Microcontrôleurs pour les sciences physiques

David THERINCOURT
Lycée Roland Garros - Académie de la Réunion
29 juin 2025

Plus d'informations sur https://microcontroleurs.david-therincourt.fr/

Table des matières

Les nouveaux programmes 2019 en seconde générale et technologique, en première générale et en terminale générale introduisent l'utilisation des **microcontrôleurs** et des **capteurs** en sciences physiques.

L'objectif de ce document est d'apporter aux professeurs de physique-chimie les notions techniques nécessaires sur les microcontrôleurs afin d'aborder au mieux les nouvelles **capacités numériques**.

Cette documentation est principalement construite autour du microcontrôleur **Arduino UNO** programmer en **langage Arduino (C/C++)**.

Chapitre 1

Introduction

1.1 Qu'est-ce qu'un microcontrôleur ?

Un microcontrôleur est un circuit intégré regroupant un microprocesseur, de la mémoire et des périphériques (ports E/S digitales, ports analogiques, timer, ...) sur la même puce.

FIG. 1 – Microtrôleur ATMEGA328P de l'Arduino Uno R3

Un **microcontrôleur est surtout utilisé pour une application électronique spécifique**. De nos jours, ils sont présents un peu partout : dans les appareils domestiques, médicaux, de télécommunication, dans les voitures, les avions, l'industrie, ...

Apparus dans les années 70, les microcontrôleurs à architecture 8 bits sont encore utilisés. Très peu chère, on les retrouve dans des petites applications (ex. télécommande, appareils de mesure, ...).

La célèbre carte **Arduino UNO R3** est construite autour d'un **microcontrôleur 8 bits** !

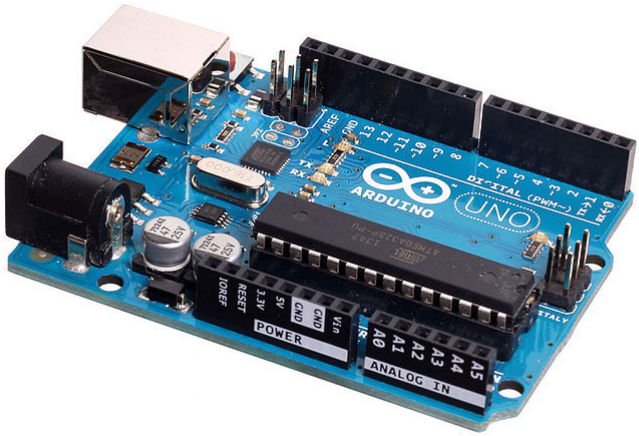


FIG. 2 – Carte Arduino UNO (microcontrôleur Atmel ATMEGA 328)

Actuellement, la tendance est aux **microcontrôleurs 32 bits** (ex. ARM Cortex-M, STM32, ...) qui sont plus adaptés aux applications plus évoluées. C'est ce type de microcontrôleur qui a permis le portage du langage Python (Micropython) au sein des microcontrôleurs. Les cartes Micro :bit, Pyboard ou encore à base d'ESP32 en sont les parfaits exemples !

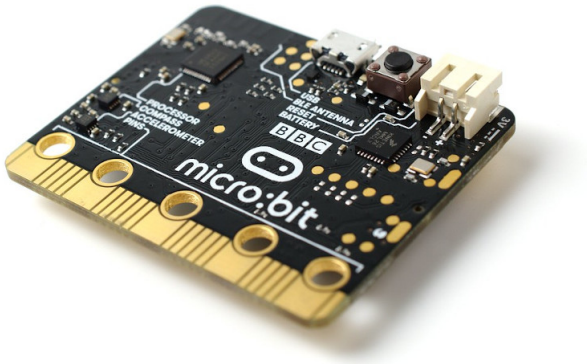


FIG. 3 – Carte micro :bit (Arm® 32-bit Cortex®-M0 / 16 MHz)

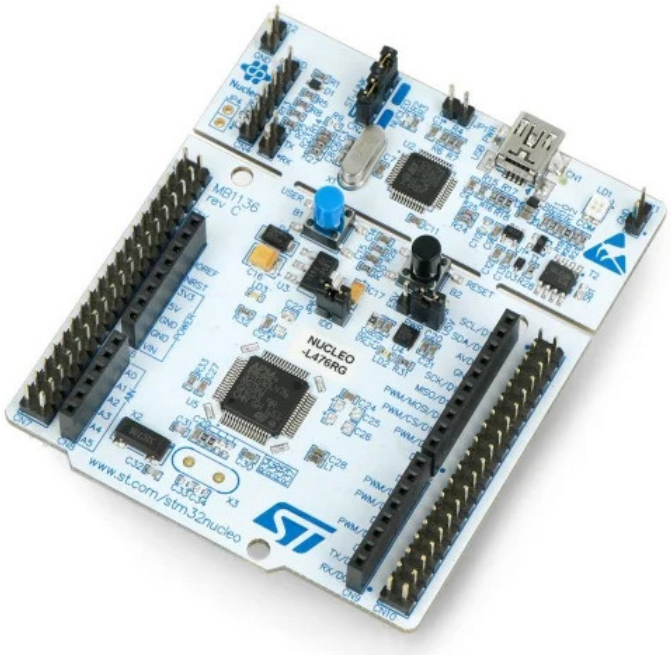


FIG. 4 – Carte Nucleo STM32L476RG (Arm® 32-bit Cortex®-M4 / 80 MHz)

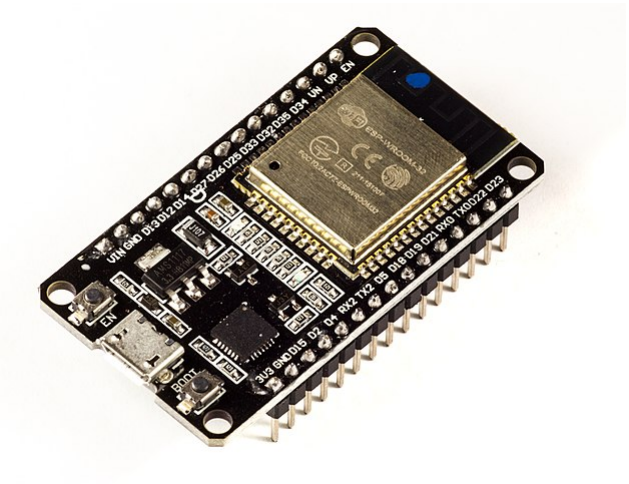


FIG. 5 – Carte ESP-WROOM-32 (Tensilica LX6 Dual-Core)

1.2 Pourquoi des microcontrôleurs en sciences physiques ?

Le monde actuel est fortement imprégné par le numérique. Par exemple, les téléphones portables et les objets connectés comportent une **multitude de capteurs** mesurant des grandeurs très variées comme la température, la fréquence cardiaque, la pression, l'accélération, les ondes sonores, ...

Il est donc important d'expliquer comment s'effectue la **mesure d'une grandeur physique analogique** dans ces appareils numériques du quotidien.

Il en est de même pour la **génération de signaux** (ex. son).

1.3 Les fonctions d'un microcontrôleur

Les microcontrôleurs permettent principalement de :

— **générer de signaux** (ex. son, impulsion de commande, ...).
— **mesurer des tensions** (ex. adaptation de capteurs analogiques, acquisition de signaux, ...).
— **mesurer des durées** (ex. période, fréquence, temps caractéristique, ...).

1.4 Applications en sciences physiques

De manière générale, les microcontrôleurs sont utilisés :

— pour interfacer des **capteurs** analogique ou numérique ;
— pour réaliser des **petites applications** (ex. thermomètre, télémètre à ultrasons, ...) en rapport avec un cours ou un TP ;
— dans des **projets** (enseignement scientifique).

Avec des capteurs, il est en plus possible de :

— réaliser des **mesures** (ex. température, célérité son, pression, ...) ;
— faire de **l'acquisition de données** en mode **autonome** (ex. mesure de pression sur un ballon sonde) ou mode **connecté** (branché à un ordinateur).

Chapitre 2

Cartes Arduino

2.1 Qu'est-ce qu'Arduino ?

Arduino est une **carte électronique** à base de microcontrôleur (ex. ATMEL AVR ATMEGA) développée par Arduino.cc sous licence libre. Tous les schémas des cartes sont disponibles librement sur le Web. A peu près une vingtaine de versions de cartes officielles ont été fabriquées dont la célèbre l'Arduino UNO.

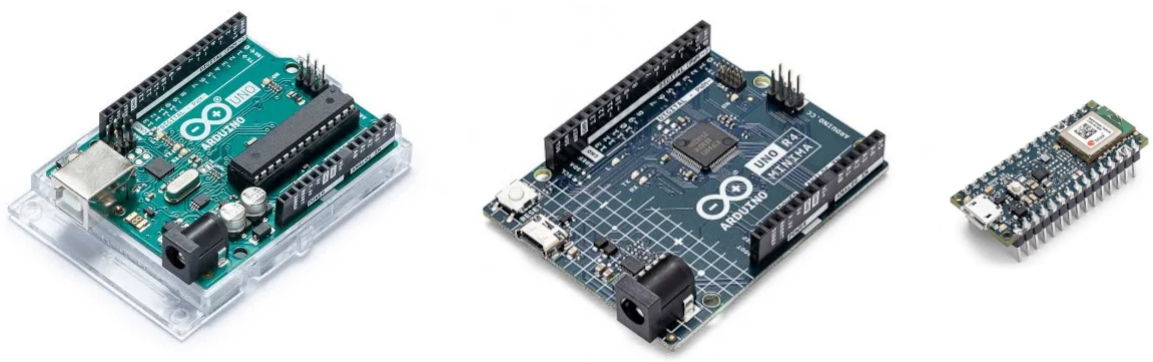


FIG. 1 – Cartes microcontrôleur Arduino

A l'origine conçue pour la **création artistique**, la carte Arduino trouve des applications dans des domaines d'applications aussi variés qu'insolites. Une carte Arduino s'utilise généralement comme :

— **dispositif autonome** dans des applications comme la domotique, la robotique, les systèmes embarqués, ...
— **interface** entre un ordinateur (logiciel tiers) et des capteurs ou des actionneurs ;

Arduino IDE est le **logiciel de développement** des cartes du même nom. Également sous licence libre, cet environnement de développement intégré (IDE) utilise C/C++ comme langage de programmation et le port USB pour le téléversement du programme.

2.2 La carte Arduino UNO (Rev 3)

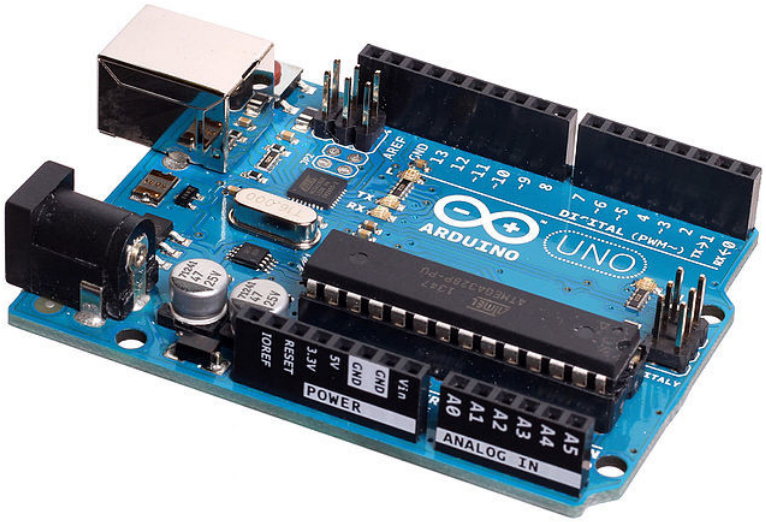


FIG. 2 – Arduino Uno R3 (image : wikipédia)

Arduino UNO est une des cartes officielles les plus récentes et économiques.

Caractéristiques principales :

- microcontrôleur 8 bits ATMEGA328P cadencé à 16 Mhz ;
- alimentation externe (7 à 12 V) ou USB (5 V) ;
- programmation et communication via port USB ;
- 14 broches d'entrées/sorties numériques dont 6 PWM ;
- 6 entrées analogiques sur 10 bits ;
- 1 port I2C (communication avec capteurs/actionneurs numériques) ;
- 1 port UART (communication série) ;
- 3 timers (comptage et mesure de temps) ;
- gestion des interruptions.

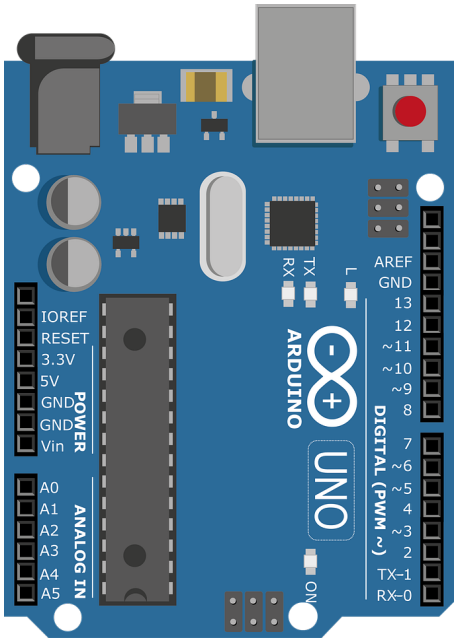


FIG. 3 – Brochage de l'Arduino Uno R3

Avertissement

Les niveaux de tension acceptables sur les broches d'entrées doivent être **comprises entre 0 V et 5 V** sous peine de détruire le microcontrôleur ou la carte !

Note

La carte Arduino UNO ne possède pas de vraies sorties analogiques mais des sorties à **modulation de largeur d'impulsion** (MLI). Ce sont les six fameuses **sorties PWM** (Pulse Width Modulation).

2.3 Educaduino Lab (Eurosmart)

https://educaduino-lab.com/

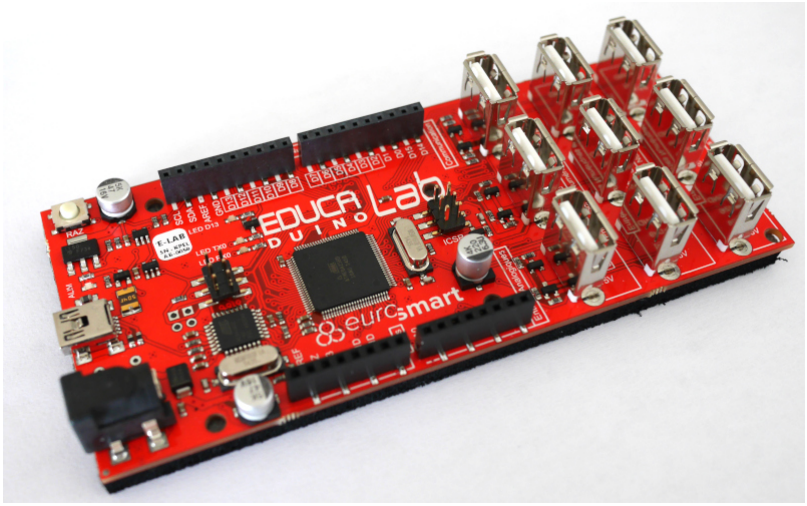


FIG. 4 – La carte Educaduino-Lab (E-LAB)

La carte **Educaduino Lab** a été conçue sur la base d'une carte Arduino MEGA 2560. Cette dernière est équivalente à une carte arduino UNO mais avec plus de mémoire et surtout **plus de ports d'entrée/sortie**. Ce qui a permis à Eurosmart d'y placer des **connecteurs USB pour ses propres capteurs** tout en gardant la connectique classique de l'Arduino UNO.

Caractéristiques principales :

— microcontrôleur ATMEGA 2560 (comme l'Arduino MEGA 2560) ;
— protection des ports d'entrée/sortie ;
— brochage compatible Arduino Uno Rev 3 (pin 0.8mm, shield Grove, ...) ;
— ports supplémentaires en USB pour capteurs Educaduino-Lab ;

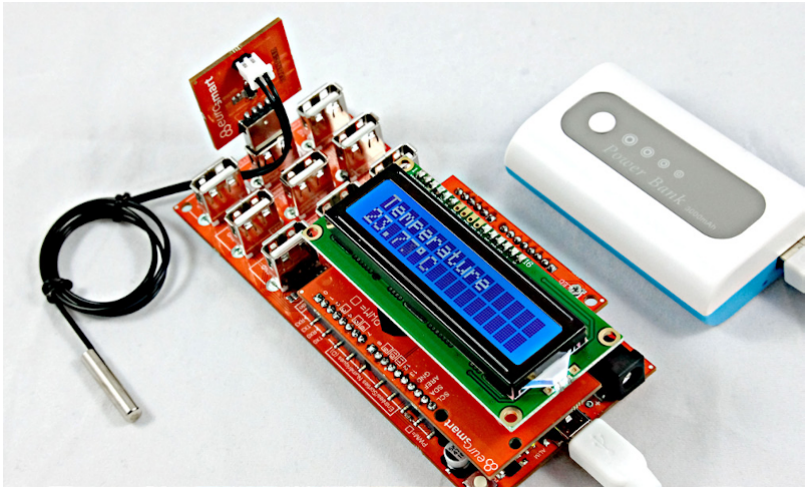


FIG. 5 – Mesure d'une température (image : Eurosmart)

Une malette avec un afficheur LCD et plusieurs capteurs adaptés au programme du lycée est également proposée.

FIG. 6 – Kit sciences-physiques 2nde/1ère (image : Eurosmart)

2.4 Plug'Uino® Uno (Sciencéthic)

https://www.sciencethic.com/

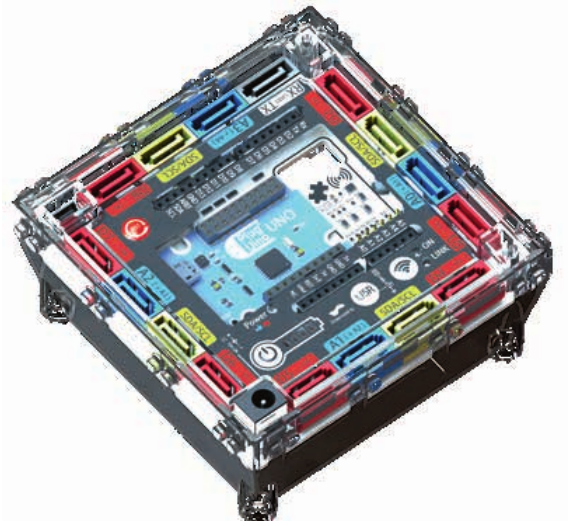


FIG. 7 – La carte Plug'Uino ® Uno (image : Sciencéthic)

Sciencéthic propose également une carte **Plug'Uino Uno** protégée contre les mauvaises manipulations et 100% compatible Arduino UNO Rev 3.

Caractéristiques principales :

— microcontrôleur ATMEGA 328P (comme l'Arduino Uno) ;

— protection des ports d'entrée/sortie ;
— brochage compatible Arduino Uno Rev 3 (pin 0.8mm, shield Grove, ...) ;
— connecteurs SATA pour les capteurs Plug'uino ;

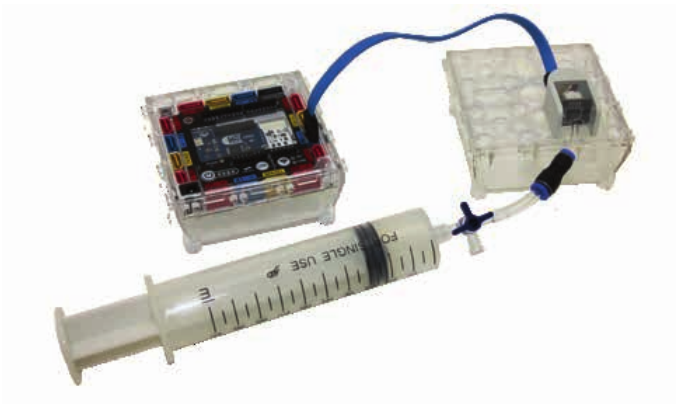


FIG. 8 – Capteur de pression et loi de Mariotte (image : Sciencéthic)

Chapitre 3

Langage Arduino

3.1 Logiciel Arduino IDE

3.1.1 Arduino IDE (version 1.8)

Le logiciel **Arduino IDE 1.8** est un environnement intégré de développement (IDE) multiplaforme. Il est téléchargeable sur le site officiel http://www.arduino.cc/en/

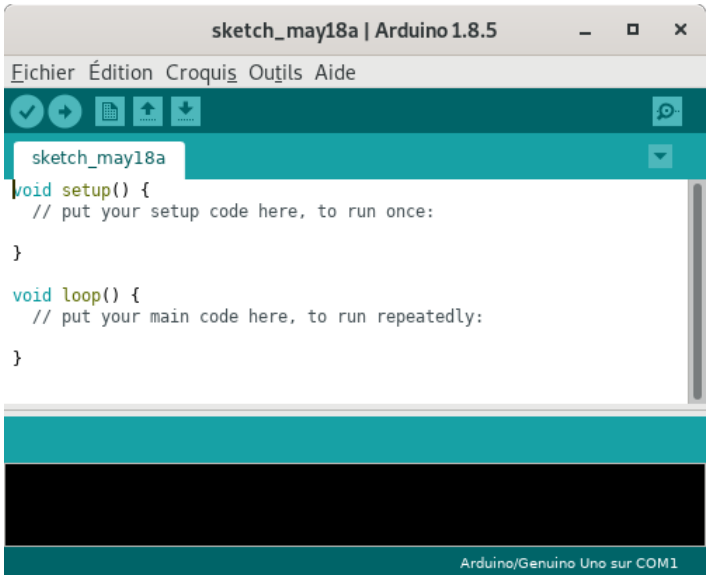


FIG. 1 – Logiciel Arduino IDE 1.8

3.1.2 Arduino IDE (version 2.x)

Le logiciel Arduino IDE 2 est la nouvelle version de l'environnement intégré de développement (IDE) d'Arduino.

FIG. 2 – Logiciel Arduino IDE 2

3.1.3 Etapes de la mise en oeuvre d'un projet Arduino

La mise en œuvre d'un projet Arduino s'effectue dans l'ordre suivant :

1. **Édition** du code source du programme dans l'éditeur de l'interface ;
2. **Compilation** du code source du programme (vérification du code et génération du programme en langage machine) ;
3. **Téléversement** du programme sur la carte Arduino ;
4. **Exécution** du programme sur le carte Arduino (de façon autonome sans ordinateur).

3.2 Premier programme : Blink

Le programme **Blink** a pour objectif de faire clignoter la LED intégrée à la carte de développement. Cette LED est connectée en interne à la broche 13.

3.2.1 Edition

Le programme **Blink** est disponible dans les exemples du logiciel **Arduino IDE**.

Dans le menu `Fichier > Exemples > Basics > Blink`.

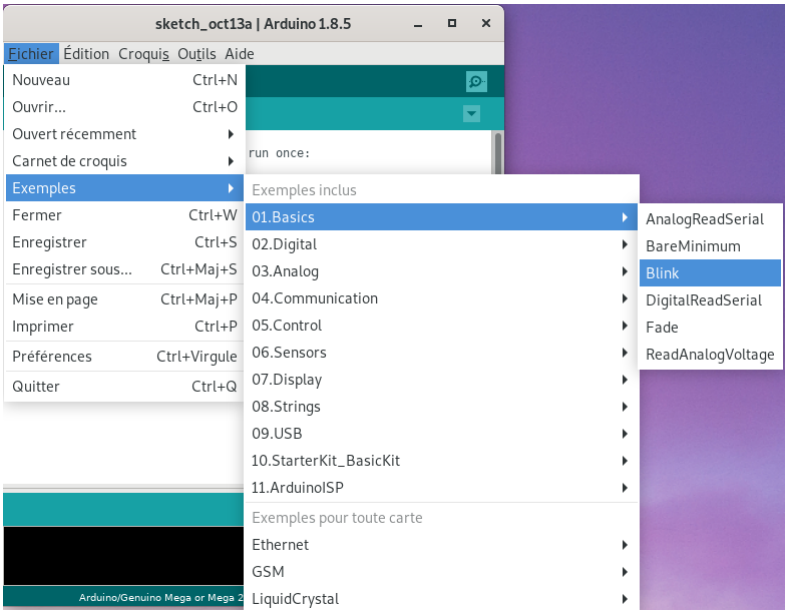


FIG. 3 – Ouvrir le code source du programme Blink

FIG. 4 – Édition du code source du programme Blink

Note

— Un programme Arduino écrit en **langage C/C++** est composé **d'une suite d'intructions**.
— Ces instructions sont exécutées dans **l'ordre des lignes de code**.
— Les **commentaires** en gris sont délimités par les caractères `/*` et `*/` sur plusieurs lignes ou commencent pas les caractères `//` sur une même ligne.

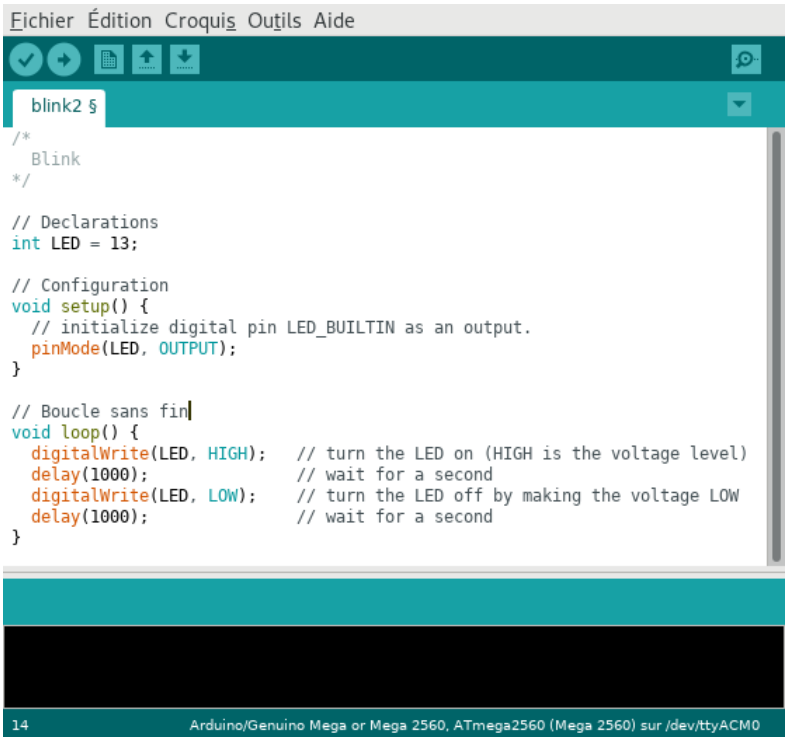


FIG. 5 – Une version modifiée du code source du programme Blink

Avertissement

Un programme Arduino respecte toujours une **structure spécifique** composée en trois parties :

- Les déclarations : **définitions** des constantes et des variables ;
- La fonction `setup()` : **configuration** de la carte (entrées, sorties, port série, ...) ;
- La fonction `loop()` : **instructions du programme exécutées** dans une **boucle infinie** (sans fin).

3.2.2 Compilation

Avertissement

Avant de lancer la compilation, il est important de **choisir le modèle de carte Arduino utilisé**. Le programme généré est dépendant du type de microcontrôleur présent sur la carte.

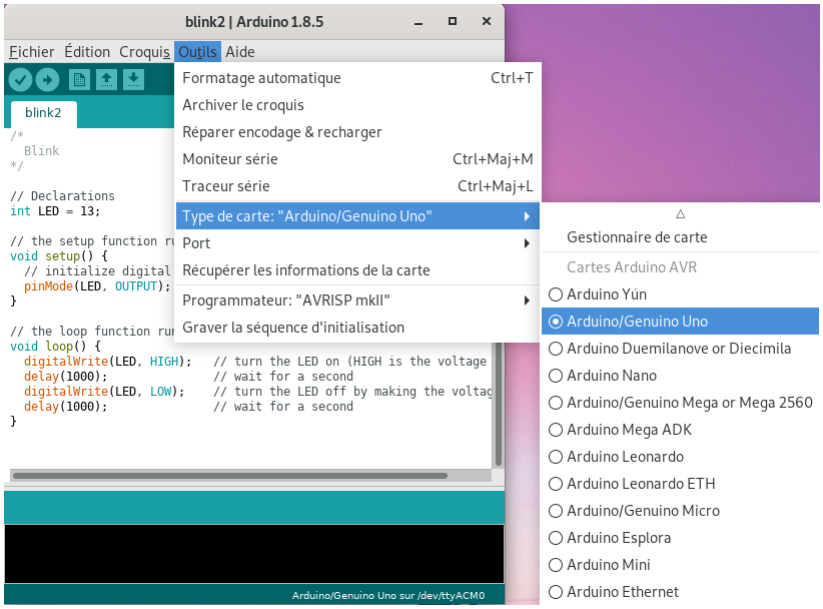


FIG. 6 – Choix du type de carte

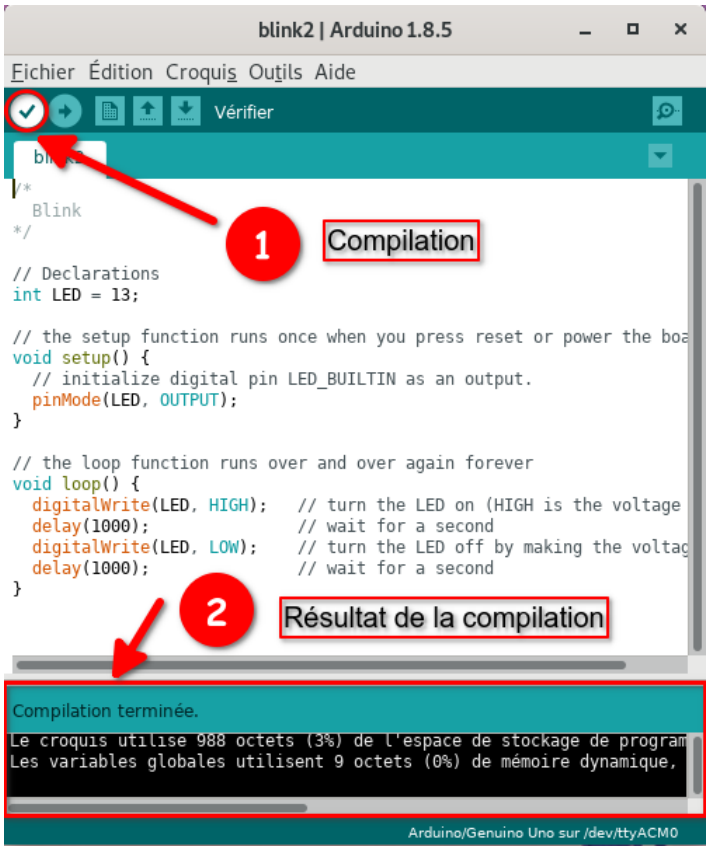


FIG. 7 – Puis la compilation peut s'effectuée !

3.2.3 Téléversement

Avertissement

Pour téléverser le programme obtenu, il faut **sélectionner le port de communication série** sur lequel est connectée la carte Arduino.

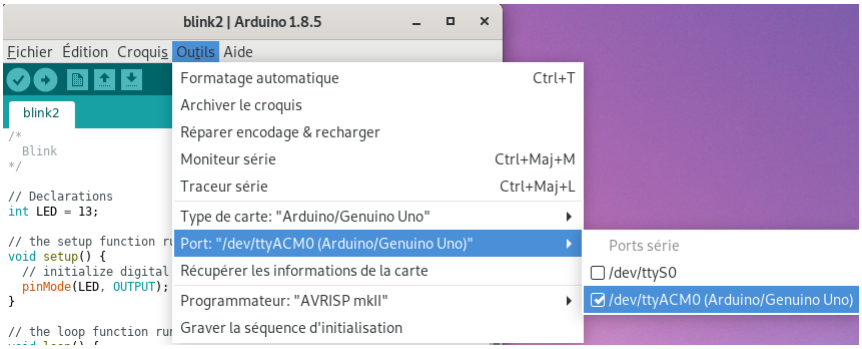


FIG. 8 – Choix du port de communication

FIG. 9 – Téléversement du programme

3.2.4 Exécution

Le programme s'exécute sur la carte Arduino de façon autonome (sans ordinateur).

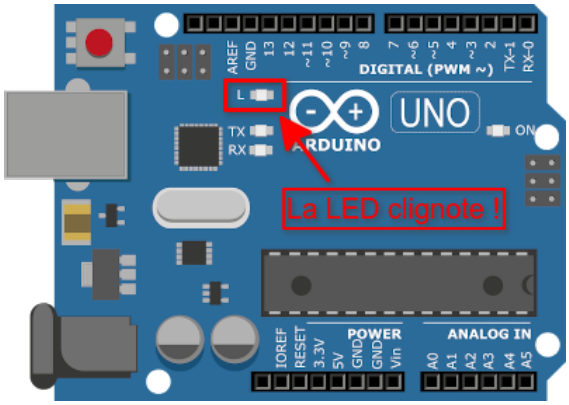


FIG. 10 – Exécution du programme Blink sur la carte Arduino Uno R3

3.3 Spécificités du langage Arduino

Le langage de programmation C/C++ est utilisé par le logiciel Arduino IDE pour programmer les microcontrôleurs Arduino.

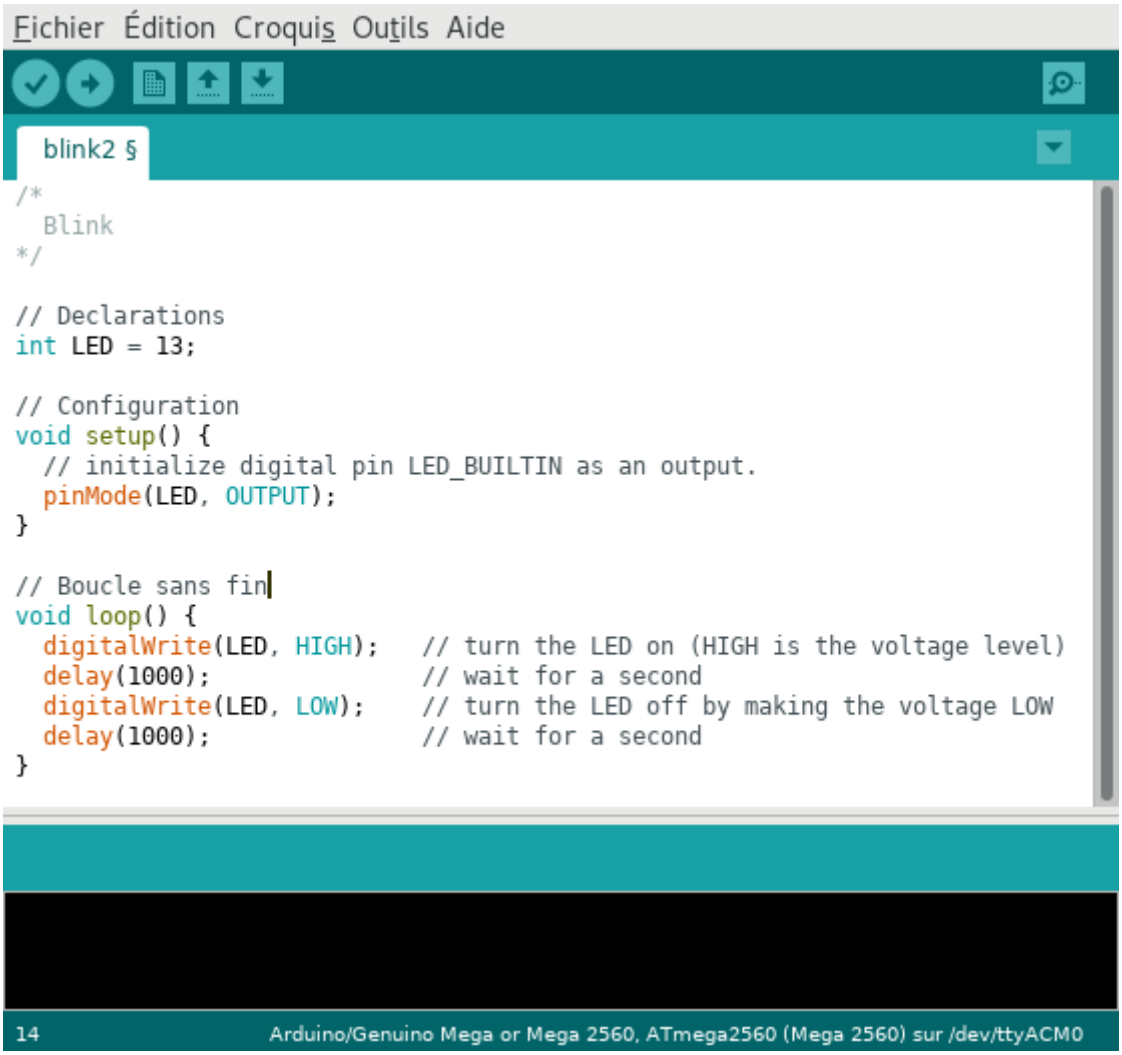


FIG. 11 – Edition d'un code source d'un programme Arduino

3.3.1 Syntaxe

— Toutes les instructions se terminent par un point virgule `;` sauf pour les directives `#include` et `#define`.
— Les **blocs d'instructions** sont délimités par des accolades `{...}`.
— Les **commentaires** en gris sont délimités par les caractères `/*` et `*/` sur plusieurs lignes ou commencent pas les caractères `//` sur une même ligne.

3.3.1.1 Typage des variables

Le type d'une variable doit être renseigné à sa déclaration.

Quelques types disponibles :

Type	Description	Valeurs
`int`	entier sur 16 bits	-32768 à 32767
`long`	entier sur 32 bits	-2147483648 à 2147483647
`float`	flottant sur 32 bits	-3.4028235E+38 à -3.4028235E+38 ;
`char`	caractère sur 8 bits	Table ASCII

Exemples :

```
int a = 5;
float pi = 3.14;
char c = 'A';
```

3.3.2 Constantes prédéfinies

Afin d'améliorer la lecture du code, des constantes sont définies.

Constante	Valeur
`LOW`	0 (niveau logique)
`HIGH`	1 (niveau logique)
`OUTPUT`	broche en sortie
`INPUT`	broche en entrée
`LED_BUILTIN`	13 (numéro de broche de la LED intégrée)

3.3.3 Structure du programme

Un programme Arduino respecte toujours une **structure spécifique** composée en trois parties :

— Les déclarations : **définitions** des constantes et des variables ;
— La fonction `setup()` : **configuration** de la carte (entrées, sorties, port série, ...) ;
— La fonction `loop()` : **instructions du programme exécutées** dans une **boucle infinie** (sans fin).

3.3.4 Documentation

Une référence complète du langage Arduino est disponible sur le site officiel ici.

Chapitre 4

Les bases d'Arduino

4.1 Allumer une LED (sorties numériques)

4.1.1 Principe

Les entrées/sorties numériques de la carte Arduino UNO sont accessibles sur les broches 0 à 13.

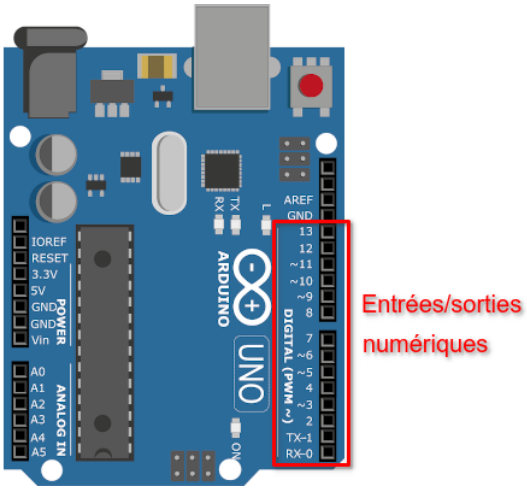


FIG. 1 – Entrées/sorties numériques

> **Avertissement**
>
> Une sortie numérique ne peut délivrer que 40 mA au maximum (200 mA pour l'ensemble des sorties). Au delà de cette valeur, la carte peut-être détériorée !

4.1.2 Montage

Une LED en série avec une résistance est connectée entre la broche 11 et la masse (GND).

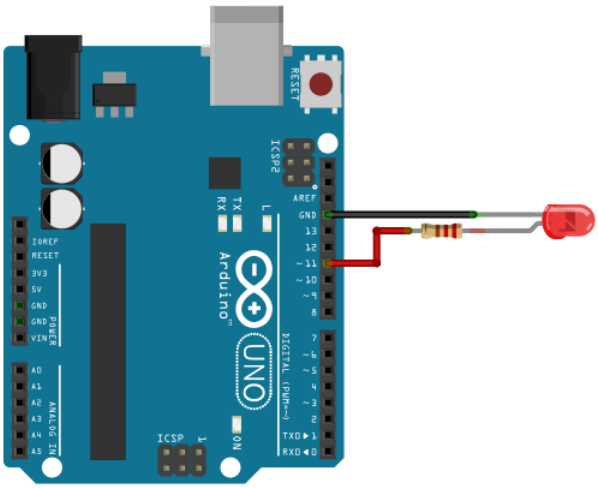


FIG. 2 – Branchement d'une LED sur la broche 11

La formule suivante donne le calcul de la résistance en fonction du courant nominal et de la couleur de la LED :

$$R = \frac{V_{CC} - V_S}{I}$$

— V_{CC} est la tension d'alimentation (5 V) ;
— $V_S \approx 2V$ est la tension de seuil de la LED (dépend de la couleur) ;
— I est l'intensité du courant généralement de l'ordre de 20 mA.

Une valeur de résistance de 220 Ω donne un bon compromis.

4.1.3 Programme en langage Arduino (C/C++)

```
int pinLED = 11;              // LED connectée sur broche 11

void setup() {
  pinMode(pinLED, OUTPUT);    // Broche LED paramétrée en sortie
}

void loop() {
  digitalWrite(pinLED,0);     // LED éteinte
  delay(1000);                // Attendre 1000 ms = 1s
  digitalWrite(pinLED,1);     // LED allumée
  delay(1000);                // Attendre 1000 ms = 1s
}
```

Dans la fonction `setup()` :

— `pinMode(LED,OUTPUT)` paramètre la broche `LED` en sortie (`OUPUT`).

Dans la fonction `loop()` :

— `digitalWrite(LED,LOW)` fixe un niveau logique 0 (`LOW`) sur la broche `LED`.
— `digitalWrite(LED,HIGH)` fixe un niveau logique 1 (`HIGH`) sur la broche `LED`.

4.1.4 Applications

— Commande d'un actionneur (LED, relais, ...) en tout ou rien.
— Communication numérique.

4.2 Modifier l'intensité lumineuse d'une LED (sorties PWM)

4.2.1 Principe

La carte Arduino ne possède pas de vraies sorties analogiques. Mais il est possible de faire **varier la valeur moyenne de la tension d'une sortie digitale** (donc de faire varier l'intensité lumineuse d'une LED) en modifiant son **rapport cyclique** (durée de l'état haut par rapport à la période). C'est le principe de la Modulation à Largeur d'Impulsion (MLI) ou Pulse Width Modulation (PWM) en anglais.

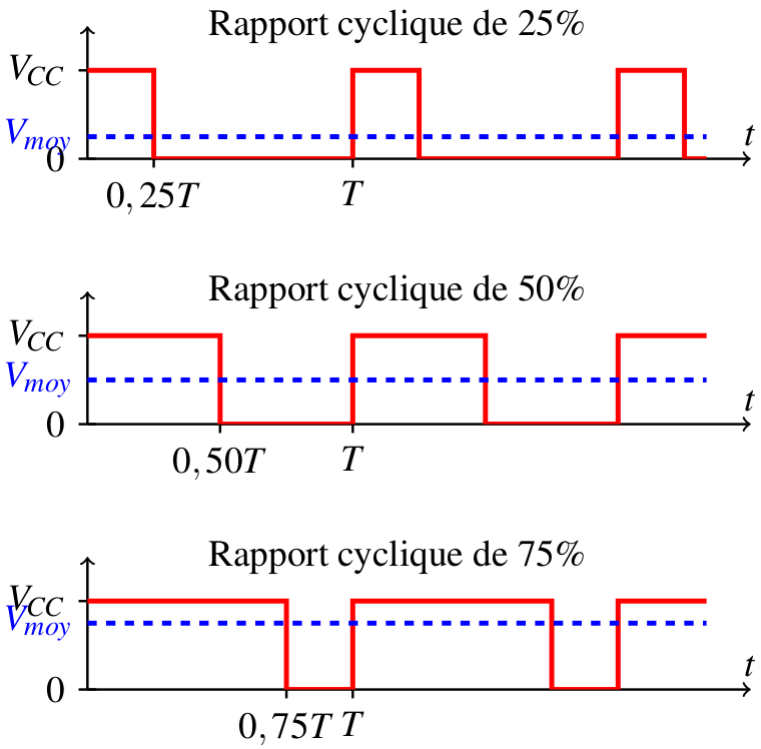


FIG. 3 – Principe de la MLI ou PWM

La carte Arduino UNO est dotée de **6 sorties PWM sur les broches 3, 5, 6, 9, 10, 11**.

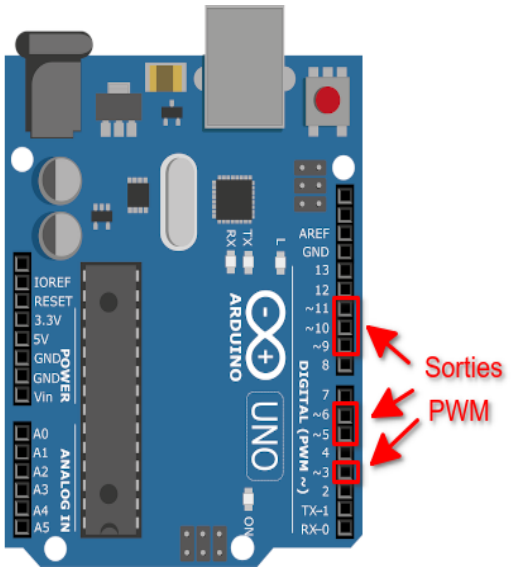


FIG. 4 – Soties PWM de l'Arduino Uno R3

Avertissement

La fréquence d'un signal PWM est fixée à 490 Hz pour un arduino Uno R3 !

4.2.2 Montage

Une LED en série avec une résistance de 220 Ω est branchée sur la broche 11.

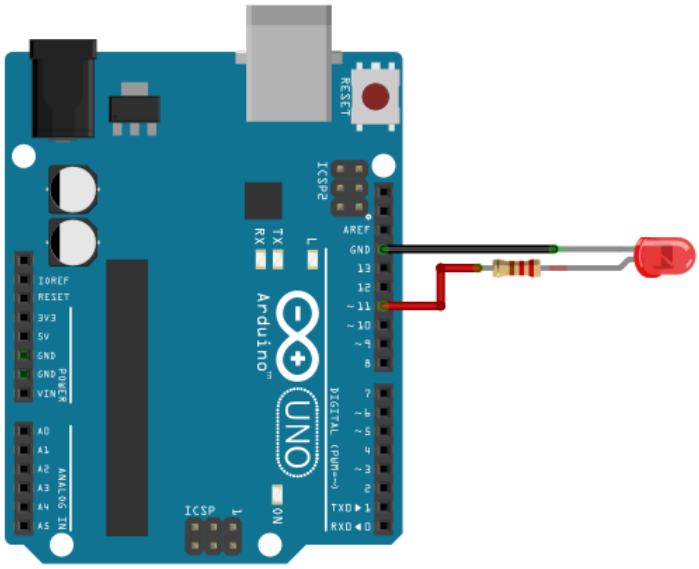


FIG. 5 – Branchement d'une LED sur la broche 11

4.2.3 Programme en langage Arduino (C/C++)

```
#define LED 11            // LED connectée à la broche 11

void setup() {
  pinMode(LED,OUTPUT);    // Configuration de la broche LED en sortie
}

void loop() {
  analogWrite(LED,0);     // PWM à 0%
  delay(1000);            // Attendre 1 s
  analogWrite(LED,100);   // PWM à 100/255 = 39%
  delay(1000);            // Attendre 1 s
  analogWrite(LED,200);   // PWM à 200/255 = 78%
  delay(1000);            // Attendre 1 s
}
```

La fonction `analogWrite(LED,duty)` génère une modulation à largeur d'impulsion sur la broche 11 où `duty` est un nombre entier entre 0 et 255 respectivement pour un rapport cyclique entre 0% et 100%.

4.2.3.1 Applications

— Variation de l'intensité lumineuse d'une LED.
— Variation de la vitesse d'un moteur à courant continu.

— Obtention d'une tension constante par filtrage passe-bas (limitée en fréquence).

4.3 Moniteur série - Afficher des messages

4.3.1 Principe

De base, les cartes Arduino ne possède pas d'écran pour afficher des messages. L'interface série (UART) reste le moyen le plus simple pour afficher des informations sur un ordinateur en provenance d'une carte Arduino.

4.3.2 Moniteur série

Le logiciel Arduino IDE intègre un **moniteur série** (dans le menu `Outils > Moniteur série`) pour **lire des données** au format texte (ASCII) envoyées par le microcontrôleur.

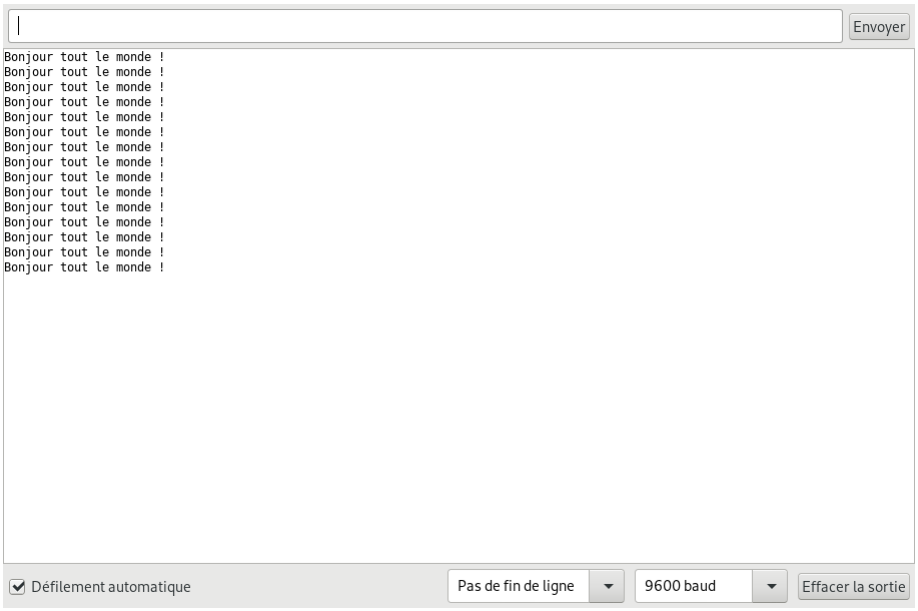


FIG. 6 – Moniteur série du d'Arduino IDE 1.8

Le moniteur série offre aussi la possibilité de **transmettre des données vers le microcontrôleur** à l'aide du clavier !

Note

Il est possible d'utiliser d'autres logiciels de communication série comme Putty ou encore Termite.

4.3.3 Programme de test

Le code source ci-dessous transmet un entier `N` dans le moniteur série d'Arduino IDE. Cet entier, initialisé à 0, est incrémenté de 1 toutes les secondes.

```
int n = 0;

void setup() {
  Serial.begin(9600);     // Paramétrage du port série
```

(suite sur la page suivante)

(suite de la page précédente)

```
}

void loop() {
  Serial.print("N = ");    // Affichage
  Serial.println(n);       // Affichage du contenu de la variable n
  n = n + 1;               // Incrémentation de la variable n
  delay(1000);             // Temporisation de 1s
}
```

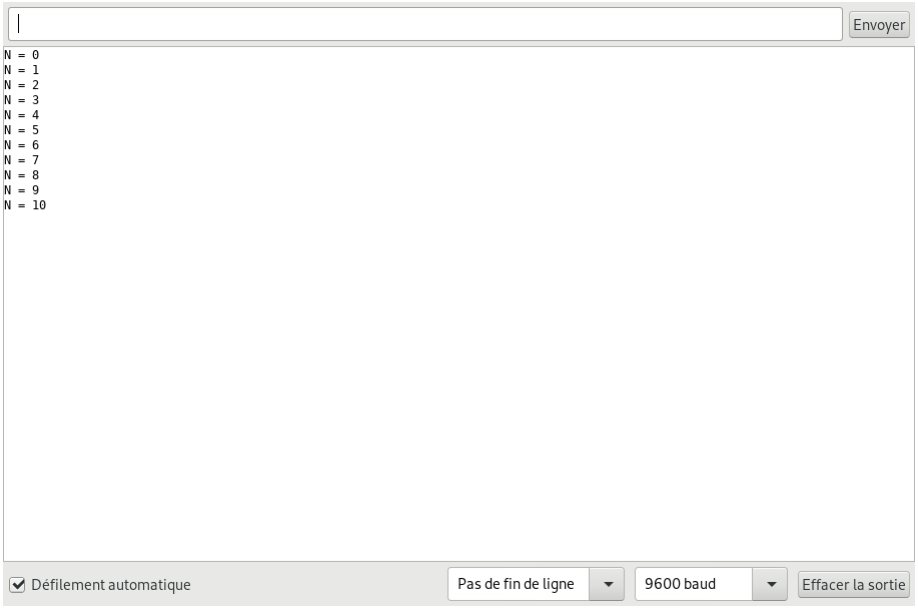


FIG. 7 – Affichage dans le moniteur série

— L'instruction `Serial.begin(9600)` paramètre le port série à 9600 baud (rapidité).
— `Serial.print("N = ")` affiche la chaîne de caractères `N =` dans le moniteur série.
— `Serial.println(n)` affiche le contenu de la variable `n` suivie d'un saut de ligne.

4.3.4 Applications

— Affichage d'une ou plusieurs mesures sur l'écran d'un ordinateur.
— Affichage des données d'une acquisition au format CSV pour exploitation par un tableur, des logiciels spécialisés (Regressi, Latis, ...) ou des calculatrices Web (ex. Desmos).

4.4 Moniteur série - Lire une valeur au clavier

4.4.1 Montage

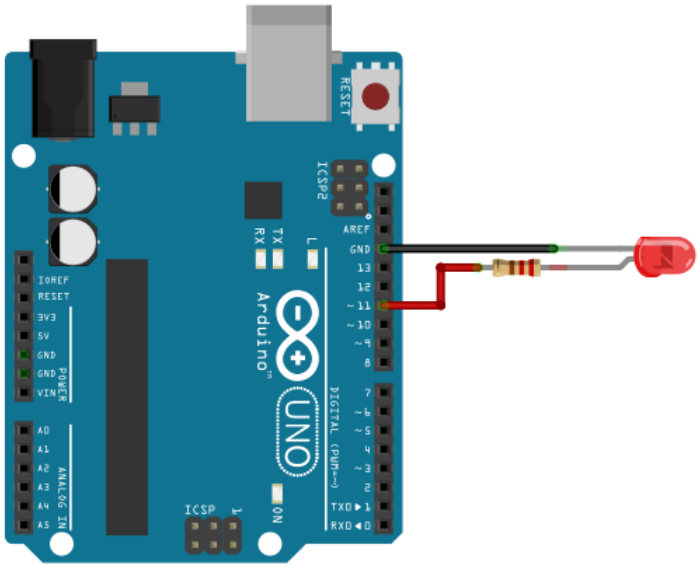


FIG. 8 – Branchement d'une LED sur la broche 11

Une LED en série avec une résistance de 220 Ω est branchée sur la broche 11.

4.4.2 Programme

Le programme ci-dessous contrôle l'intensité de la LED interne d'un Arduino avec le clavier.

- Le rapport cyclique (entier entre 0 et 255) est saisie au clavier dans le moniteur série avec la fonction `parseInt()` de la librairie `Serial`.
- Un signal PWM est généré sur la broche 11 (LED interne) avec la fonction `analogWrite()`.

```
// PWM avec saisie du rapport cyclique (entier de 0 à 255) au clavier dans le moniteur␣
 ↪série.
// ATTENTION : Sélectionner "Pas de fin de ligne" dans le monitor série !!!
// David THERINCOURT 2025

#define LED 11           // LED connectée à la broche 11

void setup()
{
   Serial.begin(9600);   // Initialisation du port série
   pinMode(LED,OUTPUT);  // Configuration de la broche LED en sortie
}

void loop()
{
   Serial.print("Rapport cyclique entre 0 et 255 : ");  // Indication à l'utilisateur
   while (Serial.available()==0){}                      // Attente d'un message (Cocher
 ↪"Pas de fin de ligne")
   int N = Serial.parseInt();                           // Extraction de la valeur␣
 ↪numérique (entier)
   Serial.println(N);                                   // Affichage de N
   analogWrite(LED, N);                                 // Ecriture sur la sortie PWM
}
```

4.5 Mesurer une tension (CAN)

4.5.1 Principe

La mesure d'une tension par un microcontrôleur est prise en charge en interne par un **Convertisseur Analogique-Numérique (CNA)**.

Ce type de conversion est importante en sciences physique. Par exemple, elle permet d'obtenir la mesure d'une grandeur physique provenant d'un **capteur** sous la forme d'une tension électrique.

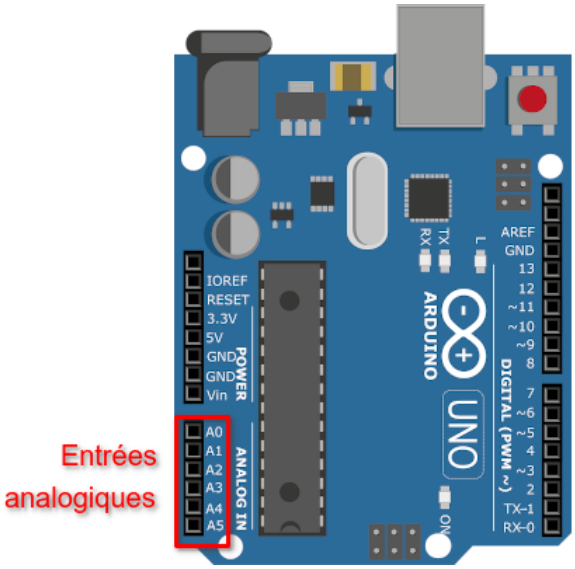


FIG. 9 – Entrées analogiques de l'Arduino Uno R3

Le microcontrôleur de l'Arduino UNO intègre **6 entrées analogiques** disponibles sur les broches A0 à A5. Par défaut, la tension d'entrée doit-être comprise entre 0 V et 5 V (Vref).

Le conversion renvoie un **nombre entier naturel sur 10 bits** compris entre 0 et 1023.

La résolution analogique (**quantum**) est donc :

$$q = \frac{5}{1023} \approx 4,89\,mV$$

Avertissement

Le tension appliquée sur les entrées analogiques doivent être **strictement comprise entre 0 V et 5 V** sous peine de détruire le microcontrôleur.

4.5.2 Montage

Un potentiomètre (pont diviseur de tension) est branché entre la masse (`GND`) et la tension d'alimentation (`5V`). Ce potentiomètre délivre donc une **tension réglable** en `0 V` et `5 V` sur l'entrée `A0`.

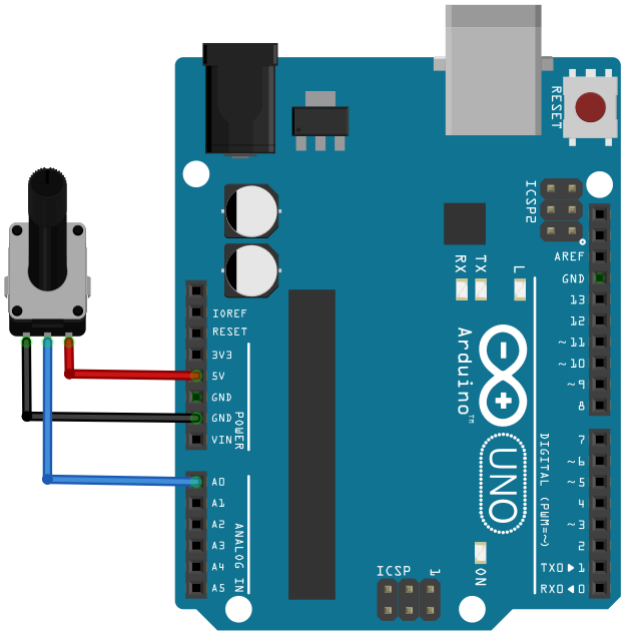


FIG. 10 – Montage potentiométrique sur l'entrée analogique A0

4.5.3 Programme Arduino

Le programme suivant lit la tension sur l'entrée `A0` et affiche sa valeur (en volt) dans le moniteur série du logiciel Arduino IDE.

```
/*
 * Lecture de l'entrée analogique A0 sur le port série.
 */

int N;           // Entier lu sur A0 compris entre 0 et 1023 (10bits)
float tension;   // La tension calculée à partir de N

void setup() {
  Serial.begin(9600);   // Paramétrage port série
}

void loop() {
  N = analogRead(A0);                    // Lecture valeur sur A0
  tension = N*5.0/1023;                  // Calcul de la tension
  Serial.print("Valeur lue sur A0 = ");
  Serial.println(N);                     // Affichage valeur sur A0
  Serial.print("Tension = ");
  Serial.println(tension);               // Affichage de la tension
  delay(1000);                           // temporisation de 1 s
}
```

— La fonction `analogRead(A0)` retourne un entier sur 10 bits compris entre 0 (pour 0V) et 1023 (pour 5V).
— L'expression `N*5.0/1023` calcule la valeur de la tension en volt.

4.5.4 Applications

— Interface avec un circuit comportant un capteur.

— Un potentiomètre est un capteur de position.

Chapitre 5

Capteur résistif (seconde générale)

Programme de seconde générale 2019 - Enseignement commun

Mesurer une grandeur physique à l'aide d'un capteur électrique résistif. **Produire et utiliser une courbe d'étalonnage** reliant la résistance d'un système avec une grandeur d'intérêt (température, pression, intensité lumineuse, etc.). Utiliser un dispositif avec microcontrôleur et capteur.

5.1 Introduction

La résistance électrique d'un capteur résistif varie en fonction de la grandeur physique mesurée (ex. température, pression, intensité lumineuse, ...).

FIG. 1 – Capteur de température CTN / Capteur de lumière LDR

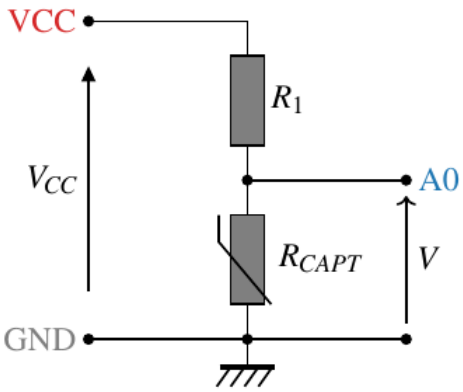
Pour obtenir une valeur de cette grandeur physique, il est nécessaire de **mesurer la résistance du capteur**.

Malheureusement un microcontrôleur ne mesure que des tensions sur ses entrées analogique (ex. A0, A1, ... pour Arduino). Il faut donc **adapter le capteur dans un montage électrique** (ex. pont diviseur de tension, pont wheatstone).

5.2 Techniques de mesure d'une résistance

La plupart des modules avec capteur résistif utilise un **pont diviseur de tension** pour la mesure de la résistance du capteur. Par rapport au pont Wheatstone, cette méthode présente l'avantage d'être simple à mettre en oeuvre.

5.2.1 Montage 1 : capteur connecté à la masse



$$R_{CAPT} = R_1 \times \frac{V}{V_{CC} - V}$$

FIG. 2 – Schéma électrique

— VCC est la tension d'alimentation du microcontrôleur.
— L'entrée analogique A0 mesure la tension V aux bornes du capteur résistif.

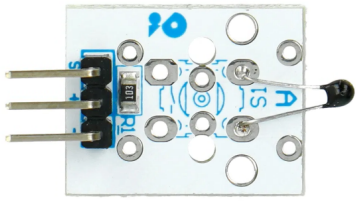
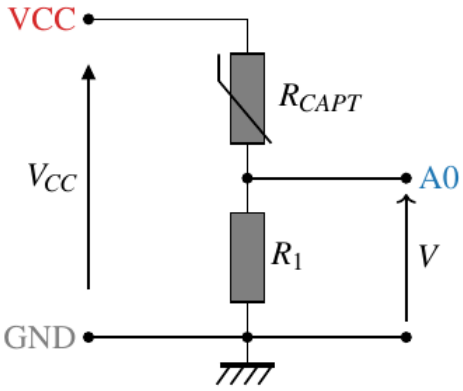


FIG. 3 – Module Velleman VMA320 ($R_1 = 10$ kΩ)

5.2.2 Montage 2 : capteur connecté à Vcc



$$R_{CAPT} = R_1 \times (\frac{V_{CC}}{V} - 1)$$

FIG. 4 – Schéma électrique

— VCC est la tension d'alimentation du microcontrôleur.
— L'entrée analogique A0 mesure la tension V aux bornes de la résistance R.

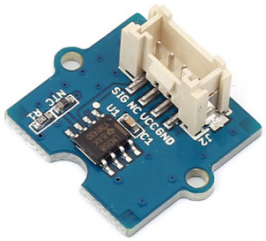


FIG. 5 – Module Grove - Temperature Sensor V1.2 ($R_1 = 100$ kΩ)

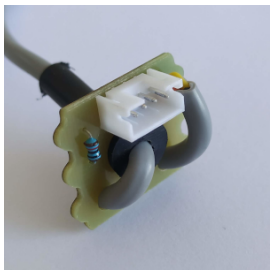


FIG. 6 – Module CTN Plug'uino ($R_1 = 10$ kΩ)

5.2.3 Montage 3 : mesure de la tension et du courant (ex. capteurs Educaduino)

En plus de la **mesure de la tension** du capteur, une **mesure du courant** est aussi réalisée à partir de la tension aux bornes de la résistance R par l'intermédiaire d'un amplificateur différentiel. La résistance du capteur est alors calculée avec la **loi d'Ohm**.

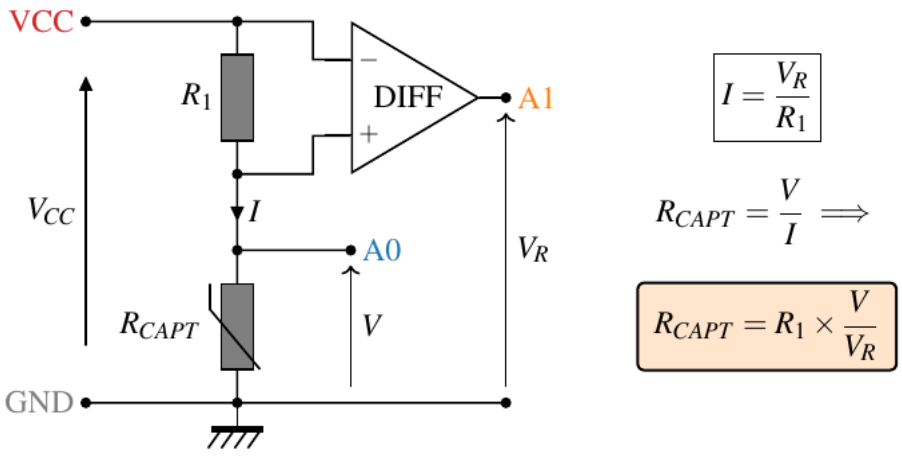


FIG. 7 – Schéma électrique

— L'entrée analogique A0 mesure la tension V aux bornes du capteur résistif.
— L'entrée analogique A1 mesure la tension V_R aux bornes de la résistance R.

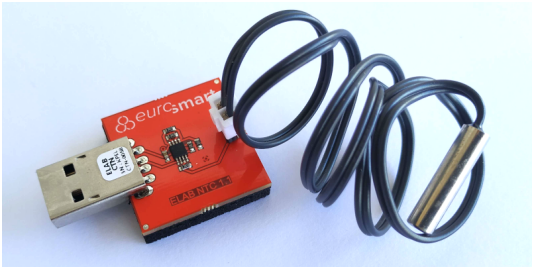


FIG. 8 – Module CTN 10K Educaduino LAB ($R_1 = 10$ kΩ)

5.3 Cas d'un capteur résistif CTN

Une CTN est un **capteur résistif à coefficient de température négatif** ...

5.3.1 Courbe d'étalonnage

Les mesures sont effectuées avec un thermomètre et un **ohmmètre**.

T (°C)	R (Ohm)
2.4	25455
5.1	22714
10.0	18622
15.1	15201
20.0	12607
25.0	10475
30.0	8740
35.1	7333
40.0	6194
45.0	5217
50.0	4358
55.1	3689
60.0	3120
65.1	2647
70.1	2264
75.1	1926
80.0	1658

Téléchargement : `data_ctn_1.txt` (mesures au format CSV).

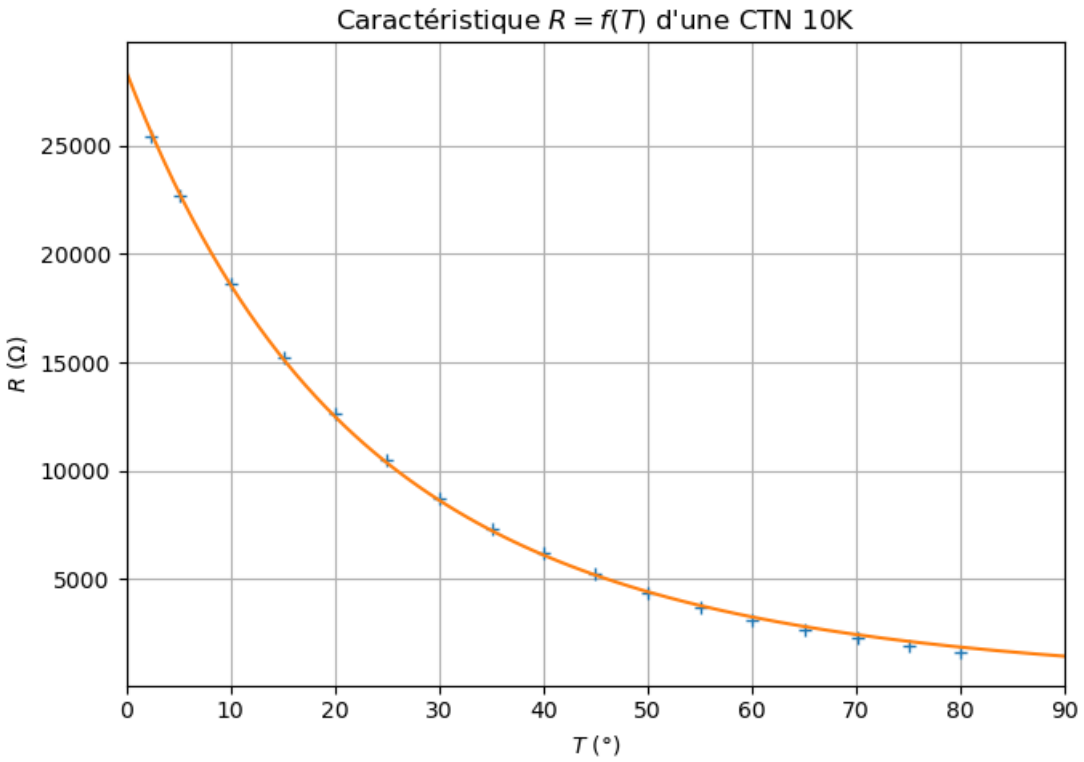


FIG. 9 – Courbe d'étalonnage d'une CTN 10k

Note

Dans cet exemple, la résistance mesurée prend la valeur particulière de 10 kΩ pour 25°C !

5.3.2 Relation de Steinhart-Hart

Sur une grande plage de variation, la relation entre la température (en K) et la résistance de la CTN est :

$$\frac{1}{T} = A + B \times \ln(R) + C \times (\ln(R))^3$$

A, B et C sont les coefficients de Steinhart-Hart. Ils sont donnés par le constructeur ou peuvent se déterminer expérimentalement à l'aide du programme Python `steinhart-hart.py` à partir de trois points de la courbe d'étalonnage.

Résultats obtenus à partir du programme Python :

$$A = 1,144 \cdot 10^{-3} K^{-1} \qquad B = 2,078 \cdot 10^{-3} K^{-1} \qquad C = 3,610 \cdot 10^{-7} K^{-1}$$

5.3.3 Relation simplifiée de Steinhart-Hart

Sur une **plage de variation plus réduite de la température**, la relation de Steinhart-Hart permet d'écrire :

$$R \approx R_0 \times e^{\beta(\frac{1}{T} - \frac{1}{T_0})}$$

— R_0 est la valeur de la résistance pour la température T_0.
— β est le coefficient de température (en K).

Ces coefficients sont généralement donnés par le constructeur (datasheet).

Module	Référence CTN	R_0 (kΩ)	β (K)
Grove - Temperature Sensor V1.2	NCP18WF104F03RC	100	4255 (25/80°C)
Velleman VMA320	NTC-MF52 3950	10	3950

Note

Expression de la constante β à partir des températures T_0 et T :

$$\beta = \frac{\ln(\frac{R}{R_0})}{\frac{1}{T} - \frac{1}{T_0}}$$

Les coefficients R_0 et β peuvent être également déterminés par une modélisation de la caractéristique sur la plage de température d'utilisation de la CTN.

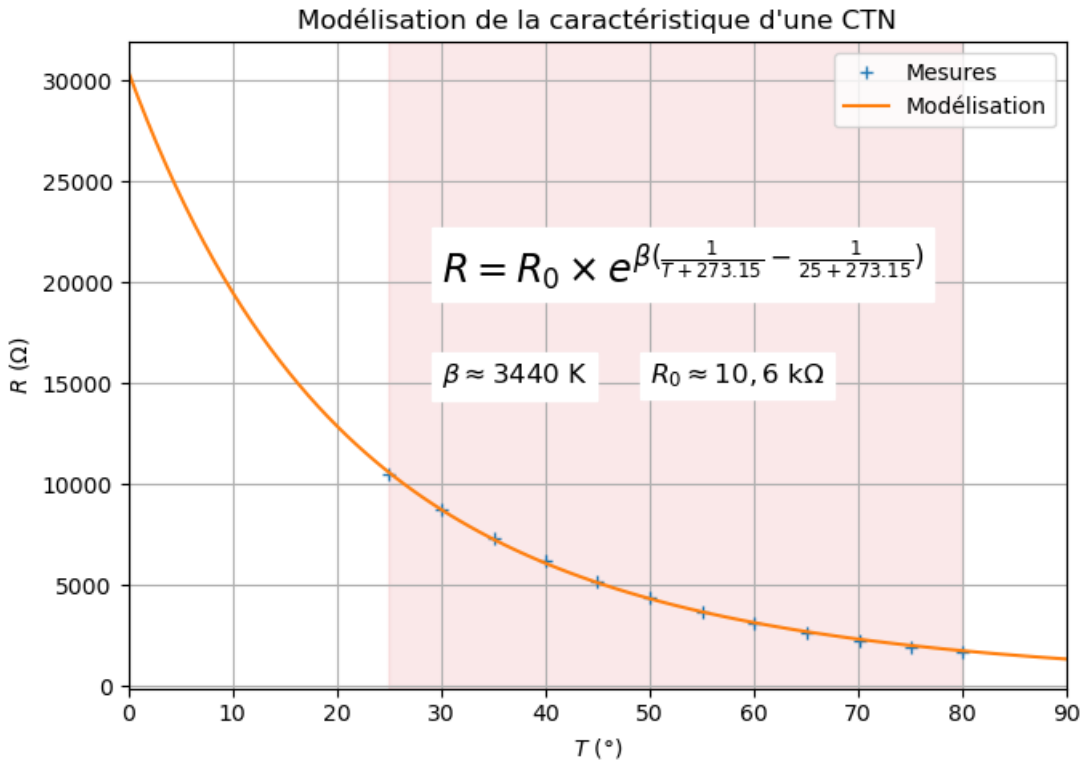


FIG. 10 – Modélisation de la caractéristique de la CTN de 25°C à 80°C

Résultat de la modélisation de $T_0 = 25$°C à 80°C :

$$R_0 \approx 10,6 \text{ kΩ} \qquad \text{et} \qquad \beta \approx 3440 \text{ K}$$

Inversement, le calcul de la température (en K) s'obtient à partir de la relation suivante :

$$\boxed{\frac{1}{T} = \frac{1}{\beta} \times \ln(\frac{R}{R_0}) + \frac{1}{T_0}}$$

5.3.4 Programme 1 : mesure de la résistance de la CTN

La CTN est connectée à la masse (montage 1). L'entrée analogique A0 mesure la tension du capteur.

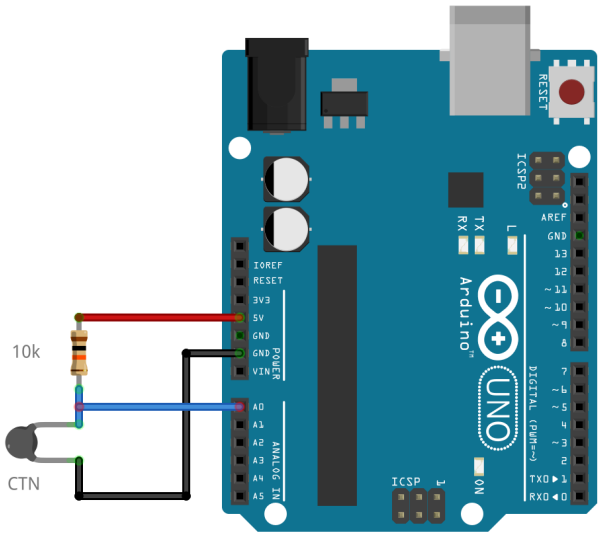


FIG. 11 – Mesure de la résistance d'une CTN sur l'entrée analogique A0

```
/*
*  Mesure de la résistance d'une CTN
*/

// Déclaration des variables pour la mesure de la résistance
float Vcc = 5.0;     // Tension d'alimentation
float R1 = 10000;    // Résistance du pont diviseur de tension
float U;             // Tension de la CTN en volt
float R_mes;         // Résistance mesurée de le CTN en Ohm



void setup() {
   Serial.begin(9600);  // Paramétrage du port série
}

void loop() {
   U = analogRead(A0)*Vcc/1023;    // Lecture tension en V
   R_mes = R1*U/(Vcc-U);           // Calcul de la résistance
   Serial.print("R = ");           // Affichage
   Serial.print(R_mes);            // Affichage
   Serial.println(" Ohm");          // Affichage + sout de ligne
   delay(1000);                    // Temporisation de 1s
}
```

FIG. 12 – Résultats dans le moniteur série d'Arduino IDE

5.3.5 Programme 2 : thermomètre numérique (application)

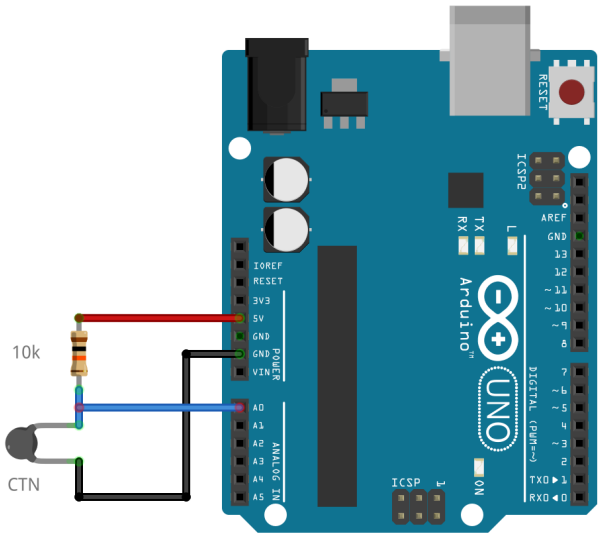


FIG. 13 – Mesure de la température donnée par une CTN sur l'entrée analogique A0

Rappel de relation simplifiée de Steinhart-Hart :

$$\frac{1}{T} = \frac{1}{\beta} \times \ln(\frac{R}{R_0}) + \frac{1}{T_0}$$

avec pour notre CTN :

$$R_0 \approx 10,5\ \text{k}\Omega \qquad \text{et} \qquad \beta \approx 3290\ \text{K}$$

```
/*
* Mesure de la résistance d'une CTN et
* Calcul de la température à partir de la relation simplifiée de Steinhart-Hart
*/
```

(suite sur la page suivante)

(suite de la page précédente)

```
// Déclaration des variables pour la mesure de la résistance
float Vcc = 5.0;     // Tension d'alimentation
float R1 = 10000;    // Résistance du pont diviseur de tension
float U;             // Tension CTN en volt
float R_mes;         // Résistance CTN en ohm

// Déclaration des variables pour la calcul de la température
float T0 = 25;       // Température de référence (25 °C)
float R0 = 10600;    // Resistance R0 à T0 = 25 °C
float beta = 3440;   // Coefficient de temperature en K
float inverse;       // Inverse de la temperature en K-1
float T_K;           // Temperature en K
float T;             // Temperature en °C


void setup() {
   Serial.begin(9600);  // Paramétrage du port série
}

void loop() {
   U = analogRead(A0)*5.0/1023;                          // Lecture de la tension en V
   R_mes = R1*U/(Vcc-U);                                 // Calcul de la résistance mesurée
   Serial.print("R = ");                                 // Affichage
   Serial.print(R_mes);                                  // Affichage
   Serial.println("Ohm");                                // Affichage + retour à la ligne

   inverse = 1/beta * log(R_mes/R0) + 1/(T0+273.15); // Calcul de l'inverse de la␣
↪temperature
   T_K = 1/inverse;                                      // Calcul de la température en␣
↪Kelvin
   T = T_K - 273.15;                                     // Caclul de la température en °C
   Serial.print("T = ");                                 // Affichage
   Serial.print(T);                                      // Affichage
   Serial.println("°C");                                 // Affichage + retour à la ligne
   delay(1000);                                          // Pause
}
```

FIG. 14 – Résultats dans le moniteur série d'Arduino IDE

Chapitre 6

Émission d'un son (seconde générale)

Programme de seconde générale 2019 - Enseignement commun

Utiliser un dispositif comportant un microcontrôleur pour produire un signal sonore.

6.1 Introduction

6.1.1 Principe

Certains microcontrôleurs (Arduino et Micro :bit) ne possédant pas de sortie analogique (CNA) pour générer des tensions sinusoïdales, une méthode simple pour produire un son est de générer une tension carrée (entre 0 et Vcc) de fréquence f à l'entrée d'un haut-parleur.

Le son obtenu par cette technique n'est pas pur car il comporte des harmoniques aux fréquences 3f, 5f, 7f, ...

Avertissement

Pour les faibles fréquences, le son devient « métallique » avec la présence importante d'harmoniques !

6.1.2 Applications

— Hauteur d'un son (relation entre fréquences et notes).
— Générer une mélodie à partir de plusieurs notes.

6.2 Générer un son à partir d'un signal carré

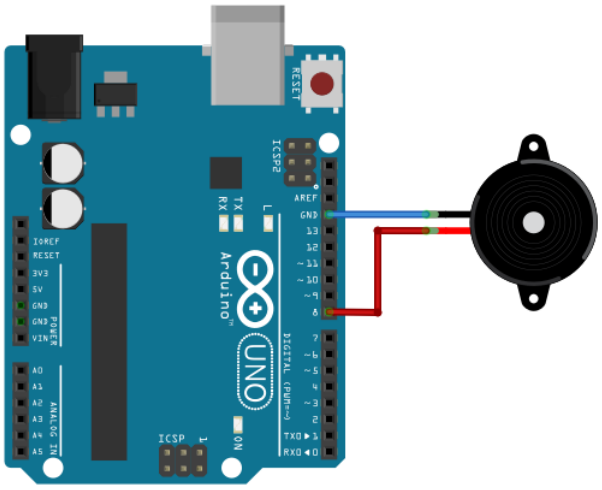


FIG. 1 – Branchement d'un haut-parleur sur la broche 8

```
#define brocheHP 8

float frequence=440;
float periode=1/frequence;

void setup(){
    pinMode(brocheHP, OUTPUT);
}

void loop(){
  digitalWrite(brocheHP,HIGH);
  delayMicroseconds(1000000*periode/2.0);
  digitalWrite(brocheHP,LOW);
  delayMicroseconds(1000000*periode/2.0);
}
```

6.3 Générer un son à partir de la fonction `tone()`

```
/*
Exemple Arduino
*/
#include "pitches.h"

// notes in the melody:
int melody[] = {NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {4, 8, 8, 4, 4, 4, 4, 4 };

void setup() {
for (int thisNote = 0; thisNote < 8; thisNote++) {
   int noteDuration = 1000/noteDurations[thisNote];
```

(suite sur la page suivante)

(suite de la page précédente)

```
   tone(8, melody[thisNote],noteDuration);
   int pauseBetweenNotes = noteDuration * 1.30;
   delay(pauseBetweenNotes);
   noTone(8);
}
}

void loop() {
// Pas de boucle ici !
}
```

Chapitre 7

Mesurer la célérité d'un son (première générale)

Programme de première générale 2019 - Enseignement de spécialité.

Exploiter la relation entre la durée de propagation, la distance parcourue par une perturbation et la célérité, notamment pour localiser une source d'onde. Déterminer, par exemple à l'aide d'un microcontrôleur ou d'un smartphone, une distance ou la célérité d'une onde.

7.1 Présentation du module HC-SR04

Les modules du type HC-SR04 sont des émetteurs-récepteurs ultrasonores fonctionnant par réflexion. Ils sont utilisés généralement dans des applications comme télémètre (< 5 m).

FIG. 1 – Module HC-SR04

7.1.1 Fonctionnement

— Le module est alimenté entre `GND` et `Vcc` (généralement 5 V ou 3,3 V sur certains modules).
— Le déclenchement d'une mesure (émission d'une salve) se fait par une brève impulsion (> 10 µs) sur l'entrée `trig`.
— La durée que prend l'onde pour aller de l'émetteur au récepteur est celle de l'impulsion renvoyée sur la sortie `echo`.

7.1.2 Modification du module

Tel qu'il est vendu, ce module n'a peu d'intérêt en sciences physiques car les signaux électriques sur l'émetteur et le récepteur ne sont pas accessibles. Il est possible de résoudre ce problème en y soudant des **connecteurs supplémen-**

taires (voir photo ci-dessous) et de **visualiser les signaux correspondants à l'oscilloscope ou avec une interface d'acquisition**.

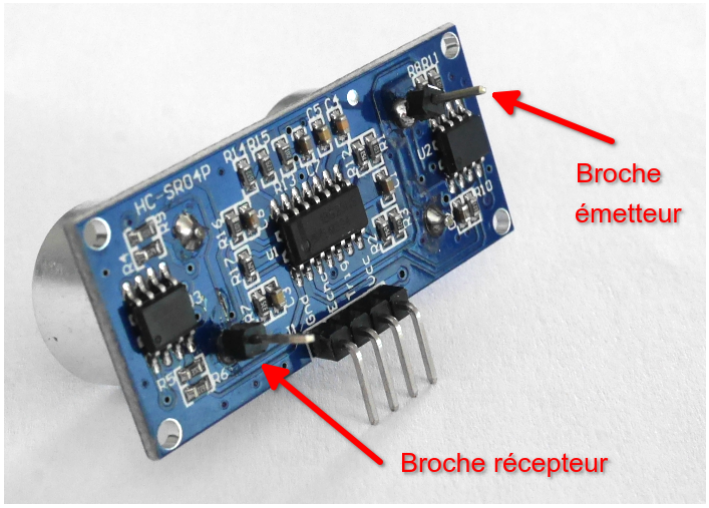


FIG. 2 – Détail du module HC-SR04

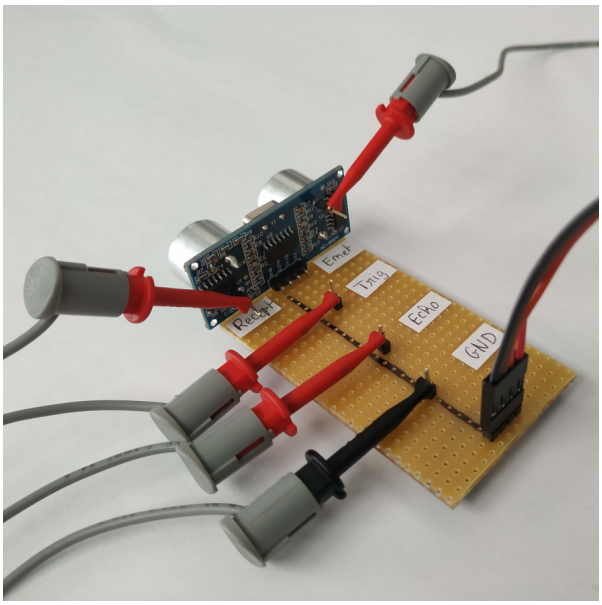


FIG. 3 – Branchement du module HC-SR04 modifié

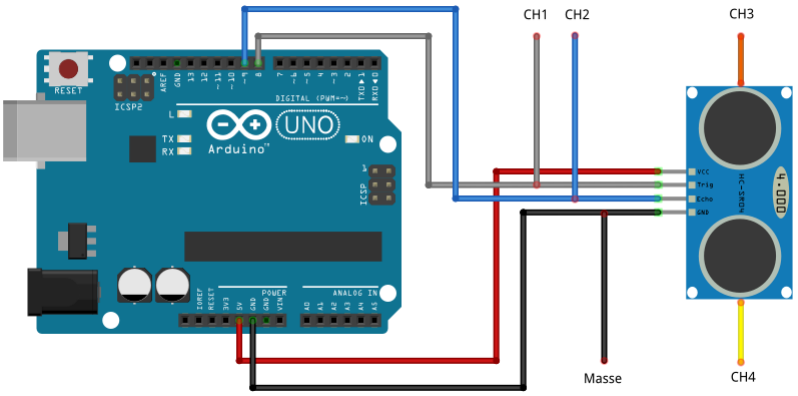


FIG. 4 – Montage avec oscilloscope ou interface d'acquisition

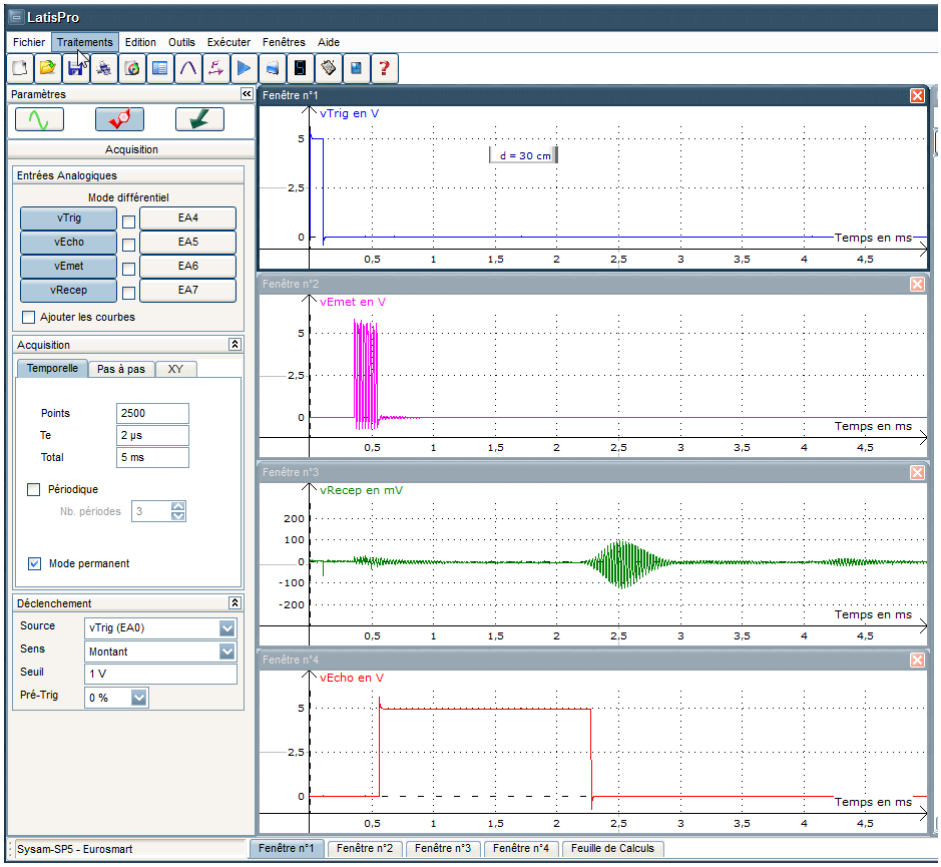


FIG. 5 – Mesures obtenues dans Latis avec Sysam SP5

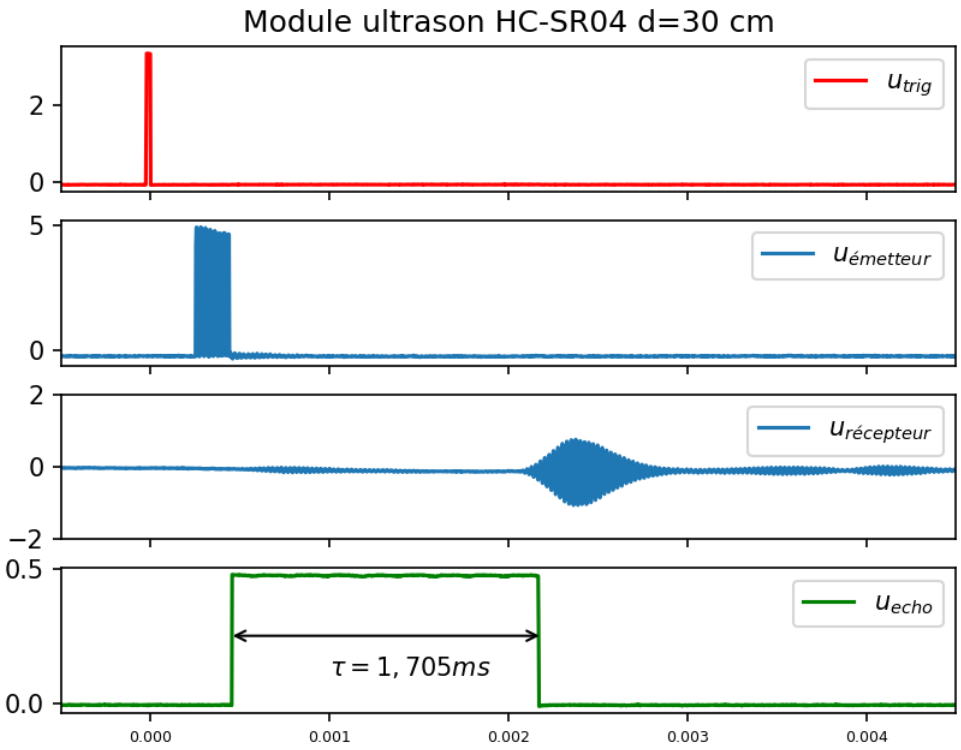


FIG. 6 – Mesures à l'oscilloscope pour une distance de 30 cm

7.2 Mesure de la célérité du son

La manipulation consiste à relever la durée de l'écho sonore à l'aide du microcontrôleur pour différentes distances et déduire la célérité du son.

$$c = \frac{2 \times d}{\Delta t}$$

Algorithme :

```
TRIG <- 0

REPETER :
   TRIG <- Vcc                          # Début impulsion sur Trig
   Attendre 10 µs
   TRIG <- 0                            # Fin impulsion sur Trig
   Dt   <- Durée impulsion sur Echo     # Mesure
   Afficher Dt
   Attendre 1 s
```

Pour plus de précision, il est possible de modifier le programme afin de **réaliser plusieurs mesures** de la durée de l'écho et d'en déduire sa **moyenne**.

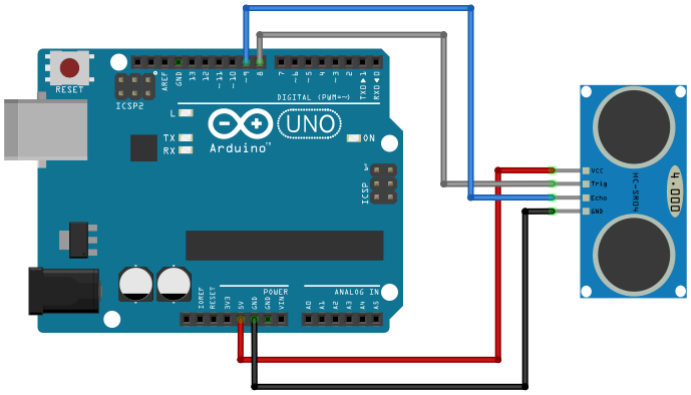


FIG. 7 – Montage célérité son

```
// Mesure de la durée de l'écho sonore

#define pinTrig 8       // Trig sur broche 8
#define pinEcho 9       // Echo sur broche 9

long dureeEcho;         // Durée de l'Echo

void setup() {
  pinMode(pinTrig,OUTPUT);      // Broche Trig en sortie
  digitalWrite(pinEcho,LOW);    // Sortie Trig à l état bas
  pinMode(pinEcho,INPUT);       // Broche Echo en entrée
  Serial.begin(9600);           // Paramétrage du port série
}

void loop() {
  digitalWrite(pinTrig,HIGH);           // Début impulsion de declenchement
  delayMicroseconds(10);                // Attendre 10 microseconde
  digitalWrite(pinTrig,LOW);            // Fin impulsion (Etat bas)
  dureeEcho = pulseIn(pinEcho,HIGH);    // Mesure de la durée de l'impulsion sur Echo
  Serial.print("Durée (µs) = ");         // Affichage sur port série
  Serial.println(dureeEcho);
  delay(1000);                          // Attendre 1s
}
```

7.3 Application : réalisation d'un télémètre

Connaissant la célérité du son, la distance par rapport à un obstacle est calculée par le microcontrôleur à l'aide de la relation suivante :

$$d = \frac{c \times \Delta t}{2}$$

Il suffit d'ajouter le calcul de la distance juste après la mesure de la durée.

```
// Mesure de la durée d'une distance

#define pinTrig 8       // Trig sur broche 8
#define pinEcho 9       // Echo sur broche 9
```

(suite sur la page suivante)

(suite de la page précédente)

```
long dureeEcho;          // Durée de l'Echo
float distance;          // Distance en module et réflecteur
float vitesse = 345 ;    // Vitesse obtenue

void setup() {
pinMode(pinTrig,OUTPUT);       // Broche Trig en sortie
digitalWrite(pinEcho,LOW);     // Sortie Trig à l état bas
pinMode(pinEcho,INPUT);        // Broche Echo en entrée
Serial.begin(9600);            // Paramétrage du port série
}

void loop() {
digitalWrite(pinTrig,HIGH);             // Début impulsion de declenchement
delayMicroseconds(10);                  // Attendre 10 microseconde
digitalWrite(pinTrig,LOW);              // Fin impulsion (Etat bas)
dureeEcho = pulseIn(pinEcho,HIGH);      // Mesure de la durée de l'impulsion sur Echo
distance = (vitesse*dureeEcho*1E-6)/2; // Calcul de la distance
Serial.print("Duree (us) = ");          // Affichage sur port série
Serial.println(dureeEcho);              //
Serial.print("Distance (m) = ");        // Affichage sur port série
Serial.println(distance);               //
delay(1000);                            // Attendre 1s
}
```

En autonome avec un écran LCD :

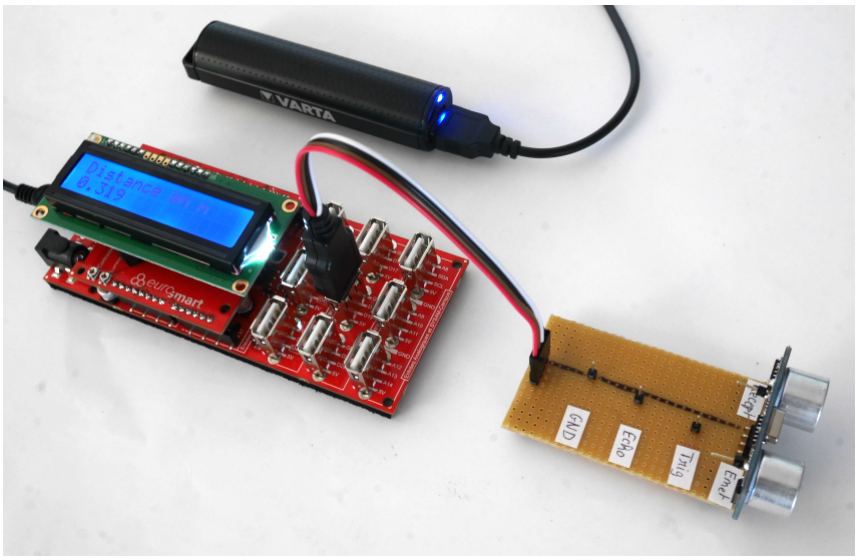


FIG. 8 – Télémètre sur Educaduino-Lab LCD

```
//Application : télémètre sur écran LCD 2x16

#include <LiquidCrystal.h>        // Importation de la librairie LiquidCrystal
#define pinTrig 8       // Trig sur broche 8
#define pinEcho 9       // Echo sur broche 9

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // Brochage de l'afficheur

float distance;          // Distance en module et réflecteur
```

(suite sur la page suivante)

(suite de la page précédente)

```
long dureeEcho;         // Durée mesurée
float vitesse = 345 ;   // Vitesse obtenue


void setup() {
 pinMode(pinTrig,OUTPUT);      // Broche Trig en sortie
 digitalWrite(pinEcho,LOW);    // Sortie Trig à l état bas
 pinMode(pinEcho,INPUT);       // Broche Echo en entrée
 lcd.begin(16, 2);               // fixe le nombre de colonnes et de lignes de l␣
 ↪afficheur
}

void loop() {
 digitalWrite(pinTrig,HIGH);                    // Début impulsion de déclenchement
 delayMicroseconds(10);                         // Attendre 10 microseconde
 digitalWrite(pinTrig,LOW);                     // Fin impulsion (Etat bas)
 dureeEcho = pulseIn(pinEcho,HIGH);             // Mesure de la durée de l'impulsion sur␣
 ↪Echo
 distance = (vitesse*dureeEcho*1E-6)/2;    // Calcul de la distance
 lcd.setCursor(0,0);                            // place le curseur au début de la ligne␣
 ↪0
 lcd.print("Distance en m");                   // Affiche la légende
 lcd.setCursor(0,1);                            // place le curseur au début de la ligne␣
 ↪1
 lcd.print(distance);                           // Affiche la valeur de la distance
 delay(1000);                                   // Attendre 1s
}
```

Chapitre 8

Mesurer une pression - Loi de Mariotte (première générale)

Programme de première générale 2019 - Enseignement de spécialité.

Tester la loi de Mariotte, par exemple en utilisant un dispositif comportant un microcontrôleur.

8.1 Introduction

8.1.1 Principe

La manipulation consiste à vérifier la loi de Mariotte (à température et à quantité de matière constantes) :

$$\boxed{P \times V = \text{Constante}}$$

Cette expérimentation nécessite un **capteur de pression absolue**.

8.1.2 Capteurs analogiques de pression absolue

La plupart des capteurs en sciences physiques sont du type analogique. Ils fournissent une tension qui est l'image de la pression. La relation est généralement linéaire.

Exemples :

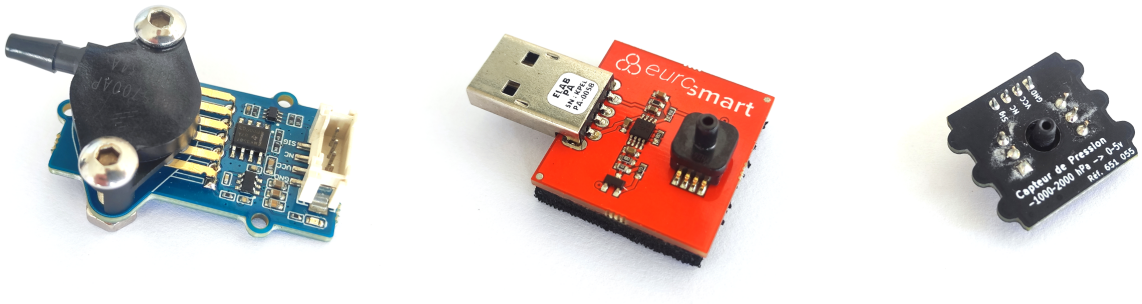


FIG. 1 – Grove MPX5700AP / Educaduino MPXHZ6400A / Plug'Uino 201DFP

Module	Grove	Educaduino	Plug'Uino
Capteur	MPX5700AP	MPXHZ6400A	XGZP201DB1F
Type	Piézorésistif	Piézorésistif	Piézorésistif
Tension	3,3V ou 5V	5V	5V
Plage	150 à 7000 hPa	200 à 4000 hPa	0 à 3000 hPa
Précision	±2,5%	±1,5%	±1%
Compensation température	Oui	Oui	NC
Pré-calibrage	Oui	Oui	NC

La précision prend en compte de la variation par rapport à la valeur nominale, de la linéarité, de la température, ...

Note

Un étalonnage de ces capteurs à la pression atmosphérique réelle améliore significativement la précision.

8.1.3 Capteurs numérique de pression absolue

Les capteurs de pression numérique fournissent directement au microcontrôleur la valeur de la pression sous forme numérique. Ils transmettent généralement les données par bus I2C.

Exemple :

FIG. 2 – Adrafruit MPRLS0025PA

Module	Adafruit MPRLS
Capteur	MPRLS0025PA
Type	Piézorésistif
Tension	3,3V ou 5V
Plage	0 à 1700 hPa
bus	I2C (24 bit)
Précision	±1, 5%
Compensation température	Oui
Pré-calibrage	Oui

8.1.4 Capteurs numériques pression atmosphérique pour étalonnage

Il existe des **capteurs numériques de précision** pour la mesure de la pression atmosphérique. Ils peuvent être utilisés pour l'étalonnage des capteurs précédents afin d'éliminer le décalage de pression.

FIG. 3 – Grove BMP280 (baromètre/altimètre) Plug'Uino (image : Sciencéthic)

Module	Grove	Grove	Grove
Capteur	HP206F	BMP280	DPS310
Type	MEMS	MEMS Résistif	MEMS Capacitif
Tension	3,3V/5V	3,3V/5V	3,3V/5V
Plage	300 à 1200 hPa	300 à 1100 hPa	300 à 1200 hPa
Bus	I2C (24 bit)	I2C/SPI (20 bit)	I2C/SPI (24 bit)
Précision absolu	±2 hPa	±1 hPa	±1 hPa
Précision relative	±0, 5 hPa	±0, 12 hPa	±0, 06 hPa

Les librairies de ces capteurs sont disponibles dans Arduino IDE. Les programmes d'exemple fournis permettent une mise en oeuvre rapide.

8.2 Mesure d'une pression absolue de référence pour calibrage

Une mesure précise de la pression atmosphérique est nécessaire pour réduire le décalage (offset) des capteurs de pression absolue.

Les **capteurs de pression atmosphérique** sont choisis pour leur **grande précision**.

En classe, une seule version de ce montage, sur un afficheur LCD par exemple, peut-être utiliser pour calibrer tout les capteurs à partir cette mesure de référence.

8.2.1 Capteur atmosphérique Grove BMP280 (I2C)

Ce capteur mesure une pression absolue de 300 à 1100 hPa avec une précision de ±1 hPa.

Plus d'information sur wiki.seeedstudio.com/Grove-Barometer_Sensor-BMP280/

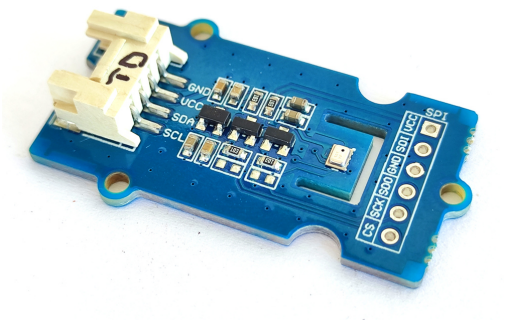


FIG. 4 – Capteur Grove BMP280 (I2C)

Installer la bibliothèque `Barometer Sensor BMP280 par Seeed Studio` directement dans Arduino IDE (connexion Internet nécessaire).

```
/*
 * Mesure pression atmosphérique - Capteur BMP280 I2C (±1hPa)
 * David THERINCOURT - 2025
 */

#include "Seeed_BMP280.h"

BMP280 bmp280;   // Capteur

float pression;  // Pression en Pascal

void setup()
{
    Serial.begin(9600);                          // Initialisation port série
    if(!bmp280.init()){                          // Initialisation BMP280
        Serial.println("BMP280 non trouvé !"); // Affichage
    }
    Serial.println("BMP280 trouvé !");         // Affichage
}

void loop()
{
    pression = bmp280.getPressure();             // Mesure pression en Pa
    Serial.print(pression/100.0);                // Affichage en hPa
    Serial.println(" hPa");                      // ...
    delay(1000);                                 // Temporisation
}
```

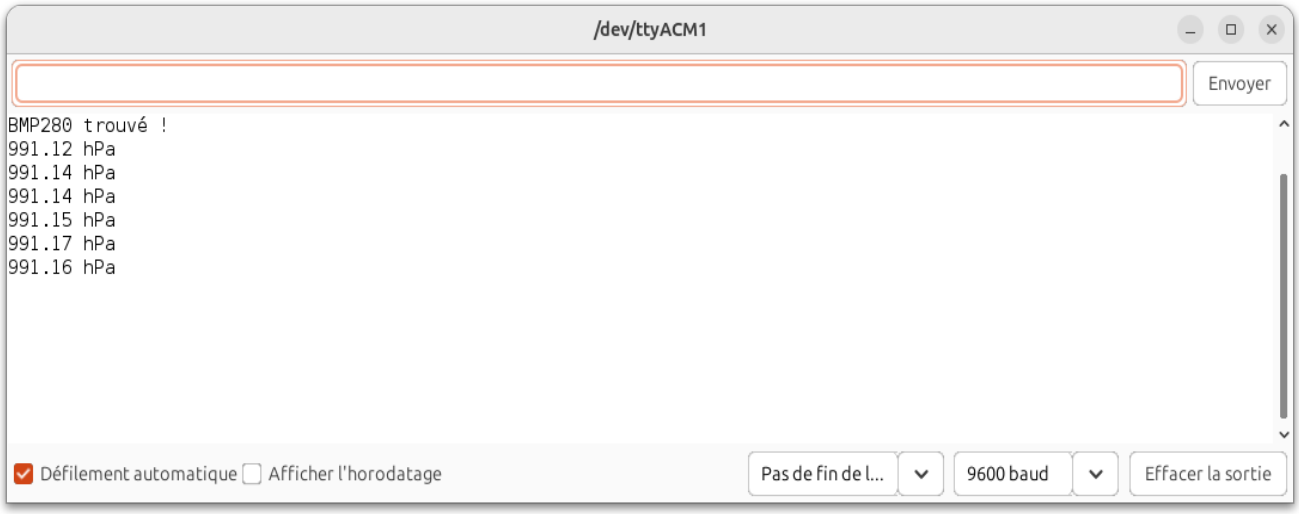


FIG. 5 – Résultats dans le moniteur série d'Arduino IDE

Avec le capteur BMP280, la pression atmosphérique de référence sera :

$$P_{ATM_{REF}} = 991 \text{ hPa}$$

8.2.2 Capteur atmosphérique Grove DPS310

Ce capteur mesure une pression absolue de 300 à 1100 hPa avec une précision de ±1 hPa.

Plus d'information sur wiki.seeedstudio.com/Grove-High-Precision-Barometric-Pressure-Sensor-DPS310/

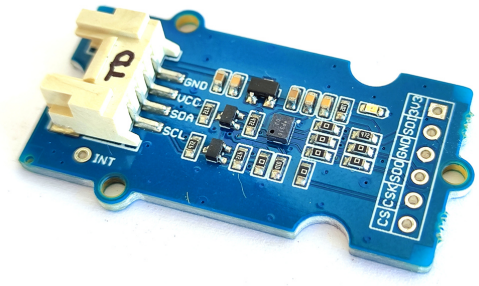


FIG. 6 – Capteur Grove DSP310 (I2C)

La bibliothèque est téléchargeable dans une archive ZIP sur le site github.com/Seeed-Studio/Seeed_Arduino_DPS310 .

Puis dans Arduino IDE, ajouter la bibliothèque à partir du menu `Croquis > Inclure bibliothèque > Ajouter la bibliothèque .ZIP ...`.

```
/*
 * Mesure pression atmosphérique - Capteur DPS310 I2C (±1hPa)
 * David THERINCOURT - 2025
 */

#include <Dps310.h>

Dps310 Dps310PressureSensor = Dps310();
```

(suite sur la page suivante)

(suite de la page précédente)

```
float pression;            // Pression en Pascal
uint8_t oversampling = 7;  // 0 à 7 - Suréchantillonnage pour meilleure précision

void setup(){
    Serial.begin(9600);                 // Initialisation port série
    Dps310PressureSensor.begin(Wire);   // Initialisation DPS310 sur port I2C
    Serial.println("DPS310 trouvé !");  // Affichage
}



void loop(){
    Dps310PressureSensor.measurePressureOnce(pression, oversampling); // Mesure en Pa
    Serial.print(pression/100);                                       // Affichage en␣
 ↪hPa
    Serial.println(" hPa");                                           // ..
    delay(1000);                                                      // Temporisation
}
```

FIG. 7 – Résultats dans le moniteur série d'Arduino IDE

Avec le capteur DPS310, la pression atmosphérique de référence sera :

$$P_{ATM_{REF}} = 991 \text{ hPa}$$

8.2.3 Capteur atmosphérique Grove HP206F

Ce capteur mesure une pression absolue de 300 à 1200 hPa avec une précision de ±2 hPa.

Plus d'information sur wiki.seeedstudio.com/Grove-Barometer-High-Accuracy/

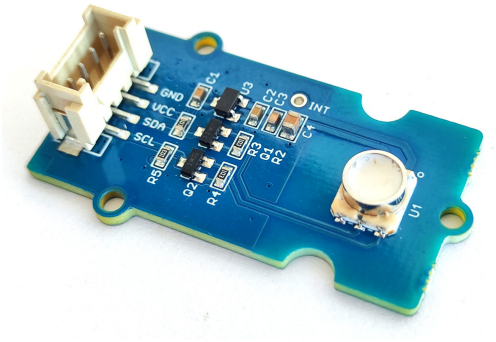


FIG. 8 – Capteur Grove HP206F (I2C)

Installer la bibliothèque `Grove barometer HP20x par Seeed Studio` directement dans Arduino IDE (connexion Internet nécessaire).

```
/*
 * Mesure pression atmosphérique - Capteur HP206F I2C (±1hPa)
 * David THERINCOURT - 2025
 */

#include <HP20x_dev.h>

long pression;                          // Pression en Pascal

void setup()
{
    Serial.begin(9600);                 // Initialisation port série
    HP20x.begin();                      // Initialisation capteur HP206F
    Serial.println("HP206F trouvé !"); // Affichage
}


void loop()
{
    pression = HP20x.ReadPressure();    // Mesure pression en Pascal
    Serial.print(pression/100.0);       // Afficahge en hPa
    Serial.println(" hPa");             // ...
    delay(1000);                        // Temporisation
}
```

FIG. 9 – Résultats dans le moniteur série d'Arduino IDE

Avec le capteur HP206F, la pression atmosphérique de référence sera :

$$P_{ATM_{REF}} = 990 \text{ hPa}$$

8.3 Loi de Mariotte avec capteur Grove

8.3.1 Capteur de pression absolue MPX5700AP

Le MPX5700AP est un **capteur piézorésistif de pression absolue** (150 à 7000 hPa).

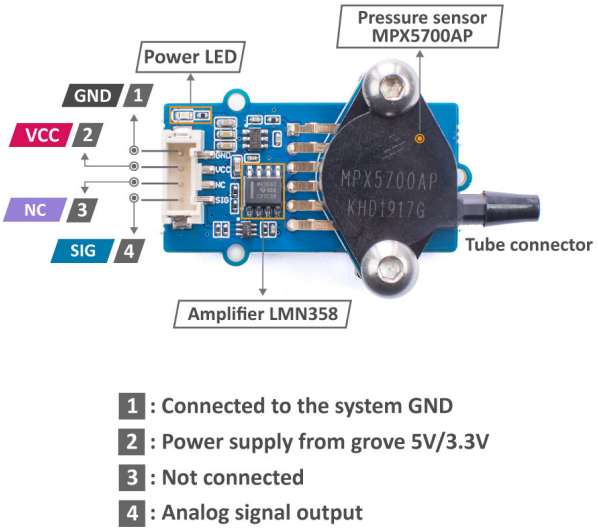


FIG. 10 – Capteur MPX5700AP Grove (image : seeedstudio)

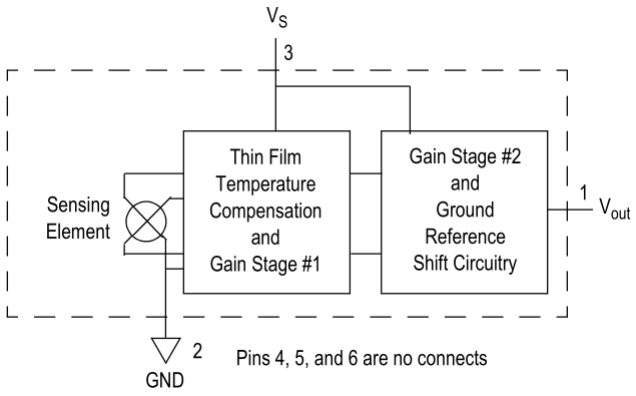


FIG. 11 – Schéma électrique (source : Freescale Semiconductor, Inc.)

La sortie du capteur est une tension analogique. La relation avec la pression suit une fonction affine telle que :

Pression	Tension de sortie
0 hPa	0,2 V
7000 hPa	4,7 V

La documentation technique donne une **précision** est de $\pm 2,5$ % de la plage totale de variation. Cette précision tient compte de la variation de la pression par rapport à la valeur nominale, de la linéarité, de l'hystérésis de pression, de l'hystérésis thermique et de la déviation de la sortie par rapport à la température.

Avertissement

Pour une meilleure précision, il convient d'étalonner le capteur à partir d'une mesure précise de la pression atmosphérique.

8.3.2 Calcul de la pression

La mesure de pression (en hPa) est donnée par la relation pour une alimentation de 5 V :

$$\boxed{P = \frac{P_{max}}{V_{max} - V_{min}} \times (v_{out} - V_{min})}$$

Soit :

$$P = \frac{7000}{4,7 - 0,2} \times (v_{out} - 0,2) \quad \Longrightarrow \quad \boxed{P = \frac{7000}{4,5} \times (v_{out} - 0,2)} \quad \text{(hPa)}$$

8.3.3 Mesure simple de la pression absolue

Le programme ci-dessous calcule et affiche la pression du capteur à partir de sa tension de sortie. La valeur de la tension de sortie du capteur est donnée à titre informatif.

```
/*
 * Mesure d'une pression absolue
 * Capteur Grove MPX5700AP (150 hPa à 7000 hPa) branché sur la broche A0
 * David THERINCOURT - 2025
 */

float VCC = 5.00;                          // Tension d'alimentation de l'Arduino
float tension;                             // Tension du capteur
float pression;                            // Pression calculée

void setup() {
   Serial.begin(9600);                      // Initialisation du port série
   Serial.println("-----");                 // Affichage initial
}

void loop() {
   tension = analogRead(A0)*VCC/1023.0 ;   // Lecture de la tension sur A0 en volt
   pression = 7000/4.5*(tension-0.2) ;     // Calcul de la pression en hPa
   Serial.print(tension);                   // Affichage
   Serial.print("V \t");                    // ...
   Serial.print(pression);                  // ...
   Serial.println(" hPa");                  // ...
   delay(1000);                             // Temporisation
}
```

FIG. 12 – Exemple de mesures obtenues dans le moniteur série d'Arduino IDE

8.3.4 Vérification de la loi de Mariotte

Le programme ci-dessous saisie au clavier le volume V puis affiche la mesure de la pression P correspondante. Ces mesures sont affichées au format CSV pour permettre une exportation des données par copier-coller dans un grapheur (ex. Regressi, www.desmos.com, ...)

```
/*
 * Mesure d'une pression absolue - Vérification de la loi de Mariotte
 * Capteur Grove MPX5700AP (150 hPa à 7000 hPa) branché sur la broche A0
 * David THERINCOURT - 2025
 */
```

(suite sur la page suivante)

(suite de la page précédente)

```
float VCC = 5.0;     // Tension d'alimentation
float volume;        // Volume de la seringue
float tension;       // Tension sortie capteur
float pression;      // Pression mesurée

void setup() {
   Serial.begin(9600);                                  // Initialisation communication
 ↪série
   Serial.println("Saisir le volume au clavier en mL"); // Affichage
   Serial.println("---- CSV ----");                     // ...
   Serial.println("Vs ; P");                            // ...
   Serial.println("mL ; hPa");                          // ...
}

void loop() {
   while (Serial.available()==0){}          // Attente message (Cocher "Pas de fin de
 ↪ligne")
   volume = Serial.parseFloat();            // Extraction de la valeur du volume
   tension = analogRead(A0)*VCC/1023.0 ;    // Lecture de la tension sur A0 en volt
   pression = 7000/4.5*(tension-0.2) ;      // Calcul de la pression en hPa
   Serial.print(volume,0);                  // Affichage de mesures en CSV
   Serial.print(" ; ");                     // ...
   Serial.println(pression,0);              // ...
}
```

Avertissement

Pour commencer une nouvelle série de mesure, il faut relancer le programme en appuyant sur le bouton RESET du microcontrôleur.

FIG. 13 – Exemple de mesures obtenus dans le moniteur série d'Arduino IDE

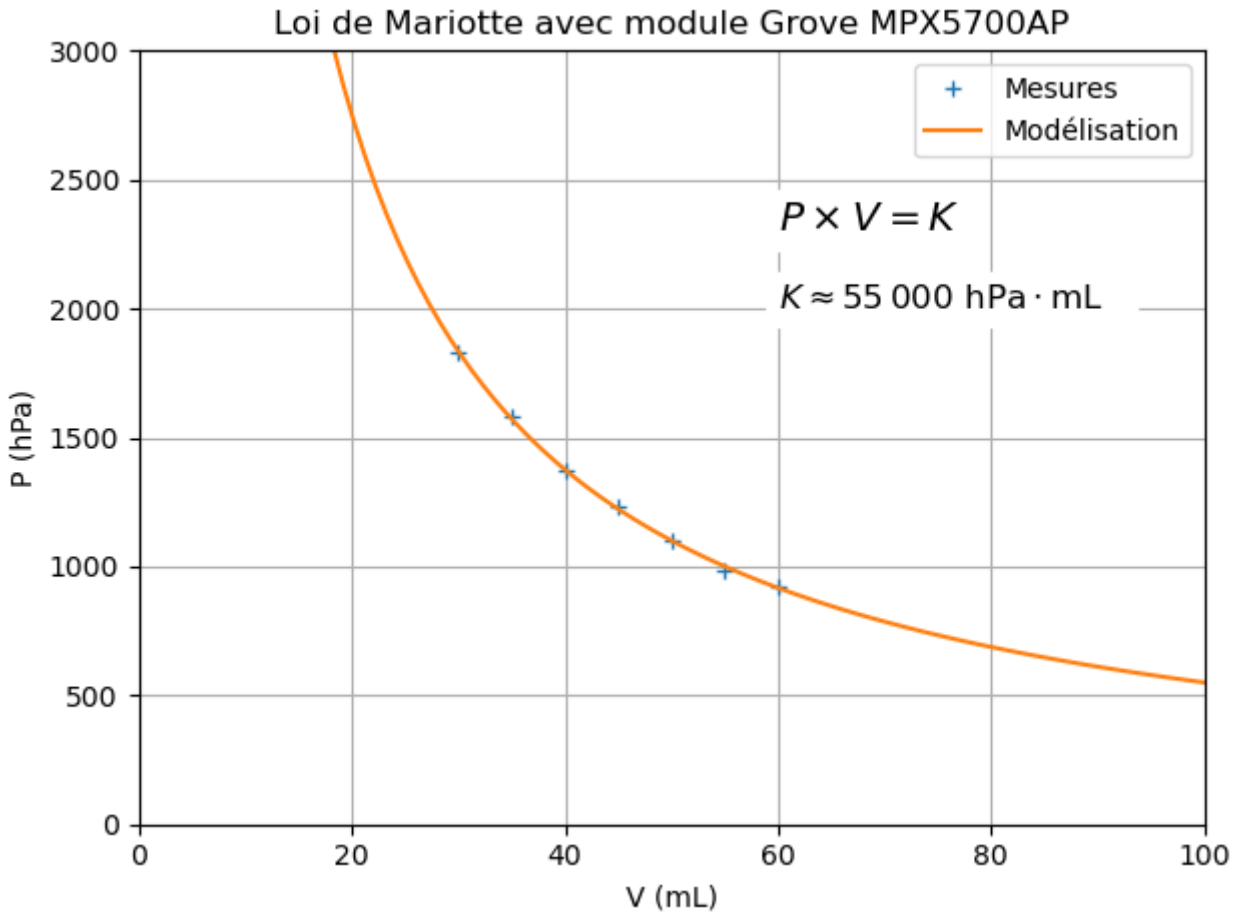
Le volume final est donné par la relation :

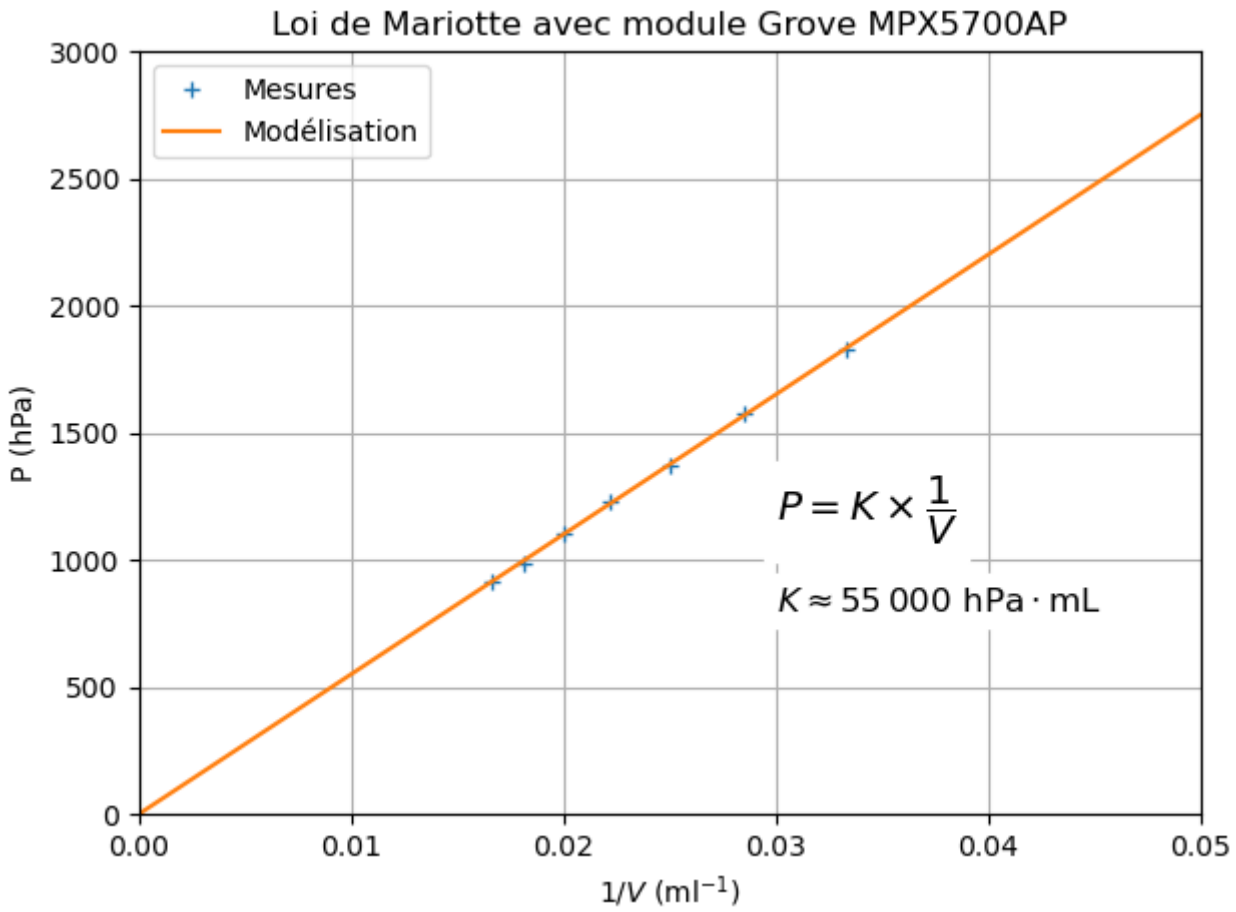
$$V = V_s + V_0$$

— V_s est la volume de la seringue ;
— V_0 est le volume additionnel contenu dans le tube et le capteur.

8.3.5 Résultats

Le volume V_0 est négligé.





8.4 Loi de Mariotte avec capteur Educaduino Lab

8.4.1 Capteur MPXHZ6400A

Le MPXHZ6400A est un capteur piézorésistif de pression absolue (200 hPa à 4000 hPa).

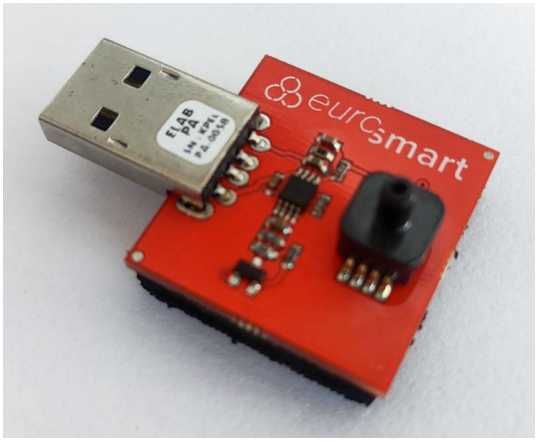


FIG. 14 – Capteur pression absolue Educaduino LAB

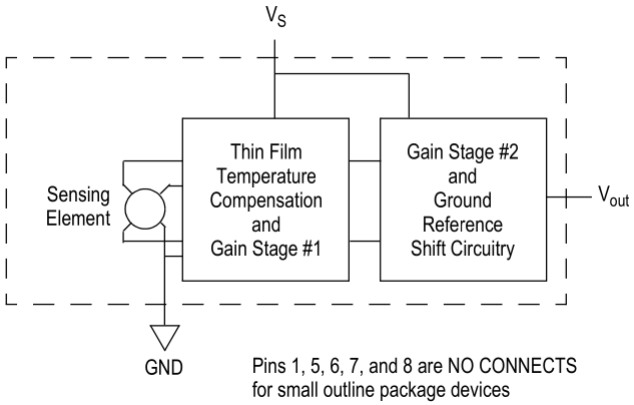


FIG. 15 – Schéma électrique (source : Freescale Semiconductor, Inc.)

L'électronique interne de ce module modifie les caractéristiques de la tension de sortie du capteur MPXHZ6400A comme le montre le tableau suivant :

Pression	Tension de sortie
200 hPa	0 V
4000 hPa	5,0 V (VCC)

La documentation technique donne une précision du capteur de $\pm 1,5\%$ de la plage totale de variation. Cette précision tient compte de la variation de la pression par rapport à la valeur nominale, de la linéarité, de l'hystérésis de pression, de l'hystérésis thermique et de la déviation de la sortie par rapport à la température.

Avertissement

Pour une meilleure précision, il convient d'étalonner le capteur à partir d'une mesure précise de la pression atmosphérique.

8.4.2 Calcul de la pression

La mesure de pression (en hPa) est donnée par la relation suivante :

$$P = \frac{P_{max} - P_{min}}{V_{CC}} \times V_{out} + P_{min}$$

Soit :

$$P = \frac{3800}{V_{CC}} \times V_{out} + 200 \qquad \text{avec} \qquad V_{CC} = 5,0 \text{ V}$$

Avertissement

Le module n'intègre pas de régulateur de tension. En conséquence, une mesure au voltmètre de la tension V_{CC}, sur la broche `5V` ou `IOREF` de l'Arduino, est vivement conseillée !

8.4.3 Mesure simple de la pression absolue

Le programme ci-dessous affiche périodiquement une mesure de la pression absolue en hPa ainsi de la tension de sortie du capteur (broche A9).

```
/*
 * Mesure d'une pression absolue
 * Capteur pression absolue Educaduino Lab (20 kPa à 400 kPa)
 * Réference : MPXHZ6400A
 * David THERINCOURT - 2025
*/

float VCC = 5.00;                          // Tension d'alimentation de l'Arduino
float tension;                             // Tension du capteur
float pression;                            // Pression calculée

void setup() {
   Serial.begin(9600);                     // Initialisation du port série
}

void loop() {
   tension = analogRead(A9)*VCC/1023.0 ;   // Lecture de la tension sur A9 en volt
   pression = 3800/VCC*tension + 200 ;     // Calcul de la pression en hPa
   Serial.print(tension);                  // Début affichage
   Serial.print("V \t");                   // caractère "\t" pour une tabulation
   Serial.print(pression,0);               // ...
   Serial.println(" hPa");                 // ...
   delay(1000);                            // Temporisation
}
```

FIG. 16 – Résultats dans le moniteur série

8.4.4 Vérification de la loi de Mariotte

Le programme ci-dessous saisie manuellement au clavier la valeur du volume V puis mesure la pression absolue correspondante. Les résultats sont affichés au format CSV dans le moniteur série. L'exportation vers une autre application peut se faire par un copier-coller de ces données CSV.

```
/* Mesure d'une pression absolue - Loi de Mariotte
* Capteur pression absolue Educaduino Lab (20 kPa à 400 kPa)
```

(suite sur la page suivante)

(suite de la page précédente)

```
* Réference : MPXHZ6400A
* David THERINCOURT - 2025
*/

float VCC = 5.00;                          // Tension d'alimentation

void setup() {
   Serial.begin(9600);                    // Initialisation du port série
   Serial.println("Saisir à chaque fois le volume au clavier.");
   Serial.println("---- CSV ----");
   Serial.println("V ; P");
   Serial.println("mL ; hPa");
}

void loop() {
   while (Serial.available()==0){}        // Attente d'un message (Cocher "Pas de fin de ligne")
   float V = Serial.parseFloat();         // Conversion en flottant
   float U = analogRead(A9)*VCC/1023.0 ; // Lecture de la tension sur A9 en volt
   float P = 3800/VCC*U + 200 ;           // Calcul de la pression en hPa
   Serial.print(V,0);                     // Affichage
   Serial.print(" ; ");                   // ...
   Serial.println(P,0);                   // ...
}
```

Avertissement

Pour commencer une nouvelle série de mesure, il faut relancer le programme en appuyant sur le bouton RESET du microcontrôleur.

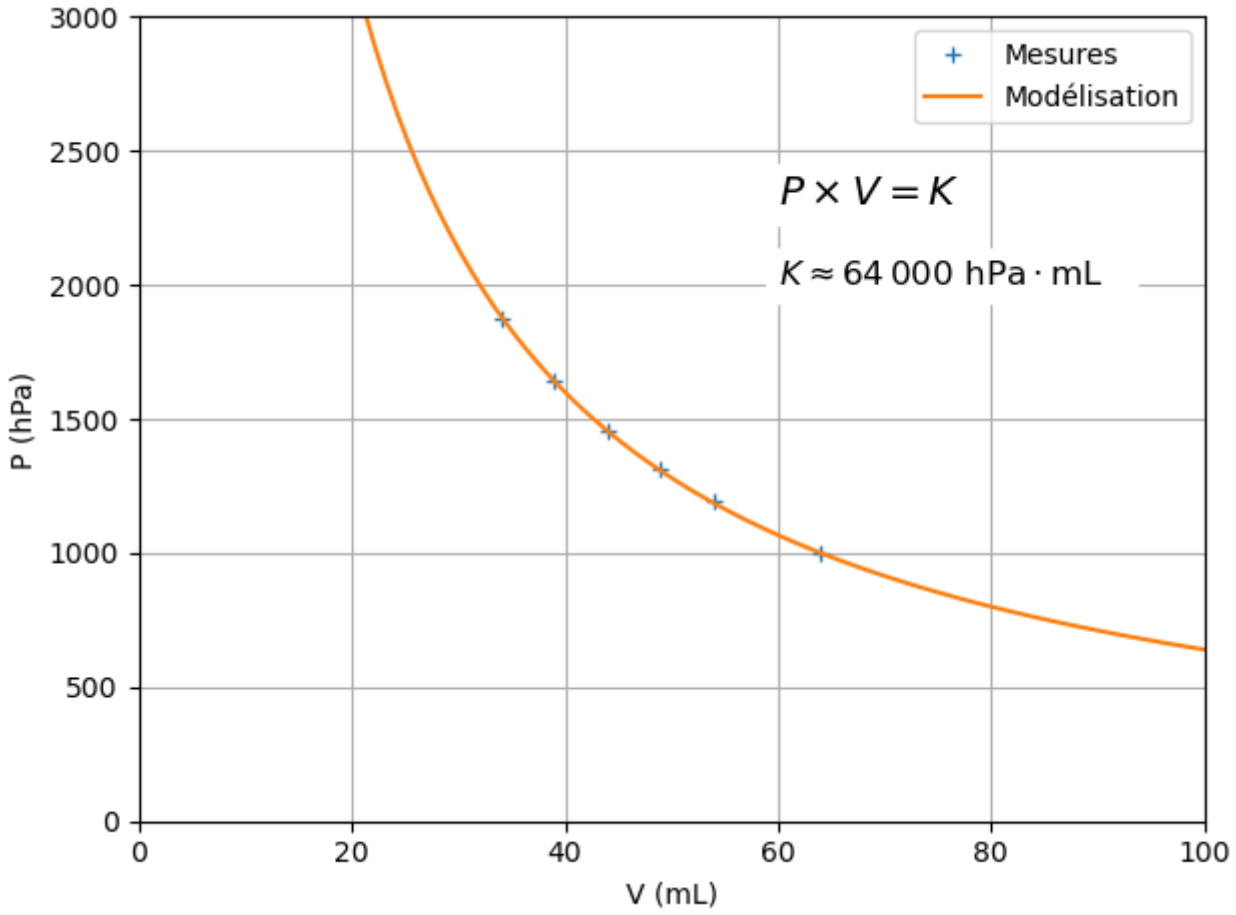
FIG. 17 – Résultats dans le moniteur série

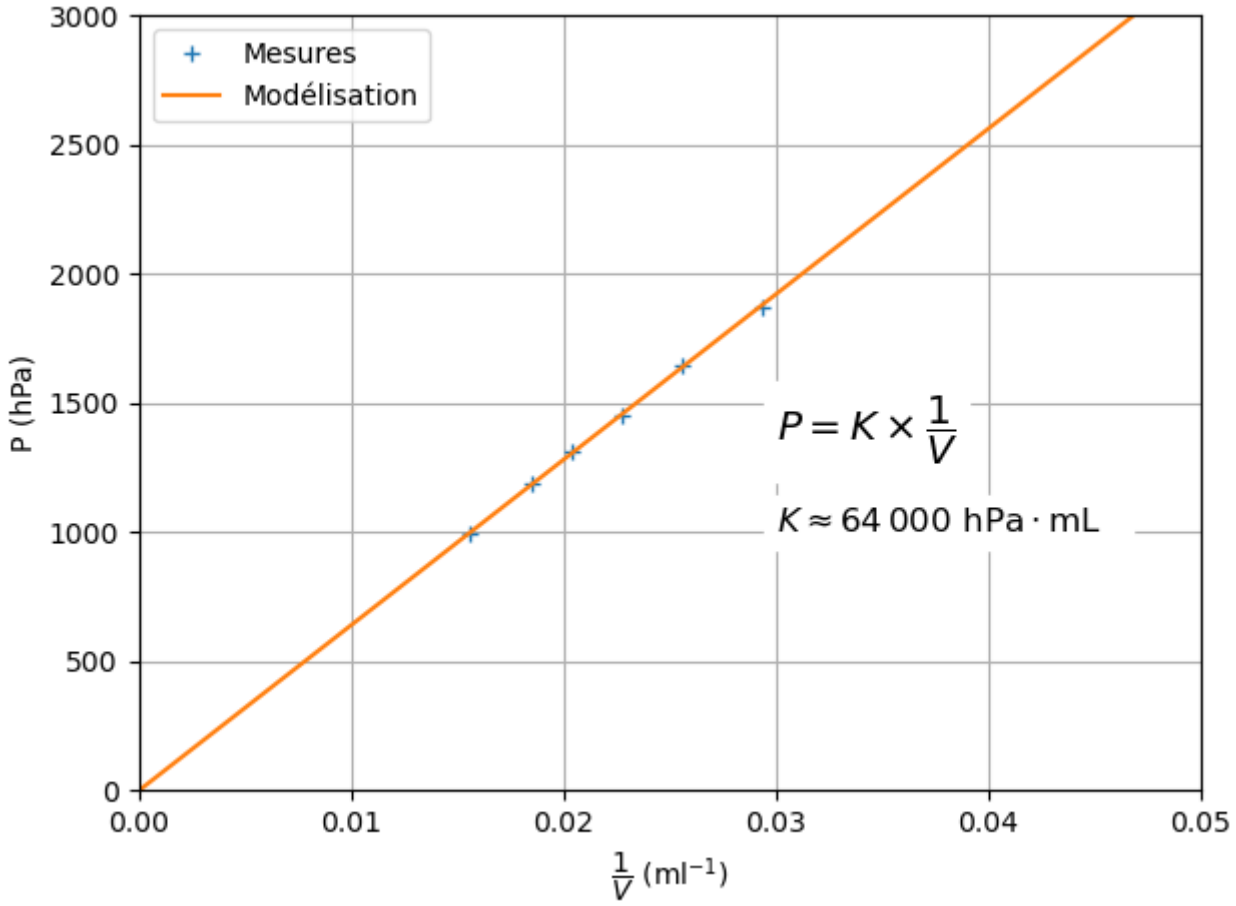
8.4.5 Résultats

Le volume final est donné par la relation :

$$V = V_s + V_0$$

— V_s est la volume de la seringue ;
— V_0 est le volume additionnel contenu dans le tube et le capteur (ici 4 mL).





8.5 Loi de Mariotte avec le capteur Adafruit MPRLS

8.5.1 Capteur MPLRS0025AP

Le circuit MPLRS0025AP est un capteur piézorésistif de pression absolue (0 à 1700 hPa). La mesure de la pression est directement donnée sous forme numérique (24 bit) via le port I2C.

FIG. 18 – Capteur pression absolue Educaduino LAB

La documentation technique donne une précision de $\pm 1,5\%$ de la plage totale de variation.

A noter que ce module fonctionne aussi bien en $3,3$ V ou 5 V.

8.5.2 Mesure simple de la pression absolue

Le programme ci-dessous affiche périodiquement une mesure de la pression absolue en hPa. Une correction du décalage de la pression a été faite à partir d'un capteur atmosphérique de haute précision (ici BMP280).

Il faut installer préalablement la bibliothèque `Adafruit MPRLS Library` dans Arduino IDE.

```
/*
 * Mesure d'une pression absolue
 * Capteur Adafruit MPRLS (0 à 1700 hPa) I2C
 * David THERINCOURT - 2025
 */

#include "Adafruit_MPRLS.h"

Adafruit_MPRLS mpr = Adafruit_MPRLS(-1, -1); // Déclaration capteur MPRLS

float Patm_mpr = 998;                        // Pression atmosphérique MPRLS
float Patm_ref = 991;                        // Pression atmosphérique de référence
float correction = Patm_ref - Patm_mpr;      // Correction à appliquer
float Pmpr;                                  // Pression MPRLS

void setup() {
   Serial.begin(9600);                       // Initialisation port série
      if (! mpr.begin()) {                   // Initialisation MPRLS
      Serial.println("MPRLS non trouvé !");  // ...
      while (1) {                            // ...
         delay(10);                          // ...
      }
   }
   Serial.println("MRPLS trouvé !");         // Affichage
   Serial.print("Correction de ");           // ...
   Serial.print(correction);                 // ...
   Serial.println(" hPa");                   // ...
}


void loop() {
   Pmpr = mpr.readPressure() + correction;   // Mesure de la pression
   Serial.print(Pmpr);                       // Affichage
   Serial.println(" hPa");                   // ...
   delay(1000);                              // Temporisation
}
```

FIG. 19 – Résultats dans le moniteur série

8.5.3 Vérification de la loi de Mariotte

Le programme ci-dessous saisie manuellement au clavier la valeur du volume V puis mesure la pression absolue correspondante. Les résultats sont affichés au format CSV dans le moniteur série. L'exportation vers une autre application peut se faire par un copier-coller de ces données CSV.

```
/*
 * Vérification de la loi de Mariotte
 * Capteur Adafruit MPRLS (0 à 1700 hPa) I2C
 * David THERINCOURT - 2025
 */

#include "Adafruit_MPRLS.h"

Adafruit_MPRLS mpr = Adafruit_MPRLS(-1, -1); // Déclaration capteur MPRLS

float Patm_mpr = 998;                    // Pression atmosphérique MPRLS
float Patm_ref = 991;                    // Pression atmosphérique référence
float correction = Patm_ref - Patm_mpr;  // Correction à appliquer

float volume;
float pression;

void setup() {
   Serial.begin(9600);                     // Initialisation port série
   if (! mpr.begin()) {                    // Initialisation MPRLS
      Serial.println("MPRLS non trouvé !");  // ...
      while (1) {                          // ...
         delay(10);                        // ...
      }
   }
   Serial.println("Capteur MPRLS trouvé."); // Affichage
   Serial.println("---- CSV ----");         // ...
   Serial.println("Vs ; P");               // ...
   Serial.println("mL ; hPa");             // ...
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
void loop() {
   while (Serial.available()==0){}          // Attente de la saisie
   volume = Serial.parseFloat();            // Extraction de la valeur du volume
   pression = mpr.readPressure();           // Mesure de la pression
   Serial.print(volume,0);                  // Affichage
   Serial.print(" ; ");                     // ...
   Serial.println(pression,0);              // ...
}
```

Avertissement

Pour commencer une nouvelle série de mesure, il faut relancer le programme en appuyant sur le bouton RESET du microcontrôleur.

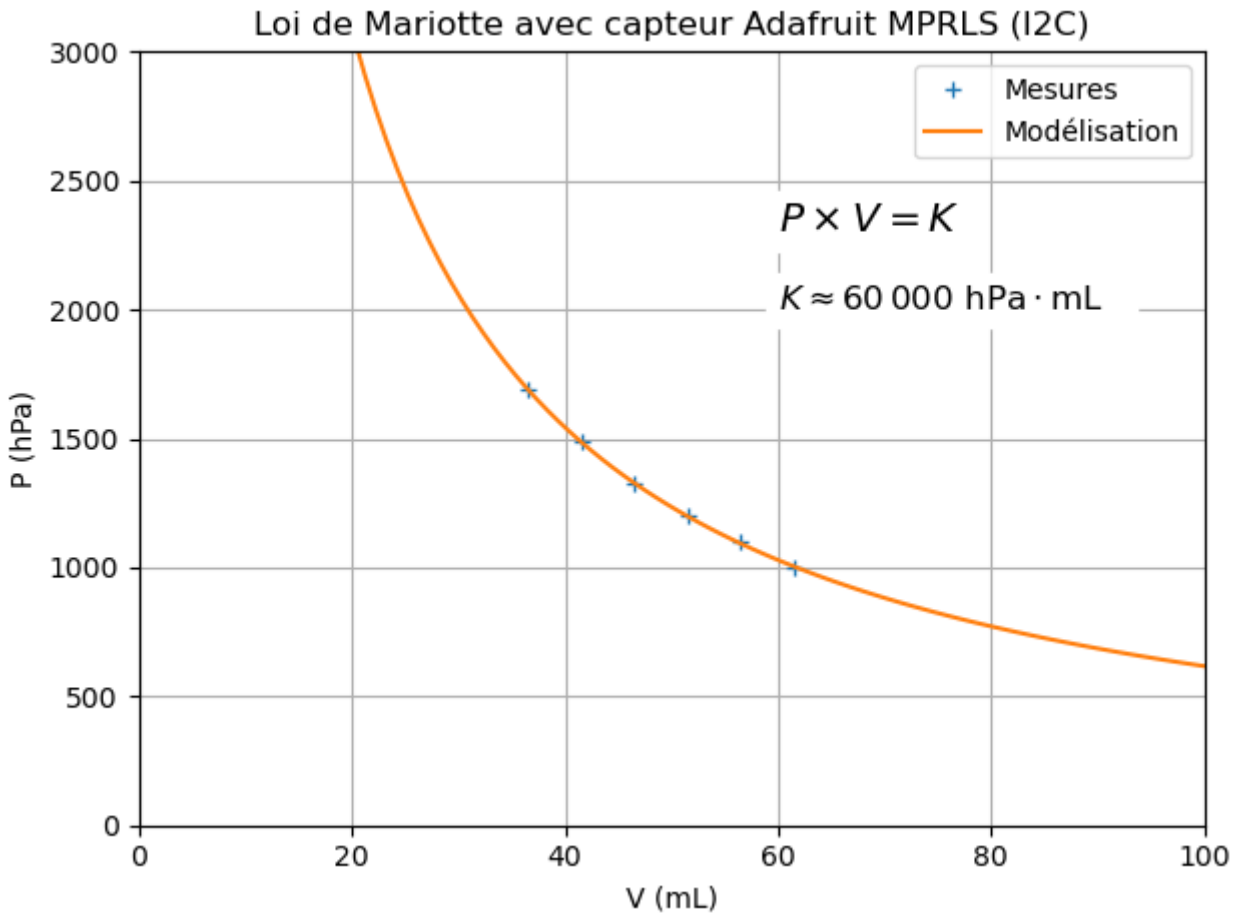
FIG. 20 – Résultats dans le moniteur série

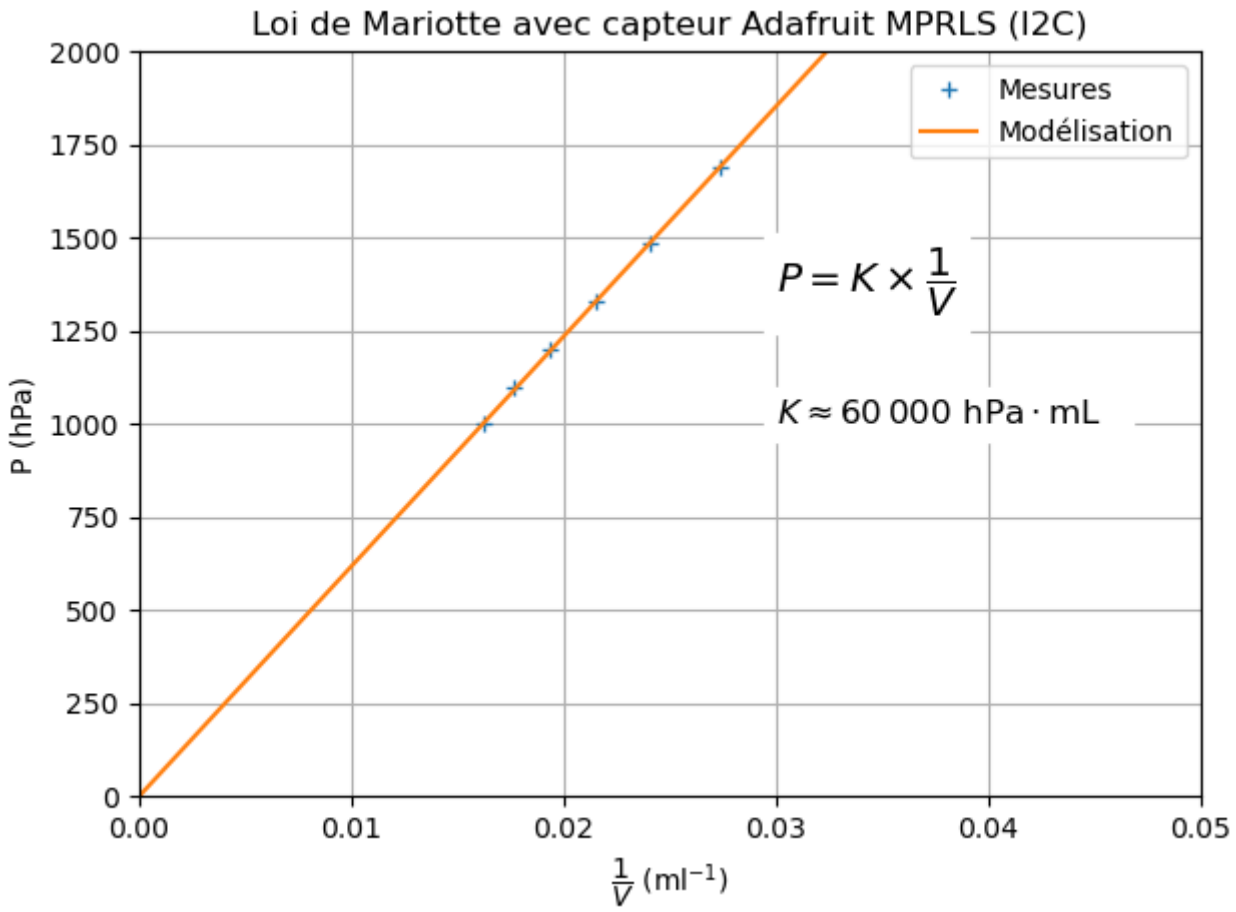
8.5.4 Résultats

Le volume final est donné par la relation :

$$V = V_s + V_0$$

— V_s est la volume de la seringue ;
— V_0 est le volume additionnel contenu dans le tube et le capteur (ici 1,5 mL).


Loi de Mariotte avec capteur Adafruit MPRLS (I2C)
Mesures
Modélisation
$P \times V = K$
$K \approx 60\,000$ hPa · mL
P (hPa)
V (mL)
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100


Loi de Mariotte avec capteur Adafruit MPRLS (I2C)
Mesures
Modélisation
$P = K \times \frac{1}{V}$
$K \approx 60\,000$ hPa · mL
P (hPa)
$\frac{1}{V}$ (ml^{-1})
2000
1750
1500
1250
1000
750
500
250
0
0.00
0.01
0.02
0.03
0.04
0.05

Chapitre 9

Mesurer une pression - Loi de la statique des fluides (première générale)

Programme de première générale 2019 - Enseignement de spécialité.

Tester la loi fondamentale de la statique des fluides.

9.1 Introduction

9.1.1 Principe

Vérifier expérimentalement la loi de la statique des fluides à l'aide d'un capteur de pression différentielle.

$$\boxed{P_B - P_A = \rho \times g \times (z_A - z_B)}$$

Cette expérimentation nécessite un capteur de pression relative (ou différentielle).

9.1.2 Capteurs analogique de pression relative (ou différentielle)

9.2 Loi de la statique des fluide avec capteur Plug'Uino

9.2.1 Présentation

9.2.2 Capteur XGZP6857A005KPG

Le capteur Plug'Uino (pression pour statistique des fluides) comporte un **capteur de pression relative** (jauge/manomètre) de référence XGZP6857A005KPG.

FIG. 1 – Capteur de pression relative XGZP6857A (manomètre)

Ce capteur délivre une tension analogique de 0, 5 V à 4, 5 V pour pression relative à la pression atmosphérique comprise entre 0 et 5 kPa.

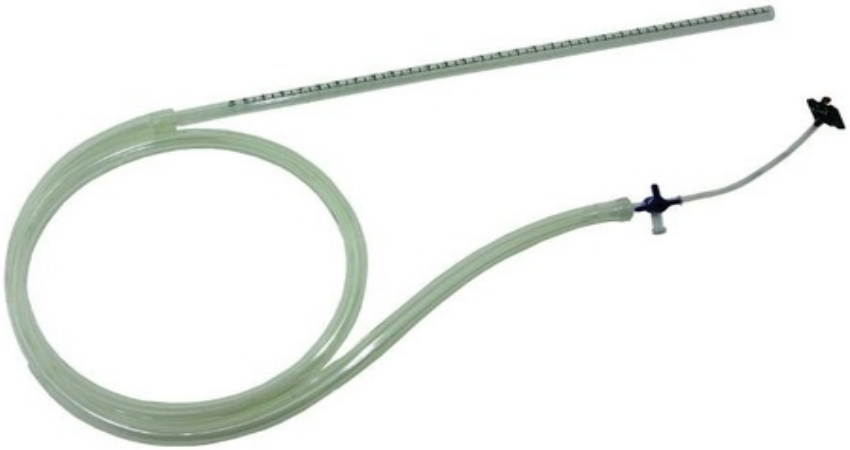


FIG. 2 – Kit Plug'Uino compatible Grove avec capteur XGZP6857A (image : Sciencéthic)

Item	Data	Unit
Power Supply	5(or 3.3 or 3)	V
Max. Excitation current	3	mA
Output Signal	0.5-4.5(or 0.2-2.7 or customized)	V
Accuracy(⩾20kPa)	±1.0(or ±0.5)	%Span
TCO(Temp. Coefficient of Offset)	±0.03	%FS/℃
TCS(Temp. Coefficient of Span)	±0.03	%FS/℃
Long Term Stability(1year)	±0.5	%Span
Over Pressure	2X	Rated
Compensation Temp.	0 ~ 60/32 ~ 140	℃/℉
Ambient Temp.	-10 ~ 85/14 ~ 176	℃/℉
Storage Temp.	-40 ~ 125/-40 ~ 257	℃/℉

FIG. 3 – Extrait de la documentation technique du capteur XGZP6857A

Le calcul de la pression obtient alors par l'expression suivante :

$$\boxed{P = \frac{P_{max} - P_{min}}{V_{max} - V_{min}} \times (v_{out} - V_{min}) + P_{min}}$$

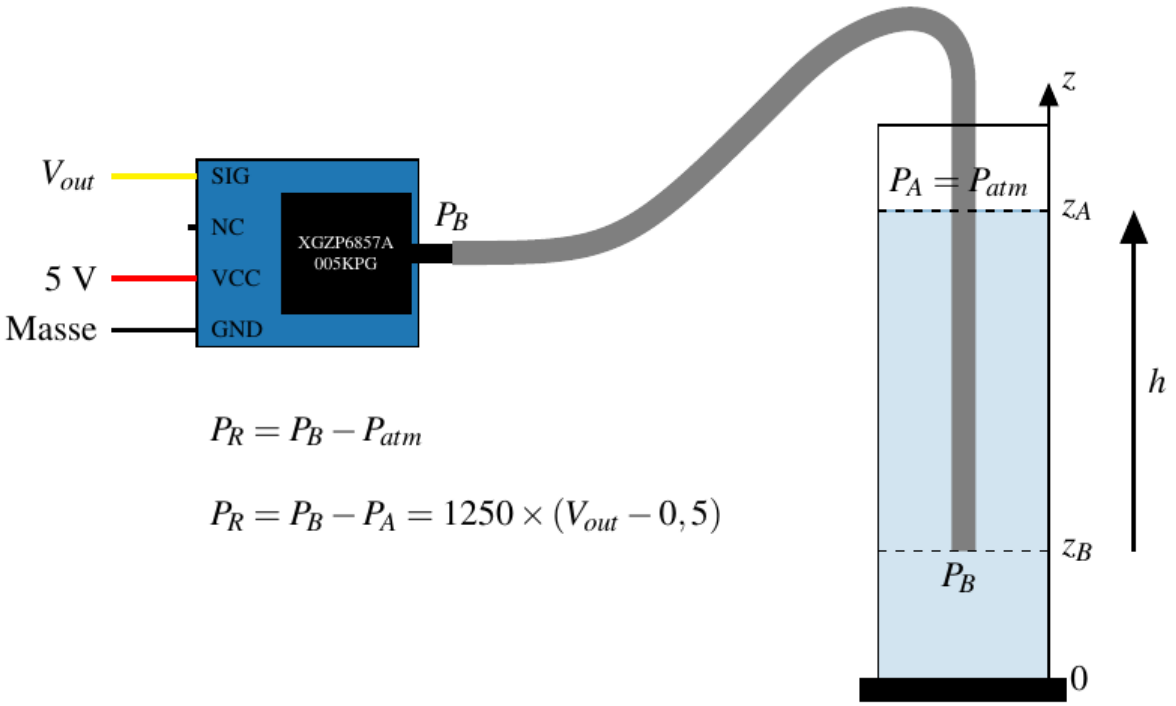
Soit :

$$P = \frac{5000 - 0}{4,5 - 0,5} \times (v_{out} - 0,5) + 0 \quad \Longrightarrow \quad \boxed{P \approx 1250 \times (v_{out} - 0,5)}$$

La précision du capteur est de :

$$\pm 0,5\% \times 5000 = \pm 25 \text{ Pa}$$

9.2.3 Montage



9.2.4 Programme Arduino

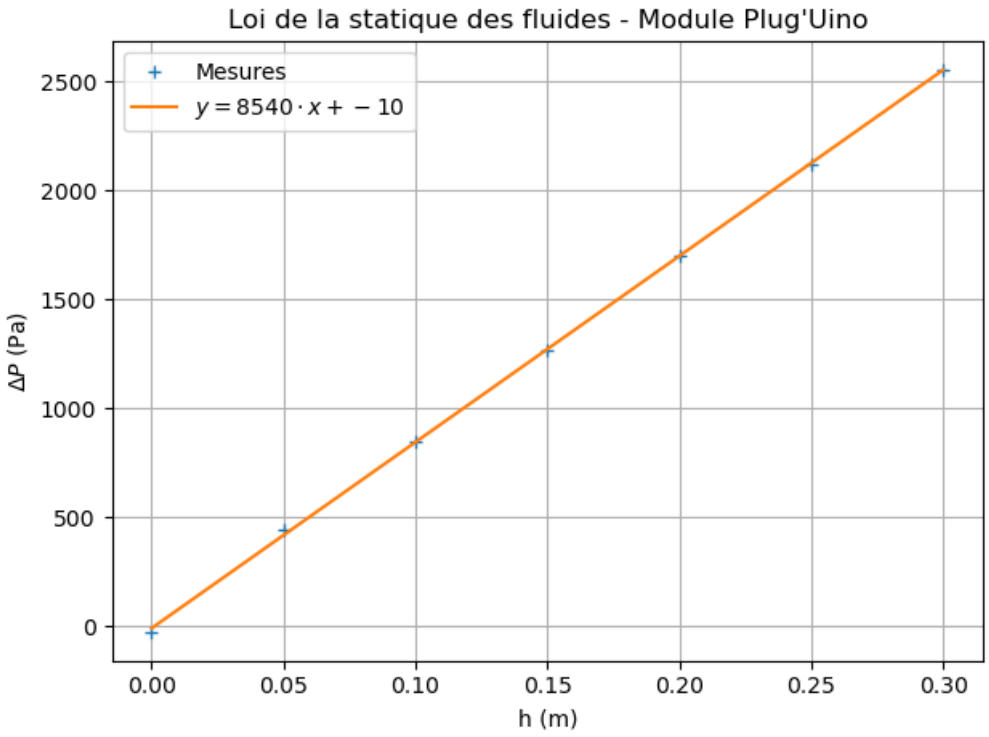
```
/*
 * Capteur de pression relative Plug'Uino
 * 0 - 5 kPa -> 0,5 - 4,5 V (capteur XGZP6857A005KPG)
 * David THERINCOURT - 2025
 */

void setup() {
   Serial.begin(9600);          // Initialisation du port série
}

void loop() {
   int N = analogRead(A0);       // Lecture sur A0
```

(suite sur la page suivante)

(suite de la page précédente)

```
    float u = N * (5.0 / 1023.0); // Calcul de la tension en volt
    float P = 1250*(u-0.5);       // Calcul de la pression en Pa
    Serial.print("U = ");         // Affichage
    Serial.print(u);              // ...
    Serial.print(" V \t");        // ... \t pour une tabulation
    Serial.print(P);              // ...
    Serial.println(" Pa");        // ...
    delay(1000);                  // Temporisation
}
```

FIG. 4 – Résultats dans le moniteur série d'Arduino IDE



9.3 Loi de la statique des fluides avec capteur Educaduino Lab

9.3.1 Présentation

9.3.2 Capteur MPX2010DP/GP

La mesure de pression s'effectue avec un capteur de pression différentiel du type MPX2010 (0 à 100 hPa) compensé en température.

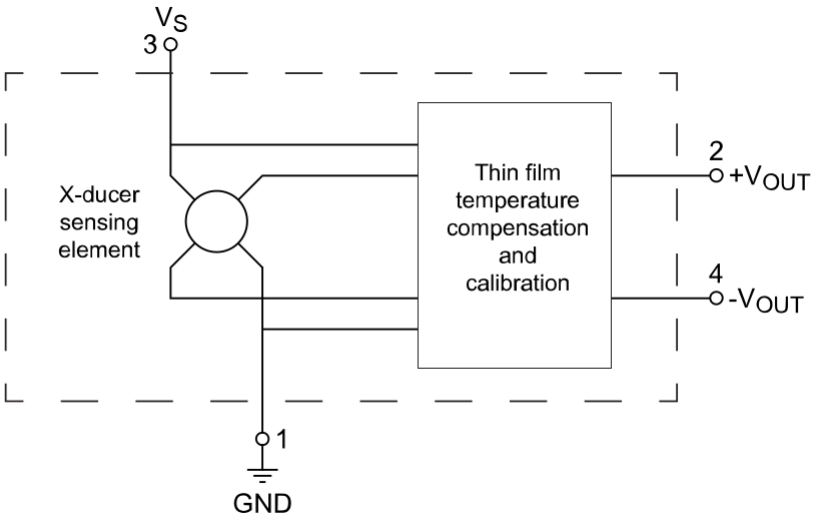


FIG. 5 – Schéma électrique du MPX2010 (image : NXP Semiconductors)

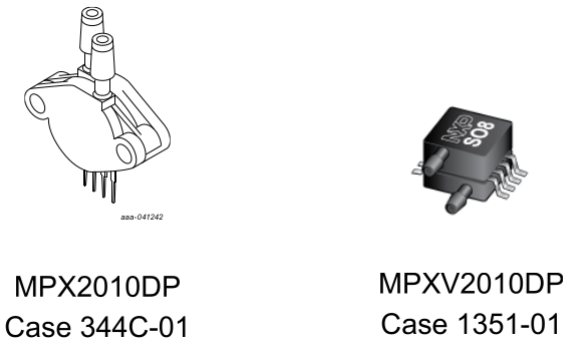
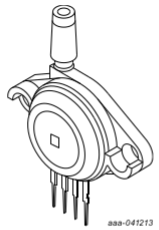


FIG. 6 – (image : NXP Semiconductors)

La **version DP** (pression différentielle) mesure la différence de pression entre deux points d'entrée.



MPX2010GP
Case 344B-01

MPXV2010GP
Case 1369-01

FIG. 7 – (image : NXP Semiconductors)

La **version GP** (pression relative - jauge) mesure la pression relative par rapport à la pression atmosphérique.

Table 9. Operating characteristics (V_S = 10 Vdc, T_A = 25 °C unless otherwise noted, P1 > P2)

Characteristic		Symbol	Min	Typ	Max	Units
Operating Pressure Range	[1]	P_{OP}	0	—	10	kPa
Supply Voltage	[2]	V_S	—	10	16	Vdc
Supply Current		I_o	—	6.0	—	mAdc
Full Scale Span	[3]	V_{FSS}	24	25	26	mV
Offset	[4]	V_{off}	–1.0	—	1.0	mV
Sensitivity		ΔV/ΔP	—	2.5	—	mV/kPa
Linearity	[5]	—	–1.0	—	1.0	$\%V_{FSS}$
Pressure Hysteresis (0 kPa to 10 kPa)	[5]	—	—	±0.1	—	$\%V_{FSS}$
Temperature Hysteresis (–40 °C to +125 °C)	[5]	—	—	±0.5	—	$\%V_{FSS}$
Temperature Coefficient of Full Scale Span	[5]	TCV_{FSS}	–1.0	—	1.0	$\%V_{FSS}$
Temperature Coefficient of Offset	[5]	TCV_{off}	–1.0	—	1.0	mV
Input Impedance		Z_{in}	1300	—	2550	Ω
Output Impedance		Z_{out}	1400	—	3000	Ω
Response Time (10% to 90%)	[6]	t_R	—	1.0	—	ms
Warm-Up Time	[7]	—	—	20	—	ms
Offset Stability	[8]	—	—	±0.5	—	$\%V_{FSS}$

FIG. 8 – Extrait datasheet (source : NXP Semiconductors)

Pour ce capteur, la tension de sortie est proportionnelle à pression relative mesurée telle que :

$$U_{out} = S \times P_R = S \times (P - P_{atm}) \qquad \text{avec} \qquad S = 2,5 \text{ mV/kPa}$$

9.3.3 Capteur Educaduino Lab (MPX2010GP)

Le module Educaduino LAB est conçu autour sur le capteur MPX2010GP.

FIG. 9 – Capteur de pression relative Educaduino LAB

Après adaptation (amplification), l'expression de la pression relative (en Pa) en fonction de la tension en sortie du capteur est :

$$P_R = P - P_{atm} = 2000 \times U \qquad \text{(Pa)}$$

9.3.4 Montage

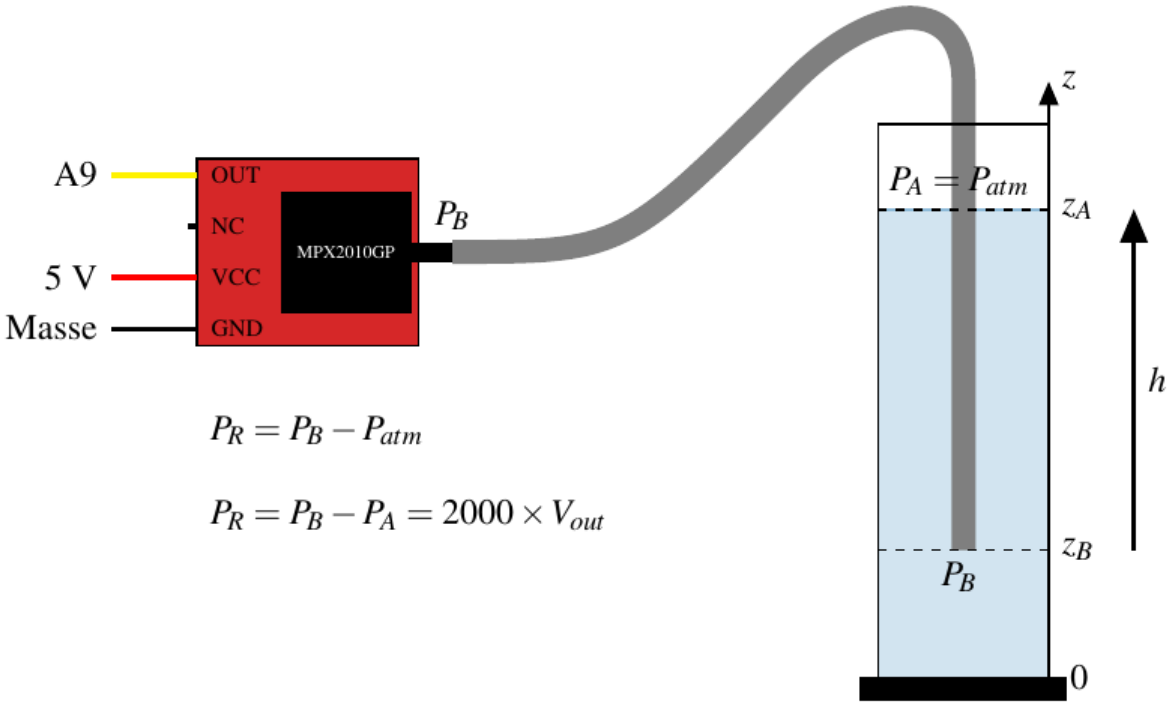


FIG. 10 – Montage de la vérification de loi de la statique des fluides à partir d'un Arduino

9.3.5 Programme Arduino

Avec écran LCD Educaduino LAB. La lecture de la tension analogique se fait sur la broche A9.

```
/*
 * Mesure d une pression relative
 * Capteur Educaduino MPX2010GP 0 à 10 kPa
 * branché sur la broche A9
 */

#define brocheCapteur A9      // Numéro de broche connectée au capteur
#include <LiquidCrystal.h>    // Librairie de gestion de l écran LCD

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // Déclaration de l écran LCD

int n;
float tension ;               // Tension mesurée
int pression ;                // Pression mesurée

void setup() {
  lcd.begin(16, 2);           // Paramétrage de l ecran LCD

}

void loop() {
  n = analogRead(brocheCapteur) ;   // Lecture de la tension
  tension = n*5.0/1023 ;            // Lecture de la tension
  pression = round(tension*2000) ;  // Calcul de la pression en Pa
  lcd.clear();                      // Début affichage
  lcd.setCursor(0,0);
  lcd.print("N");
  lcd.setCursor(6,0);
  lcd.print("P (Pa)");
  lcd.setCursor(0,1);
  lcd.print(n);                     // Fin affichage
  lcd.setCursor(6,1);
  lcd.print(round(pression));       // Fin affichage
  delay(1000);
}
```

Chapitre 10

Géométrie d'un condensateur (terminale générale)

Programme de terminale générale 2019 - Enseignement de spécialité.

Identifier et tester le comportement capacitif d'un dipôle. Illustrer qualitativement, par exemple à l'aide d'un micro-contrôleur, d'un multimètre ou d'une carte d'acquisition, **l'effet de la géométrie d'un condensateur sur la valeur de sa capacité**.

10.1 Introduction

L'expression de la capacité d'un condensateur plan est donnée par la relation :

$$C = \epsilon_0 \times \epsilon_r \times \frac{S}{e}$$

— $\epsilon_0 = 8,85 \cdot 10^{-12}$ F $\cdot$ m^{-1} est la permittivité du vide.
— ϵ_r est la permittivité relative du diélectrique.
— S est l'aire de la surface des armatures en m^2.
— e est l'épaisseur du diélectrique en m.

Cette relation met en évidence influence de la géométrie d'un condensateur plan sur la valeur de sa capacité.

Le fonctionnement des capteurs capacitifs repose sur la variation de l'un des paramètres ϵ_r, S ou e par une grandeur physique extérieure.

Paramètre modifié	Types de capteur capacitif
ϵ_r	Niveau d'eau, détection d'objet, ...
S	Position, rotation, ...
e	Pression, ...

10.2 Capteur capacitif tactile

Un capteur tactile peut-être facilement fabriqué à partie d'une plaque conductrice (ex. plaque de circuit imprimé, papier en aluminium, ...) de surface suffisante. Un film en plastique collé sur la plaque fait office d'isolant.

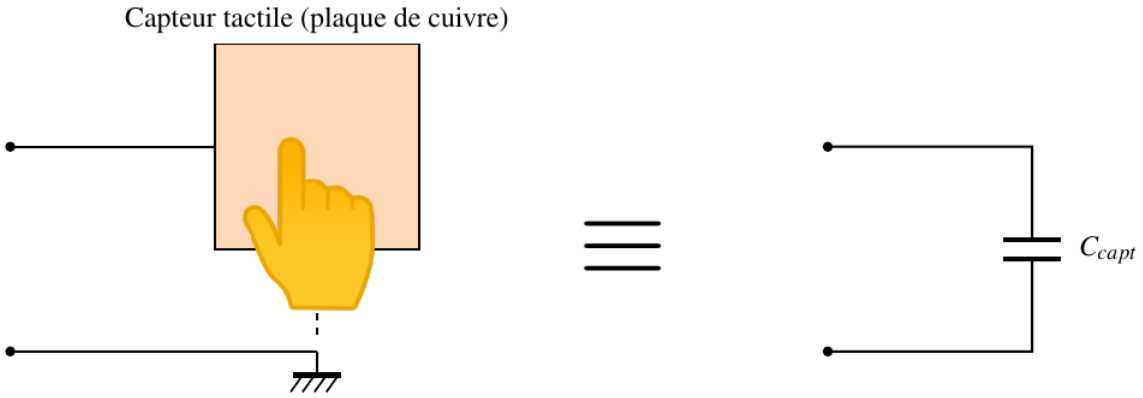


FIG. 1 – Equivalence d'un Capteur tactile capacitif

Par rapport à la terre, ce capteur tactile est équivalent à un condensateur de faible capacité (quelques centaines de pF). Cette capacité croît lorsque la main s'approche de la surface du capteur.

10.2.1 Montage

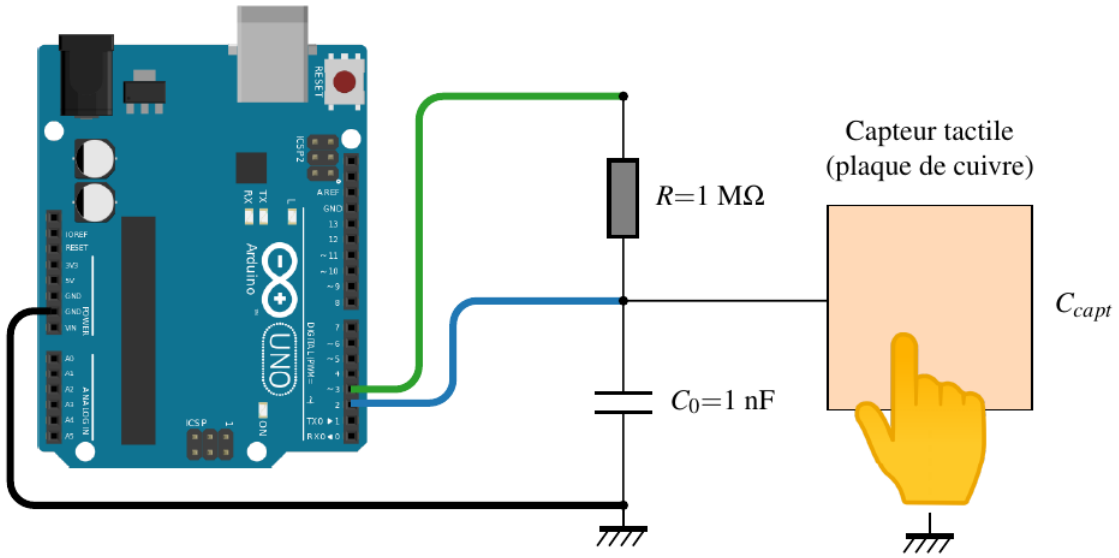


FIG. 2 – Montage Arduino Uno R3 avec capteur tactile capacitif

Un condensateur de capacité C_0 est placé en parallèle avec le capteur dans le but de fixer une constante de temps minimale. Cet ensemble a pour capacité équivalente :

$$C_{eq} = C_0 + C_{capt}$$

Ainsi, en approchant la main de la surface du capteur, la capacité équivalente C_{eq} augmente.

La broche D3 de l'Arduino est utilisée pour charger (et décharge) le condensateur équivalent à travers une résistance de 100 MΩ sous une tension constante de 5 V.

La broche D2 détecte la moitié de la charge du condensateur. En effet, une entrée digitale bascule d'un niveau logique 0 à un niveau logique 1 avec une tension de seuil environ égale à $\frac{V_{CC}}{2} = \frac{5,0}{2} = 2,5$ V.

10.2.2 Durées de basculement de la broche D2

Plus la valeur de la capacité est élevée, plus le basculement de l'entrée D2 est retardé !

On mesure les instants t_0 et t_1 pour lesquels l'entrée D2 change d'état **sans contact** puis **avec contact**.

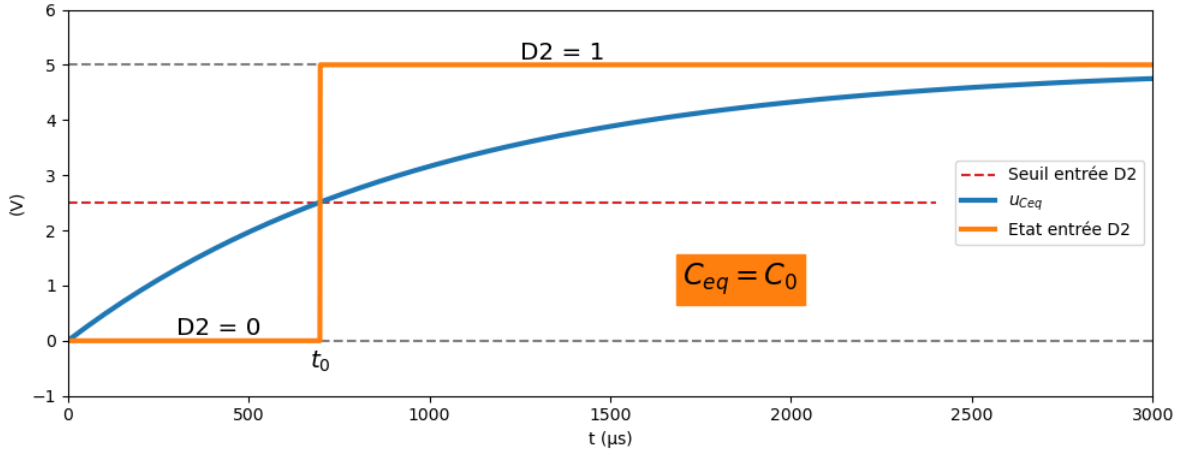


FIG. 3 – Basculement de l'entrée digitale D2 **sans contact**

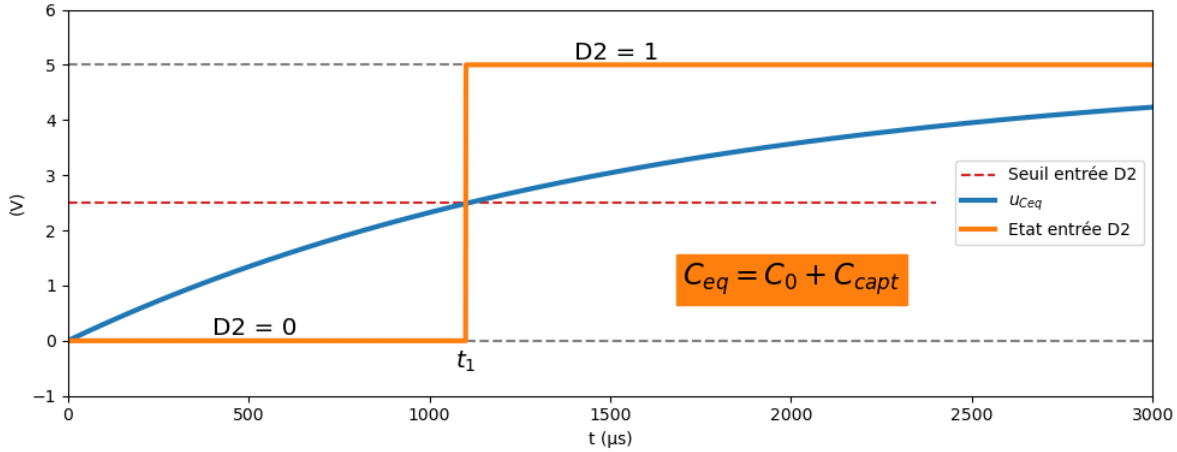


FIG. 4 – Basculement de l'entrée digitale D2 **avec contact**

Le programme ci-dessous utilise une **interruption** qui détecte le basculement sur un front montant de l'entrée D2. Au déclenchement de cette interruption, la fonction `detection()` est exécutée. Cette fonction mesure et affiche l'instant où l'entrée D2 bascule.

```
/* Capteur capacitif tactile
 * Mesure de la durée du basculement de l'entrée digitale D2 (seuil de 2,5V)
 * David THERINCOURT - 2025
 */

int pinD3 = 3;  // Broche D3
int pinD2 = 2;  // Broche D2

unsigned long t_init;           // Instant initial
unsigned long t_rise;           // Instant de basculement
```

(suite sur la page suivante)

(suite de la page précédente)

```
void setup() {
   Serial.begin(9600);             // Initialisation du port série
   pinMode(pinD3, OUTPUT);         // broche D3 en sortie (charge condensateur)
   pinMode(pinD2, INPUT);          // broche D2 en entrée (lecture tension condensateur)
   attachInterrupt(digitalPinToInterrupt(pinD2),  detection, RISING); // Parmetrage␣
 ↪interruption
}

void loop() {
   // DEBUT CHARGE COMPLETE
   digitalWrite(pinD3, 1);         // D3 à 5V
   t_init = micros();              // Mesure instant début
   delay(500);                     // Temporisation pour charge complète

   // DEBUT DECHAGRE COMPLETE
   digitalWrite(pinD3, 0);         // Début décharge du condensateur
   delay(500);                     // Temporisation pour charge complète
}

void detection() {
   t_rise = micros();              // Mesure instant du basculement de 0 à 5V
   Serial.print(t_rise - t_init); // Affichage durée de basculement
   Serial.println(" µs");          // Affichage unité + saut de ligne
}
```

FIG. 5 – Moments de basculement de l'entrée D2

On mesure donc :

$$t_0 \approx 640 \text{ s} \qquad t_1 \approx 1350 \text{ s}$$

Avertissement

La fonction `micros()` renvoie une durée avec une résolution de 4 s.

10.2.3 Programme final

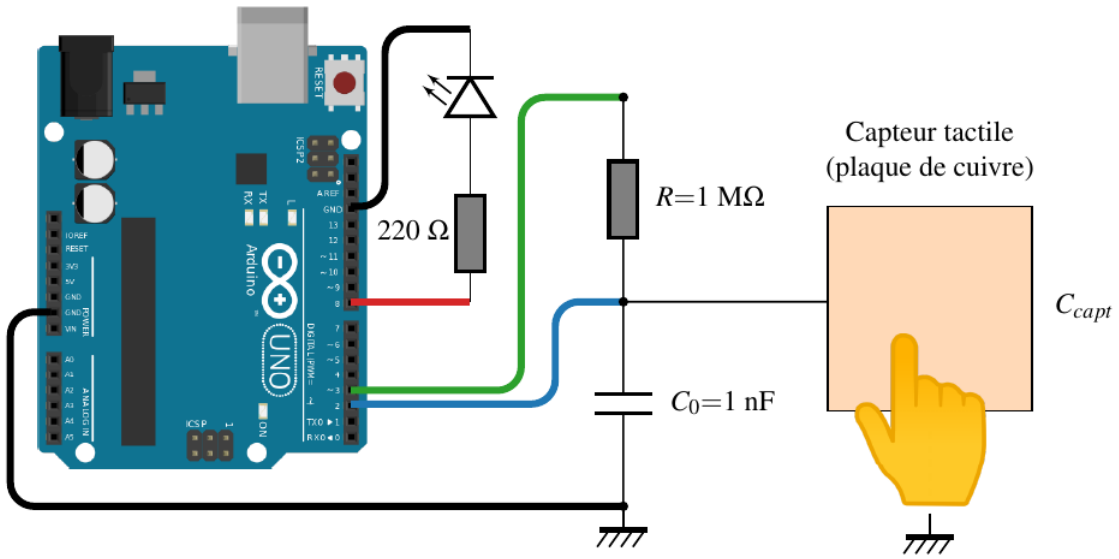


FIG. 6 – Montage complet avec une LED

En choisissant de lire l'état logique de la broche D2 à l'instant t_m tel que :

$$t_m = \frac{t_0 + t_1}{2} = \frac{640 + 1350}{2} \approx 1000\mu s$$

On obtient :

— 1 sans contact (LED allumée)
— 0 avec contact (LED éteinte)

```
/* Capteur capacitif tactile
 * R = 1M    C = 1nF  + plaque epoxy cuivrée 30cm x 10cm
 * David THERINCOURT - 2025
 */

int pinD3  = 3; // Broche D3 pour la charge
int pinD2  = 2; // Broche D2 pour la lecture de la tension du condensateur
int pinLed = 8; // Broche LED
int etat;       // Etat de D3

void setup() {
   Serial.begin(9600);          // Initialisation du port série
   pinMode(pinD3, OUTPUT);      // D2 en sortie (charge condensateur)
   pinMode(pinD2, INPUT);       // D3 en entrée (lecture tension condensateur)
   pinMode(pinLed, OUTPUT);     // LED sur broche 8
}

void loop() {
   // CHARGE DU CONDENSATEUR
   digitalWrite(pinD3, HIGH);   // D3 à 5V
   delayMicroseconds(1000);     // Temporisation en µs
   etat = digitalRead(pinD2);   // Lecture de l'état logique de D2
   Serial.println(etat);        // Affichage de l'état logique de D2
   digitalWrite(pinLed, etat);  // Allumer/éteindre la LED
```

(suite sur la page suivante)

(suite de la page précédente)

```
    // DECHARGE TOTALE
    digitalWrite(pinD3, 0);       // D3 à 0V
    delay(1000);                  // Temporisation de 1s
}
```

Chapitre 11

Capteur capacitif (terminale générale)

Programme de terminale générale 2019 - Enseignement de spécialité.

Expliquer le principe de fonctionnement de quelques capteurs capacitifs. Étudier la réponse d'un dispositif modélisé par un dipôle RC. **Déterminer le temps caractéristique d'un dipôle RC** à l'aide d'un microcontrôleur, d'une carte d'acquisition ou d'un oscilloscope.

11.1 Introduction

Soit le circuit RC série suivant :

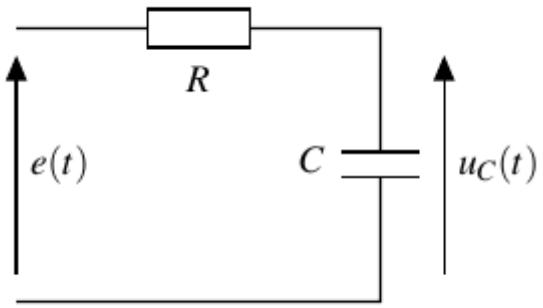


FIG. 1 – Schéma électrique

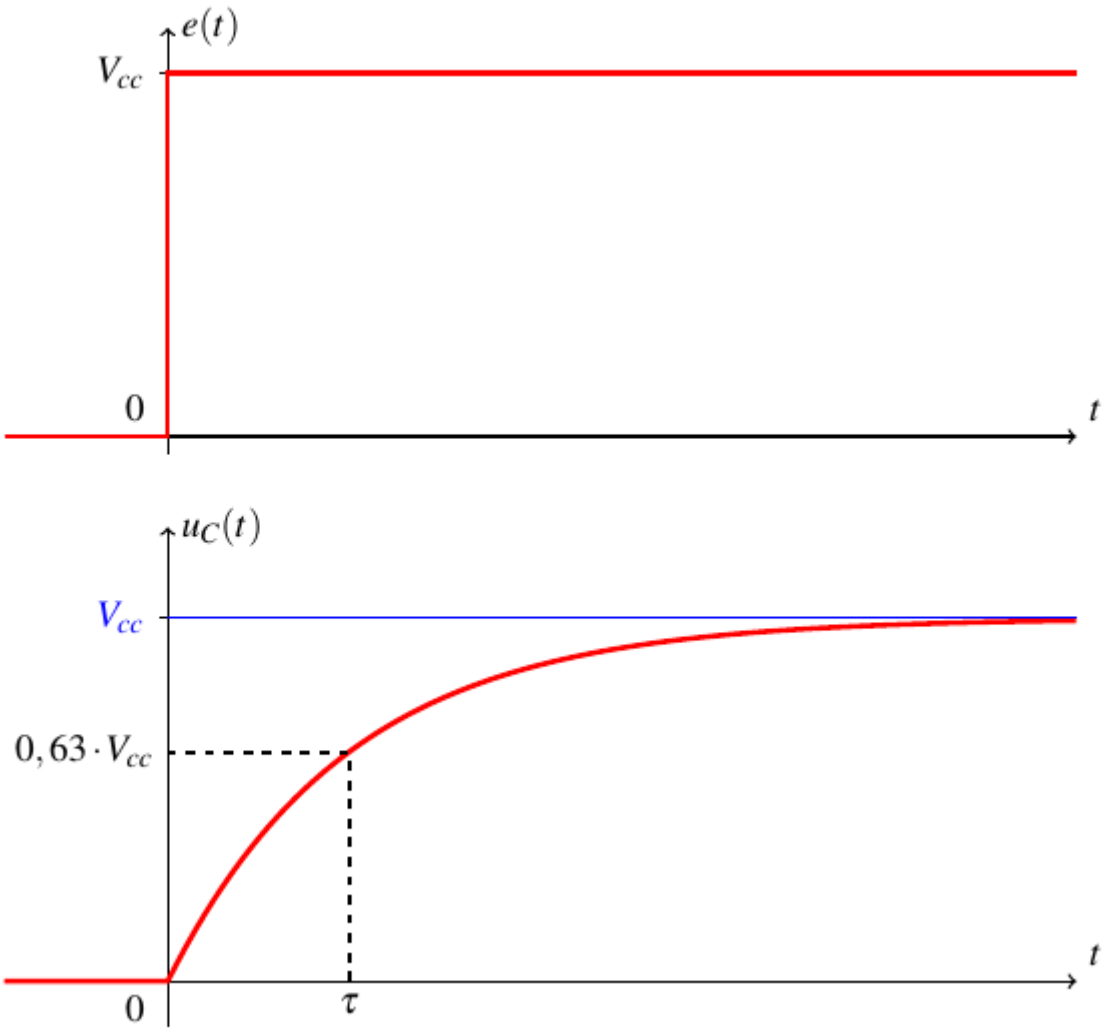


FIG. 2 – Évolution de la tension du condensateur en fonction du temps

Lors de la charge du condensateur C à travers la résistance R sous la tension constante V_{CC}, le **temps caractéristique** (ou constante de temps) τ est la durée que prend la tension u_C pour atteindre 63% de sa valeur finale V_{cc}.

Avec un microcontrôleur, il est assez facile de mesurer ce temps caractéristique par une **mesure de durée** jusqu'à la **détection du seuil de 63%** de la valeur finale de la tension du condensateur.

11.2 Mesure de la constante de temps d'un circuit RC avec l'entrée analogique

11.2.1 Montage

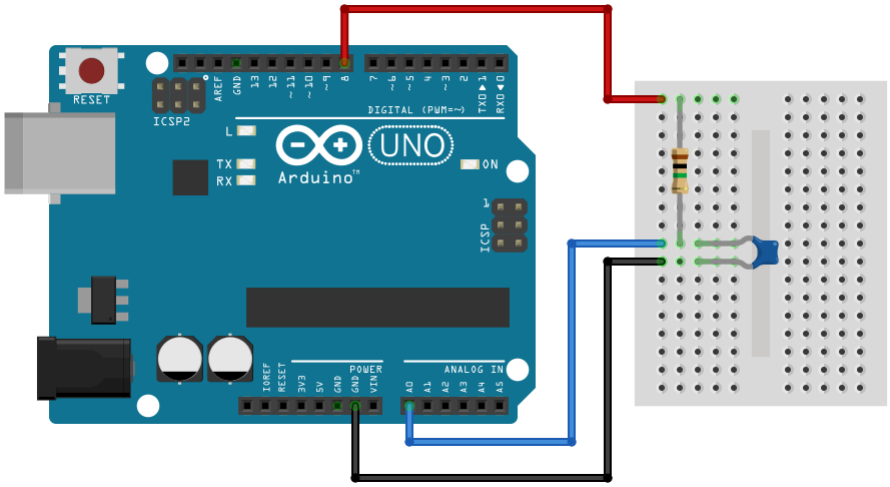


FIG. 3 – Charge d'un condensateur sous tension constante avec un Arduino UNO R3

La broche `D8`, paramétrée en sortie digitale, charge (ou décharge) le condensateur à travers la résistance.

L'entrée analogique `A0` mesure périodiquement la tension aux bornes du condensateur.

Avertissement

— La durée de conversion, sur les entrées analogiques, est en moyenne de 100 s. En conséquence, pour une bonne précision (ex. au moins 100 points de mesures sur la durée à mesurer), il faut choisir une **constante de temps minimale** telle que :

$$\tau_{min} \approx 100 \times 100 \cdot 10^{-6} \approx 10 \text{ ms}$$

— Il est conseillé de choisir une **résistance maximale** à 100 kΩ (la documentation technique précise 10 kΩ).

Données :

$$R = 100 \text{ k}\Omega \qquad C = 680 \text{ nF}$$

11.2.2 Mesure de la constante de temps

Le programme Arduino ci-dessous charge le condensateur sous la tension constante 5 V et mesure les instants `t0` (début de la charge) et `t1` (63% de la charge) afin de calculer de la constante de temps `tau`.

Le programme mesure également les valeurs de la tension initiale `Nmin` et de la tension finale `Nmax` afin de vérifier si la charge du condensateur est conforme.

Note

Il faut appuyer sur le bouton Reset de l'Arduino pour lancer la mesure. Il n'y a pas de boucle infinie !

```
/* Mesure de constante de temps d'un circuit RC
 * David THERINCOURT - 2025
 * Appuyer sur le bouton RESET de l'Arduino pour lancer la mesure
```

(suite sur la page suivante)

(suite de la page précédente)

```
*/

int pinD8 = 8;                // Broche D8
int Nmin, Nmax;               // Valeurs minale et maximale
unsigned long t0;             // Instant initiale
unsigned long t1;             // Instant à 63% de Vcc
unsigned long tau;            // Constante de temps


void setup() {
   // PARAMETRAGE
   Serial.begin(9600);         // Initialisation du port série
   pinMode(pinD8, OUTPUT);     // Broche digitale en sortie

   // DECHARGE COMPLETE
   digitalWrite(pinD8,LOW);    // D8 à 0V
   delay(1000);                // Temporisation pour décharge compléte
   Nmin = analogRead(A0);      // Tension de bébut de charge (10 bit)

   // CHARGE COMPLETE
   int N = 0;                  // Initialisation
   digitalWrite(pinD8, HIGH); // D8 à 5V
   t0 = micros();              // Mesure de l'instant initial
   while (N<646) {             // Boucle tant que la tension inférieure à seuil (0,
↪632*1023=646)
      N = analogRead(A0);       // Lecture de la tension du condensateur
   }
   t1 = micros();              // Mesure de l'instant à 63% de la valeur finale
   tau = t1 - t0;              // Calcul de la constante de temps (µs)

   delay(1000);                // Temporisation pour charge complete
   Nmax = analogRead(A0);      // Tension de fin de charge (10 bit)

   // AFFICHAGE
   Serial.println("-------");  //
   Serial.print("Nmin = ");     //
   Serial.println(Nmin);        //
   Serial.print("Nmax = ");     //
   Serial.println(Nmax);        //
   Serial.print("tau = ");      //
   Serial.print(tau/1000.0);    //
   Serial.println(" ms");       //
}

void loop() {
   // Boucle sans fin pas utilisée ici !
}
```

FIG. 4 – Mesures pour des capacités de 470 nF, 680 nF et 1000 nF (±10%)

11.2.3 Mesure de la capacité du condensateur

Sachant que le temps caractéristique est défini par la relation :

$$\tau = R \cdot C$$

Le calcul de la capacité C du condensateur est :

$$C = \frac{\tau}{R}$$

If suffit donc d'ajouter cette relation dans le code précédent !

```
/* Mesure de constante de temps d'un circuit RC
 * David THERINCOURT - 2025
 * Appuyer sur le RESET de l'Arduino pour lancer la mesure
 */

int pinD8 = 8;                // Broche D8
int Nmin, Nmax;               // Valeurs minale et maximale
unsigned long t0;             // Instant initiale
unsigned long t1;             // Instant à 63% de Vcc
unsigned long tau;            // Constante de temps
float R = 100E3;              // Valeur de la résistance
float C;                      // Valeur de la capacité à calculer

void setup() {
   // PARAMETRAGE
   Serial.begin(9600);         // Initialisation du port série
   pinMode(pinD8, OUTPUT);     // Broche digitale en sortie

   // DECHARGE COMPLETE
   digitalWrite(pinD8,LOW);    // D8 à 0V
   delay(1000);                // Temporisation pour décharge compléte
   Nmin = analogRead(A0);      // Tension de bébut de charge (10 bit)

   // CHARGE COMPLETE
   int N = 0;                  // Initialisation
   digitalWrite(pinD8, HIGH); // D8 à 5V
```

(suite sur la page suivante)

(suite de la page précédente)

```
   t0 = micros();              // Mesure de l'instant initial
   while (N<646) {             // Boucle tant que la tension inférieure à seuil (0,
↪632*1023=646)
      N = analogRead(A0);       // Lecture de la tension du condensateur
   }
   t1 = micros();              // Mesure de l'instant à 63% de la valeur finale
   tau = t1 - t0;              // Calcul de la constante de temps (µs)
   C = tau/R*1E3;              // Calcul de la capacité en nF


   delay(1000);                // Temporisation pour charge complete
   Nmax = analogRead(A0);      // Tension de fin de charge (10 bit)

   // AFFICHAGE
   Serial.println("-------");  //
   Serial.print("Nmin = ");     //
   Serial.println(Nmin);        //
   Serial.print("Nmax = ");     //
   Serial.println(Nmax);        //
   Serial.print("tau = ");      //
   Serial.print(tau/1000.0);    //
   Serial.println(" ms");       //
   Serial.print("C = ");        //
   Serial.print(C);             //
   Serial.println(" nF");       //
}

void loop() {
   // Boucle sans fin pas utilisée ici !
}
```

FIG. 5 – Mesures pour des capacités de 470 nF, 680 nF et 1000 nF (±10%)

11.2.4 A retenir

— La fonction `micros()` renvoie la durée en µs (< 70 min) depuis que la carte Arduino a été mise sous tension. La résolution est de 4 µs !
— La boucle `while` (tant que) associée à la fonction `analogRead()` détecte le seuil de 63% de la tension du condensateur.

11.3 Mesurer de la constante de temps d'un circuit RC avec le comparateur analogique

11.3.1 Principe

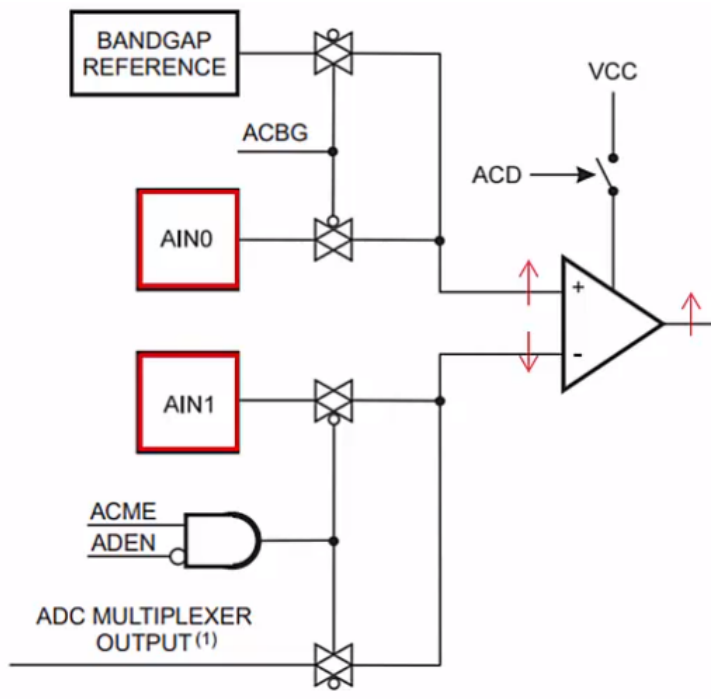


FIG. 6 – Comparateur analogique de l'Arduino Uno R3 sur les entrée `AIN0` et `AIN1`

11.3.2 Montage

— La tension de référence égale à 63% de `VCC` est appliquée sur `AIN0` à l'aide d'un potentiomètre.
— La tension du condensateur est appliquée sur `AIN1`.
— Le circuit RC est chargé par la broche D8.

Le basculement (front montant) du comparateur analogique de l'Arduino se fait lorsque la tension du condensateur atteint la tension de référence (63% de la valeur finale).

11.3.3 Programme

```
/* Mesure de la constante de temps d'un circuit RC
 * avec le comparateur analogique d'un Arduino Uno R3
 * David THERINCOURT - 2025
 */

#include "analogComp.h"

unsigned long t0;  // 32 bits
unsigned long t1;
unsigned long tau;
float C;
float R = 1.0E6;
```

(suite sur la page suivante)

(suite de la page précédente)

```
void setup() {
   Serial.begin(9600);
   Serial.println("-----------");

   analogComparator.setOn(AIN0, AIN1); //we instruct the lib to use voltages on the pins
   analogComparator.enableInterrupt(changeStatus, RISING);

   pinMode(8,OUTPUT);      // Broche digitale en sortie

   // DECHARGE COMPLETE
   digitalWrite(8,LOW);    // Décharge condensateur avant mesure
   delay(1000);               // pendant 1 s


   // DEBUT CHARGE
   digitalWrite(8,HIGH);   // Début charge condensateur
   t0 = micros();             // Mesure instant initial
}

void loop() {

}

void changeStatus() {
   analogComparator.disableInterrupt();
   t1 = micros();             // Mesure instant où seuil atteint
   tau = t1 - t0;              // Calcul de tau
   C = tau/R * 1E4;
   Serial.print("tau = ");
   Serial.print(tau);          // Début affichage
   Serial.print(" µs \t C = ");
   Serial.print(C);           // Début affichage
   Serial.println(" nF");
}
```

FIG. 7 – Mesures avec une boîte à décade de condensateurs.

Chapitre 12

Pour aller plus loin

12.1 Contrôler l'intensité d'une LED avec un potentiomètre

12.1.1 Montage

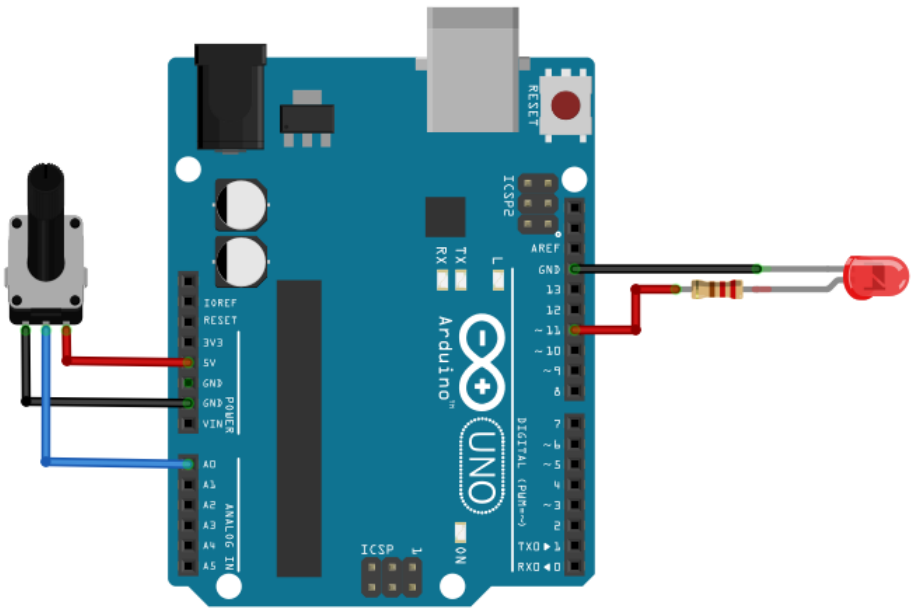


FIG. 1 – Commande de l'intensité d'une LED avec un potentiomètre

12.1.2 Programme

```
#define pinLED 11

int N;       // Valeur lue sur A0 de 0 à 1023
int duty;    // Rapport cyclique de 0 à 255

void setup() {
}

void loop() {
  N = analogRead(A0);       // Conversion analogique-numérique sur A0
```

(suite sur la page suivante)

(suite de la page précédente)

```
  duty = N/4;                // Calcul du rapport cyclique
  analogWrite(pinLED, duty); // Génération de la tension PWM
  delay(30);                 // Attendre 30 ms
}
```

Note

Pour convertir un entier sur 10 bits en un entier sur 8 bits, il suffit de la **division entière par 4** !

12.2 Grove - Afficheur LCD 16x2 sur port I2C

12.2.1 Principe

Il s'agit d'un afficheur de 32 caractères disposés sur 2 lignes (16 caractères par ligne).

La communication avec le microcontrôleur s'effectue via un port série de données (I2C) nécessitant moins de câbles (4 en tout).

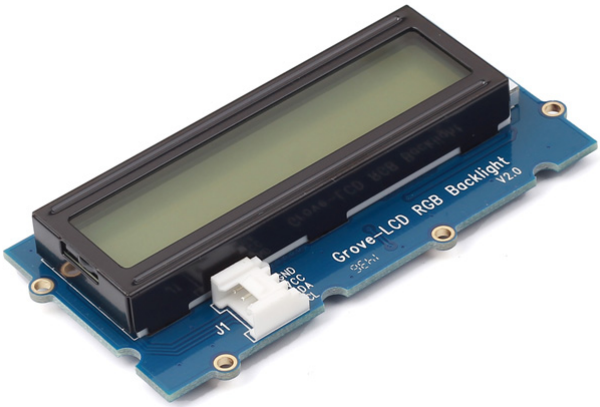


FIG. 2 – Module Grove - LCD RGB Backlight (image : http://wiki.seeedstudio.com)

Chaque afficheur utilise sa propre librairie (ex. `rgb_lcd.h` pour le Grove LCD RGB Backlight) en plus de la librairie `wire.h` qui est obligatoire pour la gestion du port I2C.

12.2.2 Grove LCD RGB Backlight

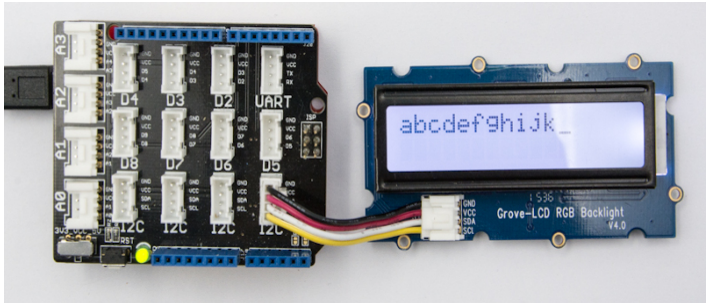


FIG. 3 – Modules Grove (image : http://wiki.seeedstudio.com)

Télécharger ici le fichier `Grove_LCD_RGB_Backlight-master.zip` pour l'installation de la librairie `rgb_lcd` avant la compilation du programme.

```
/*
 * Exemple affichage sur LCD 2x16 RGB I2C Grove
 */


#include <Wire.h>       // Importation librairie gestion port I2C
#include "rgb_lcd.h"    // Importation librairie gestion afficheur LCD I2C Grove

rgb_lcd lcd;            // Déclaration de l'afficheur

const int colorR = 255; // Couleur fond écran
const int colorG = 0;
const int colorB = 0;

void setup()
{
    lcd.begin(16, 2);                       // Fixe 2 colonnes et 16 caractères/ligne
    lcd.setRGB(colorR, colorG, colorB);     // Fixe couleur de fond
    lcd.print("hello, world!");             // Affiche texte
    delay(1000);                            // Temporisation 1s
}

void loop()
{
    lcd.setCursor(0, 1);                    // Déplace le curseur
    lcd.print(millis()/1000);               // Affiche le temps écoulé en s (timer␣
 ↪interne)
    delay(100);                             // Temporisation 1s
}
```

12.2.3 Grove LCD 16x2 (rouge/jaune/bleu)

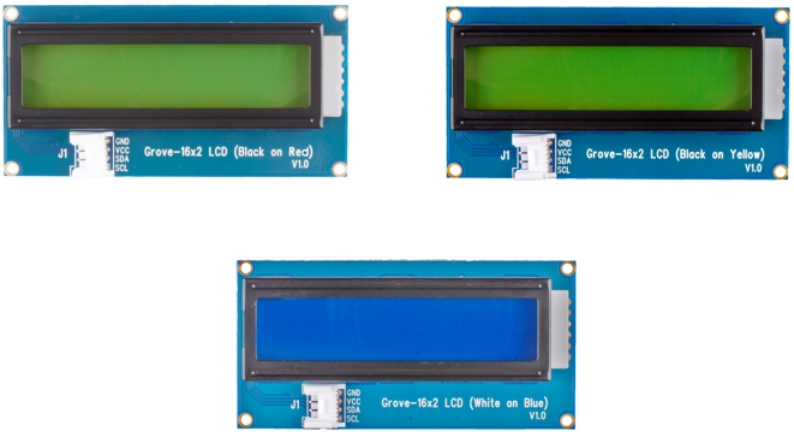


FIG. 4 – Grove LCD 16x12 (image : http://wiki.seeedstudio.com)

Toujours avec la librairie précédente `rgb_lcd`.

```
/*
 * Exemple affichage sur LCD 2x16 I2C Grove sans RGB
 */

#include <Wire.h>
#include "rgb_lcd.h"

rgb_lcd lcd;         // Déclation de l'afficheur LCD branché sur un port I2C

void setup()
{
   lcd.begin(16, 2);  // Initialisation de l'afficheur LCD sur 2 lignes à 16 caractères
}

void loop()
{
   lcd.setCursor(0, 0);         // Placement du curseur
   lcd.print("Bonjour !");      // Affichage de la valeur de la tension
   delay(1000);                 // Pause de 1000 ms
}
```

12.3 Educaduino Lab - Afficheur LCD

12.3.1 Principe

Le pilotage d'un afficheur LCD 16x2 nécessite de 6 broches numériques. Le brochage dépend de la référence de l'afficheur.

Pour éviter un câblage trop complexe, le plus simple est de fixer sur la carte de développement un « shield » afficheur comme celui proposé par Educaduino LAB.

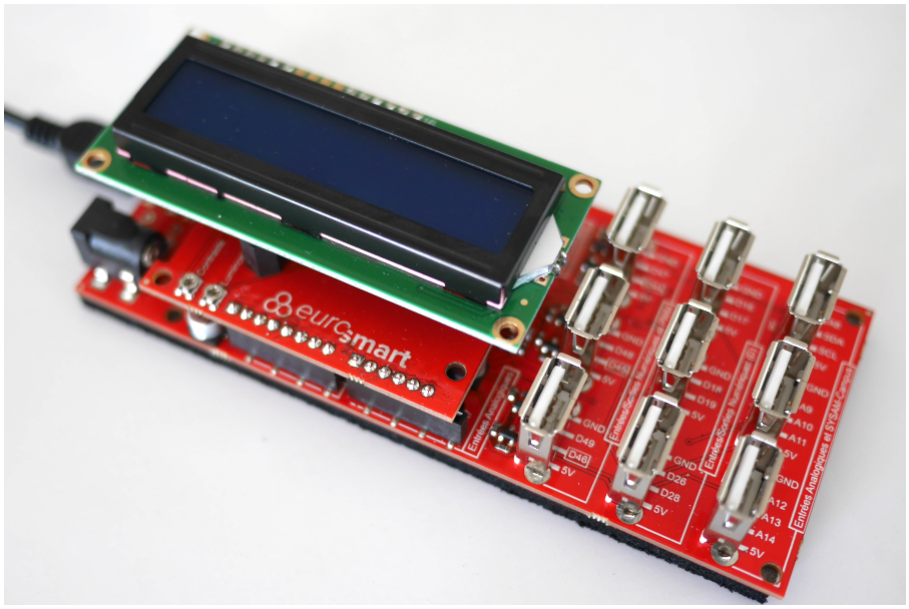


FIG. 5 – Afficheur LCD 16x2 d'Educaduino-Lab

12.3.2 Programme

La librairie `LiquidCrystal` est installée par défaut dans Arduino IDE.

```
/*
 * Exemple d'utilisation d'un écran LCD 16x2 parallèle
 */

#include <LiquidCrystal.h>        // Importation de la librairie LiquidCrystal

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // Brochage de l'afficheur

void setup() {
  lcd.begin(16, 2);               // fixe le nombre de colonnes et de lignes de l␣
 ↪afficheur
}

void loop() {
  lcd.setCursor(5,0);             // place le curseur à la colonne 5 et à la ligne 0
  lcd.print("Bonjour");           // Affiche un texte
  lcd.setCursor(0,1);             // place le curseur à la colonne 0 et à la ligne 1
  lcd.print("tout le monde !");   // Affiche un autre texte
}
```

Note

Il sera peut-être necessaire d'installer la librairie `LiquidCrystal` dans le logiciel Arduino.

12.3.2.1 En résumé

Instruction	Description
`#include <LiquidCrystal.h>`	Importe la librairie de gestion de l'afficheur LCD
`LiquidCrystal lcd(12, 11, 5, 4, 3, 2)`	Déclare l'afficheur en précisant les numéros de broches
`lcd.begin(16, 2)`	Fixe le nombre de colonnes et de lignes de l'afficheur
`lcd.setCursor(col,line)`	Positionne le cuseur
`lcd.print(variable)`	Affiche le contenu d'une variable à la position du curseur

---

Télécharger la documentation en PDF : `physique-microcontroleur.pdf`

Contacter l'auteur : david-therincourt.fr/contact