L'univers Arduino

Partie I : Aperçu de la plateforme Arduino et ses cartes d'interface (*shields*)

Par f-leb

Date de publication : 29 août 2013

Dernière mise à jour : 26 septembre 2013

TOUT PUBLIC

Véritable mini-ordinateur au succès planétaire, traitant les données provenant de composants et capteurs divers (capteur de température, luminosité, mouvement ou boutons-poussoirs, etc.) et communiquant des ordres pour allumer des lampes ou actionner des moteurs électriques, la carte électronique Arduino permet de créer et prototyper de véritables objets numériques interagissant avec le milieu extérieur.

L'environnement de programmation qui l'accompagne propose un IDE et un langage basé sur les langages C / C++.

La communauté libre du monde Arduino contribue largement à diffuser les ressources permettant la création d'objets numériques à moindre coût et accessibles à toutes personnes motivées ayant même des connaissances modestes dans les domaines de l'informatique et l'électronique.

I - À qui s'adresse ce tutoriel ?

Il s'adresse aux débutants en programmation des microcontrôleurs voulant rejoindre les adeptes du mouvement *Do It Yourself* dans le domaine du *Physical Computing* (1) . Évidemment, en grand écumeur des forums Developpez.net, la programmation en général ne vous effraie pas. Vous aimeriez juste profiter de votre métier, votre passion pour la programmation, pour interagir avec le monde physique, piloter des robots, automatiser la montée/descente de vos volets déroulants en fonction de l'ensoleillement, construire votre propre station météo et déclencher des alarmes selon des seuils de température, etc.

Seulement vos connaissances en physique/électricité/électronique remontent à vos années lycée, une éternité… Quant à souder vous-même des composants électroniques, n'en parlons pas… Évidemment vous serez obligé de vous y mettre si votre projet doit prendre de l'ampleur, mais pas tout de suite.

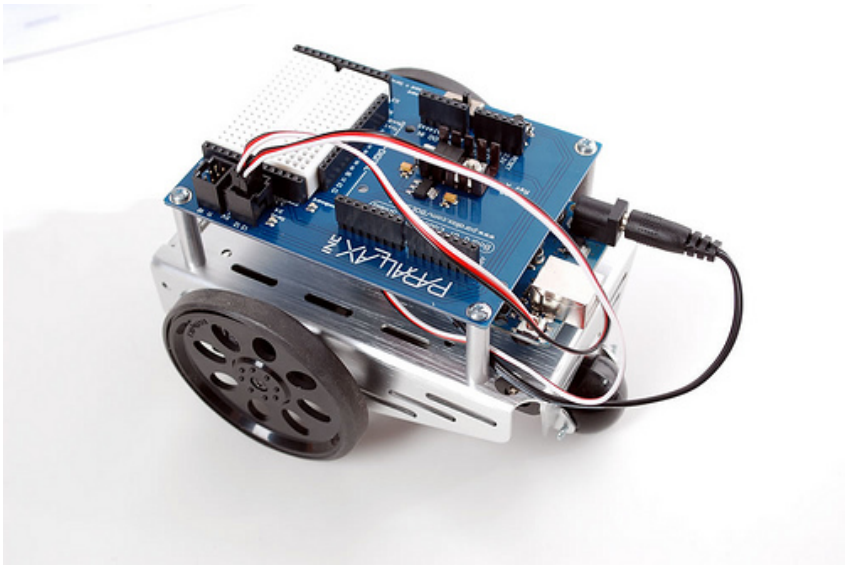
Vous avez déjà entendu parler de la plateforme Arduino, peut-être même acheté votre Arduino Uno (une vingtaine d'euros au moment de la rédaction de ces pages) avec quelques composants de base (plaque de câblage rapide avec quelques fils, LED, résistances…) et programmé votre premier « Hello World » (programme qui consiste à faire clignoter la LED intégrée à la carte via la broche n° 13). Vous avez peut-être même déjà commencé à combler vos lacunes en **électronique de base**, suivi quelques tutoriels pour allumer une, deux puis trois LED voire plus. Classique et indispensable pour débuter…

Ce tutoriel va reprendre un peu tout cela notamment dans cette première partie, mais il propose également d'aller un peu plus loin avec l'utilisation de capteurs évolués, la découverte des cartes d'interface (ou *shields*) et de bibliothèques tierces. Le programme est vaste, mais les ressources proposées vous permettront de vous lancer dans des projets plus ambitieux en vous tenant le plus possible à l'écart du fer à souder.

L'objectif reste le même que celui de la communauté Arduino, découvrir et s'amuser…

II - Rôle de la carte Arduino dans l'objet pluritechnique

Vous souhaitez créer votre premier robot à moindre coût ? De nombreux kits à base d'une carte Arduino sont disponibles sur le Net :

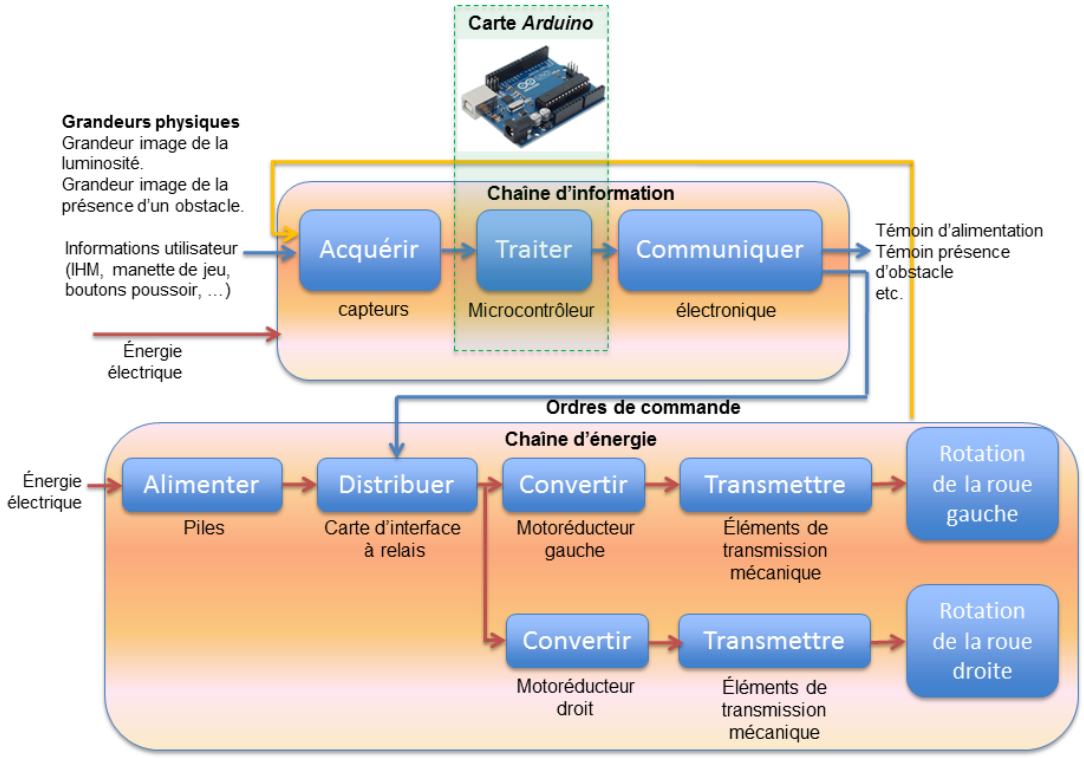


Parallax BOEBot Robot for Arduino Kit

(1) En gros et au sens large, une discipline qui marie l'informatique, la physique et l'électronique, s'intéressant aux systèmes physiques interactifs qui utilisent des logiciels et des matériels s'interfaçant avec des capteurs, des actionneurs électriques, etc.

Ce petit robot roulant à vocation pédagogique possède deux roues indépendantes motorisées et peut être équipé de multiples capteurs pour détecter des obstacles, suivre une ligne au sol, etc.

La structure de cet objet ludique et pluritechnique est résumée dans le diagramme fonctionnel ci-dessous (merci à l'Éducation nationale à qui j'ai emprunté ces magnifiques diagrammes) :



Chaîne d'information - Chaîne d'énergie

La partie inférieure comporte la « chaîne d'énergie » dont le rôle est d'adapter/convertir le flux d'énergie rentrant (de nature électrique) en énergie mécanique utilisable au niveau des deux roues motrices.

La partie supérieure comporte la « chaîne d'information ». Les flèches en bleu constituent donc des flux d'information. Cette partie doit acquérir les informations renvoyées par les capteurs (luminosité pour suivre une ligne noire au sol, ultrason pour détecter les obstacles…), les traiter et communiquer les « ordres de commande » à la chaîne d'énergie, éventuellement informer l'utilisateur (signal sonore ou lumineux par exemple).

Il apparaît clairement que c'est dans la chaîne d'information dévolue principalement au traitement de l'information que notre carte Arduino joue son rôle.

Il apparaît également que la carte Arduino est incapable de transmettre directement « la puissance » aux moteurs électriques. Effectivement chacune de ses entrées/sorties numériques peut absorber ou délivrer un courant électrique maximum de 40 mA sous 5 Volts (sans dépasser les 200 mA pour l'ensemble des entrées/sorties) ce qui est insuffisant étant donné le courant typiquement absorbé par ce genre de moteur électrique.

C'est pour cela que vous devez passer par une « carte d'interface » (ici à base de relais) qui à partir des « ordres de commande » communiquée par l'Arduino (signaux véhiculant de l'information de faible puissance électrique) permet de « distribuer » une puissance adaptée à chaque moteur.

Si vous n'avez pas tout assimilé à cette description, retenez au moins ceci :

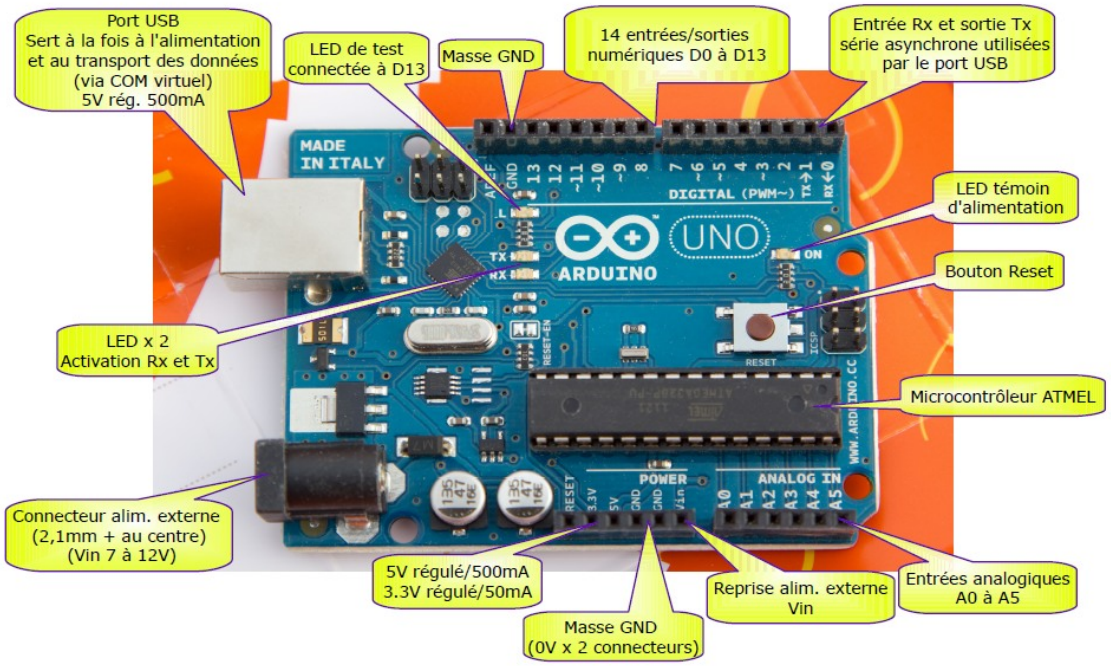
Le rôle de la carte Arduino est de véhiculer de l' « information » (signaux électriques de faible puissance), et non de transmettre directement de la puissance à un actionneur tel un moteur électrique.

Ceux qui ont essayé de transmettre de la puissance sans passer par ces composants d'interface ont seulement réussi à transmettre des signaux… de fumée

III - Description de la carte Arduino Uno

Il existe de nombreux **modèles de cartes Arduino**, la plus populaire étant probablement la **Uno**.

Le document ci-dessous (2) vous donne un aperçu de l'organisation de la carte (dimensions : 65 x 52 mm) dans sa version Uno.



III-A - Le microcontrôleur

Au moment de la rédaction de ces lignes, le cœur de la carte Arduino Uno est un microcontrôleur de la famille AVR, un Atmel Atmega 328P.

Ce microcontrôleur renferme dans un seul composant :

- un processeur 8 bits à architecture RISC ;
- de la mémoire avec des espaces physiques séparés :
  - mémoire Flash (32 Ko) pour les programmes,
  - mémoire vive SRAM (2 Ko) pour les données,
  - mémoire EEPROM (2 Ko) pour les données de sauvegarde ;
- toute la logique d'horloge (16 MHz) ;
- des circuits d'interface et des périphériques d'entrée-sortie permettant au processeur d'accéder au monde extérieur :
  - des Timers/Counters (T/C) 8 et 16 bits,

(2) Source de la photo : **Flickr:Arduino Uno** - Licence **CC Attribution 2.0 Generic**

- génération des signaux PWM,
- des interfaces de communication série (UART, SPI, TWI compatible I2C…),
- un convertisseur Analogique-Numérique (A/D Conv.),
- etc.

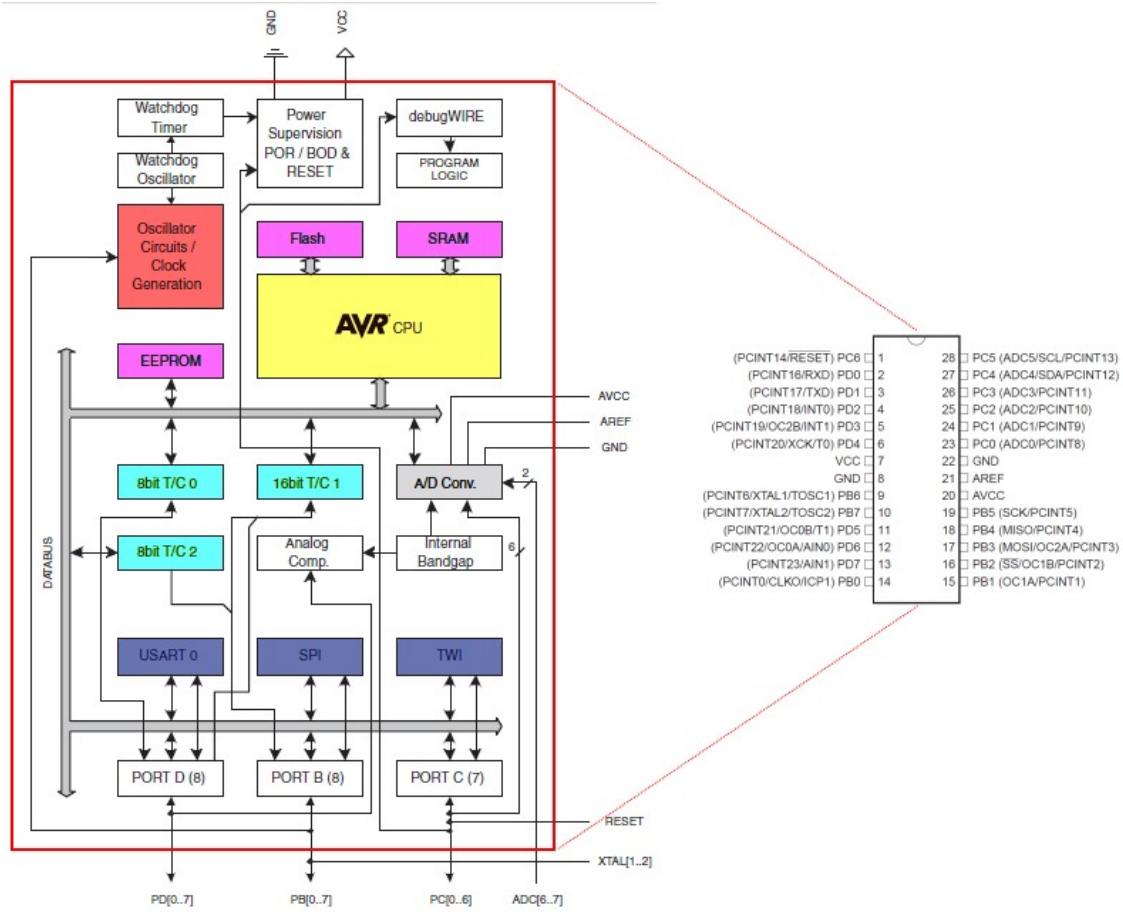
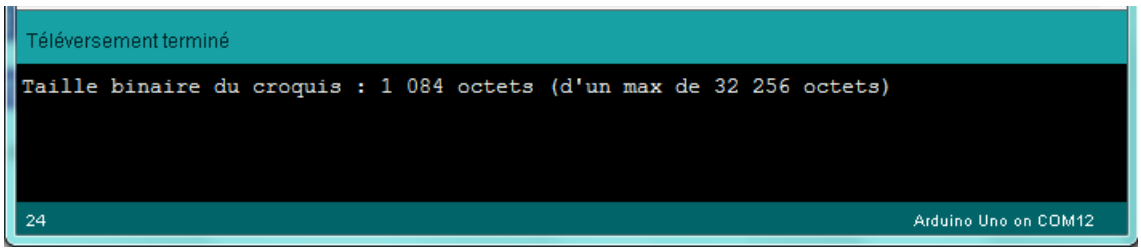


Schéma bloc de description du microcontrôleur AVR

Un processeur à 16 MHz et 32 Ko de mémoire Flash pour stocker vos programmes, ces chiffres peuvent prêter à sourire en comparaison des GHz et des Go de votre ordinateur personnel. Mais avec son format carte de crédit et une consommation inférieure au Watt, la carte Arduino satisfera pourtant vos premières exigences en termes d'embarqué.

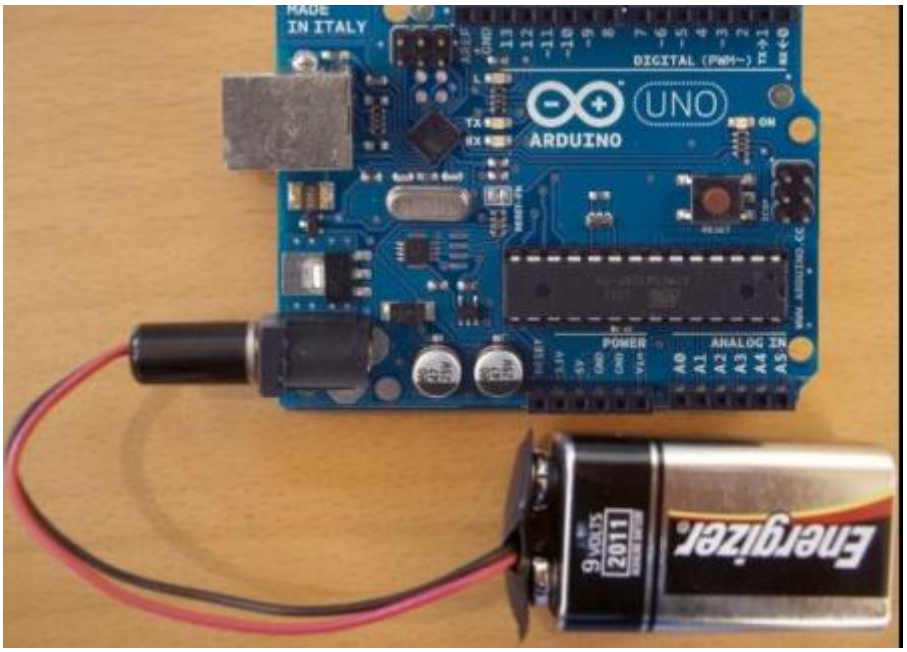


32 Ko disponibles pour vos programmes

III-B - L'alimentation

Elle a lieu sous une tension de 5 Volts. Elle peut provenir soit de la prise USB lorsque la carte est reliée à l'ordinateur, soit d'un bloc secteur externe (tension entre 7 et 12 Volts, 1 Ampère) via la prise *jack* standard.

Comme la carte consomme très peu (0,5 W), elle peut également être alimentée par une simple pile 9 V.



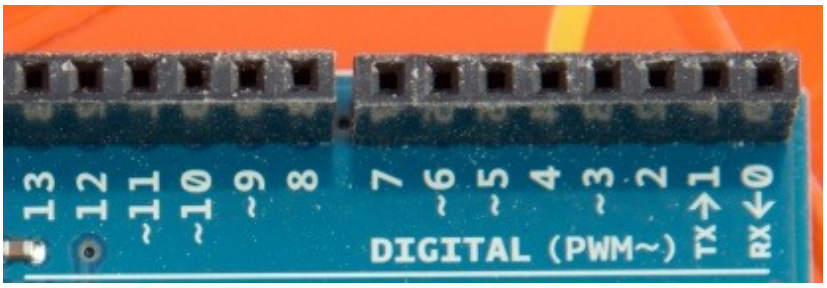
Les régulateurs montés sur la carte permettent de fournir des tensions stabilisées à 3,3 ou 5 Volts pour alimenter des périphériques (capteurs, *shields*…).

⚠ *Les tensions appliquées aux entrées ne doivent pas dépasser la tension d'alimentation sous peine de détruire le circuit.*

III-C - Les entrées/sorties

Ce sont les deux rangées de connecteurs de part et d'autre de la carte qui permettent sa connexion au monde extérieur.

III-C-1 - Les entrées/sorties numériques D0 à D13



Entrées/sorties numériques D0 à D13

Chacun des connecteurs D0 à D13 peut être configuré dynamiquement par programmation en entrée ou en sortie.

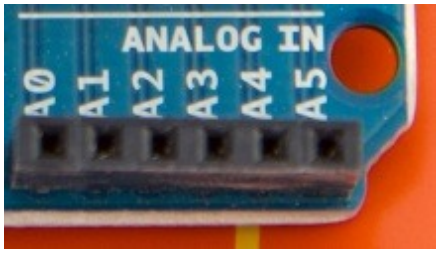
Les signaux véhiculés par ces connecteurs sont des signaux logiques compatibles TTL, c'est-à-dire qu'ils ne peuvent prendre que deux états HAUT (5 Volts) ou BAS (0 Volt).

En pratique, les connecteurs D0 et D1 réservés pour la liaison série asynchrone (port COM virtuel via le câble USB) ne sont pas exploités pour d'autres utilisations.

À noter que chacun des connecteurs ne peut fournir ou absorber un courant supérieur à 40 mA environ (200 mA pour l'ensemble des connecteurs).

Certains connecteurs peuvent être spécialisés comme sorties **PWM** (repérées par un ~) mais nous sortons ici du cadre de ce tutoriel.

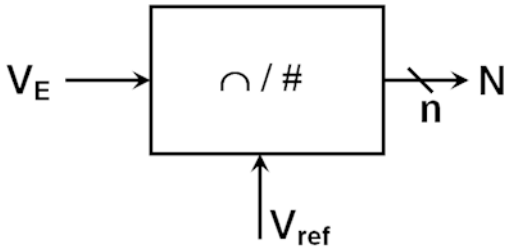
III-C-2 - Les entrées analogiques A0 à A5



Entrées analogiques A0 à A5

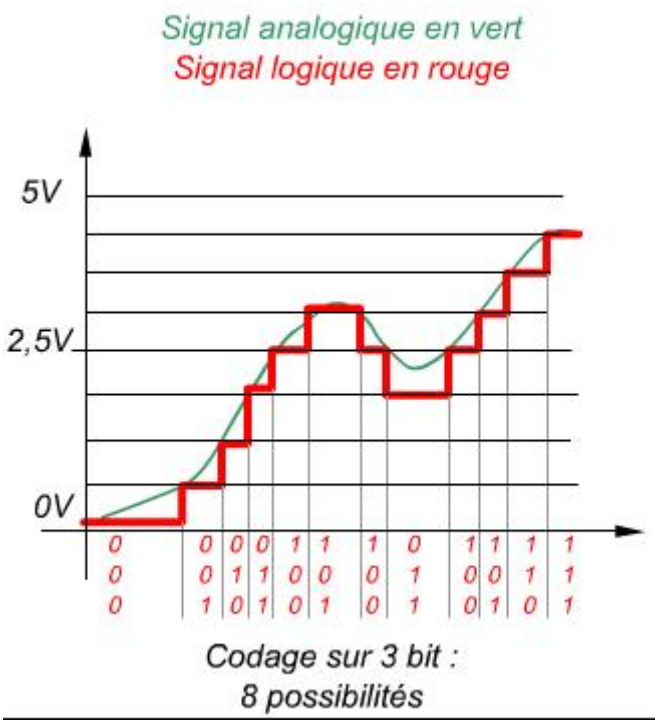
Par défaut et contrairement aux entrées/sorties numériques qui ne peuvent prendre que deux états HAUT et BAS, ces six entrées peuvent admettre toute tension analogique comprise entre 0 et 5 Volts.

Pour pouvoir être traitées par le microcontrôleur, ces entrées analogiques sont prises en charge par un CAN (Convertisseur Analogique Numérique ou ADC pour Analog Digital Converter) dont le rôle est de convertir l'échantillon de tension V_E en une grandeur numérique binaire sur n bits.



Convertisseur Analogique Numérique

Le principe de la conversion Analogique-Numérique est représenté ci-dessous (avec n=3 bits et la tension de référence Vref=5 Volts) :



CAN 3 bits - Auteur : Philippe Avi

Le convertisseur de la carte Arduino Uno possède une résolution de 10 bits, soit 2^{10} = 1024 possibilités de 0 à 1023.

Ainsi, pour n=10 bits et la tension de référence par défaut Vref=5 Volts, si la tension analogique d'entrée échantillonnée est V_E=3,8 Volts , la grandeur numérique N (ici en entier décimal) en sortie du convertisseur peut être calculée grâce aux relations :

quantum q = $Vref/2^n = 5 / 2^{10} = 5 / 1024$

N = V_E / q = 3,8 x 1024 / 5 soit N = 778

Il y aurait encore bien des choses à dire pour avoir une description plus complète de la carte et nous sommes resté sur l'essentiel.

La plateforme Arduino étant open source, une description plus complète et les schémas de la carte sont donc librement disponibles sur **le site officiel**.

IV - L'environnement de développement

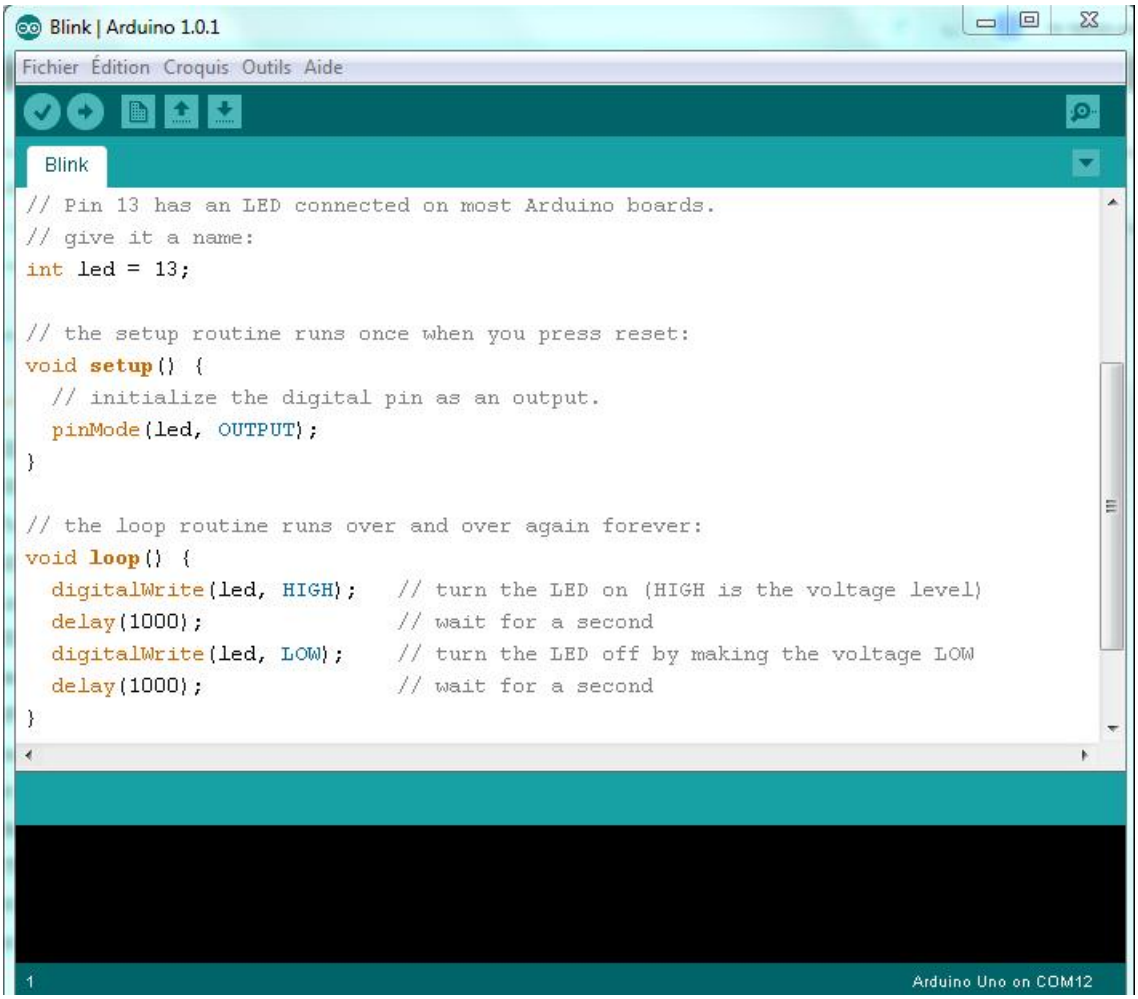
IV-A - Installation

Pour l'installation (Windows, Mac OS X et Linux), référez-vous aux indications pas à pas du site officiel : **Getting Started with Arduino**.

Une version en français est également disponible : **Téléchargement et installation du logiciel Arduino**.

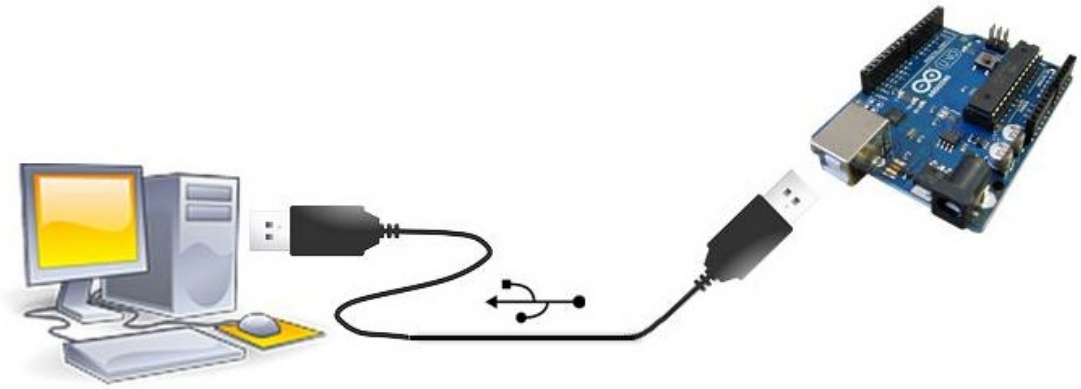
IV-B - Premier pas dans l'IDE

Voici à quoi ressemble l'IDE (Windows), ici avec le programme de démonstration **Blink** :

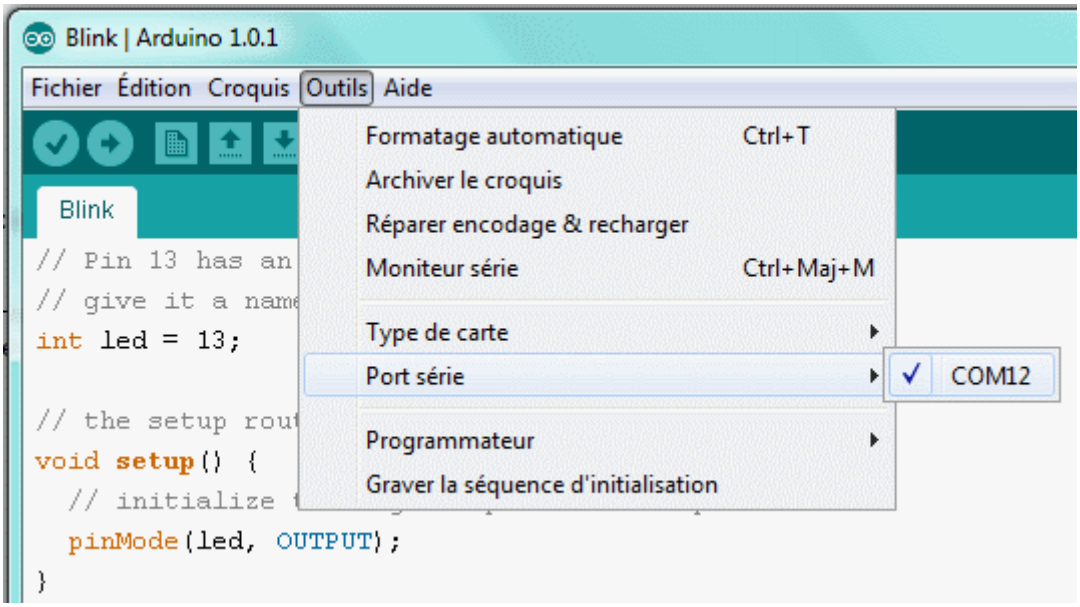


Ce premier programme se contentera de faire clignoter la LED jaune témoin qui se trouve près du connecteur D13 de la carte Arduino Uno.

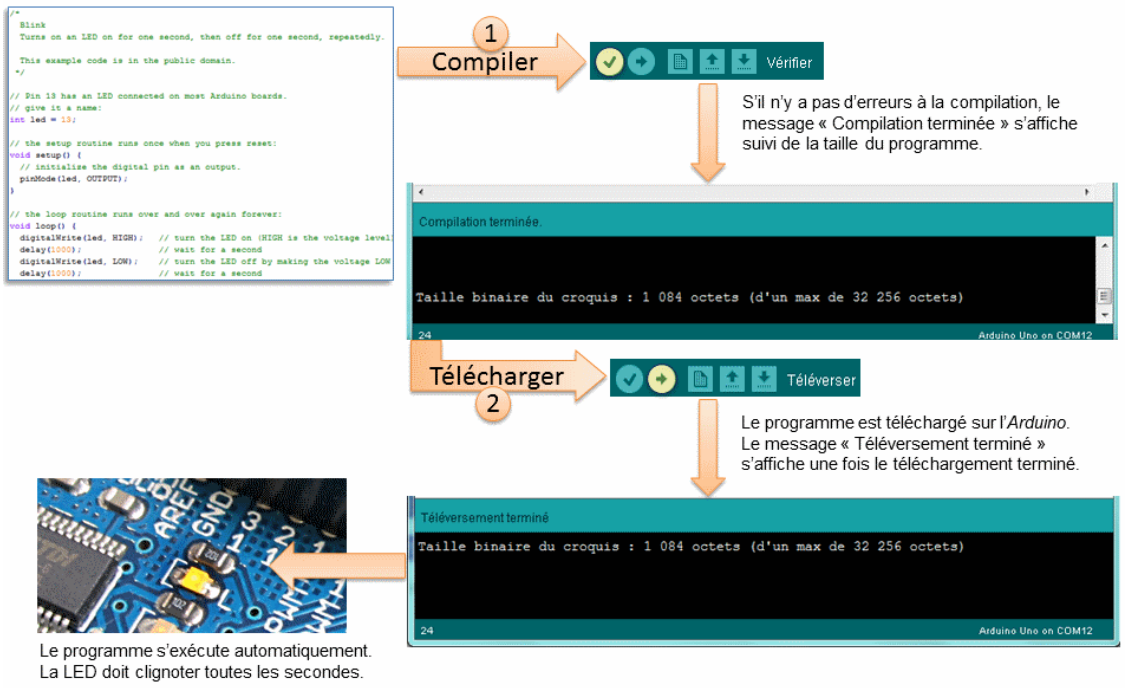
Avant de télécharger le programme dans la carte, il faut relier l'ordinateur et la carte via le câble USB :



Une fois le port activé et reconnu sur l'ordinateur, il faut éventuellement désigner le bon dans l'interface :



Finalement, le processus de rédaction du programme jusqu'à son téléchargement dans la carte peut être résumé grâce au schéma suivant :

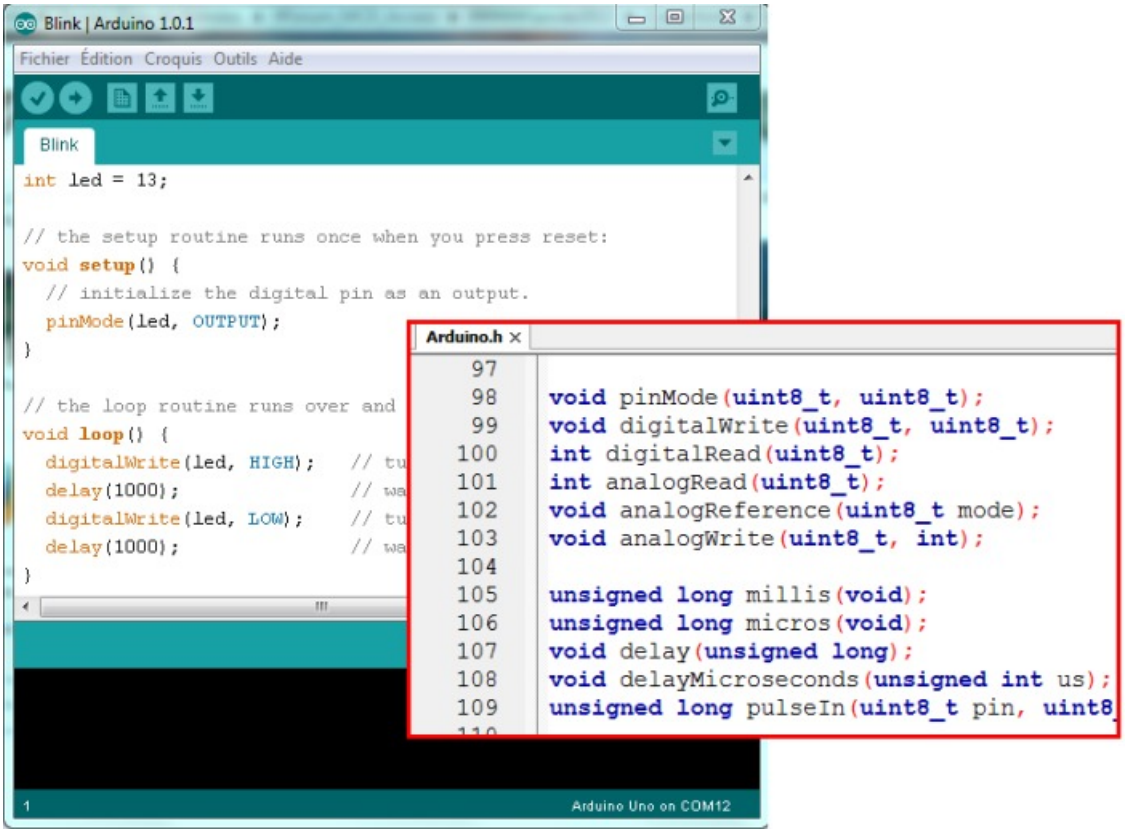


IV-C - Le langage de programmation

Sous Windows par exemple, lorsque vous lancez la compilation dans l'IDE, ce sont les outils de la suite **WinAVR** avec son compilateur GCC qui prennent le relais.

En fait, vous programmez dans un langage propre à Arduino dont la structure s'apparente aux langages C/C++. Mais lorsque vous évoquez une fonction Arduino, non standard C/C++, et pourtant reconnue et coloriée comme un mot-clé dans l'éditeur, vous faites appel en toute transparence à une ou plusieurs bibliothèques rédigées en C ou C++ qui seront incluses à la compilation.

En enveloppant le langage C/C++ de cette manière, les concepteurs de l'IDE ont pu simplifier sa syntaxe et l'adapter aux possibilités de la carte. De nombreuses fonctionnalités de haut niveau sont ainsi proposées à l'utilisateur novice qui n'a plus à se soucier de la logique interne du microcontrôleur.



Dans les coulisses de l'IDE Arduino, un compilateur AVR GCC.

Revenons au programme de démonstration **Blink** :

Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

La structure d'un programme Arduino, rédigé dans l'éditeur, ressemble à ceci et doit toujours comporter les fonctions setup() et loop() :

```
// Définition des constantes, variables globales
// directives de compilation define, include, etc.

void setup() {
  // initialisation des ressources de la carte,
  // configuration des entrées/sorties,
  // définition de la vitesse de fonctionnement du port série, etc.
  // setup() n'est exécuté qu'une seule fois.
}

void loop() {
  // les instructions contenues ici sont exécutées indéfiniment en boucle
  // Seule une coupure de l'alimentation de la carte ou un appui sur le bouton Reset
  // permet de quitter le programme.
}
```

Nous n'allons pas ici décrire toutes les instructions du langage Arduino, vous trouverez les références et la syntaxe sur le site officiel :

- **Language Reference**
- **Référence étendue du langage Arduino**

Nous insistons tout de même sur les fonctions implantées pour la gestion des entrées/sorties de la carte :

Fonction	Description (partielle)
pinMode()	Configuration des broches numériques en entrée ou en sortie.
digitalWrite()	Pour une broche numérique configurée en sortie, cette instruction permet de mettre son niveau logique à HAUT ou BAS.
digitalRead()	Lecture du niveau logique (HAUT ou BAS) d'une broche numérique configurée en entrée.
analogWrite()	La carte ne possède pas de Convertisseur Numérique Analogique. Il n'est donc théoriquement pas possible d'obtenir une tension continue qui soit l'image d'une valeur numérique (pas de sortie analogique). Il est par contre possible d'en reproduire le comportement en générant un signal rectangulaire (**PWM ou MLI**) de rapport cyclique réglable. Beaucoup de composants (LED, moteur électrique à courant continu) agissent comme des filtres passe-bas et fonctionnent comme s'ils étaient alimentés sous une tension continue égale à la valeur moyenne du signal.
analogRead()	Lecture de la valeur de la tension présente sur une entrée analogique (A0 à A5 sur la Uno). La fonction retourne la valeur issue de la Conversion Analogique Numérique, soit une valeur comprise entre 0 et 1023 (convertisseur 10 bits).

Ainsi, dans la boucle infinie loop() :

```
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
```

```
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

La broche n° 13, reliée à la LED intégrée à la carte, passe alternativement du niveau logique HAUT (HIGH) au niveau logique BAS (LOW) après un délai d'une seconde, ce qui la fait clignoter.

Comme vous le voyez à travers cet exemple, les concepteurs du langage ont prévu d'interfacer facilement vos programmes avec le monde extérieur.

V - Les cartes d'interface ou Shield

Une carte Arduino seule, même si vous disposez de quelques LED, résistances, boutons poussoir, etc. n'est pas d'une grande utilité et reste dans ce cas limitée à des fins d'apprentissage. Si vous voulez aller plus loin et piloter les moteurs de votre robot, vous devrez passer par une interface dédiée (des composants sur une carte électronique).

Une fois de plus, sous la poussée de la communauté Arduino, de nombreux fabricants proposent une pléthore de cartes d'interface capables de couvrir la majorité des besoins (capteurs, relais de puissance, commande de moteurs, Internet, affichage sur matrice LED ou écran LCD, communication Wifi…).

V-A - Qu'est-ce qu'un shield ?

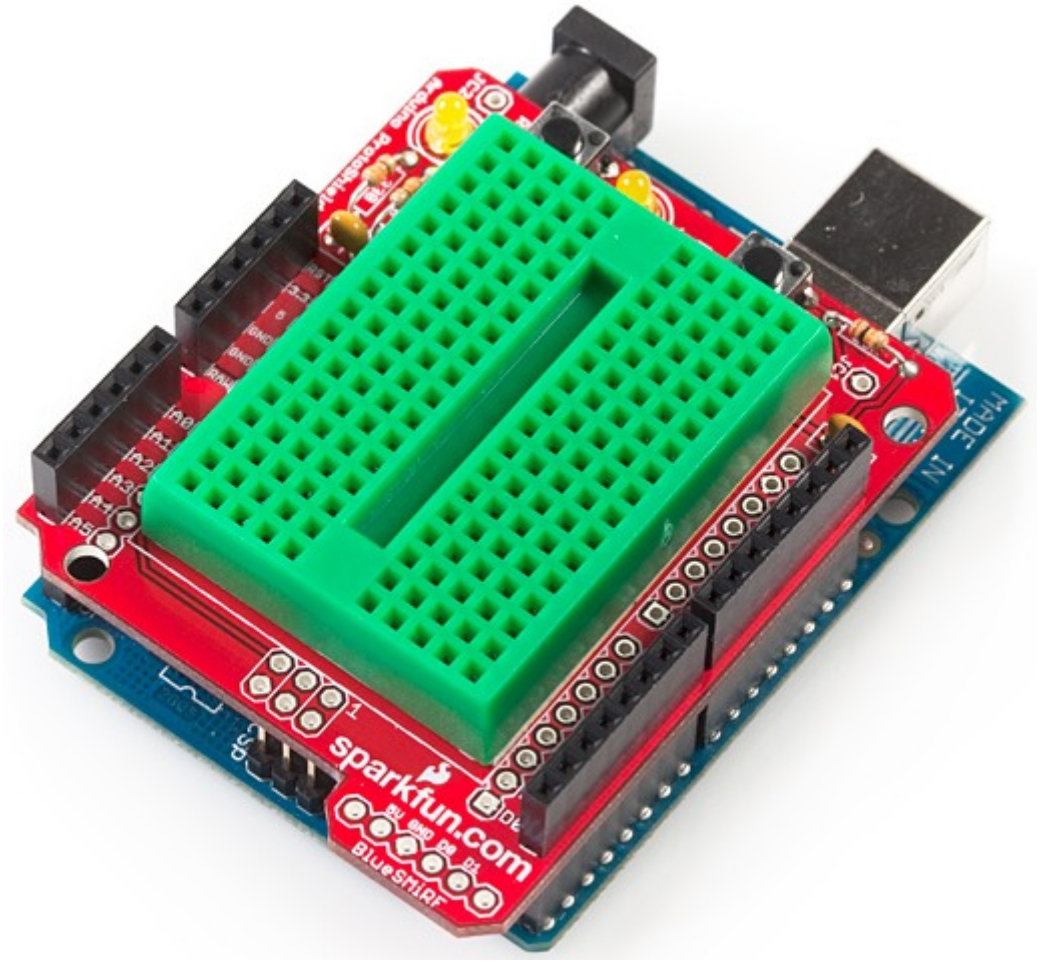
Un *shield* est une carte d'interface spécialement dédiée à l'Arduino. Ces cartes ont des dimensions sensiblement voisines de la carte Arduino et peuvent s'enficher directement sur les connecteurs de celle-ci.

Il est bien sûr impossible de recenser entièrement ici les centaines de *shields* existant sur le marché (dont quelques-uns référencés sur le **site officiel**). Vous trouverez une liste de fabricants et/ou distributeurs de *shields* pour Arduino dans la sitographie en fin de tutoriel.

On peut tout de même en signaler quelques-uns bien pratiques et donner une idée [(3)] .

(3) Les images des *shields* suivantes proviennent du site **https://www.sparkfun.com** et sont sous licence **CC BY-NC-SA 3.0**

V-A-1 - Pour le prototypage

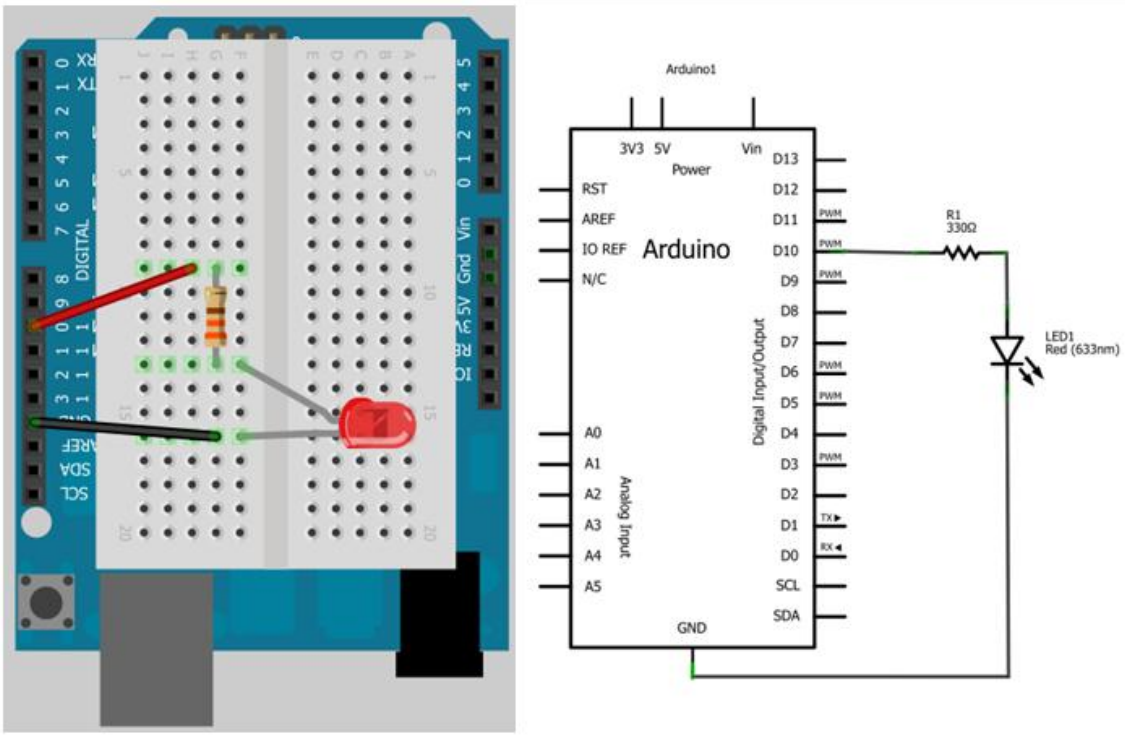


Sparkfun ProtoShield

Une fois enfichée sur la carte A*rduino*, ce *shield* reprend tels quels les connecteurs de part et d'autre de la carte. On peut alors coller une plaque de câblage rapide (*breadboard* en anglais) dans la partie centrale (en plastique vert moulé sur la photo).

Cette plaque permet de câbler de nombreux composants sans faire de soudure et garder le montage entièrement démontable.

Le montage sur plaque d'essai ci-dessous, dessiné avec le logiciel gratuit **Fritzing**, montre une LED branchée sur la broche D10 de l'Arduino avec sa résistance en série.



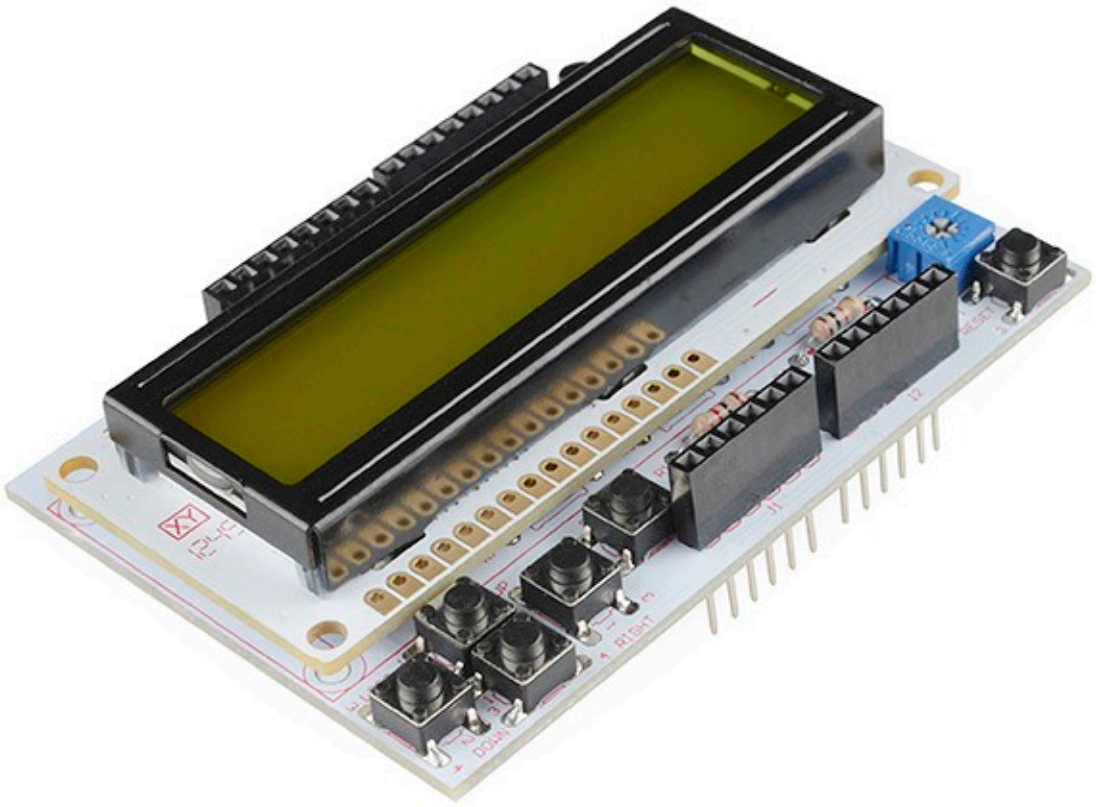
Montage électronique sur plaque de câblage rapide

Il ne reste plus qu'à reprendre le programme de démonstration **Blink** au niveau de la ligne :

```
int led=10;
```

et le tour est joué pour faire clignoter la LED rouge.

V-A-2 - Pour l'affichage

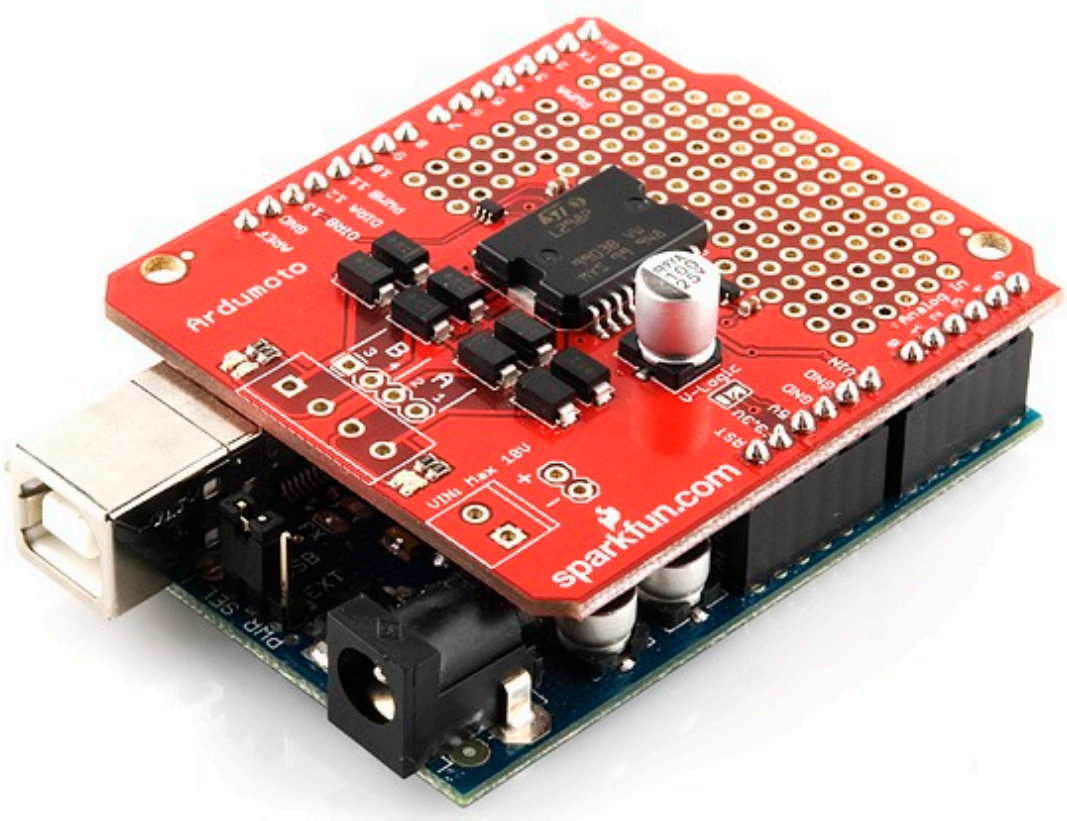


Sparkfun Button LCD Shield

Ce *shield*, en plus de son écran LCD (deux lignes de 16 caractères), comporte cinq boutons-poussoirs programmables.

La bibliothèque **LiquidCrystal** qui permet de piloter l'affichage est disponible par défaut dans l'IDE Arduino.

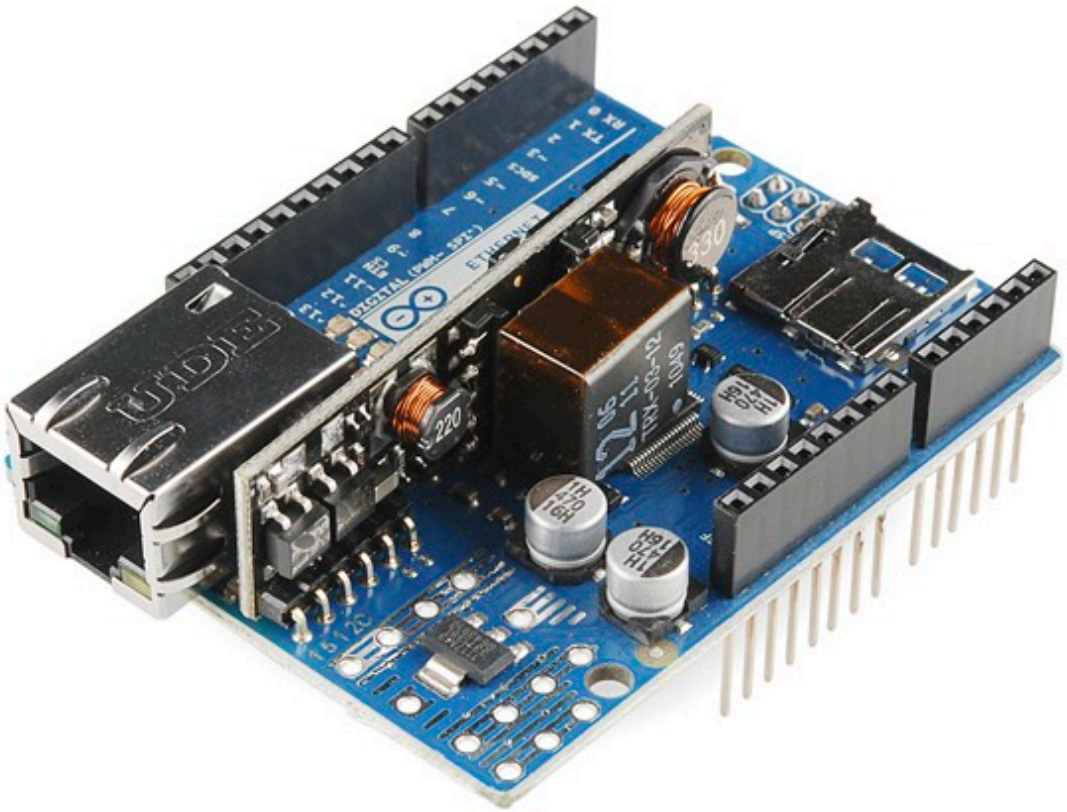
V-A-3 - Pour la commande des moteurs



Sparkfun Motor Driver Shield

Ce *shield* permet de piloter jusqu'à deux moteurs électriques simultanément (idéal pour un petit robot roulant à deux roues motorisées indépendantes). Avec deux connecteurs numériques utilisés par moteur seulement, vous contrôlez le sens et la vitesse de rotation du moteur.

V-A-4 - Pour communiquer

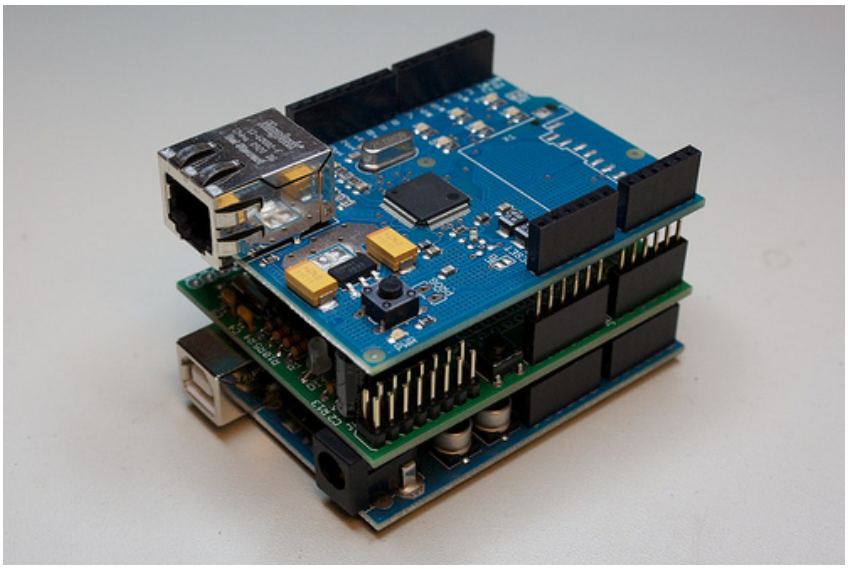


Sparkfun Ethernet Shield

Si vous souhaitez rendre disponible l'état d'un capteur de température sur un réseau local Ethernet ou sur Internet, ce *shield* évolué avec une bibliothèque très complète et de nombreux exemples de programmes saura transformer votre Arduino en véritable serveur web connecté sur l'extérieur.

V-B - Comment bien choisir son shield

Certains *shields* sont mieux pensés que d'autres et tout peut se compliquer si vous devez empiler les *shields* les uns sur les autres, car un seul ne suffit pas.



Empilement de shields

En voulant combiner les caractéristiques de plusieurs *shields*, vous risquez de faire rentrer certaines de leurs fonctionnalités en conflit matériel et/ou logiciel. Telle carte ne pourra pas s'empiler sur telle autre, car les broches de la carte du dessus sont trop courtes pour s'enficher correctement sur la carte du dessous à cause d'un composant trop épais. Telle bibliothèque logicielle dédiée à un *shield* rentre en conflit avec celle d'un autre *shield.* Ou encore plus rageant, les connecteurs des différents *shields* partagent le ou les mêmes connecteurs de la carte Arduino.

Et je passe sur les problèmes d'alimentation, d'interférences radio/électriques, etc.

Disons que tout se passe bien si vous n'utilisez qu'un seul *shield*. Dès que vous commencez à vouloir les empiler, vous devez potasser la documentation constructeur (les *data sheets*) afin de détecter les éventuels conflits.

Les fabricants proposent en principe sur leur site les documents nécessaires à la mise en œuvre : schéma électrique, *data sheet*, exemple d'utilisation, etc.

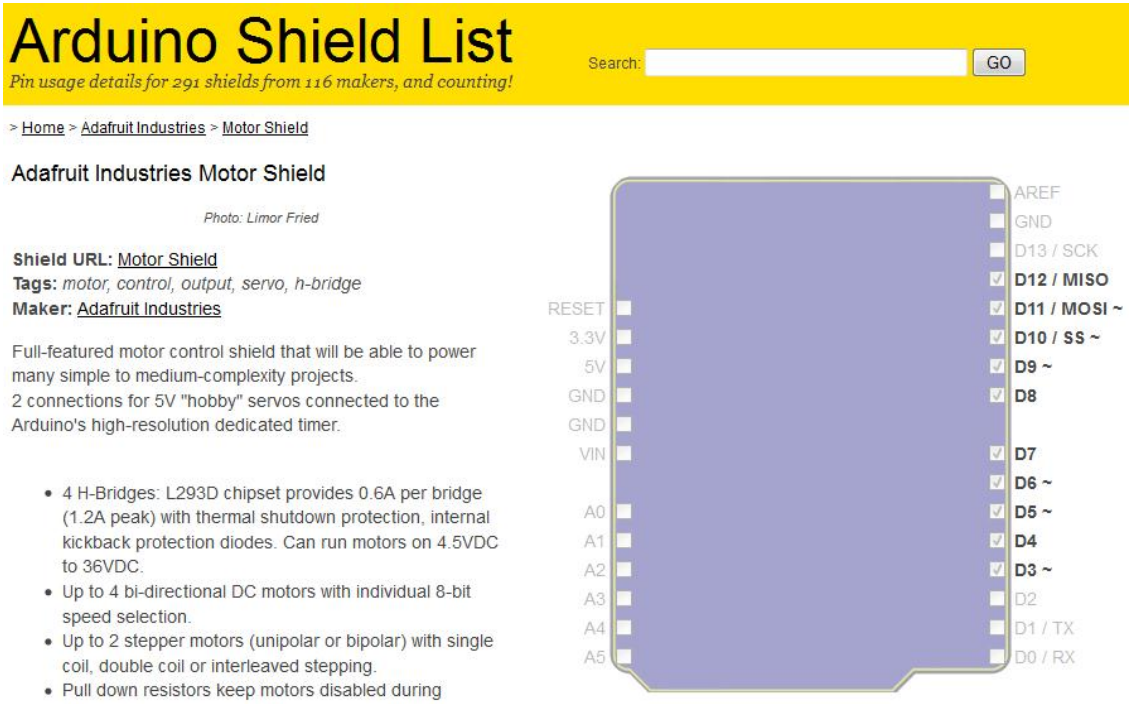
Documents:

- Schematic
- Eagle Files
- L298 H Bridge data sheet
- Example Arduino sketch
- Assembly Guide
- Quickstart Guide

d'après https://www.sparkfun.com

Le site Internet **http://shieldlist.org** vous propose un moteur de recherche intégré avec des centaines de références pour choisir vos *shields*.

Pour chaque *shield* référencé, vous trouverez les caractéristiques principales ainsi qu'un schéma d'allocation des broches bien utile.



Ce shield monopolise les broches D3 à D12

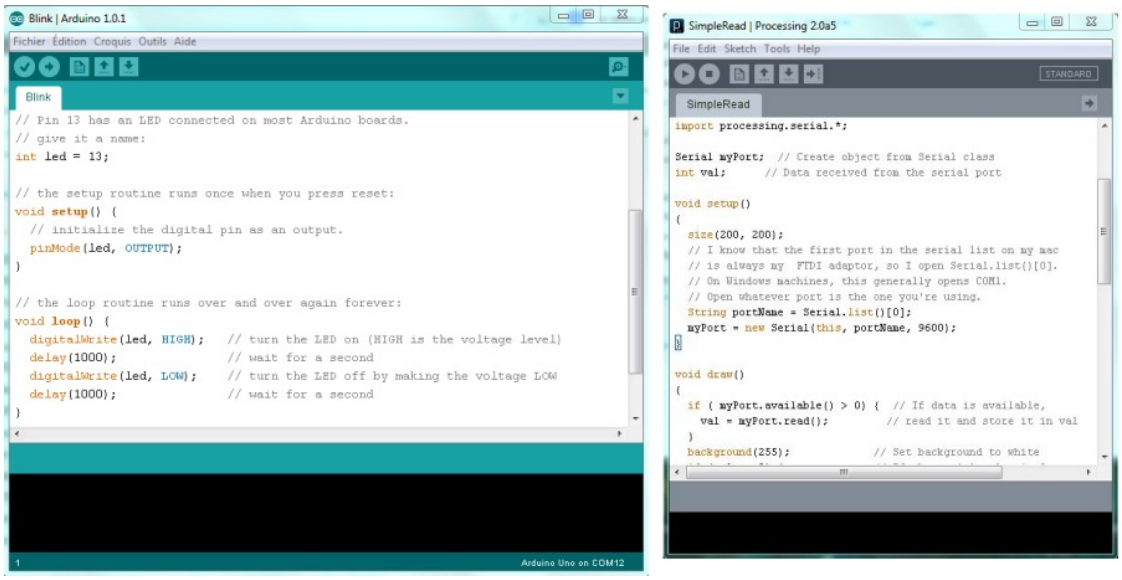
À noter que si l'allocation des broches du *shield* rend le nombre de connecteurs disponibles insuffisant pour d'autres utilisations, vous devrez probablement passer au modèle supérieur comme l'**Arduino Mega** qui dispose de connecteurs supplémentaires et qui peut parfaitement recevoir les *shields* conçus initialement pour la Uno.

VI - Processing

Processing est un environnement de développement et un langage de programmation basé sur le langage Java adapté à la création plastique et graphique interactive.

Pourquoi on vous parle de ça maintenant ?

Il nous suffit d'observer l'IDE Processing sur la copie d'écran ci-dessous. Cela ne vous rappelle rien ?



IDE Arduino et Processing

La philosophie de Processing est aussi la facilité. Outre ses dispositions pour le graphisme et le multimédia, sa bibliothèque **Serial** permet la communication sérielle entre l'Arduino et l'ordinateur hôte. L'alliance parfaite…

Voici une vidéo de démonstration qui montre l'utilisation combinée d'une carte Arduino Uno reliée par un câble USB à un PC muni du logiciel Processing.

Cliquer sur ce lien pour lancer l'animation

Un module **boussole** électronique est connecté à la carte Arduino. Le programme implanté dans le microcontrôleur se contente de récupérer les données du module (angle de rotation par rapport au nord) et les transmettre périodiquement par la liaison série établie via le câble USB (port COM virtuel).

Le programme hôte Processing du PC récupère les données numériques pour animer l'aiguille de la boussole à l'écran.

Arduino et Processing, les deux font la paire… Si vous maîtrisez l'un, l'autre vous sera déjà familier.

VII - Conclusion

Comme vous avez pu le constater, ce qui prime dans Arduino, c'est sa simplicité qui permet de mettre en œuvre de nombreux objets numériques à moindre coût sans être un spécialiste du fer à souder ou de la programmation des microcontrôleurs. C'est cette qualité-là qui donne à Arduino le succès planétaire qu'on lui connaît.

Vous êtes maintenant paré à réaliser votre premier prototype dans la deuxième partie de ce tutoriel :

Partie II : Réalisation d'un prototype à base d'Arduino

VIII - Remerciements

Je remercie vivement **Bktero** et **deletme** pour leur relecture technique et leurs conseils avisés.

Merci également à Maxime et Benjamin pour leurs travaux sur la boussole électronique et qui sont à l'origine de la vidéo.

Je remercie également l'ami **ClaudeLELOUP** pour sa relecture orthographique.

IX - Bibliographie et sitographie

IX-A - En langue française

- **http://fr.flossmanuals.net/arduino/** : un manuel francophone à licence C.C très complet, qui peut être commandé en version imprimée, exporté en PDF ou ePUB.
- **http://www.mon-club-elec.fr/** : le site de Xavier HINAULT, très complet également avec des traductions en français du site officiel.
- **Atelier Arduino - Initiation à la mise en œuvre matérielle et logicielle de l'Arduino** : un livret un peu ancien, mais très bien fait pour débuter vos premiers bidouillages.
- **Arduino - Maîtrisez sa programmation et ses cartes d'interface (shields)** par Christian Tavernier.
- **Processing - Le manuel** : le manuel du logiciel de création multimédia, un complément indispensable pour interfacer vos applications Arduino.

IX-B - En langue anglaise

- **http://www.arduino.cc/** : **le site officiel**.

- **http://tronixstuff.wordpress.com/tutorials/** : le blog *tronixstuff* avec plus de 50 tutoriels pour Arduino.
- **http://shieldlist.org/** : un site référençant des centaines de *shields* pour Arduino.
- **EathShine Design - Arduino Starters Kit Manual** : un autre manuel pour débuter.
- **Adafruit learning system** : une série de tutoriels pour débuter.
- **Arduino Projects to save the World** : on peut aussi le **télécharger** gratuitement.
- **http://www.arduinoevilgenius.com/** : *30 Arduino Projects for the Evil Genius*, existe aussi gratuitement en version PDF.

IX-C - Fabricants et/ou distributeurs de shields pour Arduino

- **Arduino Store**
- **Sparkfun**
- **Adafruit**
- **DFRobot**
- **Lextronic**
- …