

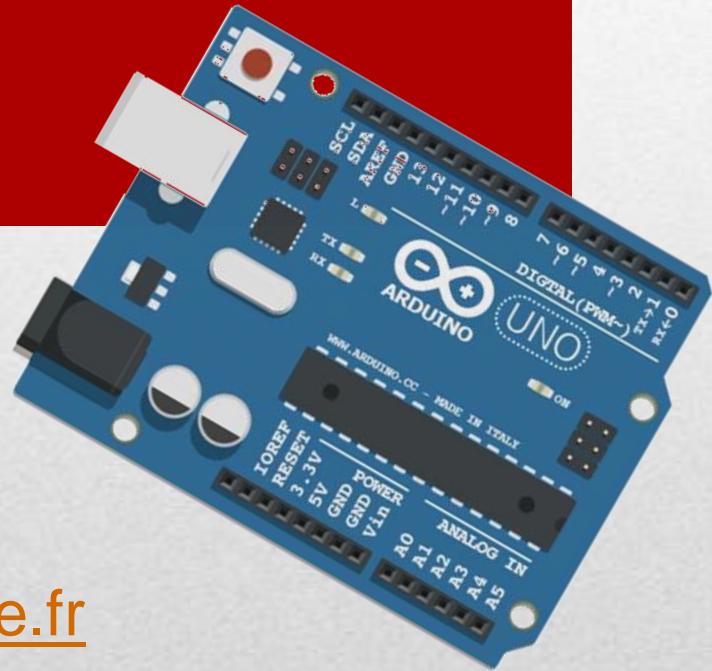
Sommaire

Partie 1 : Cours

Partie 2 : TP

Elect. Numérique

Auditoire : 2 Prépa MTIC



Enseignant : Nejmeddine Bahri

Email: nejmeddine.bahri@esiee.fr

AU: 2019-2020

1

Objectifs du cours:

- Découvrir l'architecture d'un système embarqué (structure/composants)
- Etudier le cas d'un microcontrôleur
- Connaitre l'architecture interne d'un microcontrôleur
- Découvrir la carte ARDUINO
- Programmer des applications en langage C pour ARDUINO
- Commander différents modules externes: capteurs, LED, moteurs, clavier, LCD avec la carte ARDUINO

Plan détaillé:

Les systèmes embarqués

- Définition d'un système embarqué
- Architecture d'un système embarqué
- Les différents types de calculateurs dans un système embarqué

Présentation de la carte ARDUINO

- C'est quoi ARDUINO?
- Caractéristiques techniques de l'Arduino UNO
- Un atout : les shields

Plan détaillé:

Présentation du logiciel

- IDE Arduino
- Langage Arduino

Fonctionnalités de base

- Les entrées/sorties
- La gestion du temps
- Les Interruptions

Exercices d'application

Les systèmes embarqués

Les systèmes embarqués

Définition:



- C'est un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise et possédant une taille limitée et ayant une consommation énergétique réduite.
- Le terme de système embarqué désigne aussi bien le matériel que le logiciel utilisé (l'application, système d'exploitation (ex: Linux embarqué)).

Les systèmes embarqués exécutent des tâches prédéfinies et ont un cahier des charges bien déterminé, qui peut être d'ordre :

- **De coût:** le prix de revient doit être le plus faible possible.
- **D'encombrement:** Il convient de concevoir des systèmes embarqués de taille réduite qui répondent aux besoins au plus juste pour éviter un surcoût.

Les systèmes embarqués

- **D'autonomie:** la consommation énergétique doit être la plus faible possible, due à l'utilisation de batteries et/ou, de panneaux solaires ou de piles pour certains prototypes.
- **De puissance de calcul:** Choix du processeur et des composants doit être bien étudié pour avoir la puissance de calcul juste nécessaire pour répondre aux besoins et aux contraintes temporelles de la tâche prédefinie. Ceci afin d'éviter un surcoût de l'appareil et une consommation excédentaire d'énergie (courant électrique).
- **Temporel:** les temps d'exécution et l'échéance temporelle d'une tâche sont déterminés. Certains systèmes ont des propriétés temps réel.
- **De sûreté de fonctionnement:** Ils doivent donner des résultats justes, pertinents et ce dans les délais attendus par les utilisateurs (machines et/ou humains). S'il arrive que certains de ces systèmes embarqués subissent une défaillance, ils mettent des vies humaines en danger ou mettent en périls des investissements importants.

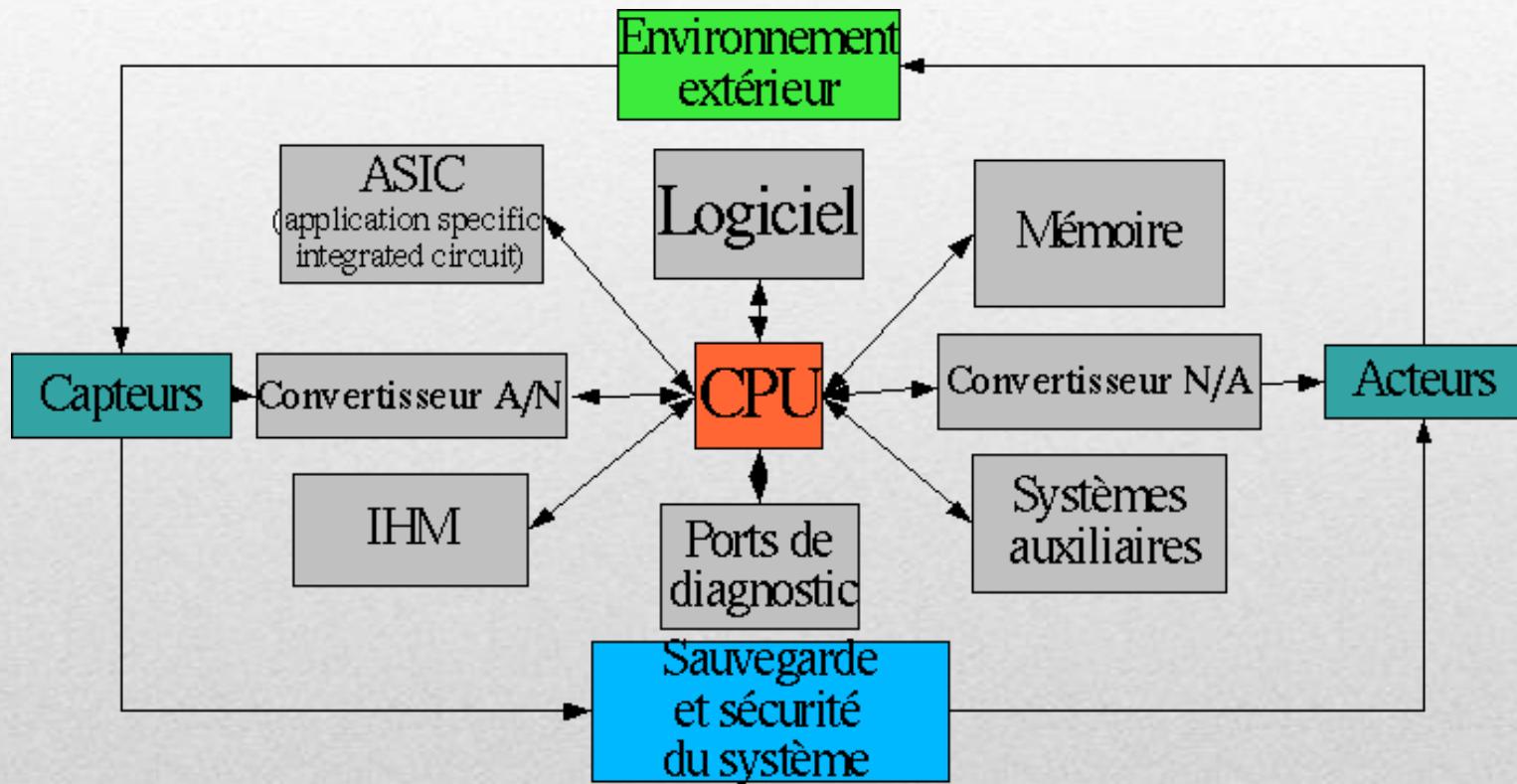
Les systèmes embarqués

Compétences requises:

Des compétences pluridisciplinaires en termes **d'électronique** (de commande et de puissance), **d'automatique** et **d'informatique industrielle** sont donc nécessaires pour concevoir de tels systèmes.

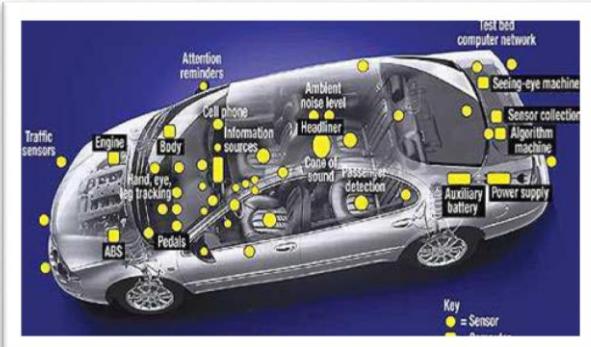
Les systèmes embarqués

Architecture d'un système embarqué:



Les systèmes embarqués

Domaines d'application:



Les systèmes embarqués

Domaines d'application:

Les domaines dans lesquels on trouve des systèmes embarqués sont de plus en plus nombreux :

- **transport** : Automobile, Aéronautique (avionique), etc.
- **astronautique** : fusée, satellite, sonde spatiale, etc.
- **militaire** : missile
- **télécommunication** : téléphonie, routeur, téléphone portable, etc.
- **électroménager** : télévision, four à micro-ondes, machine à laver
- **impression** : imprimante multifonctions, photocopieur, etc.
- **informatique** : pointeur laser, souris/clavier sans fils etc.
- **multimédia** : console de jeux vidéo, caméras numériques
- Distributeur automatique des billets (DAB)
- équipements médicaux
- Métrologie (capteurs ultrason, laser...etc pour mesurer les distances ...etc)

Les systèmes embarqués

Les différents types de calculateurs dans un système embarqué :

Les DSP (Digital Signal Processor)



Ils sont des microprocesseurs optimisés pour exécuter des applications de traitement numérique du signal (télécommunication, mobiles, routeurs.)

Certains modèles sont utilisés dans certains processus industriels comme la commande des machines électriques

Processeurs graphiques, ou GPU (*Graphics Processing Unit*)



Ils assurent en général des tâches graphiques comme le rendu 3D, le traitement du signal vidéo, la décompression MPEG, h264/AVC, les jeux vidéo etc.

Les systèmes embarqués

Les différents types de calculateurs dans un système embarqué :

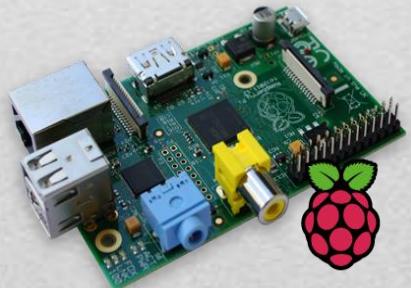
Les Processeurs ARM



Ils sont devenus dominants dans le domaine de l'informatique embarquée, en particulier les tablettes électroniques et la téléphonie mobile, les cameras numériques etc.

Produits à base de ARM

Raspberry Pi



BeagleBone



OK6410



ODROID



Freescale



Les systèmes embarqués

Les différents types de calculateurs dans un système embarqué :

Les FPGA (Field Programmable Gate Arrays)

Un FPGA est un circuit intégré contenant un nombre important de blocs logiques librement reconfigurables. Chaque bloc peut être configuré pour réaliser une fonction de base.

Contrairement aux processeurs, les FPGA sont massivement parallèles. Chaque tâche de traitement indépendante est affectée à une section spécifique du circuit, et peut donc s'exécuter en toute autonomie sans dépendre aucunement des autres blocs logiques.

Ils sont utilisés dans diverses applications nécessitant de l'électronique numérique (télécommunications, aéronautique, automobile).



Les systèmes embarqués

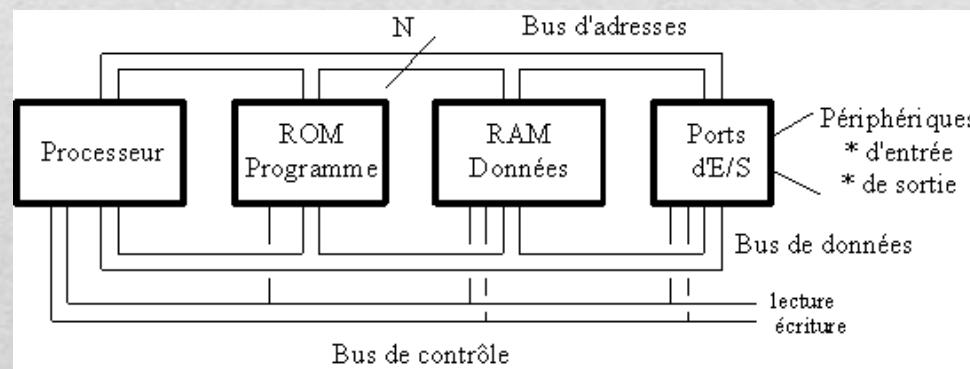
Les différents types de calculateurs dans un système embarqué :

Les microcontrôleurs



C'est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires et interfaces d'entrées-sorties sur une seule puce.

Ces sont des circuits intégrés caractérisés par un plus haut degré d'intégration, une faible consommation électrique et un coût réduit



**On s'intéresse dans cette
formation à la carte
ARDUINO UNO**

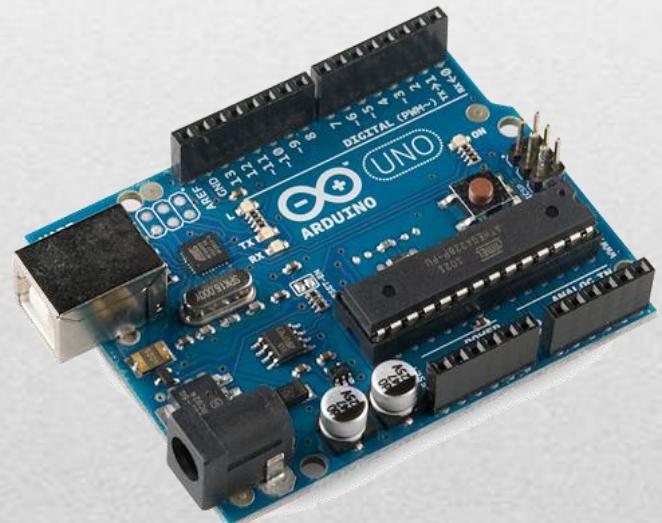
Présentation de la carte ARDUINO

Présentation de la Carte ARDUINO

C'est quoi ARDUINO?

Arduino est un **circuit imprimé en matériel libre (Open Source)** sur lequel se trouve un **microcontrôleur** qui peut être programmé pour analyser et produire des **signaux électriques**.

1- Circuit imprimé : C'est une sorte de plaque sur laquelle sont soudés plusieurs composants électroniques reliés entre eux par un circuit électrique plus ou moins compliqué.



Présentation de la Carte ARDUINO

C'est quoi ARDUINO?

Arduino est un **circuit imprimé en matériel libre (Open Source)** sur lequel se trouve un **microcontrôleur** qui peut être programmé pour analyser et produire des **signaux électriques**.

2- Matériel libre : En fait, les plans de la carte elle-même sont accessibles par tout le monde, gratuitement. La notion de libre est importante pour des questions de droits de propriété.

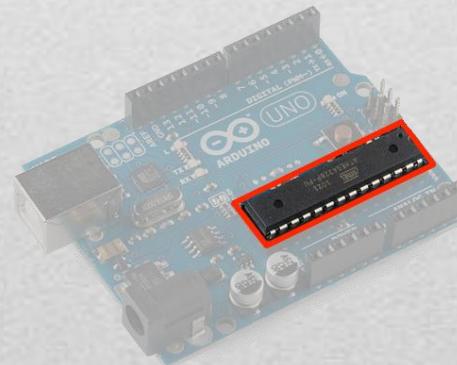


Présentation de la Carte ARDUINO

C'est quoi ARDUINO?

3- Microcontrôleur : C'est le cœur de la carte Arduino. C'est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires et interfaces d'entrées-sorties sur une seule puce. C'est lui que nous allons programmer de manière à effectuer des tâches très diverses comme la domotique (éclairage, chauffage...), le pilotage d'un robot, de l'informatique embarquée, etc.

La carte Arduino est construite autour d'un **microcontrôleur** d'architecture Atmel AVR comme par exemple ATmega328, ATmega32u4 ou ATmega2560



Présentation de la Carte ARDUINO

Les avantages de ARDUINO:

- Architectures matérielle et logicielle libres
- Platines prêtées à l'emploi
- API (fonction) de programmation du µC en C(++) → programmation simple
- Logiciel de développement simple

Les différentes cartes ARDUINO qui existent sur le marché

ENTRY LEVEL	UNO	LEONARDO	101	ESPLORA	MICRO	NANO	MINI	MKR2UNO ADAPTER
	STARTER KIT	LCD SCREEN						
ENHANCED FEATURES	MEGA	ZERO	DUE	MEGA ADK	MO	MO PRO	MKR ZERO	MOTOR SHIELD
	USB HOST SHIELD	PROTO SHIELD	MKR PROTO SHIELD	4 RELAYS SHIELD	MEGA PROTO SHIELD			
	MKR RELAY PROTO SHIELD	ISP	USB2SERIAL MICRO	USB2SERIAL CONVERTER				
	YUN	ETHERNET	TIAN	INDUSTRIAL 101	LEONARDO ETH	MKR FOX 1200	MKR1000	
INTERNET OF THINGS	YUN MINI	YUN SHIELD	WIRELESS SD SHIELD	WIRELESS PROTO SHIELD	ETHERNET SHIELD V2			
	GSM SHIELD V2	MKR IoT BUNDLE						
	CTC 101							
EDUCATION	GEMMA	LILYPAD ARDUINO USB	LILYPAD ARDUINO MAIN BOARD	LILYPAD ARDUINO SIMPLE				
	LILYPAD ARDUINO SIMPLE SNAP							
WEARABLE								
3D PRINTING	MATERIA 101							
	BOARDS	MODULES	SHIELDS	KITS	ACCESSORIES	COMING NEXT		

Les différentes cartes ARDUINO qui existent sur le marché

Entry Level



Les différentes cartes ARDUINO qui existent sur le marché

Enhanced Features



ARDUINO MEGA 2560



ARDUINO ZERO



ARDUINO DUE



ARDUINO MEGA ADK



ARDUINO M0



ARDUINO M0 PRO

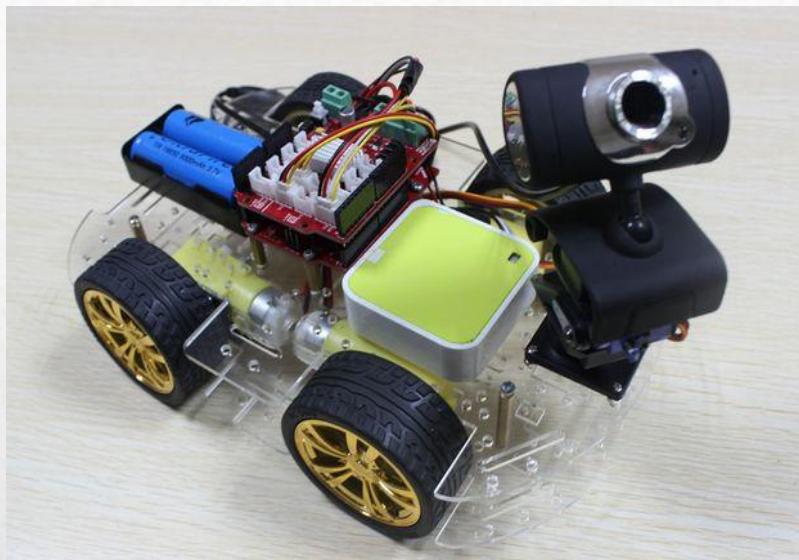
Les utilisations de la carte ARDUINO

Que qu'on peut faire avec ARDUINO ?

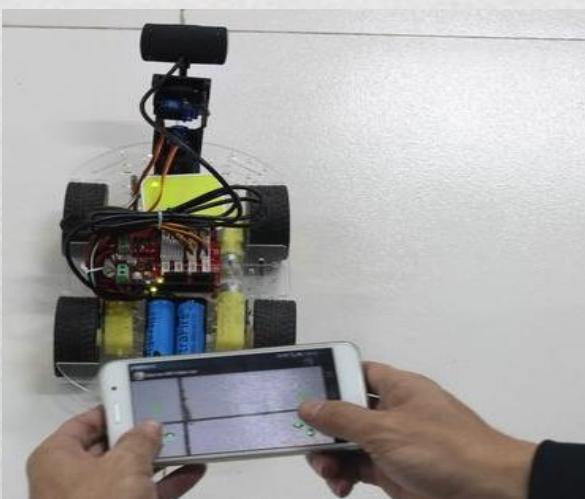
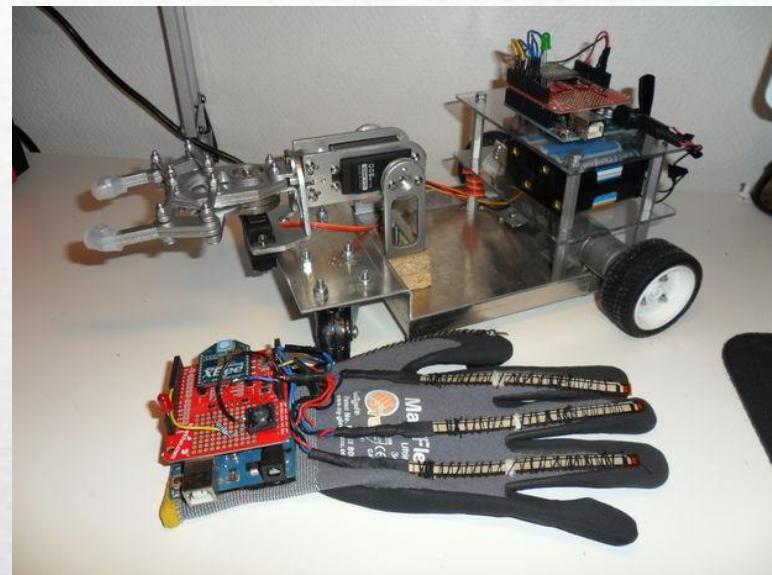
Domotique : commande à distance d'une maison via smartphone



Que qu'on peut faire avec ARDUINO ?



robotique

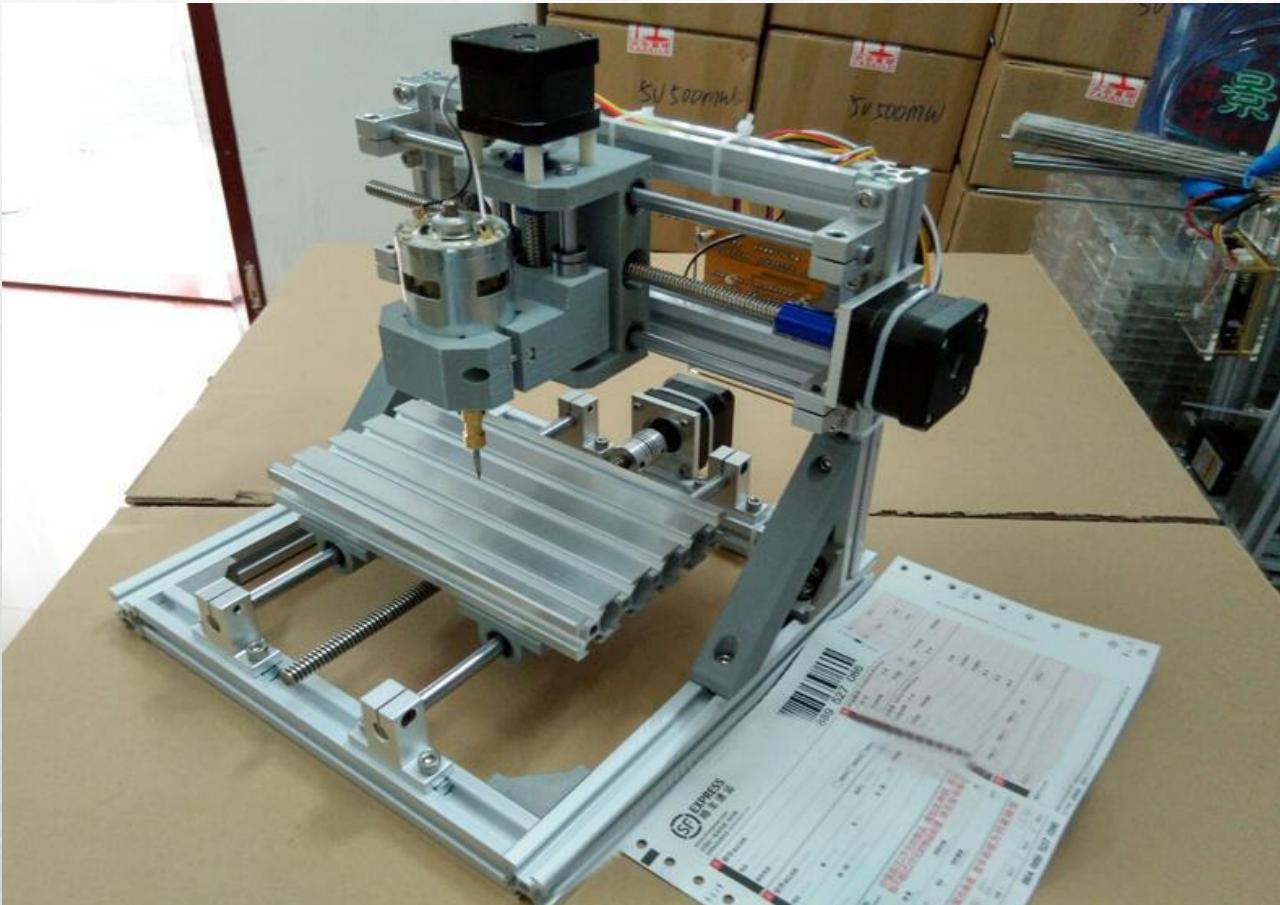


Drone



Que qu'on peut faire avec ARDUINO ?

Mini CNC 3 axes (Machine à commande numérique)



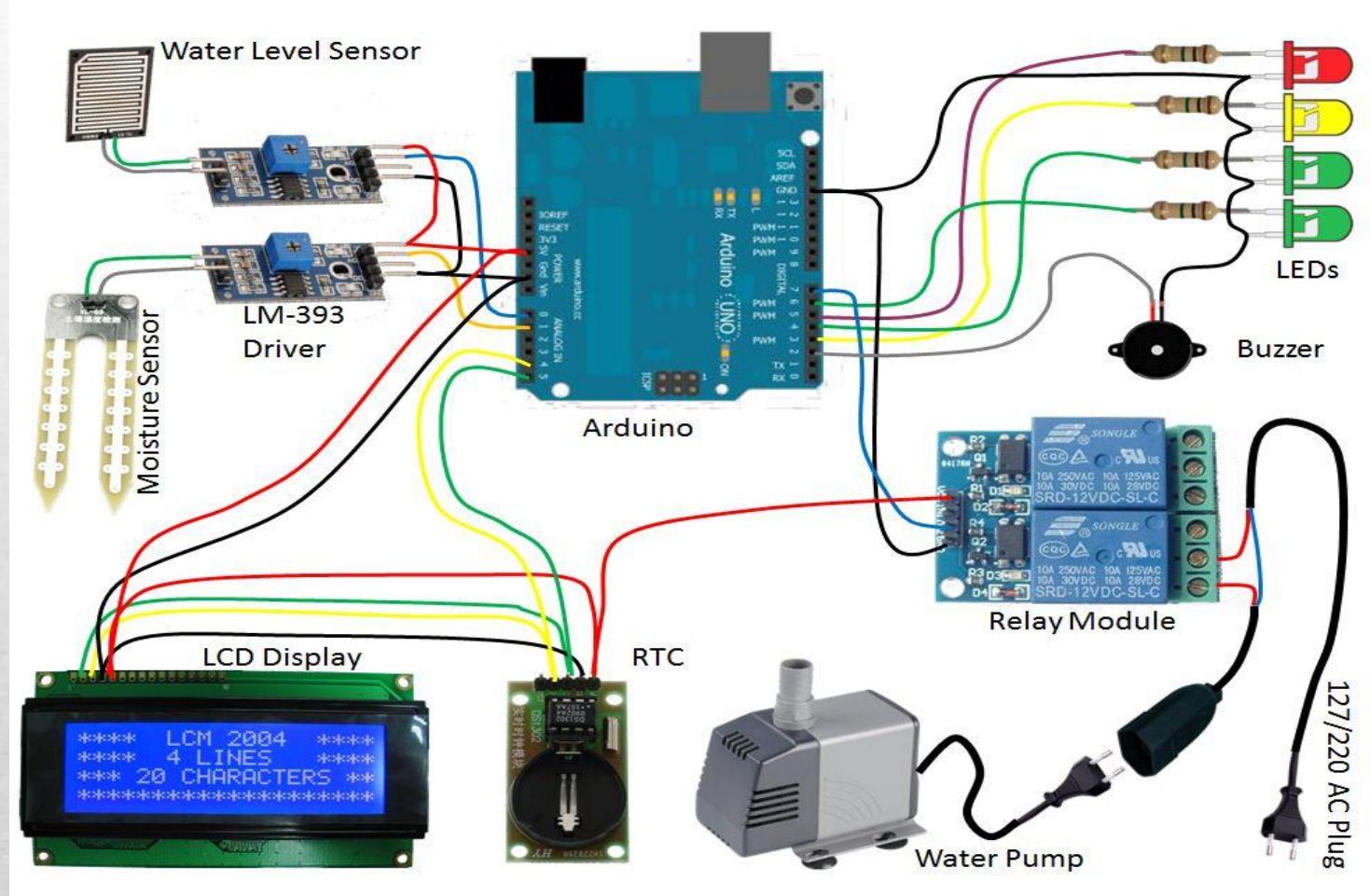
Que qu'on peut faire avec ARDUINO ?

Station de météo



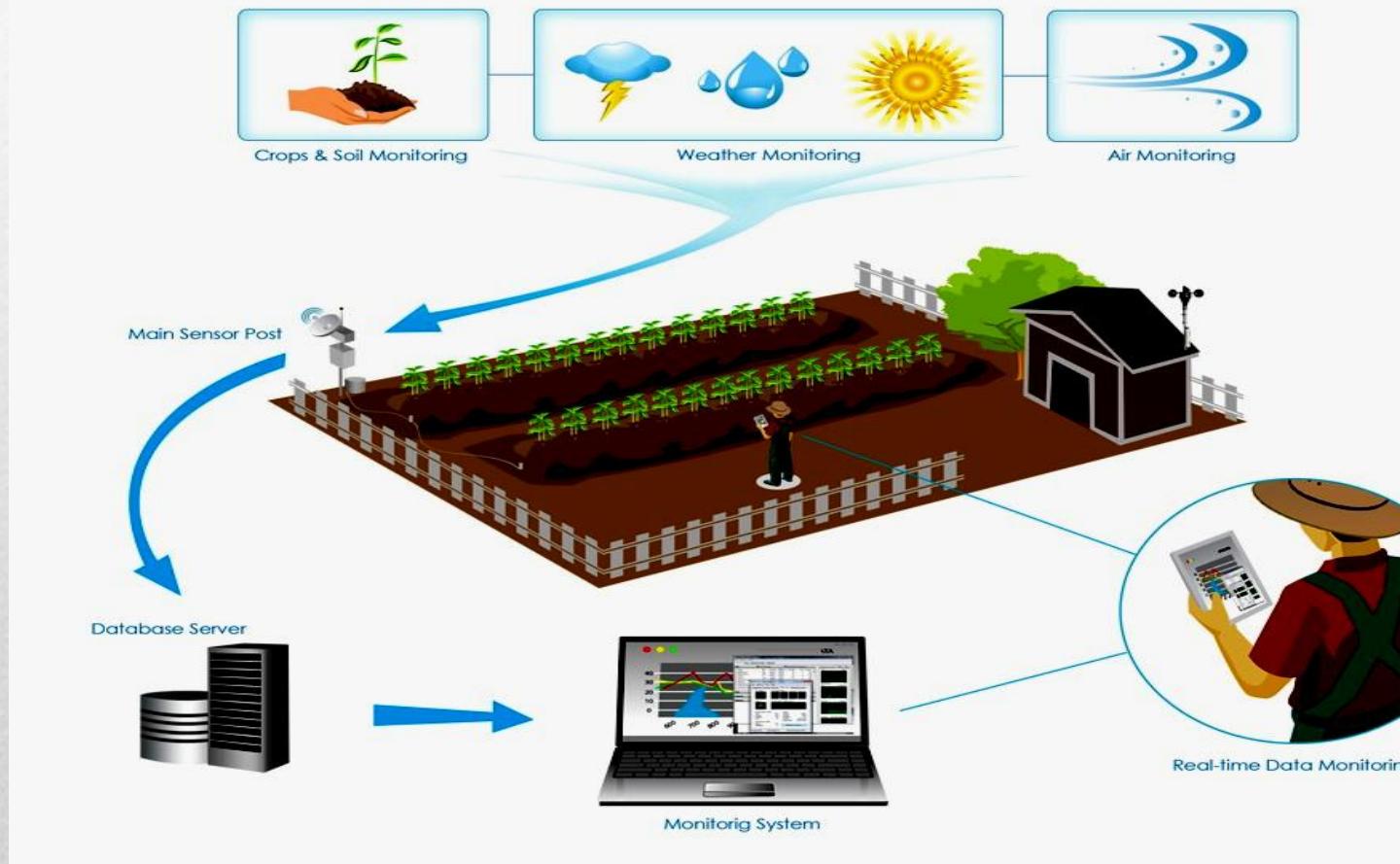
Que qu'on peut faire ARDUINO ?

Station d'irrigation



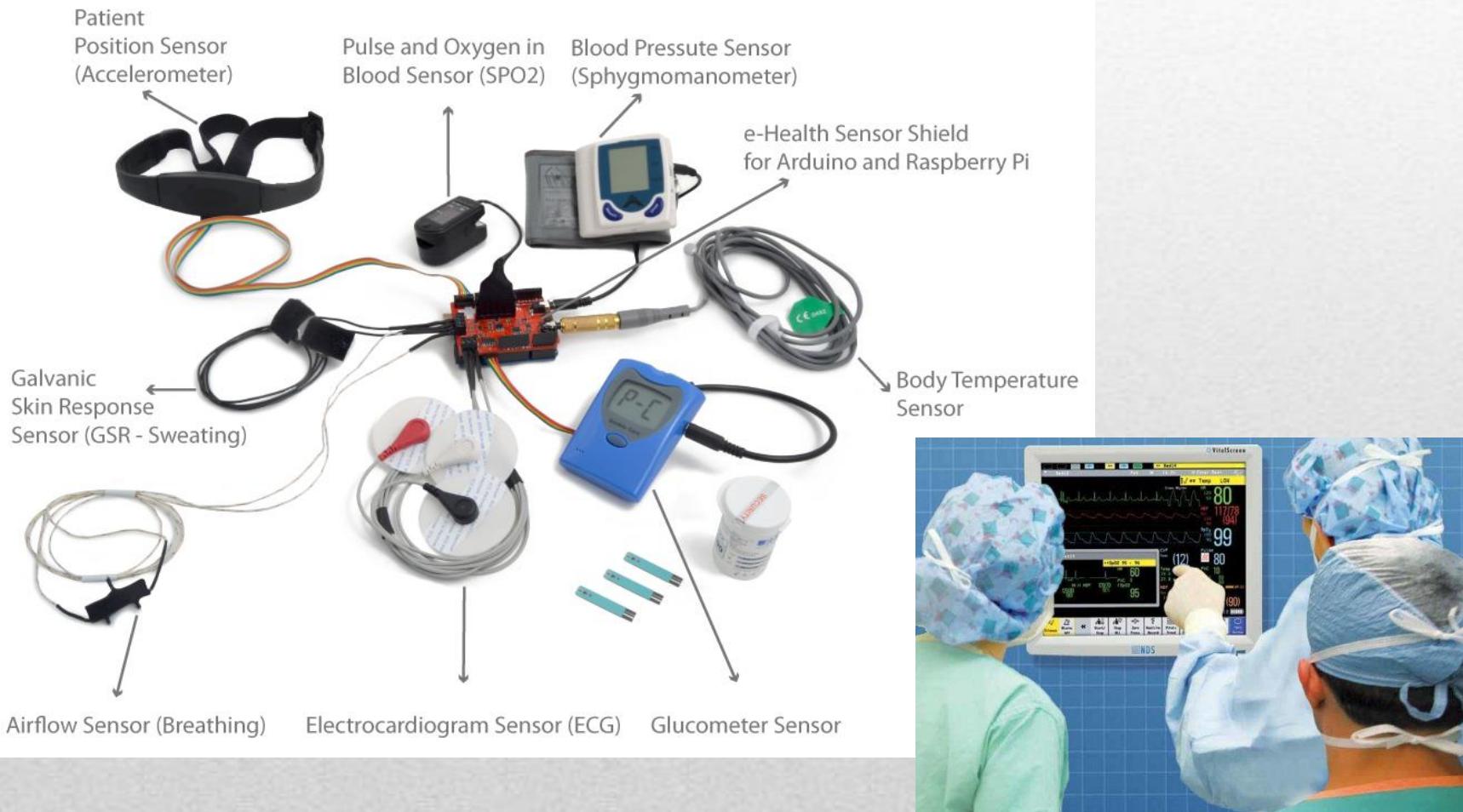
Que qu'on peut faire avec ARDUINO ?

Agriculture: contrôle à distance d'une ferme agricole :
température, humidité, vent etc...



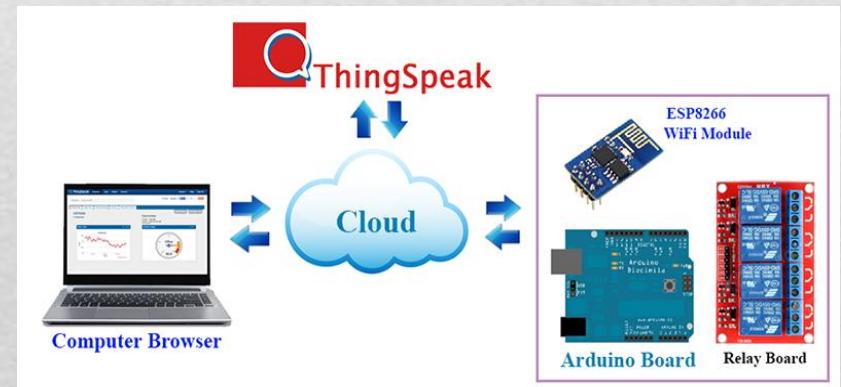
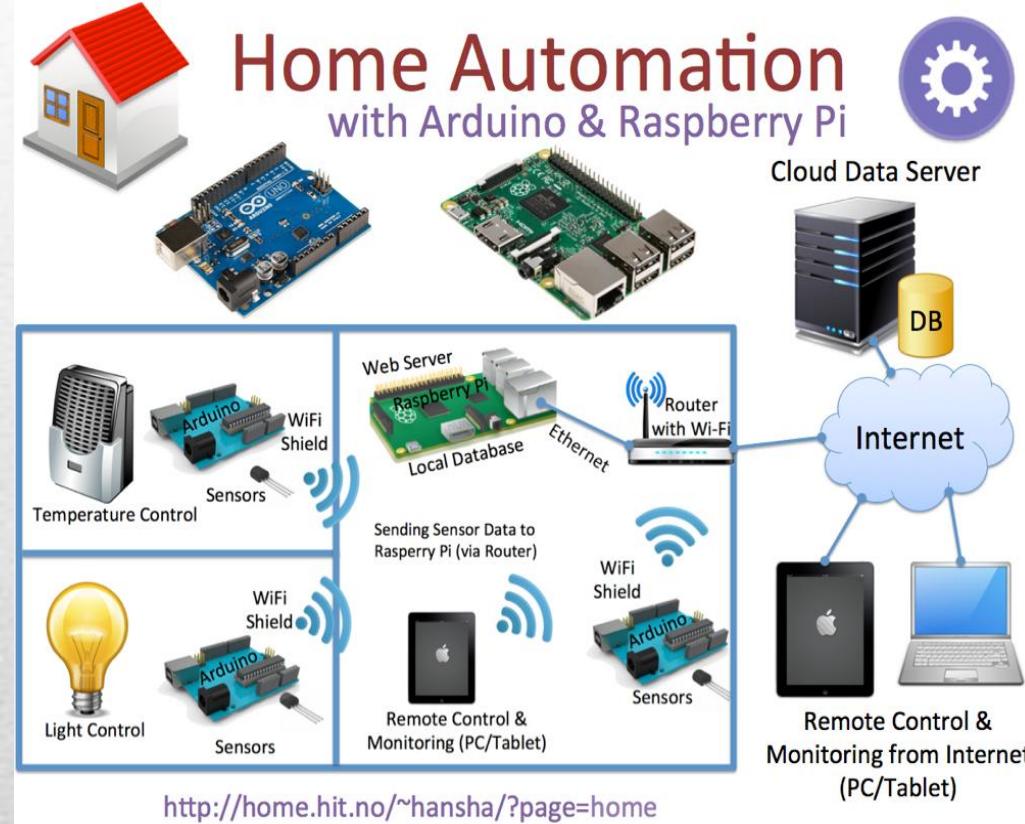
Que qu'on peut faire avec ARDUINO ?

Domaine médical: surveiller l'état d'un malade en utilisant le kit e-health



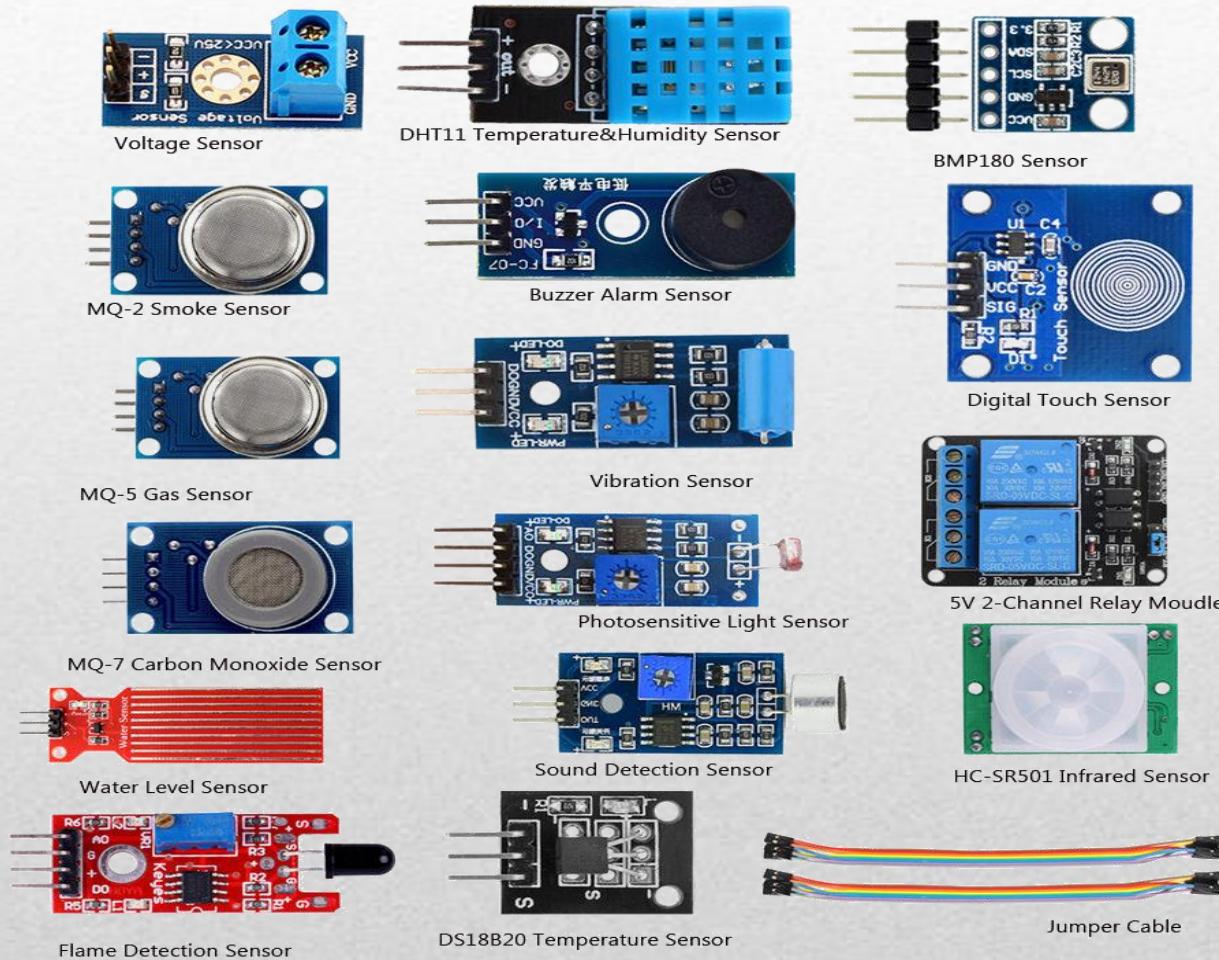
Que qu'on peut faire avec ARDUINO ?

IOT (Internet of things)



Que qu'on peut faire avec ARDUINO ?

Smart Home IOT Sensor Kit



Présentation de la Carte ARDUINO

On s'intéresse dans ce cours à la carte **ARDUINO UNO** .

C'est le modèle le plus répandu ainsi que c'est la première version stable des cartes Arduino.

Elle possède toutes les fonctionnalités d'un microcontrôleur classique en plus de **sa simplicité d'utilisation**.

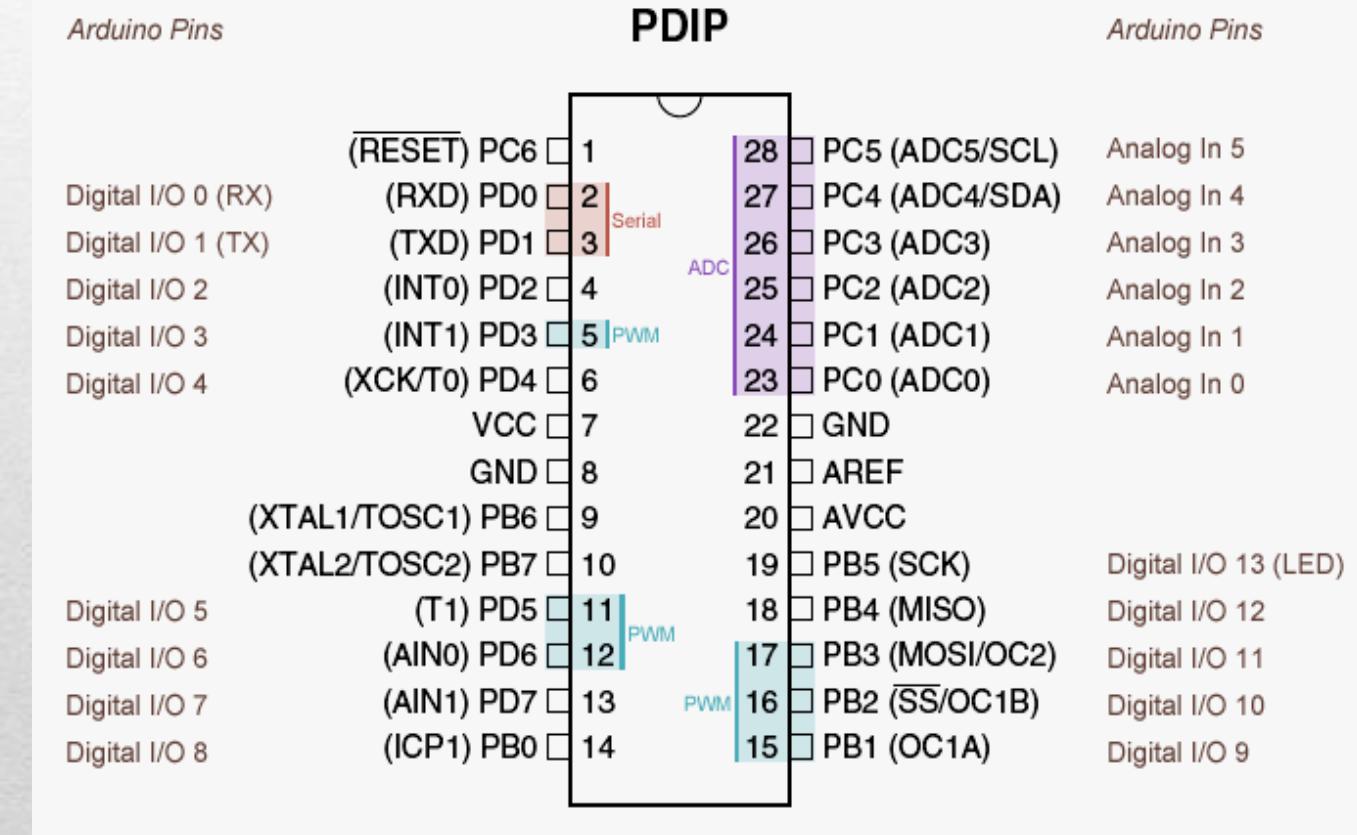
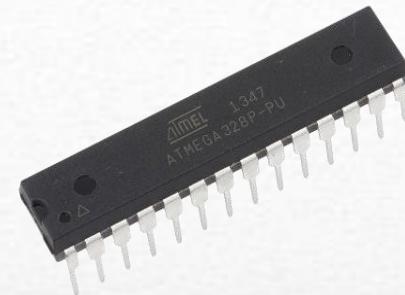
Présentation de la Carte ARDUINO

Caractéristiques techniques de l'Arduino UNO

- La carte ARDUINO UNO utilise un microcontrôleur **ATmega328P** cadencé à **16Mhz**.
- Elle possède **32ko de mémoire flash** destinée à recevoir le programme, **2ko de SRAM** (mémoire vive) et **1 ko d'EEPROM** (mémoire morte destinée aux données).
- Elle offre **14 pins d'entrée/sortie numérique** (données acceptée 0 ou 1) dont **6 pouvant générer des PWM** (Pulse Width Modulation, détaillé plus tard).
- Elle permet aussi de mesurer des grandeurs analogiques grâce à ces **6 entrées analogiques**. Chaque broche est capable de délivrer un courant de 40mA pour une tension de 5V.
- 2 timers/Counters de 8 bits et 1 Timer/Counter de 16 bits
- Cette carte Arduino peut aussi s'alimenter et communiquer avec un ordinateur grâce à son port USB . On peut aussi l'alimenter avec un alimentations comprise en 7V et 12V grâce à sa connecteur Power Jack.

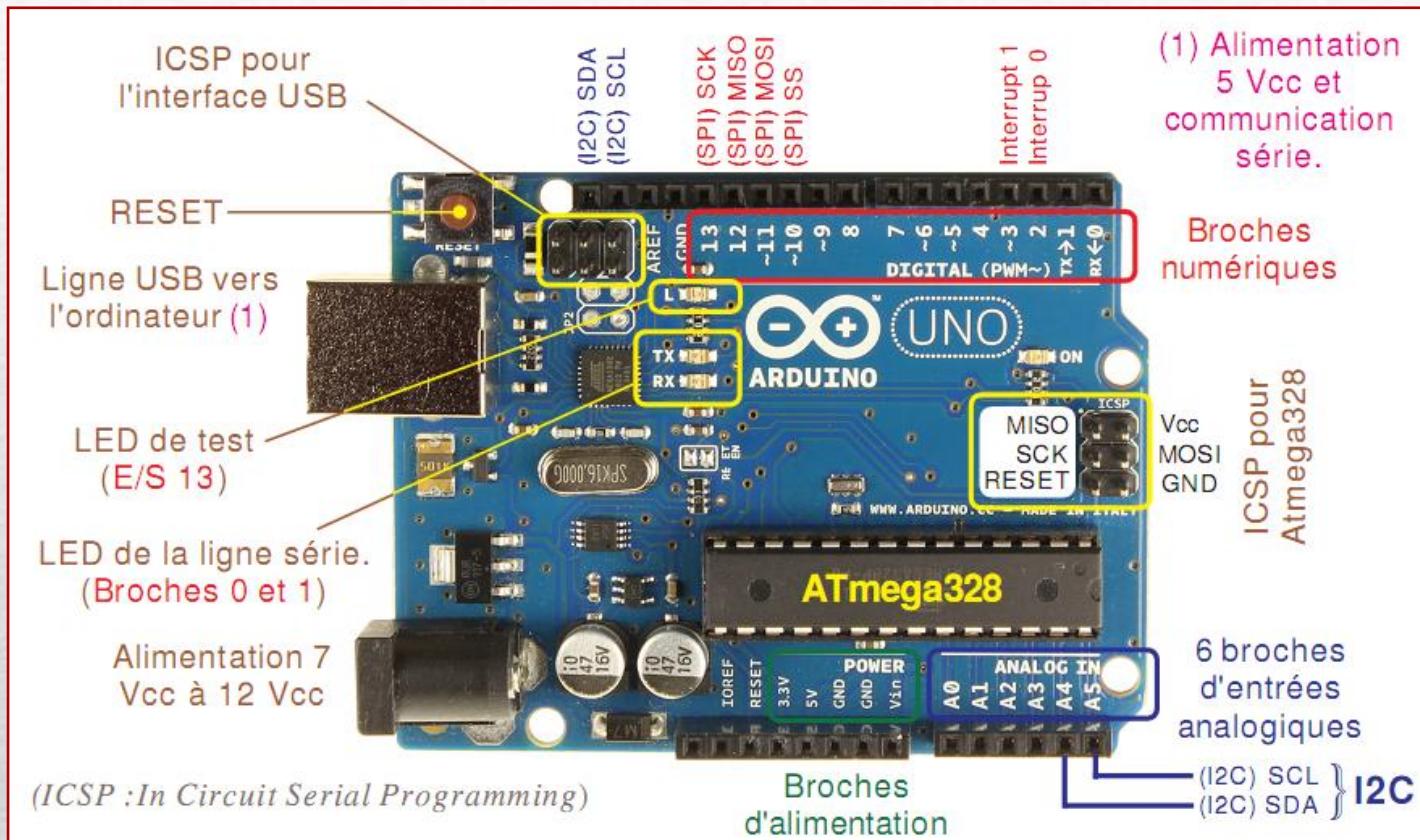
Présentation de la Carte ARDUINO

Atmega320P



Présentation de la Carte ARDUINO

Caractéristiques techniques de l'Arduino UNO



Présentation de la Carte ARDUINO

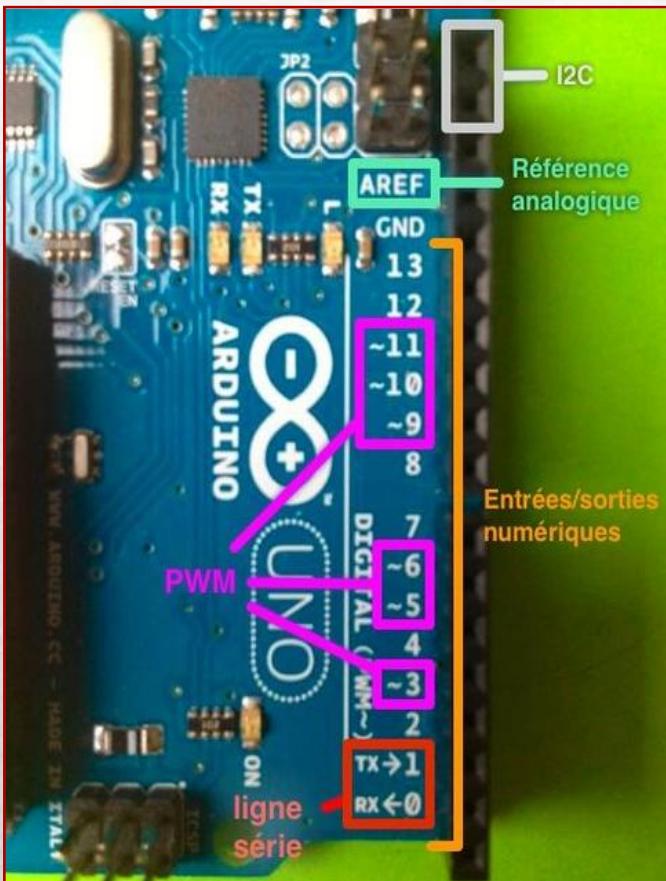
Les entrées-sorties numériques

Toutes les broches d'un Arduino peuvent être programmées en entrée ou sortie numérique (et non les deux en même temps). Par exemple, sur l'Arduino Uno, il s'agit d'une part des broches numérotées de 0 à 13 mais également des broches A0 à A5. Une broche programmée ainsi peut être :

- **une entrée:** Le programme peut lire une tension présente sur cette broche en utilisant **digitalRead(...)**. Comme cette tension est interprétée comme un **chiffre binaire (0 ou 1)**, alors toute tension **inférieure à 1V sera comprise comme un 0** et que toute **tension supérieure à 3,5V sera comprise comme un 1**.
- **une sortie:** Le programme peut écrire un chiffre binaire, au moyen de **digitalWrite(...)**, chiffre qui dans le programme sont nommées **HIGH pour le 1** et **LOW pour le 0**, qui sera traduit en une tension de 5V pour le 1 et de 0V pour le 0.

Présentation de la Carte ARDUINO

Les entrées-sorties numériques



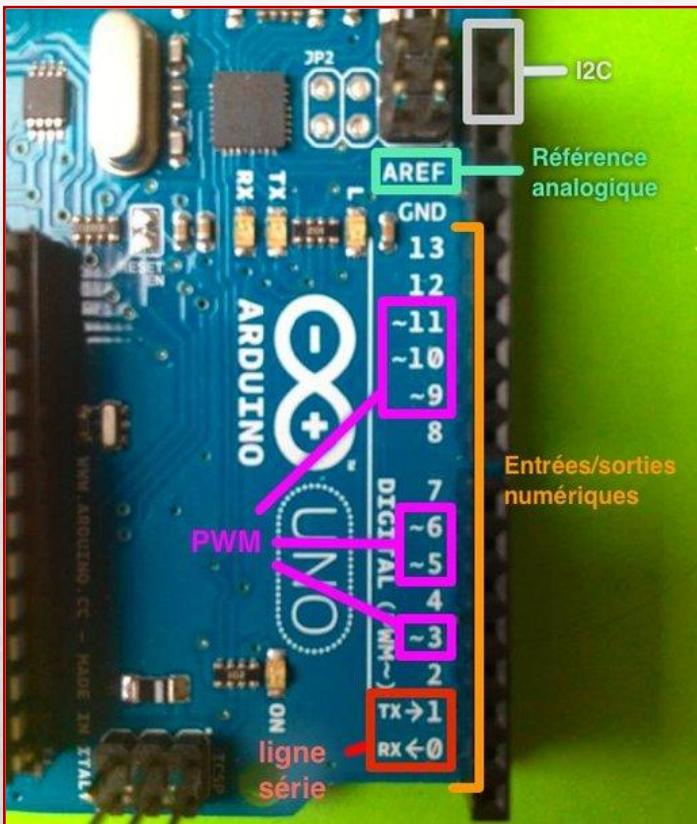
Les broches numérotées de 0 à 13 sont les entrées sorties numériques.

Certaines d'entre elles peuvent également être utilisées en **sor^{tie} PWM** (en rose), les broches 0 et 1 servent aussi à la **communication série** (en rouge). L'**I2C** en gris est en haut et non étiqueté sur la carte.

Il est préférable de ne pas dépasser une consommation de 20 mA sur une broche programmée en sortie et absolument nécessaire de ne pas dépasser 40 mA sous peine de risque la destruction de la sortie du **MCU**. Il s'agit aussi de ne pas dépasser au total 200 mA.

Présentation de la Carte ARDUINO

Les entrées-sorties numériques

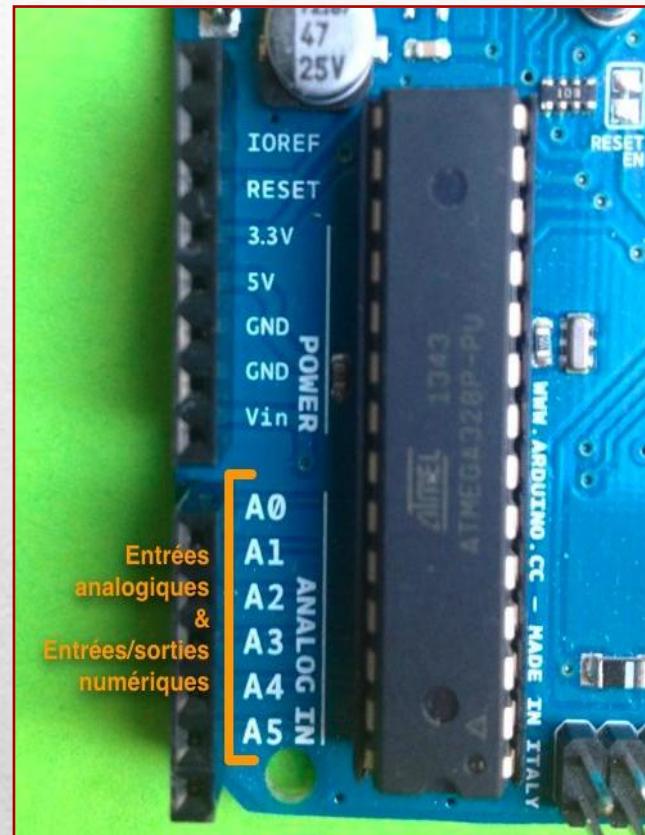


- Les sorties numériques sont employées de nombreuses manières : alimentation d'une DEL pour un éclairage ou un témoin, commande d'un transistor, connexion à un afficheur LCD, etc.
- Les entrées numériques permettent également de connecter des nombreuses choses : interrupteur, bouton poussoir, des capteurs de présence etc...

Présentation de la Carte ARDUINO

Les entrées analogiques

- ARDUINO UNO possède **6 entrées analogiques**. Ce sont les broches étiquetées **A** suivie d'un nombre (A0, A1, A2..., A5).
- Les tensions, toujours entre 0 et 5V, présentes sur ces broches, peuvent être numérisées via **un convertisseur analogique-numérique ou ADC** (Analog Digital Converter).
- La fonction **analogRead(...)** remplit ce rôle.



Présentation de la Carte ARDUINO

Les entrées analogiques

- Le convertisseur des Arduino effectue une conversion **sur 10 bits**, c'est à dire qu'il convertit la tension en un nombre entier ayant une valeur **de 0 à 1023**. 0 correspond au 0V de la tension et 1023 au 5V. **La résolution**, c'est à dire la différence entre deux valeurs successives de la tension correspondant à une différence de 1 sur l'entier résultat, est donc d'**environ 5mV**.
- Il est possible de changer la tension de référence de la conversion analogique-numérique en choisissant la tension fournie sur **la broche AREF** comme **tension de référence**.
- Plusieurs types de **capteurs analogiques** peuvent trouver place sur ces broches. On citera la mesure de courant traversant une résistance ou la mesure d'intensité lumineuse via une photo-résistance, capteur température, humidité etc...

Présentation de la Carte ARDUINO

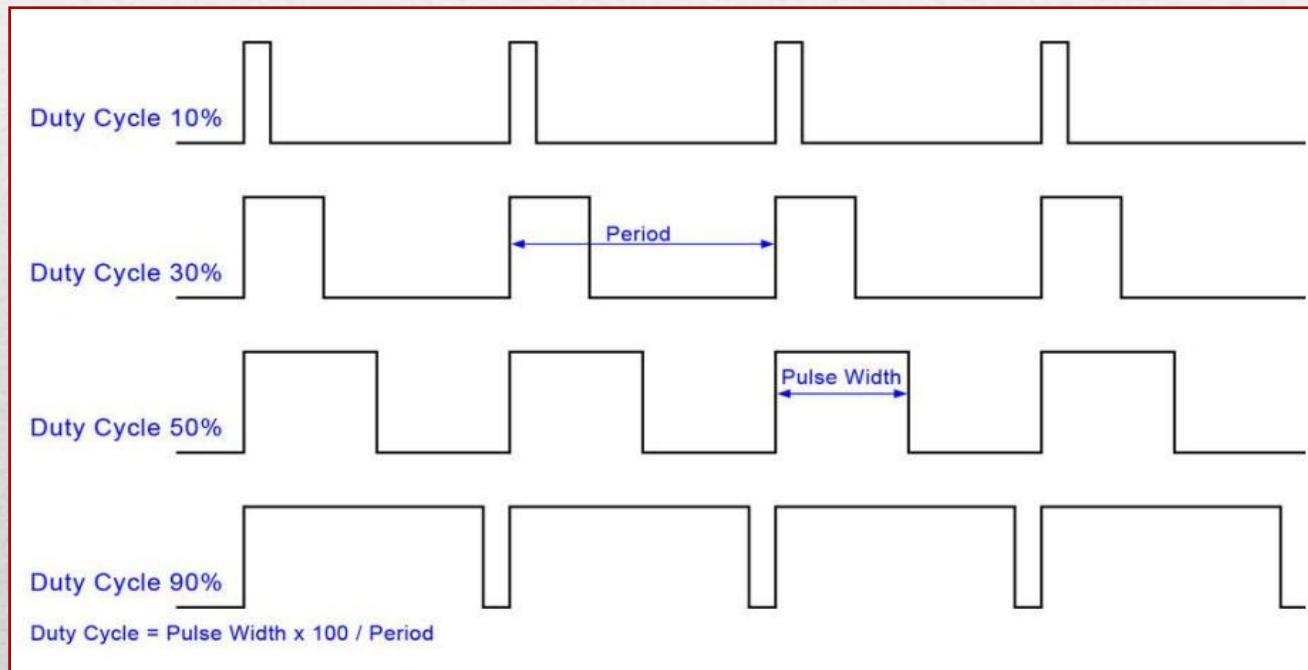
Les sorties PWM (Pulse Width Modulation)

- Ce sont les broches annotées par un tilde ~ sur la carte.
- Les PWM (Pulse Width Modulation) sont utilisées pour **synthétiser des signaux analogiques** en modulant le temps passé à l'état 1 (5V).
- En utilisant une fréquence relativement élevée, les PWM permettent de commander certains composants comme s'ils reçoivent une tension analogique. La plupart des cartes Arduino utilisent des PWM cadencées à 490Hz environ.
- La PWM trouve son emploi lorsqu'il s'agit de commander un moteur électrique à courant continu (variation de vitesse), ou une intensité lumineuse, par exemple dans une DEL.

Présentation de la Carte ARDUINO

Les sorties PWM (Pulse Width Modulation)

Le rapport cyclique (**Duty Cycle**) de la PWM est réglé via la fonction **analogWrite(...)** entre 0, le signal est constamment à l'état bas, 0V, à 255, le signal est constamment à l'état haut, 3,3V ou 5V selon l'Arduino, soit un rapport cyclique de 100%.



Présentation de la Carte ARDUINO

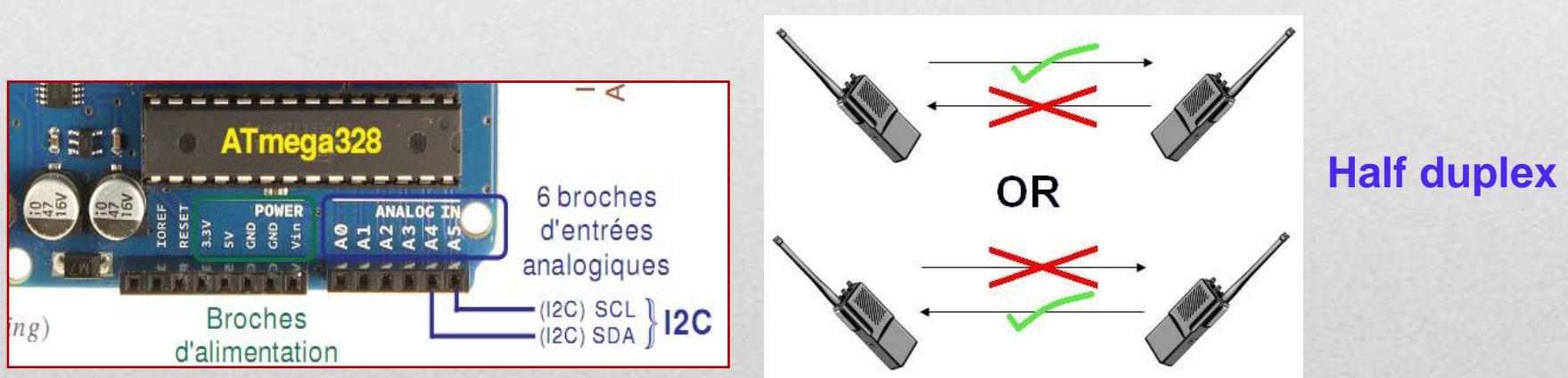
La ligne série

- Les 2 broches étiquetées **TX et RX** sont respectivement la ligne d'émission série (T pour Transmit, émettre) et la ligne de réception série (R pour receive, recevoir). Cette ligne est principalement employée pour dialoguer avec l'ordinateur hôte, votre Mac ou votre PC. Des fonctions permettent **d'afficher des messages dans une fenêtre de l'ordinateur hôte et de lire le clavier de l'ordinateur hôte.**
- Elle est également employée pour dialoguer avec certains modules comme **les modules radio XBee ou certains modules WiFi ou encore bluetooth.**

Présentation de la Carte ARDUINO

Le bus I2C (Inter-Integrated Circuit)

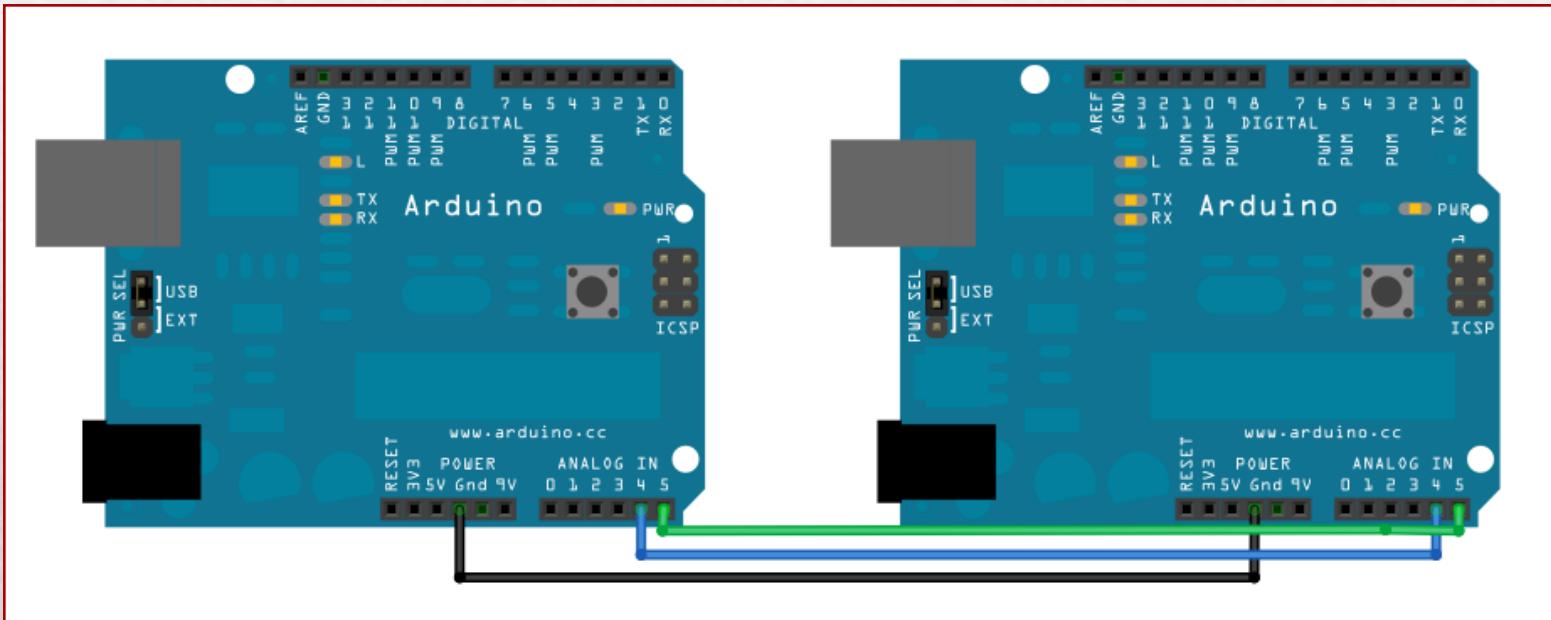
- l'I2C est l'un des bus de communication série synchrone bidirectionnel half-duplex utiliser pour dialoguer soit avec un autre Arduino soit avec des circuits périphériques disposant de ce bus.
- 2 broches sont employées pour l'I2C: l'horloge (SCL) et donnée (SDA).



Présentation de la Carte ARDUINO

Le bus I2C (Inter-Integrated Circuit)

Il faut intégrer la bibliothèque **I2C Wire** dans le code pour utiliser les fonctions en relation avec la communication I2C



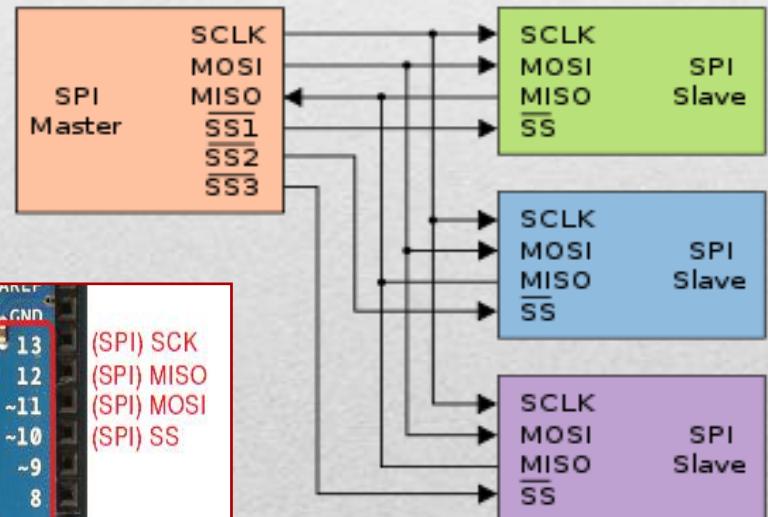
Présentation de la Carte ARDUINO

Le bus SPI

- Le SPI est également un bus de communication série synchrone bidirectionnel destiné à dialoguer avec les circuits périphériques. Il opère en mode Full-duplex. Il permet un débit de communication plus important.
- 4 broches sont employées pour le SPI, l'horloge (**SCK**), les données en sortie (**MISO**), les données en entrée (**MOSI**) et la sélection du circuit avec lequel l'Arduino veut dialoguer(**SS**).



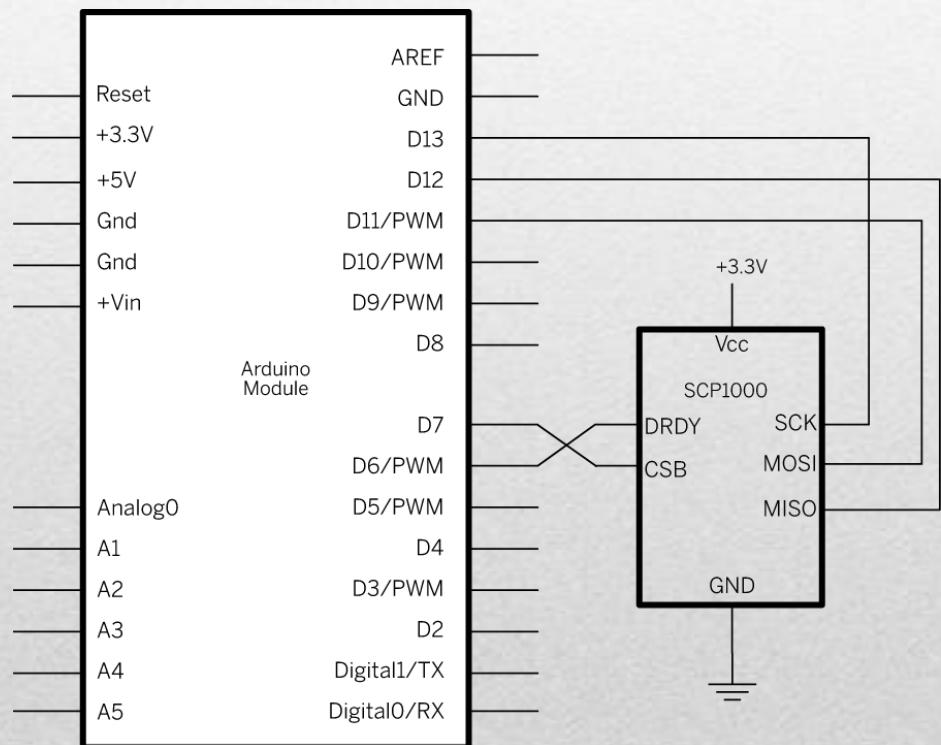
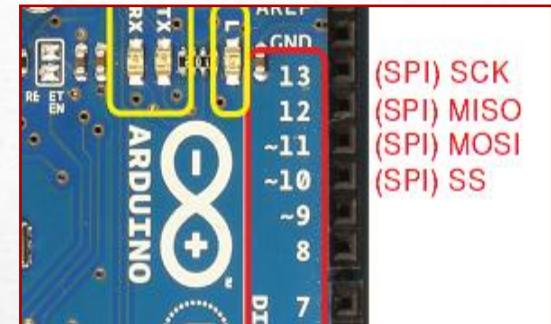
Full duplex



Présentation de la Carte ARDUINO

Le bus SPI

- Le capteur de pression barométrique SCP1000 peut lire à la fois la pression de l'air et de la température et les envoyer à la carte ARDUINO via la connexion SPI
- Il faut intégrer la bibliothèque [Arduino SPI Library](#) pour utiliser les fonctions adaptées à la communication SPI



Présentation de la Carte ARDUINO

Les broches d'alimentation

- IOREF: cette broche est destinée à indiquer aux shields la tension de fonctionnement de l'Arduino. Sur un Arduino 5V, elle aura une tension de 5V et sur un 3,3V une tension de 3,3V
- RESET: cette broche permet de réinitialiser l'Arduino et donc de redémarrer le programme au début. Pour réinitialiser l'Arduino, il suffit de mettre cette broche à 0V puis de la repasser à 5V, ou 3,3V selon l'Arduino.
- 3.3V: on peut s'en servir pour alimenter un circuit externe en 3,3V.
- 5V: utiliser pour alimenter des circuits externes en 5V.
- GND: la masse ou 0V
- Vin: on peut choisir d'alimenter l'Arduino via cette broche au lieu d'utiliser la prise d'alimentation.

Présentation de la Carte ARDUINO

Un atout : les shields

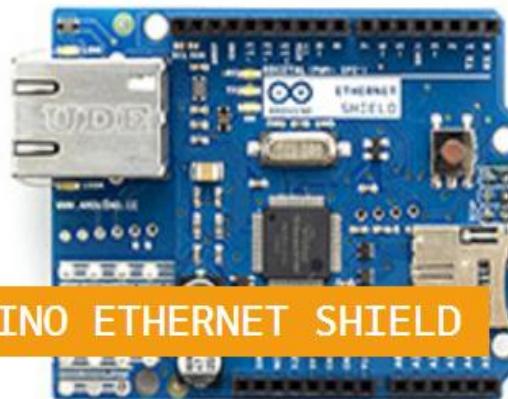
- Pour la plupart des projets, il est souvent nécessaire d'ajouter des fonctionnalités aux cartes Arduino. Plutôt que d'ajouter soit même des composants extérieurs (sur une platine d'essai, circuit imprimé, etc.), il est possible d'ajouter des shields.
- Un shield est une carte que l'on connecte directement sur la carte Arduino qui a pour but d'ajouter des composants sur la carte. Ces shields viennent généralement avec une librairie permettant de les contrôler.
- On retrouve par exemple, des shields Ethernet, de contrôle de moteur, lecteur de carte SD, etc.
- Le principal avantage de ces shields est la simplicité d'utilisation. Il suffit de les emboîter sur la carte Arduino pour les connecter.

Présentation de la Carte ARDUINO

Un atout : les shields



ARDUINO GSM SHIELD



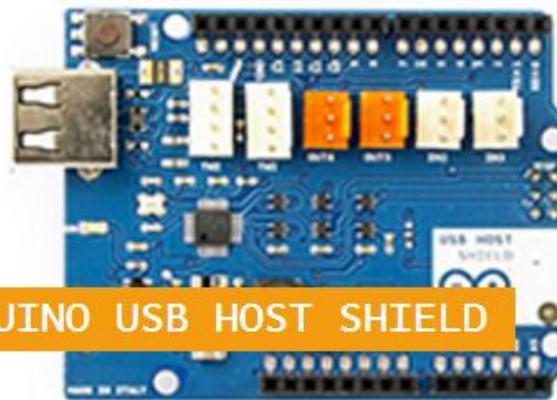
ARDUINO ETHERNET SHIELD



ARDUINO YÙN Shield



ARDUINO WIFI SHIELD



ARDUINO USB HOST SHIELD



ARDUINO MOTOR SHIELD

Présentation du logiciel

Présentation du logiciel

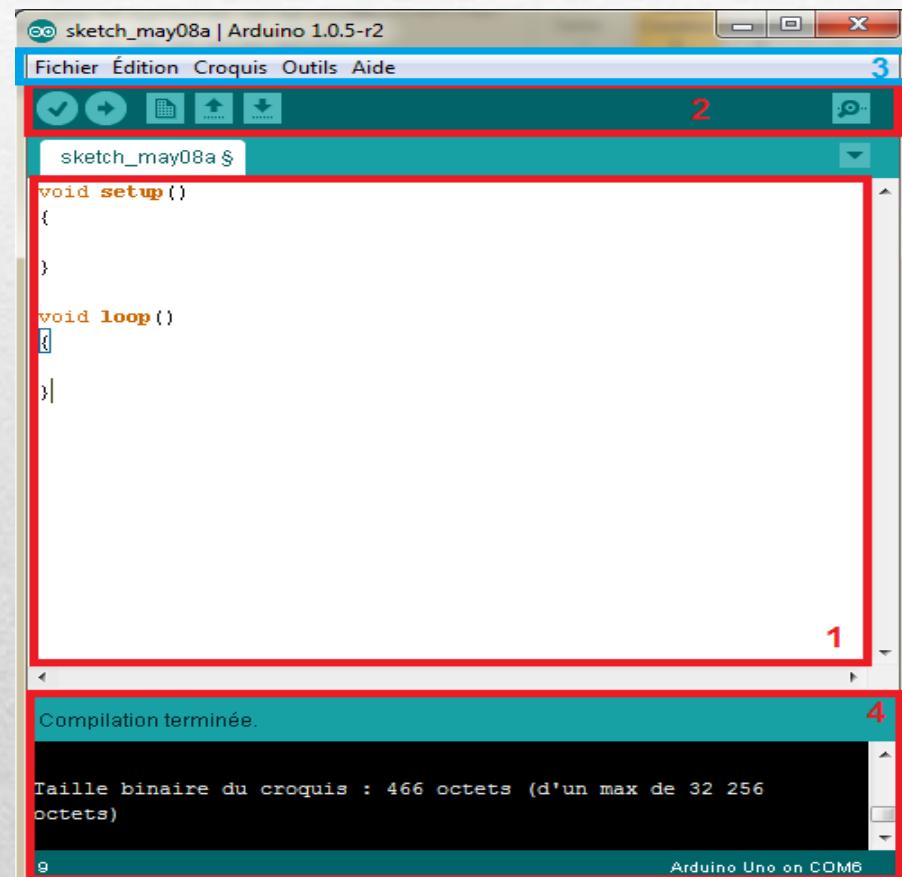
IDE ARDUINO

L'IDE (environnement de développement) est libre et gratuit et distribué sur le site d'Arduino

<http://arduino.cc/en/main/software>.

L'interface de l'IDE Arduino est plutôt simple comportant:

1. Un éditeur de code
2. Une barre d'outils rapide
3. Une barre de menus classique qui est utilisée pour accéder aux fonctions avancées de l'IDE
4. Une console affichant les résultats de la compilation du code source, des opérations sur la carte



Présentation du logiciel

Langage ARDUINO

Le langage Arduino est inspiré du langage C/C++.

La structure minimale d'un code ARDUINO est constituée :

- une 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 contiendra toutes les opérations nécessaires à la configuration de la carte (directions des entrées sorties, débits de communications série, etc.)
- une loop (boucle) : cette partie est lue en boucle ! C'est ici que les fonctions sont réalisées.
- des « sous-programmes » ou « routines » qui peuvent être appelées à tous moments dans la boucle.

Présentation du logiciel

Langage ARDUINO



The screenshot shows the Arduino IDE interface. The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar contains icons for saving, running, and uploading code. The title bar says "BareMinimum §". The code editor displays the following sketch:

```
File Edit Sketch Tools Help
BareMinimum §
int valeur = 0; // déclaration d'une variable

void setup() {
    // partie du code qui ne sera lue qu'une fois
    pinMode(13, OUTPUT);
}//fin du setup

void loop() {
    // partie du code qui sera lue en boucle
    valeur = analogRead(0);
    if(valeur > 900){
        digitalWrite(13, HIGH);
    }
    else{
        digitalWrite(13, LOW);
    }
}//fin de la boucle

Done compiling.
```

The status bar at the bottom right indicates "Done compiling."

Déclaration des variables

La déclaration des variables se fait généralement dans **l'espace global** (de façon à partager les variables les plus importantes entre les deux fonctions principales). On retrouve les types de base suivant :

Nom	Contenu	Taille (en octet)	Plage de valeurs
<i>(unsigned) char</i>	Entier ou caractère	1	(0->255) -128 -> 127
<i>(unsigned) int</i>	Entier	2	(0->65 535) -32 768 -> 32 767
<i>(unsigned) long</i>	Entier	4	(0 -> 4 294 967 295) -2 147 483 648 -> 2 147 483 647
<i>float/double</i>	Nombre à virgule flottante	4	-3,4028235E+38 -> 3,4028235E+38
<i>String</i>	Chaine de caractères (Objet)	variable	Aucune
<i>boolean</i>	Booléen	1	True / False

```
const int constante = 12 ;  
float univers = 42.0 ;  
char lettre = 'b' ;  
String chaine = "Hello World " ;  
long tableau[12] ;  
boolean vrai = true ;
```

L'opérateur d'affectation : L'opérande de gauche prend pour valeur l'opérande de droite

Type	Symbole	Exemple
Opérateur d'affectation	=	X=8 ; Y = a + c

Les opérateurs arithmétiques :

Type	Symbole	Exemple
Addition	+	a = a + b ; x = 7 + a
Soustraction	-	a = a - b ; y = z - 8
Moins unaire	-	a = - b
Multiplication	*	a = b * c ; b = y * 3
Division	/	c = 6 / b ; d = a / b
Reste de la division entière	%	r = a % b

Langage ARDUINO

Les opérateurs de comparaison ou relationnels :

Type	Symbole	Exemple
Egalité	<code>==</code>	<code>a == b ; c == 9</code>
Different	<code>!=</code>	<code>c != a</code>
Supérieur	<code>></code>	<code>a > b ; 8 > a</code>
Supérieur ou égal	<code>>=</code>	<code>a >= b ; 8 >= a</code>
Inférieur	<code><</code>	<code>a < b ; 8 < a</code>
Inférieur ou égal	<code><=</code>	<code>a <= b ; 8 <= a</code>

Les opérateurs logiques :

Type	Symbol
Et logique	<code>&&</code>
Ou logique	<code> </code>
Non logique	<code>!</code>

Langage ARDUINO

Les opérateurs binaires bit à bit :

Type	Symbole	Exemple
Et binaire	&	$x = y \& z$
Ou binaire		$x = y z$
Ou exclusif	^	$x = y ^ z$
Complément à 1	~	$x = \sim z$
Décalage de n bits à droite	>>	$x = b >> n$
Décalage de n bits à gauche	<<	$x = b << n$

Langage ARDUINO

Les structures algorithmiques:

Structure “while” : tant que ... faire ...	Structure “do ... while” : faire ... tant que...	Structure “for” : Pour <variable> allant de <valeur initiale> à <valeur finale> faire...	Structure “if ... Else” : Si <condition> faire ... sinon faire ...	Structure “switch ... case”.
void main() { while (condition vraie) { Action 1; } }	void main() { do { Action ; } while (condition vraie) }	void main() { For (i=m ; i<=n ; i=i+p) { Action ; }	void main() { if (condition vraie) { Action 1; } else { Action 2; }	switch (choix) case c1 : < sequence 1> case c2 : < sequence 2> case c3 : < sequence 3> case cn : < sequence n> default : < sequence_par_defaut> /

Fonctionnalités de base

Fonctionnalités de base

Les entrées sorties

Dans un premier temps, avant d'effectuer une mesure ou d'envoyer une commande sur une broche (pin), il est nécessaire de définir la direction des broches utilisées. Pour cela on fait appel à la fonction **pinMode** en lui donnant d'une part, la broche concernée, et d'autre part, la direction :

```
void setup() {  
    pinMode(1,OUTPUT) ; // Broche 1 en sortie  
    pinMode(2,INPUT) ; // Broche 2 en entrée  
}
```

Fonctionnalités de base

Les entrées sorties

Toutes les fonctionnalités sur les broches d'entrées sorties sont utilisables par le biais de quatre fonctions :

- **digitalRead(pin)** : mesure une donnée numérique sur une des broches, la broche en question doit être réglée en entrée.
- **digitalWrite(pin, value)** : écrit une donnée numérique sur une des broches, la broche concernée doit être réglée en sortie. Le paramètre value doit être égal à HIGH (état 1 soit 5V) ou LOW (état 0 soit 0V).
- **analogRead(pin)** : mesure une donnée analogique sur une des broches (compatible seulement), la broche doit être réglée en entrée.
- **analogWrite(pin, value)** : écrit une donnée sous forme de PWM sur une des broches (compatible uniquement), la broche doit être réglée en sortie. Le paramètre value doit être compris dans l'intervalle [0;255]

Fonctionnalités de base

La gestion du temps

Pour la plupart des applications de domotique, il est nécessaire de faire intervenir des intervalles de temps. Par exemple, pour gérer le temps d'appui sur un bouton ou pour faire une sonnerie qui se répète un certains nombre de fois. Le langage Arduino fournis quelques fonctions permettant de gérer le temps telles que

- **Delay(ms)**
- **delayMicroseconds(μs)**

Elles insèrent **une pause** suivant le paramètre passé (en milliseconde pour l'un, en microseconde pour l'autre). **Ces fonctions bloquent le microcontrôleur**, on ne peut alors plus effectuer aucune action.

Fonctionnalités de base

La gestion du temps

Il est possible de mesurer le temps. De la même manière que les fonctions de délai, on utilise les fonctions:

- **Unsigned long millis()**
- **Unsigned long micros()**

Ces fonctions donnent le nombre de millisecondes (respectivement microseconde) depuis le lancement de la carte.

Fonctionnalités de base

Les Interruptions

Une **interruption** est un signal permettant d'interrompre temporairement le fonctionnement normal d'un programme pour exécuter une routine spécifique (routine d'interruption).

une **interruption** est un arrêt temporaire de l'exécution normale d'un programme informatique par le microprocesseur afin d'exécuter un autre programme (appelé service d'interruption ou routine d'interruption).

Ce signal ou bien arrêt est déclenché lorsque qu'un **événement (interne ou externe) survient** et à besoin d'être traité sur le champ.

Il faut cependant faire attention, ce mécanisme **interrompt le code exécuté, il est prioritaire par rapport au reste du code.**

Fonctionnalités de base

Les Interruptions

On utilise généralement les interruptions pour des **codes critiques** (arrêt d'urgence par exemple) ou **des événements non-ponctuels** (transmissions de données depuis un ordinateur par exemple) ou bien **faire des choses d'une façon automatique** afin de résoudre le problème de temporisation

.

Fonctionnalités de base

Les Interruptions

Il y a deux sources d'interruption:

- Interruption externe : changement d'état d'une broche

Le nombre d'interruptions externes est limité à 2: INT0 et INT1 (**broches 2 et 3**) sur l'Arduino UNO. Pour choisir la fonction et la broche utilisée pour l'interruption, on utilise la fonction **attachInterrupt**. On peut utiliser **detachInterrupt** pour supprimer l'interruption. Il est possible de partir en interruptions sur 4 types d'événements :

- **LOW** : Lorsque la broche est à l'état 0 (0V)
- **RISING** : Lorsque la broche passe de l'état 0 (0V) à l'état 1 (5V) (front montant).
- **FALLING** : Lorsque la broche passe de l'état 1 (5V) à l'état 0 (0V) (front descendant).
- **CHANGE** : Lorsque la broche change d'état (front montant et front descendant).

Fonctionnalités de base

Les Interruptions

Il y a deux sources d'interruption:

- Interruption interne

- RESET
- Timers
- ADC
- Watchdog
- Comparateur . . .

Fonctionnalités de base

Les Interruptions (Exemple)

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(interruptPin, INPUT_PULLUP);
    attachInterrupt(0, blink, CHANGE); // 0 c'est INT0 sur Pin2
}
void loop() {
    digitalWrite(ledPin, state);
}

void blink() {
    state = !state;
}
```

Fonctionnalités de base

Communication série

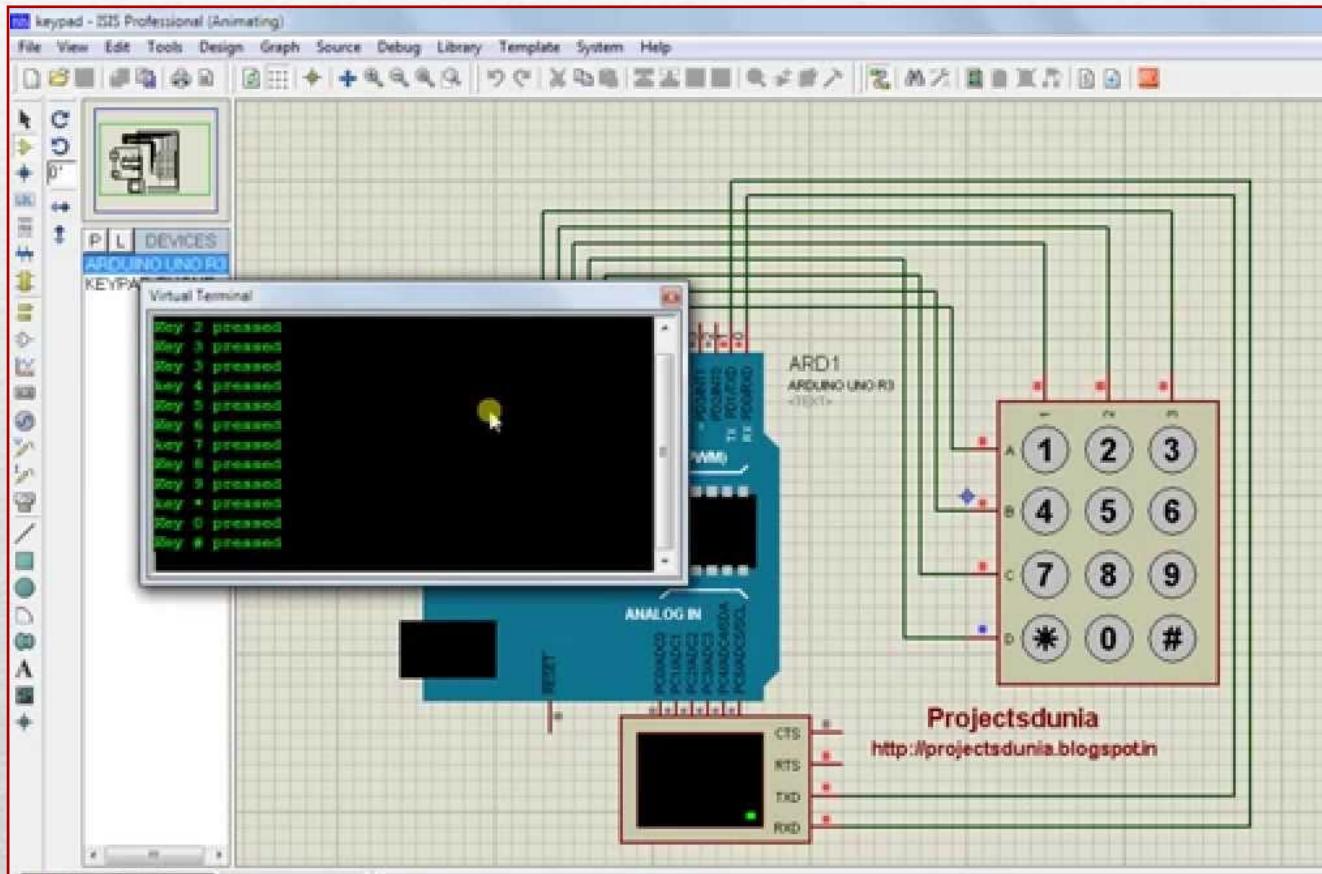
Communications série entre Arduino et autres machines ou ordinateur :

- **Serial.begin(speed)** (configuration de la vitesse de communication Série)
- **Serial.available()** (donne combien de caractères disponibles dans la zone tampon Série)
- **Serial.read()** (lit les données Série)
- **Serial.print(data)** (envoie des données Série)
- **Serial.println(data)** (envoie des données Série suivies de caractères spécifiques)

Simulateurs pour ARDUINO

Utilisation d'ARDUINO avec ISIS

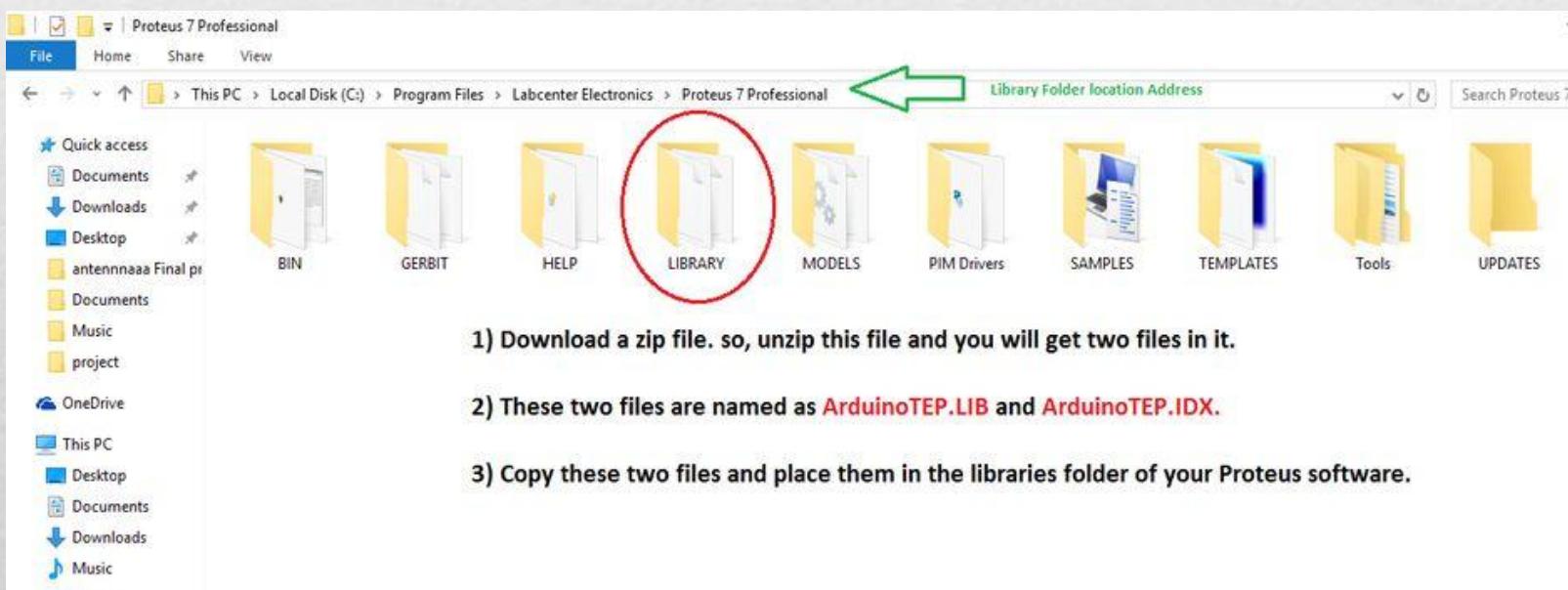
Pour simuler vos applications sans avoir la carte ARDUINO, on peut se servir du logiciel ISIS en faisant appel à la librairie ARDUINO.



Utilisation d'ARDUINO avec ISIS

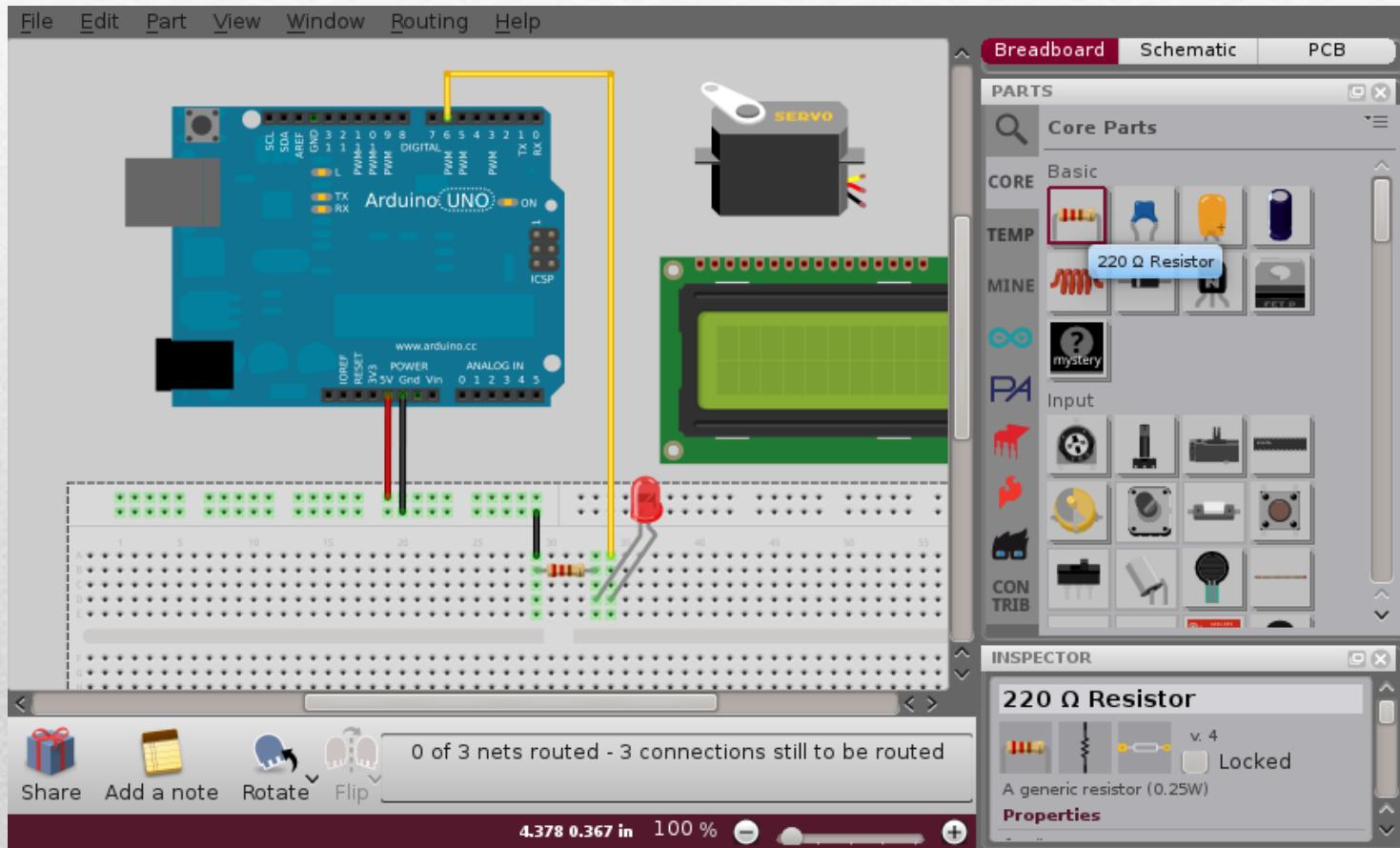
Ajouter la librairie ARDUINO à ISIS

- Télécharger la bibliothèque d'ARDUINO pour ISIS à partir du lien suivant: <http://www.instructables.com/id/How-to-add-Arduino-Library-in-to-Proteus-7-8/>
- Suivre les étapes indiquées dans la figure ci-dessous



L'outil Fritzing (pas de simulation)

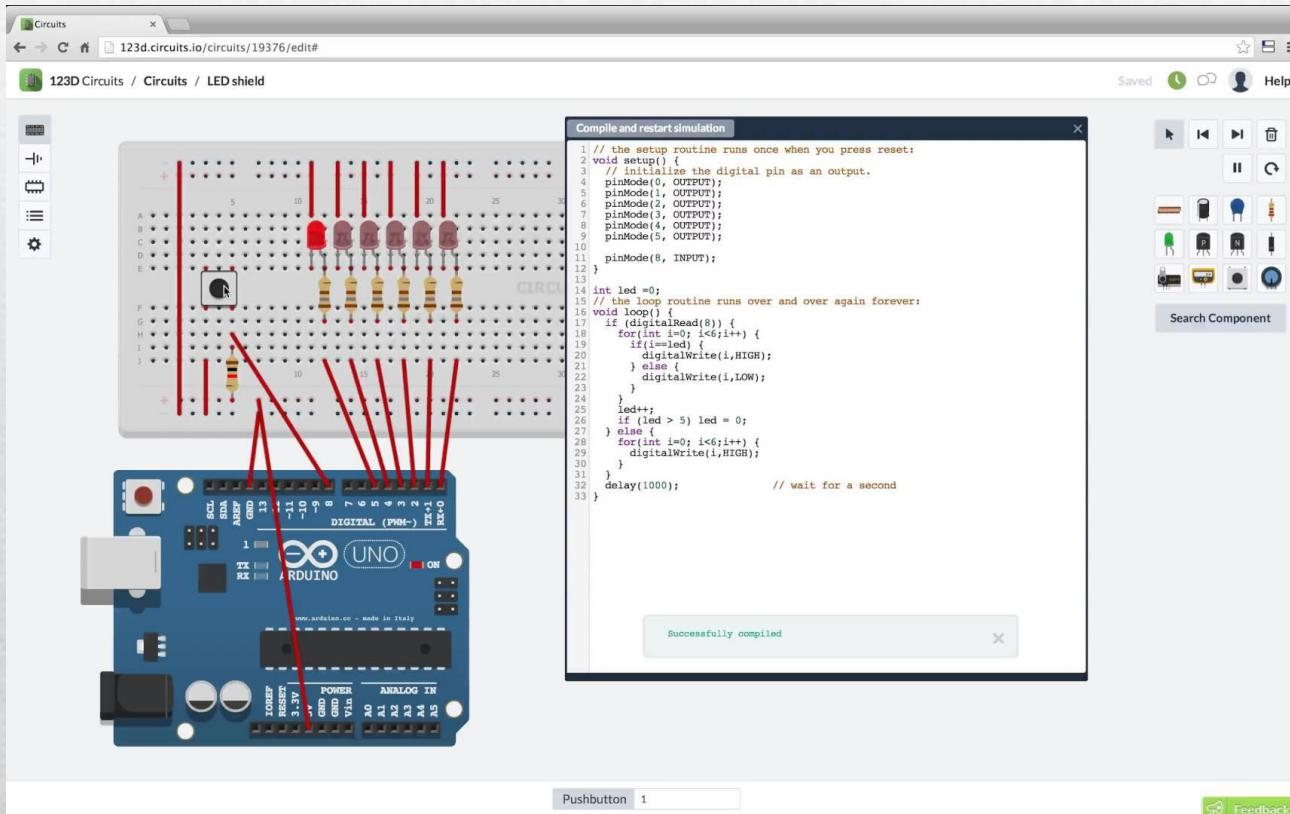
Fritzing est un logiciel libre de conception de circuit imprimé permettant de concevoir de façon entièrement graphique le circuit et d'en imprimer le typon.



123D Circuits.io

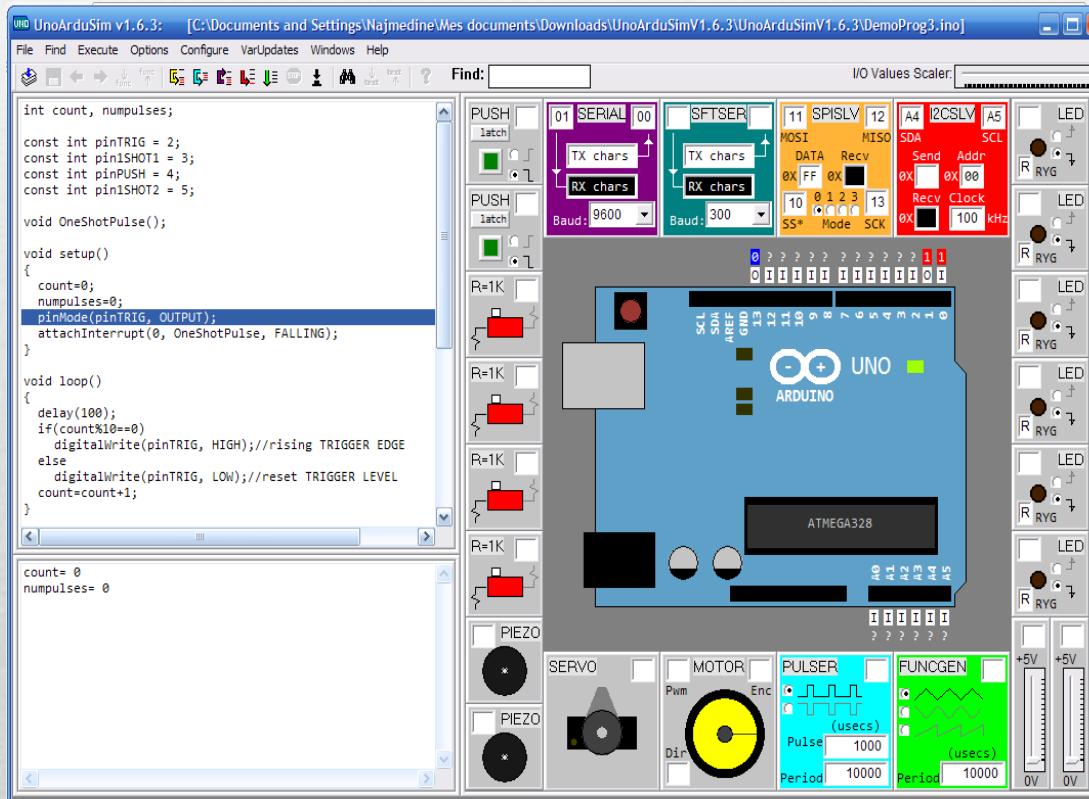
C'est un outils de conception électronique en ligne totalement gratuit.

Le principal avantage de cette solution est la possibilité de concevoir un montage un peu comme avec Fritzing. Il est ensuite **possible de simuler** son montage en y intégrant même du code arduino pour ensuite finaliser en routant son PCB.



L'outil UnoArduSim

UnoArduSim fournit une carte Uno virtuelle et permet à l'utilisateur de choisir et d'interface avec un ou plusieurs types de périphériques d'entrée / sortie virtuels (moteur DC, moteur pas à pas, servomoteurs, périphériques de communication rs232, I2C SPI, haut-parleurs piézoélectriques, LED, boutons-poussoirs, résistances pull-ups et pulldowns et potentiomètre. Tous les appareils sont électriquement (et mécaniquement) modélisées.



Exercices d'application

Exercice d'application

Clignoter une LED sur la pin 13 de la carte ARDUINO

Code avec *delay* :

```
const int pinLed = 13;

void setup()
{
    pinMode(pinLed, OUTPUT); // Broche 13 en sortie
}

void loop()
{
    delay(500); // Attente d'une demi seconde
    digitalWrite(pinLed, HIGH); // Allumage de la LED
    delay(500);
    digitalWrite(pinLed, LOW); // Eteignage de la LED
}
```

Exercice d'application

Clignoter une LED sur la pin 13 de la carte ARDUINO

Code avec *millis* :

```
const int pinLed = 13;
int temps;
int etat;

void setup()
{
    pinMode(pinLed, OUTPUT); // Broche 13 en sortie
    etat = LOW; // LED éteinte
}

void loop()
{
    int present = millis();

    if ( temps+500 < present ) // Vérification du chronomètre
    {
        temps = present; // Actualisation du chronomètre
        etat = !etat; // Changement d'état
        digitalWrite(pinLed,etat); // Changement d'état de la LED
    }
}
```

Exercice d'application

Commander une LED (pin13) par un bouton (pin2)

Code sans interruptions :

```
const int pinLed = 13;
const int pinBouton = 2;

void setup()
{
    pinMode(pinLed, OUTPUT); // Broche 13 en sortie
    pinMode(pinBouton, INPUT); // Broche 2 en entrée
}

void loop()
{
    int etat = digitalRead(pinBouton); // On lit la valeur du bouton
    digitalWrite(pinLed,etat); // On allume (ou non) la LED

    delay(50); // On peut mettre un delay pour éviter les allumages
               // intenpestifs
}
```

Exercice d'application

Commander la luminosité d'une LED (pin11) par un potentiomètre (pin A0)

```
const int pinLed = 11;
const int pinPotar = A0;

void setup()
{
    pinMode(pinLed, OUTPUT); // Broche 11 en sortie (broche PWM)
    pinMode(pinPotar, INPUT); // Broche A0 en entrée (broche analogique)
}

void loop()
{
    int valeur = analogRead(pinPotar); // Lecture de la valeur du
potentiomètre

    valeur = map(valeur,0,1023,0,255); // Change l'intervalle de valeur
[0;1023]->[0;255]

    analogWrite(pinLed,valeur); // Écriture sur la broche 11 en PWM
    delay(50);
}
```

Fin du cours

Merci pour votre attention



Travaux Pratiques

ARDUINO

Préparé par : M. Nejmeddine Bahri

Année Universitaire : 2019/2020

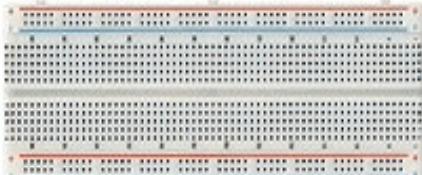
TP1

Gestion des entrées/sorties numériques

I. Objectifs de TP

- Se familiariser avec l'environnement de développement Arduino IDE
- Gérer les entrées sorties numériques de la carte Arduino Uno

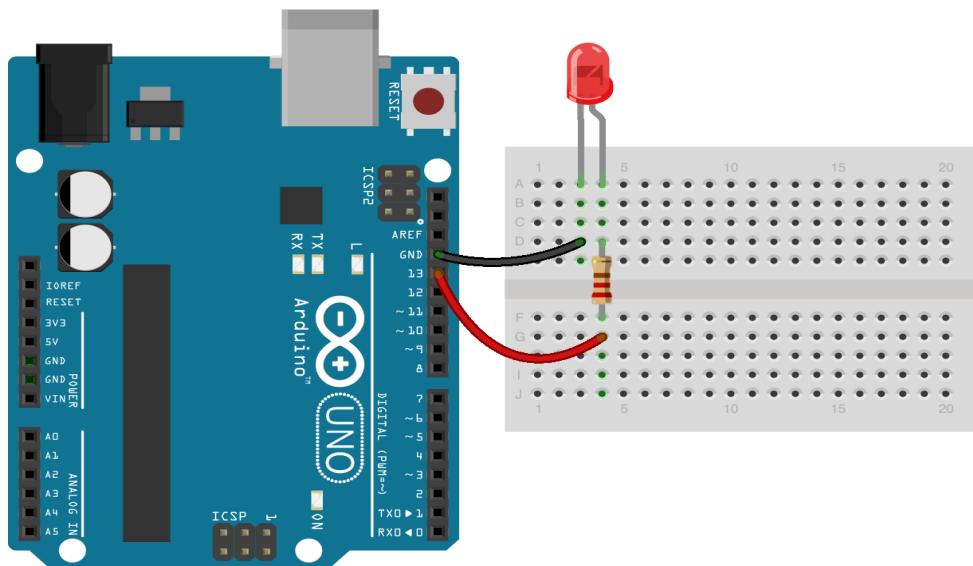
II. Matériaux

 carte ARDOUINO	 plaquette d'essai
 3 Diodes LED	 résistances 3x470 Ω /2x10k Ω
 1 Bouton poussoir	 Des fils de connexion

III. Partie Pratique

Exercice 1 : clignotement d'une LED

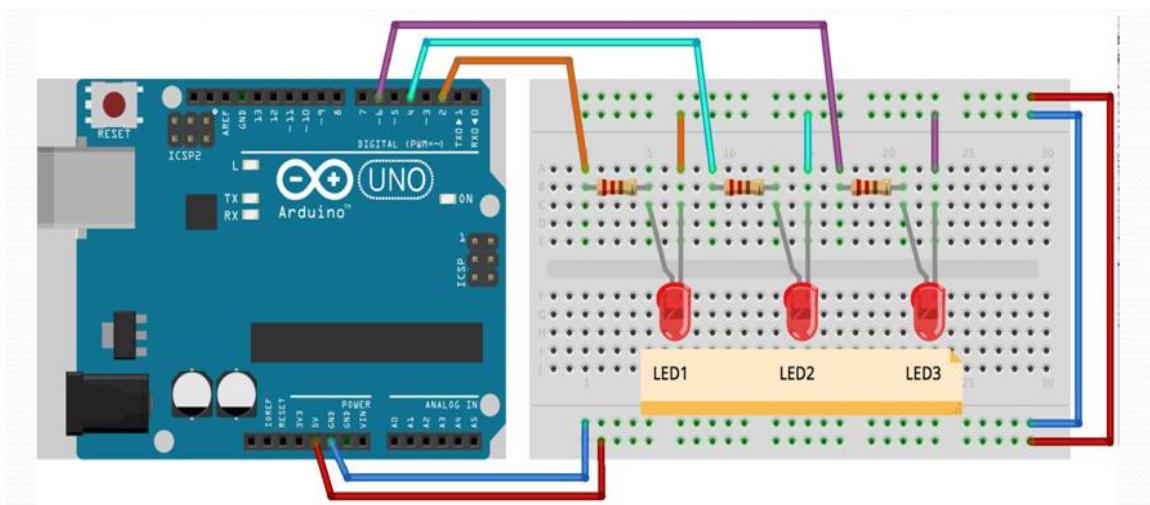
1. Réaliser le montage de la figure ci-dessous en notant que la valeur de la résistance est 220/470 Ω



2. Ecrire le programme Arduino qui permet de clignoter la LED branchée sur la broche 13 de la carte Arduino Uno avec une fréquence de 1 Hz
3. Augmenter la fréquence de clignotement à 5 Hz

Exercice 2 : Chenillard (Rotation de lumière)

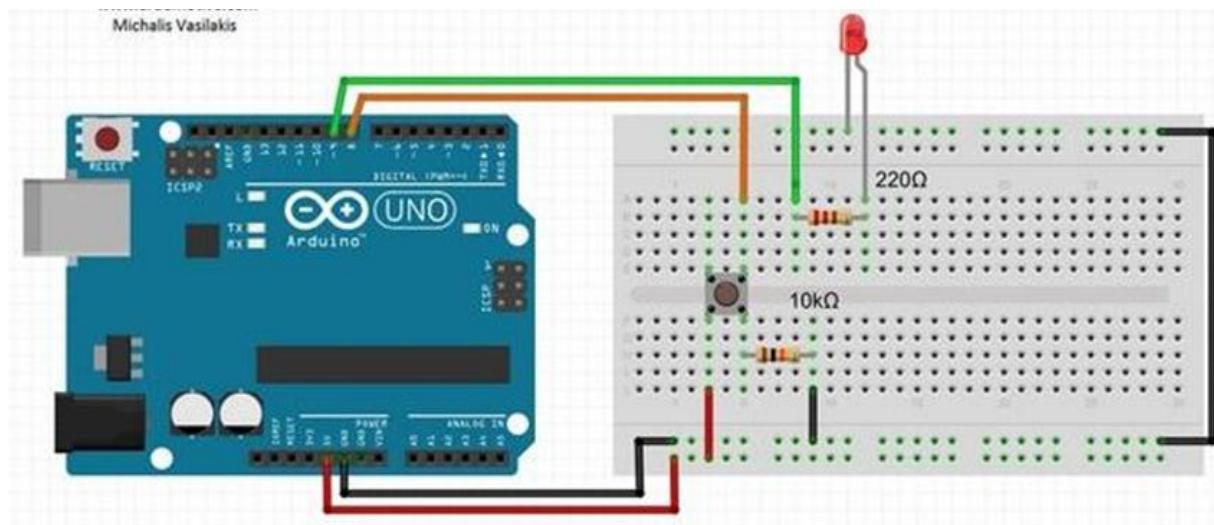
1. Réaliser le montage de la figure ci-dessous



2. Ecrire le programme Arduino qui permet d'allumer les LEDs une par une avec un retard de 0,5s. (**NB : toujours une seule LED est allumée**).

Exercice 3 : commander l'état d'une LED par un bouton poussoir

1. Réaliser le montage de la figure ci-dessous



2. Ecrire le programme Arduino qui permet d'allumer la LED si le bouton poussoir est appuyé et de l'éteindre dans le cas inverse.
3. Inverser la polarité du bouton poussoir en reliant la résistance de $10\text{ k}\Omega$ à 5V et la deuxième borne du bouton poussoir à la masse GND
4. Changer le programme C pour garder le même fonctionnement du montage (LED allumée si le bouton poussoir est appuyé).

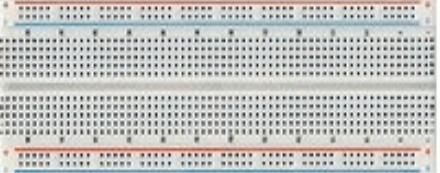
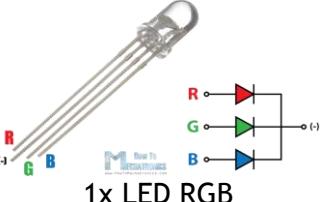
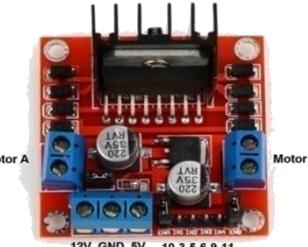
TP2

Gestion des entrées/sorties analogiques

I. Objectifs de TP

- Gérer les entrées/sorties analogiques de la carte Arduino Uno
- Mesurer la tension aux bornes d'une charge résistive
- Varier la luminosité d'une LED par un potentiomètre
- Utiliser la programmation fonctionnelle pour commander une LED RGB
- Commander un moteur à courant continu (sens de rotation et variation de vitesse)

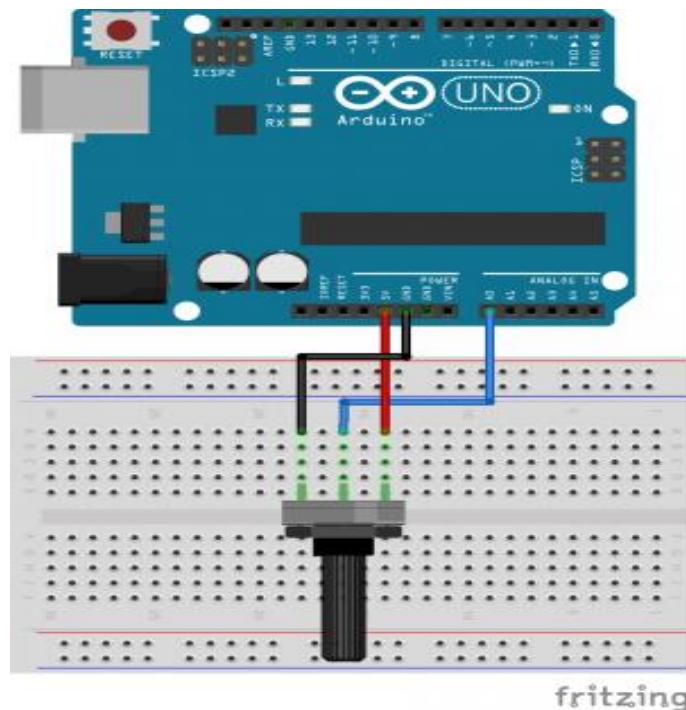
II. Matériels

	
Une carte ARDOUINO UNO	Une plaque d'essai
	 1x LED RGB
2x DIODE LED	
	
3x Resistance 220/470 Ω	2x Bouton poussoir
	
1x Potentiomètre 1KΩ	Des fils de connexion
	
1x Moteur DC 5V	1x Pont H L298N

III. Partie Pratique

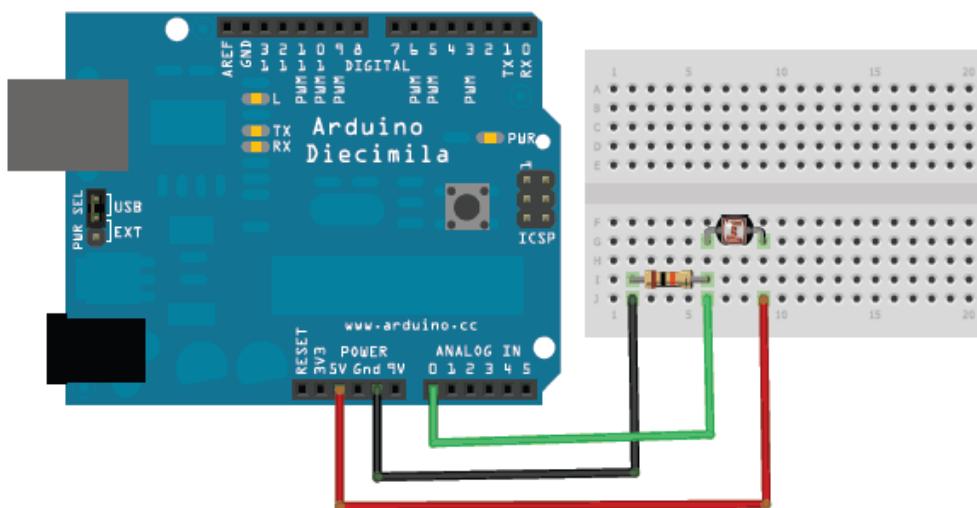
Exercice 1 : mesure de la tension aux bornes d'une charge résistive (potentiomètre)

1. Réaliser le montage de la figure ci-dessous.



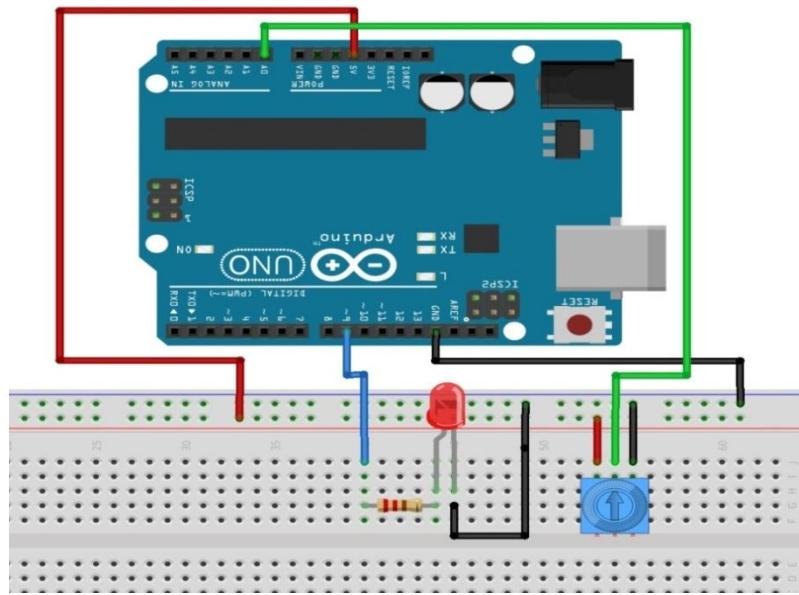
fritzing

2. Ecrire le programme Arduino qui permet de mesurer la tension lue sur la broche A0.
3. Afficher sur le moniteur série la valeur de la tension mesurée.
4. Remplacer le potentiomètre par une photorésistance comme indiqué dans la figure ci-dessous sachant que la résistance utilisée est de $1\text{K}\Omega$.
5. Ecrire le programme Arduino qui permet de déterminer la valeur de la résistance de la photorésistance en lumière et en obscurité.



Exercice 2 : varier la luminosité d'une LED par un potentiomètre

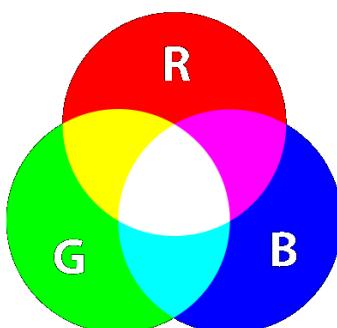
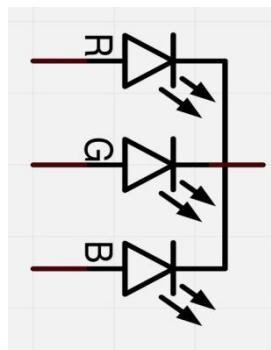
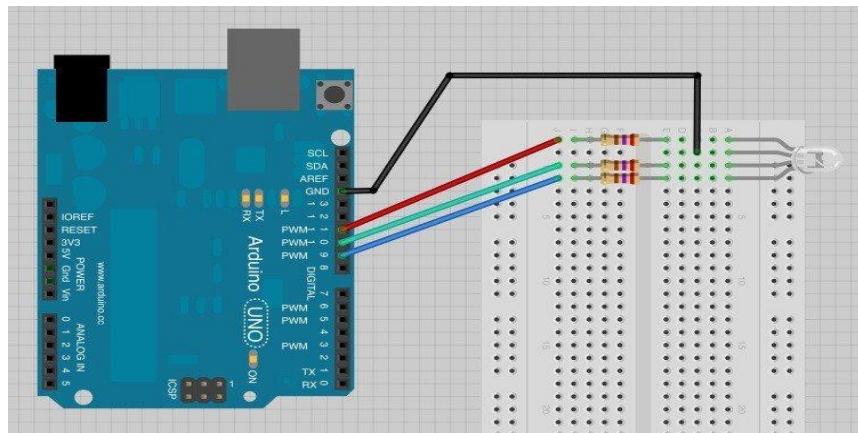
1. Réaliser le montage de la figure ci-dessous.



2. Ecrire le programme Arduino qui permet de varier la luminosité de la LED suite à la variation du potentiomètre. La LED est branchée sur une broche PWM.
3. Afficher sur le moniteur série la tension fournie à la LED.

Exercice 3 : Programmation fonctionnelle

1. Réaliser le montage de la figure ci-dessous en prenant des résistances de 470/220Ω



2. Ecrire le programme Arduino qui permet de visualiser une variété de couleurs sur la LED RGB.

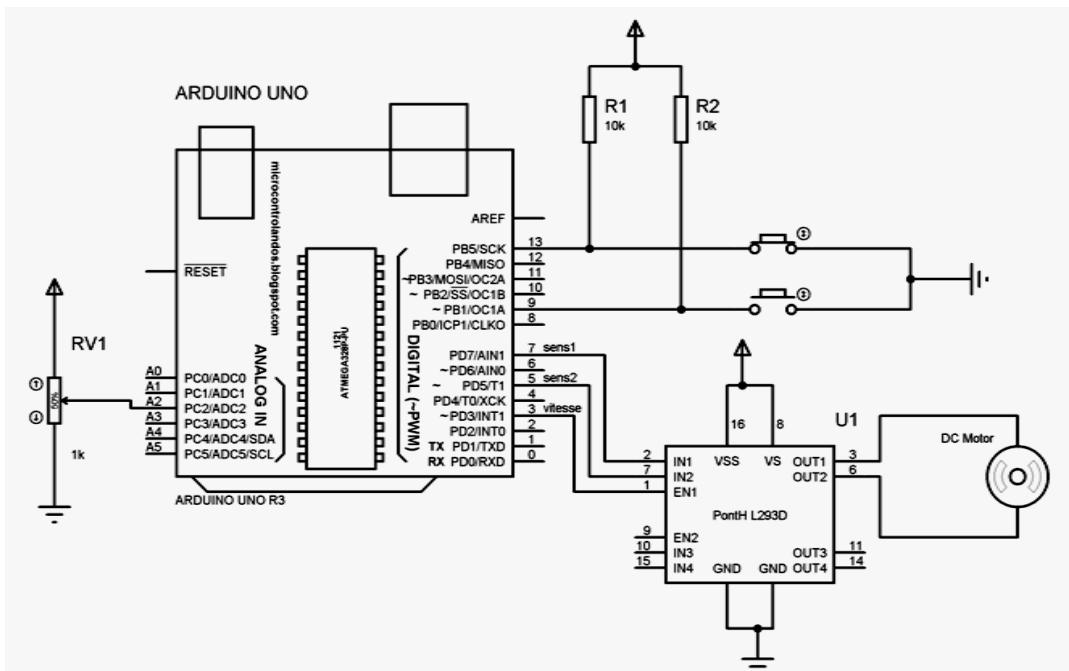
On vous demande de déclarer une **fonction** qui prend comme paramètres l'intensité de couleur souhaitée pour chaque LED (R, G, et B) et faire appel à cette fonction dans la structure loop.

Exemple: **setcolor** (int Red_val, int Green_val, int Blue_val)
 {

 }

Exercice 4 : Commander un moteur à courant continu (sens de rotation et variation de vitesse)

1. Réaliser le montage de la figure ci-dessous.



2. Ecrire le programme Arduino qui permet de commander un moteur à courant continu selon le cahier des charges décrit ci-dessous :
- Si le bouton1 lié à la broche 13 est appuyé, le moteur tourne dans le sens1
 - Si le bouton2 lié à la broche 9 est appuyé, le moteur tourne dans le sens2
 - Si les **deux boutons sont relâchés** (ouverts) **ou bien les deux sont appuyés** en même temps, le moteur s'arrête.
 - Le moteur tourne avec une vitesse variable commandée par un potentiomètre RV1 lié à la broche A2. En faisant tourner le potentiomètre, la vitesse du moteur varie proportionnellement avec la valeur lue sur la broche A2.
 - Pour faire tourner le moteur dans le sens 1, la broche IN1 du pont H doit avoir 5V et IN2 doit avoir 0V et inversement pour le sens2.
 - La vitesse du moteur est commandée par la broche EN1 du pont H qui est de même commandée par le potentiomètre sur la broche A2.

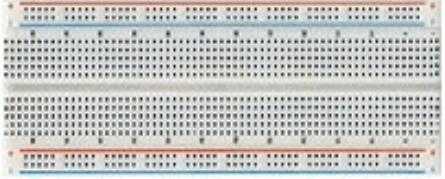
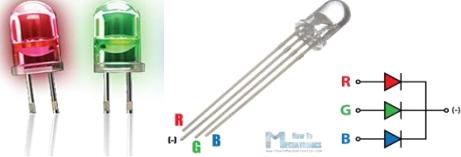
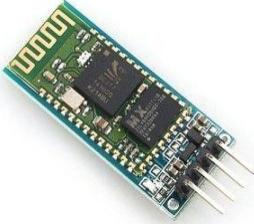
TP3

Transmission série & affichage sur LCD

I. Objectifs de TP

- Manipuler l'envoi et la réception des données sur la voie série de la carte Arduino
- Commander des diapositifs électroniques à travers des consignes envoyées sur la voie série
- Utiliser des applications Android pour commander des dispositifs électroniques via Bluetooth (protocole de communication série)
- Afficher des messages textuels sur un afficheur LCD

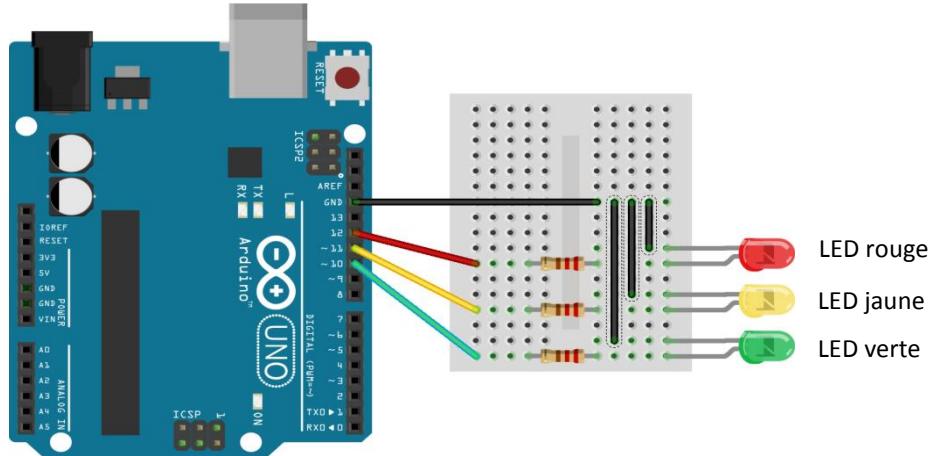
II. Matériels

 Une carte ARDOUINO UNO	 Une plaque d'essai
 3x DIODE LED + 1x LED rgb	 Bluetooth hc-05
 3x Resistance 220Ω	 LCD 16x2
 1x Potentiomètre 1KΩ	 Des fils de connexion

III. Partie Pratique

Exercice 1 : commander des LEDs via la voie série

1. Réaliser le montage de la figure ci-dessous.



2. Ecrire le programme Arduino qui permet de recevoir un caractère envoyé à partir du PC sur la voie série afin d'allumer/éteindre une LED bien définie.

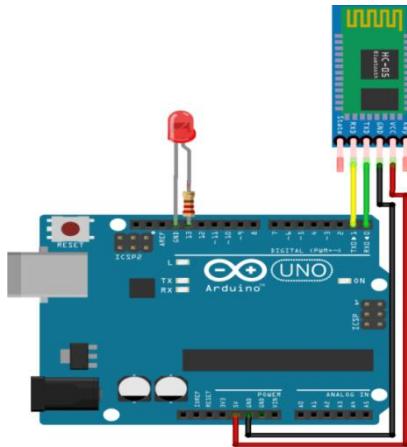
- Si le caractère reçu correspond à la lettre ‘R’, alors la LED rouge change d’état, si elle est éteinte, elle sera allumée et inversement.
- Les caractères ‘J’ et ‘V’ seront destinés pour commander les LEDs jaune et verte.
- Un message sera envoyé vers le PC pour indiquer le changement d’état de la LED correspondante.

Exercice 2 : commander une LED via une application Android

1. Télécharger l’application Android « LED Controller.apk » à partir du site web Appstore et l’installer sur votre Smartphone



2. Réaliser le montage de la figure ci-dessous en reliant la broche Tx du module Bluetooth avec la broche Rx de la carte Arduino et la Rx du module Bluetooth avec la Tx de Arduino.

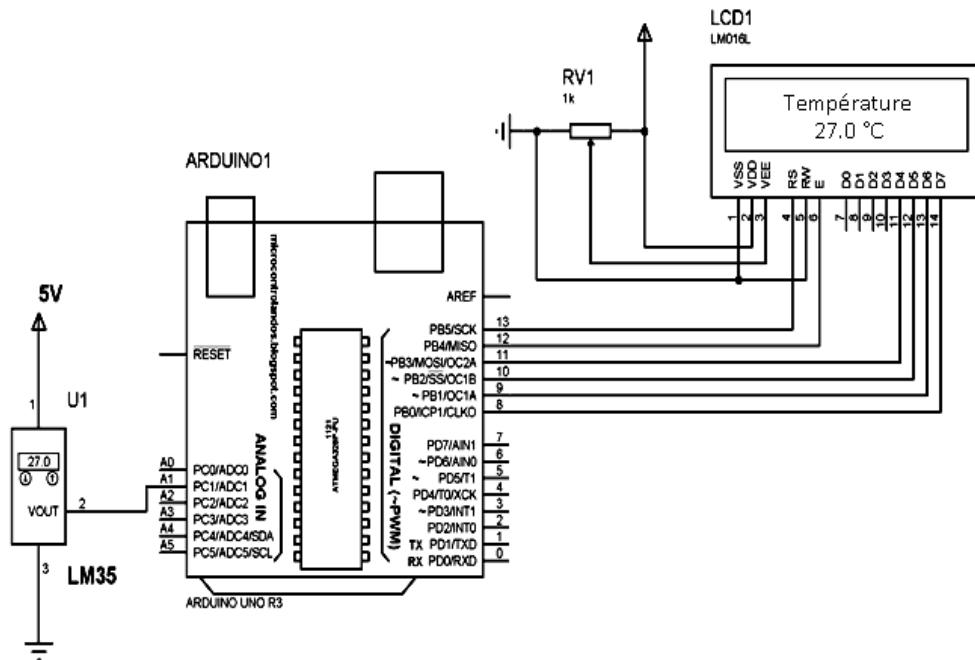


3. Ecrire le programme Arduino qui permet d'allumer la LED branchée sur la pin 13 ou bien de l'éteindre selon le caractère reçu sur la voie série.

NB: un appui sur le bouton ON au niveau de l'application Android permet d'envoyer le caractère '1' à travers le Bluetooth ainsi que l'appui sur OFF permet d'envoyer le caractère '0'.

Exercice 3 : Affichage sur un LCD

1. Réaliser le montage de la figure ci-dessous comportant un capteur de température LM35 et un écran LCD 2x16.



2. Ecrire un programme ARDUINO qui permet de:

- Mesurer la température chaque 2 secondes.
On vous donne : Température= tension sur la broche A1 *100.0
- Afficher la température mesurée sur l'écran LCD.

NB : vous pouvez utiliser l'exemple HelloWorld de la bibliothèque LCD fourni avec les exemples du logiciel Arduino en allant vers Fichier→exemples→LiquidCrystal→HelloWord.

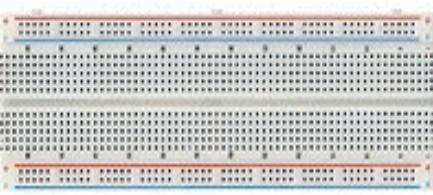
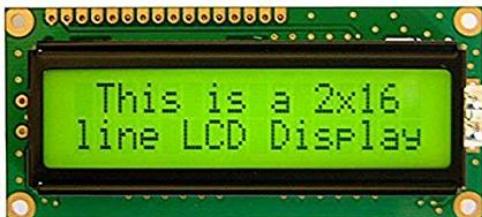
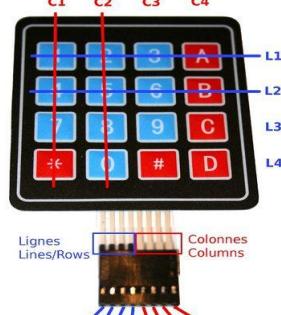
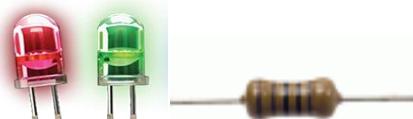
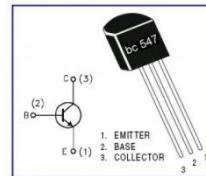
TP4

Gestion d'un clavier 16 touches

I. Objectifs de TP

- Concevoir un système d'accès par un mot de passe saisi au clavier 16 touches
- Afficher des messages textuels sur un afficheur LCD

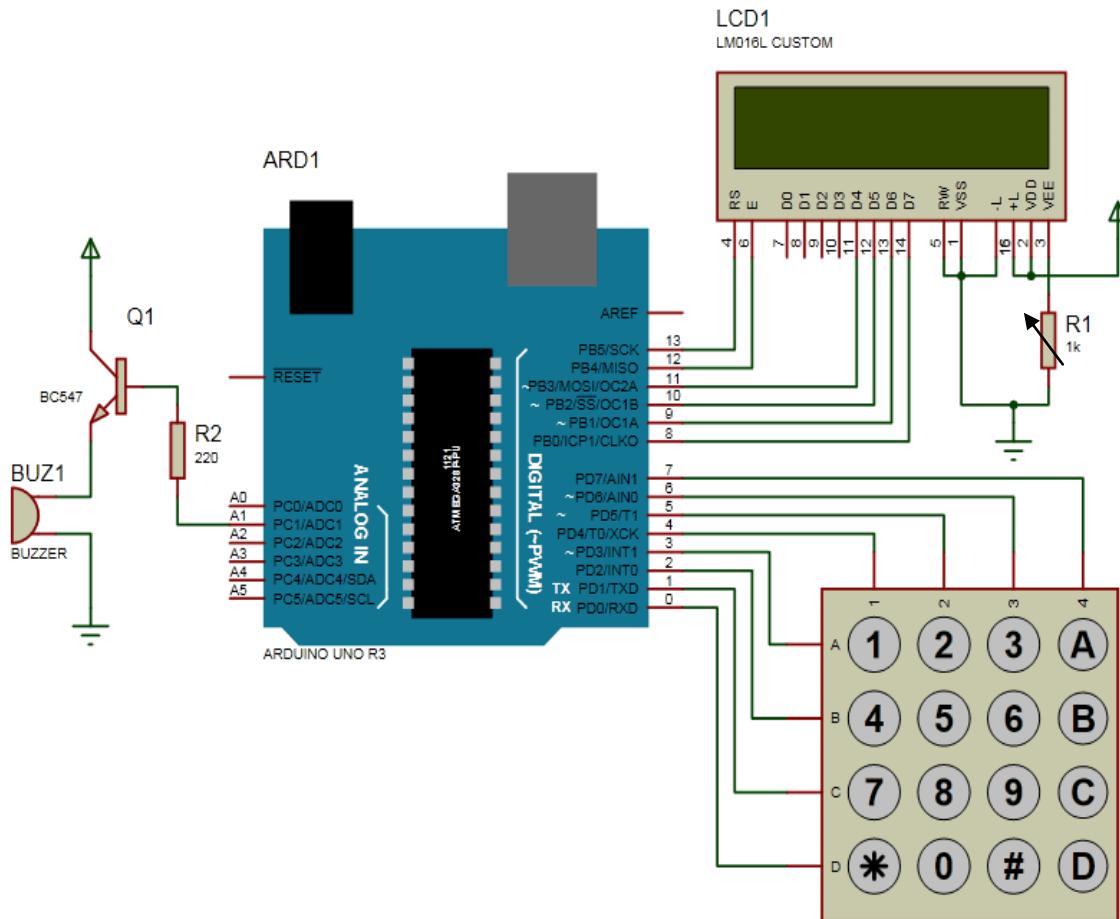
II. Matériels

 <p>Une carte ARDOUINO UNO</p>	 <p>Une plaque d'essai</p>
 <p>LCD 16x2</p>	 <p>Clavier 16 touches</p>
 <p>2x DIODE LED+ 3x Resistance 220Ω</p>	 <p>Transistor BNP BC547</p>
 <p>1x Potentiomètre 1KΩ</p>	 <p>Buzzer piezo</p>

III. Partie Pratique

Exercice 1 : Système d'accès par mot de passe

1. Réaliser le montage de la figure ci-dessous.
2. Connecter une LED rouge et une LED verte respectivement sur la broche A3 et A4.



1. Ecrire un programme ARDUINO qui permet de:

- Saisir un mot de passe à l'aide du clavier 16 touches.
- Activer le buzzer et allumer la LED rouge si le mot de passe saisi est erroné.
- Désactiver le buzzer et allumer la LED verte si le mot de passe saisi est correcte.
- Afficher dans chaque cas un message textuel sur LCD tel que : Entrer votre password, password erroné, password correcte.
- on suppose que le mot de passe enregistré est le suivant:
char password[4]={'2','0','1','8'};
- **L'appui sur # permet de valider le mot de passe saisi.**

NB : vous pouvez utiliser l'exemple HelloWorld de la bibliothèque LCD et l'exemple HelloKeypad de la bibliothèque Keypad en allant vers Fichier → exemples → LiquidCrystal → HelloWord
Fichier → exemples → Keypad → HelloKeypad

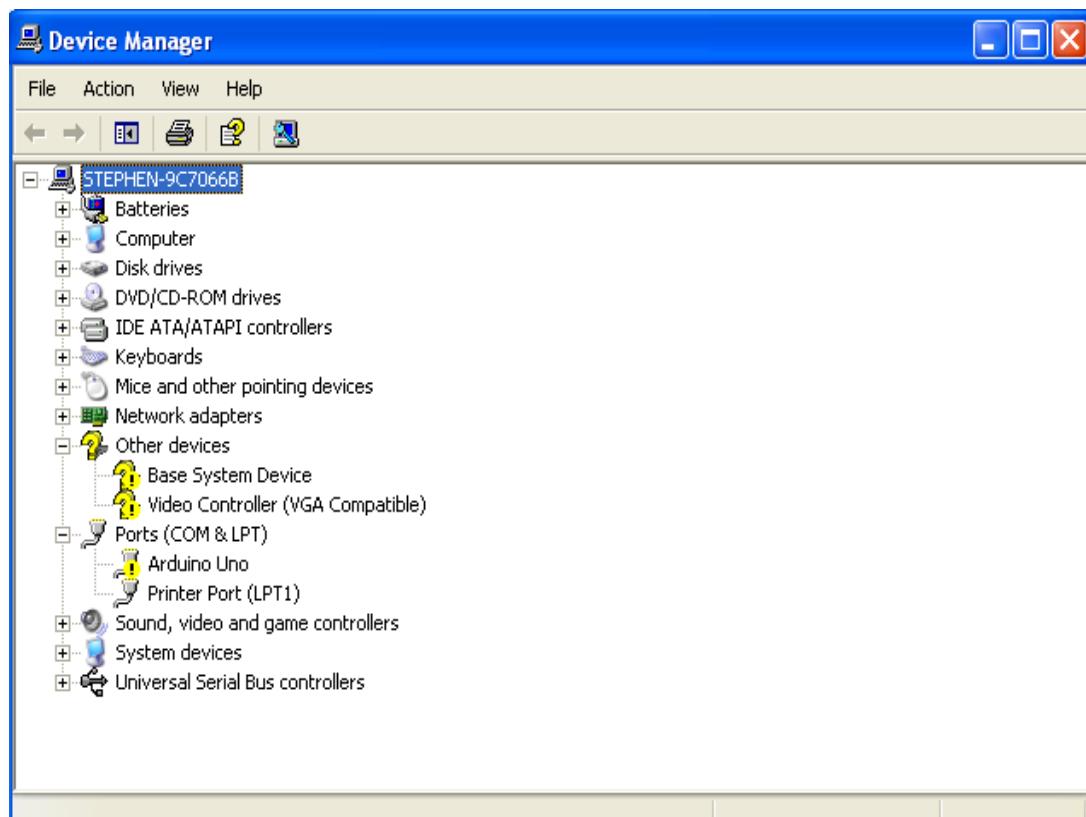
Annexe

Les étapes d'installation des pilotes de la carte Arduino

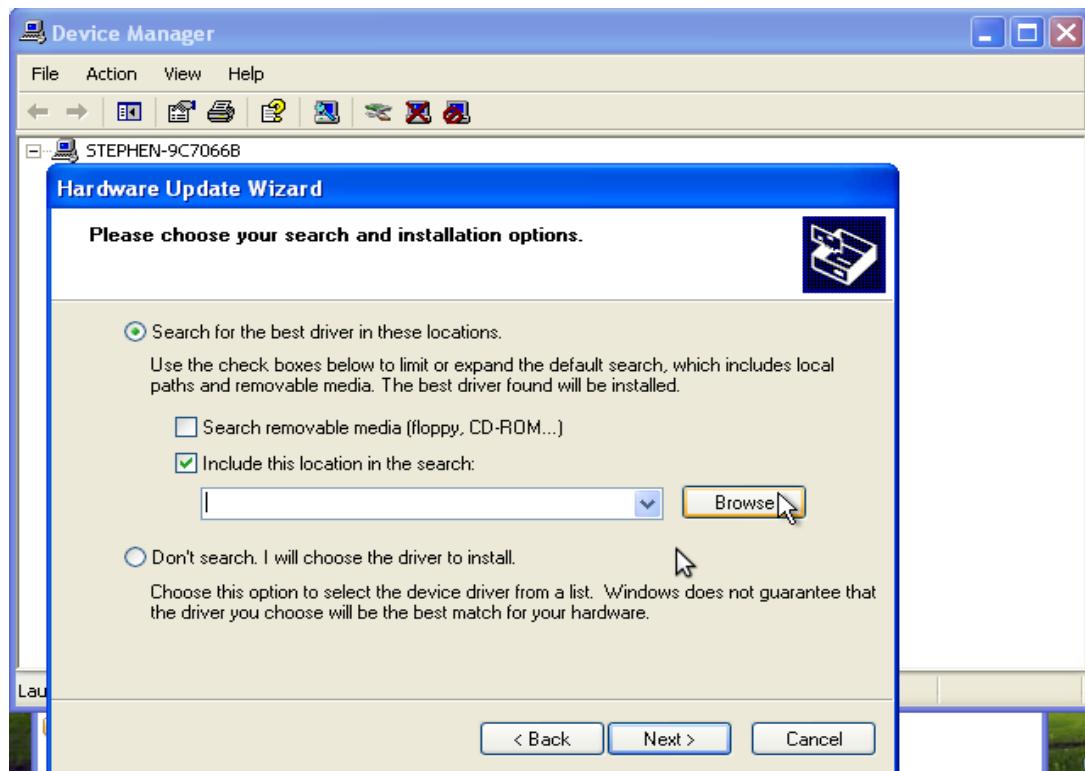
- 1) Brancher la carte ARDUINO avec le PC par le câble USB



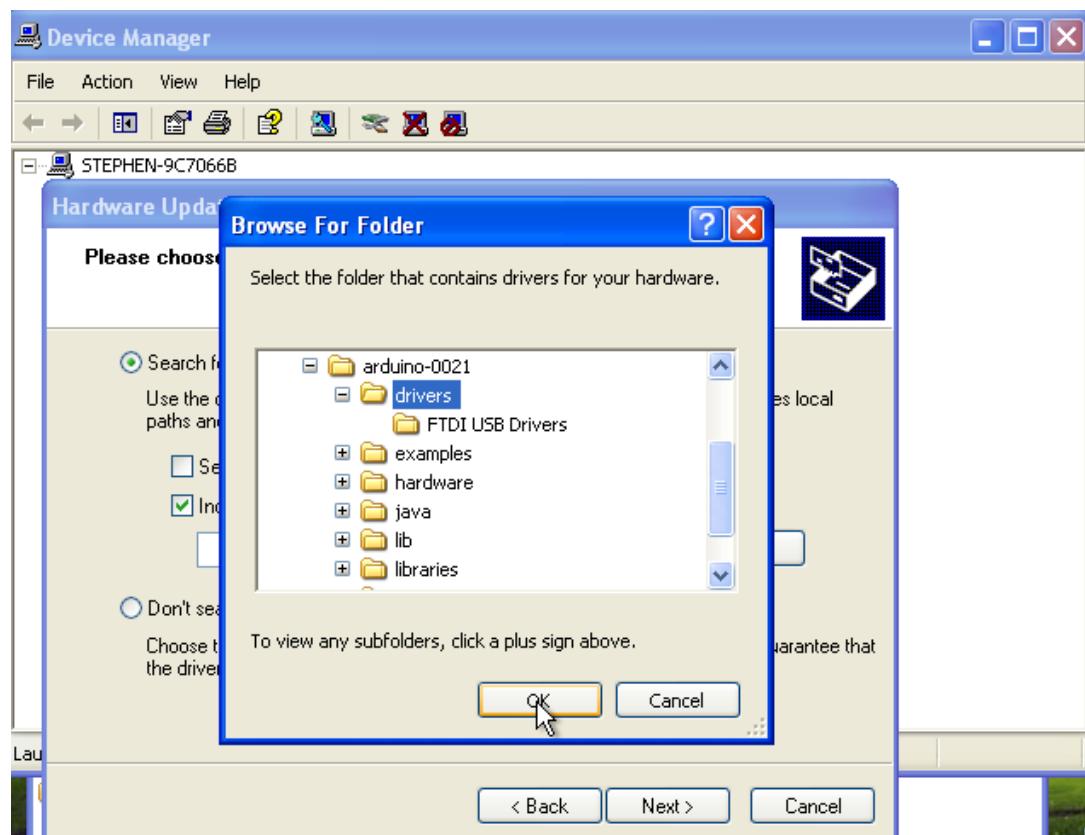
- 2) Aller vers le gestionnaire de périphériques pour installer le pilote de la carte Arduino
- 3) cliquer sur Arduino Uno par le bouton droit de la souris et choisir mettre à jour le pilote



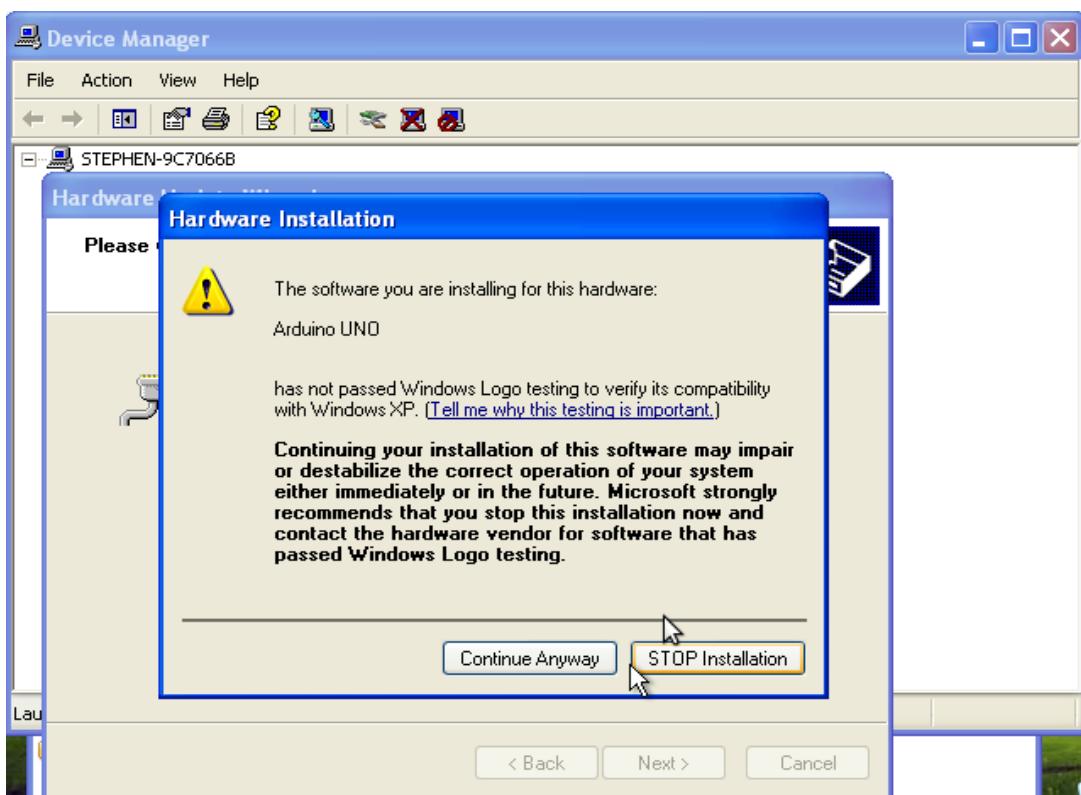
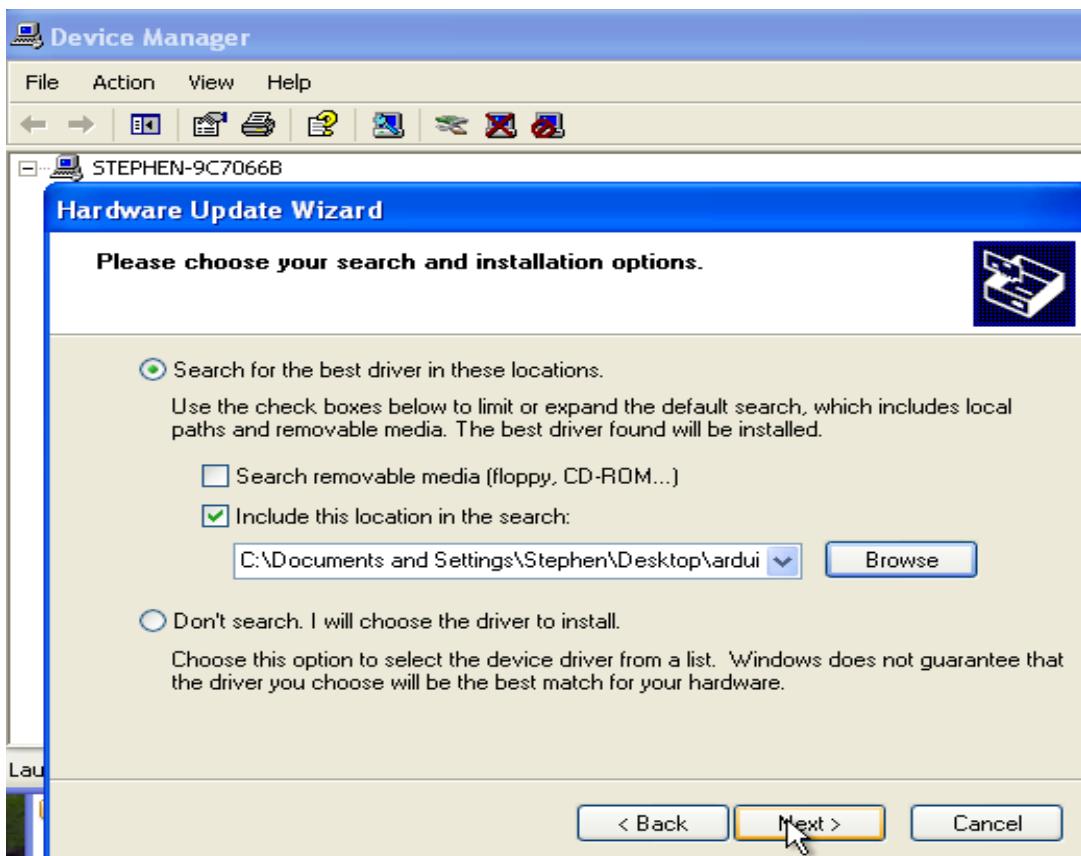
- 4) Aller vers le répertoire d'installation du logiciel Arduino et sélectionner le dossier driver

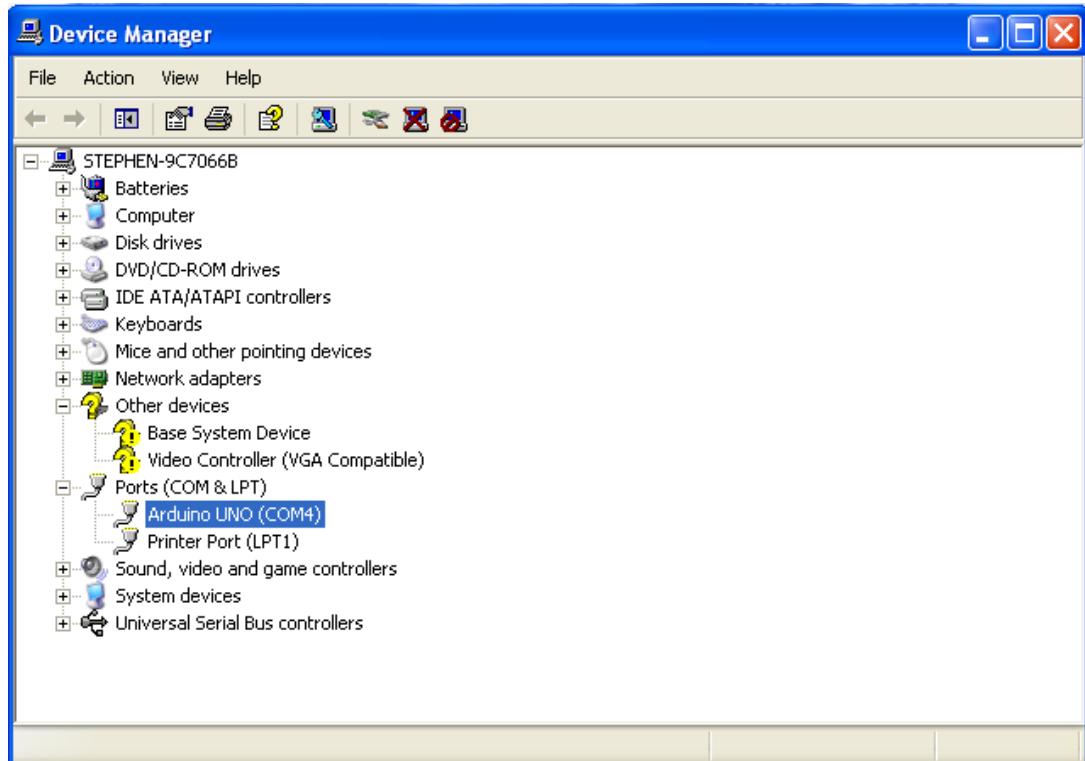
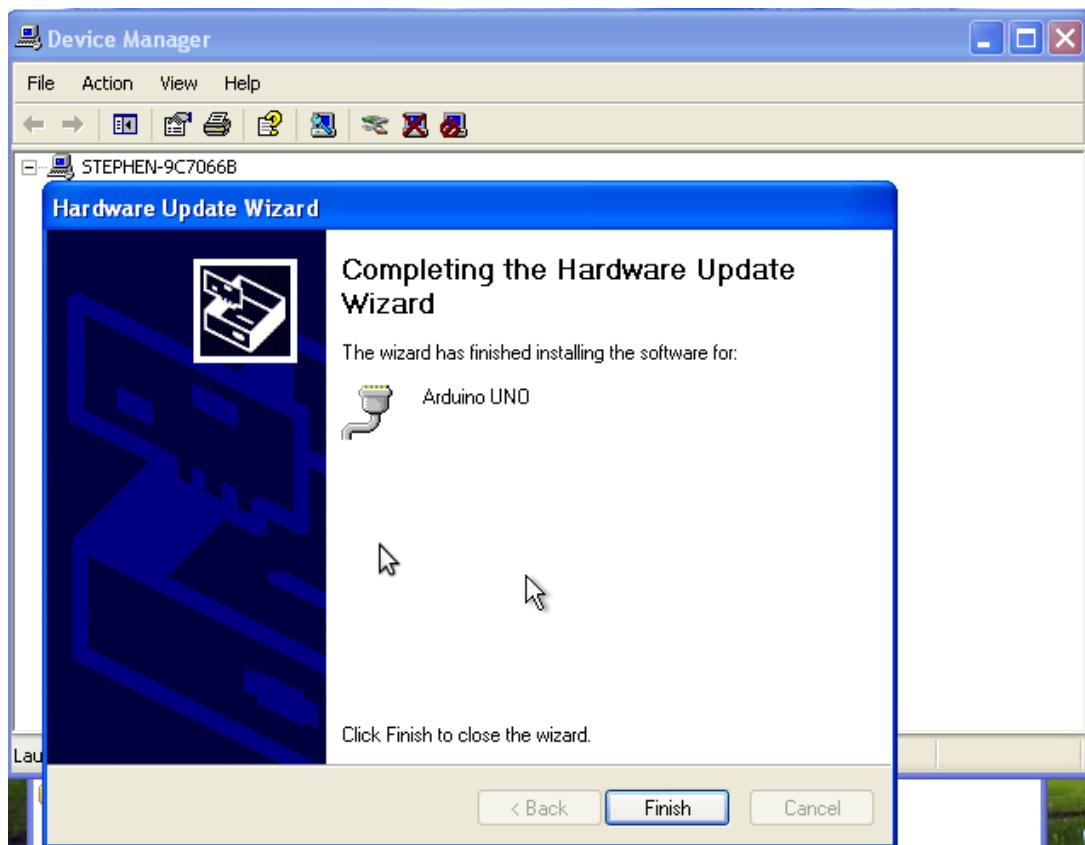


- 5) Sélectionner le dossier FTDI USB drivers puis appuyer sur ok

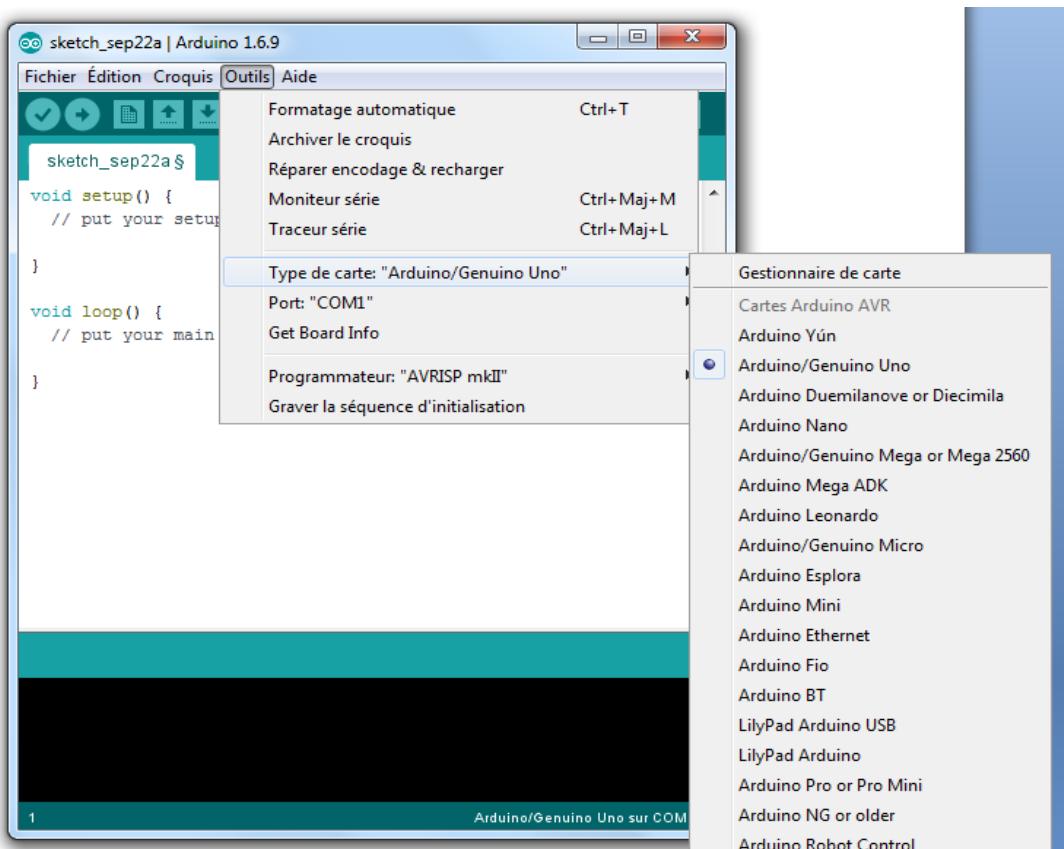


6) Cliquer sur Next





- 7) Lancer le logiciel Arduino et aller vers outils puis choisir le type de la carte cible



- 8) Sélectionner le port série sur lequel votre Arduino est branché.

Vous pouvez le faire par l'intermédiaire du menu **Outils > Port**. Cela doit être quelque chose qui ressemble à COM3 ou supérieur (COM1 et COM2 sont habituellement réservé aux ports séries matériels).

