# Formation IoT

PAR ALAIN CARIOU, MAI 2019

# Introduction : petit brainstorming

- **Pour vous, qui signifie l'IoT, l'Internet of Things ?**
- **A quoi est-ce que cela fait référence ?**
- **A quoi est-ce que cela vous fait penser ?**

# I – Introduction aux systèmes embarqués

# Qu’est-ce qu’un système embarqué ?

- Un système embarqué (***Embedded system***) est un système électronique et informatique autonome qui effectue une tâche précise. Ce terme qualifie aussi bien le matériel que le logiciel

- Il doit faire face à des contraintes d’espace (la taille du système et la mémoire auquel il a accès), d’énergie et de fiabilité

- Ils sont utilisés dans de nombreux domaines : l’astronautique, les automates, le secteur militaire, le multimédia, les télécommunications, les transports, le secteur médical, l’informatique, et dans l’IoT !

# Définition d’un microcontrôleur

- Il s’agit d’un circuit intégré composé de plusieurs éléments :
  - un processeur
  - des mémoires
  - des interfaces entrées-sorties

- Ils se caractérisent par une plus faible consommation d’énergie et un coût moins important par rapport aux microprocesseurs classiques
- De ce fait ils sont très utilisés dans les systèmes embarqués

# Les microcontrôleurs

- Il existe différentes familles de microcontrôleurs, chacune avec ses propres particularités :
  - différences d'architecture, de langage supporté, de jeu d'instructions, etc

- Ils sont souvent intégrés à une carte électronique qui permet ensuite d'y connecter d'autres composants

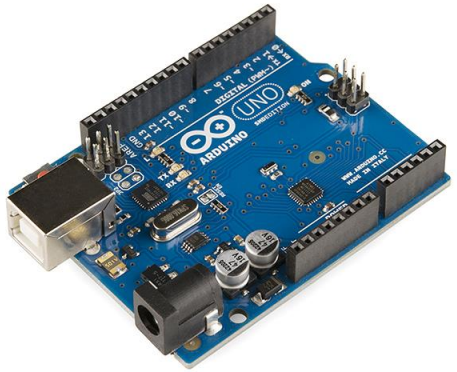
# II – L’Arduino et ses dérivés

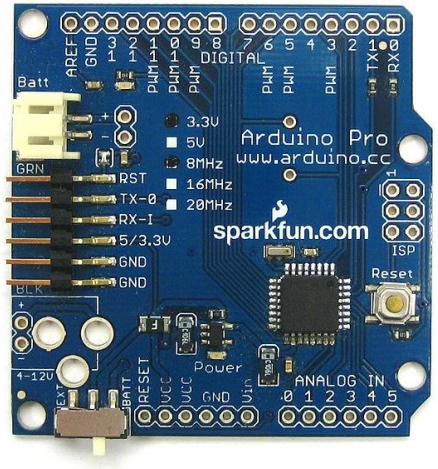
# Qu’est-ce qu’un Arduino ?

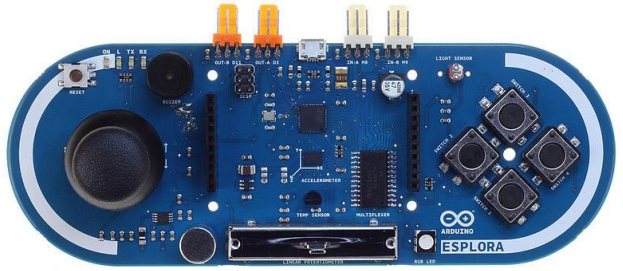
- Un Arduino est une carte électronique composée d’un microcontrôleur, différents ports, des entrées-sorties numériques et analogiques
- Il existe 17 versions des cartes Arduino, chacune ayant ses particularités
- Elles sont développées par la société Arduino.cc
- De nombreuses autres cartes sont inspirées du format Arduino et présentes des architectures similaires et compatibles

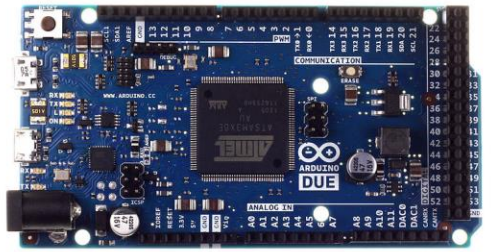
# Les différents types d'Arduino

- Quelques exemples avec, de gauche à droite : l'Arduino Uno, l'Arduino Pro, l'Arduino Esplora et l'Arduino Due



# La structure d’un Arduino

- On remarque plusieurs éléments notables :
  - les prises USB et jack
  - le microcontrôleur
  - le bouton de reset (en rouge)
  - l’horloge interne
  - les pins numériques et analogiques
  - les indicateurs LED
  - une alimentation de 5V est nécessaire

# Petit rappel sur l'électronique

- Faites attention au sens du courant et à la polarité de certains composants (comme les LED)
- Débranchez votre carte avant de manipuler ses composants électroniques
- Faites attention à la tension que peuvent recevoir vos composants : ajoutez des résistances adéquates pour contrôler cela, utilisez un ampèremètre si besoin
- Ne posez pas votre carte sur une surface conductrice

# Développer sur Arduino

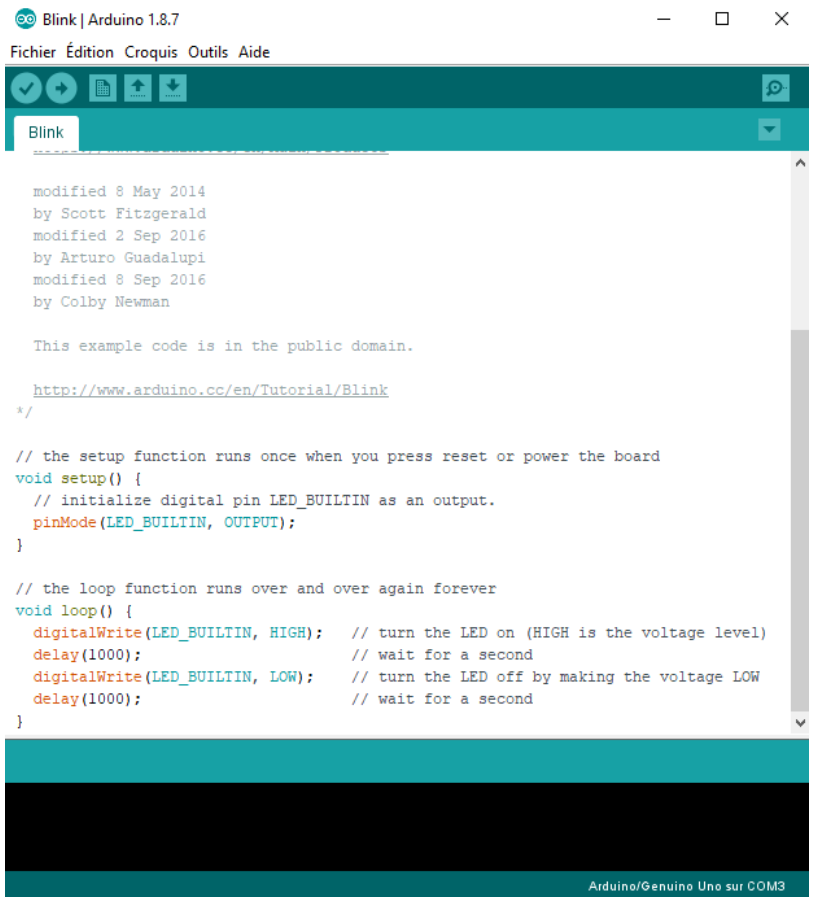
- Installation de l'IDE :
- https://www.arduino.cc/en/Main/Software

- Vous pouvez aussi utiliser d'autres IDE comme Code::Block ou Eclipse

- Lors de l'installation vous aurez sûrement besoin d'installer plusieurs drivers pour Arduino, acceptez tout

# Présentation de l'IDE

- L'IDE d'Arduino est très simple d'utilisation
- Il est nécessaire d'indiquer le type de carte utilisé
- Ainsi que le port auquel est connecté votre Arduino
- Enfin vous disposez de nombreux sketchs d'exemple



# Une première application

- Nous allons ouvrir un des sketchs d'exemple dans 01.Basics – Blink
- Connectez votre Arduino à l'ordinateur et téléversez le sketch
- Vous remarquerez que l'IDE vous informe de l'espace utilisé par votre programme sur l'Arduino

Faites très attention à l'espace utilisé par votre programme !

# Les fonctions spécifiques à Arduino - 1

- Comme vous l'avez remarqué dans ce premier sketch, on utilise plutôt du C/C++ pour développer sur Arduino

- Il y a deux fonctions principales qui existent de base :
  - la fonction **setup()** dans laquelle vous devez initialiser les éléments de votre programme
  - et la fonction **loop()** qui va exécuter en boucle les instructions se trouvent à l'intérieur

- A cela s'ajoute les fonctions standards ainsi que plusieurs librairies

# Les fonctions spécifiques à Arduino - 2

- **delay(ms)** permet de stopper le programme pour le nombre de millisecondes passé en paramètre

- **digitalRead(pin)** retourne la valeur digitale (**HIGH** ou **LOW**) de la pin passée en paramètre
- **digitalWrite(pin, value)** écrit la valeur **value** (**HIGH** ou **LOW**) sur la pin indiquée

- **pinMode(pin, mode)** permet de configurer la pin en paramètre au mode **INPUT** ou **OUTPUT**

# Les fonctions spécifiques à Arduino - 3

- **analogRead(pin)** lit la valeur de la pin analogique. La valeur lue sera entre **0** et **1023**
- **analogWrite(pin, value)** écrit une valeur analogique sur la pin. La valeur doit être comprise entre **0** et **255**

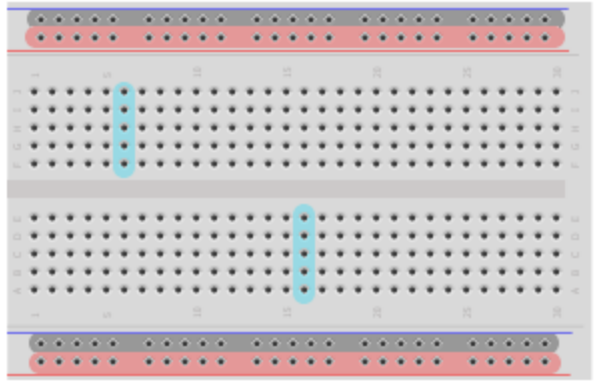
- **tone(pin, frequency, [duration])** permet de générer un son sur une pin en indiquant sa fréquence et, facultativement, pour une durée donnée. Utile pour certains buzzers.
- **noTone()** stop la génération d'un son

# Les fonctions spécifiques à Arduino - 4

- Serial est une objet qui contient de nombreuses fonctions pour communiquer en série avec l'ordinateur. Parmi les plus simples :
  - **Serial.begin(value)** initialise le nombre de données par seconde pour les communications en série, aussi appelé baud
  - **Serial.print(value, [format])** écrit une valeur sur le port en série
  - **Serial.println(value, [format])** écrit une valeur sur le port en série avant de faire un retour à la ligne
- Documentation :
  - https://www.arduino.cc/reference/en/

# Se connecter à d’autres composants

- Néanmoins vous avez peut-être envie de faire plus de choses avec votre Arduino
- Pour cela il faudra le connecter à d’autres composants grâce à une **breadboard** ou **plaque d’essai sans soudure**
- Pour rappel les lignes verticales sont reliées entre elles



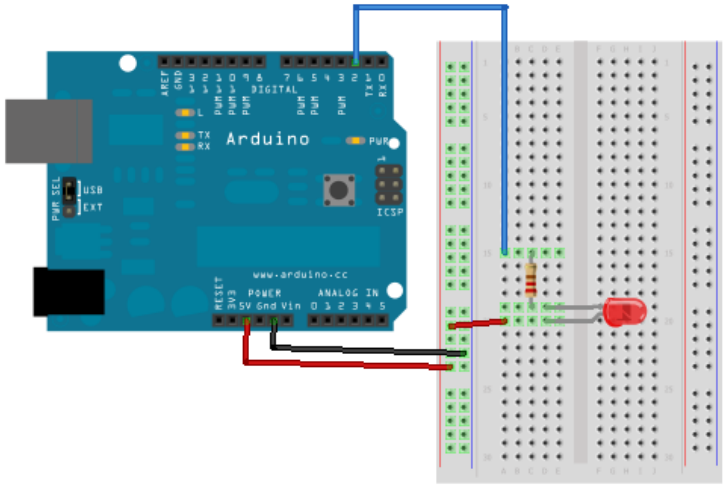
# Pensez aux résistances : la loi d'Ohm

- Cette loi établit la relation entre la valeur d'une résistance, la tension reçue et l'intensité du courant qui circule

- Avec **U** pour la tension (en volt) aux bornes de la résistance,
- **I** pour l'intensité du courant (en ampère),
- **R** pour la valeur de la résistance (en Ohm) telle que :
  - **U = R x I**

# Premier exercice :

- Réalisez un programme qui fait clignoter une LED externe.



# Les entrées et sorties digitales

- Pour utiliser les entrées et sorties digitales, il faut utiliser les pins digitales de votre carte

- A chaque pin utilisée, vous devrez indiquer son rôle : si elle est une sortie ou une entrée via la fonction **pinMode()**

- Une pin digitale envoie et reçoit des valeurs binaires : 0 ou 1. Pour cela, on utilise les mot-clés **HIGH** et **LOW**

# La communication en série - 1

- La communication avec un ordinateur se fait via la voie série. Le port série est émulée à travers l'USB

- L'IDE d'Arduino nous permet simplement d'ouvrir un terminal série sur l'ordinateur et de déterminer sa vitesse en **bauds** (le nombre de symbole - 8 bits - envoyé par seconde)

- N'oubliez pas de démarrer la liaison dans le **setup** via la fonction **Serial.begin()**

# La communication en série - 2

- Ensuite la communication peut fonctionner dans les deux sens :
  - on peut lire des caractères via **Serial.read()**
  - écrire des données via **Serial.write()**, **Serial.print()** et **Serial.println()**
  - vérifier s'il y a des données à lire via **Serial.available()**
  - et terminer une communication via **Serial.end()**

- Pour en savoir plus :
  - https://www.arduino.cc/reference/en/language/functions/communication/serial/

# Les entrées et sorties analogiques - 1

- Avec les signaux logiques, nous sommes obligés de capter des valeurs binaires. Cependant une valeur analogique pourra prendre une infinité de valeurs selon un intervalle précis. Par exemple entre 0 et 5V

- Arduino nous permet de convertir ces grandeurs analogiques en grandeurs numériques

- Pour cela on utilise la fonction **analogRead()** qui retourne une valeur entre **0** et **1023**
- Pour la convertir en voltage il suffit de faire : **voltage = valeur x 5 (tension de la carte) / 1023**

# Les entrées et sorties analogiques - 2

- Pour envoyez des données analogiques, on utilise la **Pulse Width Modulation** (**Modulation à largeur d'impulsion**), un signal numérique dont on contrôle le rapport cyclique

- Ce signal peut être généré à partir des pins ayant un symbole **tilde ~**

- On utilise la fonction **analogWrite()** pour transformer ce signal numérique en pseudo signal analogique en lui donnant une valeur entre **0 et 255** qui représente le rapport cyclique

- En variant ce rapport cyclique, on peut par exemple faire varier la luminosité d'une LED

# Les capteurs

- Ces signaux analogiques vont être utiles lorsque l'on va utiliser des capteurs. En effet on va recevoir des valeurs variées suivant l'intensité des données reçues

- Il existe de très nombreux capteurs : humidité, température, luminosité, infrarouge, mouvement, etc

- La difficulté ici sera de déterminer leurs propriétés afin de pouvoir les utiliser !

# Exercices et application du cours - 1

- Vous êtes assez libres de réaliser les montages que vous souhaitez, néanmoins voici quelques pistes si vous avez besoin :
- Les LED :
  - Faire clignoter une LED
  - Faire clignoter deux LED l'une après l'autre
  - Faire clignoter un ensemble de LED les unes après les autres
  - Contrôler l'allumage et le clignotement de plusieurs LED via des commandes de son ordinateur (pouvoir allumer toutes les LED, seulement les rouge, etc)
  - gérer l'allumage des LED via un bouton

# Exercices et application du cours - 2

- Les capteurs :
  - Avoir un capteur de luminosité qui allume une LED s'il faut nuit
  - Récupérer la mesure de la température via un capteur de température
  - Gérer l'allumage d'un buzzeur
  - Utiliser un détecteur de mouvement qui active un buzzeur lorsqu'un mouvement est détecté
  - Récupérer le taux d'humidité via un capteur d'humidité

# Exercices et application du cours - 3

- Des éléments plus complexes :
  - activer un servo-moteur
  - récupérer les valeurs d'une télécommande via un capteur infrarouge
  - récupérer les valeurs d'un joystick
  - afficher des données sur un écran LCD

- Bref, essayez d'utiliser les composants qui sont à votre disposition !