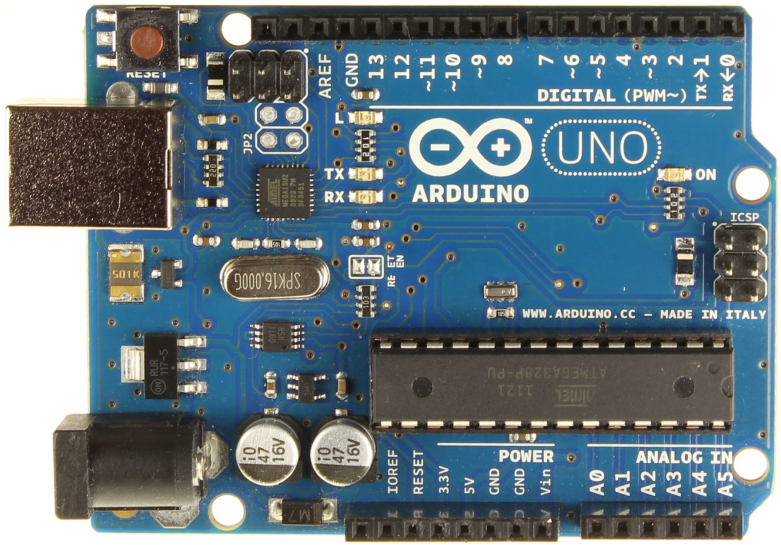
Introduction à l'Arduino

L'arduino UNO

L'arduino UNO est un module/carte de développement (*dev board*) avec un microcontrôleur Atmel ATMEGA328P, grand public, peu onéreux, qui connaît un grand succès (= grande communauté)



Microcontrôleur

Microcontrôleur: (=MCU) Sorte de mini-ordinateur, tenant sur une "puce électronique" (=circuit intégré)

Pratiquement tout ce qui est nécessaire pour fonctionner est dans la puce.

Comme un ordinateur, il est programmable et polyvalent.

La puce contient, notamment:

- un CPU (Central Processing Unit) = Processeur (équivalent sur votre ordinateur : le processeur Intel Core i7...)
- de la mémoire RAM (équivalent sur votre ordinateur : les barrettes de RAM DDR3)
- de la mémoire ROM (équivalent sur votre ordinateur : disque dur ou SSD)
- des entrées-sorties

Arduino / Atmel

Arduino: entreprise Italienne qui conçoit des cartes de développement grand public, avec un environnement de développement facilement utilisable.

l'Arduino UNO n'est pas son seul produit. Elle propose d'autres cartes, basés sur des microcontrôleur Atmel ou autres.

Atmel: Fabriquant de composant électroniques, spécialisé dans les microcontrôleur.

D'autres fabriquants existent

Les microcontrôleur ont des caractéristiques qui peuvent beaucoup différer

ATMEGA328

L'ATMEGA328 est un microcontrôleur d'Atmel.
Ses caractéristiques:

- 8 bits
- 20 Mhz = 20 million d'instructions à la seconde
- 32Ko de mémoire "Flash", pour le programme
- 2Ko de RAM, pour les données
- 1Ko de EEPROM, pour les données qui doivent être conservées

Ses entrées-sorties:

- 23 entrées/sorties (binaires) programmables
- 6 entrés analogiques (ADC)
- 6 sorties "PWM" (Pulse Width Modulation)
- Bus: SPI, I2C, série...

Parenthèse: les mémoires

RAM = Random Access Memory

- = mémoire “vive”
- mémoire “volatile”: quand le circuit n’est plus alimenté, les données sont perdues
- mémoire en lecture et écriture
- accès rapide

ROM = Read Only Memory

- = mémoire “morte”
- mémoire non volatile: les données subsistent même si le circuit n’est plus alimenté
- mémoire en lecture seule
- l’écriture est (des fois) possible, mais plus longue et complexe

EEPROM = Electrically Erasable Programmable ROM

- Mémoire ROM qui peut être effacée et reprogrammée électriquement

“Flash” = un certain type de mémoire EEPROM.

L'Arduino UNO

L'arduino UNO est un ATMEGA328, avec des composants autour pour nous simplifier la vie:

- un oscillateur à 16Mhz (un "quartz")
- un régulateur de tension : l'ATMEGA doit être alimenté en 5V, l'arduino peut être alimenté entre 6V et 20V
- une interface USB pour :
- - pouvoir programmer l'ATMEGA via l'USB
- - pouvoir communiquer avec l'ATMEGA pendant qu'il fonctionne (via le bus série)
- des connecteurs pour facilement brancher des fils

Microcontrôleur : à quoi ça sert ?

Les microprocesseurs/microcontrôleur sont présents partout

Pour une utilisation polyvalente: dans les ordinateurs et smartphones

Pour une utilisation spécifique/dédiée: dans les télés, les machines à laver, les téléphones...
(Il est souvent plus facile et moins cher de programmer un microcontrôleur pour faire une tâche spécifique que de la “câbler”)

Pour les bidouilleurs: faire des objets connectés, imprimantes 3D, robots, drones...

Pourquoi utiliser Arduino ?

Utiliser une carte de développement plutôt qu'un composant nu:

- les composants périphériques nécessaires sont déjà présent
- pas besoin de souder

Pourquoi Arduino ?:

- C'est pas cher (20€pour l'original, 3€pour la version chinoise)
- l'ATMEGA est assez complet, facile à utiliser et plutôt "solide"
- Grosse communauté
- Beaucoup de librairies pour interfacer avec d'autres modules

Mais d'autres microcontrôleur sont aussi intéressant. Par exemple le NodeMCU (ESP8266) (2€, avec gestion du WIFI)

Microcontrôleur: comment on s'en sert ?

Il faut, en premier lieu, "programmer" le microcontrôleur, pour qu'il fasse ce qu'on veut

Le CPU du microcontrôleur interprète le programme présent dans la mémoire flash.

Le programme est en "langage machine" (du binaire). Le langage machine dépend du microcontrôleur. Chaque famille de microcontrôleur a son propre code machine.

Coder en langage machine (ou en "assembleur") est fastidieux: long, lent et beaucoup d'erreurs possibles...

Microcontrôleur: comment on s'en sert ?

Heureusement, on peut les programmer en C ou d'autres langages de "haut niveau", via une interface

Plusieurs étapes:

- créer le programme (par exemple en C), via un éditeur de texte
- compiler le programme: c'est à dire traduire le C en langage machine
- téléverser ("uploader") le programme sur le microcontrôleur.

Sur l'arduino, tout cela peut se faire via une interface.

L'interface Arduino

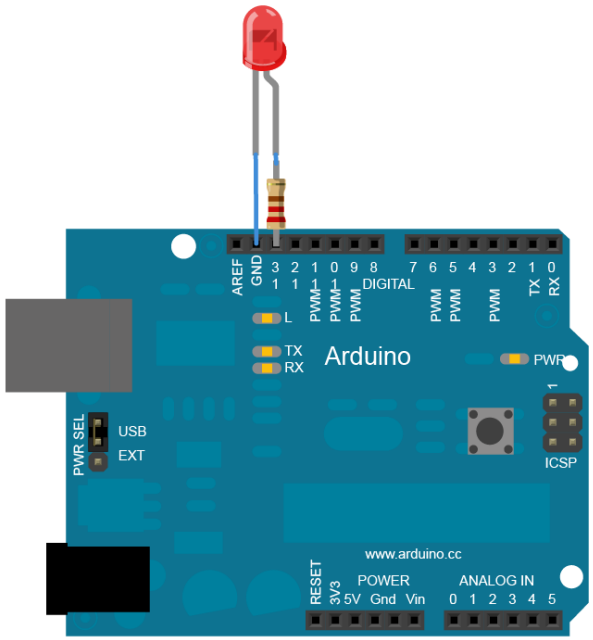
Pratique

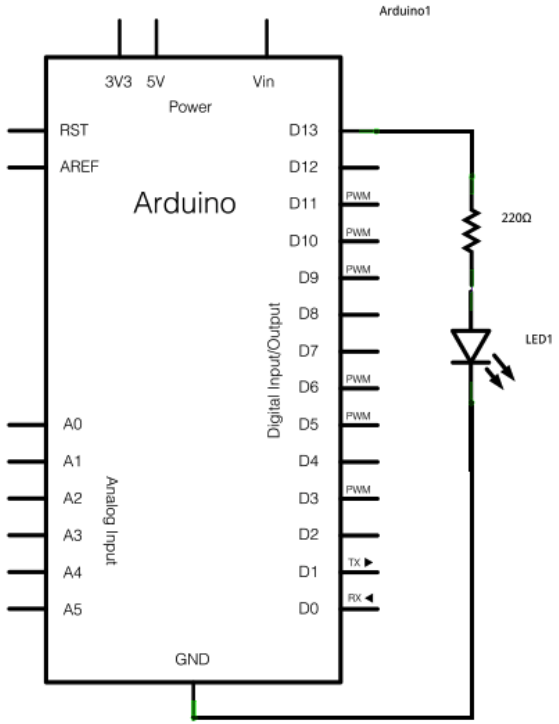
Avant de commencer:

- Certains composants électroniques sont fragiles
- Notamment ils n'aiment pas : l'électricité statique, être branchés à l'envers ou recevoir plus de tension (volts) que ce qui est prévu
- Évitez de toucher les pattes des circuits intégrés
- Revérifiez que tout est bien branché avant d'alimenter le circuit

Pratique: faire clignoter une LED

- (Installez et) Lancez l'interface Arduino
- Créez un nouveau programme ("sketch")
- Entrez le code donné
- Branchez la LED sur l'Arduino (en série avec la résistance) comme indiqué
- Branchez l'Arduino au PC par l'USB
- Téléversez, et la LED devrait clignoter
- Opt.: Modifiez le programme pour faire d'autres effets






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

Pratique: aller plus loin

- À quoi sert pinMode() ? digitalWrite() ? delay() ?
- Comment faire clignoter une 2eme LED ?
- Comment “lire” une entrée ?
- À quoi sert la résistance ? que pourrait-il se passer si on l'oublie ?