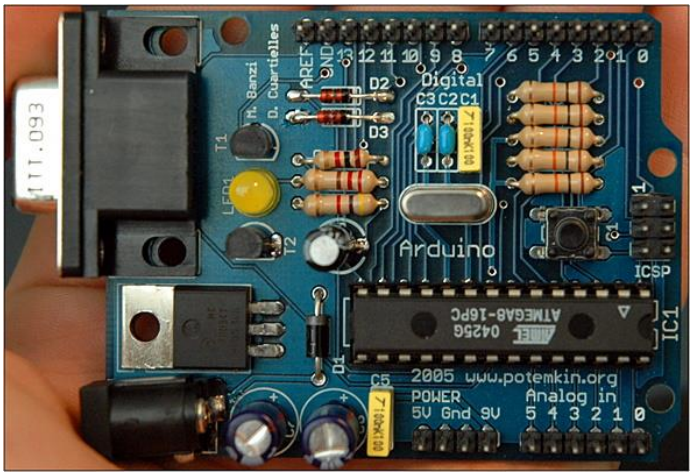
L'Arduino

Historique de l'Arduino

L’Arduino est à l’origine un projet d’étudiants de l’école de Design d’Interaction d’Ivrea en Italie. Au début des années 2000, les outils de conception de projets dans le domaine du design d’interaction étaient onéreux, proches d’une centaine d’euros.

Ces outils étaient pour la plupart conçus pour le domaine de l’ingénierie et de la robotique. Maîtriser et utiliser ces composants demandait beaucoup de temps et d’apprentissage et ralentissait fortement le processus de création pour ces jeunes étudiants.



Premier modèle de la carte Arduino

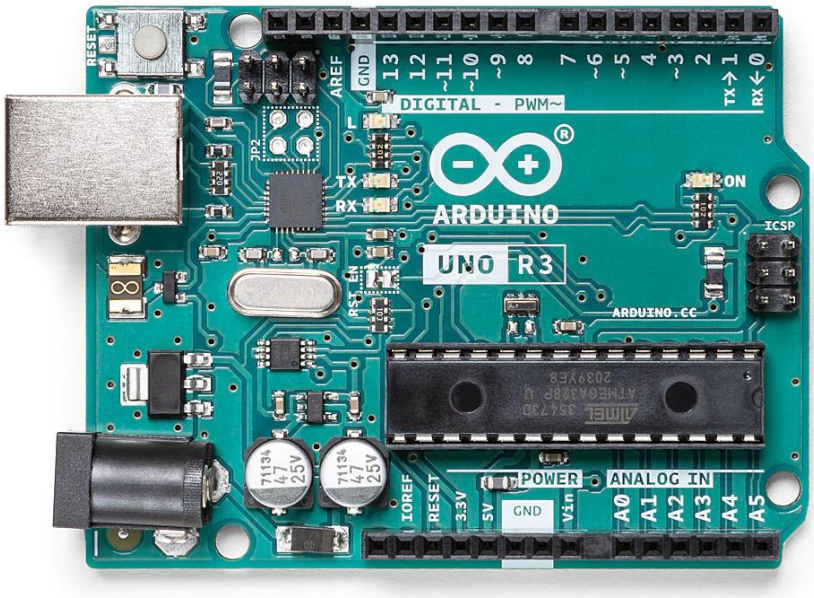
Il leur vient alors à l’idée de créer une plateforme plus abordable et plus simple à utiliser, reposant sur l’environnement de développement Processing mis au point en 2001 par des étudiants du MIT (Massachusetts Institute of Technology)

C’est donc en 2003 que, pour un projet de fin d’études, fut conçue la carte Wiring, ancêtre de l’Arduino. Visant à rendre la plateforme toujours moins chère et plus accessible, une équipe d’étudiants et de professeurs finirent par concevoir la toute première Arduino en 2005

Entièrement open source, l’Arduino présentait l’avantage d’être multiplateforme et d’être en perpétuelle optimisation par la communauté d’utilisateurs.

C’est en l’honneur de ce bar où Massimo Banzi a pour habitude d’étancher sa soif que fut nommé le projet électronique Arduino (dont il est le cofondateur). Arduino est une carte microcontrôleur à bas prix qui permet — même aux novices — de faire des choses époustouflantes. Vous pouvez connecter l’Arduino à toutes sortes de capteurs, lampes, moteurs, et autres appareils, et vous servir d’un logiciel facile à appréhender pour programmer le comportement de votre création.

L'Arduino Uno (rev3)



Spécifications Techniques e la carte Arduino Uno :

Microcontrôleur ATmega328P

Tension de fonctionnement 5V

Tension d'entrée (recommandée) 7-12V

Tension d'entrée (limite) 6-20V

Broches d'E/S numériques 14 (dont 6 fournissent une sortie PWM)

Broches d'E/S numériques PWM 6

Broches d'entrée analogique 6

Courant CC par broche d'E/S 20 mA

Courant continu pour broche 3,3 V 50 mA

Mémoire Flash 32 Ko (ATmega328P) dont 0,5 Ko utilisé par le bootloader

SRAM 2 Ko (ATmega328P)

EEPROM 1 Ko (ATmega328P)

Vitesse d'horloge 16 MHz

LED intégrée Pin 13

Longueur 68,6 mm Largeur 53,4 mm Poids 25 grammes

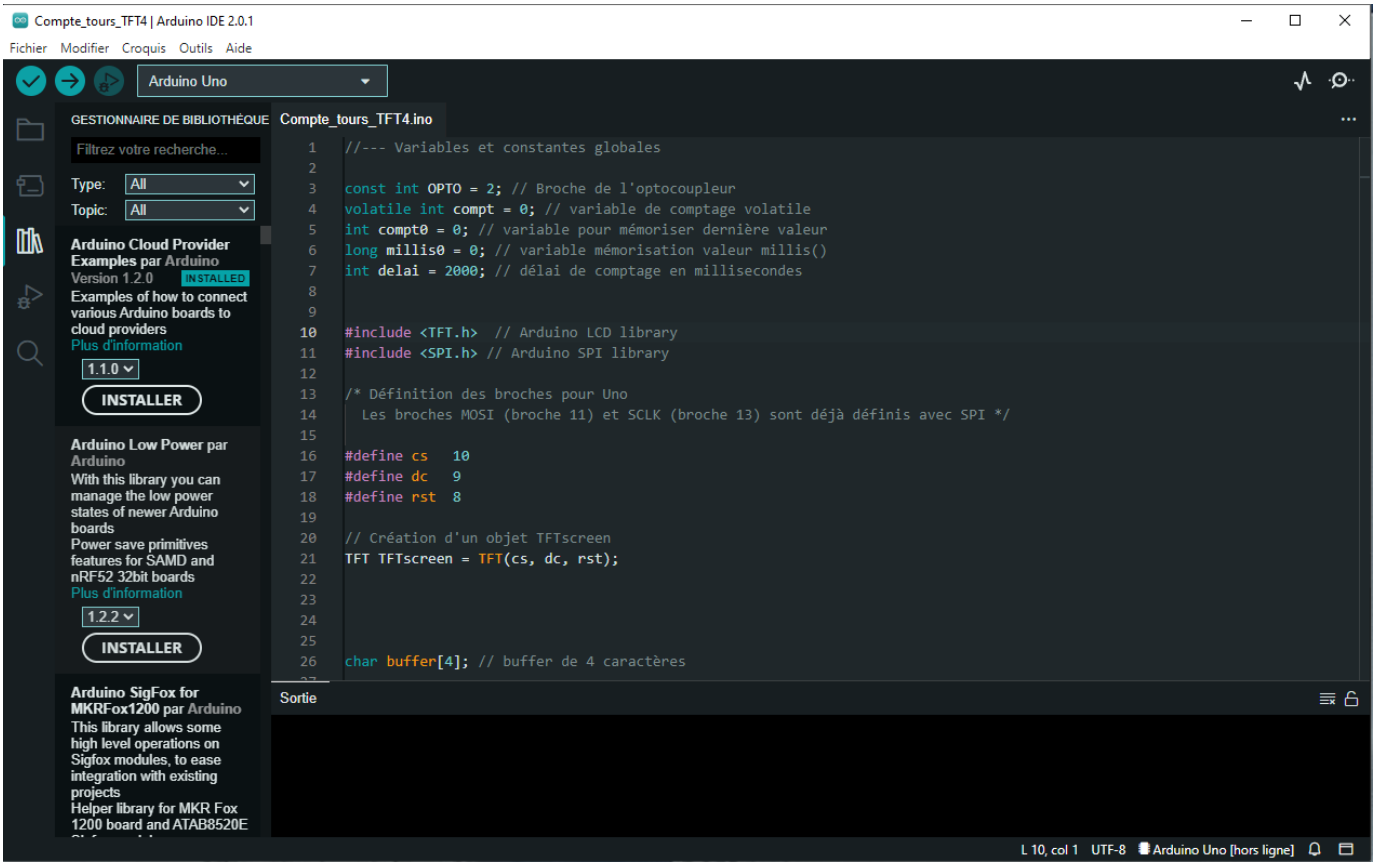
Logiciel

Le logiciel de programmation des modules Arduino, dont l'interface, appelée *Arduino IDE (*Integrated Development Environment), est une application Java, libre et multiplateforme dérivée de Processing servant d'éditeur de code et de compilateur, et qui peut transférer le firmware et le programme au travers de la liaison série (RS-232, Bluetooth ou USB selon le module).

Le langage de programmation utilisé est le C++, compilé avec avr-g++, et lié à la bibliothèque de développement Arduino, permettant d'utiliser la carte et ses entrées/sorties. La mise en place de ce langage standard rend aisé le développement de programmes sur les plates-formes Arduino à toute personne maîtrisant le C ou le C++.

Exemple d'écran avec IDE 2.0.1

(Mon compte tours TFT4)



Langage Arduino

https://www.arduino.cc/reference/en/

Interruptions :

Une interruption, consiste à interrompre momentanément le programme que l'Arduino exécute pour qu'il effectue un autre travail. Quand cet autre travail est terminé, l'Arduino retourne à l'exécution du programme et reprend à l'endroit exact où il l'avait laissé.

Cet autre travail s'appelle le **programme d'interruption** ou la **routine d'interruption** ou encore une *ISR* pour *Interrupt Service Routine* en anglais.

Les plus faciles à utiliser :

INT0 = Interruption externe sur la broche 2

INT1 = Interruption externe sur la broche 3

Exemple : Compte tours

La mesure utilise un capteur opto électronique associé à un disque qui comporte 30 lumières.

La sortie du capteur est reliée à la broche 2 de l'Arduino Uno (Int0)

(L'Arduino Uno comporte 2 broches d'interruption.)

L'alimentation 5 V du capteur est fournie par l'Arduino Uno

Au départ on lance une horloge (millis)

A chaque impulsion nouvelle sur la broche 2 l'exécution du programme est interrompue pour incrémenter un compteur (compt)

Le programme vérifie en permanence que le temps écoulé est inférieur au cycle de mesure que j'ai fixé arbitrairement à 2s (2000 ms)

Si le temps écoulé est de 2000 ms, la valeur du compteur (compt) est mémorisée et affichée.

On ne dispose que de 2s pour compter les impulsions, vérifier la valeur de l'horloge, effacer le nombre de tours N-1 calculer et afficher la valeur N du nombre de tours.

Par exemple si la vitesse est de 3000 Rpm, soit 50 tours / s l'Arduino devra compter 50 x 30 x 2 = 3000 impulsions en 2s

Si la vitesse est de 10 Rpm soit 10/60 tours /s l'Arduino va compter 10/60 x 30 x 2 = 10 impulsions en 2s

Bibliothèques Arduino

Les librairies (ou bibliothèques) sont un ensemble de fonctions permettant de simplifier l'utilisation d'un capteur ou d'une fonctionnalité. Dès qu'un programme Arduino contient une ligne commence par #include alors, il appelle une librairie. (Library)

Par exemple, si on utilise un afficheur LCD, un servomoteur ou encore si l'on utilise un shield Ethernet (réseau), on va intégrer dans notre programme la librairie dédiée correspondante.

Pour intégrer une librairie dans un programme Arduino, c'est très simple : il suffit d'ajouter en début de programme une ligne de la forme :

#include <nomlibrairie.h>

Exemples de Library :

La librairie Serial pour les communications séries entre la carte Arduino et l'ordinateur ou d'autres composants
La librairie LCD - pour l'utilisation et le contrôle d'un afficheur LCD alpha-numérique standard.
La librairie Servo - pour contrôler les servomoteurs.
La librairie Stepper - pour contrôler les moteurs pas à pas (nécessite une interface de commande)
La librairie Ethernet - pour se connecter à Internet en utilisant le module Arduino Ethernet
La librairie EEPROM - référence - pour lire et écrire dans la mémoire EEPROM non volatile.
La librairie SD - référence - pour utiliser une carte mémoire SD (utilis (utiliser des fichiers, stocker des données, ...)

Par exemple pour le compte tours j'utilise les librairies suivantes :

#include <TFT.h> // Library pour l'écran TFT

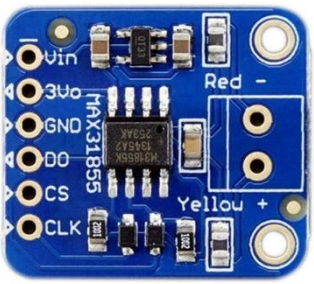
#include <SPI.h> // Library pour le protocole de communication entre l'Arduino et l'écran TFT

Quand on utilise une carte externe le fabricant fournit parfois une library spécifique:

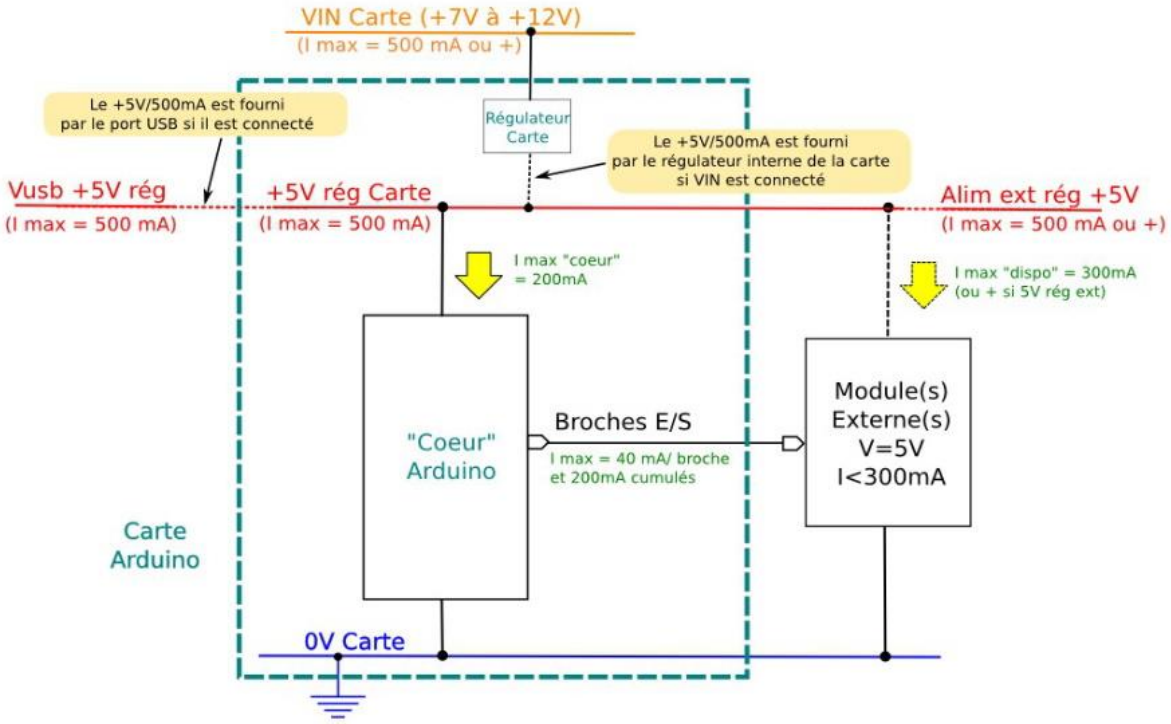
Application :

#include "Adafruit_MAX31855.h"

Library pour un Module de capteur de température Adafruit (MAX31855) Type K, pour Thermocouple, de -200 ℃ à + 1350 ℃



Alimentation de l'Arduino



Alimentation de l'Arduino

L'Arduino comporte en entrée un régulateur 5 V

On peut l'alimenter avec une fiche coaxiale (5.5 x 2.1 mm + au centre) avec une tension de 7 à 12V (7V à cause du régulateur 5 V) (Vin)

Si le courant de cette alimentation est de 0.5 A ou plus on pourra l'utiliser aussi pour fournir de la puissance à d'autres composants (Moteurs par exemple)

On peut aussi alimenter l'Arduino avec une prise USB. Dans ce cas on ne disposera que d'un courant de 0.5 A (Suffisant dans beaucoup d'applications)

On peut utiliser une Alimentation 5V. Dans ce cas on disposera d'une tension de 5V avec le courant que pourra fournir l'alimentation.

Dans tous les cas, la consommation des modules externes (Entrées / sorties de l'Arduino) ne peut dépasser un total de 0.3 A (300 mA)

Il convient de bien comprendre ce qui précède pour ne pas tout casser

Utilisation des entrées / Sorties de l'Arduino Uno

Sorties numériques :

Notions 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.

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.

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

Les broches numériques de la carte Arduino

La carte Arduino Uno 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.

D'un point de vue électrique, retenir que :

Chaque broche numérique E/S peut supporter une intensité de 40 mA en sortie ou en entrée sortie ou en entrée

L'ensemble des broches numériques E/S ne doit pas dépasser 200mA en entrée ou en sortie.

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

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

Pilotage d'une Led avec Arduino

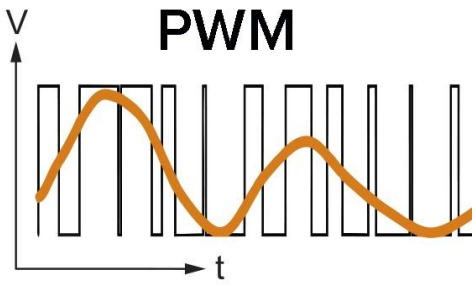
L'alimentation de l'Arduino n'est pas représentée

Avec ce montage et le programme adapté on peut faire clignoter la Led

Mais aussi faire varier l'intensité de la lumière :

L'Arduino dispose de 6 broches générant une PWM : **3, 5, 6, 9, 10 et 11**

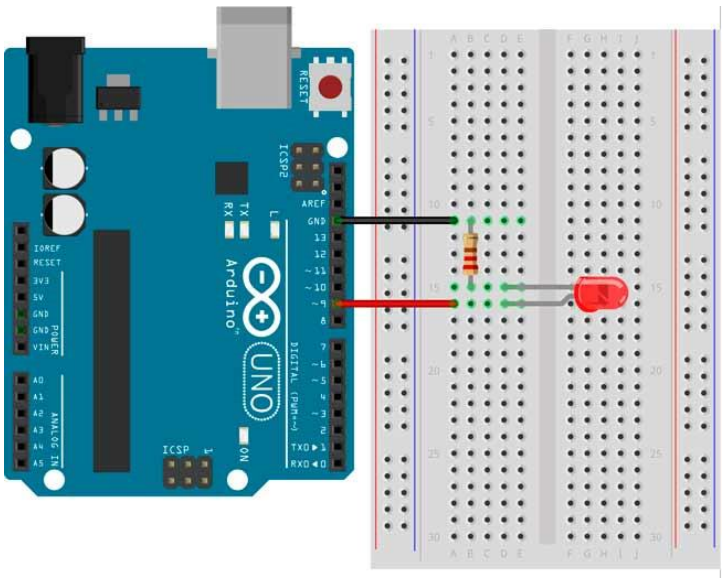
PWM : (Atelier Arduino 3a)



La modulation de largeur d'impulsions (MLI ; en anglais : *Pulse Width Modulation*, soit *PWM*), est une technique couramment utilisée pour synthétiser des signaux pseudo analogiques à l'aide de circuits numériques (tout ou rien, 1 ou 0)

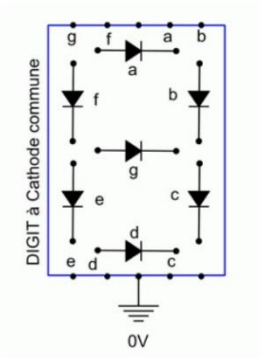
Elle sert à générer un signal pseudo analogique à partir d'un environnement numérique

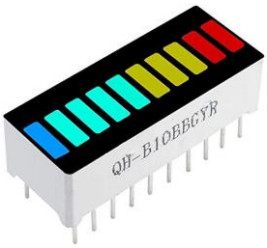
Le principe général est qu'en appliquant une succession rapide d'états discrets avec des ratios de durée bien choisis, on peut obtenir *en ne regardant que la valeur moyenne du signal* n'importe quelle valeur intermédiaire



On peut réaliser des montages plus complexes utilisant des Leds

Les Digits à Leds





L'utilisation des digits à LEDs en pratique, ce n'est pas très pratique.
Avec une carte Arduino, on pourra réaliser un montage avec 2 digits mais guère plus de façon simple : au-delà de 2 digits, la mise en œuvre des Digits peut être assez complexe et on utilisera alors des modules dédiés qui réalisent le multiplexage de l'affichage.

Interface de Puissance

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 quinzaine de broches en sortie simultanément, considérer que l'on peut utiliser sans danger 200mA/15 = 13mA / broche soit en pratique 10 à 15mA par broche.

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

De nombreux dispositifs ont caractéristiques bien différentes :

Un moteur à courant continu nécessitera 6 à 12V et 200mA à 2A

Un relais fonctionnera en 12V et nécessitera 100mA par exemple

Un ruban à LED fonctionnera en 12V et jusqu'à plusieurs A

Pour prendre une image, c'est comme si on voulait brancher une lance à incendie sur un robinet de cuisine

La solution passe par une "interface de puissance" : c'est un circuit électronique ou une carte électronique selon les cas, qui va assurer un double rôle :

Adapter la tension
Adapter l'intensité

De cette façon, à partir de la sortie numérique 5V/10mA, on va pouvoir contrôler des appareils :

Entre 5V et 12V voire 30V (adaptation en tension)
Consommant 100mA, 300mA voire plusieurs A (adaptation en intensité)

On pourra de cette façon potentiellement contrôler toutes sortes de dispositifs à partir d'une simple carte Arduino

Interface de puissance

Le but est de piloter l'environnement extérieur

On dispose de plusieurs possibilités parmi lesquelles il convient de choisir la plus adaptée.

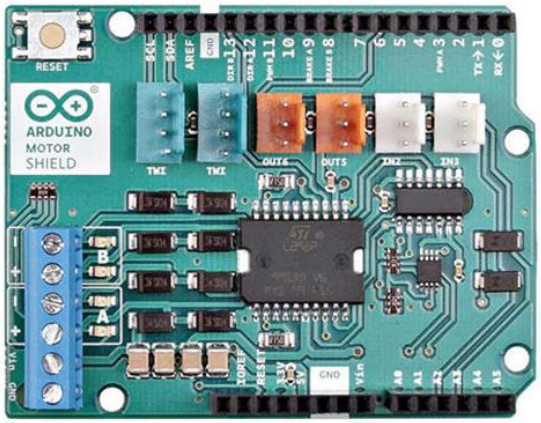
Par exemple : un transistor pour commander un relais

Un circuit intégré ULN 2803 :

L'utilisation de ce circuit est décrite dans l'Atelier Arduino "7d.atelier_arduino_moteurs_cc"

Les Shields :

Les Shields Arduino sont des cartes qui se branchent sans soudure aux cartes Arduino pour augmenter leur capacité (Wifi, 4G, écran, Bluetooth … Pilotage de moteurs, Cartes relais)



Shield Arduino Motor Shield Rev 3 (Atelier Arduino 7e)

Basée sur un circuit L298 (Double pont en H)

Permet de contrôler 2 moteurs CC ou 1 moteur Pas à pas bipolaire

Chaque moteur est contrôlé par une broche de sens et une broche de vitesse PWM

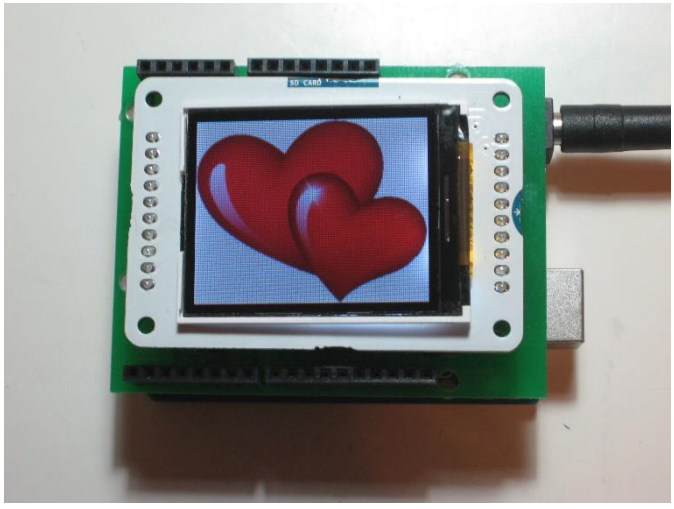
Intensité 2A / phase ou moteur (4 A en tout)

Capteur analogique d'intensité par phase ou moteur intégré

Alimentation 7 à 12 V DC

Bien entendu on peut fabriquer un shield dédié.

C'est ce que j'ai fait pour l'afficheur mon compte tours de fraiseuse :



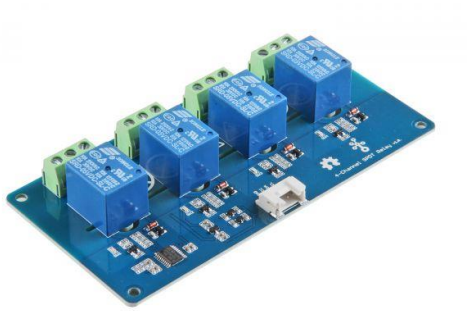
Il existe aussi des cartes qu'il est possible de relier par câblage aux entrées/sorties de l'Arduino

Les modules "Grove"

Grove est un système de connecteur plug-and-play open-source développé par Seeed Studio. Les modules Grove ont été conçus pour l'éducation et le prototypage rapide. Grove propose une large gamme de capteurs et d'actionneurs. Les modules sont enfichables sans soudure.

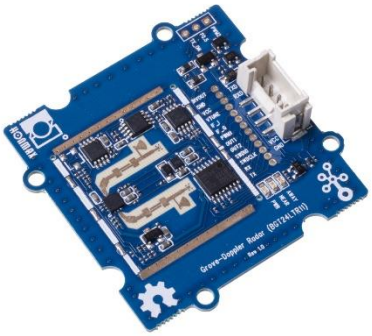
https://www.seeedstudio.com/category/Grove-c-1003.html?

Module 4 relais



Détecteur de tension

Radar Doppler



Les Entrées numériques

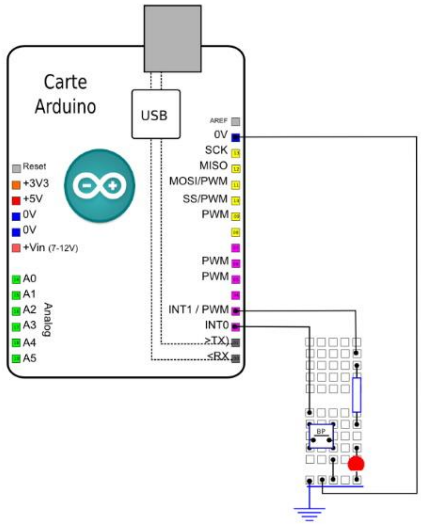
Exemple : Utilisation d'un bouton poussoir

Atelier Arduino 5a :

A comprendre :

Utilisation des résistances de rappel au + ainsi que les notions de rebond et anti rebond.

Exemple d'utilisation d'un bouton poussoir pour commander une LED



Atelier Arduino 5b :

Exemples de capteurs numériques :

Opto-coupleur en fourche

	Ce capteur livré **brut** est facile à utiliser avec 2 résistances complémentaires. Un opto-coupleur en fourche associe face à face une photo-diode et un photo-transistor : un objet passant dans la fente coupe le faisceau lumineux et sera détecté. Utilisable à grande vitesse (détection d'objet en rotation). Existe aussi en version large. Intéressant pour des comptages de vitesse de rotation (anémomètre, moteurs, etc...)

Dispo ici : http://www.gotronic.fr/art-interrupteur-optique-lth301-07-2325.htm

Capteur de ligne DFRobot

	Ce capteur livré **monté et prêt à connecter, livré avec son câble**. Il associe une photo-diode et un photo-transistor qui permettent la détection d'une ligne. Sensible, efficace et facile à utiliser. Idéalement par 3 pour optimiser le centrage sur la ligne avec un robot mobile. Niveau logique bas pour le NOIR, niveau logique haut pour le BLANC.

Dispo ici : http://www.zartronic.fr/module-suiveur-de-ligne-pour-arduino-p-129.html

Capteur détecteur de mouvement à infra-rouge

	Ce capteur livré **monté et prêt à connecter** permet de détecter le mouvement d'un être vivant dans son « champ de vision ». Nécessite de s'acclimater à la pièce au préalable. Assez sensible. Alim : 5-12V. Conseillé de l'alimenter en 12V.

Dispo ici : http://www.watterott.com/en/PIR-Motion-Sensor

Détecteur numérique de vibration

	Ce capteur livré **monté et prêt à connecter** est sensé détecter les vibrations... Utiliser une capsule piézo-électrique simple plutôt...

Dispo ici : http://www.zartronic.fr/capteur-num%C3%Agrique-de-vibrations-p-106.html

Détecteur numérique d'inclinaison (Tilt sensor)

	Ce capteur livré **monté et prêt à connecter** est sensé détecter l'inclinaison... capricieux... Utiliser plutôt un accéléromètre analogique...

Dispo ici : http://www.zartronic.fr/capteur-dinclinaison-num%C3%Agrique-tilt-compatible-arduino-p-105.html

Détecteur numérique capacitif de contact

	Ce capteur livré **monté et prêt à connecter**

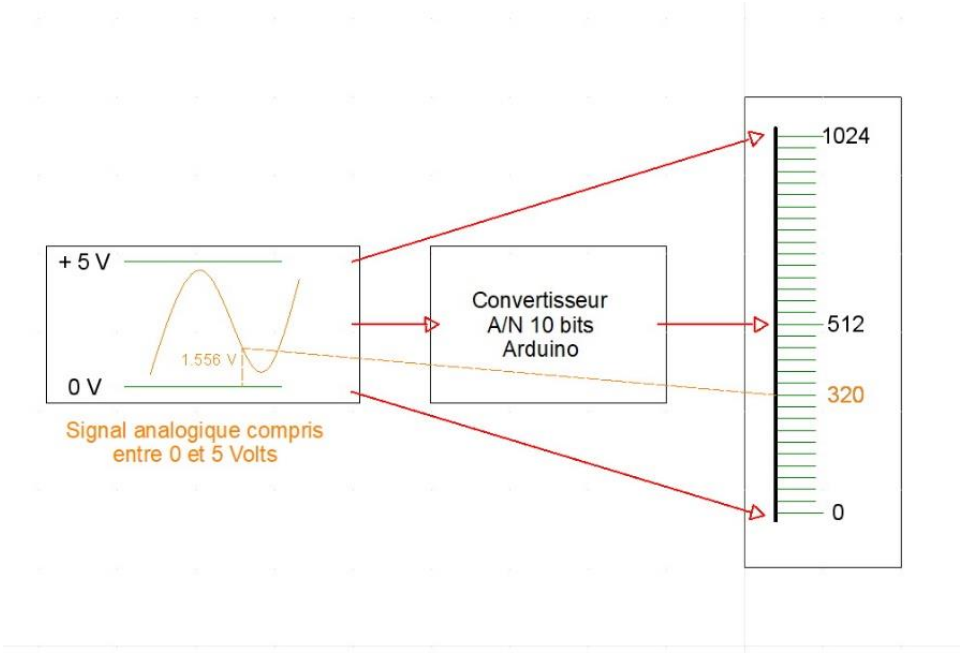
Dispo ici : http://www.alpha-crucis.com/fr/capteurs/3449-capacitive-touch-sensor-3700386600302.html

Les entrées Analogiques

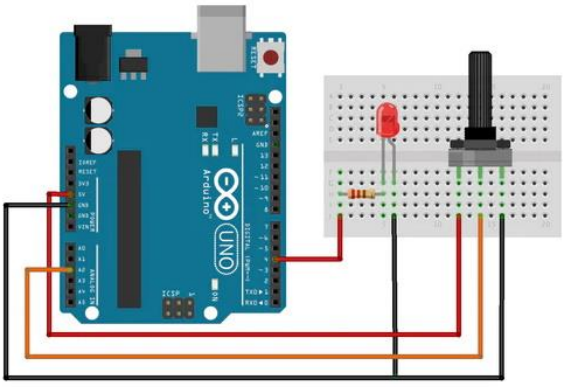
A savoir : les broches numériques 14 à 19 sont numérotées de 0 à 5 (ou désignées par A0, A1, A2, A3 et A5) lorsqu'elles sont utilisées en tant que broches analogiques.

Lorsqu' une tension analogique comprise entre 0 et 5 V est appliquée sur une de ces entrées

Elle est convertie par l'Arduino en nombre de 0 à 1024 par l'Arduino ce qui donne une résolution de 5 / 1024 Volts soit approximativement 0.005 V



Ci-dessous, on utilise un potentiomètre pour faire varier la fréquence de clignotement de la LED



Les Moteurs

Vue d'ensemble des différents types de moteurs

Un des intérêts majeurs d'une carte Arduino est de pouvoir réaliser le contrôle de motorisations variées selon ses besoins, en interaction avec des capteurs, des boutons poussoirs, etc... On pourra de la sorte construire assez simplement un robot motorisé et interactif intelligent

Vue d'ensemble des différents types de moteurs :

Il existe 3 grandes familles de motorisation utilisables avec Arduino, chacune ayant ses avantages et ses inconvénients comme nous allons le voir :

Les moteurs à courant continu (dits CC) avec:

Les moteurs CC simple (rotation rapide)
Les motoréducteurs (rotation plus lente avec engrenage de force)

Les servomoteurs qui se subdivisent en :

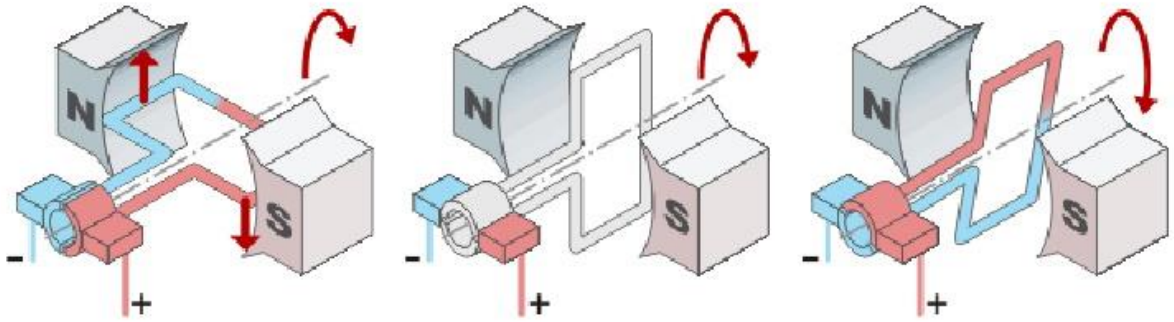
Servomoteurs standards (maintien de position entre 0° et 180°)
Servomoteurs à rotation continue (rotation continue)

Les moteurs dits "pas-à-pas" (rotation à vitesse lente ou moyenne de précision) avec 2 technologies :

Les moteurs pas à pas unipolaires
Les moteurs pas à pas bipolaires

Les moteurs à courant continu

Principe :



Le moteur à courant continu

Un moteur à courant continu est mis en rotation grâce à une force magnétique induite: la force de LAPLACE.
Cette force s'applique à un conducteur parcouru par un courant et placé dans un champ magnétique.
Les pôles Nord et Sud des aimants permanents créent un flux dans le moteur. La spire est alimentée et plongée dans ce flux. Elle est soumise à un couple de forces F (force de Laplace). Le moteur se met en rotation. On dit qu'il y a création d'un couple moteur. Compte tenu de la disposition des balais et du collecteur, le sens du courant I dans la spire change à chaque demi-tour, ce qui permet de conserver le même sens de rotation (sinon, la spire resterait en position d'équilibre)

Moteur avec réducteur

Caractéristiques d'un moteur à courant continu :

Un moteur ou un moto-réducteur à courant continu tourne dans un sens lorsqu'il est mis sous tension. Le sens de rotation est inversé, si on inverse la polarité de l'alimentation.

Type de mouvement :

Rotation continue maximale très rapide et à couple moyen à faible pour les moteurs CC simples (plusieurs milliers de tours par minute)

Rotation continue maximale moins rapide (dizaines ou centaines de tours par minute) avec un couple élevé pour les moto-réducteurs.

Brochage :

Un moteur ou un moto-réducteur à rotation continue dispose de 2 broches d'alimentation le 0V et le +V

Le sens de rotation d'un moteur ou moto-réducteur CC est contrôlé par la polarité d'alimentation : le moteur tourne dans un sens à la mise sous tension et tourne dans l'autre sens si on inverse la polarité.

La vitesse de rotation d'un moteur CC (supporte une plage d'alimentation +/- large selon les modèles) peut être contrôlée par :

La tension d'alimentation utilisée : plus elle est élevée et plus la vitesse est rapide,
La largeur de l'impulsion PWM utilisée pour l'alimenter

Caractéristiques mécaniques :

Un moteur ou un moto-réducteur CC se caractérise par :
◦ la plage d'alimentation supportée : par exemple de 6 à 12V
◦ l'intensité à vide et bloquée (plusieurs centaine de mA voire plusieurs A)

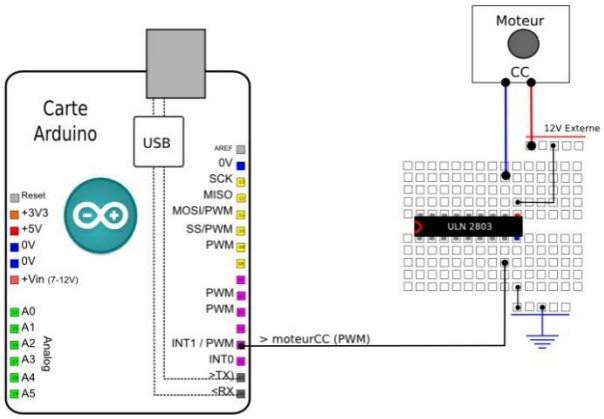
Son couple en g.cm ou Kg.cm, bloqué et à vide
Sa vitesse de rotation en tours/minute, bloqué et à vide

Caractéristiques électriques :

La plage de tension d'alimentation est variable selon les moteurs, entre 3V à 30V selon les modèles. Avec Arduino, prendre un modèle 6-12V.

L'intensité de fonctionnement est de l'ordre de quelques centaines de mA à quelques A selon les modèles : une interface est indispensable

Utilisation d'un circuit intégré ULN 2803 comme interface de puissance.



A noter : Ce montage permet de contrôler la vitesse du moteur avec une impulsion PWM

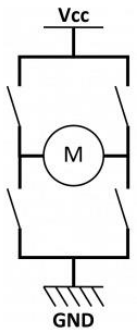
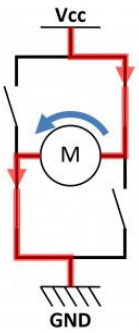
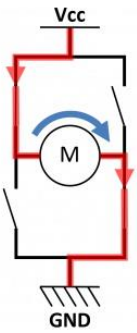
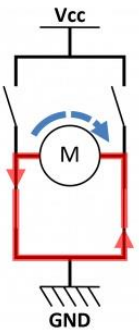
Mais ne permet pas de contrôler le sens du moteur.

Si l'intensité absorbée par le moteur est > 500 mA, on peut coupler plusieurs sorties du ULN 2803 ensemble ainsi que les entrées correspondantes.

Si on veut contrôler aussi le sens du moteur Il faut utiliser un circuit ayant un pont en H

Principe :

Les ports In1, In2 pour le moteur A et In3, In4 pour le moteur B, permettent de contrôler le pont en H et par conséquent le sens de rotation des moteurs. Par exemple, pour le moteur A :

		Arrêt (moteur libre)	Sens +	Sens –	Arrêt (moteur freiné)
					
Moteur A	Moteur B				
In1	In3	LOW	HIGH	LOW	HIGH
In2	In4	LOW	LOW	HIGH	HIGH

Les 2 broches numériques (In1, In2) et (In3, In4) permettent ainsi de contrôler les 2 sens de rotation :

Tout se passe au final comme si chaque broche au niveau HAUT contrôlait un sens.

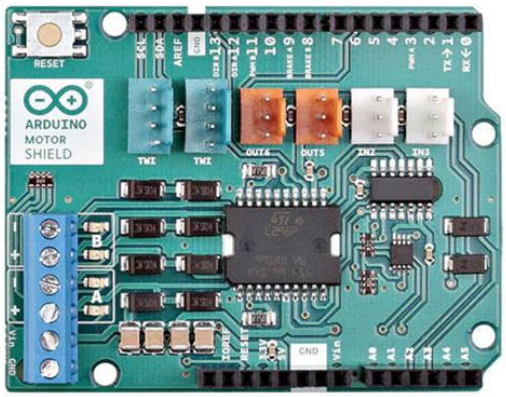
L'Arduino motor Sield rev 3 :

Ce Shield comporte un double pont en H permettant de piloter 2 moteurs CC ou un moteur pas à pas bipolaire.Il est basé sur le circuit intégré L298

Caractéristiques :

Alimentation : 7 à 12 V

Courant max 2 A / Canal – 4 A au total



Broches de direction: Canal A = D12 / Canal B = D13

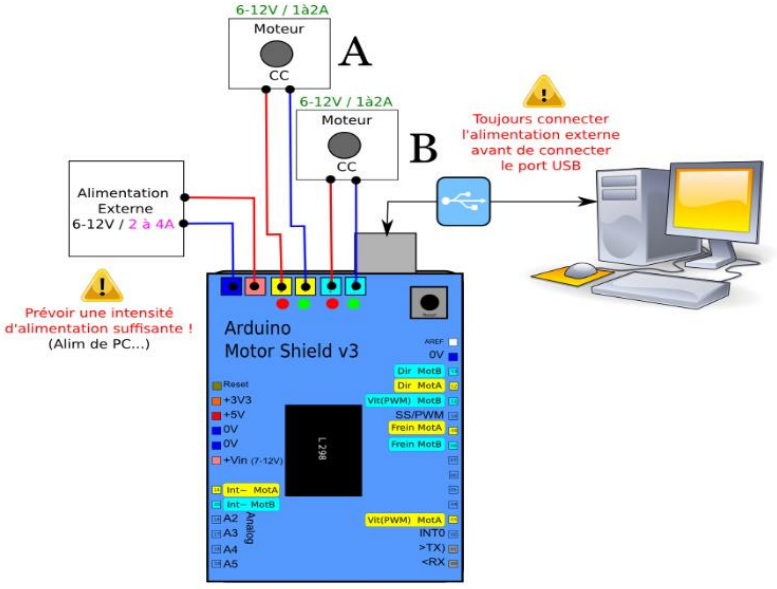
Broches de vitesse PWM: Canal A = D3 / Canal B = D11

Frein : Canal A = D9 / Canal B = D8

Mesure du courant canal A = A0 / Canal B = A1

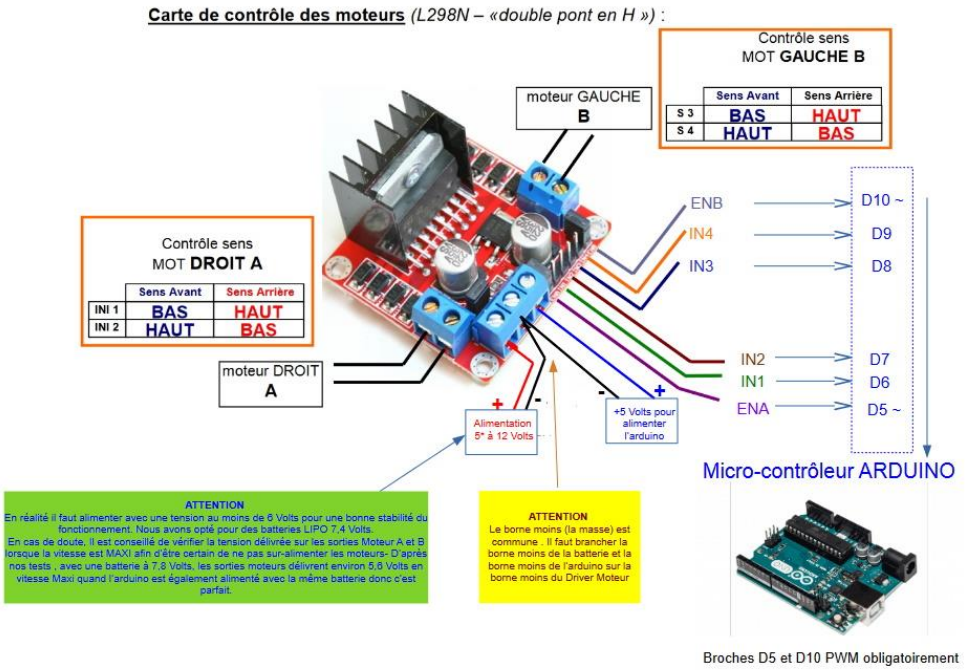
A gauche en bas le bornier bleu permet de brancher l'alimentation et les deux moteurs CC ou les 4 fils du moteur pas à pas bipolaire.

Branchement de 2 moteurs CC sur le Shield Arduino Motor Shield :



Un autre circuit avec double pont en H basé sur le circuit intégré L298 :

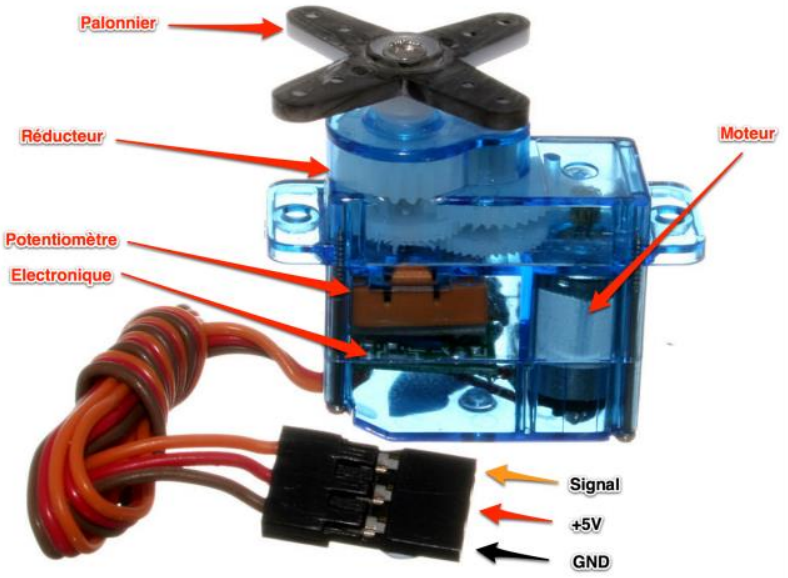
Là, il ne s'agit pas d'un shield qui s'enfiche sur l'Arduino Uno



Les Servomoteurs

Un servomoteur de loisir standard se compose généralement d'un petit moteur électrique, d'un potentiomètre, d'une électronique de commande et d'une boîte de vitesses. La position de l'arbre de sortie est constamment mesurée par le potentiomètre interne et comparée à la position cible définie par le contrôleur (par exemple l'Arduino).

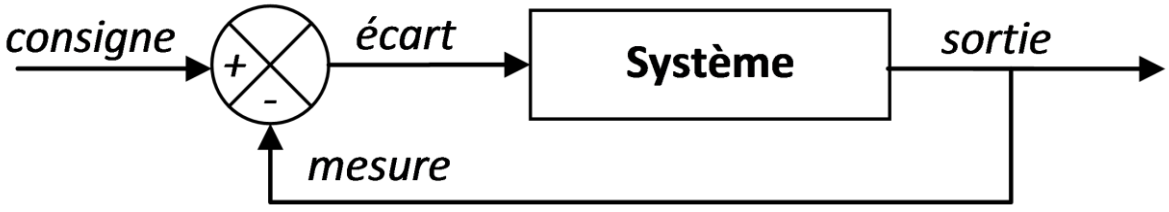
En fonction de l'erreur, l'électronique de commande ajuste la position réelle de l'arbre de sortie afin qu'elle corresponde à la position cible. C'est un système de contrôle en boucle fermée.



Ateliers Servomoteurs de mon Club Elec :

7b / 07b2 / 7c

Je n'ai pas encore fait les exercices proposés dans ces ateliers. Il est possible que j'apporte des compléments quand je les aurai faits.



Les servomoteurs standards : fiche technique.

Description :

Un servomoteur est un boitier plastique contenant un moteur associé
à une mécanique et une électronique internes permettant d'assurer le
maintien de position à l'angle voulu de l'axe du servomoteur (un peu à la manière d'un
gouvernail d'un bateau)

Brochage

Un servomoteur dispose d'un connecteur de 3 broches:
2 broches d'alimentation +5V et 0V,
1 broche de commande de type numérique.

Principe de contrôle du servomoteur

Un servomoteur standard est contrôlé par 1 broche numérique par
impulsion de type PWM : La largeur de l'impulsion va fixer la position
de l'axe du servomoteur entre 0° ou 180° voire 360° selon les modèles.

Caractéristiques mécaniques

• Un servomoteur est caractérisé par:
Son couple (ex: 3.2 kg/cm) et sa vitesse de positionnement (ex : 0.23 sec / 60°)

Caractéristiques électriques

L'alimentation du servomoteur nécessite une tension entre 4,8 et 6V typiquement
Et une intensité de 100mA pour un modèle de base, voire beaucoup plus.

La broche de commande du servomoteur est de type numérique (0V /
+5V) et consomme quelques mA.

Maintien de position "hors tension" Un servomoteur est capable de maintenir raisonnablement
une position bloquée "Hors alimentation" (Résistance mécanique modérée)

La photo : Servomoteur analogique à rotation continue Ref. FS 510 3R de Feetech

Avec palonniers et vis de montage.

Acheté chez GOTRONIC

Caractéristiques :

Alimentation: 4,8 à 6 Vcc.

Course: 360°

Consommation à 6 Vcc:
- à vide: 180 mA
- bloqué: 850 mA

Couple de blocage:
- 3 kg.cm à 4,8 Vcc
- 3,2 kg.cm à 6 Vcc

Vitesse:
- 57 t/min en 4,8 Vcc
- 64 t/min en 6 Vcc

Axe de sortie à 25 dents

Longueur du cordon: 300 mm

Température de service: -15 à 70 °C

Dimensions: 41 x 20 x 38 mm

Poids: 40 gr

Codage

Facile à mettre en œuvre à l'aide de la librairie Servo du langage Arduino. Mise en position voulue en 1 ligne seulement :

#include <Servo.h> // inclut la librairie Servo

La librairie dispose des fonctions suivante:
attach() : attache le servomoteur à une broche
write() : positionne le servomoteur à l'angle voulu
writeMicroseconds() : génère une impulsion PWM de la largeur indiquée en microsecondes
read() : renvoie l'angle actuel du servomoteur
boolean attached() : renvoie true si servomoteur attaché à une broche
detach() : détache le servomoteur d'une broche

Pour orienter l'axe du servomoteur à l'angle voulu, il faut appliquer sur la broche de contrôle de type numérique une impulsion PWM un peu particulière ayant une largeur de durée précise.

Typiquement :
1ms (1000µs) pour la position 0°
1,5 ms (1500µs) pour la position médiane 90°
2 ms (2000 µs) pour la position 180°

Toutes les positions intermédiaires sont possibles
La période de cette impulsion PWM est de l'ordre de 20 ms.

Les valeurs données ici sont indicatives et peuvent varier selon le modèle de servomoteur utilisé : il faudra donc réaliser un test de position au préalable pour connaître la largeur de l'impulsion des positions extrêmes.

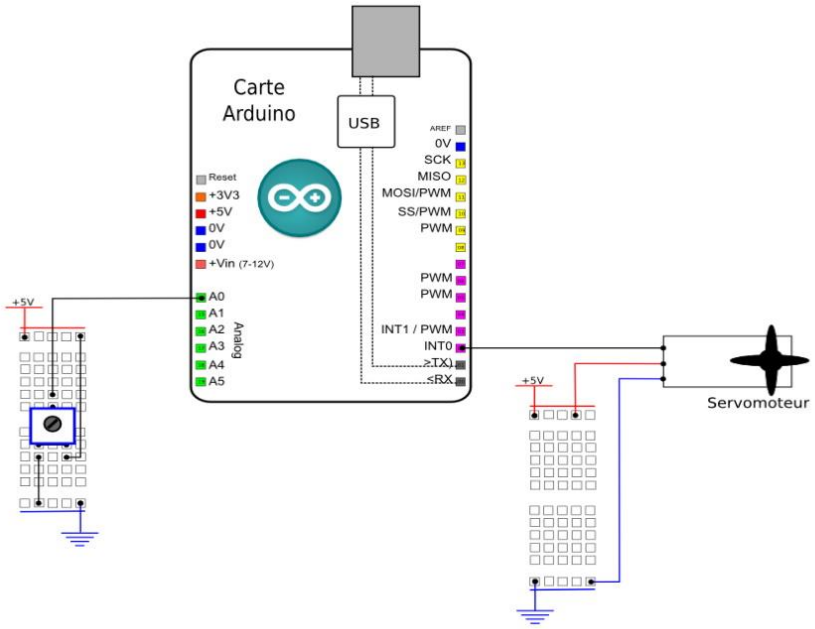


Schéma : Position de l'axe d'un servomoteur en fonction de la valeur d'une résistance variable

Les servomoteurs à rotation continue (Atelier 7c)

Un servomoteur à rotation continue est comparable à un servomoteur classique : c'est un boitier plastique contenant un moteur associé à une mécanique et une électronique internes contrôlant la rotation continue de l'axe.

Type de mouvement

La différence majeure avec le servomoteur standard réside dans le type de mouvement de l'axe : ici, l'axe tourne de façon continue en fonction de l'impulsion reçue par le servomoteur : le sens et la vitesse de rotation sont contrôlés simplement par la largeur d'impulsion

Brochage

Un servomoteur dispose d'un connecteur de 3 broches :
2 broches d'alimentation +5V et 0V,
1 broche de commande de type numérique.

Principe de contrôle du servomoteur

Un servomoteur à rotation continue est contrôlé par 1 broche numérique par impulsion de type PWM : la largeur de l'impulsion va fixer le sens et la vitesse de rotation

Caractéristiques mécaniques

Un servomoteur à rotation continue est caractérisé par sa vitesse de rotation maximale et son couple.

Caractéristiques électriques

L'alimentation du servomoteur nécessite une tension entre 4,8 et 6V typiquement et une intensité de 100mA pour un modèle de base, voire beaucoup plus

La broche de commande du servomoteur est de type numérique (0V / +5V) et consomme quelques mA.

Maintien de position "hors tension"
Un servomoteur est capable de maintenir raisonnablement une position bloquée "hors alimentation" (résistance mécanique modérée)

Principe de contrôle du servomoteur:

Un servomoteur à rotation continue est contrôlé tout comme un servo standard par 1 broche numérique par impulsion de type PWM. Il existe cependant une différence notable : la largeur de l'impulsion va fixer simultanément la vitesse de rotation et le sens de rotation
L'arrêt sera obtenu pour l'impulsion de « position » neutre.
Le sens est déterminé par la largeur de l'impulsion par rapport à l'impulsion neutre :

<1,5ms (1500µs) = rotation dans un sens
1,5 ms (1500µs) pour la « position » médiane = arrêt
>1,5 ms (1500µs) = rotation dans l'autre sens

La vitesse variera dans un sens comme dans l'autre en fonction de l'écart entre la largeur de l'impulsion neutre et la largeur courante :

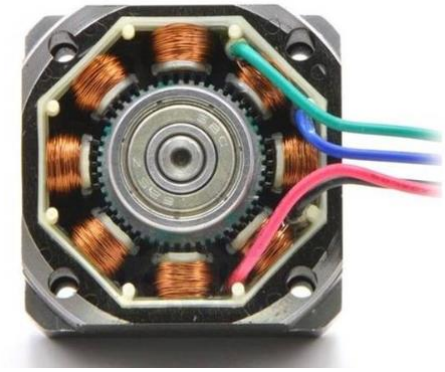
On aura 100% de la vitesse dans un sens à 1ms, 50% à 1,25ms, etc...
On aura 100% de la vitesse dans l'autre sens à 2ms, 50% à 1,75ms, etc...
La période de cette impulsion PWM est de l'ordre de 20 ms.

Les valeurs données ici sont indicatives et peuvent varier selon le modèle de servomoteur utilisé : il faudra donc réaliser un test de position au préalable pour connaître la largeur de l'impulsion de la position médiane d'arrêt.

Codage:

Utilise la librairie "Servo" du langage Arduino

Les moteurs Pas à pas



Le type de moteur

Il existe au moins 2 grands types de moteurs pas à pas :

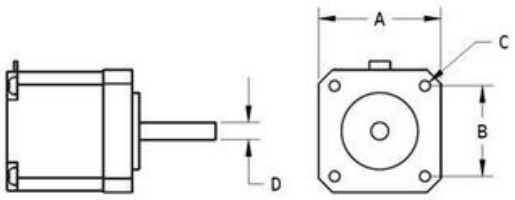
- les moteurs pas à pas bipolaires (2 bobines = 4 fils)
- les moteurs pas à pas unipolaires (2 bobines + 1 commun central = 5 fils)

Le modèle à prendre en pratique est le modèle bipolaire (4 fils de sortie), c'est celui dont nous parlons ici.

La taille du moteur

Les moteurs pas à pas conseillés sont les moteurs de type NEMA (Pour National Electrical Manufacturers Association) dont les dimensions sont standardisées. On caractérise un moteur NEMA par un chiffre NEMA 14, NEMA 17, NEMA 23, NEMA 34, etc. Le chiffre définit la dimension de la façade (carrée) et la position des 4 trous de fixation.

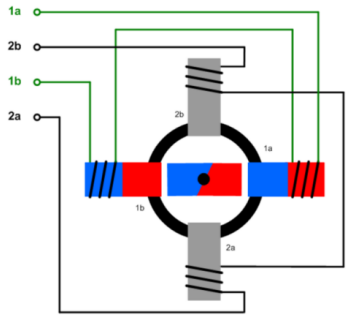
Les dimensions des façades sont les suivantes :



SIZE	A	B	C	D (Dia)
NEMA 11	28.2	23	M2.5 Thread	5
NEMA 14	35.2	26	M3 Thread	5
NEMA 17	42.3	31	M3 Thread	5
NEMA 23	56.4	47.1	5.5 Dia	6.35
NEMA 34	86	69.6	5.5 Dia.	14
NEMA 42	110	89	8.5	19

Dans chaque dimension de façade, on a des moteurs de longueurs différentes, qui correspondent à des modèles ayant des couples de plus en plus élevés.

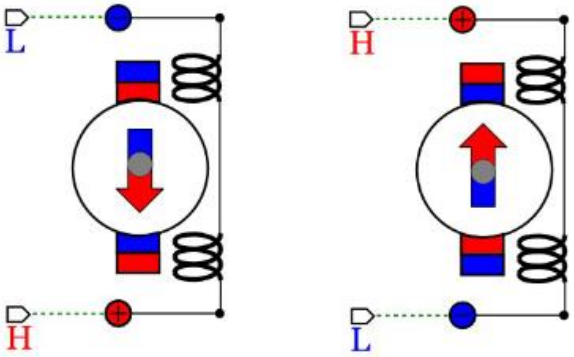
Schéma d'un moteur pas à pas bipolaire (Atelier Arduino 7e)



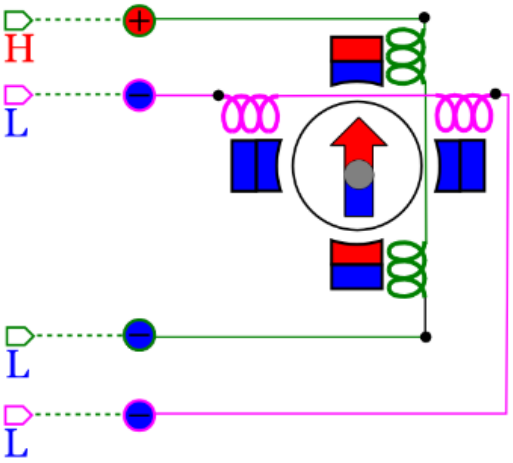
Le moteur pas à pas est construit autour de 2 bobines appelées phases (ici chaque bobine est divisée en 2 sous-bobines reliées entre-elles) et contrôlées chacune par 2 broches, permettant d'assurer la rotation de l'axe par "pas"

Principe de contrôle à l'aide d'un double pont en H :

Une bobine peut être assimilée à un moteur d'un point de vue électrique et dès lors son contrôle pourra se faire de la même façon qu'un moteur CC, à savoir à l'aide d'un pont en H :

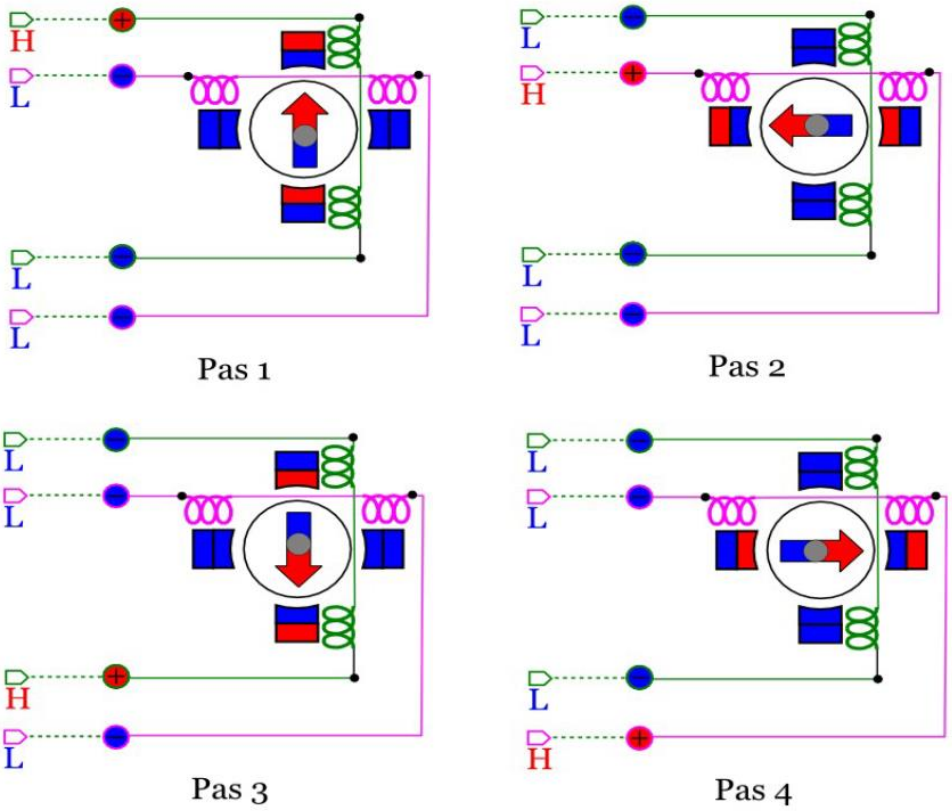


De la même façon le contrôle des 2 bobines (ou phases) nécessitera 2 ponts en H ou "double-pont en H" Vous comprenez ainsi pourquoi une interface capable de contrôler 2 moteurs CC sera capable de contrôler 1 moteur pas à pas bipolaire.



Comment réaliser le contrôle du moteur pas à pas bipolaire à partir de 4 broches numériques de commandes et un double "pont en H" de façon à obtenir une rotation :

Mouvement du moteur pas à pas



Dans un moteur pas à pas réel, les bobines de phases sont subdivisées en plusieurs sous-bobines de façon à entraîner de façon harmonieuse le rotor cranté et qui possède un grand nombre de pas (typiquement 200 soit 1,8 ° / pas)

La tension du moteur n'est pas essentielle (La commande est faite en courant) Et un étage pourra fonctionner indifféremment avec un moteur 2.3V ou 12V

L'important, c'est de calculer la puissance maximale que peut recevoir le moteur (P = tension nominale x I max) et l'adapter à la tension utilisée. En pratique, la température de fonctionnement du moteur sera un bon moyen: tiède çà ira, chaud faudra baisser l'I max de l'étage.

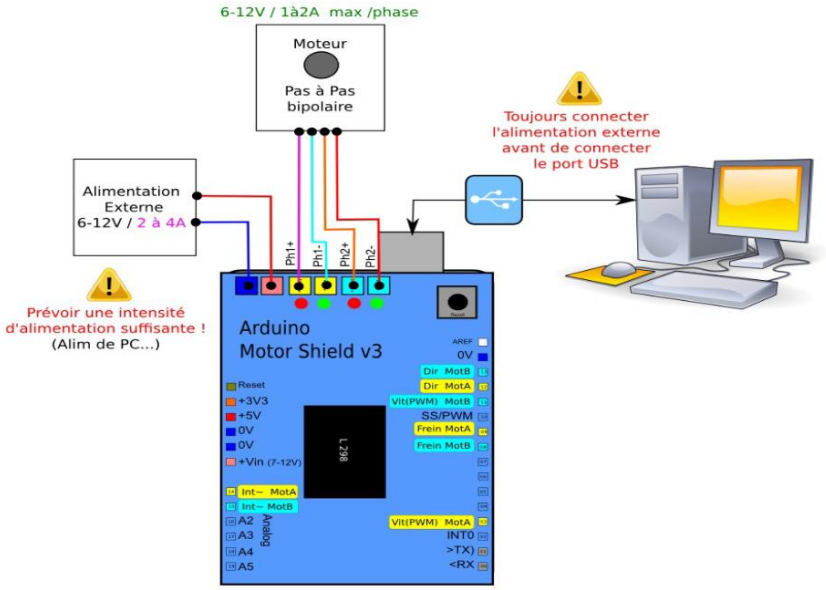
Le couple d'un moteur pas à pas diminue avec la vitesse.

Un simple ohmmètre permettra de repérer les phases.

Maintien de position "Hors-tension":

Hors tension, un moteur pas à pas est en « roue libre » et ne tient pas la position courante. La position reste bloquée sous tension.

Montage Type d'un moteur pas à pas avec Arduino et Arduino Motor Shield V3



Rappel : L'Arduino Motor Shield permet de contrôler 2 moteurs CC ou 1 moteur pas à pas bipolaire

Chaque moteur est contrôlé par une broche de sens et une broche de vitesse PWM
Dispose de LEDs de fonctionnement et de diodes de protection + bouton reset
Intensité maximale de 2A / phase ou moteur (4A en tout)
Dispose d'un capteur analogique d'intensité intégré(1,65V/A) pour chaque phase/moteur
Alimentation entre 7 et 12V / 4000mA

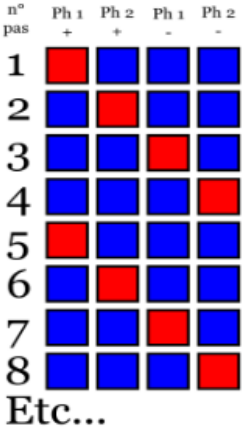
Bornier à vis pour connecter les moteurs et leur alimentation

2 connecteurs TinkerKit pour deux entrées analogiques (en blanc), connectés à A2 et A3.

2 connecteurs TinkerKit pour deux sorties Analog (en orange au milieu), connectés aux sorties PWM sur les broches D5 et D6.

2 connecteurs TinkerKit pour l'interface TWI (en blanc avec 4 broches), un pour l'entrée et l'autre pour la sortie. (Bus I^2C)

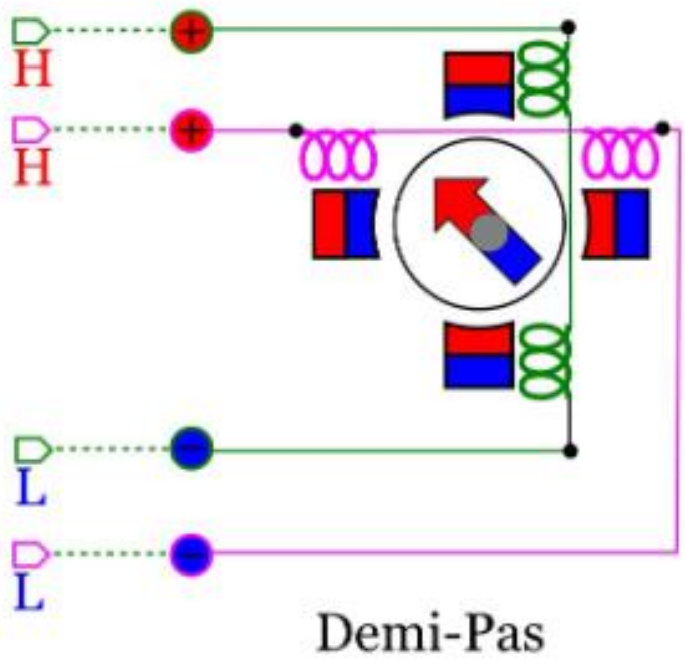
La séquence de base : (Mode Full Step)



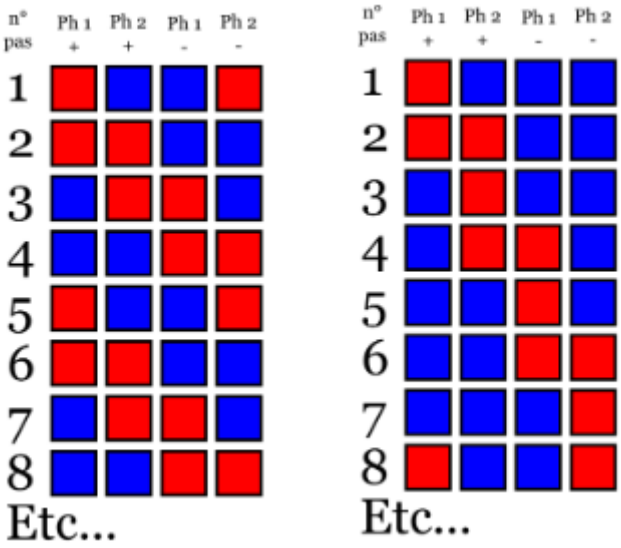
La rotation inverse est obtenue simplement en inversant l'ordre des pas.

Séquence "Half Step" (Demi pas)

Il est possible également de créer des pas intermédiaires en activant simultanément 2 phases, le rotor se positionnant alors de façon intermédiaire



On obtient alors soit la séquence simple mais en se basant sur des « demi-pas », soit la combinaison de la séquence simple et de la séquence « half-step » ce qui double le nombre de pas :



Library ,pour séquence Half step avec Arduino motor shield Rev 3 :

https://www.arduinolibraries.info/libraries/half-stepper

Autres séquences

En combinant astucieusement et de façon complexe l'activation des phases et l'intensité circulant dans les phases, on peut ainsi obtenir des séquences de plus en plus poussées permettant d'utiliser le 1/4 de pas, le 1/8e de pas et même le 1/16e de pas ! (Atelier 07e 2 consacré au mode "micropas")
Ces séquences ne sont pas accessibles à l'aide d'un « double-pont en H » mais en utilisant des circuits spécialisés de contrôle de moteurs pas à pas

---

La librairie Stepper pour le contrôle des moteurs pas à pas

La librairie Stepper est une librairie qui fournit les fonctions de base utiles pour utiliser un moteur pas à pas avec contrôle direct de la séquence.

La librairie Stepper s'intègre dans un programme avec la ligne :
#include <Stepper.h> // inclut la librairie Stepper

Le constructeur de la classe :
Le constructeur de la classe existe sous 2 formes :

Stepper moteurPAP(nombre_pas, broche1, broche2); // déclare un objet Stepper contrôlé par 2 broches

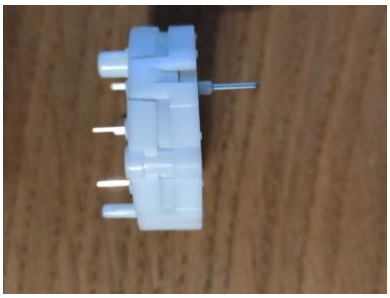
Stepper moteurPAP(nombre_pas, broche1, broche2,broche3,broche4); //
déclare un objet Stepper contrôlé par 4 broches

Les fonctions de la librairie:

La librairie dispose des 2 fonctions suivantes :

setSpeed(vitesse) : fixe la vitesse de rotation en nombre de tours/min
step(nombre_pas) : fixe le nombre de pas dont doit tourner le moteur : un nombre positif fait tourner dans un sens et un nombre négatif fera tourner dans l'autre sens.

Attention : la fonction step() est bloquante : elle durera tout le temps
nécessaire pour exécuter la rotation voulue



Moteurs pas à pas en mode Microstep avec Arduino

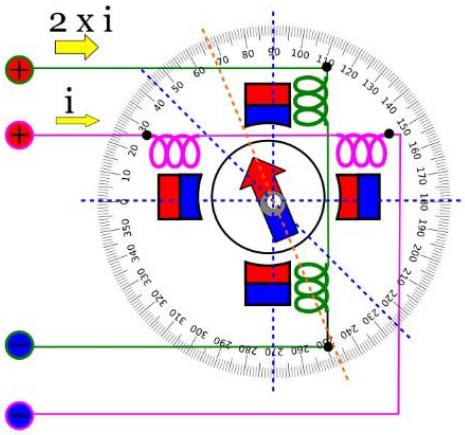
(Atelier 07e2)

Moteur de 200 pas : Avec une interface basée sur un "double-pont en H" classique, il va être possible d'utiliser 200 pas en mode « full step » ou 400 pas en se basant sur une séquence mixte "full+half step"

Dans tout ce qu'on l'a vu jusqu'à présent, l'intensité utilisée pour chaque phase est la même. Dans le mode "micropas", l'idée est de créer un nombre supplémentaire de pas en jouant sur l'intensité circulant dans les phases

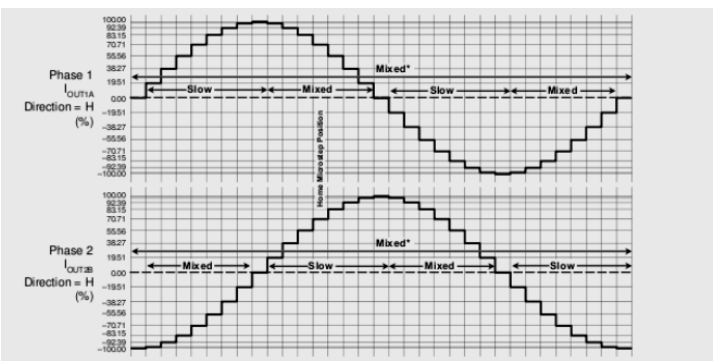
Principe général du micro-pas :

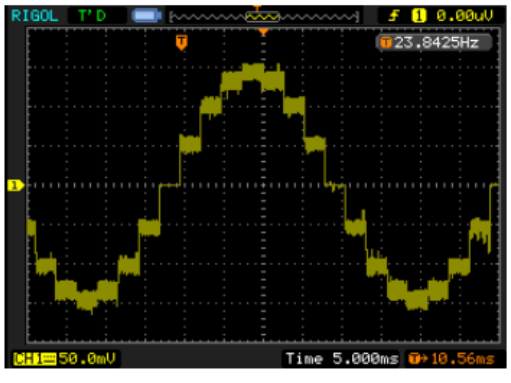
Imaginons qu'au lieu de faire circuler la même intensité dans chaque phase, on soit en mesure de faire circuler une intensité deux fois plus importante dans une phase que dans l'autre : le rotor va se positionner logiquement de façon intermédiaire entre la position de demi-pas et la position du pas. On a donc ainsi créé un pas supplémentaire... : c'est le principe du "micro-pas"



Micropas (Quart de Pas)

Courant dans les phases :





Mode quart de step.

Pour contrôler un moteur pas à pas bipolaire en mode "micro-pas" on va donc utiliser un circuit électronique spécialisé. Paradoxalement, le contrôle "numérique" en sera simplifié, la gestion des pas étant laissée à la charge de l'interface. Du coup, pour contrôler un moteur pas à pas, on utilisera seulement:

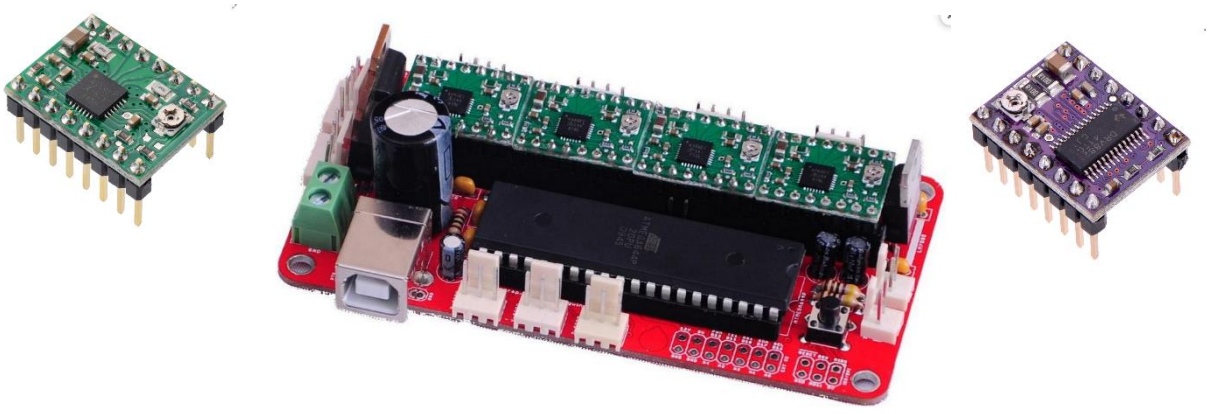
Une broche numérique de sens ON/OFF
Une broche numérique de vitesse des pas qui donnera non pas une impulsion PWM mais une simple « cadence » qui va fixer la vitesse.

Ainsi par exemple, il devient possible d'alimenter avec une tension de 12V un moteur pas à pas bipolaire prévu pour fonctionner en 2,7V si on fixe une intensité de phase maximale compatible avec ce moteur ! Ceci est très souple, notamment avec les moteurs de récupération

Une caractéristique essentielle de ces interfaces est de permettre de régler l'intensité maximale pouvant circuler dans une phase à l'aide d'une résistance variable sur le circuit imprimé.

Exemple d'interfaces de contrôle "micropas" utilisables avec Arduino:

A 4988 Pololu	DRV 8825 Pololu
V min 8 V	V min 8.2 V
V max 35	V max 45 V
CC / Phase 1 A (Sans dissipateur)	CC / Phase 1.5 A (Sans dissipateur)
CC / Phase 2 A (Avec dissipateur)	CC / Phase 2.2 A (Sans dissipateur)
Logique V min 3 V	Logique V min 2.5 V
Logique mas 5.5 V	Logique mas 5.25 V
Microstep ½, ¼, 1/8, 1/16,	Full, ½, ¼, 1/8, 1/16, 1/32

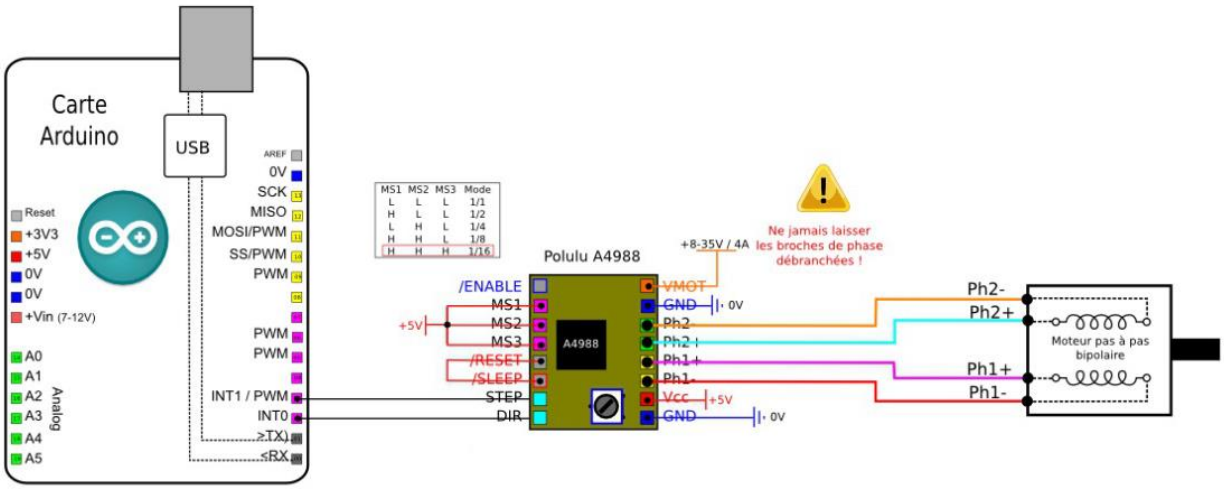


Carte Sanguinolulu avec 4 Drivers pour imprimante 3D

Pratique pour le dépannage

Le réglage de l'intensité maximale se fait avec une résistance variable sur l'interface (Voir les conseils de l'atelier 07e2)

Montage Type pour contrôler un moteur pas à pas avec un Arduino en mode micropas.

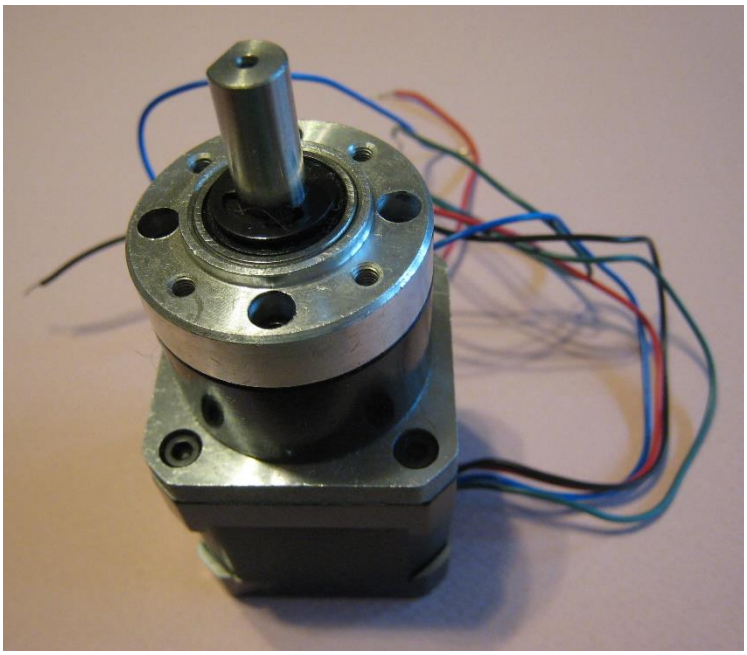


Contrôle de la vitesse du moteur

Pour contrôler la vitesse, il suffit d'appliquer une impulsion de fréquence voulue les fronts montants sur la broche STEP : c'est vite laborieux de le faire par le code lui-même

Une solution simple peut consister à utiliser l'instruction tone() : Cette instruction utilisée pour les sons génère concrètement une impulsion de fréquence voulue et par conséquent peut être utilisée pour générer une fréquence voulue sur une broche, et donc permettra de contrôler la vitesse d'un moteur pas à pas en mode micro-pas dans notre cas.

Exemple de moteur pas à pas avec réducteur:



Type : Nema 17 Bipolaire

Angle par pas: 1,8° (0,35° après réduction)

Tension: 2,8 V

Courant/phase: 1,7 A

Résistance/phase: 1,7 Ohms

Couple de maintien: 18 kg.cm

Réduction: 5,18:1 ➔ Ce qui fait 1036 pas pour l'arbre de sortie soit environ 20.85 minutes d'angle

Connexion: 4 fils

Axe de sortie: Ø8 x 17 mm (avec méplat)

Axe pour encodeur: Ø 3,8 mm x 11,5 mm

Section: 42 x 42 mm

Longueur: 105 mm (axes inclus)

Poids: 450 gr

Communication de l'Arduino avec le PC – Ateliers 8a / 8b / 8c / 8c2

A partir de l'DE, La carte Arduino est (re-)programmable à volonté via le
port série USB du PC : c'est ce qui fait toute la simplicité de son utilisation

IDE = Integrated Development Environment (Environnement de Développement "Intégré" en français)

La carte Arduino est également capable très simplement de communiquer
avec le PC pendant l'exécution d'un programme:

- Pour envoyer des messages vers le PC (chaine texte, valeurs numériques) afin de les visualiser sur l'écran sous forme texte ou même sous forme de graphiques (usage avancé)

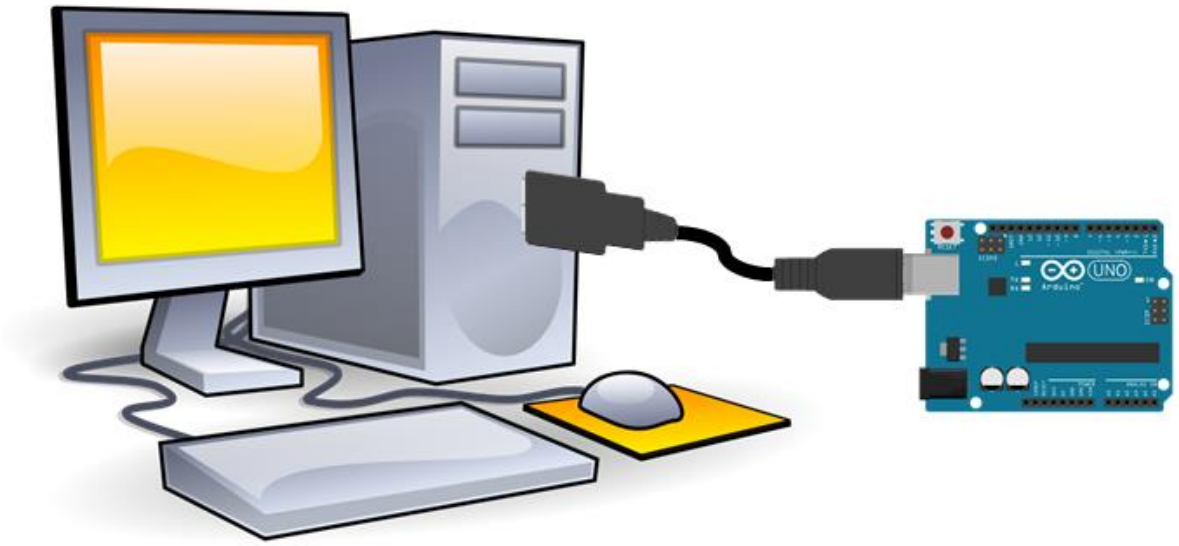
- Pour recevoir des messages depuis le PC (chaine texte, valeurs numériques) ce qui permettra de contrôler la carte Arduino à partir du clavier ou de la souris par exemple (usage avancé)

La possibilité offerte par le langage Arduino d'afficher des messages sur le PC est l'une des grandes forces de ce système : il est ainsi possible de "voir" de l'intérieur comment un programme fonctionne, ce qui est très puissant pour comprendre, mais aussi mettre au point ses programmes.

Au final, la carte Arduino programmée pourra fonctionner de 2 façons:

Soit en autonomie, déconnectée du PC une fois programmée.

Soit en communiquant avec le PC pendant l'exécution du programme



Classe : On appelle "Classe" un regroupement de fonctions

La Classe "Serial" :

Les fonctions en Emission

begin() : fonction d'initialisation de la communication USB
print() : fonction d'affichage d'un message sans saut de ligne
println() : fonction d'affichage d'un message avec saut de ligne

Les fonctions en Réception

available() : renvoie le nombre d'octets présents en réception sur le port Série (Sera donc true si caractère présent et false sinon...)
read() : lit et renvoie le premier octet entrant en réception sur le port série
flush() : vide la file d'attente en réception du port Série

Ces fonctions appartiennent à la classe Serial et nous les utiliserons donc sous la forme : Serial.available() ou Serial.read() ……

Exemple pour l'initialisation de la communication série.

```
void setup() {
Serial.begin(115200); // initialise la communication série
```

Initialise le débit de la communication ici 115200 Bauds (Le Baud est l'unité de mesure du nombre de symboles transmis par seconde)

Le détail de l'utilisation de ces instructions ne sera pas exposé ici : Il convient d'utiliser les ateliers cités au début de ce chapitre pour aller plus loin. Des exemples sont proposés dans ces ateliers.

Traceur Série

Le traceur série est un nouvel outil intégré dans l'interface de l'IDE Arduino, à partir de la version 1.8.9. (Version actuelle 2.0.3)

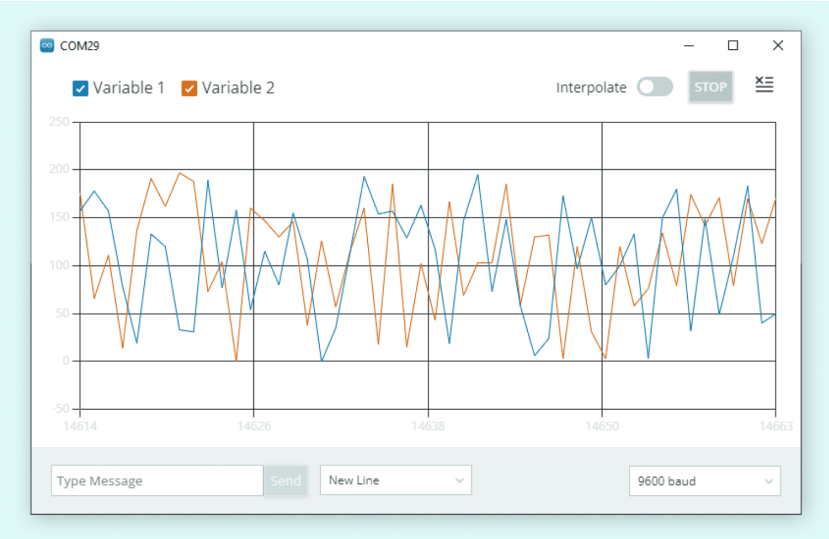
Le traceur série nous permet d'afficher une ou plusieurs courbes en même temps façon oscilloscope.

Le système affecte une couleur différente pour chaque courbe qui est de l'ordre de quatre couleurs. Les courbes supplémentaires se trouvant ensuite dans une même couleur

On peut afficher toutes sortes de valeurs qu'elles soient entières, a virgule flottante.

Le format d'affichage est normalisé en fonction de la courbe à visualiser.

Le système de courbes fonctionne sous le principe appelé FIFO ou la méthode du premier entré premier sorti. Quand la taille est atteinte, chaque nouveau point de la courbe supprime l'ancien point du début de celle-ci, l'affichage est donc dit glissant vers la gauche

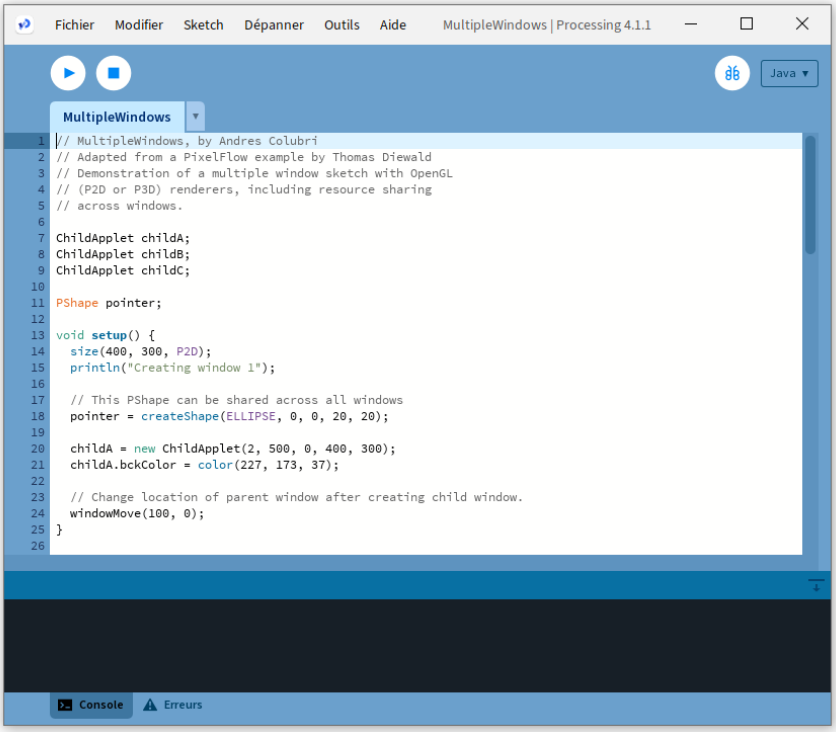


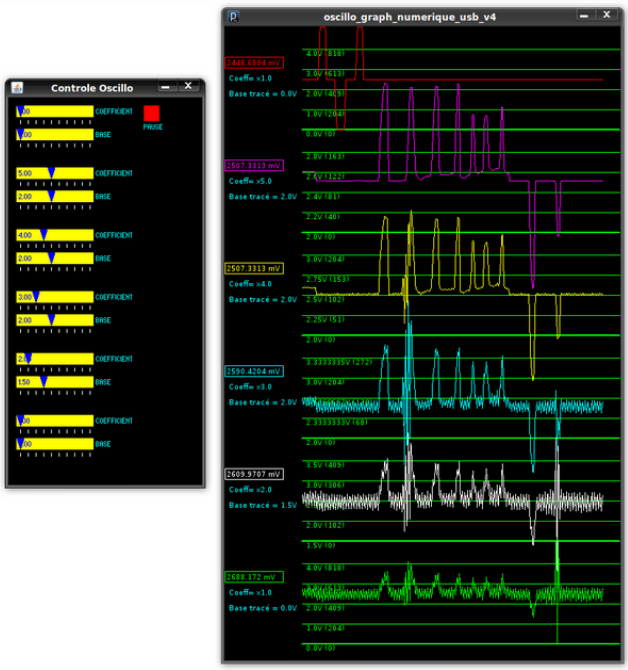
Documentation ici : https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-plotter

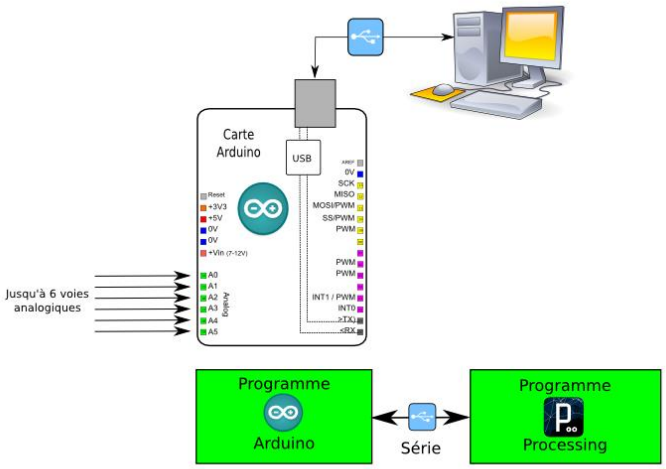
Lorsque j'ai acheté les ateliers Arduino, l'IDE ne comportait pas cette fonction et l'auteur des articles conseillait l'interface "Processing"

Processing est une interface graphique programmable, libre et open source, qui va permettre par programmation de réaliser des dessins, manipuler et modifier des images, de la vidéo, de la 3D, de gérer le clavier ou la souris et même de communiquer avec Arduino par le port Série.

Le langage de programmation s'appelle le langage Processing et ressemble dans ses principes de base au langage Arduino, (Atelier 8b) L'interface Processing (à télécharger sur https://processing.org/) ressemble étrangement à l'interface Arduino : c'est normal, car l'interface Arduino est dérivée l'interface Processing. (On parle de la version 1.8.9 de l'IDE de l'Arduino)







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