Initiation Arduino

Vous avez dit Arduino ?

(extraits du manuel floss : http://fr.flossmanuals.net/arduino/index)

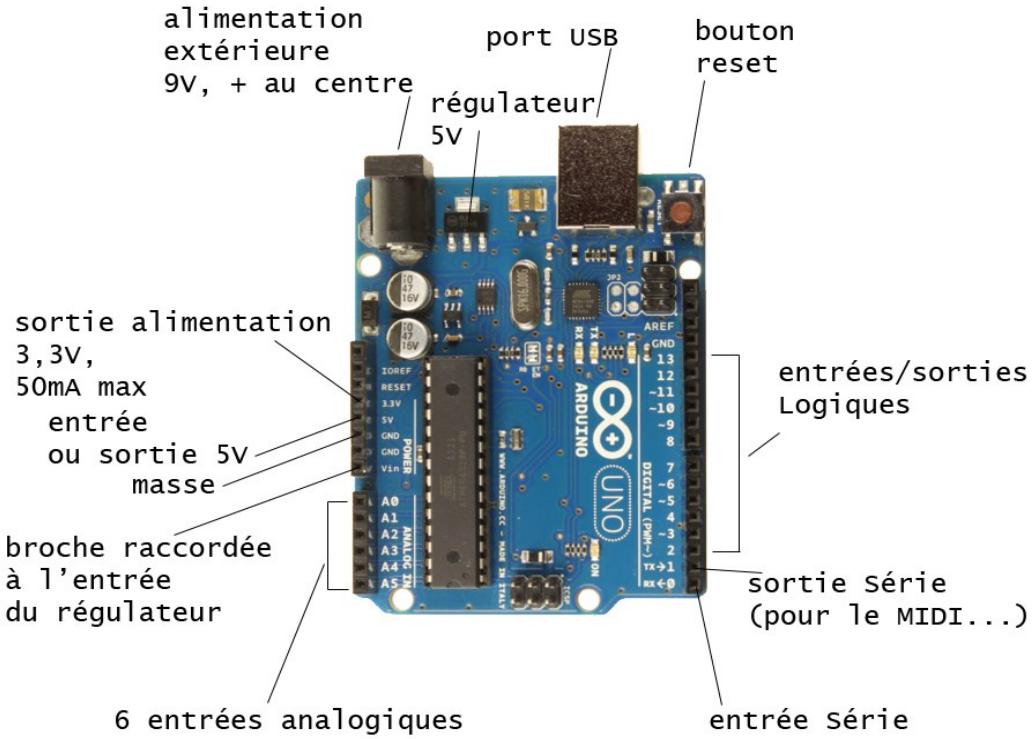
Arduino est une plate-forme de prototypage d'objets interactifs à usage créatif constituée d'une carte électronique et d'un environnement de programmation.

Sans tout connaître ni tout comprendre de l'électronique, cet environnement matériel et logiciel permet à l'utilisateur de formuler ses projets par l'expérimentation directe avec l'aide de nombreuses ressources disponibles en ligne.

Pont tendu entre le monde réel et le monde numérique, Arduino permet d'étendre les capacités de relations humain/machine ou environnement/machine.

Arduino est un projet en source ouverte : la communauté importante d'utilisateurs et de concepteurs permet à chacun de trouver les réponses à ses questions.

Matériel



La carte Arduino repose sur un circuit intégré (un mini ordinateur appelé également micro-contrôleur) associée à des entrées et sorties qui permettent à l'utilisateur de brancher différents types d'éléments externes.

Il existe un grand nombre de variantes :

le Lilipad, pour fixer sur des vêtements, le RBBB qui est une carte très petite et économique etc...
http://arduino.cc/en/Main/Hardware

le Teensy, ne fait pas proprement partie des cartes Arduino, mais on peut le programmer à partir de l'environnement Arduino. Son processeur est plus puissant, les E/S sont plus nombreuses, il est minuscule et on peut disposer d'un interface USB MIDI natif...

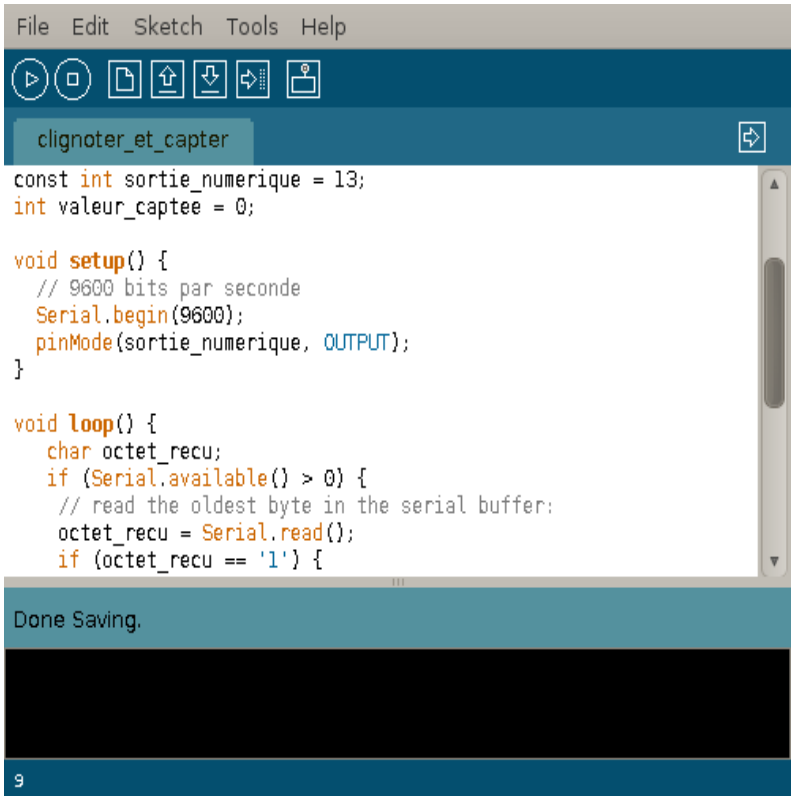
http://www.pjrc.com/teensy/

La série Mapple (et compatibles) est une plateforme 100 fois plus puissante. La programmation est similaire à celle de l'Arduino.

Logiciel

L'environnement de programmation Arduino (IDE en anglais) est une application écrite en Java inspirée du langage Processing.

L'IDE permet d'écrire, de modifier un programme et de le convertir en une série d'instructions compréhensibles pour la carte.



Quelques exemples d'applications :
http://www.semageek.com/category/electronique/arduino-electronique/

 *New* : pour créer un nouveau programme (sketch).

 *Open* : ouvrir un programme existant. Le menu n'est pas déroulant à cause d'un bug...pour obtenir un menu déroulant passer par file/open

 *Save : sauvegarde le programme, si vous voulez le sauvegarder sous un autre nom, passer par file/save as*

 *Serial Monitor* : pour ouvrir la fenêtre qui permet de visualiser les données transmise par le port série de l'Arduino (très pratique pour le débogage...)

historique

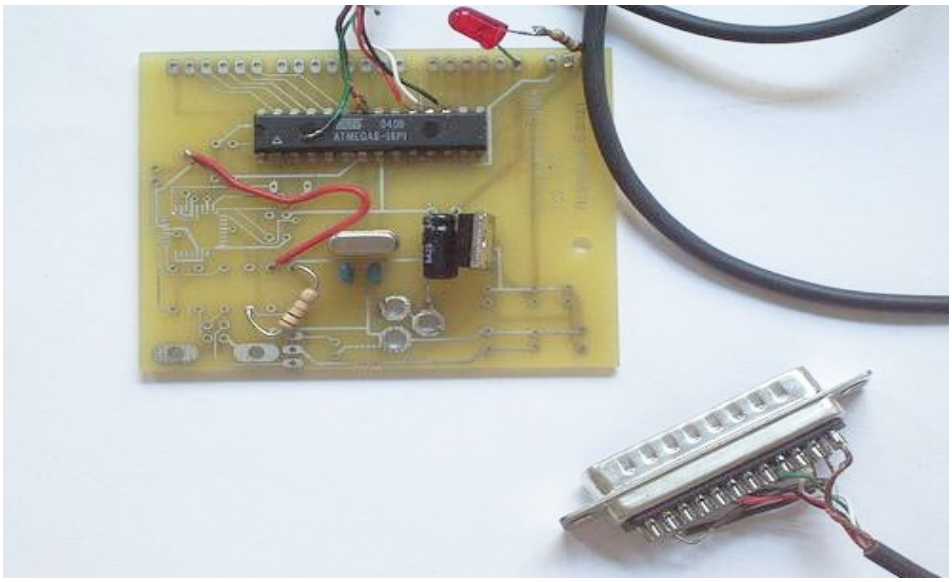
D'après Wired Magazine :

Le projet Arduino est né en hiver 2005. Massimo Banzi enseigne dans une école de Design à Ivrea en Italie, et souvent ses étudiants se plaignent de ne pas avoir accès à des solutions bas prix pour accomplir leurs projets de robotique. Banzi en discute avec David Cuartielles, un ingénieur Espagnol spécialisé sur les micro-contrôleurs…

Ils décident de créer leur propre carte en embarquant dans leur histoire un des étudiant de Banzi, David Mellis qui sera chargé de créer le langage de programmation allant avec la carte. En deux jours David écrira le code! Trois jours de plus et la carte était crée…Ils décidèrent de l'appeler Arduino (un bar fréquenté par les élèves à proximité de l'école)…

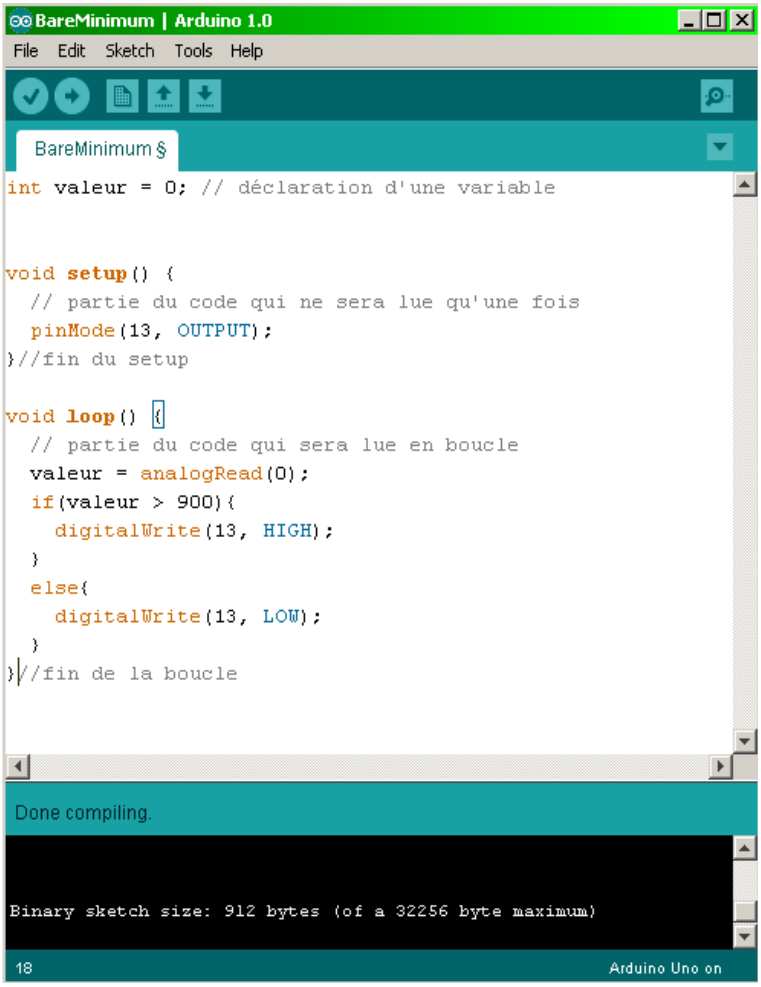
Ca devient un hit tout de suite auprès des étudiants. Tout le monde arrive à en faire quelquechose très rapidement sans même avoir de connaissances particulière ni en électronique ni en informatique: réponse à des capteurs, faire clignoter des leds, contrôler des moteurs… Ils publient les schémas, investissent 3000 euros pour créer le premier lots de cartes: 200.

Les 50 premières partent directement à des élèves de l'école. En 2006 5 000 cartes vendues…En 2007 plus de 30 000! en 2011 : >120 000, sans compter les clones !



Prototype de l'Arduino.

Structure d'un programme



Le programme est lu par le micro-contrôleur de haut vers le bas.
Une variable doit être déclarée avant d'être utilisée par une fonction.

La structure minimale est constituée :

- en tête : déclaration des variables, des constantes, indication de l'utilisation de bibliothèques etc...
- un setup (= initialisation) cette partie n'est lue qu'une seule fois, elle comprend les fonctions devant être réalisées au démarrage (utilisation des broches en entrées ou en sortie, mise en marche du midi, du port série de l' I2C etc.....)
- une loop (boucle) : cette partie est lue en boucle ! C'est ici que les fonctions sont réalisées.

En plus de cette structure minimale, on peut ajouter :

- des « sous-programmes » ou « routines » qui peuvent être appelées à tous moments dans la boucle, très pratiquer pour réaliser des morceaux de codes répétitifs.
- Des « callbacks », ce sont des fonctions qui sont rappelées automatiquement depuis une bibliothèque.

référence :

Structure

Fonctions de base

Ces deux fonctions sont obligatoires dans tout programme en langage Arduino :

- void setup()
- void loop()

Structures de contrôle

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Syntaxe de base

- ; (point virgule)
- {} (accolades)
- // (commentaire sur une ligne)
- /* */ (commentaire sur plusieurs lignes)
- #define
- #include

Opérateurs arithmétiques

- = (égalité)
- + (addition)
- - (soustraction)
- * (multiplication)
- / (division)
- % (modulo)

Opérateurs de comparaison

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

Opérateurs booléens

- && (ET booléen)
- || (OU booléen)
- ! (NON booléen)

Pointeurs

- * pointeur
- & pointeur

Voir également :

- Manipulation des Ports

Opérateurs bit à bit

- & (ET bit à bit)
- |(OU bit à bit)
- ^(OU EXCLUSIF bit à bit)
- ~(NON bit à bit)
- << (décalage à gauche)
- >> (décalage à droite)

Opérateurs composés

- ++ (incrémentation)
- -- (décrémentation) (à revoir)
- += (addition composée)
- -= (soustraction composée)
- *= (multiplication composée)
- /= (division composée)
- &= (ET bit à bit composé)
- |= (OU bit à bit composé)

Variables et constantes

Les variables sont des expressions que vous pouvez utilisez dans les programmes pour stocker des valeurs, telles que la tension de sortie d'un capteur présente sur une broche analogique.

Constantes prédéfinies

Les constantes prédéfinies du langage Arduino sont des valeurs particulières ayant une signification spécifique.

- HIGH | LOW
- INPUT | OUTPUT
- true | false

A ajouter : constantes décimales prédéfinies

Expressions numériques

- Expressions numériques entières
- Expressions numériques à virgule

Types des données

Les variables peuvent être de type variés qui sont décrits ci-dessous.

Synthèse des types de données Arduino

- boolean
- char
- byte
- int
- unsigned int
- long
- unsigned long
- float (nombres à virgules)
- double (nombres à virgules)
- Les chaînes de caractères
- objet String NEW
- Les tableaux de variables
- le mot-clé void (fonctions)
- word
- PROGMEM

Voir également :

- Déclaration des variables

Pour info : les types de données avr-c

Conversion des types de données

- char()
- byte()
- int()
- long()
- float()
- word()

Portée des variables et qualificateurs

- Portée des variables
- static
- volatile
- const

Utilitaires

- sizeof() (opérateur sizeof)

Référence

- Code ASCII

Fonctions

Entrées/Sorties Numériques

- pinMode(broche, mode)
- digitalWrite(broche, valeur)
- int digitalRead(broche)

Entrées analogiques

- int analogRead(broche)
- analogReference(type)

Sorties "analogiques" (génération d'impulsion)

- analogWrite(broche, valeur) - PWM

Entrées/Sorties Avancées

- tone()
- noTone()
- shiftOut(broche, BrocheHorloge, OrdreBit, valeur)
- unsigned long pulseIn(broche, valeur)

Communication

- Serial

Temps

- unsigned long millis()
- unsigned long micros()
- delay(ms)
- delayMicroseconds(us)

Math

- min(x, y)
- max(x, y)
- abs(x)
- constrain(x, a, b)
- map(valeur, toLow, fromHigh, toLow, toHigh)
- pow(base, exposant)
- sq(x)
- sqrt(x)

Pour davantage de fonctions mathématiques, voir aussi la librairie math.h : log, log10, asin, atan, acos, etc...

Nombres randomisés (hasard)

- randomSeed(seed)
- long random(max)
- long random(min, max)

Trigonométrie

- sin(rad)
- cos(rad)
- tan(rad)

Bits et Octets

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

Interruptions Externes

- attachInterrupt(interruption, fonction, mode)
- detachInterrupt(interruption)

Interruptions

- interrupts()
- noInterrupts()

Voir également la librairie interrupt.h.

http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceEtendue

les bibliothèques (librairies).

Les utilisateurs les plus avertis concoctent des bibliothèques pour interfacer, le plus simplement possible, une vaste diversité de composants (I2C, SPI...) et de fonctionnalités (MIDI, Ethernet, OSC...)

Sans ces bibliothèques, la programmation serait vraiment plus complexe ! À utiliser sans modération.

Les bibliothèques doivent être installées dans le répertoire « libraries » et doivent être inclues dans le programme (exemple : #include <MIDI.h>).

Démarrer avec Arduino sous Windows

1 | Obtenir une carte Arduino et un câble USB

2 | Télécharger l'environnement Arduino

Télécharger le fichier ici : http://arduino.cc/en/Main/Software

décompresser le fichier et le copier dans « mes documents »

3 | Raccorder la carte à l'ordinateur

La diode verte doit s'allumer.

4 | Installation des pilotes du périphérique Série-USB

- lors d'un premier raccordement d'une carte UNO à l'ordinateur, ce dernier recherche automatiquement un pilote. Après quelques instants, l'ordinateur va indiquer qu'il n'a pas trouvé de pilote...
- ouvrir « executer » en passant par le menu démarrer
- taper **devmgmt.msc** et taper sur la touche « entrée »
- Chercher et déployer la rubrique « ports (COM et LPT).
- Il doit apparaître une ligne «Arduino UNO (COMxx) »
- faire un click droit sur «Arduino UNO (COMxx) » et choisir « mettre à jour le pilote... ».
- Choisir « rechercher un pilote sur mon ordinateur »
- selectionner le fichier **"ArduinoUNO.inf"**, qui se trouve dans le repertoire « drivers » de l'installation de l'Arduino...

• Windows va terminer l'installation des pilotes.

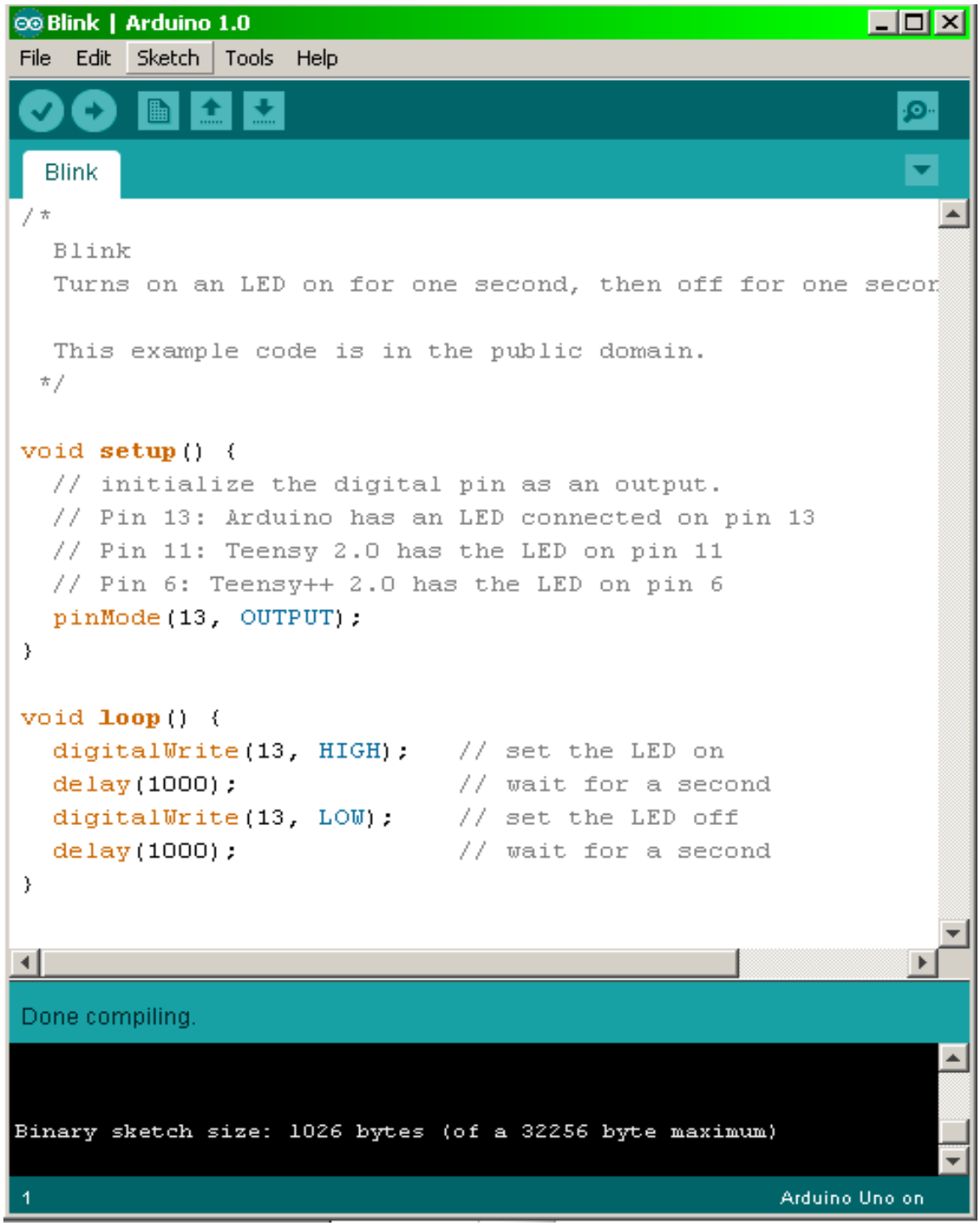
5 | Lancer l'aplication Arduino

Double -cliquer sur

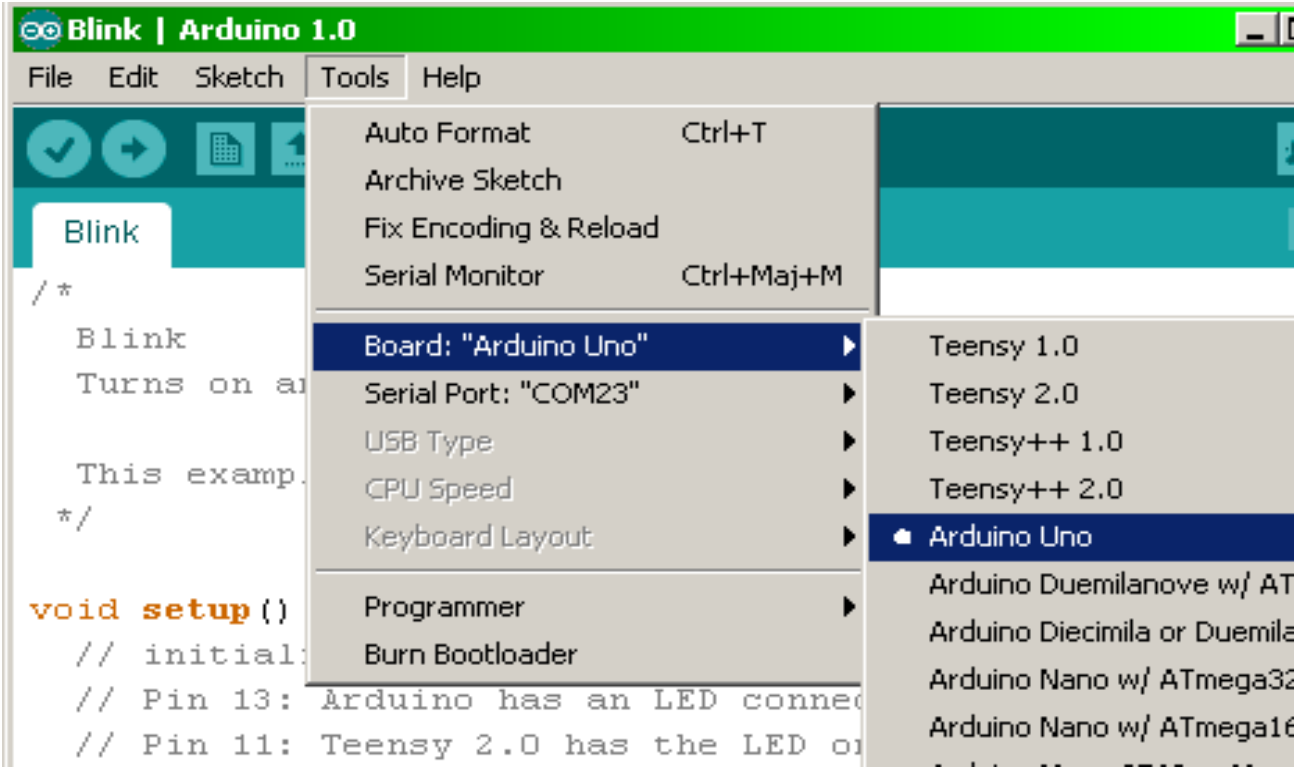
6 | Ouvrir l'exemple « blink »

Ouvrir le programme (sketch)

File > Examples > 1.Basics > Blink.



7 | choisir la carte UNO



8 | sélectionner le port série

Choisir le port série qui est utilisé par la carte arduino : tools > serial port > …

pour connaître quel port série est utilisé par la carte Arduino, lancer le gestionnaire de périphérique (voir item 4).

9 | charger le programme dans la carte Arduino

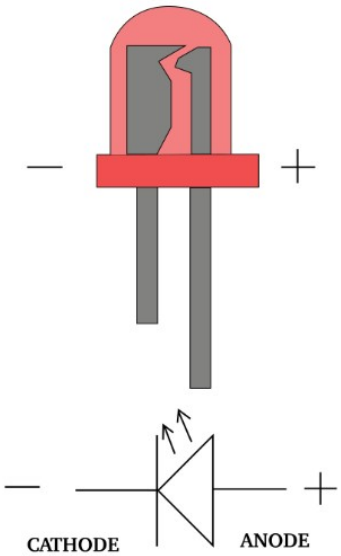
Cliquer sur la touche « upload »

après quelques secondes, la LED orange qui est raccordée à la broche 13 devrait clignoter.

En cas de problème : troubleshooting suggestions.

Confronter la théorie à l'épreuve de la réalité...

La carte Arduino est une petite chose fragile, il convient de bien en comprendre ses limites d'utilisation et vérifier la compatibilité avec les matériels que l'on souhaite y raccorder...



Tension :

le microcontrôleur placé sur la carte est prévu pour fonctionner entre 3,3V et 5V.

Intensité :

le courant de sortie de chaque broche (D0 à D13) ne doit pas dépasser 40mA

le courant issu du port USB ne doit pas dépasser 500mA

le courant soutiré à la broche « 3,3V » ne doit pas dépasser 50mA

exemple :

on souhaite actionner un moteur de 6W sous 5V par l'Arduino, alimenté par l'USB, est-ce réalisable ?

P = U.I => I = P/U => I = 1,2A

Au niveau du port USB ?

Au niveau d'une broche de sortie ?

Schémas de référence

LED :

Une **diode électroluminescente** (**DEL** ou **LED)** est un composant opto-électronique capable d'émettre de la lumière lorsqu'il est parcouru par un courant électrique.

Une diode électroluminescente ne laisse passer le courant électrique que dans un seul sens (le sens passant).

lorsqu'elle est traversée par un courant, la LED oppose une tension fixe

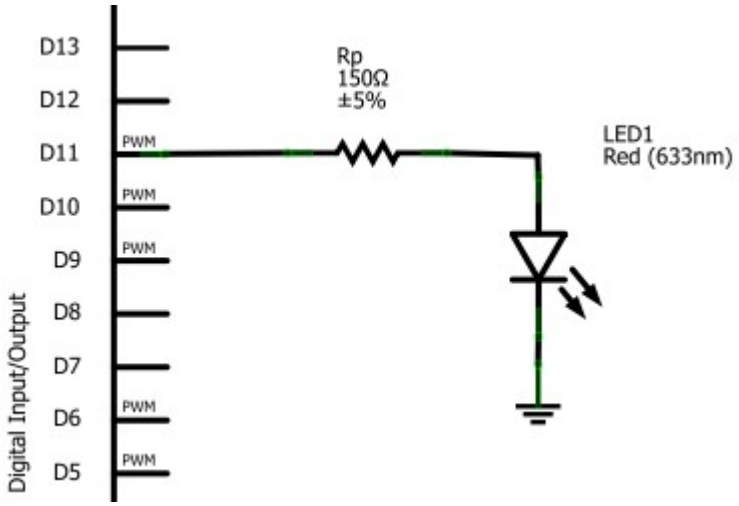
- 1,9V pour une LED rouge,
- 3,2V pour les diodes blanches, ou autres couleurs
- voir les notices des diodes avant d'utiliser.

Le courant dans la LED est aussi limité :

pour les LED ordinaires de 5mm, 24mA environ.

Certaines LED munies de 5 élèments dans un même boitier absorbe jusqu'à 100mA. (ces LED ne peuvent donc pas être directement commandées par une sortie d'Arduino...)

raccordement à l'Arduino :



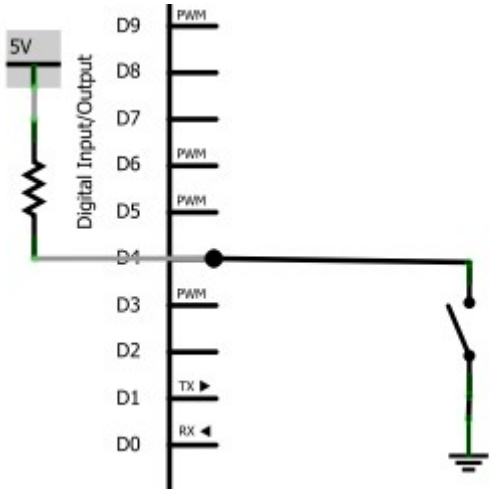
calcul de la résistance de protection :

$$R = \frac{tension\ d'alimentation - tension\ LED}{intensité\ en\ Ampère}$$

Exemple : pour une diode rouge, l'Arduino alimenté en 5V :

R =

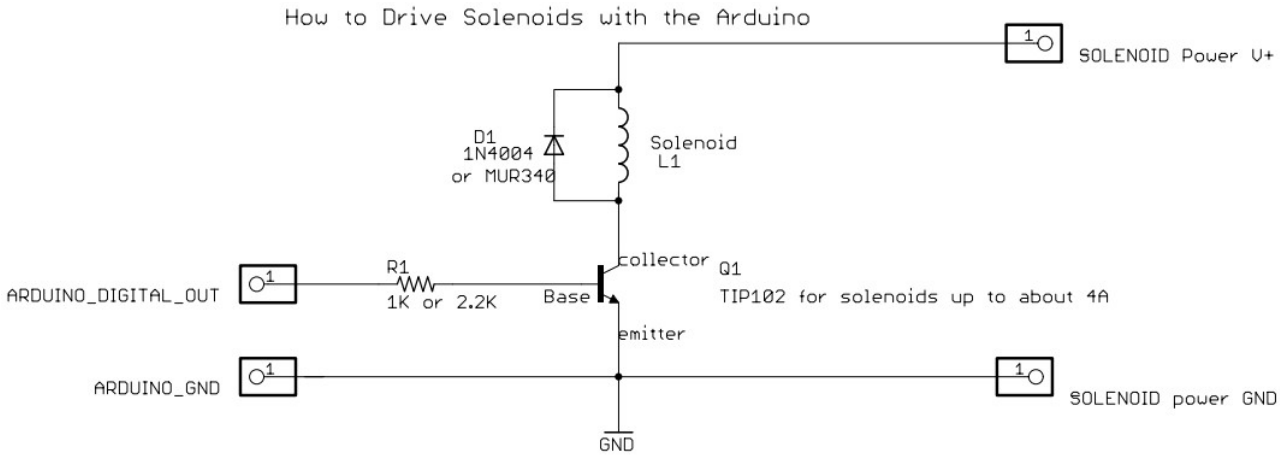
bouton, interrupteur :



Remarque : lorsque le bouton n'est pas actionné, la broche 5 est « en l'air », le fil va se comporter comme une antenne de radio et l'état de la broche va varier en permanence. Il faut donc que le niveau logique de la broche d'entrée soit fixé en permanence. Pour cela on met en oeuvre une résistance de « pullup » qui va tirer le potentiel de la broche vers le 5V, lorsque le bouton est au repos.

```
Void setup() {
pinMode(4, INPUT);     // la broche 5 se comporte comme une entrée
digitalWrite(4, HIGH);  // une résistance est raccordée entre la broche 5 et le+5V dans le microcontrolleur
}

void loop() {

        …

        if(digitalRead(4) == LOW) // attention, logique négative...

                {

                        //faire quelque chose lorsqu'on appuie sur le bouton.

                }

}
```

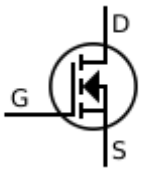
moteur :



Les moteur électriques à courant continu consomment trop de courant pour être commandés directement par une broche de l'Arduino, de plus ils sont inductifs : ils créent des surtensions à leurs bornes, il faut donc protéger le montage avec une diode de roue libre.

Il en va de même avec les relais.

On peut aussi utiliser un transistor à effet de champ type N : par exemple IRF Z34N



Servomoteur :

Un servomoteur est constitué d'un moteur, d'un capteur de position et d'un régulateur électronique. Le fonctionnement du moteur est asservi à la position de l'axe.

On le commande en lui indiquant quel angle doit prendre son axe (entre 0 et 180°) le moteur se met alors en marche jusqu'à ce que la position soit atteinte.

En utilisant une bibliothèque toute faite : #include <Servo.h>

il devient très simple de mettre en œuvre des servomoteurs de modélisme.

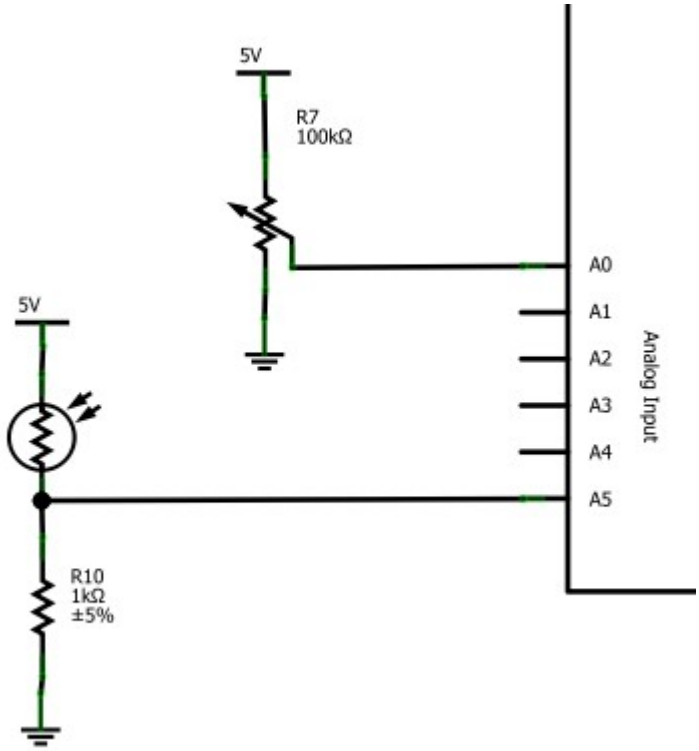
Raccordement :

marron : masse

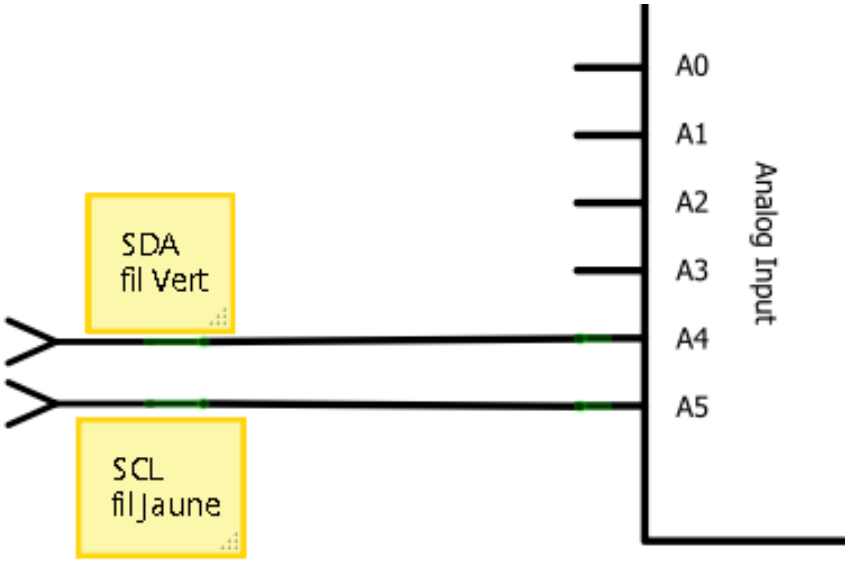
rouge : +5V

orange : commande, à raccorder à une broche de sortie de l'Arduino

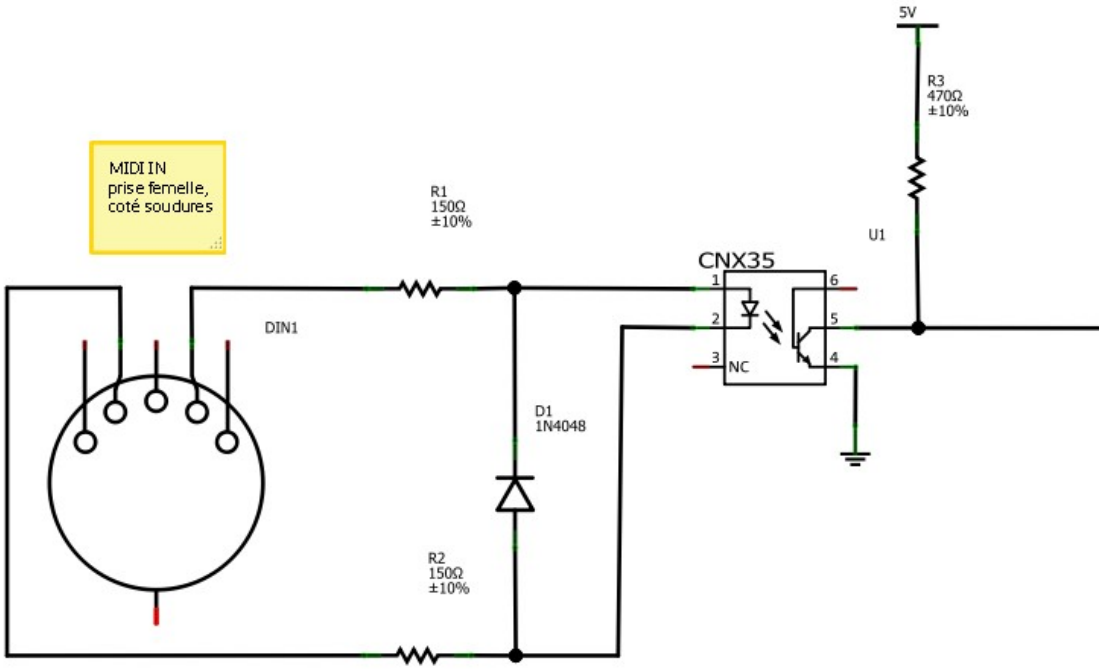
potentiomètre, utilisation des entrées analogiques



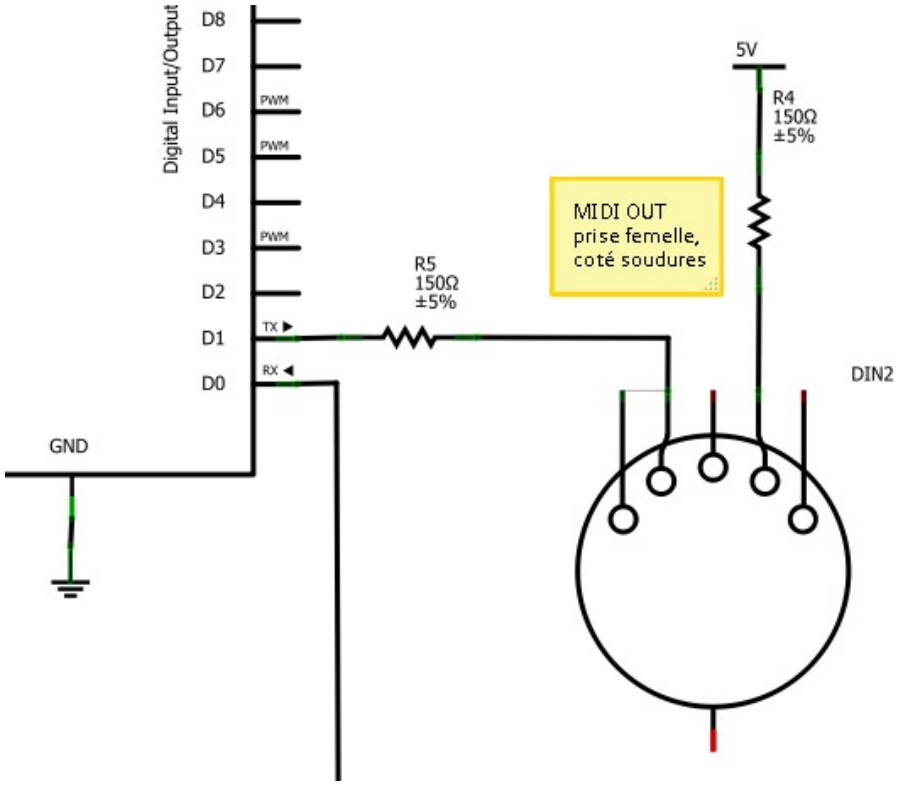
interface I2C :



Entrée midi



Sortie midi :



les liens :

http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.DebuterInstallationWindows

http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.DebuterPresentationLogiciel

http://arduino.cc/fr/Main/DebuterPresentationLogiciel

http://arduino.cc/en/Guide/Windows#toc1

sources : http://arduino.cc/