LES MICROCONTROLEURS

• Un microcontrôleur est un circuit intégré rassemblant dans un même boîtier un microprocesseur, plusieurs types de mémoires et des périphériques de communication (Entrées-Sorties).

A. Le matériel

1. Le microcontrôleur

• Il existe de nombreuses marques et de nombreux modèles de microcontrôleurs. Arduino™ est la plus connue d'entre elles, au point de devenir une antonomase.

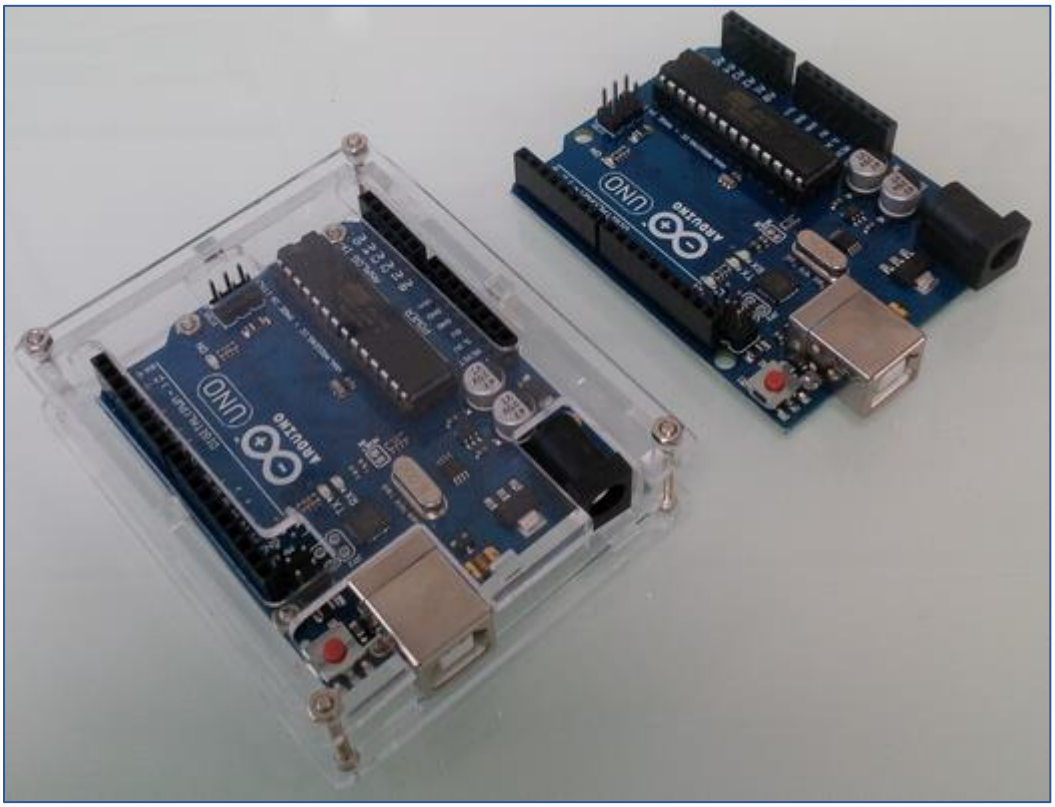
• L'intégralité des expériences de ce manuel a été réalisée avec une carte Arduino™ Uno R3.

• Pour une utilisation relativement simple de la carte, les montages et programmes sont similaires d'un modèle à l'autre.

• Certains modèles de marque Adafruit™ présentent l'avantage d'être programmables en Python.

• Quel que soit le modèle choisi, il sera indispensable de l'équiper d'un boitier de protection.

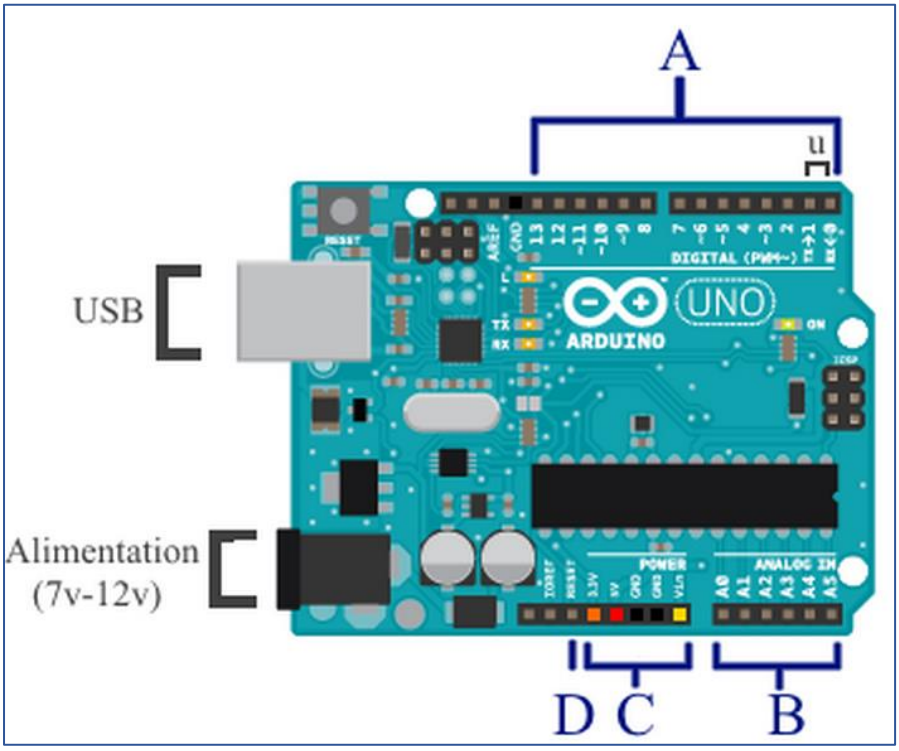
Doc. 1 Microcontrôleur « nu » et dans un boitier de protection



2. Les principales séries de broches

- A : Broches dites numériques (HIGH ou LOW en anglais - tout ou rien en français – 0 ou 5 V).
- B : Broches dites analogiques, valeur entre 0 V et 5 V.
- C : Broches d'alimentation :
  - Rouge : sortie 5 V (+),
  - Orange : sortie 3,3 V (+),
  - Noire : les masses (−),
  - Jaune : entrée reliée à l'alimentation (7 V-12 V).

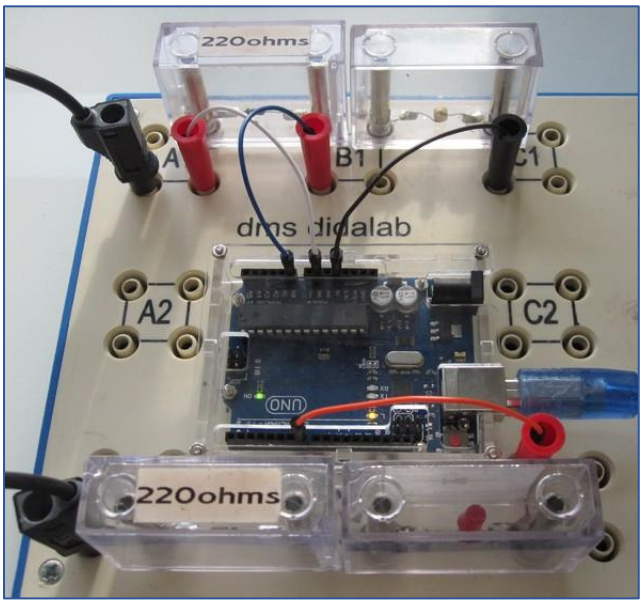
Doc. 2 Schéma des broches du microcontrôleur



3. Les accessoires nécessaires

- Pour les programmes du tronc commun de Seconde et de la spécialité de Première, il n'est pas indispensable d'investir dans les kits proposés sur les sites marchands ou chez les fournisseurs habituels.
- Les laboratoires de physique-chimie sont déjà équipés de la plupart du matériel nécessaire : résistances, DEL, photorésistances, thermistances, capteurs de pression, …
- Il faudra néanmoins envisager l'achat de quelques fournitures complémentaires : buzzers actifs, capteurs résistifs de force, plaques d'essais LAB 170 points, câbles LAB mâles (d'au moins 20 cm) dont certains seront équipés de fiches bananes 4 mm à l'une des extrémités pour s'adapter aux dipôles classiques.
- Les microcontrôleurs sont habituellement fournis avec un câble USB A Mâle / B Mâle permettant de les relier aux ordinateurs et ainsi de les alimenter. Relativement courts, il est possible de leur substituer les câbles prévus pour les interfaces d'acquisition, les spectrophotomètres, …

Doc. 3 Exemple de montage

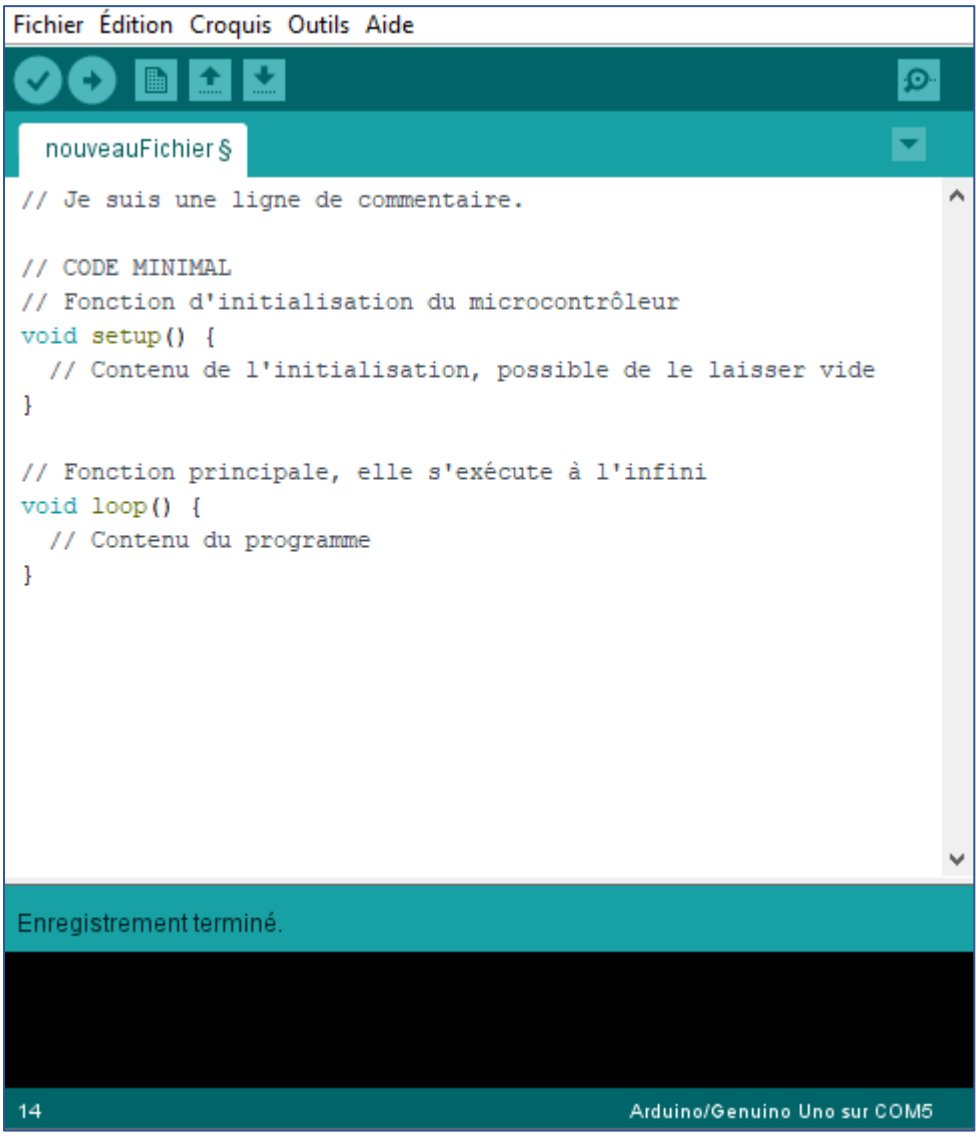


B. L'environnement de développement et le langage Arduino

1. Le logiciel pour écrire les programmes appelés croquis

- Le logiciel de développement Arduino tout comme le langage sont libres de droit. Il est téléchargeable à l'adresse suivante : https://www.arduino.cc/en/main/software.
- L'installation de cet environnement de développement suffit pour la plupart des microcontrôleurs, aucun pilote supplémentaire n'est nécessaire.

Doc. 4 Interface de l'environnement de développement



- Dans un premier temps, les 6 icônes du bandeau supérieur suffisent à la mise en œuvre des microcontrôleurs.

Doc. 5 Rôle des 6 icônes du bandeau supérieur

	Vérifier l'absence d'erreurs de syntaxe dans le code saisi
	Téléverser (envoyer) le programme sur le microcontrôleur
	Créer un nouveau fichier similaire au doc 3 (mais en anglais)

	Ouvrir un programme
	Enregistrer un programme
	Ouvrir une nouvelle fenêtre dans laquelle s'affiche les résultats des mesures

• Les fichiers Arduino ont une extension .ino reconnue par l'application. Pour les ouvrir, il est nécessaire de les placer dans un répertoire portant le même nom que le fichier.

Doc. 6 Arborescence des fichiers

2. Le langage Arduino

• Le langage Arduino est très proche des langages C et C++. Seules quelques instructions sont nécessaires pour les activités de ce manuel et les 2 fonctions indispensables, commentées en anglais, sont systématiquement écrites lors de la création d'un nouveau fichier.

Doc. 7 Création d'un nouveau fichier

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

• Pour traiter les données collectées par le microcontrôleur et envoyées à l'ordinateur, il est préférable d'utiliser Python afin de ne pas multiplier les langages (Voir la fiche technique Python).

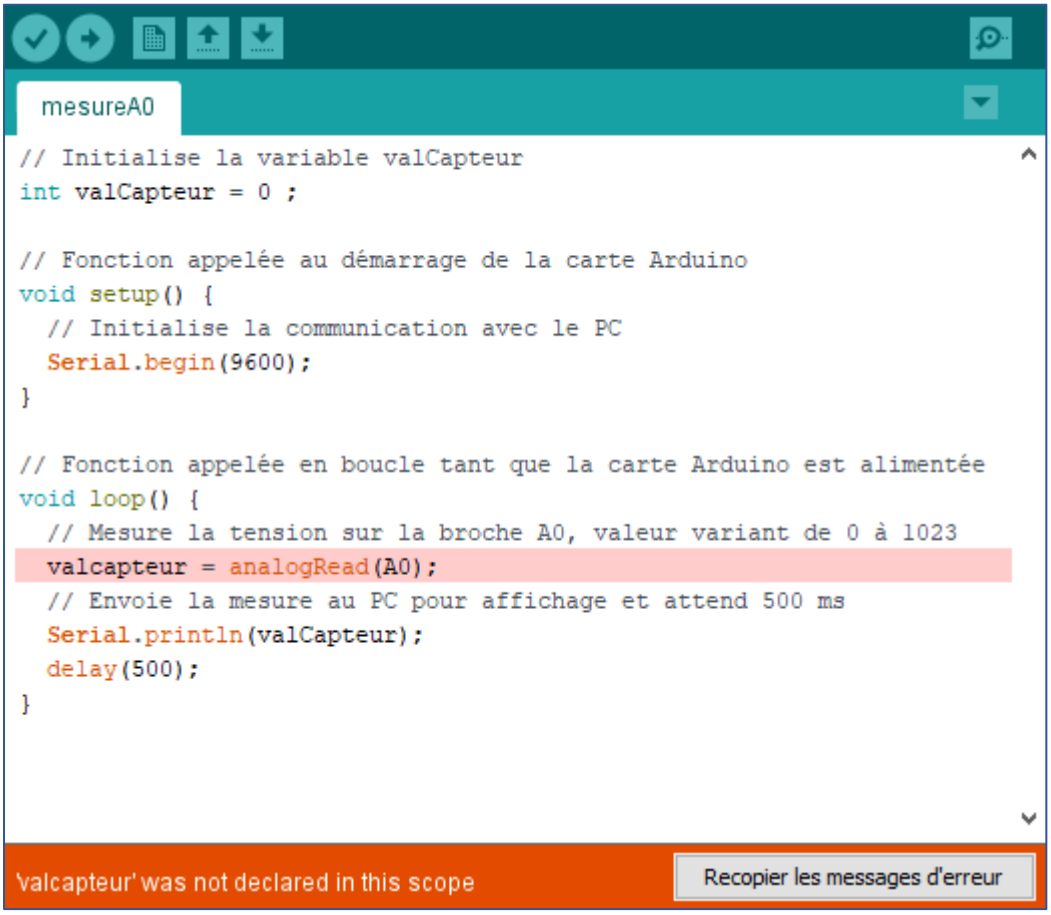
• Certaines marques de microcontrôleurs n'utilisent d'ailleurs que Python.

3. Les problèmes classiques lors du téléversement

• Le téléversement est une étape indispensable mais parfois capricieuse, notamment sur les ordinateurs des réseaux pédagogiques où élèves et enseignants n'ont pas des droits d'administrateur.

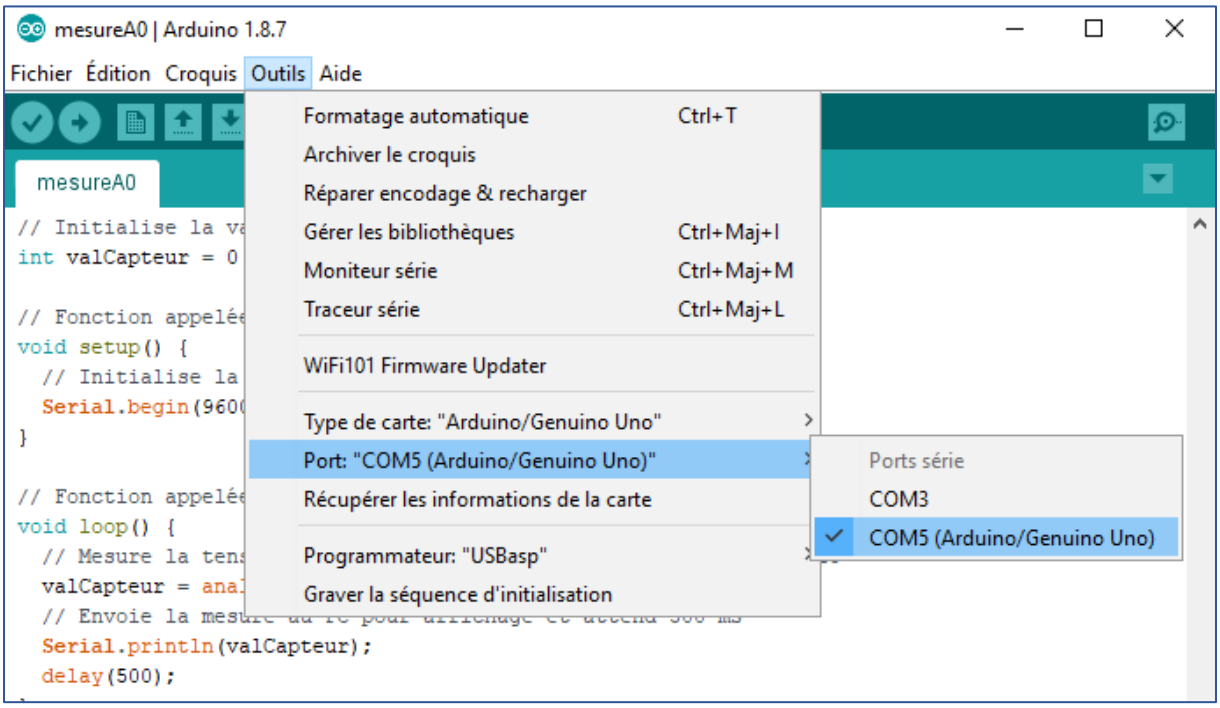
• Dans un premier temps il est indispensable de vérifier le programme avant de le téléverser (Icône Vérifier).

Doc. 8 Exemple d'erreur détectée : mauvais nom de variable valCapteur => valcapteur



• Les problèmes de téléversement proviennent souvent d'une erreur de port, avant de téléverser le programme il est nécessaire de vérifier que le port choisi correspond bien à votre microcontrôleur.

Doc. 9 Vérification du port



• Pour téléverser un programme modifié ou un nouveau programme, il faut s'assurer que la fenêtre du Moniteur série est fermée.

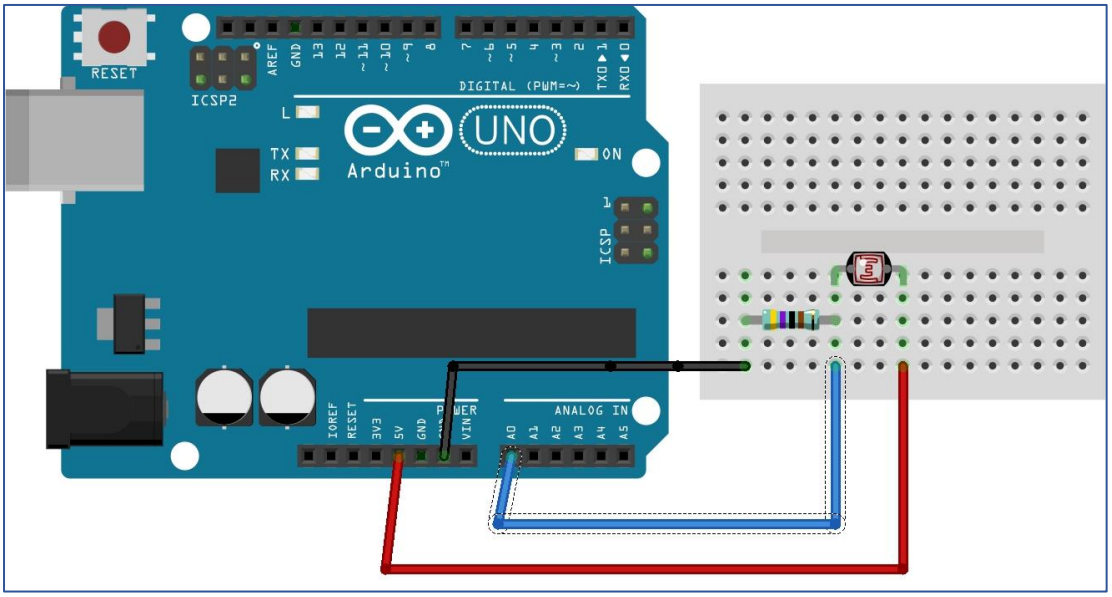
• Si toutes les étapes précédentes ont été respectées et que le téléversement échoue malgré tout, vous pouvez essayer une seconde fois, fermer puis ouvrir le logiciel. En général, cela permet de résoudre la plupart des problèmes rencontrés.

C. Quelques exemples d'utilisation

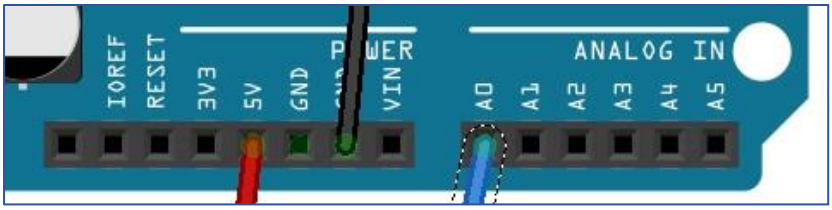
1. Mesure de la tension aux bornes d'un capteur résistif

• Pour les capteurs résistifs le montage utilisé est un diviseur de tension. En fonction des grandeurs étudiées et de l'évolution de la résistance on peut choisir de mesurer la tension aux bornes de la résistance fixe (voir ci-dessous) ou aux bornes du capteur.

Doc. 10 Exemple de montage diviseur de tension



Doc. 11 Broches utilisées



Doc. 12 Programme Arduino correspondant

```
// Initialise la variable valCapteur
int valCapteur = 0 ;

// Fonction appelée au démarrage de la carte Arduino
void setup() {
  // Initialise la communication avec le PC
  Serial.begin(9600);
}

// Fonction appelée en boucle tant que la carte Arduino est alimentée
void loop() {
  // Mesure la tension sur la broche A0, valeur variant de 0 à 1023
  valCapteur = analogRead(A0);
  // Envoie la mesure au PC pour affichage et attend 500 ms
  Serial.println(valCapteur);
  delay(500);
}
```

• Pour effectuer une mesure il est possible d'utiliser les broches A0 à A5, la masse est notée GND. L'ensemble peut être alimenté par le microcontrôleur lui-même sous 5 V.

• Il est nécessaire de définir le débit de transmission des données entre le microcontrôleur et l'ordinateur dans la fonction setup( ).

• Elle s'exprime en baud c'est-à-dire le nombre de caractères transmis par seconde. Les valeurs sont normalisées et varient entre 300 et 115200 bauds.

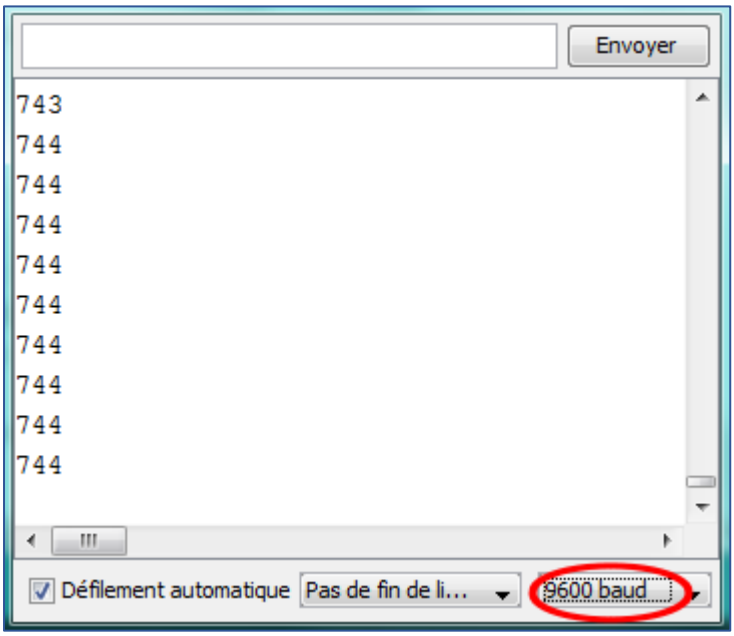
• Pour effectuer une mesure sur une seule broche, un débit de 9600 bauds est suffisant. En revanche, il peut être nécessaire de l'augmenter pour suivre l'évolution temporelle d'une grandeur.

• Il est nécessaire d'indiquer une valeur identique de débit dans la fenêtre du moniteur série (doc 10).

• Par défaut, les valeurs qui s'affichent dans la fenêtre du moniteur série sont des nombres compris entre 0 et 1023, correspondant à 5,00 V. Dans l'exemple ci-dessous la tension est :

$$U(A0) = 5{,}00 \times \frac{744}{1023} = 3{,}63\ V$$

Doc. 13 Fenêtre Moniteur série



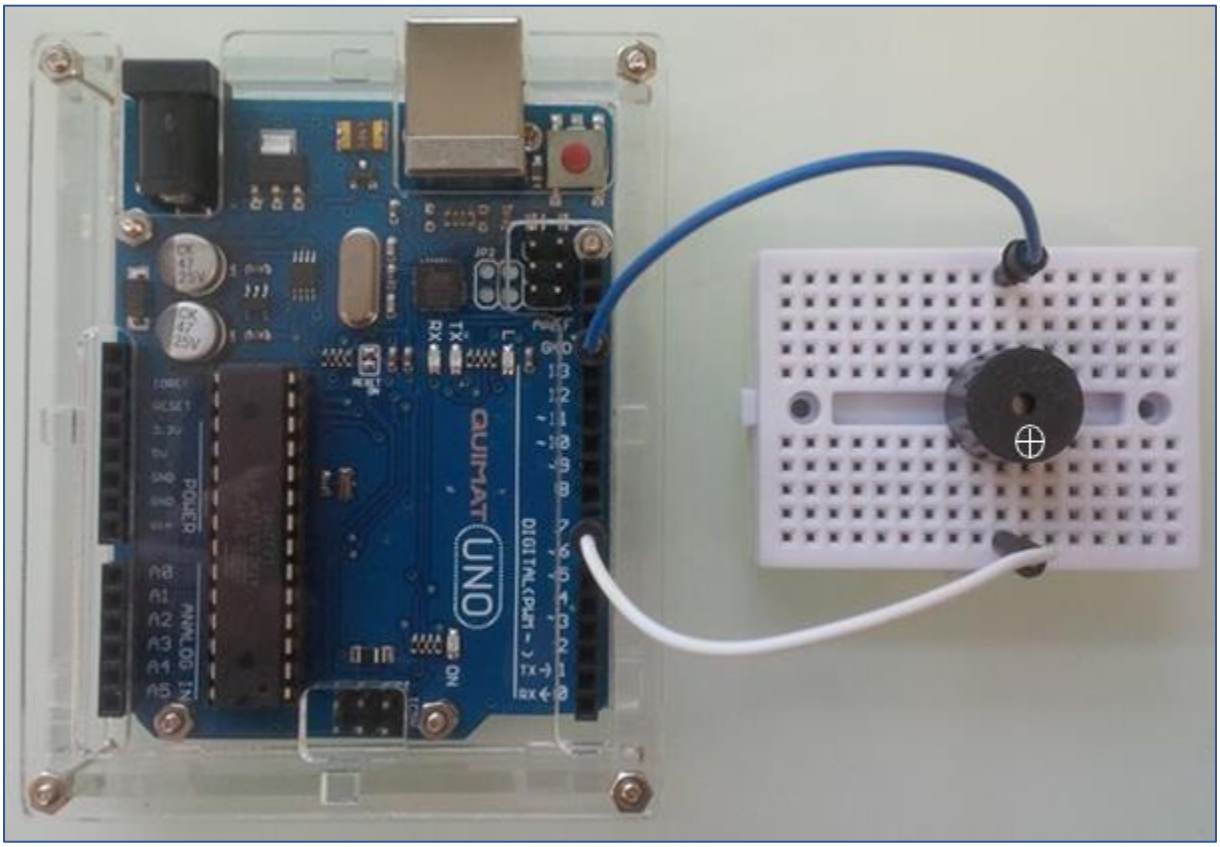
• Certains capteurs, comme les capteurs de pression, délivrent des tensions de l'ordre de quelques dizaines de mV : dans ce cas, il est possible de changer la tension de référence. L'instruction nécessaire et la valeur ainsi obtenue sont variables suivant les modèles.

Doc. 14 Instruction à mettre dans la fonction setup( ) pour un microcontrôleur Arduino™ Uno

```
// Permet de travailler entre 0 et 1,1 V (au lieu de 5 V)
// Dans ce cas tension = valeur x 1,1 / 1023
analogReference(INTERNAL);
```

2. Utilisation d'une sortie du microcontrôleur : émission d'un signal sonore

Doc. 15 Montage d'un buzzer actif



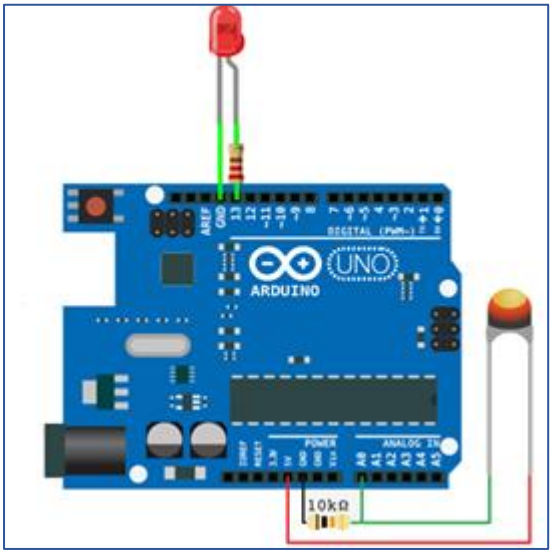
Doc. 16 Programme pour émettre un signal sonore

```
// Indiquer que le + du buzzer est branché sur la borne 7 (autre borne reliée à GND)
char DIO_Buzzer = 7 ;

// Fonction appelée au démarrage de la carte Arduino
void setup() {
}

// Fonction appelée en boucle tant que la carte Arduino est alimentée
void loop() {
  // Emettre un signal de 500 Hz pendant 100 ms.
  tone(DIO_Buzzer, 500, 100) ;
  // Attendre 400 ms avant de recommencer la fonction loop().
  delay(400) ;
}
```

3. Montage d'allumage du témoin « Hot » d'une plaque à induction

- Dans un premier temps, il faut réaliser un montage similaire au doc 7 en remplaçant la photorésistance par une thermistance.
- Il est possible d'étalonner le capteur ou simplement de repérer la valeur renvoyée par le microcontrôleur lorsque la température dépasse 37°C (valCapteur devient supérieur à 800 dans l'exemple ci-dessous).
- Il faut ensuite compléter le montage comme indiqué dans le doc 14.

Doc. 17 Montage d'allumage du témoin « Hot »



Doc. 18 Exemple de programme pour une thermistance de type CTN

```
// Initialise la variable valCapteur de type entier (integer)
int valCapteur = 0;

// Fonction appelée au démarrage de la carte Arduino
void setup() {
  // Indique que la borne 13 est utilisée comme sortie
  pinMode(13,OUTPUT);
}

// Fonction appelée en boucle tant que la carte Arduino est alimentée
void loop() {
  // Mesure la tension sur la broche A0, valeur variant de 0 à 1023
  valCapteur = analogRead(A0) ;
  if (valCapteur > 800) {
    digitalWrite(13,HIGH); // Délivre une tension entre les bornes GND et 13
  } else {
    digitalWrite(13,LOW); // Ne délivre pas de tension entre les bornes GND et 13
  }
  delay(500);
}
```