Systèmes embarqués

Introduction à la programmation microcontrôleurs
et à l'impression 3D.

Pape Abdoulaye BARRO, Ph.D,

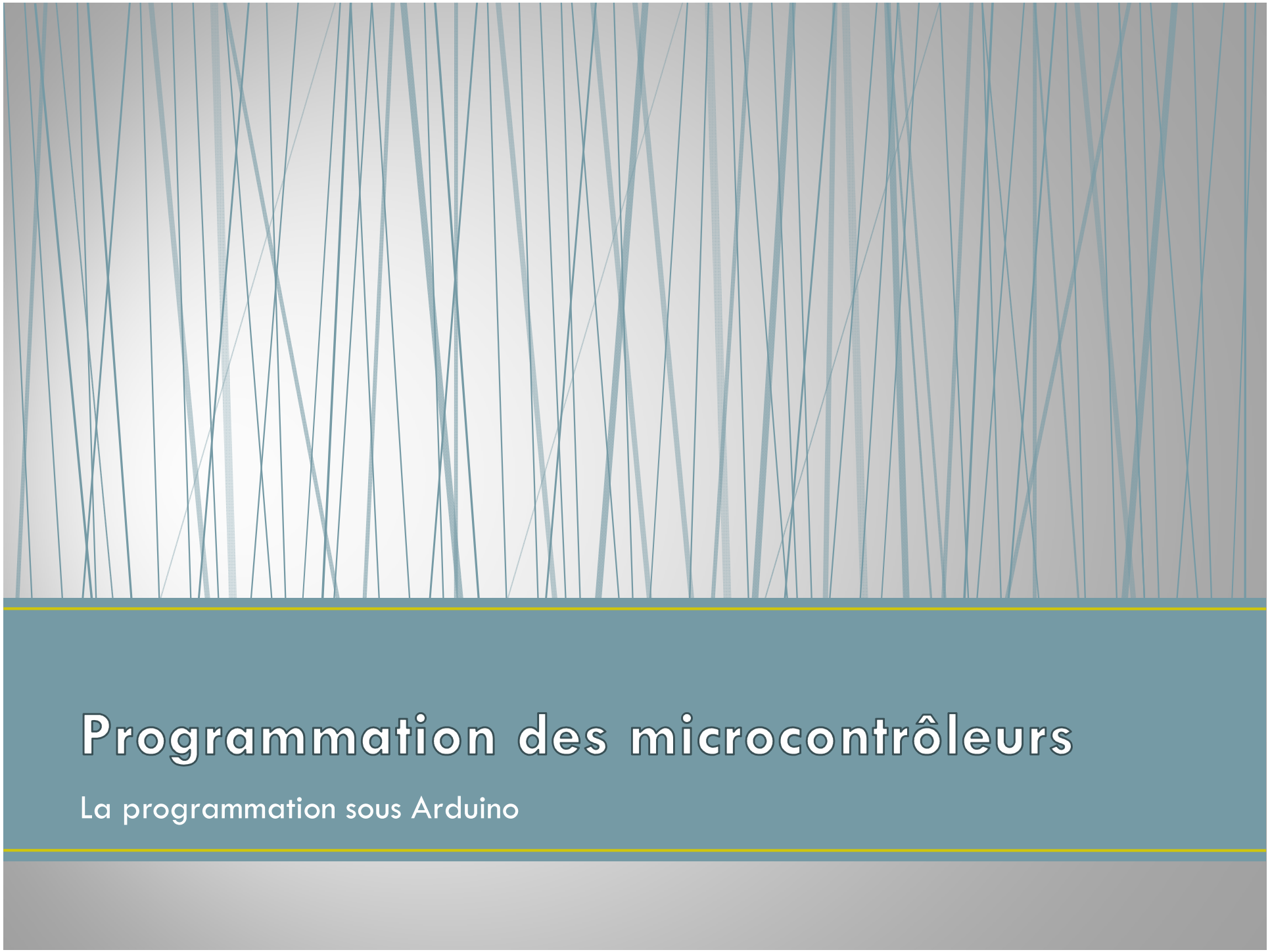
Enseignant-chercheur
UFR des Sciences et technologies
Département Informatique

E-LabTP, Laboratoire des TP à Distance, UFR-SET,
Marconi-Lab, Laboratoire de Télécommunications, ICTP, Italie

Email: pape.abdoulaye.barro@gmail.com

PLAN

- Généralités
- Architecture et familles de microcontrôleur
- Capteurs et actionneurs
- **Programmation des microcontrôleurs**
  - Programmation sous Arduino
    - Etudes de cas
  - Programmation sous Raspberry Pi
    - Etudes de cas
- Initiation à l'impression 3D



Programmation des microcontrôleurs

La programmation sous Arduino

Capteurs, actionneurs et programmation des microcontrôleurs

Capteurs, actionneurs et programmation des microcontrôleurs

Tout microcontrôleur est appelé à être programmé car étant sous le contrôle d'un programme stocké. Dans ce cas, il suffit de connaitre le langage dans lequel il est implémenté, de savoir manipuler les bibliothèques qu'il propose et puis d'être familier avec IDE (Integrated Development Environment) utilisé pour la programmation. Dans cette section, nous serons intéressé par la programmation sous Arduino et sur Raspberry.

I. La programmation sous Arduino

Arduino est un microcontrôleur open-source qui permet la programmation et l'interaction. Il est basé sur C/C++ avec une bibliothèque Arduino pour lui permettre d'accéder au matériel.

Jusqu'à présent, grâce à sa nature open-source, les utilisateurs ont construit des cartes Arduino de différentes tailles, formes et niveaux de puissance pour contrôler leurs projets.

- Arduino est composé de deux parties principales :
  - La carte Arduino, qui est le matériel sur lequel vous travaillez lorsque vous construisez vos objets ;
  - L'IDE Arduino, qui est le logiciel que vous exécutez sur votre ordinateur pour créer un croquis (un petit programme informatique) que vous téléchargez sur la carte Arduino.

Capteurs, actionneurs et programmation des microcontrôleurs

Capteurs, actionneurs et programmation des microcontrôleurs

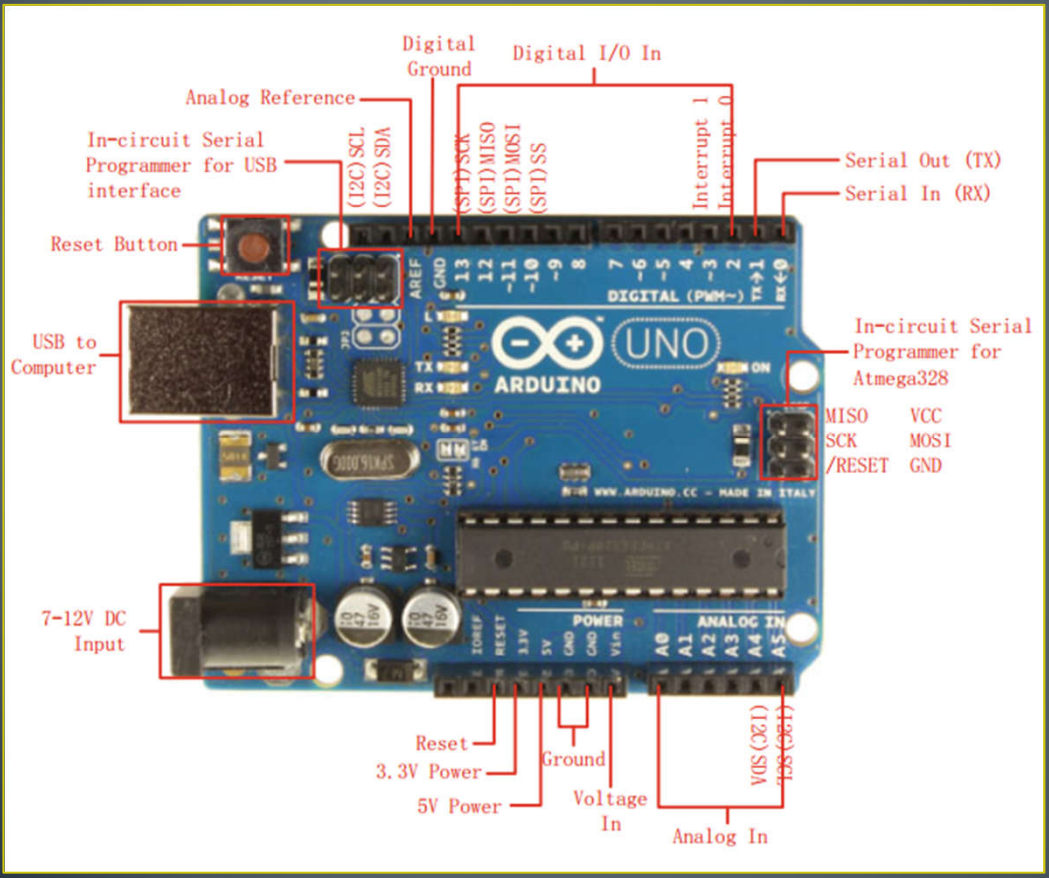
Il existe de nombreux types de microcontrôleurs Arduino qui diffèrent non seulement par leur conception et leur fonctionnalité, mais aussi par leur taille et leurs capacités de traitement. Cependant, seuls deux modèles utilisent des puces complètement différentes: le modèle Standard utilise la puce Atmega 8/168/328 et le modèle Méga utilise la puce Atmega1280, plus robuste, avec plus de broches d'entrée/sortie.

Il existe de nombreux autres fabricants qui utilisent des schémas open-source fournis par Arduino pour fabriquer leurs propres planches (soit identiques à l'original, soit avec des variations pour ajouter des fonctionnalités), par exemple, DFRobot.

Pour des besoins de simulation, nous utilisons uniquement la carte Arduino Uno R3.

Capteurs, actionneurs et programmation des microcontrôleurs

Capteurs, actionneurs et programmation des microcontrôleurs



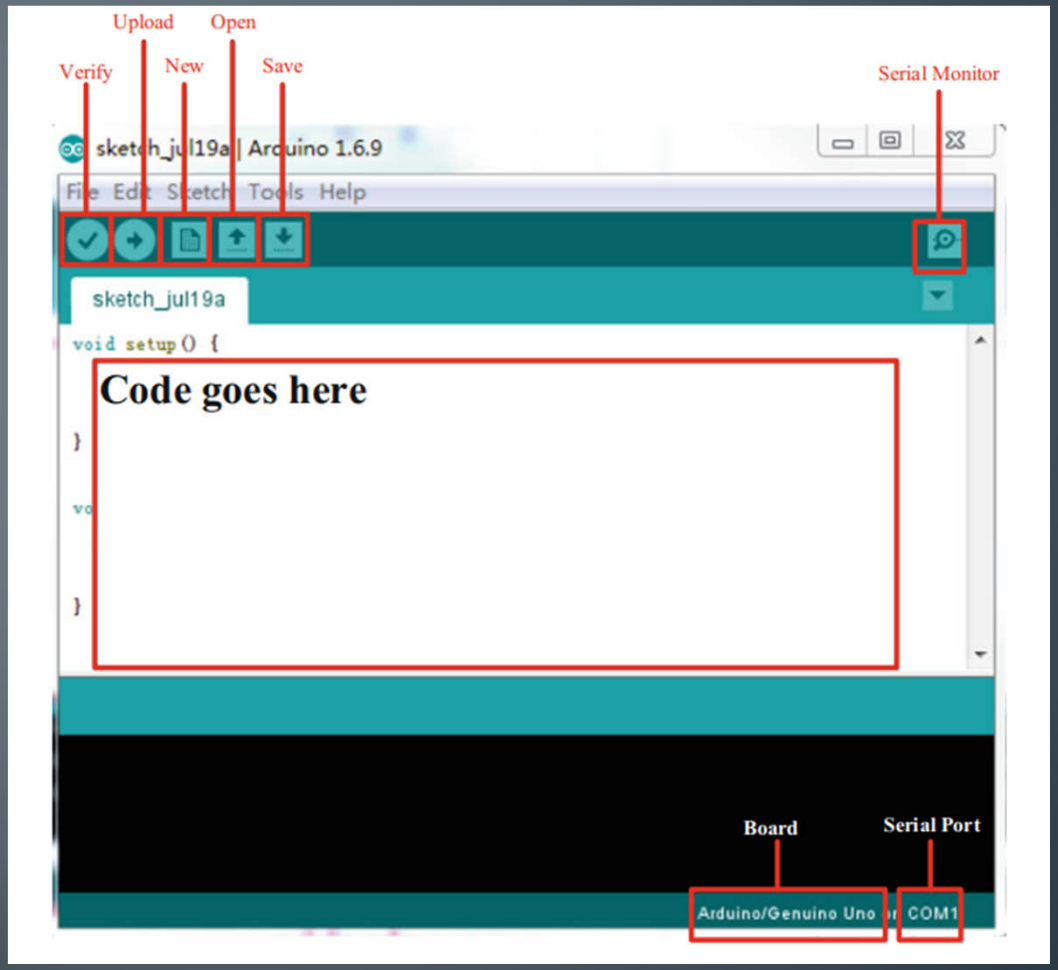
Uno est essentiellement constituée de :

- 14 broches d'entrée/sortie numériques ;
- 6 entrées analogiques;
- un résonateur céramique de 16 MHz ;
- une connexion USB ;
- une prise d'alimentation (7 – 12V DC) ;
- un bouton de réinitialisation ;
- Etc.

Capteurs, actionneurs et programmation des microcontrôleurs

Capteurs, actionneurs et programmation des microcontrôleurs

- Avant de commencer votre travail, vous devez d'abord télécharger l'environnement de développement (l'IDE) sur : www.arduino.cc/en/Main/Software .



Capteurs, actionneurs et programmation des microcontrôleurs

Capteurs, actionneurs et programmation des microcontrôleurs

Le langage Arduino est basé sur C/C++ et supporte toutes les constructions C standard ainsi que certaines fonctionnalités C++. Il est lié à AVR Libc et permet l'utilisation de n'importe laquelle de ses fonctions.

Structures	
void setup()	La fonction est appelée quand un sketch commence. Utilisez-la pour initialiser des variables, les pin modes, commencer à utiliser des bibliothèques, etc. La fonction de configuration ne sera exécutée qu'une seule fois, après chaque mise sous tension ou réinitialisation de la carte Arduino.
void loop()	la fonction fait des boucles consécutives, permettant à votre programme de changer et de répondre. Utilisez-le pour contrôler activement la carte Arduino.

Capteurs, actionneurs et programmation des microcontrôleurs

Structure de contrôle	

if (condition) {
//instructions
}
	Teste si une certaine condition a été atteinte. Utilisé en conjonction avec un operateur de comparaison.

switch (var) {
case label : //instructions
break ;
…
default : //instructions
}
	Vous permet de spécifier différents codes qui doivent être exécutés dans diverses conditions.

for(init; condition; incrément){
//instructions
}
	Crée une boucle pour les opérations dont on connait le nombre de répétition.

while (condition) {
//instructions
}
	Boucles continues, et infinies, jusqu'`a ce que l'expression à l'intérieur de la parenthèse, devienne fausse.

do {
//instructions
} while (condition);
	Fonctionne de la même manière que la boucle while, à l'exception du fait que l'état est testé à la fin de la boucle, tout comme la boucle do, qui sera toujours exécutée au moins une fois.

Capteurs, actionneurs et programmation des microcontrôleurs

Structure de contrôle	
break	Sorties d'une boucle do, for ou while, en contournant la condition normale de la boucle.
continue	Saute le reste de l'itération actuelle d'une boucle (do, for, ou while). Il continue en vérifiant l'expression conditionnelle de la boucle, et procède à toutes les itérations suivantes.
return	Termine une fonction et renvoie une valeur d'une fonction à la fonction appelante.
goto	Transfère le flux du programme vers un point labellisé dans le programme.

Capteurs, actionneurs et programmation des microcontrôleurs

Autre syntaxe	
;	Chaque déclaration se termine par un point-virgule.
{ }	Les accolades bouclées viennent toujours par paires ; ils sont utilisés pour définir le début et la fin des fonctions, des boucles et des instructions conditionnelles.
//	Commentaire d'une seule ligne.
/* */	Commentaire de plusieurs lignes.
#define	Utilisé pour donner un nom à une valeur constante.
#include	Inclure les bibliothèques extérieures dans votre croquis.

Capteurs, actionneurs et programmation des microcontrôleurs

Opérateurs de comparaison	
x == y	x est égal à y.
x != y	x n'est pas égal à y.
x < y	x est inférieur à y.
x > y	x est supérieur à y.
x <= y	x est inférieur ou égal à y.
x >= y	x est supérieur ou égal à y

Les opérateurs booléens	
&&	ET logique. Vrai seulement si les deux opérandes sont vrais.
||	OU logique. Vrai si l'un des opérandes est vrai.
!	NON logique. Vrai si l'opérande est faux

Capteurs, actionneurs et programmation des microcontrôleurs

Constantes	
HIGH / LOW	Lors de la lecture ou de l'écriture sur une broche numérique, il n'y a que deux valeurs possibles qu'une broche peut prendre (ou être réglée sur) : HIGH et LOW
true / false	Niveaux logiques (résultat d'une comparaison) : false est défini comme 0, true est défini comme 1 (mais plus largement, tout sauf 0).
INPUT / OUTPUT	Les broches numériques peuvent être utilisées en INPUT (entrée) ou en OUTPUT (sortie). Changer une broche de INPUT à OUTPUT avec pinMode() change radicalement le comportement électrique de la broche.

Capteurs, actionneurs et programmation des microcontrôleurs

Types de données	
void	Utilisé dans les déclarations de fonction pour indiquer que la fonction ne renvoie aucune information.
boolean	Un booléen possède l'une des deux valeurs suivantes, vraie ou fausse.
char	Un type de données qui stocke une valeur de caractère.
unsigned char	Un type de données non signé qui occupe 1 octet de mémoire. Identique au type de données byte. Le type de données char non signé encode les nombres de 0 à 255.
byte	Un byte stocke un nombre non signé de 8 bits, de 0 à 255.
int	Les nombres entiers sont votre principal type de données pour le stockage des nombres, et stockent une valeur de 2 octets. Cela donne une plage de -32 768 à 32 767
unsigned int	Les ints non signés (entiers non signés) sont identiques aux int en ce sens qu'ils stockent une valeur de 2 octets. Au lieu de stocker des nombres négatifs, ils ne stockent que des valeurs positives, ce qui donne une plage utile de 0 à 65 535 .

Capteurs, actionneurs et programmation des microcontrôleurs

Types de données	
word	Un word stocke un nombre non signé de 16 bits, de 0 à 65536. Même chose qu'un int non signé.
long	Les variables de type long sont des variables de taille étendue pour le stockage des nombres et stockent 32 bits (4 octets), de -2 147 483 648 à 2 147 483 647.
unsigned long	Contrairement aux longs standards, les longs non signés (unsigned long) ne stockent pas de nombres négatifs, ce qui fait que leur plage de stockage va de 0 à 4 294 967 295.
float	Type de données pour les nombres à virgule flottante, un nombre qui a un point décimal.
double	Les nombres à virgule flottante sont souvent utilisés pour approximer des valeurs analogiques et continues car ils ont une plus grande résolution que les nombres entiers. Les nombres à virgule flottante ont une précision de 6 à 7 chiffres décimaux.
string	Les chaînes sont représentées sous forme de tableaux de type char et se terminent par null.
arrays	Il est souvent pratique, lorsqu'on travaille avec de grandes quantités de texte, comme dans le cas d'un projet avec un écran LCD, de mettre en place une série de chaînes.

Capteurs, actionneurs et programmation des microcontrôleurs

Fonctions : E/S numériques	
pinMode(pin, mode)	Configure la broche spécifiée pour qu'elle se comporte comme une entrée ou une sortie. pin est le numéro de pin.
digitalWrite(pin, value)	Ecrire une valeur HIGH ou LOW sur une broche numérique.
digitalRead(pin)	Lire la valeur à partir d'une broche numérique spécifiée. Le résultat sera soit HIGH, soit LOW.

Fonctions : E/S analogiques	
analogReference(type)	La tension de référence par défaut est de 5 V. Cela peut être changé en un autre type et une résolution différente en utilisant cette fonction.
analogRead(pin)	Lire la valeur de la broche analogique spécifiée et renvoie une valeur comprise entre 0 et 1023 pour représenter une tension entre 0 et 5 V (par défaut). Il faut environ 0,0001 s pour lire une broche analogique.
analogWrite(pin,value)	Ecrire une valeur analogique (**onde PWM**) sur une broche. valeur est le rapport cyclique : entre 0 (toujours désactivé) et 255 (toujours activé). Fonctionne sur les broches 3, 5, 6, 9, 10 et 11.

Capteurs, actionneurs et programmation des microcontrôleurs

Fonctions : communication série	
Serial.begin(9600)	Utilisé pour commencer les communications série, généralement à un débit de 9600 bauds (bits par seconde).
Serial.print(val,format)	Imprime les données sur le port série sous forme de texte ASCII lisible par l'homme.
Serial.println(val)	Imprime le val suivi du retour chariot.
Serial.available()	Obtenez le nombre d'octets (caractères) disponibles pour la lecture à partir du port série. Il s'agit des données déjà arrivées et stockées dans le tampon de réception série (qui contient 128 octets).
Serial.read()	Lire les données série entrantes.
Serial.write()	Ecrire des données binaires sur le port série.
Serial.end()	Désactive la communication série, ce qui permet d'utiliser les broches RX et TX pour les entrées et sorties générales.

Capteurs, actionneurs et programmation des microcontrôleurs

Temps	
delay(ms)	Suspend le programme pendant la durée (en millisecondes) spécifiée en paramètre.
delayMicroseconds(us)	Met le programme en pause pendant la durée (en microsecondes) spécifiée en paramètre.
micros()	Retourne le nombre de microsecondes depuis que la carte Arduino a commencé à faire fonctionner le programme actuel.
millis()	Renvoie le nombre de millisecondes écoulées depuis que la carte Arduino a commencé à exécuter le programme en cours.

Capteurs, actionneurs et programmation des microcontrôleurs

Fonctions : mathématiques	
min(x, y)	Calcule le minimum de deux nombres.
max(x, y)	Calcule le maximum de deux nombres.
abs(x)	Calcule la valeur absolue d'un nombre.
pow(base, exponent)	Calcule la valeur d'un nombre élevé à une puissance.
sqrt(x)	Calcule la racine carrée d'un nombre.
map(value, fromLow, fromHigh, toLow, toHigh)	Remappe un nombre d'une plage à une autre. Autrement dit, une valeur de fromLow serait mappée à toLow, une valeur de fromHigh à toHigh, des valeurs intermédiaires à des valeurs intermédiaires.

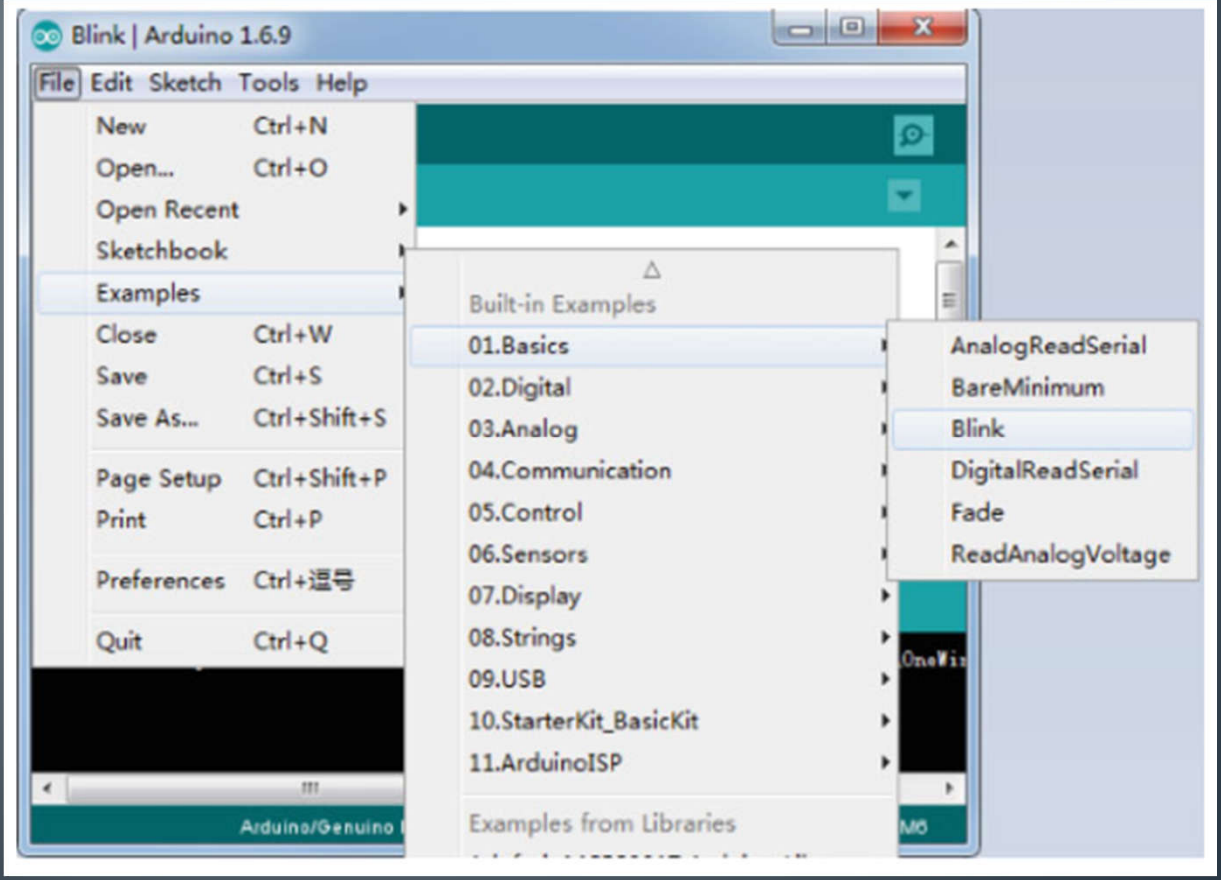
Capteurs, actionneurs et programmation des microcontrôleurs

Fonctions : trigonométrique	
sin()	Calcule le sinus d'un angle (en radians). Le résultat sera compris entre -1 et 1.
cos()	Calcule le cos d'un angle (en radians). Le résultat sera compris entre -1 et 1.
tan()	Calcule la tangente d'un angle (en radians). Le résultat sera compris entre l'infini négatif et l'infini.

Fonctions : nombres aléatoires	
randomSeed(seed)	Initialise le générateur de nombres pseudo-aléatoires, le faisant démarrer à un point arbitraire de sa séquence aléatoire.
random()	La fonction aléatoire génère des nombres pseudo-aléatoires.

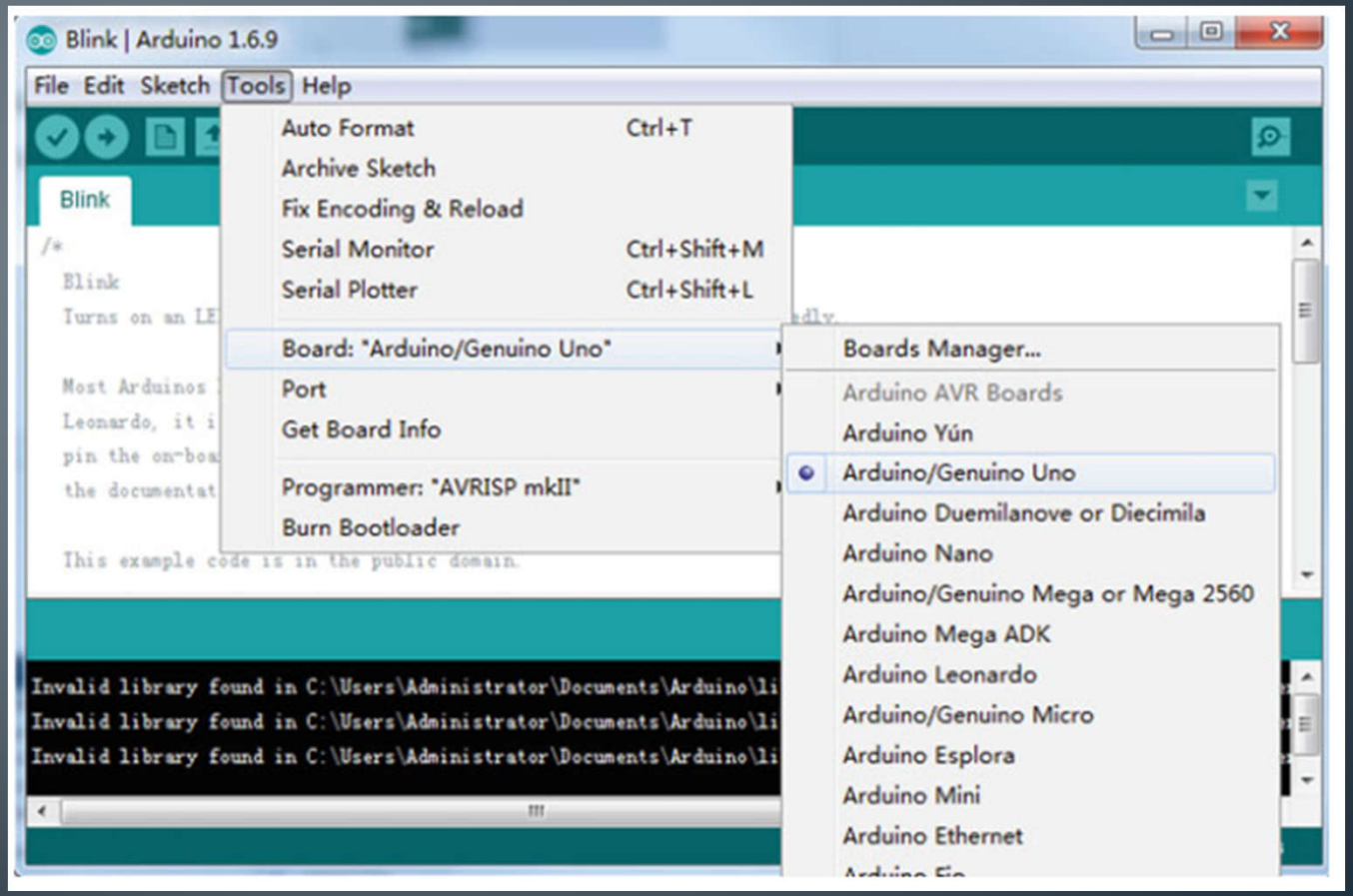
Capteurs, actionneurs et programmation des microcontrôleurs

- **Exemple avec "Blink" dans IDE** : Modifiez l'exemple de Blink pour allumer/éteindre une led ;



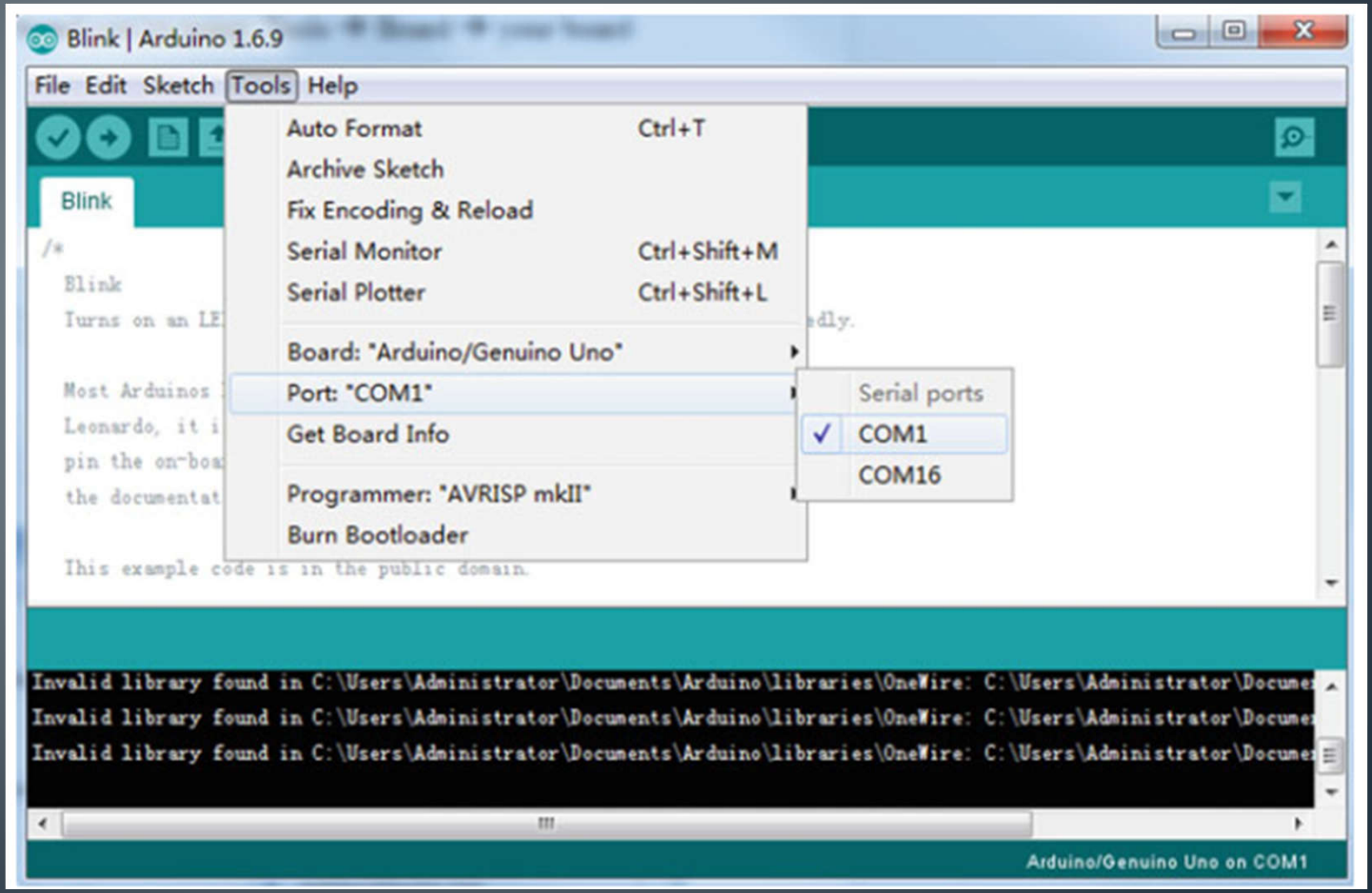
Capteurs, actionneurs et programmation des microcontrôleurs

- **Exemple avec "Blink" dans IDE** : Sélection du type de carte Arduino dans l'IDE



Capteurs, actionneurs et programmation des microcontrôleurs

- **Exemple avec "Blink" dans IDE** : Sélection du port série



À suivre

Feedback sur:

pape.abdoulaye.barro@gmail.com