Chapitre 2

Arduino

Plan du chapitre 2

- Electronique, électricité, automatique
- L'environnement de développement pour Arduino
- Un site émulateur
- Le langage C
- Codage de circuits

Electronique et électricité : définitions

- "L'électronique est une branche de la physique appliquée, traitant de la mise en forme et de la gestion de signaux électriques, permettant de transmettre ou recevoir des informations." Elle traite plutôt les courants faibles
- source : `https://fr.wikipedia.org/wiki/%C3%89lectronique`
- "L'électricité est l'effet du déplacement de particules chargées, à l'intérieur d'un "conducteur", sous l'effet d'une différence de potentiel aux extrémités de ce conducteur"
- source : `https://fr.wikipedia.org/wiki/%C3%89lectricit%C3%A9`
- Bref électricité = le phénomène physique, électronique = la science étudiant ce phénomène

Un circuit électrique

- L'électricité est le déplacement d'électrons
- ~ un courant d'eau est un déplacement d'eau
- => courant électrique
- Un générateur d'électricité récupère les électrons qu'il a envoyés => les électrons tournent dans un circuit !
- Le point d'entrée où les électrons entrent dans le circuit est appelée la source électrique
- Le point de retour des électrons est appelé la terre

Intensité d'un courant

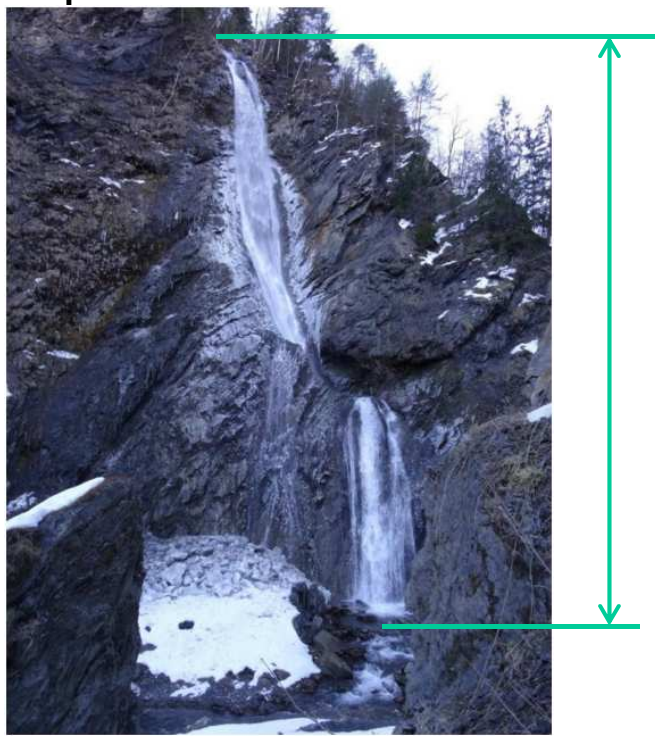
- L'intensité du courant est la "quantité d'électricité" traversant une section de circuit pendant une seconde. C'est le débit de quantité d'électricité
- Elle se mesure en ampères notés A
- ~ débit de l'eau d'un torrent, d'un fleuve

Courant continu et alternatif (DC, AC)

- Le courant continu ou CC (DC pour direct current en anglais) est un courant électrique dont l'intensité est indépendante du temps (constante)
- Le courant alternatif ou CA (AC pour alternating current en anglais) est un courant électrique périodique qui change de sens deux fois par période et qui transporte des quantités d'électricité alternativement égales dans un sens et dans l'autre. Un courant alternatif a donc une composante continue (valeur moyenne) nulle

Voltage, différence de potentiel

- ~ différence d'altitude pour un courant d'eau



- Figure : cascade du morel, aigueblanche savoie
- La différence de potentiel est mesurée en volts et est appelée aussi tension

Résistance

- ~ ce qui peut ralentir le débit de l'électricité,
  ~ un obstacle dans le courrant
- ~ ce qui résiste au courant

- Exemples et symbole :

- N'a pas de sens d'utilisation
- Est mesuré en Ohm (Ω)

Valeur d'une résistance

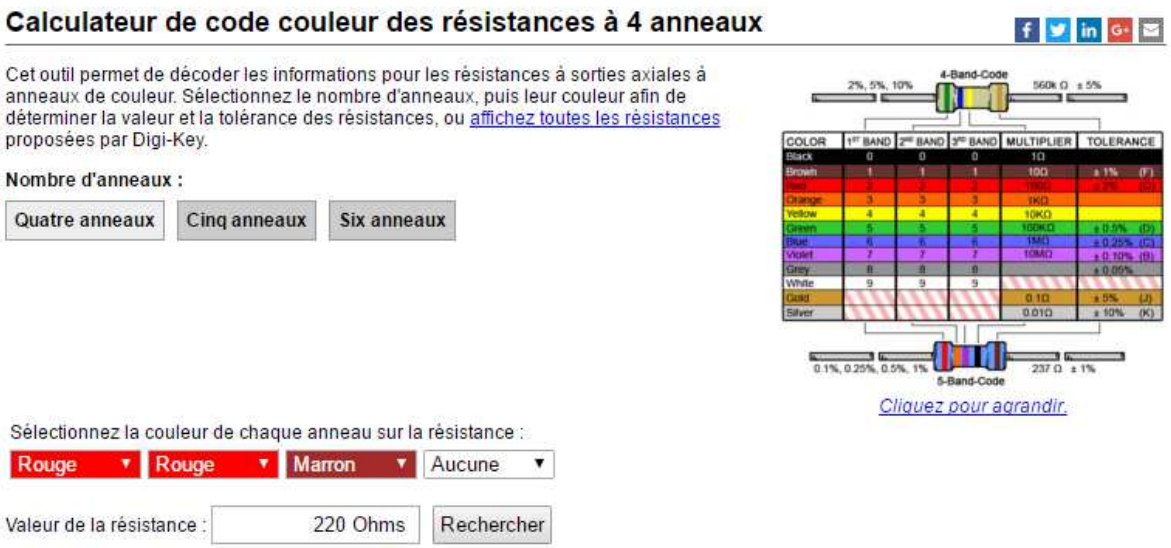
- sous la forme (mantisse, exposant)
- Plus précisément mantisse x 10^{exposant}
- Les valeurs des chiffres de 0 à 9 sont indiqués par les valeurs sous forme des couleurs :

Color	Digit value	Multiplier	Multiplied Out	Tolerance
Black	0	10^0	1	
Brown	1	10^1	10	
Red	2	10^2	100	
Orange	3	10^3	1,000	
Yellow	4	10^4	10000	
Green	5	10^5	100,000	
Blue	6	10^6	1,000,000	
Violet	7	10^7	10,000,000	
Gray	8	10^8	100,000,000	
White	9	10^9	1,000,000,000	
Gold				±5%
Silver				±10%

- Moyen mnémotechnique : "Big brown rabbits often yield great big vocal groans when gingerly snapped." (de gros lapins bruns gémissent fortement quand ils sont ???)

Pour lire une valeur de résistance

- Orienter la résistance en mettant la précision (or ou argent) à droite
- Le site `http://www.digikey.fr/fr/resources/conversion-calculators/conversion-calculator-resistor-color-code-4-band` peut être fort utile

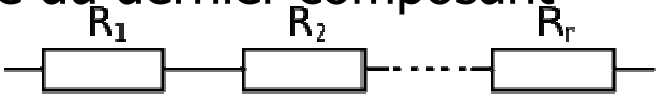


Loi d'Ohm et conséquences

- U = RI
- U en volts, R en Ohm, I en ampère
- Arduino ne doit pas dépasser 40 mA => on utilise les résistances
- Souvent pour une LED, une résistance de 10 kΩ soit au plus 0,5 mA

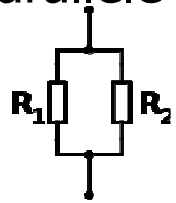
Cablage en série

- On dit que des composants électriques (résistances, LED, etc.) sont en série, si ils sont les uns à la suite des autres. Le cablage est dit en série
- Lorsque plusieurs composants sont reliés et câblés en série, si l'un d'entre eux est détruit, cela ouvre le branchement et le courant électrique ne passe plus dans ce branchement
- Dans un cablage en série, tous les composants de ce cablage sont traversés par la même intensité (en ampère)
- Dans un cablage en série, la somme des différences de potentiels (en volts) traversant chaque composant est égale à la différence de potentiel entre l'entrée du premier et la sortie du dernier composant
- Pour une connexion de résistances en série, la résistance totale est égale à la somme des résistances

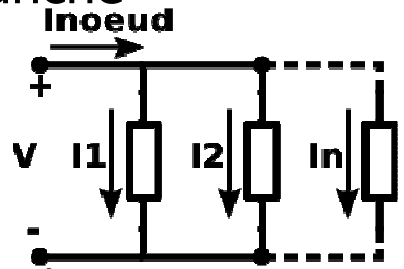


Cablage en parallèle

- On dit que des composants électriques (ou des groupes de composants) sont en parallèle, si leurs extrémités sont réunis (dans un même nœud). Le cablage est dit en parallèle
- Exemple : deux résistances en parallèle

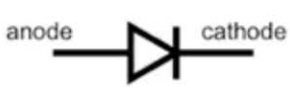


- => il existe plusieurs chemins par lesquels le courant peut passer
- Dans un branchement en parallèle, les branches sont soumises à la même tension (en volt)
- Dans un branchement en parallèle, l'intensité n'est pas obligatoirement la même dans chaque branche
- Avec n branches, on a la relation :
  $I_{noeud} = I_1 + ... + I_n$ où
  I_j est le courant qui traverse la branche j



Diode

- ~ valve unidirectionnelle
- Le courant ne passe que dans un sens pour le faible voltage d'Arduino. Il faut un voltage énorme pour faire passer le courant dans l'autre sens => une diode a un sens d'utilisation
- Symbole :



Diode

- LED = light-emitting diode = diode électroluminescente : émet de la lumière au passage du courant
- Symbole :

Light-Emitting Diode (LED)

Remarque sur les LEDs

- Les led sont des diodes (le D de LED)
- Elles sont donc orientés
- La partie liée au potentiel le plus fort (le +) correspond à la branche la plus grande : c'est l'anode
- La partie liée au potentiel le plus faible (le -), en général la terre, correspond à la branche la plus petite : c'est la cathode

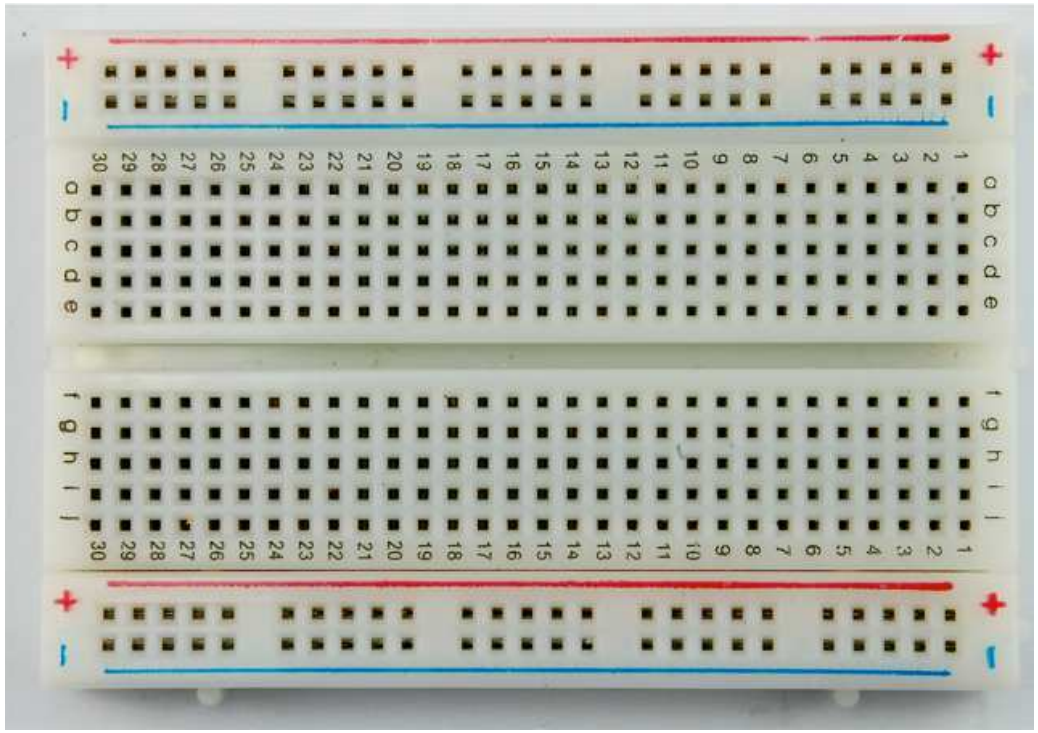
Interrupteur

⌘= Switch = Button, PushButton

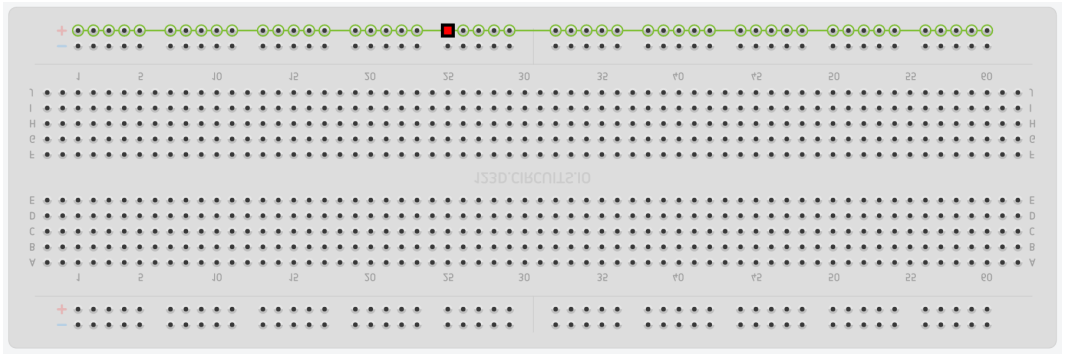
⌘Lors d'un appui sur le boutons les broches d'un même coté sont connectées. Pas les broches en face à face !

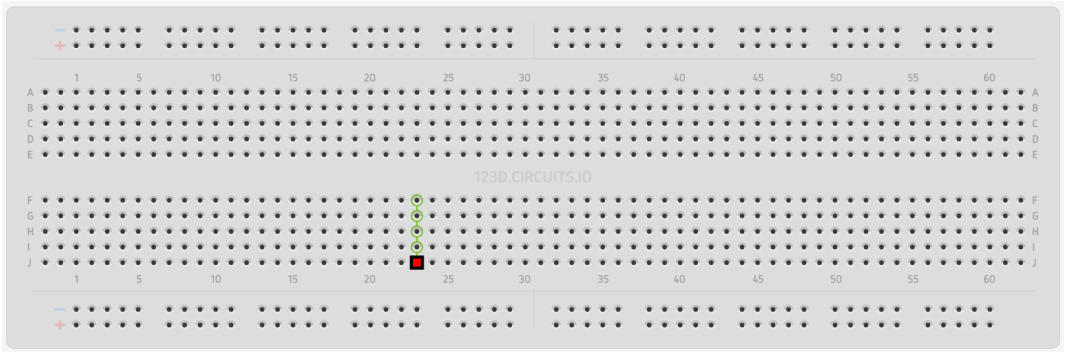
La "planche à pain" (breadboard)

- Permet de connecter des composants entre eux ...
- ... sans soudure !



Les lignes de contact du breadboard



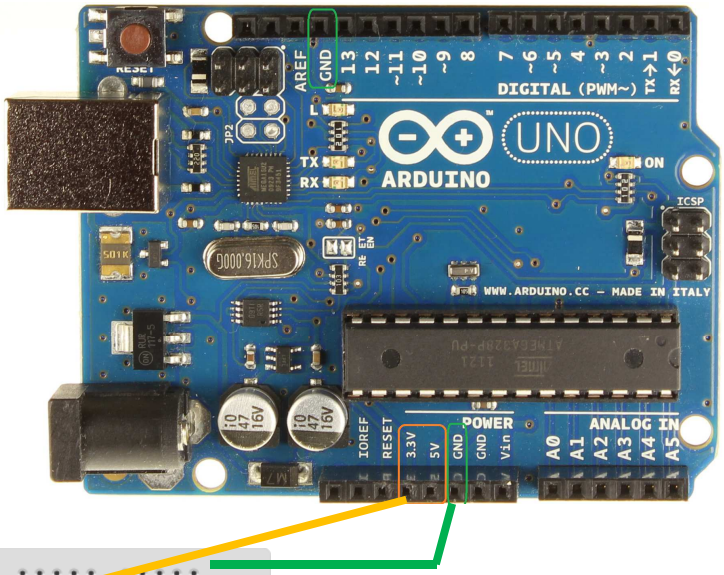


Electronique et électricité : pour commencer

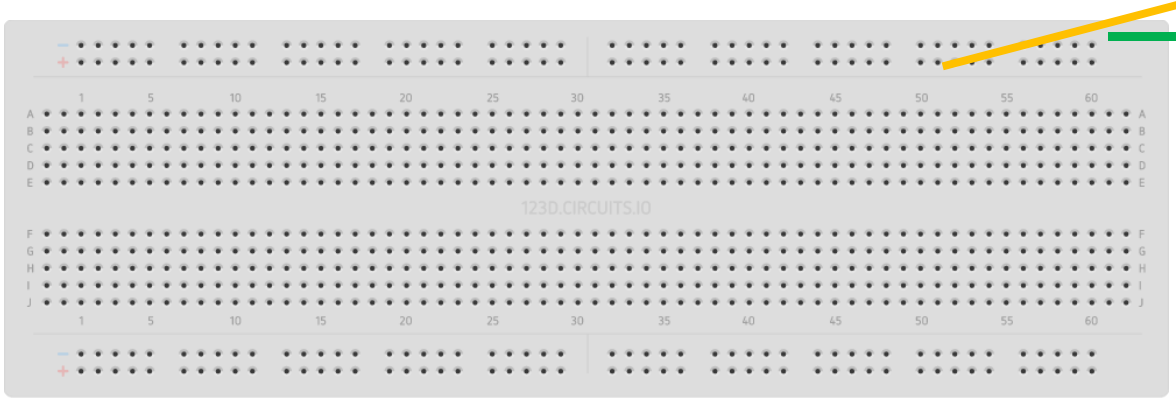
- Comment lire un schéma électrique : How to Read a Schematic: https://learn.sparkfun.com/tutorials/how-to-read-a-schematic
- La "planche à pain" = Breadboard = https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard
- Les résistances = Résistance = https://learn.sparkfun.com/tutorials/resistors
- Diode = https://learn.sparkfun.com/tutorials/diodes
- Light-emitting Diodes (LEDs) = https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds

Retour sur la carte Arduino

- Deux pins terre (en vert), deux pins de potentiel (3,3 volts et 5 volts). On utilise généralement le 5 volts



- Conseil : à relier au lignes "horizontales" de pins du breadboard

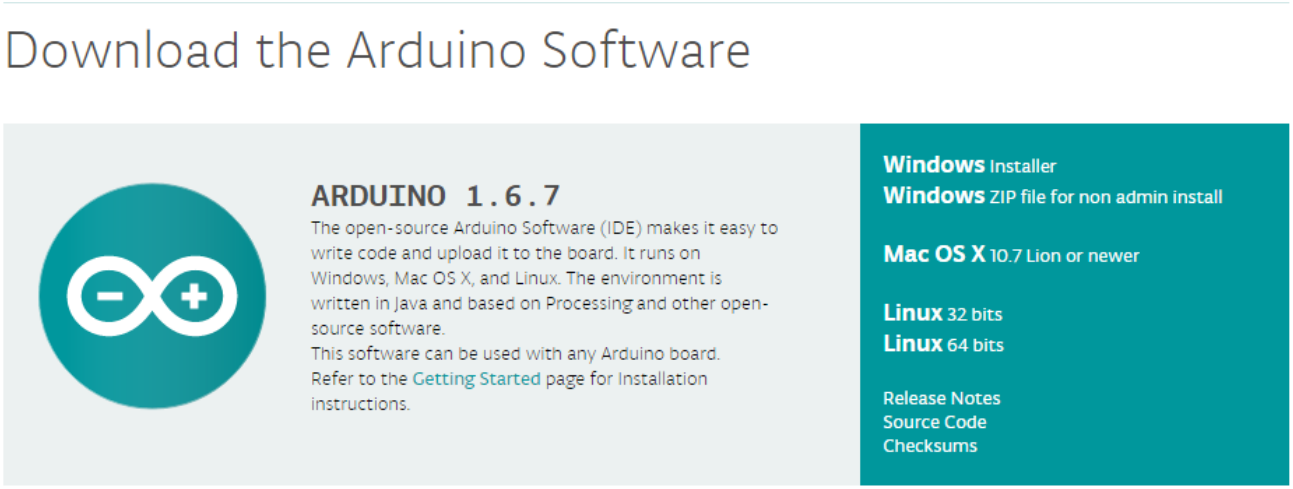


Exercice

- Que de l'électricité (ou de l'électronique)
- Une led, 2 interrupteurs, éventuellement d'autres composants

Arduino : l'environnement de développement

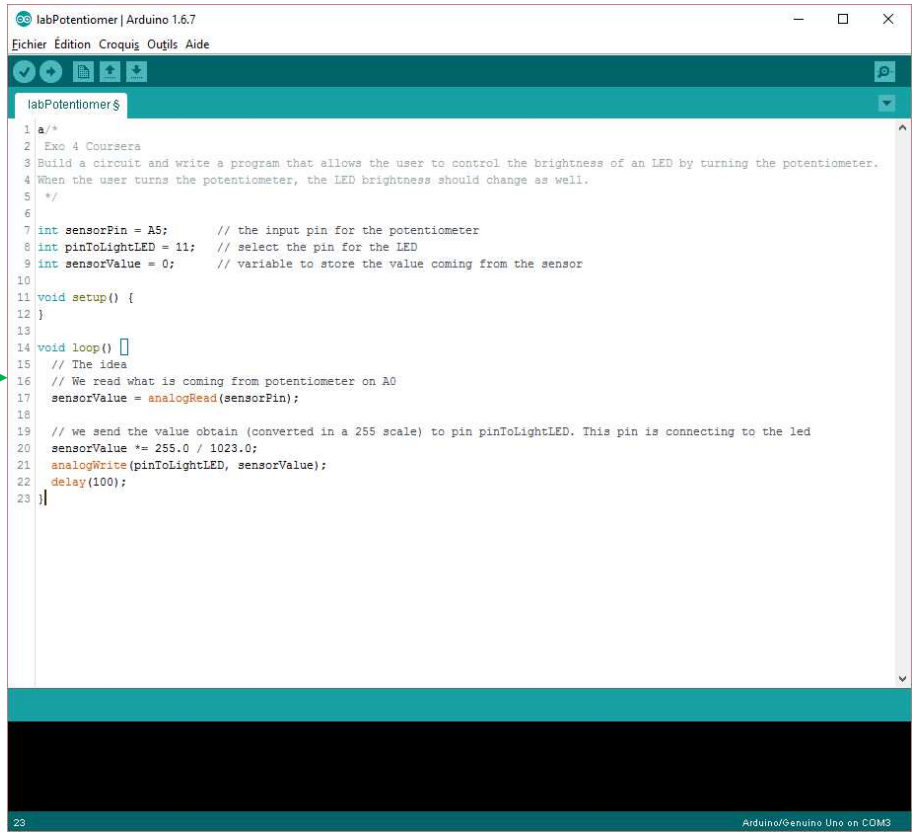
⌘Voir à `https://www.arduino.cc/en/Main/Software`



⌘Le télécharger, c'est gratuit ! et l'installer

Lancement de l'IDE Arduino

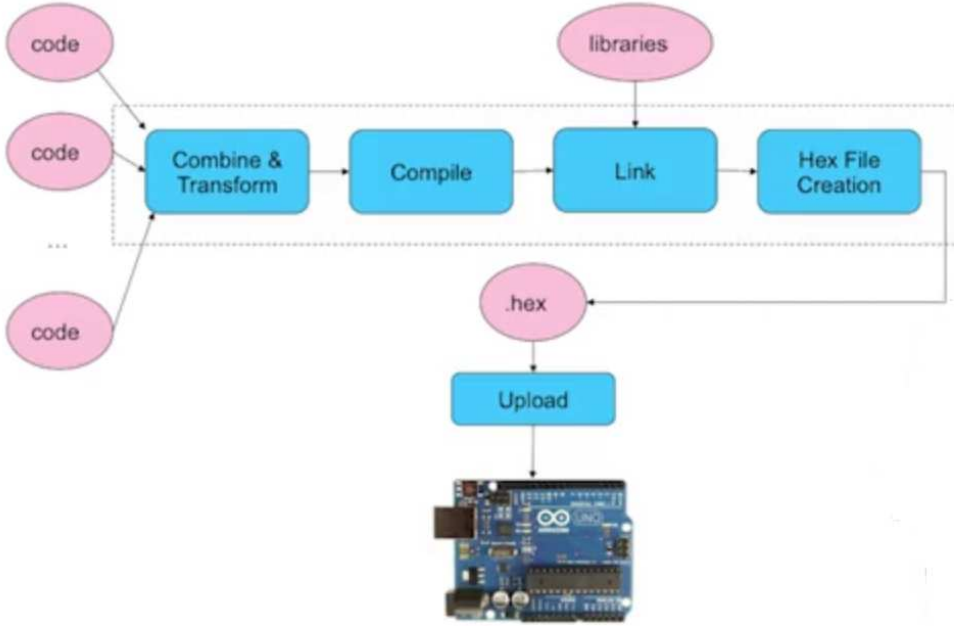
- IDE = Integrated Development Environment = Environnement de développement intégré
- Au lancement on a :
- Editeur de texte pour rédiger le programme



Arduino, matériel et logiciel : résumé

- "Your learning will be enhanced if you purchase the recommended hardware for this course. I recommend the Arduino Uno Rev 3 Ultimate Starter Kit, `http://www.vilros.com/ultimate-starter-kit.html` which costs approximately $54.99 USD. You don't need to purchase any software but you will need to download the Arduino IDE for free from `https://www.arduino.cc/`
- If you do not have an Arduino, you can use the web-based Arduino simulator at `123d.circuits.io`. You will need to create a free account. There are instructional videos on that website that will teach you how to use the simulator."
- Bref on peut avoir un émulateur de cartes et de composants pour Arduino à partir du site `123d.circuits.io`. Il suffit de se créer un compte gratuit. On peut ainsi garder (dans le cloud) ses (ces) développements

La chaine de traitement par l'IDE



- première ligne obtenu en cliquant sur Vérifier, seconde ligne en cliquant sur Téléverser
- Combine & Transform = le préprocesseur + mise en un seul fichier

Arduino : la programmation

- On programme en langage C (euh plutôt en C++)
- Voir un bon tutorial pour le langage C à partir de :
  `http://www.cprogramming.com/tutorial/c/lesson1.html`

Le langage C

- Le grand langage des années 70-80
- Mais aussi encore aujourd'hui !
- Car proche de la machine
- Un macro assembleur (accès indirect post incrémenté) structuré (boucles, tests, fonctions, ...)
- Très utilisé dorénavant pour l'informatique embarquée (microcontrôleurs), les jeux, pour les calculs et algorithmes performants, les systèmes d'exploitation, bref modules où la rapidité de traitement est importante
- La syntaxe du langage C a inspiré énormément de langages de programmation : C++, Java, JavaScript, PHP, C#, ...
- Le langage C est un des langages les plus utilisés
- source : `http://fr.wikipedia.org/wiki/Langage_C`

Un premier programme (1/2)

```c
#include <stdio.h>

int main(void)
{
  printf("Bon courage\n");
  return 0;
}
```

- `#include <stdio.h>` inclut l'en-tête standard `<stdio.h>`, contenant les déclarations des fonctions d'entrée-sortie de la bibliothèque standard de C (entre autre la fonction `printf` utilisée)
- `main` est le nom de la fonction principale, aussi appelée point d'entrée du programme
- `int` est le type renvoyé par la fonction main
- Le mot clé `void` entre les parenthèses signifie que la fonction `main` ne prend aucun paramètre

Un premier programme (2/2)

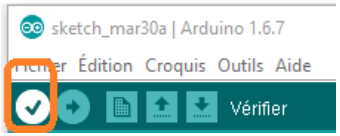
```c
#include <stdio.h>

int main(void)
{
  printf("Bon courage\n");
  return 0;
}
```

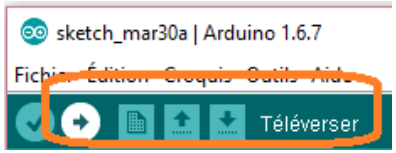
- Les accolades `{` et `}` entourent les instructions constituant le corps de la fonction `main`
- `printf` est une fonction d'écriture dans la sortie standard (la console par défaut)
- Le caractère `"` délimite une chaîne de caractères. Dans ce cas, c'est la chaîne à afficher
- Un point-virgule `;` termine toute instruction (symbole de fin d'instruction)
- L'instruction `return 0;` indique que la fonction `main` retourne la valeur `0` (non utilisé ici)

Compilation et exécution

- Le langage C est un langage compilé
- Après avoir édité le programme précédent, il faut le compiler = le traduire en langage exécutable
- Il existe de nombreux compilateurs ... gratuits comme `gcc`
- Si le fichier source est sauvegardé dans `courage.c`, la compilation est : `gcc courage.c`
- Il est alors créé un fichier exécutable. Avec Arduino, l'environnement de développement lance la compilation par le bouton Vérifier :



- Pour exécuter, il faut téléverser sur la carte Arduino : bouton Téléverser



La compilation pour Arduino

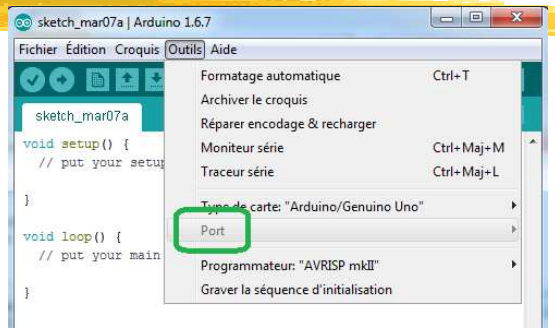
- Le compilateur est `avr-gcc`. Le résultat sera exécuté sur une plateforme AVR pas Intel
- Le code généré est un `.hex`
- Les programmes Arduino sont appelés des croquis (sketchs). Pourquoi ? Je ne sais pas !
- Ils ont écrits en C++ même si on utilise essentiellement du C

Un premier exemple : `Blink`

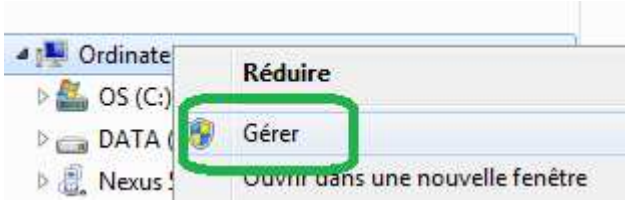
- Charger, dans l'IDE Arduino, l'exemple `Blink` à partir `%REP_INSTALL_ARDUINO%\examples\01.Basics\Blink`
- Compiler cet exemple
- Faire le branchements de la carte Arduino avec l'ordinateur avec le cable USB
- Téléverser ce croquis
- La "led 13" devient un clignotant !
- Remarque : `Blink` est le "Hello, world" des systèmes embarqués

Arduino : si pb lors du téléversement (1/2)

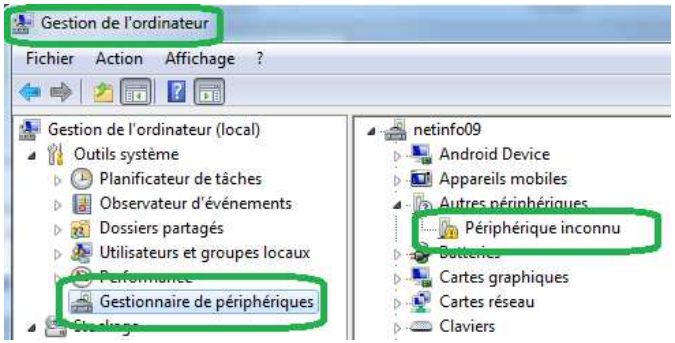
- Si pb car l'item port est grisé, il faut installer le pilote Arduino



- Pour cela, clic droit sur icône Ordinateur,



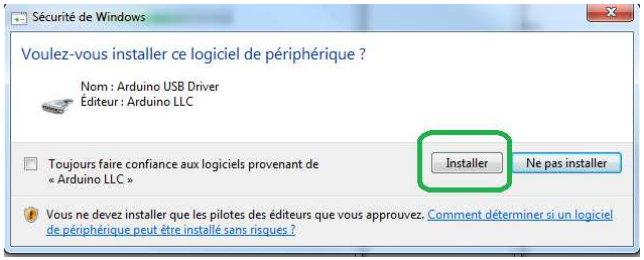
- puis Gérer | Gestionnaire de périphériques



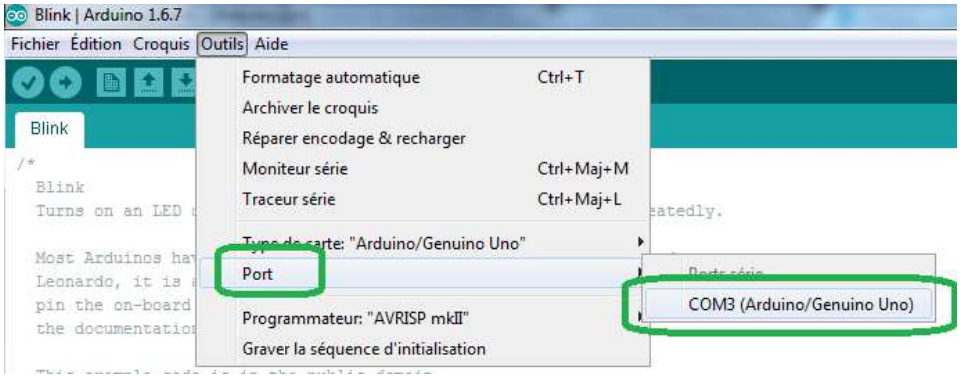
- Clic droit sur le périphérique fautif item "Mettre à jour le pilote..."

Arduino : si pb lors du téléversement (2/2)

- Rechercher le pilote sous le répertoire d'installation Arduino. Cliquer Installer



- L'item port n'est plus grisé. Il propose un menu où apparaît la carte Arduino : COM3 (Arduino/Gebuino Uno) sous windows



- biblio : https://www.youtube.com/watch?v=CdE72XUYC7k

Structure d'un croquis (sketch)

- Un croquis n'a pas de ... `main()` !
- Le `main()` est déjà codé dans le micro-controlleur. Il lance certaines fonctions que le programmeur a codées
- En fait, le microcontrôleur possède déjà le programme :

```
int main() {
     setup();
     while (1) {
          loop();
     }
}
```

  et ajoute notre code
- Dans certains microcontrôleurs (mais pas pour Arduino), il faut écrire ce code

Structure d'un croquis (sketch)

- En général, le programmeur écrit la fonction `void setup(){...}`
- `setup()` est lancée une et une seule fois au début du programme, euh du croquis ;-)
- En général, le programmeur écrit la fonction `void loop(){...}`
- `loop()` est exécutée après `setup()`. Ce qui est écrit dans `loop()` est exécuté en … boucle (eh oui) infinie. Cela peut paraître currieux mais est naturel (obligatoire ?) dans les systèmes embarqués
- Ces deux fonctions ne retournent rien (`void`) et n'ont pas d'arguments

Le code de Blink

- Le code du programme Blink est :

```
void setup() {
     pinMode(13, OUTPUT);
}
void loop() {
     digitalWrite(13, HIGH);
     delay(1000);
     digitalWrite(13, LOW);
     delay(1000);
}
```

- Dans `setup()` est indiqué que la pin 13 va fournir des tensions électriques. Derrière cette pin 13, il y a la "diode 13" de la carte
- Le microcontrôleur envoie (écrit) la tension maximale (`HIGH`) sur cette diode (`digitalWrite(13, HIGH);`)
- On attend 1 seconde (`delay(1000);`)
- Le microcontrôleur envoie (écrit) la tension minimale (`LOW`) sur cette diode (`digitalWrite(13, LOW);`) = éteint la diode
- On attend 1 seconde (`delay(1000);`)
- Ces quatre instructions seront exécutées en boucle (fonction `loop()`)

Exercice

- Programmation de premiers circuits

Fin