Physique

TP 13 – Loi de Boyle-Mariotte et Arduino

13 et 27 mai 2024

Objectifs

— Utiliser une carte à microcontrôleur.
— Tester expérimentalement la loi de Boyle-Mariotte pour l'air.

Compétences à acquérir

- Mettre en œuvre un capteur de pression en identifiant son caractère différentiel ou absolu.
- Mettre en œuvre un protocole d'étude des relations entre paramètres d'état d'un fluide à l'équilibre.

1 Présentation

Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur, principalement le processeur, des mémoires et des interfaces d'entrées sorties. Ces dernières permettent aussi bien de récupérer des informations de capteurs (température, pression, etc.) que de commander des systèmes (LED, moteur, etc.). Ils sont fréquemment utilisés dans les systèmes embarqués (automobiles, télécommandes, appareils de bureau, électroménager, jouets, téléphonie mobile, etc.) mais aussi au laboratoire comme la carte SYSAM SP5 utilisée en TP. La carte pourra en outre être avantageusement mise en œuvre pour les TIPE.

Parmi les microcontrôleurs très utilisés figure la carte Arduino, issue d'un projet d'étudiants de l'école de design d'Ivrée en Italie. Elle emprunte son nom au *Bar di Re Arduino*, « bar du roi Arduin », lieu de réunion des concepteurs de la carte. Nous utiliserons ici le modèle de base, la carte « Arduino Uno ». L'architecture est détaillée en figure 1.

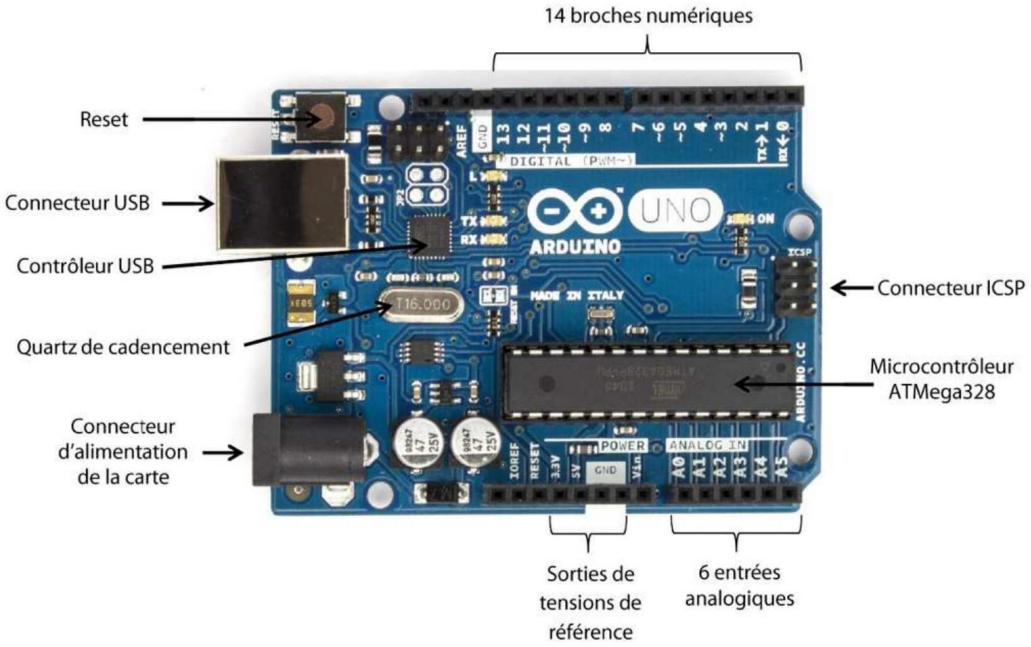


Figure 1 – *Architecture de la carte « Arduino Uno ».*

Cette carte fonctionne, entre-autres, grâce à :
— un microcontrôleur ATMega328 ;
— un quartz 16 MHz (horloge de cadencement) ;
— une connexion USB permettant la programmation du microcontrôleur ainsi que l'alimentation de la carte ;
— un connecteur d'alimentation jack pour alimenter la carte si le cordon USB est déconnecté après programmation, pour des applications nomades. Une tension comprise entre 7 et 12 V est requise.

Elle met à notre disposition :
— 14 broches numériques d'entrées ou de sorties, numérotées de 0 à 13. Le niveau « 0 » correspond à une tension de 0 V, le niveau « 1 » correspond à une tension de 5 V ;
— 6 entrées analogiques, notées de A0 à A5, acceptant des valeurs comprises entre 0 et 5 V ;
— différentes sorties de tension : masse, 5 V.

⚠ *Il est hors de question de travailler avec des tensions supérieures à* 5 V*, de faire circuler des courants de plus de quelques* mA*, ou de faire tourner un moteur ! Pour des systèmes nécessitant plus de puissance, il faut utiliser une alimentation autonome pour le circuit de puissance, l'Arduino ne gérant que la partie commande.*

La carte communique avec l'ordinateur grâce au logiciel « Arduino IDE » (environnement de développement intégré). Elle se programme en langage C++. En pratique, il est possible de partir d'un *sketch*, c'est-à-dire un programme déjà existant, et de l'adapter plutôt que de partir d'une page blanche. Par ailleurs, il existe un certain nombre de bibliothèques « clés en main » pour divers composants auxquelles faire appel.

À l'ouverture, la fenêtre de l'IDE se présente sensiblement comme montré en figure 2. Un fragment de code est déjà présent. Comme expliqué par les commentaires (balisés par `//` au lieu de `#` avec Python) :
— la partie `setup` accueille les instructions destinées à n'être exécutées qu'une seule fois (chargement de bibliothèques, initialisation de variables, etc.) ;
— la partie `loop` celles à répéter (lecture au cours du temps de la valeur d'une entrée, etc.).

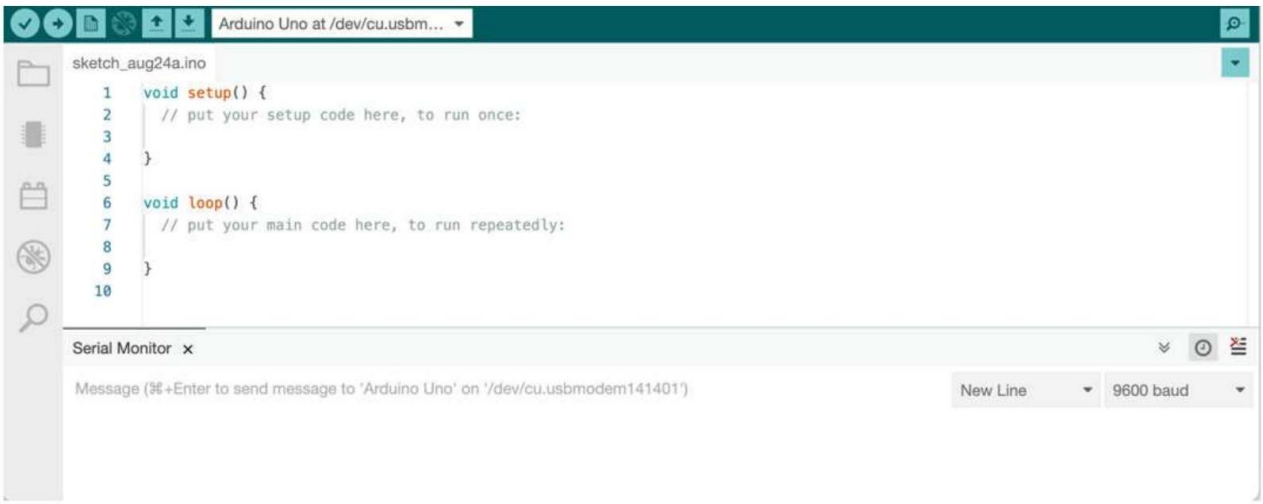


Figure 2 – *Allure du logiciel « Arduino IDE » à l'ouverture.*

▶ La carte étant connectée à l'ordinateur *via* le câble USB, les premiers réglages sont :
— Menu « Outils » puis « Type de carte » : vérifier que « Arduino Uno » est bien coché (cocher sinon) ;
— Vérifier que la carte est bien reconnue en allant dans « Port » du même menu. C'est bien le cas si un port est affiché.

Une fois le script saisi, la procédure d'utilisation de la carte est la suivante :
— Vérification du code (et correction si nécessaire !) ;

— Téléversement du programme dans la carte ;
— La communication bidirectionnelle entre l'ordinateur et la carte se fait alors grâce au « moniteur série » (icône en haut à droite et fenêtre en bas).

Les données alors recueillies dans ce moniteur série, si elles sont bien formatées, pourront faire l'objet d'un traitement ultérieur une fois exportées dans un fichier texte.

2 Utilisation

2.1 Premiers essais

▶ Recopier le code ci-dessous permettant d'afficher « Hello world ! » dans le moniteur série.

```
1 void setup() {
2     Serial.begin(9600) ;
3     Serial.println("Hello world!") ;
4 }
5
6 void loop() {
7     // put your main code here, to run repeatedly:
8
9 }
```

▶ Modifier ce code pour que « PCSI 1 » s'affiche toutes les secondes dans le moniteur série.
Pour attendre `x` millisecondes avant de poursuivre l'exécution du code, il faut utiliser `delay(x)`.

2.2 Commande d'une LED

La LED RGB utilisée est une LED à cathode commune, la borne la plus longue, qui sera reliée à la masse GND, figure 3.

⚠ *Attention, l'absence de résistances de protection entraînerait la destruction immédiate de la diode lors de son branchement.*

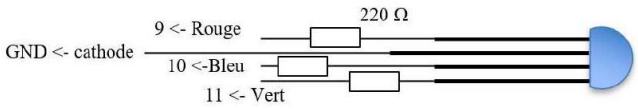


Figure 3 – *Connexions de la LED RGB.*

▶ Appliquer un potentiel positif à l'aide des trois broches numériques 9, 10 et 11 à travers un résistor de protection de 220 Ω sur l'une des autres bornes de la LED permet d'allumer la couleur correspondante à l'aide de la fonction `digitalWrite(port, état haut ou bas)`. Ces composants peuvent être implantés sur une carte additionnelle (*shield* = carte d'interface) montée sur la carte Arduino, figure 4a.

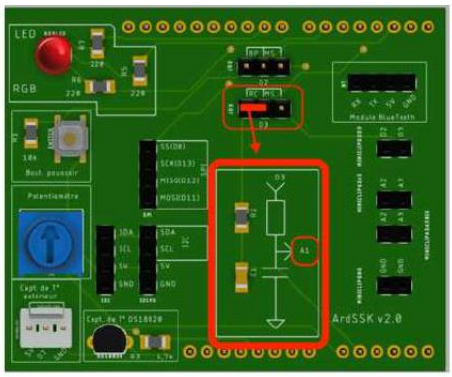
À défaut, ces composants seront implantés sur une plaque de prototypage, figure 4b, et reliés à la carte par des fils de connexion de couleur. Sur cette carte se trouvent en particulier :
— deux lignes « - » horizontales, en bleu, non connectées entre elles et dont l'une sera reliée à la masse de l'Arduino ;
— deux lignes « + » horizontales, en rouge, non connectées entre elles et dont l'une sera reliée au +5 V de l'Arduino ;
— 60 lignes verticales permettant de réaliser le montage.

Les connecteurs sont reliés entre eux perpendiculairement à l'axe de la plaque et se trouvent donc au même potentiel.

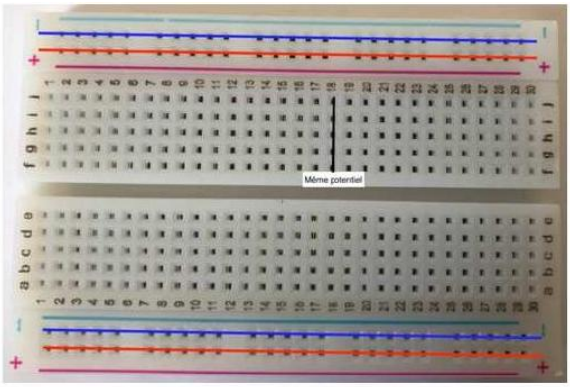
Allumage simple

▶ Ouvrir le fichier « `RGB.ino` » fourni et l'exécuter pour tester le bon fonctionnement de la LED.

Après avoir défini trois constantes reliant chaque couleur au port correspondant, chaque couleur est successivement allumée (`HIGH`) pendant deux mille millisecondes (`delay`) avant de l'éteindre (`LOW`).



(a) Shield *ou carte d'interface.*



(b) *Plaque de prototypage.*

Figure 4

Clignotant

▶ Modifier légèrement le programme pour faire clignoter indéfiniment la LED avec une couleur blanche et une fréquence de 0,2 Hz.

Valeur analogique Jusque là, seules des comportements « tout out rien » (`HIGH` ou `LOW`) sont utilisés. Il est aussi possible d'attribuer une valeur quelconque entre `LOW` et `HIGH` grâce à `analogWrite`.

La fonction `analogWrite(port, i)` permet en effet de fixer la valeur de la tension d'un port, mais à une valeur comprise entre 0 et 5 V, à $\frac{5\times i}{255}$ V. Le programme `crescendo.ino` ci-dessous permet d'augmenter progressivement l'intensité de la couleur verte de la LED à l'aide d'une boucle `for` qui fait croitre par valeurs entières successives la valeur de `i` de 0 à 255, puis de recommencer indéfiniment.

```
1  const int greenLEDPin = 11 ;
2
3  void setup() {
4      pinMode(greenLEDPin, OUTPUT) ;
5  }
6
7  void loop() {
8      for (int i = 0 ; i < 255 ; i++) {
9          analogWrite(greenLEDPin, i) ;
10         delay(20) ;
11     }
12 }
```

▶ Recopier ce code et tester son bon fonctionnement.

Feu tricolore

▶ Programmer un feu tricolore « à trois états » (séquence utilisée en France).

▷ Les couleurs suivantes doivent se répéter indéfiniment :

1. rouge pendant 4 s
2. vert pendant 4 s
3. orange (RGB = 255, 165, 0) pendant 2 s.

2.3 Décharge d'un condensateur dans un résistor

Outre la production des signaux de commande précédents, la carte peut aussi procéder à des acquisitions de signaux provenant de capteurs par l'intermédiaire des **entrées analogiques**.
L'objectif ici est de réaliser la mesure de la constante de temps d'un circuit RC série, par l'acquisition de la courbe de décharge du condensateur dans un résistor, obtenue en mesurant l'évolution de la tension aux bornes du condensateur au cours du temps.

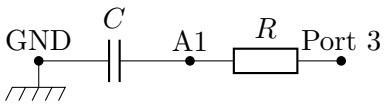


Figure 5 – *Circuit RC à réaliser.*

▶ Réaliser le circuit de la figure 5 avec $C = 10\,\mu F$ et $R = 10\,k\Omega$ sur la platine de prototypage.

▶ Relier le circuit RC à la sortie analogique/numérique de la broche n° 3 et à l'entrée analogique A1 conformément aux indications de la figure 5.

▷ À un facteur multiplicatif 5/1023 près, A1 permet de lire la tension en volt aux bornes du condensateur. Ainsi, si A1 indique « 850 », la tension est donc $850 \times 5/1023 = 4{,}15\,V$.

Le programme `RC_decharge.ino` est le suivant :

```
1  #define analogPin A1
2  #define commandePin 3
3  long start = 0 ;
4  int sensorValue = 0 ;
5
6  void setup() {
7      Serial.begin(9600);
8      pinMode(commandePin, OUTPUT) ;
9      digitalWrite(commandePin, HIGH) ;
10     delay(3000) ;
11     sensorValue =
   ↪   analogRead(analogPin);
12     digitalWrite(commandePin, LOW) ;
13     start = millis() ;
14 }
15
16 void loop() {
17     int sensorValue =
   ↪   analogRead(analogPin);
18     if (sensorValue > 1) {
19         String msg = "DATA" ;
20         msg = msg + ":TIME:" +
   ↪   String(millis()-start) ;
21         msg = msg + ":A1:" +
   ↪   String(sensorValue) ;
22         Serial.println(msg) ;
23     }
24 }
```

Description des lignes

Setup :

9 une tension de 5 V est imposée à l'entrée 3.

10 un délai de 3 s est laissé au condensateur pour se charger.

12 et 13 à l'instant $t = 0$, repéré grâce à `start = millis()`, une tension nulle est imposée sur l'entrée 3.

Loop :

17 la tension sur l'entrée A1 est lue.

18 condition de lecture, quand A1 devient inférieure à 1 (donc $1 \times 5/1023 = 4{,}9\,mV$ soit environ 0,16 % de sa valeur initiale), les mesures s'arrêtent.

19-21 Construction du texte à afficher.

22 Affichage du texte sur le port série.

▶ Exécuter le programme `RC_decharge.ino` mis à disposition.

▶ Copier puis coller les données du moniteur série dans un fichier `.txt` (en utilisant l'application de bloc-notes, par exemple).

▶ Enregistrer le fichier texte sur votre clé USB.

▶ Ouvrir le fichier sur Python et extraire les données afin d'obtenir deux listes : l'une contenant le temps et l'autre les valeurs de tension.

▷ Utiliser par exemple `genfromtxt` de la bibliothèque `numpy`.

▶ Déterminer la constante de temps et comparer à la valeur théorique attendue.

3 Loi de Boyle-Mariotte

Loi de Boyle-Mariotte [1] Découverte en 1662 par Robert Boyle (1627-1691) puis redécouverte par Edme Mariotte (1620-1684).
Le produit PV d'un gaz parfait est une fonction de T uniquement.
Il s'agit d'une relation approchée, valable pour un gaz parfait, mais qui n'est plus vérifiée quand la température devient trop basse.

L'objectif est de tester la validité de cette loi pour l'air. Le matériel à disposition est une seringue, un capteur de presion et une carte Arduino Uno.
Un extrait de la notice du capteur est donné dans le tableau 1.

Symbol	Characteristic	Min	Typ	Max	Unit
-	Accuracy (0 °C to 85 °C)	-	-	±1.5	$\%V_{FSS}$
$\Delta V/\Delta P$	Sensitivity	-	20	-	mV/kPa

Tableau 1 – *Extrait de la notice du capteur de pression.*

▶ Fixer le capteur à la seringue et le brancher au port A0 de la carte.

⚠ *Attention à ne pas appuyer trop fort sur la seringue afin de ne pas détruire le capteur !*

▶ Transférer le programme `tension.ino` vers la carte Arduino.

▶ Tester le fonctionnement du système en vérifiant que la tension affichée dans le moniteur série change lorsque la pression est modifiée par action sur le piston de la seringue.

▶ Brancher l'afficheur LCD sur un port I2C de la carte.

▶ Téléverser `affichage-tension.ino` permettant d'afficher la tension sur l'écran LCD.

▶ Quelle tension est obtenue pour la pression atmosphérique ?

▶ Quelle formule relie la pression et la tension fournie par le capteur ?

▶ Modifier le code pour afficher la pression en hPa.

▶ Régler le volume V à 40 mL. Insérer de nouveau le capteur.

▶ Mesurer la pression pour V valant 25, 30, 35, 40, 45 puis 50 mL.

▶ Quelle est l'incertitude-type $u(V)$ sur le volume ? Quelle est l'incertitude-type $u(p)$ sur la pression ? En déduire l'incertitude sur le produit PV.

▶ En déduire si la loi de Boyle-Mariotte est vérifiée ou non.

Références

[1] R. Taillet, L. Villain & P. Febvre. *Dictionnaire de physique*. 3e éd. De Boeck, 2013.