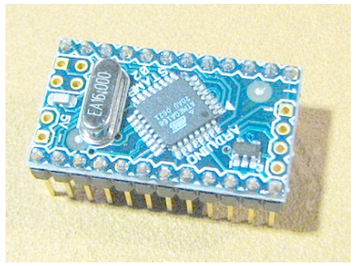
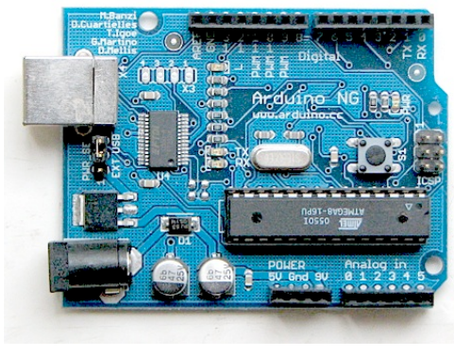
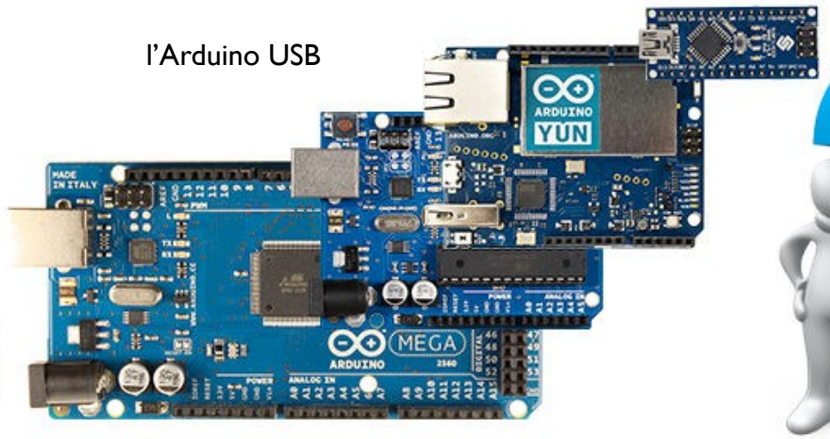
L'Arduino est une carte électronique en Matériel Libre pour la création artistique interactive.
Elle peut servir:
1/ pour des dispositifs interactifs *autonomes* simples
2/ comme *interface* entre capteurs/actionneurs et ordinateur
3/ comme programmateur de certains microcontrôleurs.

l'Arduino mini





l'Arduino USB



Le projet

Le projet Arduino comprend à la fois le développement matériel de cette carte, mais aussi le développement de son environnement de programmation, adaptation du logiciel de programmation pour la carte.L'Arduino n'est cependant pas exclusivement liée à Processing, et peut être utilisée en fonctionnement piloté avec la quasi totalité des logiciels de gestion d'événements multimédia interactifs. L'Arduino peut également être utilisée comme *carte de programmation* pour des microcontrôleurs utilisables dans d'autres montages électroniques autonomes ou pilotés. Pour les utilisateurs chevronnés, la carte peut également être programmée en langage AVR-C.

La licence

L'Arduino est un Logiciel Libre et Matériel Libre sous license Creative Commons " paternité, non commercial et licence contaminante", toute liberté est permise à qui voudrait faire évoluer le matériel ou la plateforme de programmation dans le respect de la licence. Le site officiel du projet Arduino est http://www.arduino.cc

Technologie

L'Arduino est une carte basée sur un microcontrôleur (mini-ordinateur) Atmel ATMEGA8 ou ATMEGA168. Elle dispose dans sa version de base de 1 Ko de mémoire vive, et 8Ko de mémoire flash pour stocker ses programmes. Elle peut être connectée à 13 entrées ou sorties numériques, dont 3 PWM (pouvant donner 3 sorties analogiques et 6 entrées analogiques convertissant en 10 bit. Dans la version la plus courante, la communication avec l'ordinateur se fait par un port USB. Il existe plusieurs versions de l'Arduino, dont une version miniaturisée, et d'autres projets sont également en gestation. La carte dispose d'un logiciel système interne (modifiable) et des programmes utilisateur.

Version USB

Description de la carte

Référence analogique (AREF)

LED jaune de test

Masse des connecteurs numériques (GND)

Port Série In/out
RX TX (supprime une entrée-sortie numérique si utilisé)

Trou de fixation par vis
(fixer sur un support non conducteur)

13 **entrées/sorties numériques**
numérotées de 1 à 13

LED verte témoin d'alimentation

Port USB:
pour le transport des données et pour l'alimentation électrique

Bouton de **réinitialisation** (reset)

Connecteurs pour le téléchargement du système d'exploitation du microcontrôleur

3 broches, avec **un cavalier** de sélection d'alimentation:

-cavalier sur les deux broches du haut: alimentation par le port USB, comme sur cette photo

-cavalier sur les deux broches du bas si alimentation par connecteur ci-dessous

Microcontrôleur

Trou de fixation par vis
(fixer sur un support non conducteur)

Connecteur d'alimentation de l'Arduino (si besoin)
9v à 12 V
Connecteur 2,1 mm
avec le + au centre
courant continu
(DC)

2 LEDs jaunes ou vertes, actives lors d'un téléchargement de programme

3 connecteurs numériques
pouvant également être utilisés en générateurs d'impulsions
(par ex: sortie analogique avec montage spécifique PWM numérotés de 0 à 2)

6 entrées analogiques
numérotées de 0 à 5

Masse (2 connecteurs)
(GND)

Source 5V faible puissance pour alimenter des montages divers comme des capteurs 0-5 V par exemple

Source 9V faible puissance pour alimenter des montages divers

Le logiciel Arduino

C'est un logiciel de programmation par code, code qui contient une cinquantaine de commandes différentes. A l'ouverture, l'interface visuelle du logiciel ressemble à ceci: des boutons de commande en haut, une page blanche vierge, une bande noire en bas

Mise en oeuvre de l'environnement Arduino:

-On conçoit d'abord un programme avec le logiciel Arduino

-On vérifie ce programme avec le logiciel (compilation)

-Des messages d'erreur apparaissent éventuellement...on corrige puis vérifie à nouveau...

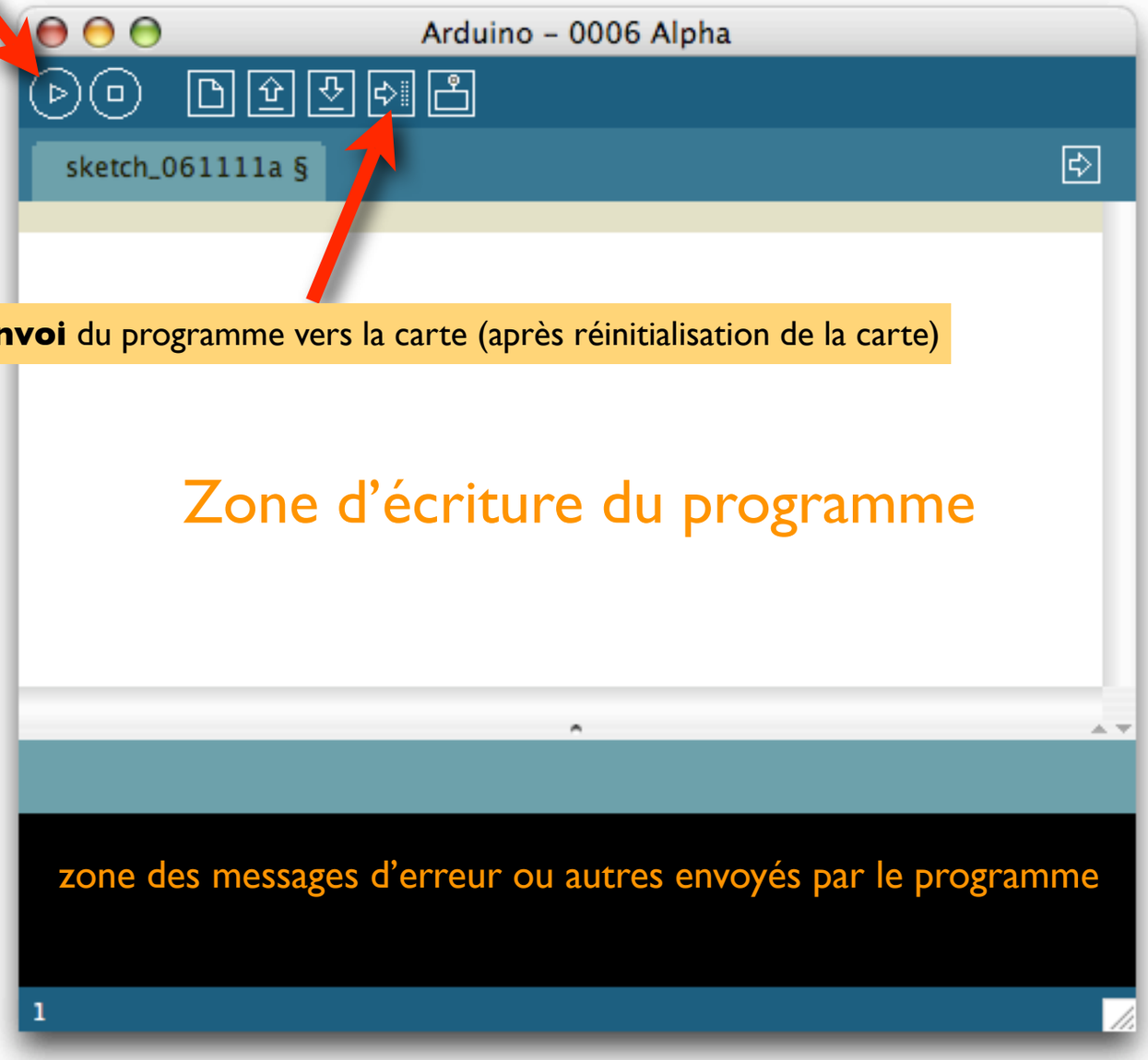
-On enlève le précédent programme sur la carte Arduino (Bouton réinitialisation)

-On envoie ce programme sur la carte Arduino dans les 5 secondes qui suivent l'initialisation

-L'exécution du programme sur la carte est automatique quelques secondes plus tard ou à ses prochains branchements sur une alimentation électrique (Alim 9/12V ou port USB).

Bouton :**Vérification** du programme après écriture = compilation

Bouton: **Envoi** du programme vers la carte (après réinitialisation de la carte)



Programmer avec Arduino

Un programme utilisateur Arduino est une suite d'instructions élémentaires sous forme textuelle, ligne par ligne.
La carte lit puis effectue les instructions les unes après les autres, dans l'ordre défini par les lignes de code.

Structure d'un programme

Il y a trois phases consécutives:

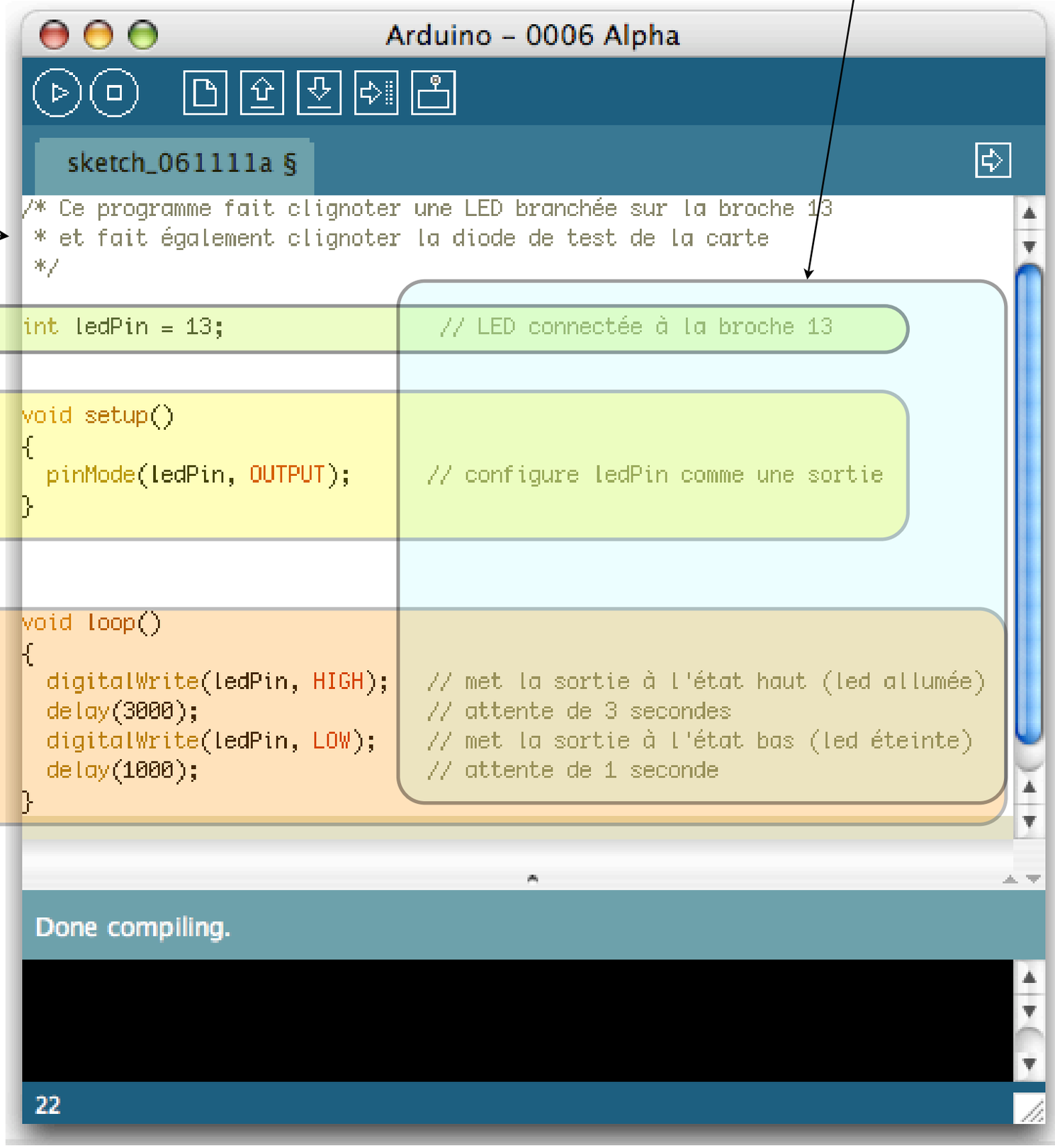
Commentaires multilignes pour se souvenir du patch ==>

1/La définition des constantes et des variables

2/La configuration des entrées et sorties
void setup()

3/La programmation des interactions et comportements
void loop()

Commentaires



Une fois la dernière ligne exécutée, la carte revient au début de la troisième phase et recommence sa lecture et son exécution des instructions successives. Et ainsi de suite.

Cette **boucle** se déroule des milliers de fois par seconde et anime la carte.

Introduction à la syntaxe des commandes Arduino

La cinquantaine d'éléments de la syntaxe Arduino est visible ici http://www.arduino.cc/en/Reference/HomePage ainsi qu'à partir du document "index.html" (dans le dossier "Reference" que vous avez téléchargé avec Arduino), également accessible dans le menu "Aide" du logiciel.
Revoyons d'un peu plus près le programme de la page précédente, qui sert à faire clignoter une LED à partir d'une sortie numérique:

(Syntaxe en marron, paramètres utilisateur en vert)

Commentaires
Toujours écrire des commentaires sur le programme: soit en multiligne, en écrivant entre des /****/, soit sur une ligne de code en se séparant du code avec //

Définition des variables:
Pour notre montage, on va utiliser une sortie numérique de la carte, qui est par exemple la 13 ème sortie numérique. Cette variable doit être définie et nommée ici: on lui donne un nom arbitraire BrocheLED . Le mot de la syntaxe est pour désigner un nombre *entier* est int

Configuration des entrées-sorties void setup()**:**
Les broches numériques de l'Arduino peuvent aussi bien être configurées en entrées numériques ou en sorties numériques. Ici on va configurer BrocheLED en sortie.
pinMode (nom, état) est une des quatre fonctions relatives aux entrées-sorties numériques.

Programmation des interactions void loop()**:**
Dans cette boucle, on définit les opérations à effectuer, dans l'ordre:
• digitalWrite (nom, état) est une autre des quatre fonctions relatives aux entrées-sorties numériques.
•delay(temps en millisecondes) est la commande d'attente entre deux autres instruction
•Chaque ligne d'instruction est terminée par un point virgule
•Ne pas oublier les accolades, qui encadrent la boucle.

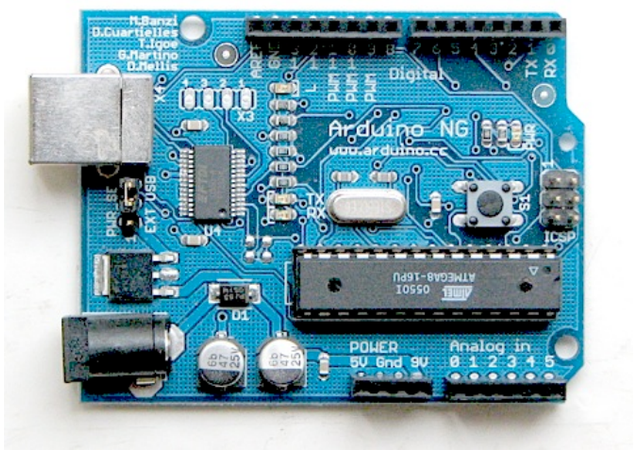
```
/* Ce programme fait clignoter une LED branchée sur la broche 13
 * et fait également clignoter la diode de test de la carte
 */



int BrocheLED = 13;     // Définition de la valeur 13 et du nom de la broche à
utiliser



void setup()
{
  pinMode(BrocheLED, OUTPUT);      // configure BrocheLED comme une
sortie
}



void loop()
{
  digitalWrite(BrocheLED, HIGH);   // met la sortie num. à l'état haut (led
allumée)
  delay(3000);                     // attente de 3 secondes
  digitalWrite(BrocheLED, LOW);    // met la sortie num. à l'état bas (led
éteinte)
  delay(1000);                     // attente de 1 seconde
}
```

Définition des variables

Qu'est ce qu'une variable ?
Une variable est un espace de stockage nommé qui permet de stocker une valeur utilisable par la suite dans la boucle d'un programme. Une variable peut aussi bien représenter des données lues ou envoyées sur un des ports analogiques ou numériques, une étape de calcul pour associer ou traiter des données, que le numéro 'physique' de ces entrées ou sorties sur la carte. Une "variable" n'est donc pas exclusivement un paramètre variant dans le programme.

*Pour composer un programme ex-nihilo, il est nécessaire de définir toutes les **composantes** d'entrée et de sortie qui vont affecter le **montage matériel** et les **calculs à effectuer**. Chaque entrée et chaque sortie sera une variable.*

Exemple:
Si on a **un** capteur (une cellule photo-électrique qui capte les variations de lumière par exemple) branché à une entrée de l'Arduino, et **un** actionneur (une LED) branché à une sortie l'Arduino, et si on veut que **la valeur** de luminosité change **la valeur** de l'intervalle de clignotement de la LED, alors on a 2 variables "stables" "pour la définition du matériel" et, en théorie, "2 variables pour les calculs " à déclarer en début de programme. En théorie seulement, car on va se servir directement de la valeur issue du capteur pour définir la valeur de l'intervalle de temps de durée d'extinction et durée d'allumage. On a donc besoin que de trois variables en tout. On va leur donner des noms arbitraires mais précis afin de bien les reconnaître dans le programme.



Le montage, ici réalisé sur une plaque d'expérimentation spécifique que l'on peut fixer sur l'Arduino

Matériel:
-cellule photo-electrique
-LED
-2 résistances

on verra ce montage en détail plus loin dans le livret

Au début de notre programme:

(Syntaxe en marron, paramètres utilisateur en vert)

Le code ci-dessous *déclare* (et dénomme arbitrairement) la variable capteur1, puis lui affecte (par exemple) le numéro de l'entrée analogique numéro 0. (L'Arduino possède 6 entrées analogiques numérotées de 0 à 5) :

```
int capteur1 = 0; // déclaration de la variable identifiant le portanalogique 0 de la carte
```

La ligne ci-dessous *déclare* (et dénomme arbitrairement) la variable LED1, puis lui affecte (par exemple) le numéro de la sortie numérique numéro 13 . (L'Arduino possède 13 entrées ou sorties numériques numérotées de 1 à 13) :

```
int LED1 = 13; // déclaration de la variable identifiant le autre port numérique 13 de la carte
```

La ligne suivante *déclare* (et dénomme arbitrairement) la variable qui correspond à la valeur de luminosité envoyée par le capteur. De plus, sa première valeur à l'allumage de la carte sera (arbitrairement) 0 :

```
int lum1 = 0; // déclaration de la variable identifiant la valeur de la luminosité du capteur 1
```

Nos trois variables sont maintenant déclarées et définies, passons à la configuration des entrées-sorties de la carte.

Configuration logicielle du matériel

Rappel des premières lignes du programme:

(Syntaxe en marron, paramètres utilisateur en vert)

```
int capteur1 = 0; // variable identifiant le port ana. 0 de la carte
int LED1 = 13; // variable identifiant le port num. 13 de la carte
int lum1 = 0; // variable identifiant la valeur de la luminosité du capteur 1
```

2/ Configuration du matériel (entrées et sorties)

Pour ce montage, il n'y a qu'une broche à configurer: la broche numérique sur laquelle on va brancher la LED (car elle pourrait être aussi bien configurée en sortie ou en entrée).

Ici, on va configurer cette broche numérique en sortie, car la LED est un actionneur. La broche d'entrée analogique pour le capteur n'est pas à configurer, car la carte Arduino possède 6 entrées analogiques qui ne font que cela.

après le void setup(), qui précise qu'on est à l'étape de configuration, on définit donc l'état de la broche 13 :

```
void setup()
{
  pinMode(LED1, OUTPUT);      // configure la broche 13 comme une sortie
}
```

et on ferme la phase de configuration par une accolade (touche clavier alt -parenthèse)

On peut maintenant passer à la boucle, c'est à dire le coeur du programme, qui définit les actions à effectuer avec ces variables.

Programmation des interactions

Rappel des premières lignes du programme: *(Syntaxe en marron, paramètres utilisateur en vert)*

```
int capteur1 = 0; // variable identifiant un port ana. 0 de la carte
int LED1 = 13; // variable identifiant le port num. 13 de la carte
int lum1 = 0; // variable identifiant la valeur de la luminosité du capteur 1

void setup()
{
  pinMode(LED1, OUTPUT);      // configure la broche 13 comme une sortie
}
```

3/ Programmation de l'interaction

- On indique maintenant qu'on crée une boucle avec `void loop() {`
- Puis on effectue la première opération: lire la valeur du capteur = lire la variable lum1 identifiant la valeur de luminosité

```
lum1 = analogRead( capteur1);
```

- On peut maintenant allumer la LED

```
digitalWrite(LED1, HIGH);
```

- On patiente un certain temps: en fonction de la valeur de la variable luminosité lum1

```
delay(lum1);
```

- On peut maintenant éteindre la LED

```
digitalWrite(LED1, LOW);
```

- On patiente un certain temps: en fonction de la valeur de la variable luminosité lum1

```
delay(lum1);
```

- On peut maintenant boucler, avec une accolade, c'est-à-dire faire remonter automatiquement au début de la boucle pour lire la nouvelle valeur du capteur et ainsi de suite...jusqu'à ce qu'on éteigne l'Arduino.

```
}
```

Notre programme est terminé, terminons les commentaires

(Syntaxe en marron, paramètres utilisateur en vert)

```
/* Ce programme fait clignoter une LED branchée sur la broche 13
 * avec une vitesse de clignotement proportionnelle à l'éclairage ambiant
 * capté par une cellule photo-électrique.
 *
 */



int capteur1 = 0; // variable identifiant un port ana. 0 de la carte
int LED1 = 13; // variable identifiant le port num. 13 de la carte
int lum1 = 0; // variable identifiant la valeur de la luminosité du capteur 1

void setup()
{
  pinMode(LED1, OUTPUT);      // configure la broche 13 comme une sortie
}

void loop()
{
lum1 = analogRead( capteur1); // lire la donnée capteur
digitalWrite(LED1, HIGH);      // allumer la LED 1
delay(lum1);                  // attendre pendant la valeur donnée par le capteur en millisecondes
digitalWrite(LED1, LOW);       // éteindre la LED 1
delay(lum1);                  // attendre pendant la même valeur
}
```

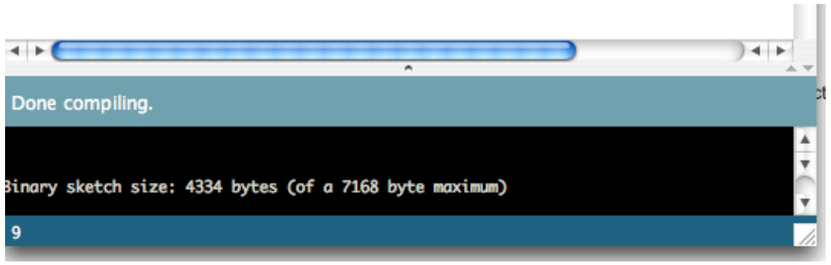
-Vérifions maintenant qu'un point-virgule finit bien chaque ligne de code, que les espaces soient bien placés
-Testons le programme sur le logiciel, avant de le transférer sur la carte:

Test et téléchargement

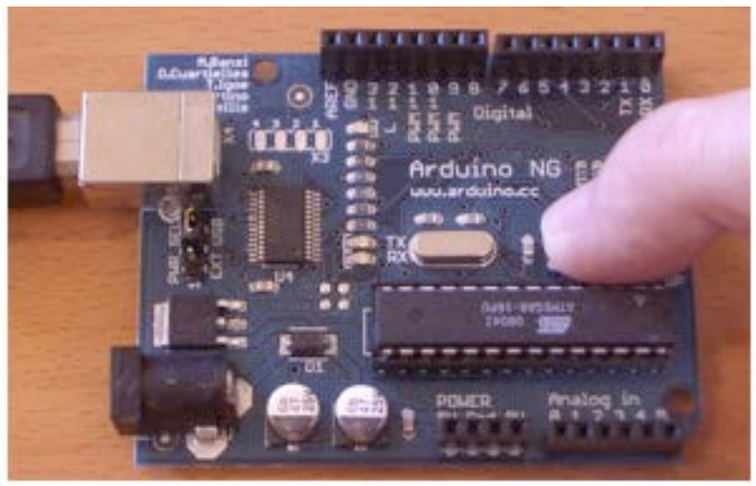
```
/* Ce programme fait clignoter une LED branchée sur la broche 13
 * avec une vitesse de clignotement proportionnelle à l'éclairage ambiant
 * capté par une cellule photo-électrique.
 *
 */


int capteur1 = 0; // variable identifiant un port ana. 0 de la carte
int LED1 = 13; // variable identifiant le port num. 13 de la carte
int lumT = 0; // variable identifiant la valeur de la luminosité du
capteur 1


void setup()
{
  pinMode(LED1, OUTPUT);      // configure la broche 13 comme une
sortie
}

void loop()
{
lum1 = analogRead( capteur1); // lire la donnée capteur
digitalWrite(LED1, HIGH);       // allumer la LED 1
delay(lum1);                   // attendre pendant la valeur donnée
par le capteur en millisecondes
digitalWrite(LED1, LOW);       // éteindre la LED 1
delay(lum1);                   // attendre pendant la même valeur
}
```

Vérifier

S'il n'y a pas d'erreurs on verra s'afficher: "done compiling", suivi de la taille du programme.



On peut sauver le fichier sur l'ordinateur, puis appuyer sur le bouton de ré-initialisation de la carte, ci-dessous.



Ré-initialiser

Télécharger

Et enfin, télécharger le programme sur l'Arduino: attention, vous avez 5 secondes après l'appui sur le bouton de ré-initialisation pour cliquer sur le bouton "Upload " !

Et voilà ! les deux petites LEDs TX RX sur la carte clignotent pendant le chargement, puis, quelques secondes plus tard, le programme se met en route.....jusqu'à ce qu'on éteigne la carte...

Syntaxe du langage Arduino

Commandes de structure du programme

Structure générale

- void setup() (configuration-préparation)
- void loop() (exécution)

Contrôle et conditions

- if (si...)
- if...else (si...alors...)
- for (pour...)
- switch case (dans le cas où...)
- while (pendant que ...)

Opérations de comparaison

- == (équivalent à)
- != (différent de)
- < (inférieur à)
- > (supérieur à)
- <= (inférieur ou égal à)
- >= (supérieur ou égal à)

Operations booléènnes

- && (et)
- || (ou)
- ! (et pas)

Autres commandes

- // (commentaire simple ligne)
- /* */ (commentaire multi-lignes)
- #define (donner une valeur à un nom)

Variables

Variables

- char (variable 'caractère')
- int (variable 'nombre entier')
- long (variable 'nombre entier de tès grande taille')
- string (variable 'chaine de caractères')
- array (tableau de variables)

Niveaux logiques des connecteurs numériques

- HIGH (état 1)
- LOW (état 0)
- INPUT (configuré en entrée)
- OUTPUT (configuré en sortie)

Fonctions

Entrées-sorties numériques

- pinMode(broche, état) (configuration des broches)
- digitalWrite(broche, état) (écrire un état sur une broche num.)
- digitalRead(broche) (lire un état sur une broche num.)
- unsigned long pulseIn(broche, état) (lire une impulsion sur une broche num.)

Entrées analogiques

- int analogRead(broche) (lire la valeur d'une broche ana.)
- analogWrite(broche, valeur) (PWM : écrire une valeur analogique sur les broches 9, 10 ou 11)

Gestion du temps

- unsigned long millis() (temps de fonctionnement du programme)
- delay(ms) (attente, en millisecondes)
- delayMicroseconds(us) (attente, en microsecondes)

Syntaxe du langage Arduino

syntaxe Arduino et dont voici la table des matières. Chaque instruction est suivie de sa traduction, entre-parenthèses et en noir.

Nombres aléatoires

- randomSeed(seed) (aléatoire ‘piloté’)
- long random(max) (aléatoire à partir de telle valeur)
- long random(min, max) (aléatoire entre deux valeurs)

Communications série entre Arduino et autres machines ou ordinateur

- Serial.begin(speed) (configuration de la vitesse de communication Série)
- Serial.available() (donne combien de caractères disponibles dans la zone tampon Série)
- Serial.read() (lit les données Série)
- Serial.print(data) (envoie des données Série)
- Serial.println(data) (envoie des données Série suivies de caractères spécifiques)

Un peu d'électronique interactive

Vous savez maintenant programmer des opérations simples, mais comment aborder la partie électronique ?

Faire des montages électroniques simples est à la portée de tous et les ressources sur l'électronique interactive (physical computing) sont multiples sur le web, il y a cependant des notions de base à avoir pour se lancer dans la réalisation de ses propres montages, même si on ne fait que copier un montage sans le comprendre :

- il est toujours utile de savoir reconnaître les composants
- il est toujours utile de savoir déterminer la valeur d'un composant (le code visuel des couleurs, les sigles et les abréviations)
- il est (souvent) utile de connaître la fonction d'un composant
- il est (parfois) utile de savoir lire un schéma, c'est-à-dire reconnaître les symboles des composants, et la raison des cablages.

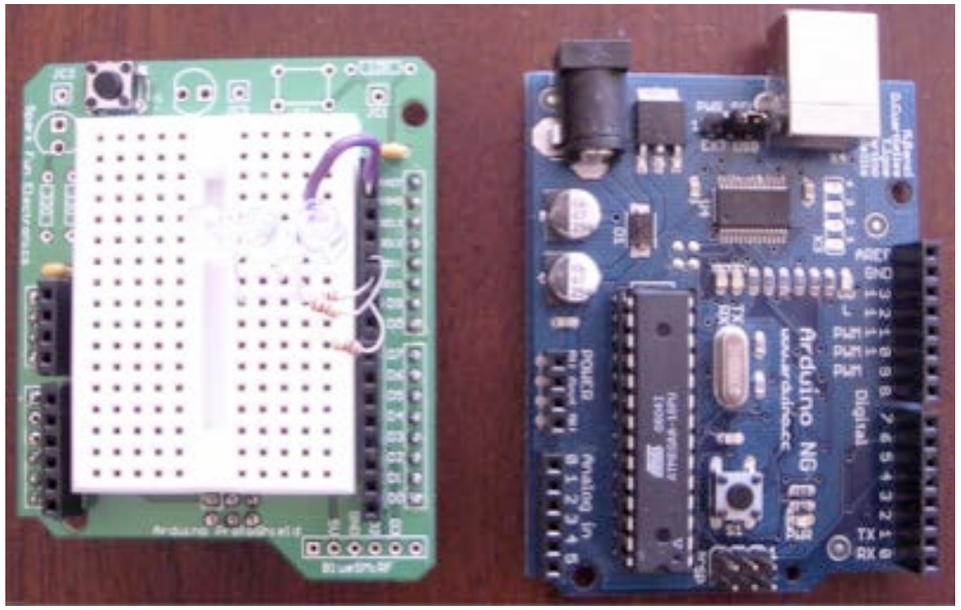
Il faut aussi prendre quelques précautions:

- Eviter de faire des court-circuits
- Utiliser les sources électriques recommandées
- Certains composants électroniques ont un sens = ils sont polarisés. Exemple: la LED, certains condensateurs, les diodes..
- Certains composants ne peuvent pas fonctionner seuls , comme la LED, qui a besoin d'une résistance appropriée pour limiter le courant.
- Certains composants se ressemblent mais n'ont pas du tout la même fonction: toujours bien regarder leur signalétique
- Ne pas manipuler de 230V sans connaissances appropriées.
- Préferer faire les essais et les montages avec une alim externe plutôt que celle de l'USB

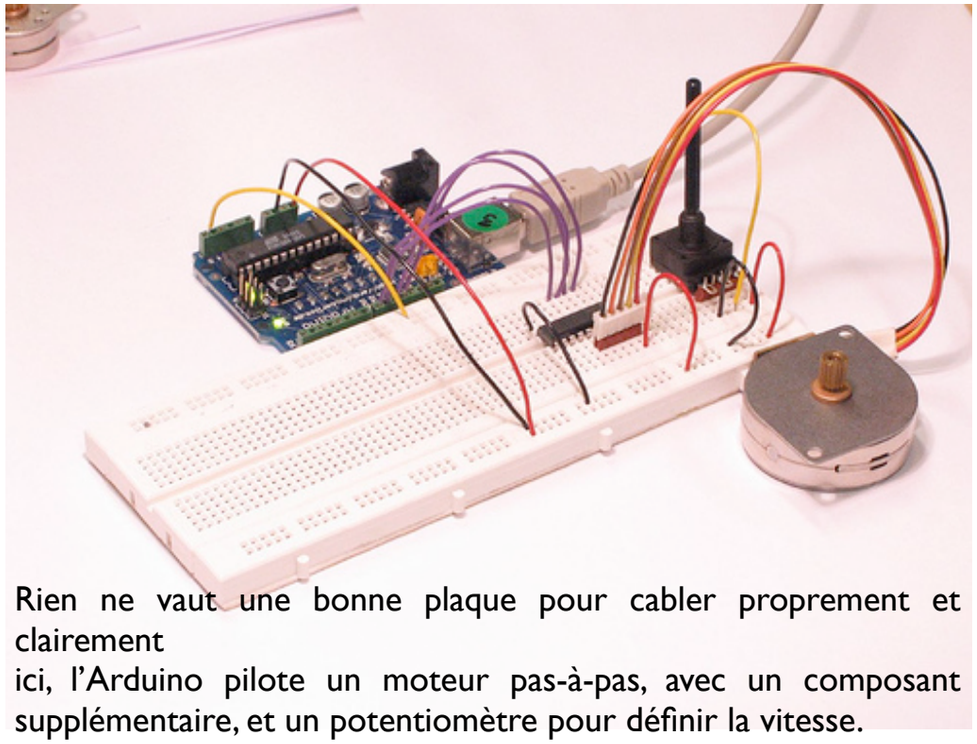
Les ressources du web, et notamment de Wikipedia vous aideront sur les concepts les plus difficiles.

Equipement en électronique interactive

Une plaque d'expérimentation (breadbord en anglais) permet de cabler de nombreux composants sans faire de soudure, et en gardant un montage entièrement démontable. Le projet Arduino propose une plaque adaptée à l'Arduino, dotée de connecteurs reproduisant exactement le plan d'implantation de la carte. On peut aussi se la bricoler soi-même avec quelques composants

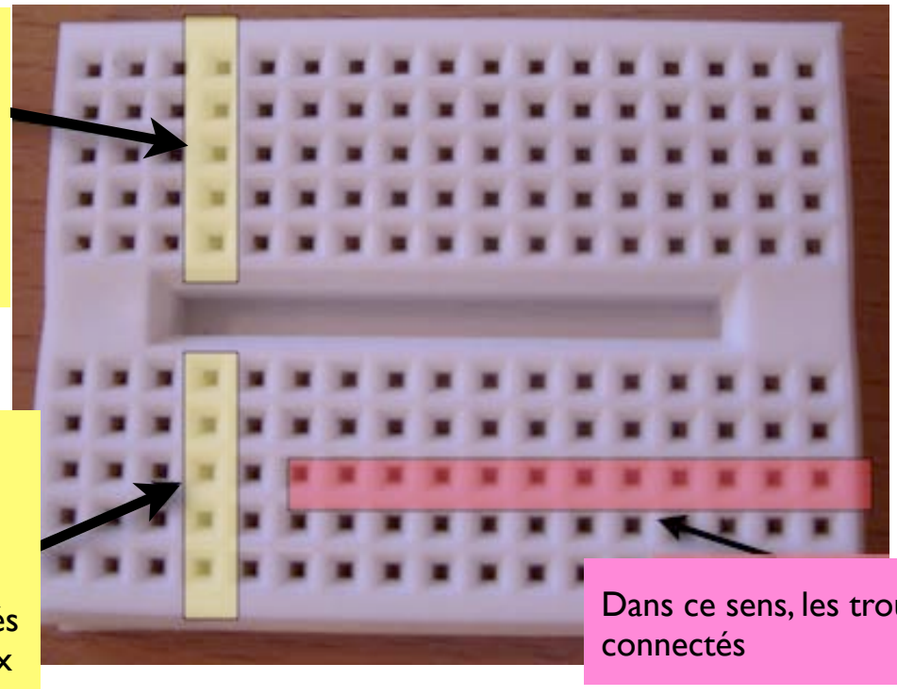


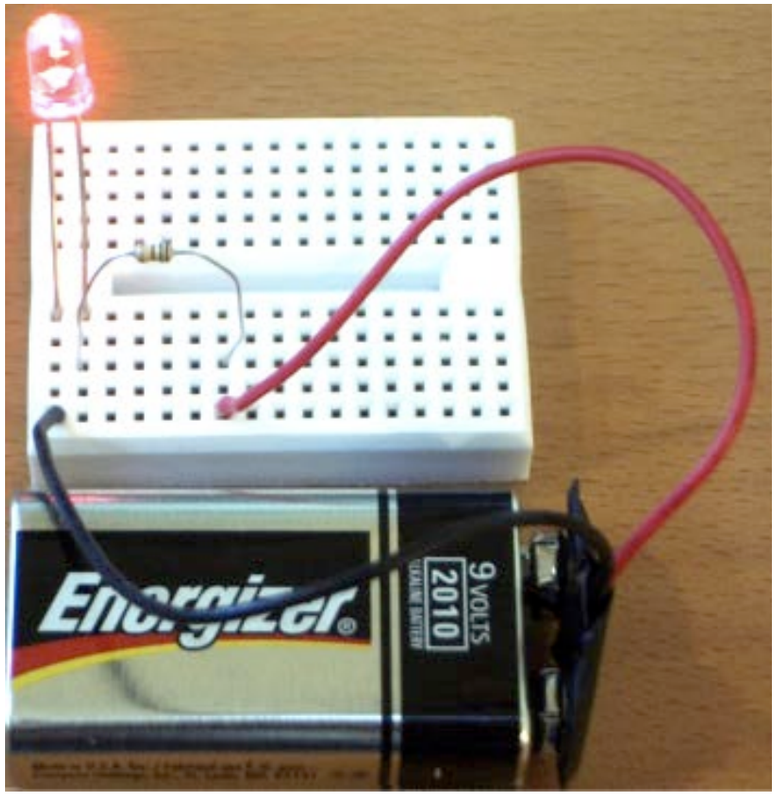
La plaque d'expérimentation Arduino "officielle": ici démontée, car elle vient normalement couvrir l'Arduino



Rien ne vaut une bonne plaque pour cabler proprement et clairement
ici, l'Arduino pilote un moteur pas-à-pas, avec un composant supplémentaire, et un potentiomètre pour définir la vitesse.

Aller plus loin en électronique





Une pile, une LED, sa résistance , voilà un bon début pour commencer sur une plaque.

Si les fils souples de la pile ne veulent pas rentrer, on peut les étamer avec un peu de soudure.

Electronique interactive

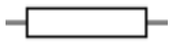
Connaitre les composants

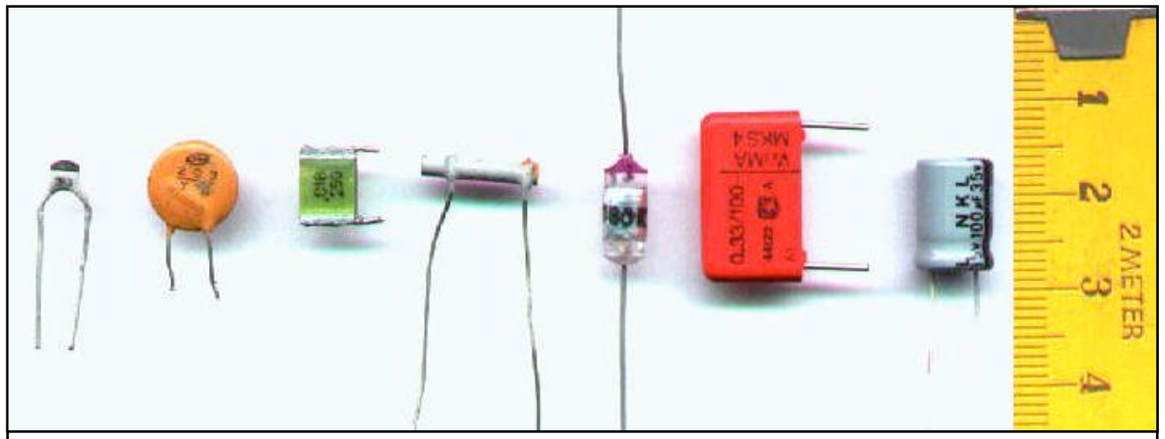
Savoir reconnaitre quelques composants électroniques, savoir reconnaitre leur symbole, connaitre leur usage, vous permettra de concevoir vos propres expériences et de progresser à partir de données trouvées sur le web. Vous saurez ainsi fabriquer quelques capteurs, et actionner quelques matériels.

La résistance

La résistance s'oppose au passage du courant, proportionnellement à sa " résistance" exprimée en Ohm. Un code de couleurs, ci dessous permet de reconnaître cette valeur.

Symbole européen





Les condensateurs

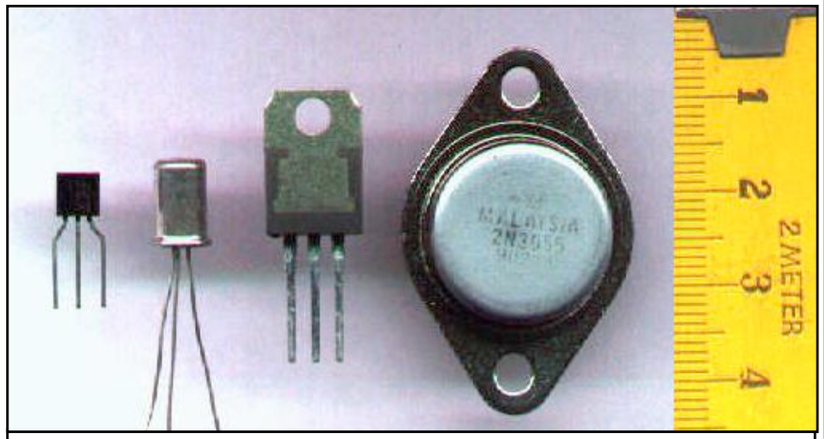
Les condensateurs peuvent stocker un peu de courant si on les charge, mais comme un tonneau percé, ils renvoient ce courant instantanément si ils sont branchés à un organe consommateur de courant.

Ils peuvent être polarisés ou non, dans des boitiers très divers. Leur valeur s'exprime en Farad (F)

non polarisé polarisé

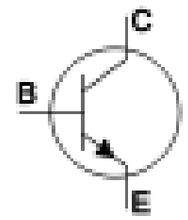
Symbole

Reconnaissance de leur valeur:
A cause de la diversité des modèles, se reporter aux ressources sur le web



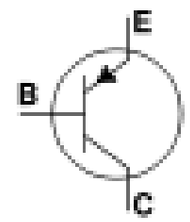
Le transistor

Le transistor est généralement utilisé comme une sorte de multiplicateur de puissance: lorsqu'on lui fait passer un courant faible, mais variable dans un de ses 3 pattes, il autorise proportionnellement le passage d'un gros courant dans une autre des 3 pattes.



transistor NPN

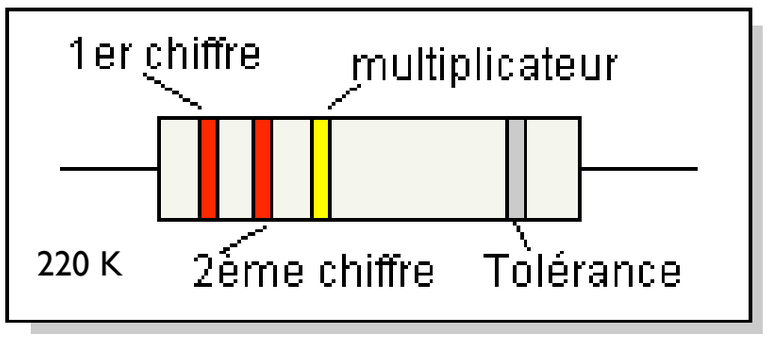
Symboles



transistor PNP

Code des couleurs des résistances

au delà de 1000 Ohms, on parle en KiloOhms, par exemple 10 K est 10 KiloOhms, puis en MegaOhms notés M

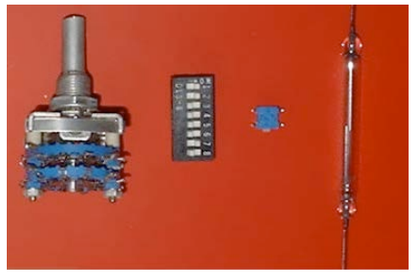


Chiffre	0	1	2	3	4	5	6	7	8	9	Or	Argent
Multiplicateur	1	10	10^2 100	10^3 1000	10^4 10000	10^5	10^6					
Précision	20%										5%	10%

Electronique interactive

Reconnaitre les composants

L'interrupteur



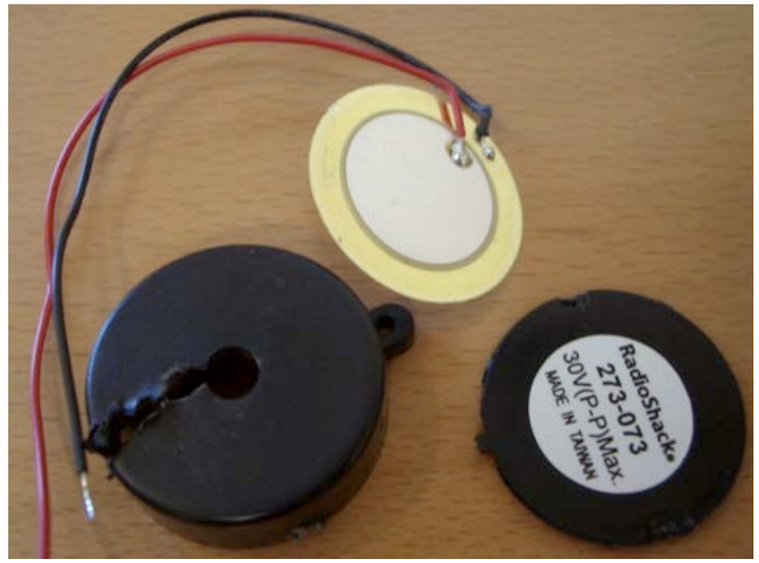
L'interrupteur ouvre ou ferme un circuit. Il y a toutes sortes d'interrupteurs.
Sur l'Arduino, utiliser un interrupteur pour déclencher un événement nécessite d'utiliser un composant supplémentaire: une résistance de 10K ohms. Voir " Montages d'électronique interactive".

La cellule photo-électrique (LDR)

La cellule photo-électrique (LDR)

C'est une résistance variable, en fonction de la luminosité qu'elle reçoit. Sa résistance diminue quand elle reçoit de la lumière. On s'en sert donc de capteur de luminosité. Non polarisée. Pour lire sa valeur avec une Arduino, il faut également l'associer avec une résistance équivalente à sa résistance maxi (dans le noir) Voir " Montages d'électronique interactive".

Le piezo



Le transducteur piezo-électrique est un composant réversible: il peut aussi bien être utilisé en capteur de chocs ou de vibrations qu'en actionneur pouvant émettre des sons stridents parfois modulables.

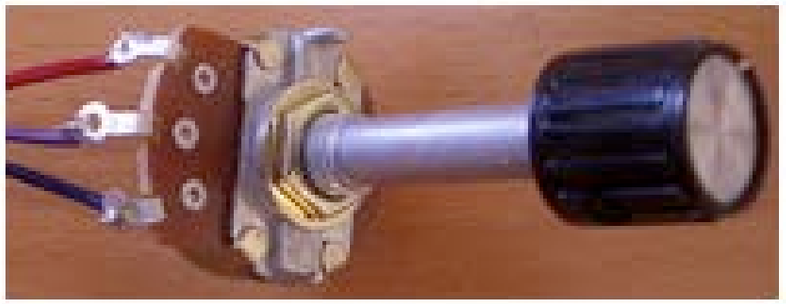
Electronique interactive

Reconnaitre les composants

Le servo moteur

Le servo-moteur est un moteur (rotatif) qui peut effectuer des rotations très précises (dans une portion de tour seulement) et en un certain nombre de pas (de micro-déplacements). Il y a toutes sortes de servo moteurs.. Un des avantages des servo moteurs est sa possibilité de maintenir avec force une position donnée. On peut piloter des rotations avec l'Arduino, quelques fois directement avec la carte si le moteur n'est pas trop puissant, sinon en passant par un montage associé.

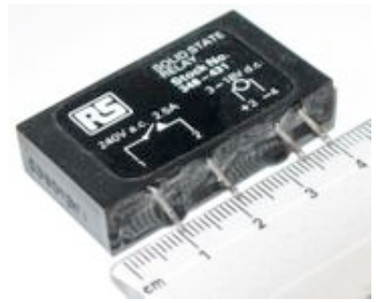
Le potentiomètre

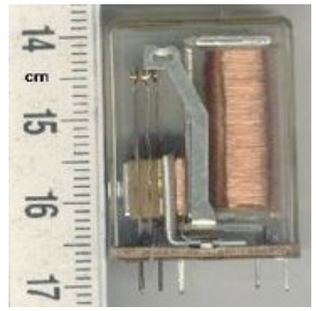


Le potentiomètre

Le potentiomètre, rotatif comme ici, ou à glissière, est une résistance variable. Entre les extrémités, il y a la résistance maximale. La patte centrale est le curseur. C'est la résistance entre cette patte centrale et une extrémité que l'on peut faire varier en tournant le bouton. Le potentiomètre est donc un capteur. Il se branche sur les entrées analogiques de l'Arduino.
De très nombreux capteurs sont basés sur le principe de résistance variable et se cablent presque de la même façon: la cellule photo-électrique, le capteur de pression, le fil résistif, etc

Le relais



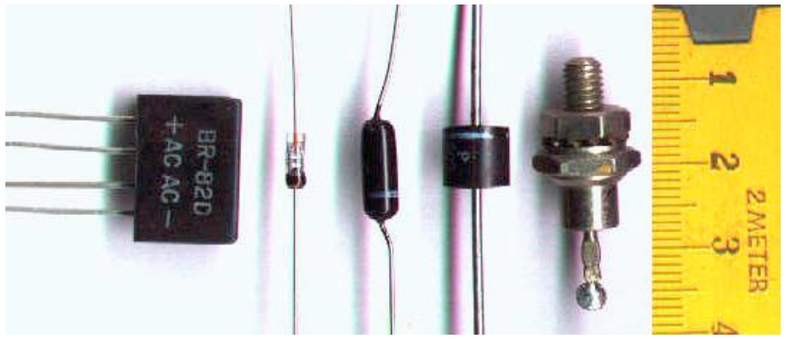


Le relais

Le relais est un composant à 4 broches minimum. C'est un interrupteur que l'on peut commander en envoyant un petit courant. Au repos, il est normalement fermé, ou normalement ouvert, selon le modèle. On peut s'en servir avec l'Arduino pour commander des machines en haute tension (230V par exemple), ou pour déclencher toute machine ou lumière.

Electronique interactive

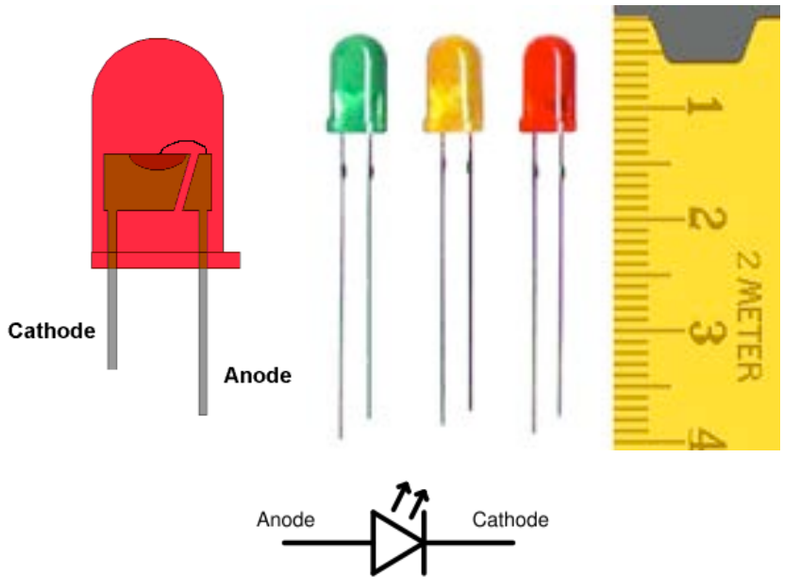
La diode



La diode

La diode ne laisse passer le courant que dans un seul sens. C'est un composant polarisé: on reconnait toujours son anneau coloré d'un coté du composant, correspondant à la cathode.

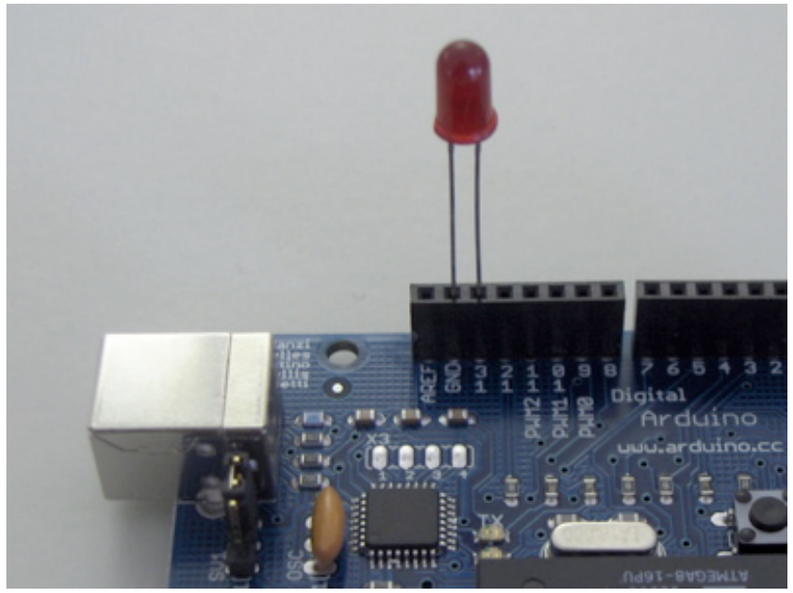
La LED



La diode électroluminescente (LED) émet de la lumière. Elle est polarisée: la patte "+" est la plus longue, l'autre patte est la patte "-" .
Les broches numériques de l'Arduino, lorsqu'elles sont configurées en sorties et qu'elles sont à l'état 1 ou haut (HIGH) , fournissent une tension de 5 volts, supérieure à ce que peut accepter une LED. Les LED doivent donc être couplées en série avec une résistance.

Reconnaitre les composants

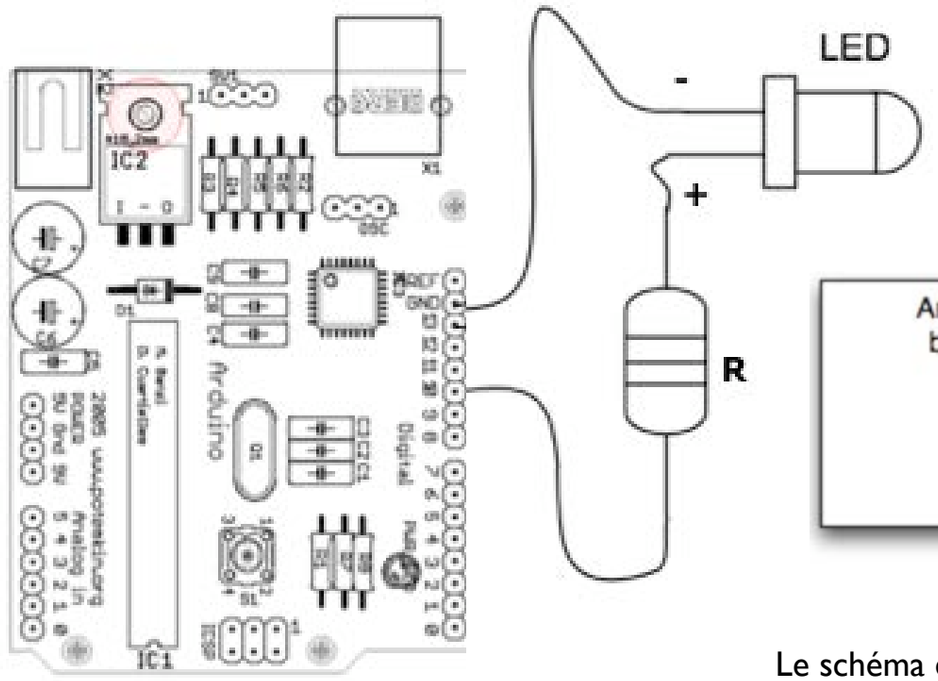
La broche numérique 13 de l'Arduino est déja cablée en série avec une résistance de valeur moyenne pour une LED (1K ohm), on peut donc, dans la plupart des cas, directement lui brancher une LED, comme sur la photo-ci-dessous. Le ***moins*** sur la masse (nommée GND comme Ground) et le ***plus*** sur la broche 13. Il ne reste plus qu'à déclarer dans le programme que la broche 13 est configurée en sortie, et le tour est joué pour faire quelques essais. Si on a des LEDs particulières, ou si on est sur un autre port , on calcule et on rajoute une résistance.



Couleurs	Tension de seuil ou Vf	If (mA)	Longueur d' onde
Rouge	1,6 V à 2 V	6à20	650 à 660 nm
Jaune	1,8 V à 2 V	6à20	565 à 570 nm
Vert	1,8 V à 2 V	6à20	585 à 590 nm
Bleu	2,7 V à 3,2 V	6à20	470 nm
blanc	3,5 v à 3,8 v	30	

Comment calculer la résistance à appairer avec une LED ?

Résistance (en Ohms) = (5-Vf) / If

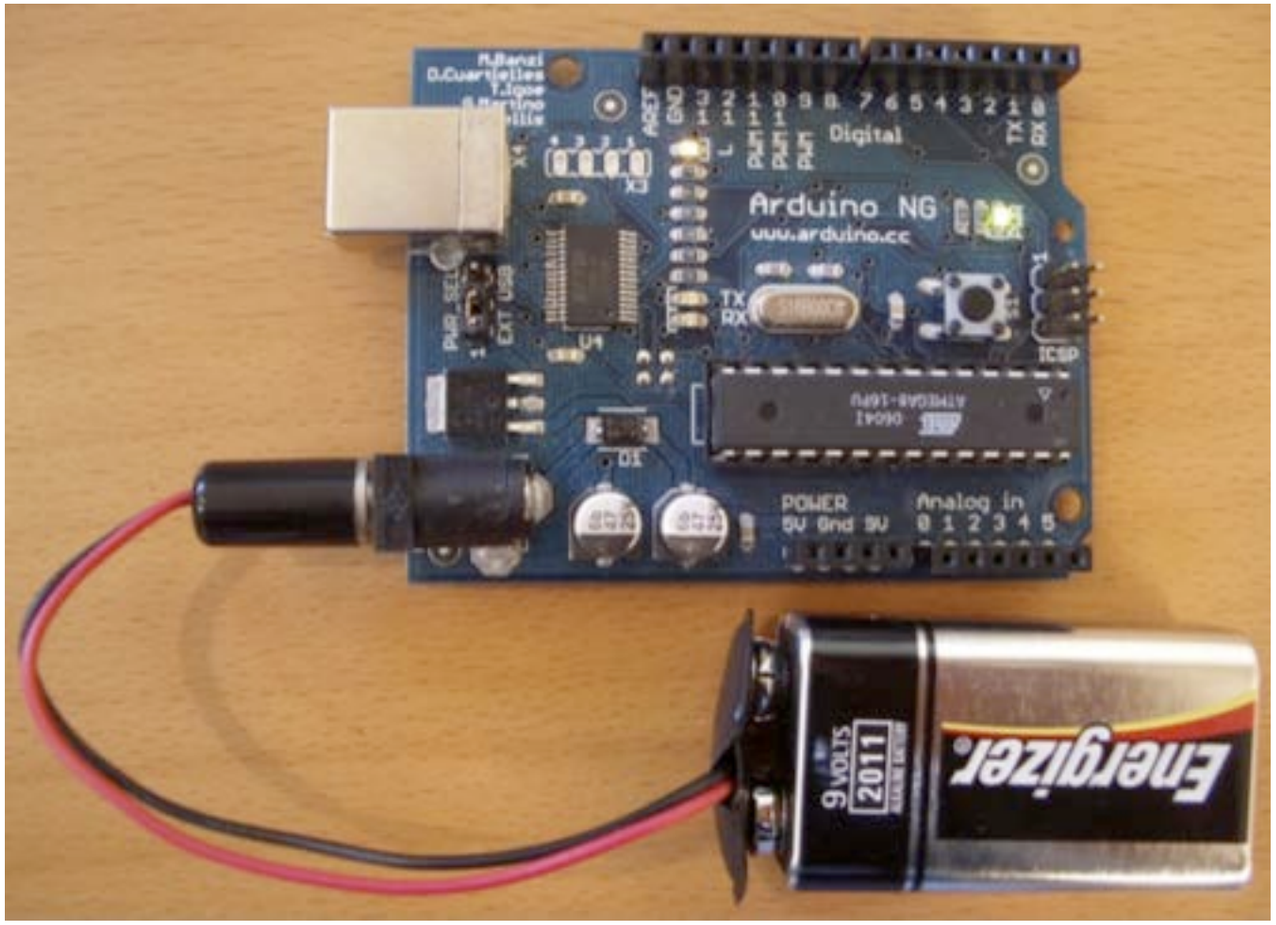


Une LED branchée sur la broche 10 de l'Arduino, avec sa résistance en série

Le schéma électronique correspondant

En mode autonome sans ordinateur

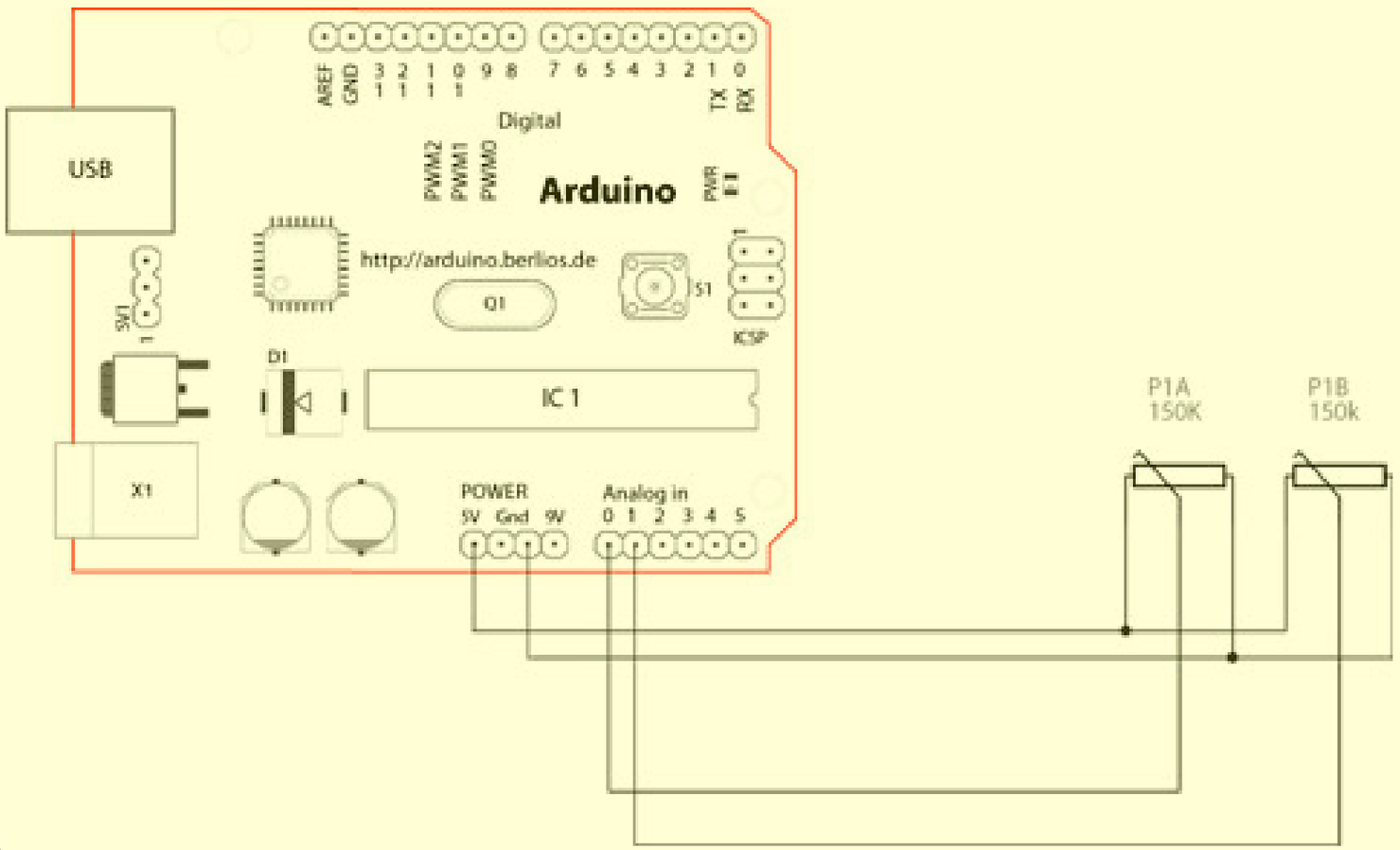
Alimenter l'Arduino



Avec une pile 9V et un connecteur
C'est une solution très pratique

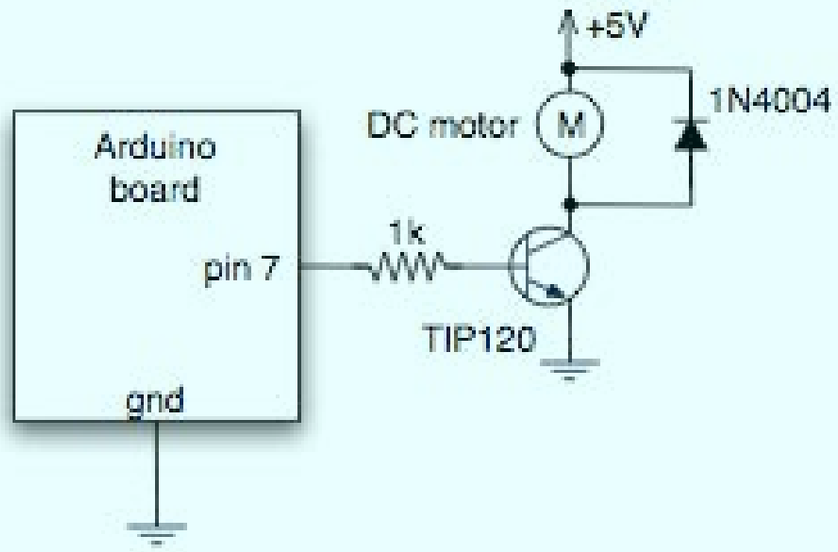
Avec un adaptateur secteur 9 à 12 V
Connecteur 2,1 mm avec le + au centre,
courant continu (DC)

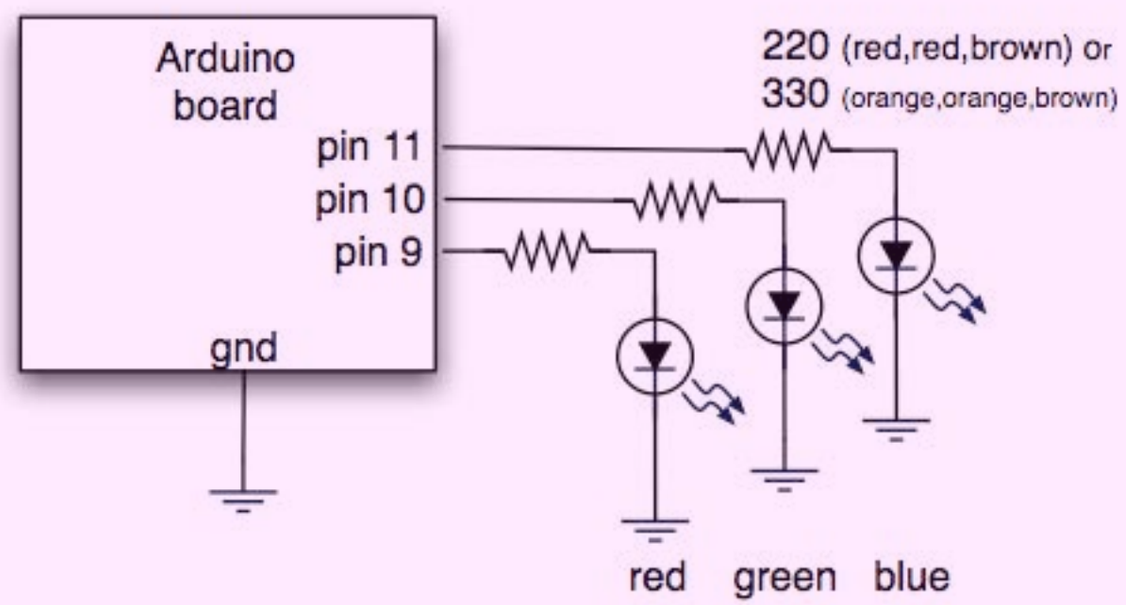
Brancher un joystick



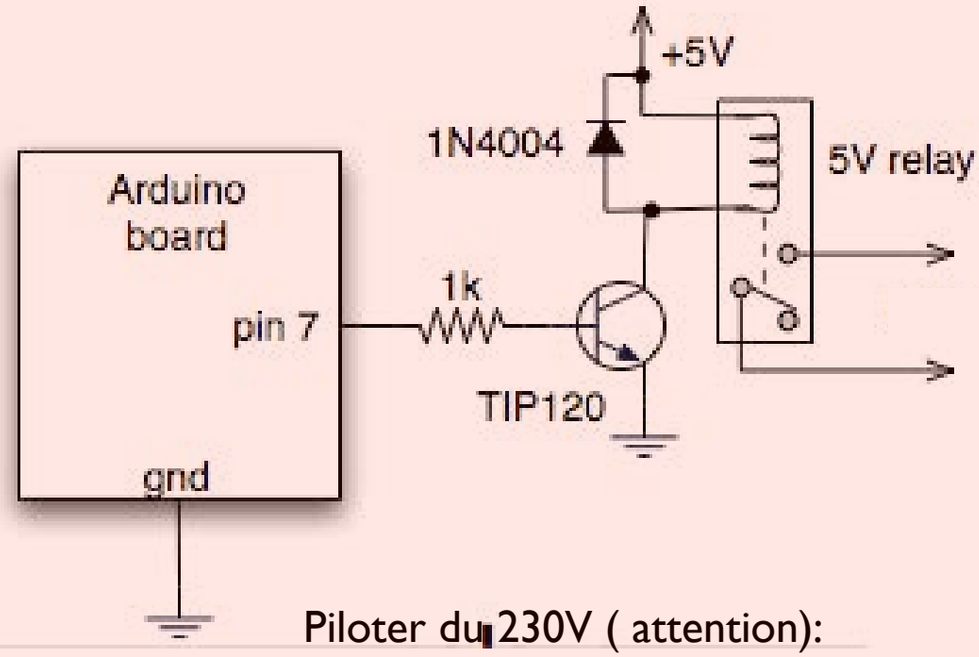
Quelques schémas à expérimenter

Piloter un moteur à courant continu:
(Résistance, transistor, moteur, diode, alim 5 V



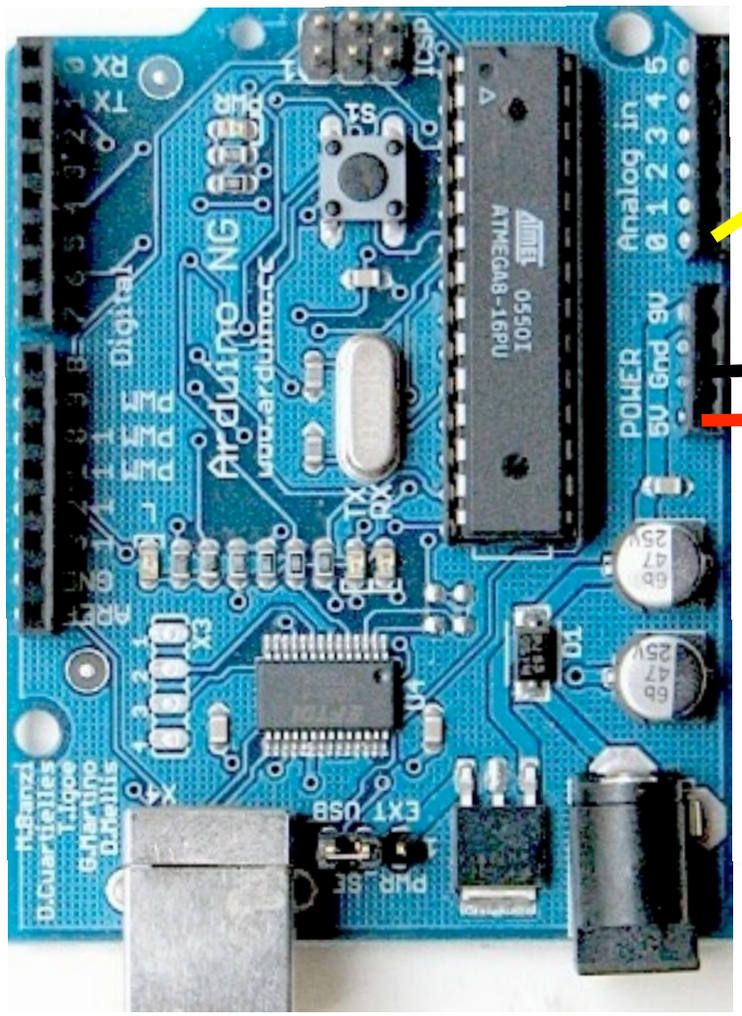


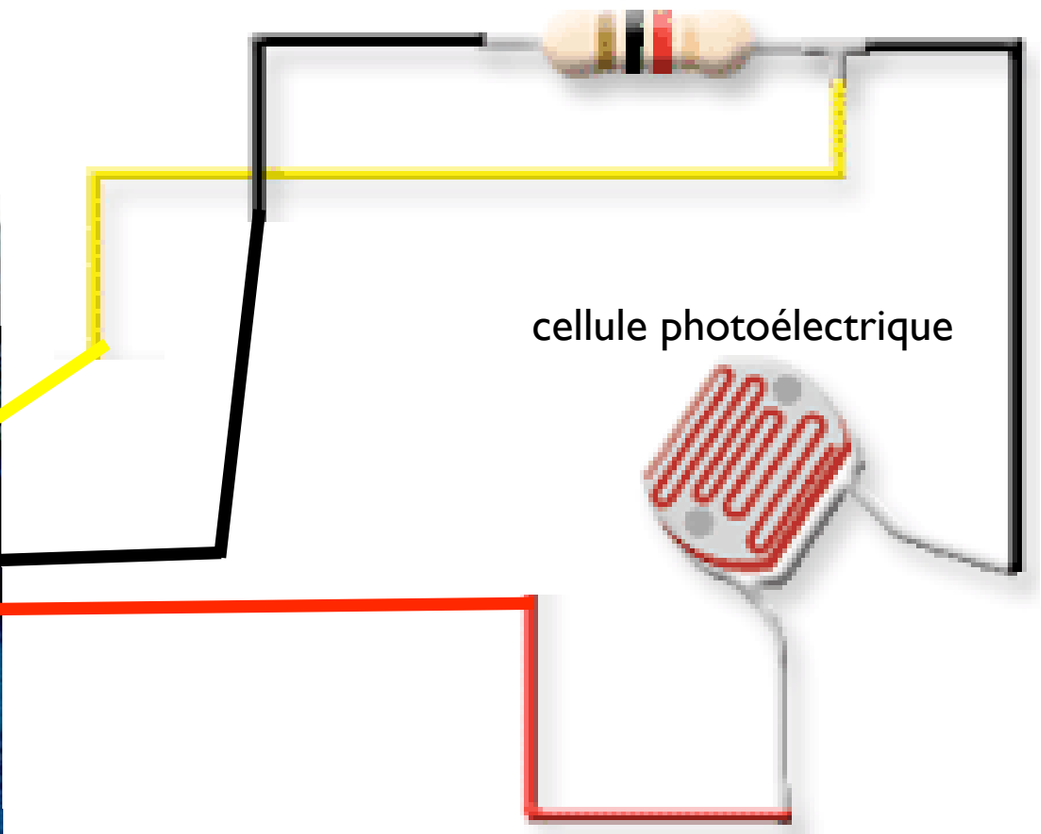
Piloter un éclairage à Leds RVB en milliers de couleurs, avec les sorties PWM
(Résistance, LED rouge, verte et bleue)

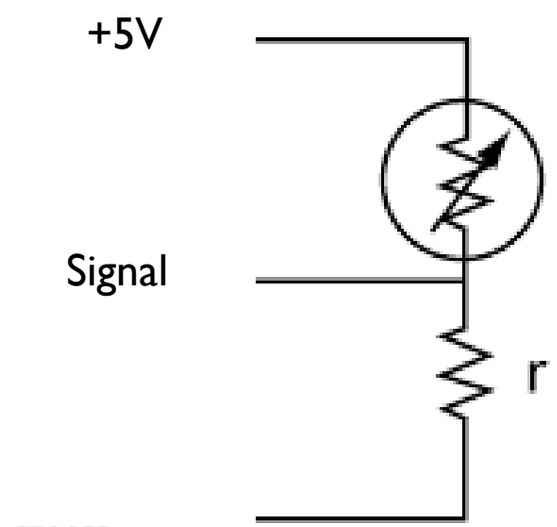


Piloter du 230V (attention):
(Résistance, transistor, diode, alim 5 V, relais)

Montage d’un capteur résistif







Montage d’un capteur résistif, type cellule photoélectrique, capteur de pression, capteur de flexion, etc:

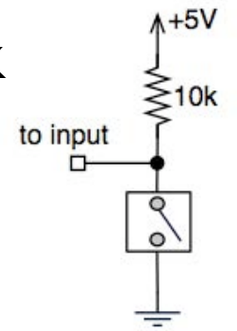
La résistance additionnelle doit être au moins équivalente à la plus forte valeur de résistance du capteur

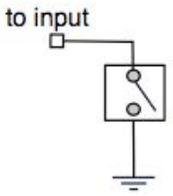
Conseil :Pour éviter qu’un fil ou qu’un composant branché au + vienne endommager un port USB dans l’ordinateur, isoler le métal du port USB avec un scotch d’électricien. Attention également au dessous de la carte, à ne pas poser sur un support conducteur.

Montage de boutons poussoirs et interrupteurs

Il y a deux solutions et donc deux configurations pour le montage de boutons poussoirs ou d'interrupteurs: il est nécessaire de respecter le cablage suivant, qui permet de se passer de résistances, car elles sont incluses dans le microcontroleur de l'Arduino.
Le montage ci-dessous comporte trois boutons poussoirs.

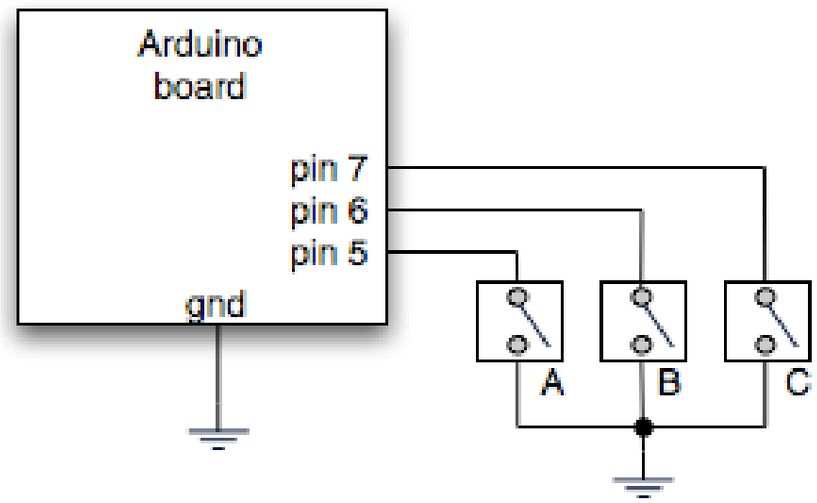
première solution:
avec des résistances de 10K





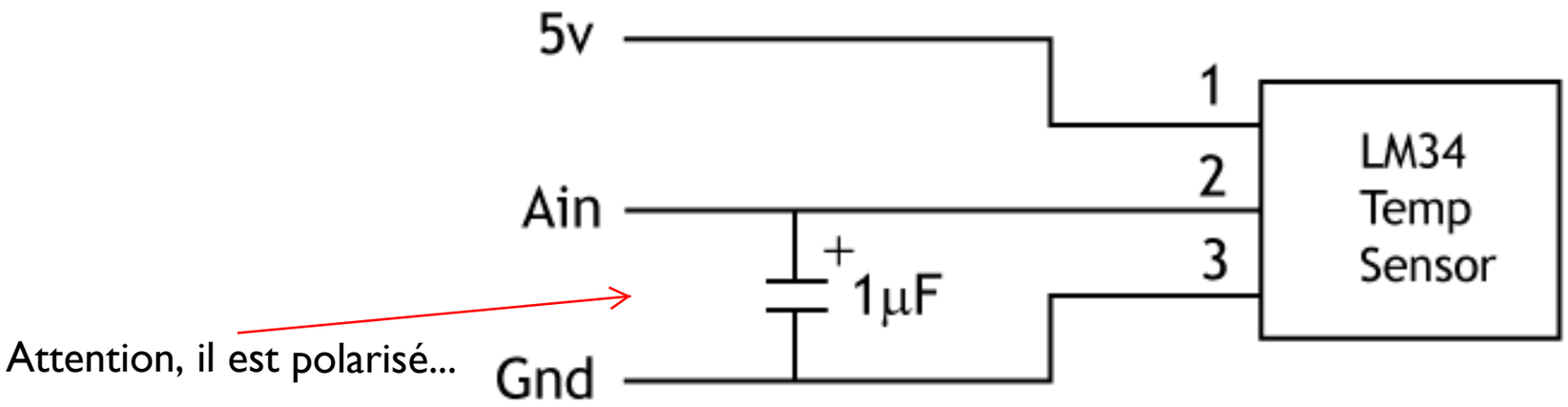
deuxième solution:
sans résistances mais avec ce code (ici pour trois interrupteurs)

```
void setup() {
  pinMode(switchAPin, INPUT);
  pinMode(switchBPin, INPUT);
  pinMode(switchCPin, INPUT);
  digitalWrite(switchAPin, HIGH);  // turn on internal pullup
  digitalWrite(switchBPin, HIGH);  // turn on internal pullup
  digitalWrite(switchCPin, HIGH);  // turn on internal pullup
}
```

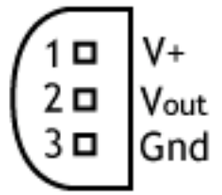


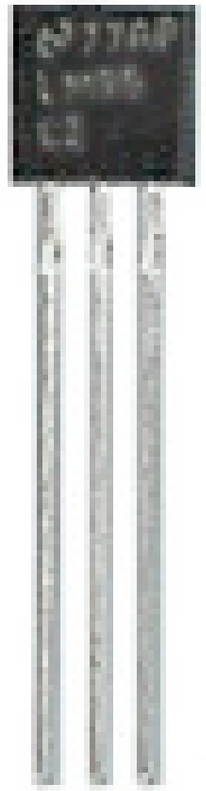
Le code impératif à respecter est:
digitalWrite(nomdelabrochenumérique, HIGH);

Cabler ce capteur de température sur la carte ?



Attention, il est polarisé...

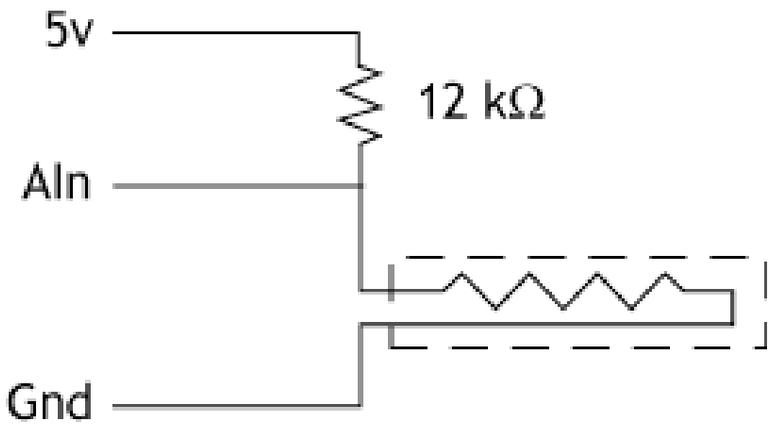


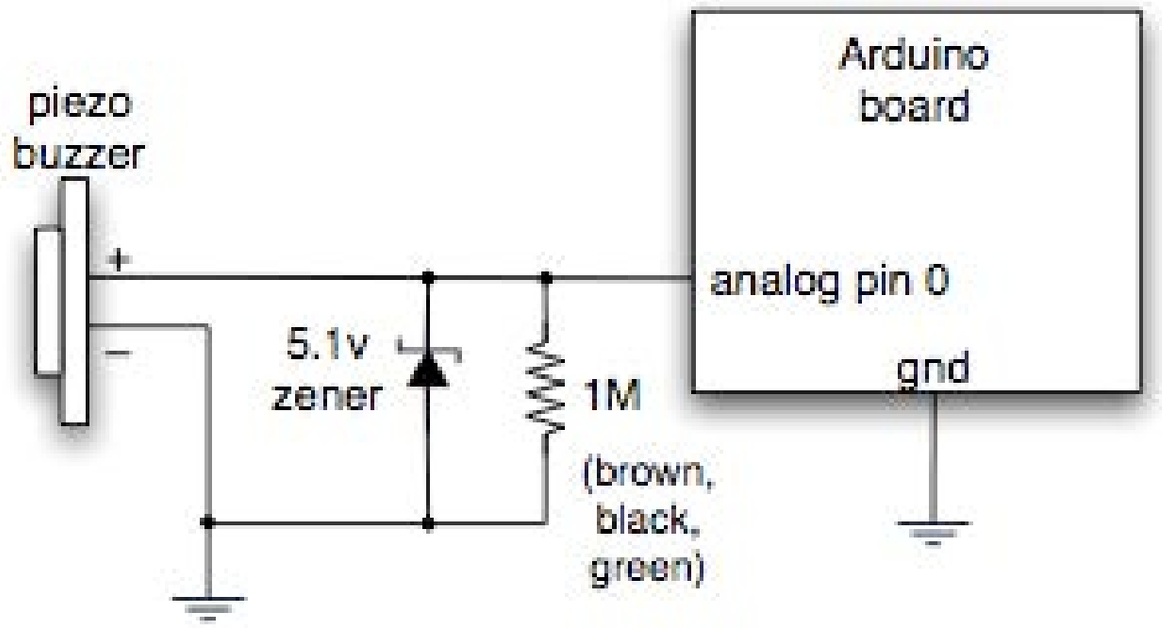


Ain = analog in = entrée analogique

Ce capteur de flexion (Bend sensor) ?

Ne pas oublier le condensateur





Ce capteur piezo-électrique,
en capteur de choc ?

piezo_read

```
  Serial.println("ready");       // indicate we're waiting
}


void loop() {
  digitalWrite(ledPin,LOW);      // indicate we're waiting

  val = analogRead(piezoPin);    // read piezo
  if( val >= THRESHOLD ) {       // is it bigger than our minimum?
    digitalWrite(ledPin, HIGH); // tell the world
    t = 0;
    while(analogRead(piezoPin) >= (THRESHOLD/2)) {
      t++;
    } // wait for it to go LOW  (with a little hysteresis)
    if(t!=0)
      Serial.println(t);
  }
}
```