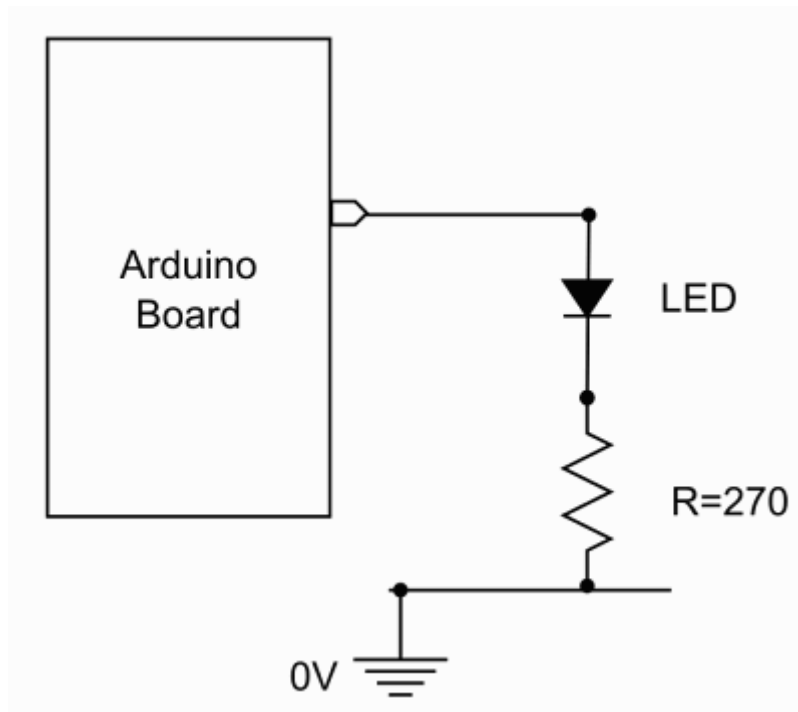
Power Screwshield (Snootlab)

23. Faire clignoter une LED : Le montage.

A présent, votre premier montage avec la carte Arduino ne devrait pas vous poser de problème.

Pour faire clignoter une LED, on va connecter une LED en série avec une résistance sur une broche E/S de la carte Arduino 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 $R = U/I = 3,5V/0,013A = 270 \text{ Ohms}$.

24. Faire clignoter une LED : Le programme.

A présent, nous allons ré-écrire par nous mêmes le fameux programme « Blink » qui fait clignoter une LED. Avec tout ce que vous savez, cela ne devrait pas vous poser de problème :

Entête déclarative

Aucune variable à déclarer ici.

Fonction setup()

A ce niveau, on va :

- initialiser la broche utilisée en sortie (la broche 2) avec l'instruction `pinMode()`

Fonction loop()

A ce niveau on va :

- Allumer la LED = mettre la broche 2 au niveau HAUT (5V) avec l'instruction `digitalWrite()`
- Attendre 1 seconde (=1000 millisecondes) avec l'instruction `delay()`
- Eteindre la LED = mettre la broche 2 au niveau BAS (0V) avec l'instruction `digitalWrite()`
- Attendre 1 seconde (1000 millisecondes) avec l'instruction `delay()`
- le code de la fonction loop se répète sans fin...

A vous de jouer (la solution suit...) :

faire la même chose en utilisant une constante pour désigner la broche et une variable pour la vitesse de clignotement

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

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

    pinMode(2, OUTPUT); // met la broche en sortie

} // fin de la fonction setup()

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

    digitalWrite(2,HIGH); // allume la LED
    delay(1000); // pause d'une seconde
    digitalWrite(2,LOW); // éteint la LED
    delay(1000); // pause d'une seconde

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

25. Prendre de bonnes habitudes : nommer les broches et utiliser des variables globales !

Ok, alors cette fois on va reprendre le même programme mais en utilisant :

- une constante pour désigner la broche
- et une variable pour fixer la vitesse de clignotement.

Entête déclarative

On déclare :

- une constante de type **int** pour désigner la broche 2, appelée LED
- une variable de type **int** pour fixer la vitesse, appelée vitesse

Fonction setup()

A ce niveau, on va :

- initialiser la broche utilisée en sortie (avec la constante LED) avec l'instruction **pinMode()**

Fonction loop()

A ce niveau on va :

- Allumer la LED = mettre la broche au niveau HAUT (5V) avec l'instruction **digitalWrite()**
- Attendre le temps voulu (= la valeur de la variable vitesse en millisecondes)
- Eteindre la LED = mettre la broche au niveau BAS (0V)
- Attendre le temps voulu (= la valeur de la variable vitesse en millisecondes) avec l'instruction **delay()**
- le code de la fonction loop se répète sans fin...

En pratique, pour écrire des programmes « propres » :
Renommer vos broches dans l'entête déclarative avec des constantes !
Pour toutes les valeurs qui reviennent souvent, utiliser des variables !

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

const int LED=2; // constante désignant la broche de la LED
int vitesse=250; // variable fixant la durée de la pause en ms

//--- 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() {

    digitalWrite(LED,HIGH); // allume la LED
    delay(vitesse); // pause de n millisecondes
    digitalWrite(LED,LOW); // éteint la LED
    delay(vitesse); // pause de n millisecondes

} // fin de la fonction loop()

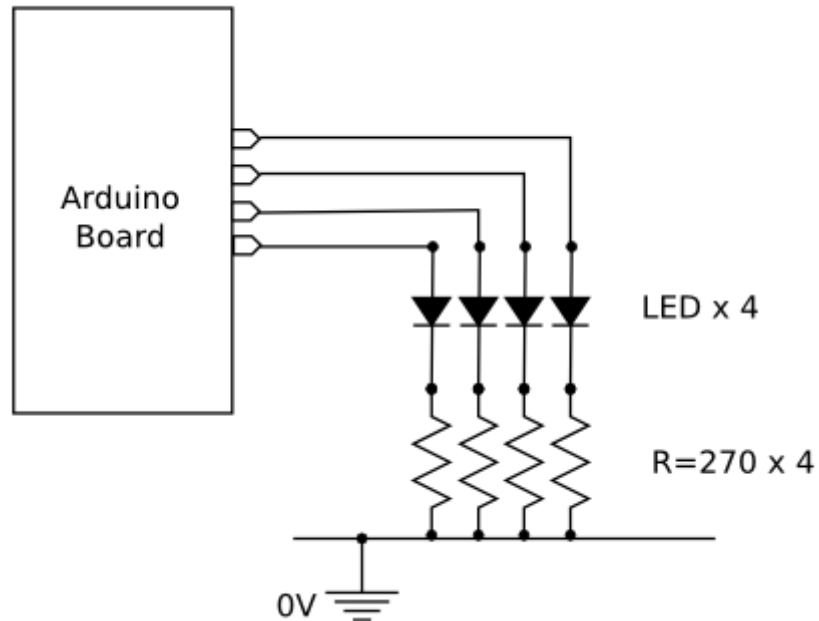
// NB : les lignes précédées de // sont des commentaires
```

Note : En utilisant des constantes pour désigner les broches, les programmes sont plus simples à modifier et la fonction des broches est plus claire ! Vos codes sont également plus faciles à réutiliser dans un autre programme ou une fonction.

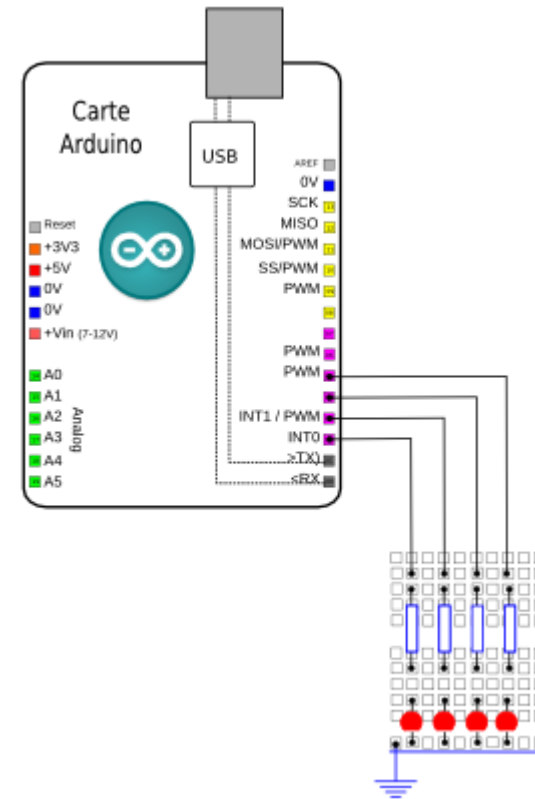
26. Faire clignoter 4 LEDs : le montage

Pour faire clignoter 4 LEDs, on va connecter 4 LEDs chacune en série avec une résistance sur 4 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 $R = U/I = 3,5V/0,013A = 270 \text{ Ohms}$.



27. Faire clignoter 4 LEDs : le programme

Ok, alors cette fois on va reprendre le même programme mais en réalisant quelques adaptations :

- une constante pour désigner chaque broche
- et une variable pour fixer la vitesse de clignotement.

Entête déclarative

On déclare :

- 4 constantes de type int pour désigner les broches 2,3,4 et 5, appelées LED1, LED2, LED3, LED4.
- 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 LED1, LED2, LED3, LED4)

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 les 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...

Apprenez à « penser » le déroulement de votre code :

Ici, les 4 LEDs sont allumées successivement mais c'est tellement rapide que c'est instantané, quasi-simultané. Retenez que le microprocesseur va très vite.

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

const int LED1=2; // constante désignant la broche de la LED
const int LED2=3; // constante désignant la broche de la LED
const int LED3=4; // constante désignant la broche de la LED
const int LED4=5; // constante désignant la broche de la LED

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

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

    pinMode(LED1, OUTPUT); // met la broche en sortie
    pinMode(LED2, OUTPUT); // met la broche en sortie
    pinMode(LED3, OUTPUT); // met la broche en sortie
    pinMode(LED4, OUTPUT); // met la broche en sortie

} // fin de la fonction setup()

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

    digitalWrite(LED1,HIGH); // allume la LED
    digitalWrite(LED2,HIGH); // allume la LED
    digitalWrite(LED3,HIGH); // allume la LED
    digitalWrite(LED4,HIGH); // allume la LED

    delay(vitesse); // pause de n millisecondes

    digitalWrite(LED1,LOW); // éteint la LED
    digitalWrite(LED2,LOW); // éteint la LED
    digitalWrite(LED3,LOW); // éteint la LED
    digitalWrite(LED4,LOW); // éteint la LED

    delay(vitesse); // pause de n millisecondes

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

28. Déclarer un tableau de variables

Imaginez maintenant que l'on veuille non pas 4, mais 8 LEDs : le code va sensiblement s'alourdir. D'où l'idée de pouvoir simplement « répéter » pour chaque variable le même code. Ceci sera rendu possible grâce à une boucle comme on va le voir.

Mais pour parcourir une liste de variables à l'aide d'une boucle, il faut au préalable créer un tableau de variables. Le principe : toutes les variables ont le même nom, le nom du tableau, mais sont numérotées de 0 à n-1 dans le tableau.

Exemple d'un tableau appelé LED désignant les broches 2 à 9 :

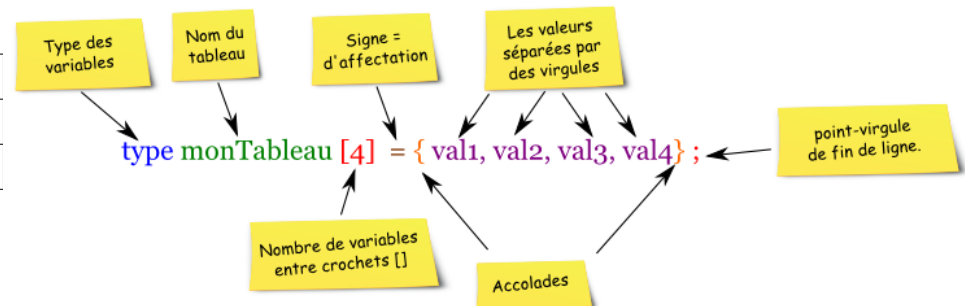
Indice	0	1	2	3	4	5	6	7
Variable	LED[0]	LED[1]	LED[2]	LED[3]	LED[4]	LED[5]	LED[6]	LED[7]
Valeur	2	3	4	5	6	7	8	9

Par exemple, si on déclare un tableau de 8 variables LED :

- on déclarera un tableau de 8 variables appelé LED en faisant `int LED[8]` (pour un tableau de constantes, idem avec `const` avant)
- on initialise le tableau sous la forme `{2, 3, 4, 5, 6, 7, 8, 9}` (On doit initialiser autant de valeurs qu'il y a de variables !)
- chaque élément du tableau sera accessible sous la forme `LED[i]` où `i` est l'indice dans le tableau.
- Ainsi, `LED[0]` sera la première variable, `LED[1]` la 2ème, etc... BIEN NOTER LE DECALAGE -1: 1er élément à l'index 0, etc.. !

Ne pas oublier : la première variable d'un tableau a l'indice 0, la 2ème, l'indice 1... la n-ième, l'indice n-1

Déclaration d'un tableau de variables avec initialisation.



Exemples

```
int mesValeurs[8] = {1,2,3,4,5,6,7,8}; // Tableau de variables
```

```
const int LED[4] = {2,3,4,5}; // Tableau de constantes
```

29. Faire clignoter 4 LED en utilisant un tableau de constantes

Cette fois on va reprendre le même programme mais en utilisant, comme nous venons de le voir, un tableau de constantes pour désigner les broches :

- un tableau de 4 constantes pour désigner les 4 broches
- et une variable pour fixer la vitesse de clignotement.

Entête déclarative

On déclare :

- un tableau de 4 constantes appelé LED de type int pour désigner les broches 2,3,4 et 5.
- 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])

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...

En utilisant un tableau de variables pour désigner un ensemble de broches : on simplifie et on allège d'une part le code et on permet de plus une adaptation simplifiée du programme, seule la ligne d'initialisation du tableau des broches utilisées étant à modifier si on change de broches !

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

const int LED[4] = {2,3,4,5}; // 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

} // fin de la fonction setup()

//--- 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

    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

    delay(vitesse); // pause de n millisecondes

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

30. Les éléments du langage Arduino étudiés dans cet atelier

Structure

Variables et constantes

Fonctions

Constantes prédéfinies

- [HIGH](#) | [LOW](#)
- [INPUT](#) | [OUTPUT](#)

Entrées/Sorties numériques

- [pinMode](#)(broche, mode)
- [digitalWrite](#)(broche, valeur)
- int [digitalRead](#)(broche)

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

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

- D'utiliser les broches E/S en sortie avec 1 ou plusieurs LEDs

Table des matières

NIVEAU DEBUTANT |

Sorties numériques : découvrir et apprendre à utiliser les broches de la carte Arduino en sorties numériques.

Intro |

Notion d'électronique numérique |

Une broche numérique ne peut avoir que 2 états : HAUT ou BAS, « y'a ou y'a pas » ! |

Une broche numérique est caractérisée par son SENS : en SORTIE ou en ENTREE ! |

Les broches numériques de la carte Arduino |

Truc technique : les broches E/S de la carte Arduino sur borniers à vis avec un screwshield.... ! |

Les instructions du langage Arduino pour la gestion des broches numériques |

Ecrire un programme qui met une broche en sortie au niveau HAUT |

Notion d'électricité élémentaire à bien connaître |

Notion de tension alternative, continue, régulée.. |

Truc technique... : une alimentation régulée de « labo » à pas cher ! |

Caractéristiques électriques globales de la carte Arduino |

Caractéristiques électriques d'une broche Numérique Arduino en sortie |

Technique : Le schéma électrique interne de l'alimentation de la carte Arduino |

Découvrir les composants et accessoires de base pour faire des montages avec la carte Arduino |

Découvrir la résistance et son principe d'utilisation |

Découvrir la LED et son principe d'utilisation |

Principe d'utilisation d'une plaque d'essai (ou « breadboard ») |

Truc pratique : préparer ses composants pour une utilisation facilitée avec la plaque d'essai |

Un exemple de montage simple sur une plaque d'essai : à vous de jouer ! |

|

Truc technique : rendre définitif un montage sur plaque d'essai avec un proto-shield.... ! |

|

Faire clignoter une LED : Le montage. |

Faire clignoter une LED : Le programme. |

Prendre de bonnes habitudes : nommer les broches et utiliser des variables globales ! |

Faire clignoter 4 LEDs : le montage |

Faire clignoter 4 LEDs : le programme |

Déclarer un tableau de variables |

Faire clignoter 4 LED en utilisant un tableau de constantes |

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