ARDUINO

1. Qu'est-ce qu'Arduino?

Le système Arduino est un outil pour fabriquer de petits ordinateurs qui peuvent capter et contrôler davantage de choses du monde matériel que votre ordinateur de bureau. C'est une plateforme open-source d'électronique programmée qui est basée sur une simple carte à microcontrôleur (de la famille AVR), et un logiciel, véritable environnement de développement intégré, pour écrire, compiler et transférer le programme vers la carte à microcontrôleur.

Arduino peut être utilisé pour développer des objets interactifs, pouvant recevoir des entrées d'une grande variété d'interrupteurs ou de capteurs, et pouvant contrôler une grande variété de lumières, moteurs ou toutes autres sorties matérielles. Les projets Arduino peuvent être autonomes, ou bien ils peuvent communiquer avec des logiciels tournant sur votre ordinateur (tels que Flash, Processing ou MaxMSP). Les cartes électroniques peuvent être fabriquées manuellement ou bien être achetées pré-assemblées; le logiciel de développement open-source peut être téléchargé gratuitement.

2. Pourquoi Arduino ?

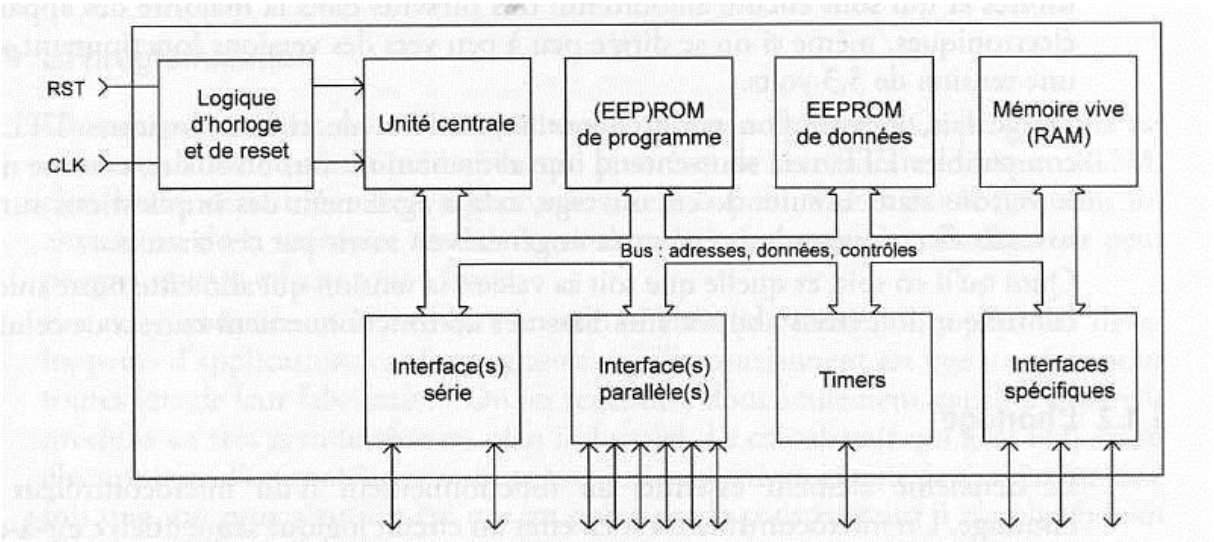
Il y a de nombreux microcontrôleurs et de nombreuses plateformes basées sur des microcontrôleurs disponibles pour l'électronique programmée. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, et beaucoup d'autres qui offrent des fonctionnalités comparables. Tous ces outils prennent en charge les détails compliqués de la programmation des microcontôleurs et les intègrent dans une présentation facile à utiliser. De la même façon, le système Arduino simplifie la façon de travailler avec les microcontrôleurs, tout en offrant plusieurs avantages pour les enseignants, les étudiants et les amateurs intéressés par les autres systèmes :

- **Pas cher** : les cartes Arduino sont relativement peu coûteuses comparativement aux autres plateformes. La moins chère des versions du module Arduino peut être assemblée à la main, et même les cartes Arduino pré-assemblées coûtent moins de 25 Euros (microcontrôleur inclus...) !!!
- **Multi-plateforme** : Le logiciel Arduino, écrit en Java, tourne sous les systèmes d'exploitation Windows, Macintosh et Linux. La plupart des systèmes à microcontrôleurs sont limités à Windows.
- **Un environnement de programmation clair et simple**: L'environnement de programmation Arduino (= le logiciel Arduino) est facile à utiliser pour les débutants, tout en étant assez flexible pour que les utilisateurs avancés puisse en tirer profit également.
- **Logiciel Open Source et extensible** : Le logiciel Arduino et le langage Arduino sont publiés sous licence open source, disponible pour être complété par des programmeurs expérimentés. Le langage peut être aussi étendu à l'aide de librairies C++, et les personnes qui veulent comprendre les détails techniques peuvent reconstruire le passage du langage Arduino au langage C pour microcontrôleur AVR sur lequel il est basé.

- **Matériel Open source et extensible** : Les cartes Arduino sont basé sur les microcontrôleurs Atmel ATMEGA8, ATMEGA168, ATMEGA 328, etc... Les schémas des modules sont publiés sous une licence Creative Commons, et les concepteurs de circuits expérimentés peuvent réaliser leur propre version des cartes Arduino, en les complétant et en les améliorant.

3. Qu'est-ce qu'un microcontrôleur ?

Un microcontrôleur est l'équivalent d'un petit ordinateur, tel votre PC par exemple, contenu dans un seul boîtier de circuit intégré à plus ou moins grand nombre de pattes.
Il contient ainsi une unité centrale — l'équivalent du microprocesseur qui équipe votre PC — de la mémoire vive, de la mémoire morte, des interfaces diverses pour communiquer avec le monde extérieur et toute la circuiterie électronique et logique nécessaire pour faire fonctionner tout cela ensemble.

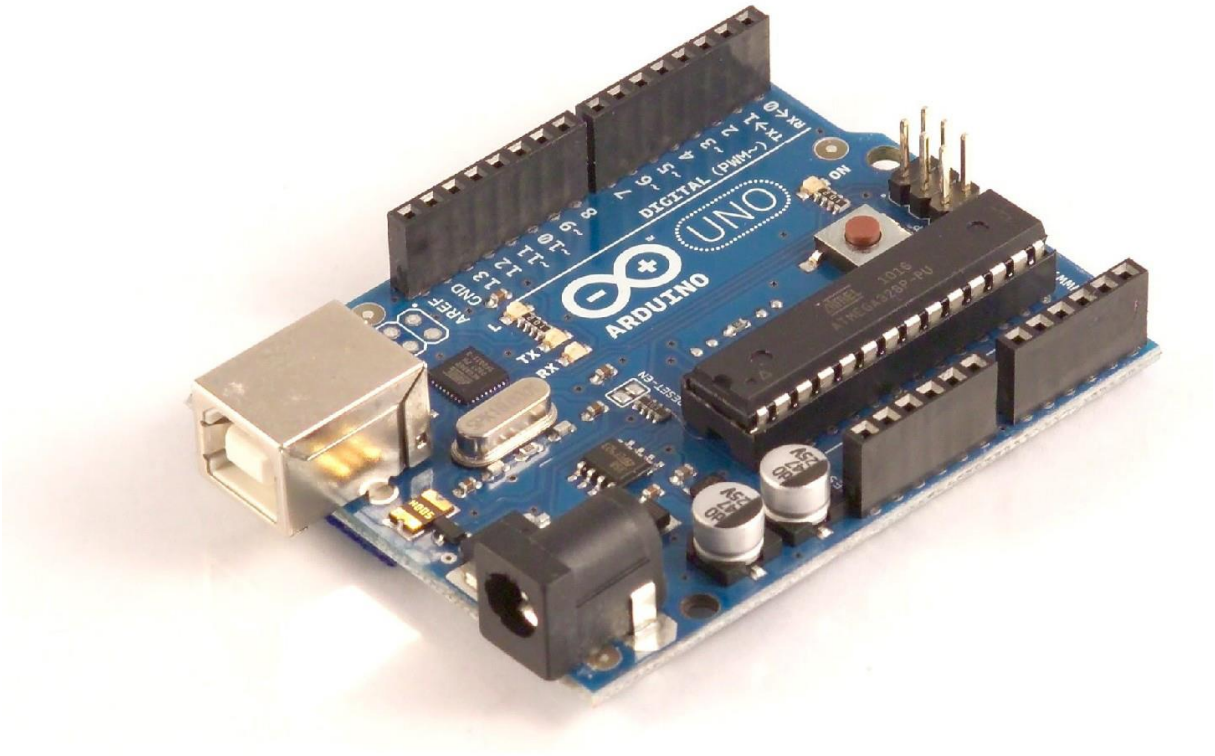


La figure ci-dessus présente le contenu, très simplifié d'un microcontrôleur. On y constate que tous les éléments contenus dans le boîtier sont reliés entre eux par ce qui s'appelle un bus, qui est en fait un ensemble de connexions véhiculant les adresses, les données et les signaux de contrôle échangés entre ces différents sous-ensembles.

Pour ce qui est de ces sous-ensembles internes, leur nombre et leurs types sont variables et dépendent du microcontrôleur choisi. L'unité centrale est évidemment toujours présence puisque c'est elle le cerveau du microcontrôleur. La mémoire également puisqu'elle est indispensable pour contenir le programme que va exécuter le circuit, mais son type et sa taille varient énormément d'un circuit à un autre ;
Pour ce qui est des sous-ensembles d'interface, leur nombre et leurs types varient selon le microcontrôleur choisi, mais l'on rencontre quasiment toujours un ou plusieurs timers ou compteurs, des entrées/sorties parallèles, des entrées/sorties séries et des convertisseurs analogiques/numériques et numériques/ analogiques.

Compte tenu de l'intégration de tous ces éléments dans un seul et unique boîtier de circuit intégré, il ne faut que très peu de composants électroniques externes autour d'un microcontrôleur pour le faire fonctionner.

4. L'Arduino Uno



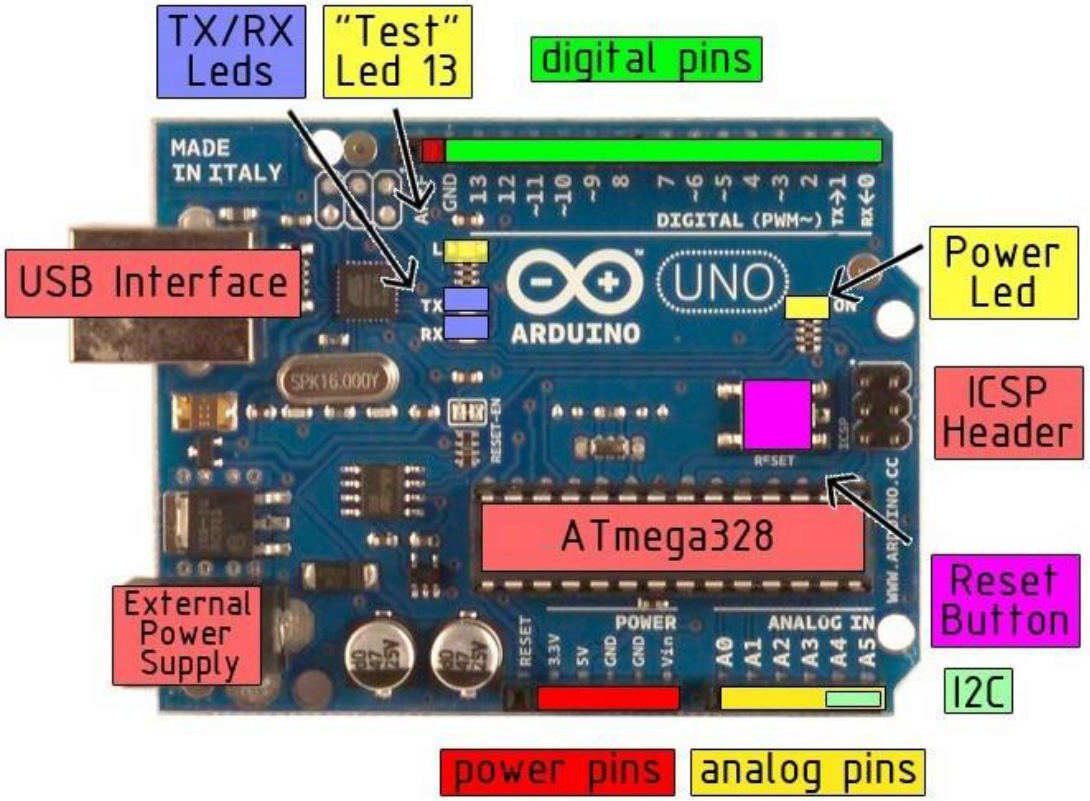
La carte Arduino uno

Résumé des caractéristiques de la carte Arduino uno :

Micro contrôleur : ATmega328
Tension d'alimentation interne = 5V
tension d'alimentation (recommandée)= 7 à 12V, limites =6 à 20 V
Entrées/sorties numériques : 14 dont 6 sorties PWM
Entrées analogiques = 6
Courant max par broches E/S = 40 mA
Courant max sur sortie 3,3V = 50mA
Mémoire Flash 32 KB dont 0.5 KB utilisée par le bootloader
Mémoire SRAM 2 KB
mémoire EEPROM 1 KB
Fréquence horloge = 16 MHz
Dimensions = 68.6mm x 53.3mm
La carte s'interface au PC par l'intermédiaire de sa prise USB.
La carte s'alimente par le jack d'alimentation (utilisation autonome) mais peut être alimentée par l'USB (en phase de développement par exemple).

4.1 Dimensions et organisation

Ces cartes sont au format « standard » Arduino c'est-à-dire qu'elles mesurent environ 52 mm sur 65 mm et se présentent comme ci-dessous.



Elle est équipée d'un microcontrôlcur Atmel ATmega 328. Deux rangées de connecteurs, situées de part et d'autre de la carte, permettent sa connexion au monde extérieur et, si l'on utilise les cartes d'interface, ou shields, au format Arduino, il est possible d'enficher directement ces derniers dans la carte de base, sur plusieurs niveaux si nécessaire.

4.2 Alimentation

L'alimentation du microcontrôleur qui équipe les cartes a lieu sous une tension de 5 volts qui peur provenir soit de la prise USB dont elles sont munies, ce qui est le cas lorsque la carte est reliée à un ordinateur ; soit d'un bloc secteur externe .

Par ailleurs, et par mesure de sécurité pour l'ordinateur auquel sera relié l'Arduino, un fusible réarmable ou Polyfuse" est présent sur la connexion d'alimentation 5 volts de la prise USB. Toute consommation excessive sur cette dernière, c'est-à-dire en pratique supérieure à 500 mA, provoque le déclenchement de ce fusible,protégeant ainsi le port USB de l'ordinateur auquel la carte est reliée. Le fusible étant de type réarmable, il n'a pas besoin d'être remplacé lorsqu'il a été activé et il suffit de patienter quelques secondes pour qu'il retrouve sont état normal.

Deux tensions stabilisées sont générées par les régulateurs montés sur les cartes Arduino : 5 volts et 3,3 volts et ces deux tensions sont disponibles sur les connecteurs placés sur le pourtour des cartes pour alimenter les shieids (les cartes d'interface).

4.3 Horloge

L'horloge est pilotée par quartz et fonctionne à la fréquence de 16 MHz.

Ainsi par exemple, la fonction delay(xxx), qui génère un délai de xxx ms, sait que l'horloge fonctionne à 16 MHz et fait donc automatiquement appel aux instructions nécessaires pour générer le délai que vous avez réellement demandé dans votre programme.

4.4 Reset

Toutes les cartes Arduino actuelles sont équipées d'un poussoir de reset manuel. Un appui sur celui-ci permet donc de relancer l'exécution d'un programme si nécessaire, soit parce qu'il s'est « planté » soit tout simplement parce que l'on souhaite le faire repartir de son début.Mais il existe aussi sur les cartes Arduino une autre source de reset.

Un reset automatique à la mise sous tension qui permet ainsi au programme contenu en mémoire du microcontrôleur de démarrer automatiquement dès que la carte Arduino est alimentée.

4.5 Les mémoires

L'ATmega 328 dispose de 32 kilo-octets de mémoire de programme. Cette mémoire est de type Flash, analogue à celle que l'on trouve par exemple dans les clés USB.

L'ATmega 328 contient aussi de la mémoire vive ou RAM, analogue dans son principe à la mémoire vive de vos PC ou Mac, mais en beaucoup plus petite quantité puisque l'on ne dispose ici que de 2 kilo-octets (contre plusieurs Giga octets dans un PC par exemple). Cette mémoire est généralement utilisée pour les variables employées dans les programmes, pour stocker des résultats temporaires lors de calculs, etc. Elle présente la particularité de pouvoir être lue et écrite à tout instant par le microcontrôleur mais elle « oublie » son contenu dès qu'il n'est plus alimenté.

L'ATmega 328 dispose également de mémoire EEPROM, acronyme qui signifie mémoire programmable et effaçable électriquement. La taille de cette EEPROM est seulement de 1 kilo-octets

. Cette mémoire EEPROM est une mémoire dans laquelle le microcontrôleur peut lire à tout instant. Il peut aussi y écrire et effacer son contenu, avec plus de facilité comme dans la mémoire Flash de programme, mais avec moins de facilité que dans la mémoire vive. En contrepartie, et comme c'est le cas pour la mémoire flash de programme, le contenu de l'EEPROM est conservé même lorsqu'elle n'est plus alimentée. C'est donc une mémoire qui sera utilisée pour conserver des données ou des paramètres que l'on doit pouvoir retrouver d'une utilisation à l'autre de l'Arduino.

Pour faire un parallèle avec des produits que vous connaissez, c'est par exemple dans une mémoire de ce type que votre téléviseur stocke les informations relatives aux chaînes de la TNT que vous lui faites découvrir lorsque vous le mettez en marche pour la première fois. Il conserve ensuite ces informations même lorsqu'il est éteint mais, si vous changez de région, vous pourrez à nouveau lui faire mémoriser de nouvelles informations qui remplaceront alors les précédentes dans son EEPROM.

4.6 Les entrées/sorties

Les entrées/sorties de l'Arduino sont les éléments qui permettent de communiquer avec le monde extérieur et donc avec les cartes d'interface ou shields.

• Entrées/sorties numériques

Quatorze lignes d'entrées/sorties numériques parallèles, repérées 0 à 13, sont disponibles ; chacune d'entre elles pouvant fonctionner en entrée ou en sortie sous le contrôle du programme et ce sens de fonctionnement pouvant même changer de manière dynamique pendant son exécution. Ces entrées/sorties admettent et délivrent des signaux logiques compatibles TTL c'est-à-dire compris entre 0 et 5 volts.

Ces lignes partagent cependant leur rôle avec certaines interfaces spécialisées contenues dans le microcontrôleur. Si ces interfaces ne sont pas utilisées par l'application, les entrées/sorties parallèles fonctionnent comme telles. Si ces interfaces sont utilisées par l'application, les lignes d'entrées/sorties parallèles qui sont partagées avec elles ne sont évidemment plus disponibles en tant que telles.

Sous ces conditions de partage on peut ainsi disposer de :

- une entrée (Rx) et une sortie (Tx) série asynchrone, partagées respectivement avec les lignes 0 et 1 ;
- deux entrées d'interruptions externes partagées avec les lignes 2 et 3 ;
- six sorties PWM {Pulse Width Modulation ou modulation de largeur d'impulsions) partagées avec les lignes 3, 5, 6, 9, 10 et 11 ;
- quatre entrées/sorties d'interface avec le bus série normalisé de type SPI réparties comme suit : /SS sur 10, MOSI sur 11, MISO sur 12 et SCK sur 13 ;
- deux entrées/sorties d'interface série I2C réparties comme suit : SDA sur A4 et SCLsurA5.

Toutes ces entrées/sorties véhiculent des signaux numériques TTL, c'est-à-dire des signaux logiques ou signaux de type « tout ou rien » dont le niveau bas est nul et le niveau haut vaut 5 volts, mais l'Arduino sait aussi traiter les signaux analogiques.

. Entrées analogiques

II dispose pour cela de six entrées, repérées AO à A5, qui peuvent admettre toute tension analogique comprise entre 0 et 5 volts (par défaut mais cela peut être modifié), mais pouvant bien sûr prendre n'importe quelle valeur dans cette plage puisque ce sont des entrées analogiques, contrairement aux signaux logiques évoqués précédemment.

Ces entrées analogiques sont gérées par un convertisseur analogique/numérique de 10 bits de résolution c'est-à-dire encore dont la sortie peut varier de 0 à 1023.

5. Les shields

Il existe de nombreux shields. Un « shield » Arduino est une petite carte qui se connecte sur une carte Arduino pour augmenter ses fonctionnalités. Quelques exemples de « shields » :

• Afficheur graphique
• Ethernet et carte SD
• GPS
• XBee
• etc...

Image : Arduino & ethernet par Interactive Matter - flickr.com

6. Développement d'un projet

Le développement sur Arduino est très simple :

• on code l'application : Le langage Arduino est basé sur les langages C/C++ , avec des fonctions et des librairies spécifiques à Arduino (gestions des e/s).

• on relie la carte Arduino au PC et on transfert le programme sur la carte,

• on peut utiliser le circuit

Le logiciel de programmation des modules Arduino est une application Java multiplateformes (fonctionnant sur tout système d'exploitation), servant d'éditeur de code et de compilateur, et qui peut transférer le firmware (et le programme) au travers de la liaison série (RS232, Bluetooth ou USB selon le module).

7. Le Logiciel Arduino : Environnement de développement Intégré (EDI)

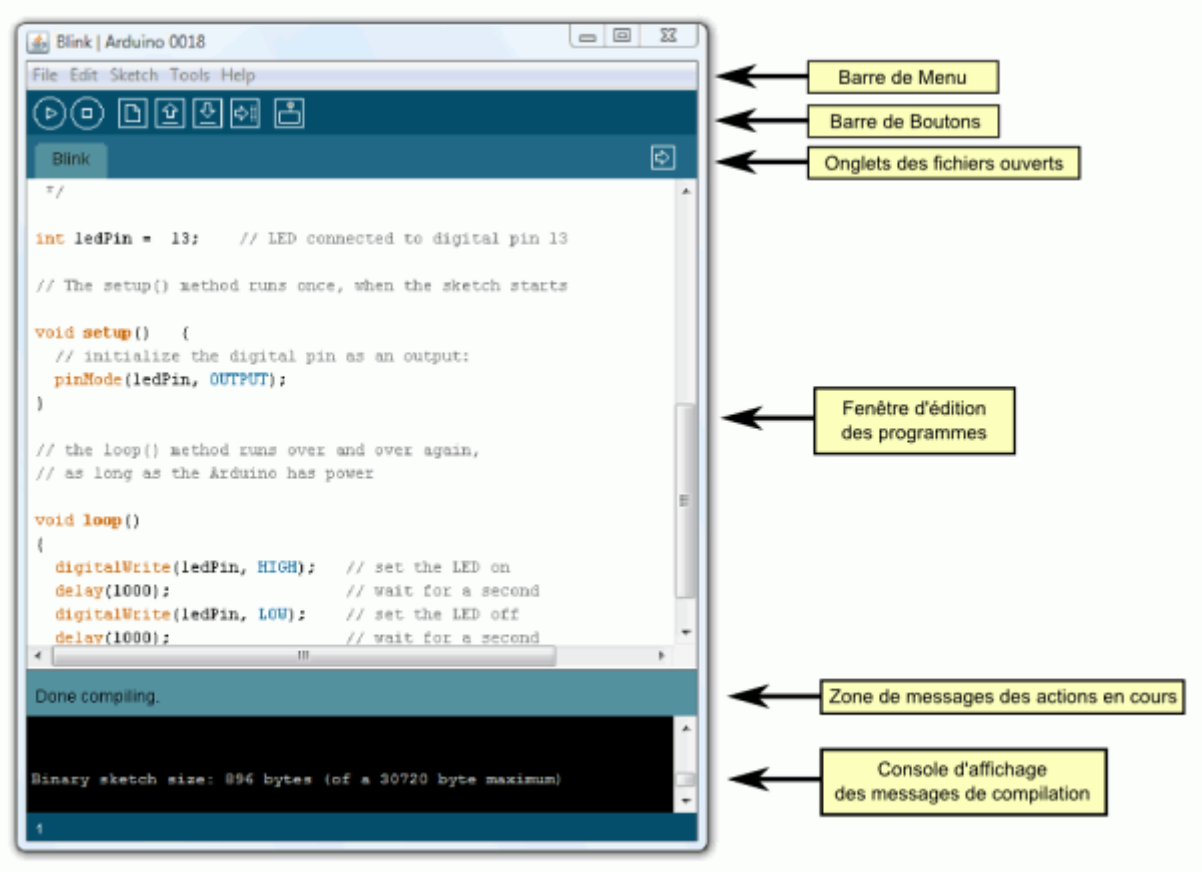
7.1 Description

Le logiciel Arduino a pour fonctions principales :

- de pouvoir écrire et compiler des programmes pour la carte Arduino
- de se connecter avec la carte Arduino pour y transférer les programmes
- de communiquer avec la carte Arduino

Il comporte :

- une **BARRE DE MENUS** comme pour tout logiciel une interface graphique (GUI),
- une **BARRE DE BOUTONS** qui donne un accès direct aux fonctions essentielles du logiciel et fait toute sa simplicité d'utilisation,
- un **EDITEUR** (à coloration syntaxique) pour écrire le code de vos programmes, avec onglets de navigation,
- une **ZONE DE MESSAGES** qui affiche indique l'état des actions en cours,
- une **CONSOLE TEXTE** qui affiche les messages concernant le résultat de la compilation du programme



Description de la barre des boutons

Vérifier/compiler : Vérifie le code à la recherche d'erreur.

Stop : Stoppe le moniteur série ou les autres boutons activés.

Nouveau : Crée un nouveau code (ouvre une fenêtre d'édition vide)

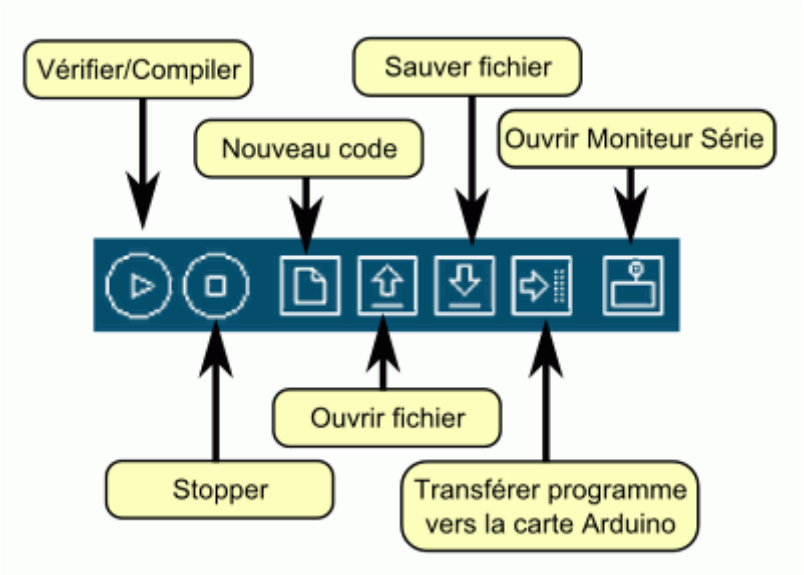
Ouvrir : Ouvre la liste de tous les programmes dans votre "livre de programmes". Cliquer sur l'un des programmes l'ouvre dans la fenêtre courante.

Note: en raison d'un bug dans Java, ce menu ne défile pas. Si vous avez besoin d'ouvrir un programme loin dans la list, utiliser plutôt le menu **File > Sketchbook**.

Sauver : Enregistre votre programme.

Transférer vers la carte : Compile votre code et le transfère vers la carte Arduino. Voir ci-dessous "Transférer les programmes" pour les détails.

Moniteur Série : Ouvre la fenêtre du moniteur (ou terminal) série.



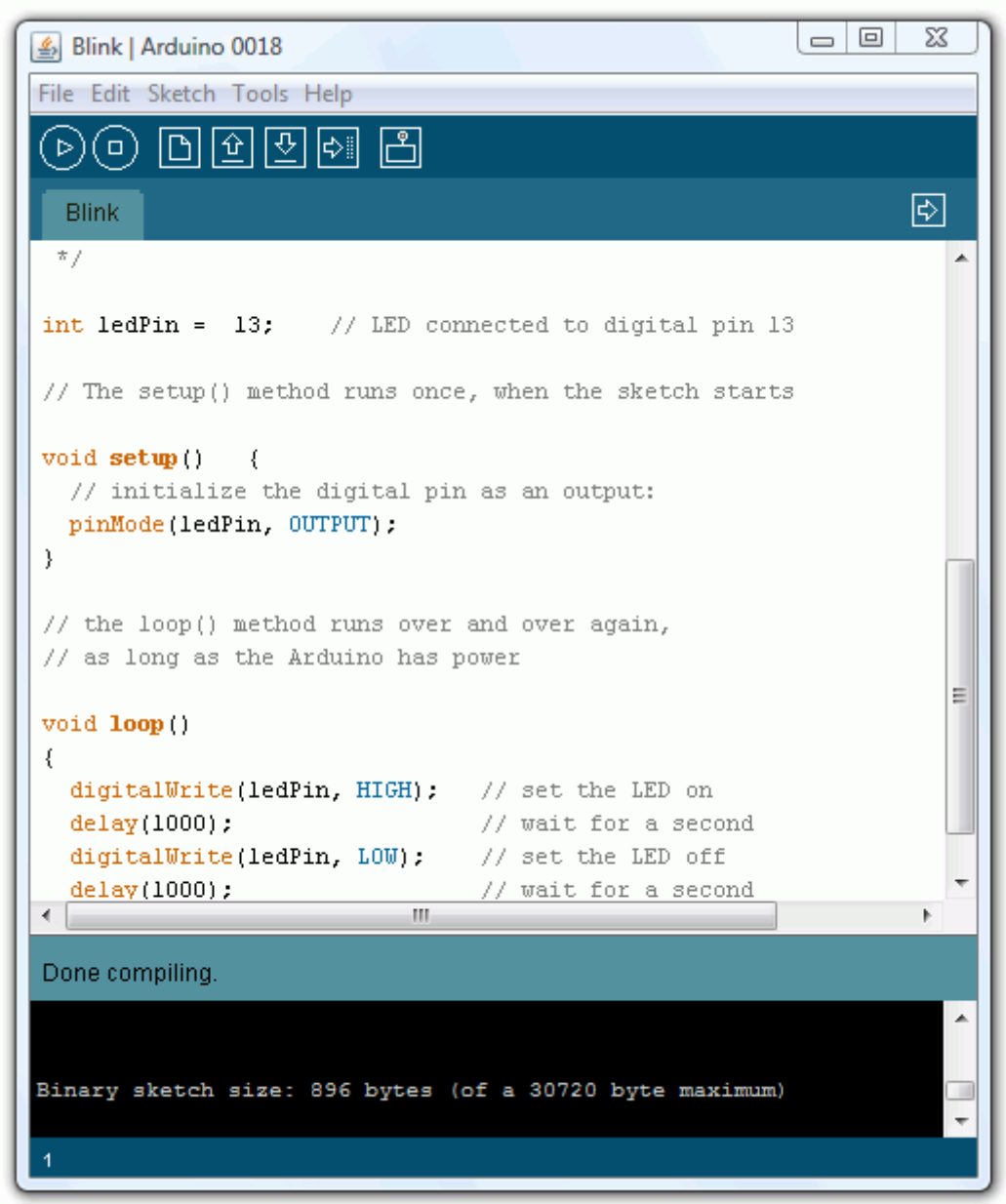
7.2 Transfert des programmes vers la carte Arduino

7.2.1 Saisir votre programme et vérifier le code.

On suppose ici qu'un programme correctement écrit se trouve dans la fenêtre éditeur. Pour votre première programmation de la carte, aller dans le menu **File>Examples>Digital>Blink** : un programme s'ouvre avec du code dans la fenêtre éditeur.

Appuyez alors sur le bouton **Verify** de la barre d'outils pour lancer la vérification du code :

Si tout va bien, aucun message d'erreur ne doit apparaître dans la console et la zone de message doit afficher **Done Compiling** attestant que la vérification s'est bien déroulée.



7.2.2 Sélectionner le bon port série (et la bonne carte Arduino...).

Avant de transférer votre programme vers la carte Arduino, vous devez vérifier que vous avez bien sélectionné la bonne carte Arduino depuis le menu **Tools>Board** (Outils>Carte). Les cartes sont décrites ci-dessous. Votre carte doit évidemment être connectée à l'ordinateur via un câble USB.

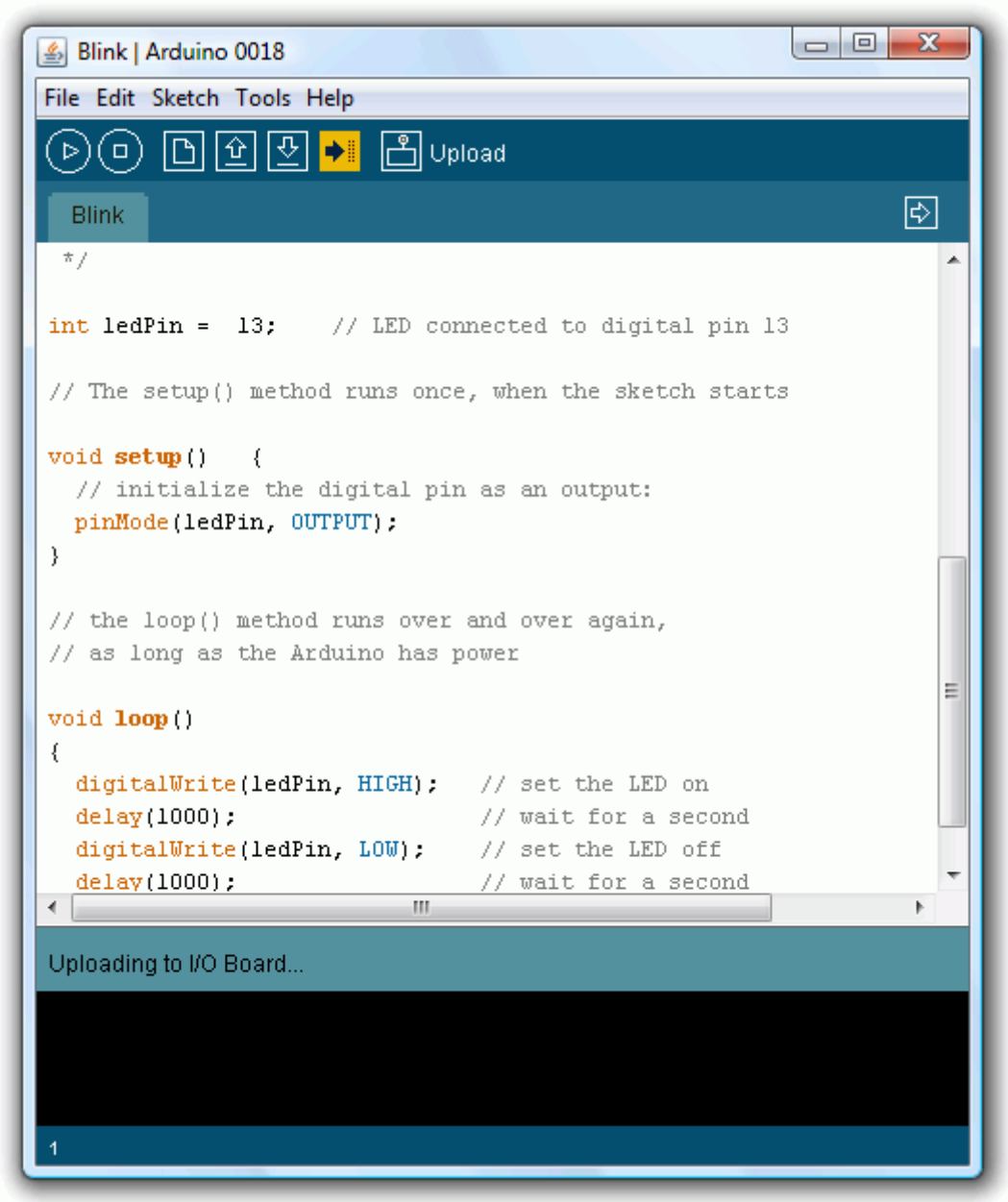
Vous devez également sélectionner le bon port série depuis le menu **Tools > Serial Port** (Outils > Port Série) :

- Sous Winsows, c'est probablement COM1 ou COM2 (pour une carte série) ou COM4, COM5, COM7 ou supérieur (pour une carte USB) - pour trouver le bon, vous pouvez chercher dans la rubrique des ports série USB dans la section des ports du panneau de configuration ou du gestionnaire de périphériques.
- Sous Linux, çà devrait être /dev/ttyUSB0, /dev/ttyUSB1 ou équivalent.

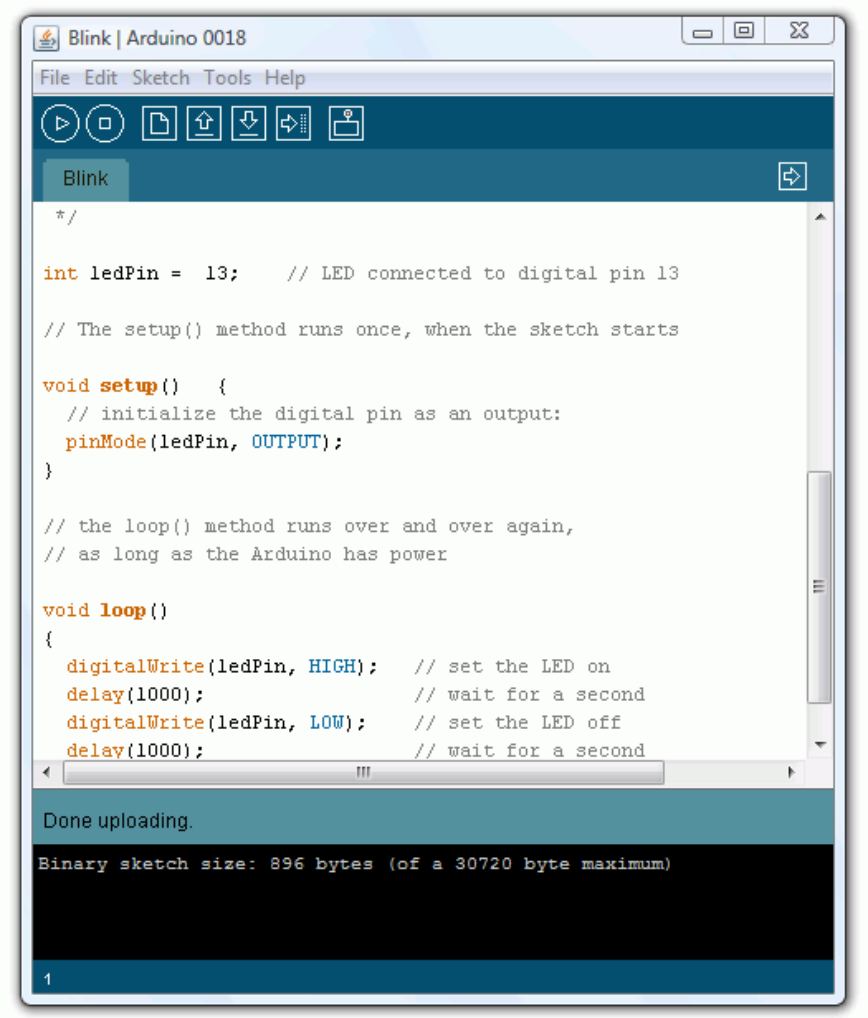
7.3.3 Clic sur le bouton UPLOAD (Transfert du programme)

Une fois que vous avez sélectionné le bon port série et la bonne carte Arduino, cliquez sur le bouton **UPLOAD** (Transfert vers la carte) dans la barre d'outils, ou bien sélectionner le menu **File>Upload to I/O board** (Fichier > Transférer vers la carte). La carte Arduino va alors automatiquement se réinitialiser et démarrer le transfert.

Sur la plupart des cartes, vous devez voir les LEDs des lignes RX et TX clignoter rapidement, témoignant que le programme est bien transféré. Durant le transfert, le bouton devient jaune et le logiciel Arduino affiche un message indiquant que le transfert est en cours :



Une fois le transfert terminé, le logiciel Arduino doit afficher un message indiquant que le transfert est bien réalisé, ou montrer des messages d'erreurs.



7.4.4 Le programme transféré avec succès se lance

Quand vous transférez un programme, en utilisant le bootloader Arduino, un petit programme (code binaire) a été chargé dans le microcontrôleur sur votre carte Arduino. Cette technique vous permet comme vous avez pu le voir de transférer votre programme sans aucun matériel externe. Une fois le transfert terminé, le bootloader est actif une petite seconde ("écoute" pour voir si un nouveau programme arrive...) une fois que la carte est réinitialisée à la fin du transfert; puis le dernier programme programmé dans la carte s'exécute.

Note : Le bootloader fait clignoter la LED de la carte (broche 13) quand il démarre (c'est à dire quand la carte est réinitialisée).

Liens
Le site Arduino : http://www.arduino.cc
Traduit en français (partiellement) : http://www.arduino.cc/fr