République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

**Université Amar Thelidji- Laghouat**

**FACULTE ou INSTITUT :** Sciences et Technologies

**DEPARTEMENT :** Electronique

# MEMOIRE DE MASTER

**Présenté par :**

**Chihila Dalila**

**Badji Rayane**

**DOMAINE :** Sciences et Technologies

**FILIERE :** Electronique

**OPTION :** Électronique des systèmes embarqués

## *Thème*

# Simulation numérique d'un capteur de pression à l'aide d'un montage Arduino et modélisation des données via Python

**Jury de soutenance :**

| Nom et Prénom | Grade | Qualité |
|---|---|---|
| Mr : B. Mouhoub | MCA | Président |
| Melle : L. A. Vilbois | MAA | Examinateur |
| Mr : A. Mouhoub | MCA | Rapporteur |

**Promotion : Juin – 2025**

# *Remerciements*

*Louange à Dieu, par Sa grâce les bonnes œuvres s'accomplissent. Je Le remercie infiniment pour Ses bienfaits innombrables et pour m'avoir permis d'achever ce travail humble.*

*Je tiens à exprimer ma profonde gratitude et mes sincères remerciements à mon honorable encadrant, Monsieur le* ***Dr. Abdelhafid Mouhoub*** *Maître de conférences à l'université de Laghouat, pour son encadrement précieux, ses conseils éclairés et son accompagnement bienveillant tout au long de l'élaboration de ce mémoire. Que Dieu le récompense généreusement et fasse de ses efforts une source de bénédiction dans cette vie et dans l'au-delà.*

*J'adresse également mes remerciements les plus sincères aux membres du jury, pour l'honneur qu'ils m'ont accordé en acceptant de juger ce travail, ainsi que pour leurs observations constructives et enrichissantes.*

*Mes remerciements vont aussi à ma chère famille, pilier de mon parcours, pour leur soutien inconditionnel, leurs prières et leur présence constante. Je remercie également mes amis et tous ceux qui, de près ou de loin, m'ont apporté aide, motivation, ou même une simple parole d'encouragement.*

*Enfin, je rends grâce à Dieu pour son aide et sa guidance, et je prie pour que ce travail soit utile et accepté*

# Sommaire

Liste des symboles et des abréviations

Liste des figures

Liste des tableaux

# Liste des symboles et des abréviations

| Symbole | Signification | Unité |
|---|---|---|
| P | Pression | Pa (Pascal) |
| F | Force | N (Newton) |
| S | Surface | $m^2$ |
| r | Rayon | M |
| g | Accélération gravitationnelle | $m/s^2$ |
| h | Hauteur | M |
| $v^2$ | Carré de la vitesse | $(m/s)^2$ |
| R | Résistance électrique | Ω (Ohm) |
| L | Inductance | H (Henry) |
| C | Capacité | F (Farad) |
| αs | Coefficient de dilatation solide | 1/°C |
| αm | Coefficient de dilatation liquide | 1/°C |
| s | Temps | S |
| m | Longueur | M |

| | | |
|---|---|---|
| So | Surface initiale | $m^2$ |
| ΔR | Variation de résistance | Ω |
| ΔL | Variation de longueur | M |
| NEMS | Nano Electro Mechanical Systems | - |
| TRMS | Valeur efficace réelle | - |
| ABS | Valeur absolue | - |
| PWM | Modulation de largeur d'impulsion | - |
| SPI | Interface série périphérique | - |
| VCC | Tension d'alimentation | V |
| Vout | Tension de sortie | V |
| TYP | Valeur typique | - |
| K | Constante ou température en Kelvin | K |
| V | Tension | V |
| 1/VC | Inverse de la vitesse du son | s/m |
| B | Champ magnétique | T (Tesla) |
| ρ | Masse volumique / densité | $kg/m^3$ |
| ZB - ZA | Différence d'altitude | M |

# Liste des figures

# Liste des tableaux

# INTRODUCTION GÉNÉRALE

## Introduction générale

La Pression Atmosphérique est un paramètre important dans les sciences météorologiques puisqu'elle permet de contribuer dans la prévision de la météo ; elle permet entre autre de nous informer sur l'évolution du climat dans une zone donnée de la Terre. Ce sens de variation du climat dépend étroitement de la pression atmosphérique de la Terre en un point donné [1].

Les capteurs de pression sont largement utilisés dans de nombreuses applications industrielles, médicales et environnementales grâce à leur capacité à convertir une grandeur physique (la pression) en un signal électrique exploitable. Dans le domaine du suivi géotechnique, ces capteurs sont essentiels pour surveiller l'intégrité structurelle d'infrastructures telles que les barrages, les ponts ou les conduites sous pression. Le MPX2200AP est un capteur de pression piézorésistif en silicium, conçu pour fournir une mesure précise et linéaire de la pression. Il convertit la pression appliquée en une tension de sortie proportionnelle, ce qui le rend idéal pour de nombreuses applications où la surveillance et le contrôle de la pression sont essentiels [2].

A cet effet et à travers ce projet nous allons réaliser et simuler un montage électronique qui nous permet l'étude d'un système électronique programmable basé sur le capteur de pression MPX2200AP, la carte Arduino UNO et le langage Python. Ce système permet d'acquérir, de traiter et de visualiser graphiquement des données de pression en fonction du volume ou de la profondeur dans un environnement expérimental contrôlé.

Contrairement à l'utilisation de protocoles standards comme Firmata, ce projet exploite principalement la bibliothèque PySerial en Python, qui permet une communication directe via le port série avec la carte Arduino. Cette méthode offre davantage de souplesse et de contrôle sur l'acquisition et le traitement des données. Le rôle de Python ne se limite donc pas à la réception des mesures, mais s'étend à l'analyse des relations physiques et à leur représentation graphique.

L objectif principale de ce projet est de :

- Étudier le comportement d'un système physique modélisé par un capteur de pression différentiel.
- Caractériser la réponse du capteur à travers des mesures expérimentales et des outils de programmation.
- Exploiter les bibliothèques Python, en particulier PySerial, pour la collecte et l'affichage des données en temps réel.

- Vérifier expérimentalement des lois physiques comme la loi de Boyle-Mariotte, le principe fondamental de la statique des fluides..

Pour atteindre cet objectif, ce travail est abordé selon les chapitres suivants :

- Le premier chapitre présente les notions fondamentales liées à la pression, ses unités, ses formes, ainsi qu'un aperçu des différents types de capteurs de pression (résistifs, capacitifs, inductifs...).
- Le deuxième chapitre est consacré à la carte Arduino UNO, à sa structure matérielle, à sa programmation, et à la communication série via PySerial.
- Le troisième chapitre développe la réalisation pratique du système : câblage électronique, codage (Arduino et Python) et exécution des expériences de mesure de la pression en fonction de la tension, du volume et de la profondeur.
- Le travail se termine par une conclusion générale, des remarques sur les résultats obtenus, ainsi que des perspectives pour des travaux futurs.

# Chapitre I

# Généralités sur les capteurs de pression et moyens de mesure

## I.1 Introduction

La mesure de la pression constitue l'une des fonctions fondamentales dans de nombreuses applications industrielles, médicales, environnementales et même domestiques. En tant que grandeur physique essentielle, la pression joue un rôle déterminant dans le contrôle et la surveillance des systèmes techniques tels que les conduites, les moteurs, les réservoirs de fluides ou encore les dispositifs biomédicaux. Pour mesurer cette grandeur avec précision, diverses méthodes et instruments spécialisés ont été développés, en particulier les capteurs de pression, qui ont considérablement évolué au fil du temps.

En effet, les capteurs de pression ne se limitent plus à convertir une variation de pression en un simple signal électrique. Grâce aux avancées technologiques, ils intègrent aujourd'hui des fonctionnalités avancées telles que le traitement du signal, la compensation thermique, la conversion numérique, ainsi que la communication avec des systèmes de contrôle intelligents. Ainsi, la compréhension des principes de mesure de la pression, des différentes formes de pression mesurables, ainsi que des divers types de capteurs disponibles, est devenue indispensable pour tout ingénieur, technicien ou chercheur opérant dans les domaines des systèmes embarqués, de l'automatisation ou de la maintenance industrielle.

Dans ce chapitre,nous allons citer quelques notions utiles dans le domaine de mesure de la pression. Pour cela nous commencerons par définir la notion de pression et ses différentes formes. Ensuite, nous présenterons un aperçu complet des capteurs de pression, en abordant leur principe de fonctionnement, leur classification et leurs caractéristiques techniques, dans le but d'offrir une vue d'ensemble des moyens de mesure utilisés dans les différents contextes d'application. Une citation des avantages et des inconvénients sera fournie.

### I.1.1 Définition de la pression :

Un corps liquide ou gazeux enfermé dans un récipient, qu'il remplit entièrement, exerce sur toutes les parois de celui-ci une force dite de pression. La pression est une grandeur dérivée du système international. Elle est définie comme le quotient d'une force par une surface :

$$P = \frac{F}{S} \qquad \text{(I.1)}$$

$P$ : pression en (1 Pa = 1N / $m^2$ ).

$F$ : force en Newton.

$S$ : surface en $m^2$.

La pression est souvent exprimée en bar (1bar = $10^5$ Pa).

Pour convertir facilement les unités de pression sans effectuer de calculs complexes ce diagramme permet de comparer rapidement les valeurs lorsque l'on travaille avec différents systèmes de pression comme montre la figure (I.1) [3]

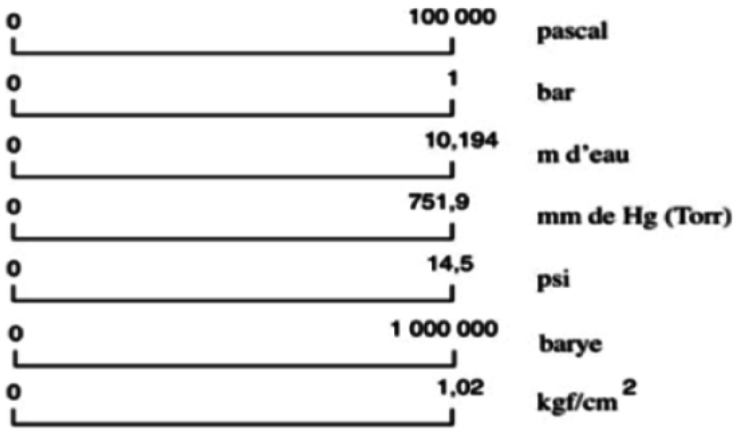



**Figure (I.1) :**unités de pression travaille avec différents systèmes de mesure

D'après le diagramme ?

- 1 bar = 100 000 pascals ≈ 10,194 mètres d'eau ≈ 751,9 mm Hg (Torr)
- 1 bar ≈ 14,5 psi (livres par pouce carré) = 1 000 000 baryes ≈ 1,02 kgf/cm²

## I.2 Les Différents types de pressions :

### I.2.1 La pression absolue :

Pression mesurée au-dessus du vide total ou de zéro absolu. Le zéro absolu représente une absence de pression.

Le vide : il correspond théoriquement à une pression absolue nulle. Il ne peut être atteint, ni même dépassé. Quand on s'en approche, on parle alors de vide poussé.

### I.2.2. La pression atmosphérique (ou barométrique) :

C'est la pression exercée par l'atmosphère de la terre. La pression atmosphérique au niveau de la mer est de 1.012 bar. Elle peut varier de +/- 25 mbar avec la pluie ou le beau temps. La valeur de la pression atmosphérique décroit lorsque l'altitude augmente [4]

### I.2.3. La pression relative :

C'est une pression au-dessus de la pression atmosphérique. Elle représente la différence positive entre la pression mesurée et la pression atmosphérique existante. C'est celle qui est le plus souvent utilisée, parce que la plupart des capteurs sont soumis à la pression atmosphérique et mesurent en relatif. Pour faire une mesure en absolu, il leur faut un vide poussé dans une chambre de référence (pression de gonflage d'un pneu par exemple) [5]

### I.2.4 Les capteurs différentiels :

C'est la différence de deux pressions où la différence de grandeur entre une valeur de pression donnée et une pression de référence donnée.

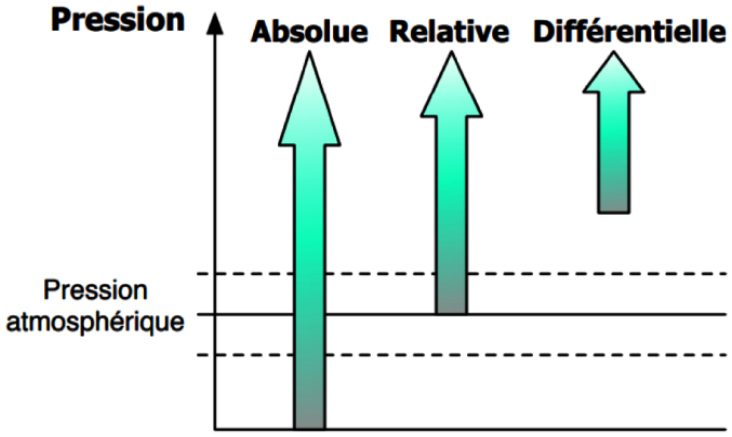



**Figure (I .2) :** Les différentes pressions existantes.

✓ **Pression absolue :**

- C'est la pression mesurée par rapport au vide absolu (zéro).
- Elle est composée de la pression atmosphérique plus la pression du système.
- Sur le schéma, la flèche part du bas (zéro absolu) jusqu'au-dessus de la pression atmosphérique.

✓ **Pression relative (ou manométrique) :**

- C'est la différence entre la pression absolue et la pression atmosphérique.
- Elle est mesurée par rapport à la pression atmosphérique, qui est considérée comme le point zéro.
- Sur le schéma, la flèche commence au niveau de la pression atmosphérique.

### I.2.5 La pression hydrostatique :

C'est la pression exercée au-dessous de la surface d'un liquide par le liquide situé au-dessus, quand le fluide est au repos. A l'intérieur d'une colonne de fluide se crée une pression due au poids de la mesure de fluide sur la surface considérée [6]

Cette pression est :

$$P= \rho .g .h \qquad (I.2)$$

Avec *r* masses volumiques du fluide

### I.2.6 La pression hydrodynamique :

Elle résulte de la vitesse du fluide en mouvement. Un fluide qui se déplace crée une pression supplémentaire [7]:

$$P= \frac{1}{2} P v^2 \qquad (I.3)$$

Avec : *v* est la vitesse de déplacement du fluide en m/s.

### I.3.1 capteur :

Au cours de ces dernières années, l'automatisation, le contrôle et la surveillance de la plupart des processus ont induit un besoin croissant en capteurs. En effet, dans la plupart des applications, le manque de capteurs adéquats et d'actionneurs pour coupler l'électronique de contrôle avec l'environnement extérieur est le principal problème dans le développement de nouveaux systèmes. Des recherches, mettant à profit les progrès de la microélectronique et d'autres techniques compatibles, ont permis d'une part de réaliser, à faible coût, des capteurs et des actionneurs

miniaturisés et de hautes performances, et d'autre part d'élargir le spectre d'utilisation des capteurs[8].

Ces derniers sont largement utilisés dans plusieurs domaines santé, automobile, électromagnétisme etc…, et sont employés dans la vie de tous les jours pour transformer des événements mécaniques, chimiques ou thermiques en signal électrique. Ils sont utilisés comme systèmes de détections de grandeur physiques qui relèvent la technologie microélectronique, et plus particulièrement celle du silicium, et offrent des avantages techniques et économiques. Les détections piézorésistives et résistif sont parmi les principes de détection les plus utilisées.

Le capteur est considéré comme le premier élément d'une chaîne de mesure (mesurande : *m*). Il a pour fonction essentielle de traduire une grandeur physique à mesurer, en une autre grandeur exploitable généralement électrique s. Qui peut être une impédance *R, L, C* on parle dans ce cas de capteurs passifs ou une charge, un courant ou une différence de potentiel ddp on parle ici de capteurs actifs, la fonction S=F(m) dépend souvent d'autres grandeurs physiques propres à l'environnement température, humidité, etc.. Ces grandeurs sont appelées grandeurs d'influence [9].

En général, le signal délivré par le capteur n'est pas directement utilisable et à besoin d'être amplifié et adapté. L'ensemble des circuits et appareils qui assurent ces opérations est appelé chaîne de mesure et du traitement du signal. Plusieurs paramètres entrent en jeu pour classifier les capteurs des uns des autres. Le tableau (I.1) donne les principales caractéristiques de ces paramètres.

**Tableau (I.1)** : Principales caractéristiques des capteurs.

| **Paramètres (Grandeurs)** | **Définitions** |
|---|---|
| Sensibilité et la Linéarité | La sensibilité « S » d'un capteur est la variation de la mesure « s » prélevée pour une variation du mesurande « m » : S= $\frac{\partial s}{\partial m}$. Pour que la sensibilité soit indépendante de m (constante), il faut que le capteur soit linéaire : s = S . m + $s_0$ Généralement, on peut toujours définir une plage de valeurs de m pour lesquels *S* est constante. |
| La précision | C'est la capacité d'un instrument de mesure de donner la même valeur lorsque la mesure est répétée plusieurs fois sous les mêmes conditions. Autrement dit, si plusieurs mesures sont effectuées, l'écart entre les résultats est très faible. |
| L'incertitude | C'est la marge d'erreur ΔR entre la valeur donnée par l'appareil de mesure et la vraie valeur du mesurande. L'erreur relative donne une meilleure appréciation sur l'incertitude d'un appareil de mesure. |
| Le temps de réponse | C'est l'aptitude d'un capteur à répondre aux variations du mesurande avec le temps. |
| L'étendu de la mesure | Elle définit la zone dans laquelle les caractéristiques du capteur sont assurées par rapport à des spécifications données. On peut classer cette zone en trois familles :<br>▪ Zone nominale d'emploi : zone dans laquelle le mesurande peut évoluer sans modification des caractéristiques du capteur.<br>▪ Zone de non détérioration : c'est une zone définie par des valeurs limites des grandeurs influençant le capteur (mesurande, température environnante, etc...) sans que les caractéristiques du capteur ne soient modifiées après annulation de surcharges éventuelles.<br>▪ Zone de détérioration : dans laquelle, le capteur peut y avoir des modifications permanentes des caractéristiques. |
| Le seuil | Le seuil d'un capteur est la valeur minimum du mesurande à partir de laquelle le capteur devient sensible. En dessous de cette valeur le capteur ne mesure rien et il affiche zéro. |
| L'hystérésis | C'est la non coïncidence entre la courbe de charge et celle de décharge. |
| Finesse | C'est la qualité d'un capteur à ne pas venir modifier, par sa présence, la grandeur à mesurer. Cela permet d'évaluer l'influence du capteur sur la mesure. On la définit non seulement vis à vis du capteur mais aussi vis à vis de l'environnement d'utilisation du capteur. |
| Résolution | C'est la plus petite variation du mesurande que le capteur est susceptible de déceler. |
| Rapidité | C'est la qualité d'un capteur à suivre les variations du mesurande. On peut la chiffrer de plusieurs manières. |
| Etalonnage des capteurs | C'est l'opération qui établit la relation entre le mesurande et la grandeur électrique de sortie. Cette relation peut dépendre non seulement du mesurande mais aussi des grandeurs d'influence. S' il n y a pas de grandeurs d'influence, l'étalonnage est simple, sinon il est multiple. |

### I.3.2. Les capteurs de pression :

Un capteur de pression est un système constitué de deux parties : une partie de détection, appelée Cellule sensible et une partie traitement de l'information appelée Circuit électronique de traitement ou encore Circuit conditionneur Un capteur de pression peut donc être représenté par le schéma de la figure ci-dessous [10]

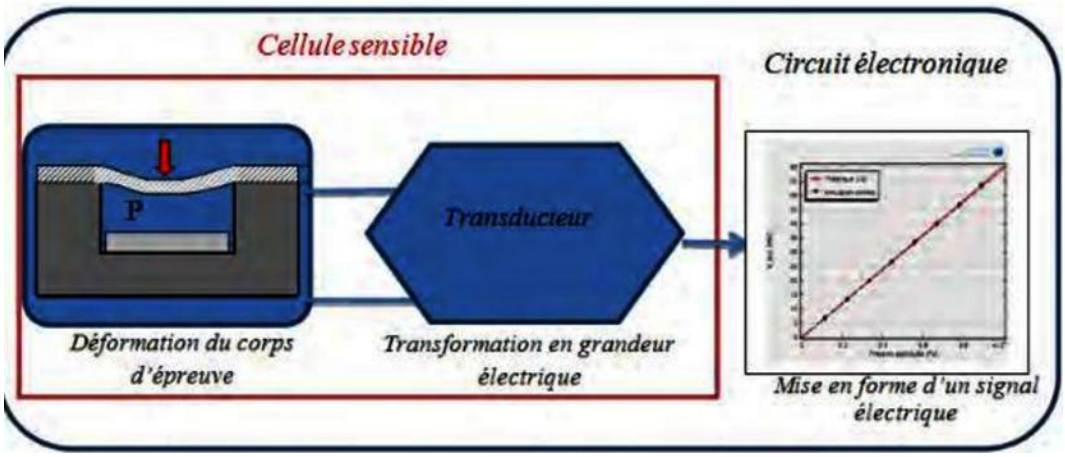



**Figure (I.3):** Schéma synoptique d'un capteur de pression

Le corps d'épreuve est l'élément mécanique qui soumis aux variations de la grandeur à mesurer (mesurande) a pour rôle de la transformer en une grandeur physique mesurable. Pour les capteurs de pression cette grandeur est généralement une déformation, les corps d'épreuve les plus utilisés sont les plaques, les poutres et les membranes. Le transducteur est l'élément sensible qui, lié au corps d'épreuve, traduit les réactions de ce dernier en signal électrique. Le module électronique est le module de traitement du signal en vue d'une éventuelle exploitation

Dans tous les cas, les capteurs de pression peuvent se ramener au schéma synoptique.

Figure (1.4)

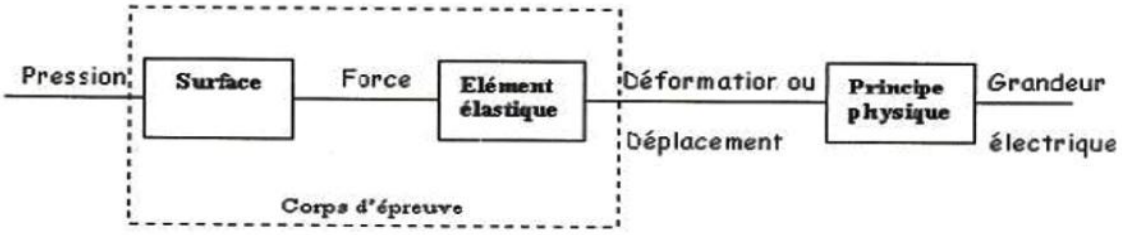



**Figure (I.4) :** Schéma synoptique d'un capteur

Le corps d'épreuve est l'élément mécanique qui soumis aux variations de la grandeur à mesurer a pour rôle de transformer celle-ci en grandeur physique mesurable.

## I.4 Les capteurs différentiels :

### I.4.1 capteurs résistifs :

Le capteur de pression résistif est un dérivé de la technologie à film épais, utilisant la variation de la résistivité des matériaux.

Les changements de contrainte résultant de l'application de charges extérieures sur la membrane déformable sont transformés en changement de résistance dans le réseau de jauges de contrainte. Créant ainsi un déséquilibre du pont. Ce déséquilibre produit un signal émanant de l'autre diagonale du pont de Wheatstone, proportionnellement à la contrainte appliquée .

Aujourd'hui, avec plus de 20 ans d'expérience dans l'industrie, Interlien Electronics continue d'innover en concevant et en fabriquant des capteurs résistifs pour une gamme complète d'applications telles que les appareils industriels, militaires, électroniques grand public, mobiles, médicaux et de pointage.

L'une des premières utilisations de cette technologie brevetée Force Sensing Resistor® (FSR) en couches minces concernait les tambours électroniques et autres instruments de musique. Les téléphones mobiles, les lecteurs multimédias portables, les appareils de navigation, les jeux portables, les appareils photo numériques et autres appareils électroniques portables ne sont que quelques-uns des appareils qui utilisent la technologie FSR [4].

**Figure (I.5):** capteurs résistifs

### I.4.2 Les avantages :

- Signal de sortie élevé.
- Utilisable sans conditionneur.

### I.4.3 Les inconvénients :

- Durée de vie
- Sensibilité aux vibrations.

### I.4.4 Utilisations des capteurs résistifs:

- Industrie : Surveillance de la pression dans les systèmes hydrauliques et pneumatiques.
- Automobile : Mesure de la pression d'huile, de carburant et des systèmes de freinage.
- Médical : Tensiomètres électroniques, respirateurs, surveillance des patients.
- Laboratoires : Mesures précises de pression pour les expériences scientifiques.
- Bâtiment : Contrôle des contraintes dans les ponts, barrages et structures en béton.

### I.4.5 Capteur capacitif :

Une solution élégante imaginée par certains constructeurs a été de transformer la déformation de la membrane sous l'effet d'une pression (ou d'une force) en une variation de capacité plutôt qu'une variation de résistance. En effet, il suffit de placer l'une des armatures d'un

condensateur sur la membrane qui se déforme et l'autre sur une pièce solidaire du corps d'épreuve, mais non soumise à la déformation, comme le montre le schéma ci-dessous, pour réaliser un condensateur plan dont la capacité est en relation directe avec la pression appliqué [11]

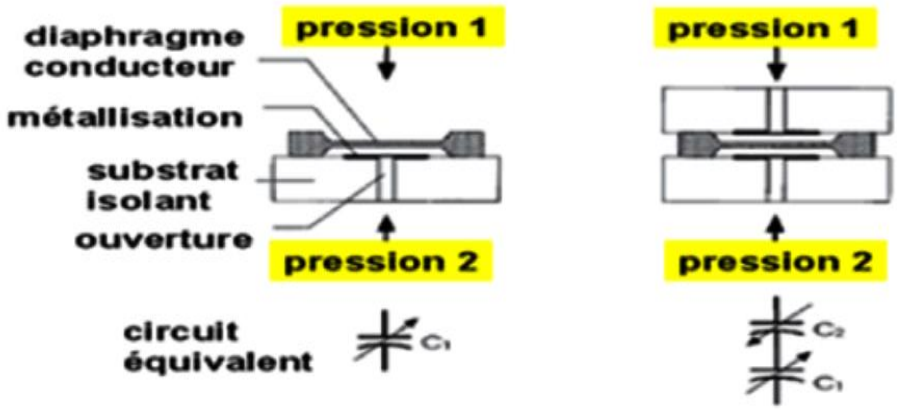



**Figure (I.6) :** Principe de fonctionnement d'un capteur de pression capacitif différentiel à deux condensateurs [7].

Il est clair qu'on peut imaginer des géométries d'armatures permettant d'obtenir la meilleure linéarité possible entre variation de capacité et variation de pression, que l'on peut mettre en œuvre simultanément plusieurs condensateurs et donc faire un montage en pont peu sensible aux contraintes thermiques. L'intérêt habituel du montage capacitif se retrouve évidemment dans cette application, à savoir qu'on intégrera généralement ce condensateur variable dans un circuit oscillant et qu'en conséquence la mesure de pression se ramènera à une mesure de fréquence et l'opportunité de disposer de deux oscillateurs semblables. L'un de fréquence fixe et l'autre variant avec la pression, dont on exploite via un mélangeur la différence des fréquences permet évidemment une très grande précision puisqu'on réduit ainsi par soustraction l'importance des dérives éventuelle de chaque oscillateur pris séparément [5].

### I.4.6 Les avantages :

- Faible masse.
- Peu sensible aux accélérations.

### I.4.7 Les inconvénients :

- Sensibilité à la température.
- Sortie haute impédance.

### I.4.8 Utilisations des capteurs capacitif :

1. Écrans tactiles : smartphones, tablettes.
2. Boutons tactiles : électroménager, claviers sans contact.
3. Industrie : détection d'objets (solides ou liquides) sans contact.
4. Automobile : détection de présence, commandes tactiles.
5. Domaine médical : surveillance sans électrodes directes.
6. Domotique : contrôle d'éclairage ou d'appareils par simple toucher ou proximité.

**Figure (I.7) :** capteur de pression capacitif [6].

### I.4.9 Capteurs inductifs:

Un bobinage de fils conducteurs parcouru par un courant électrique, crée un champ magnétique B, on peut canaliser les lignes de champs on ajoutant un circuit magnétique [8].

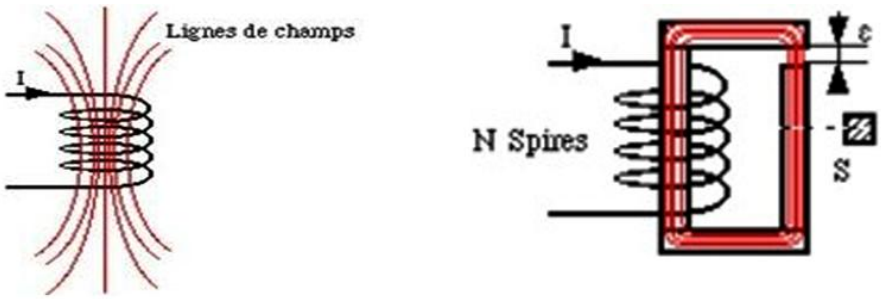



**Figure (I.8) :** Principe de fonctionnement d'un capteur de pression capacitif différentiel à deux condensateurs

### I.4.10 Principe de fonctionnement :

Un noyau magnétique se déplace à l'intérieur d'une bobine, ce déplacement entraine une variation de l'inductance la bobine.

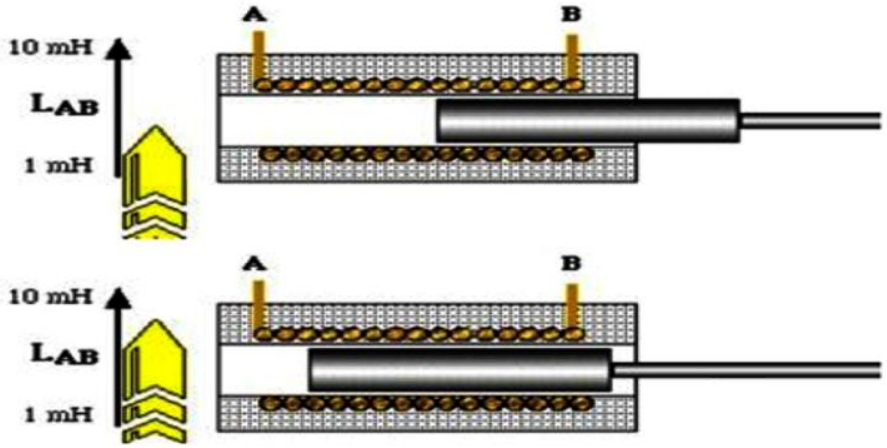



**Figure (I.9) :** Effet de la position du noyau en fer sur l'inductance d'une bobine [9].

### I.4.11 Les avantages :

- Faible hystérésis.
- Très bonne résolution

### I.4.12 Les inconvénients :

- Sensible aux chocs et aux vibrations.

## I.5 Étude d'un capteur de pression à inductance variable :

### I.5.1 Principe général :

Nous allons utiliser un capteur comportant non seulement un seul bobinage mais deux symétriques par rapport à la position de référence d'un noyau ferromagnétique, ce dernier est déplacé par l'action d'un piston, de géométrie plus ou moins complexe, chacune des inductances varie en sens inverse, en fonction de d + x pour l'une et d – x pour l'autre, on pourra donc obtenir via une connexion électrique adéquate, une différence de potentiel en fonction du déplacement entraîné par les deux pressions $P_1$ et $P_2$ [7].

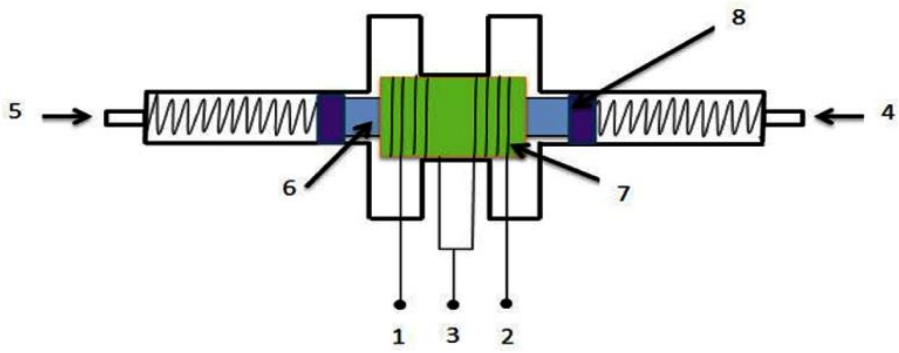



**Figure (I.10) :** Capteur à inductance variable .

1 : Entré de la bobine B1.

2 : Entré de la bobine B2.

3 : Point milieu des deux bobines B1 et B2.

4 : Arrivée d'air (pression P1).

5 : Arrivée d'air (pression P2).

6 : Noyau magnétique mobile.

7 : bobine B1 en série avec la bobine B2 .

8 : Piston.

L'inductance étant directement en fonction de L1 (l'inductance de la bobine B1) et L2 (l'inductance de la bobine B2) est liée au déplacement relatif, il en résulte une variation

d'impédance qu'on va l'exploiter dans un dispositif oscillant aussi bien que dans un simple montage en tension.la figure ci-dessous (Figure 8) montre un exemple de réalisation pratique du cœur du capteur. Le noyau magnétique se déplace à l'intérieur des deux bobines, se déplacement entraine une variation des inductances B1 et B2. Notons que ce déplacement peut être provoqué par une pression mais aussi par une force ou même un déplacement quelconque d'une pièce. C'est donc une structure utilisable aussi bien comme capteur de déplacement, de force ou de pression [4].

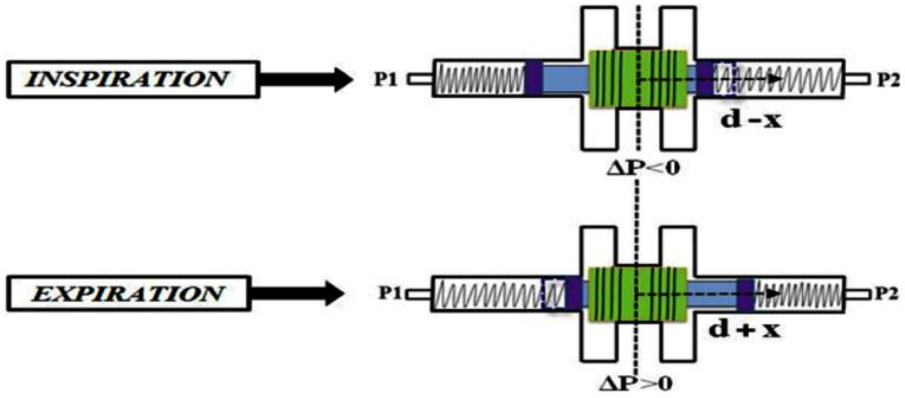



**Figure (I .10) :** Principe d'un capteur à inductance variable

La différence de pression appliquée sur chacun des faces va entraîner un déplacement au niveau des pistons (fixé d'un côté et de l'autre d'un noyau) qui assurent la variation de l'inductance des bobines.

### I.5.2 Utilisations des capteurs inductifs :

- Industrie : détection d'objets métalliques sur les chaînes de production.
- Automobile : mesure de la vitesse (ABS), position du moteur ou des roues.
- Robotique : positionnement des bras ou des pièces métalliques.
- Ascenseurs : localisation de la cabine.
- Sécurité : capteurs de fin de course, détection d'ouverture/fermeture de portes métalliques.

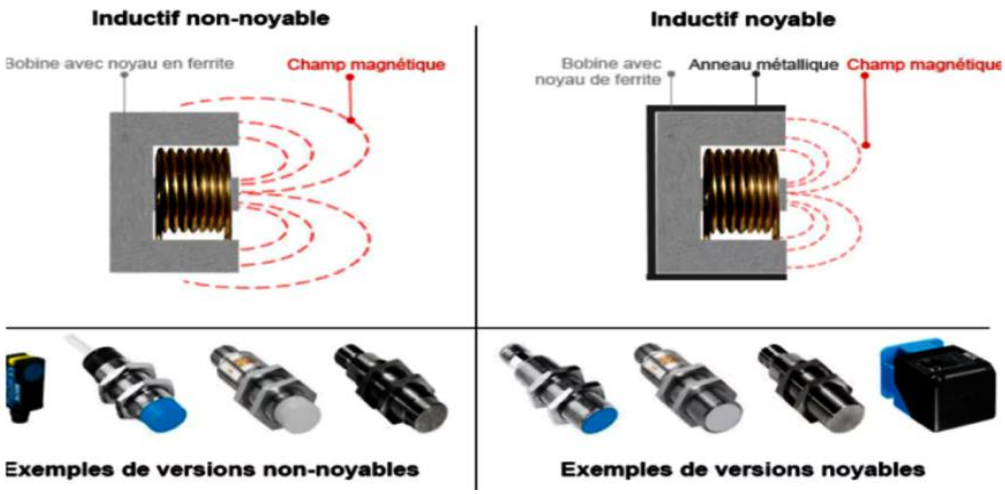



**Figure (I.11) :** Capteur inductifs avec et sans version noyable [10]

### I.5.3 Les capteurs piézorésistifs :

Les capteurs piézorésistifs forment la majorité des capteurs de pression disponibles sur le marché. Ces composants, extrêmement sensibles à la température, ne peuvent fonctionner correctement sans l'ajout de circuits de compensation coûteux.

Ce type de capteurs qui est certainement le moins connu se caractérise par une excellente linéarité et une faible impédance de sortie. Leur sensibilité à la température est par contre problématique.

On utilise les matériaux piézoélectriques, parce qu'ils permettent de convertir une contrainte mécanique en polarisation électrique donc en tension. Les matériaux piézorésistifs possèdent la particularité de convertir une contrainte mécanique en une variation de résistance ce qui leurs permet d'être utilisées comme capteurs de pression ou d'accélération [12]

Les jauges de contraintes sont des piézorésistances c'est-à-dire des résistances dont la valeur varie selon leur déformation. Lorsque la membrane se déforme, les jauges sont soit comprimées, soit étirées, en fonction de leur position sous la membrane.

L'utilisation des capteurs à base de silicium est largement répandue et permet une bonne intégration des jauges de déformation avec les circuits bipolaires ou CMOS [13]

Une contrainte appliquée sur du silicium va modifier sa conductivité pour deux raisons : sa variation géométrique mais aussi sur la conductibilité intrinsèque du matériau. Il en résulte une

amplitude bien plus importante que pour des capteurs métalliques (Smith 1954). Cela a permis une grande gamme d'utilisation de la piézorésistance. Beaucoup d'appareils commerciaux comme les capteurs d'accélération utilisent des capteurs en silicium.

Généralement, le capteur de pression piézorésistif est constitué d'une membrane en silicium sur laquelle sont diffusées ou implantées des piézorésistances. Cette membrane est susceptible de se déformer sous l'action d'une pression différentielle P appliquée. La variation de la pression engendre des contraintes internes faisant varier valeurs des piézorésistances [10].

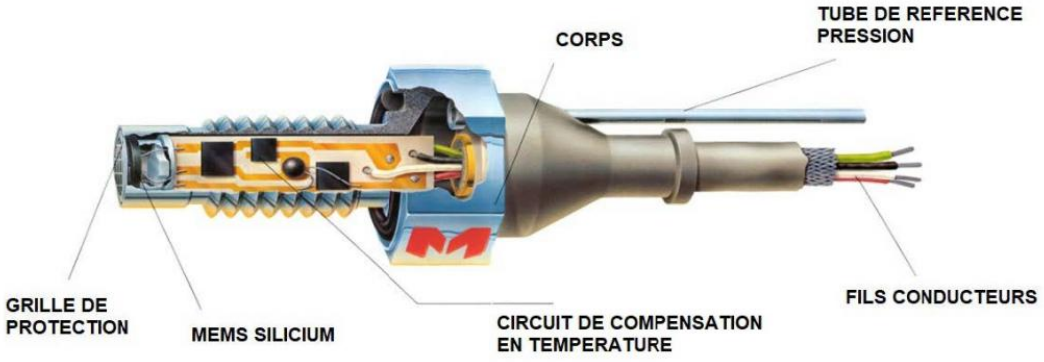



**Figure (I. 12) :** Schéma synoptique de fonctionnement du capteur de pression piezorésistif.

Malgré les défauts des capteurs de pression au silicium à détection piézorésistive ils restent parmi les microsystèmes les plus vendus dans le monde.

Les qualités mécaniques du silicium et l'usinage collectif relativement simple, qu'il soit chimique (KOH...) ou physique (gravure plasma, micro-forage), rendent aujourd'hui possible l'industrialisation de certaines microstructures tels que les microcapteurs et microsystèmes visant les marchés de grand volume et de faible coût. La dualité faible taille et faible coût peut permettre de répondre à de nouveaux besoins dans plusieurs domaines notamment la médecine, l'automobile, l'industrie etc. Plusieurs travaux de recherches ont été effectués afin d'améliorer leurs performances et de les adapter aux nouvelles exigences industrielles. Parmi ces travaux on peut citer la conception et la simulation de capteur de pression piézorésistif pour la mesure de la pression sanguine [14].

Une autre application récente de ces dispositifs à détection piézorésistive, développée au LAAS, consiste en la conception d'un capteur de pression pour la mesure de la pression intra crânienne.

### I.5.4 Caractéristiques :

- Très sensible aux petites déformations
- Réponse rapide et linéaire
- Peut être intégré facilement dans des circuits électroniques (MEMS)

### I.5.5Avantages :

- Haute précision et sensibilité
- Taille réduite (miniaturisation facile)
- Intégrable sur silicium (compatible avec les technologies MEMS)

### I.5.6 Inconvénients :

- Sensibles aux variations de température
- Nécessitent souvent un amplificateur pour le signal .

### I.5.7 Utilisations des capteurs piézorésistifs :

- Médical : mesure de la pression artérielle (tensiomètres).
- Automobile : surveillance de la pression des pneus (TPMS), capteurs d'airbags.
- Industrie : mesure de pression ou de force dans les machines.
- Microsystèmes (MEMS) : détection de très faibles pressions.
- Textiles intelligents : semelles connectées, vêtements avec capteurs de pression.

**Figure (I.13) :** Appareils commerciaux d'un capteurs piézorésistifs [15].

## I.6 Les microsystèmes électromécaniques (MEMS) :

### I.6.1 Définitions et caractéristiques générales :

Les MEMS, acronyme de Micro Electro Systèmes mécaniques sont des dispositifs miniaturisés combinant plusieurs principes physiques. Ils intègrent généralement des éléments mécaniques couplés à de l'électronique et sont réalisés par des procédés de fabrication issus de la micro-électronique.

Les MEMS exploitent, entre autres, des effets liés à l'électromagnétisme, thermique et fluidique. Ils sont dans notre quotidien, au cœur de la téléphonie, de l'automobile, du médical, des chaînes de production ou des manettes de consoles de jeux. Leur taille est de l'ordre du millimètre carré et les éléments de leurs structures (mécaniques) sont à l'échelle du micron. Utilisé en tant que capteur, un MEMS possède une partie mobile sensible à la variation d'une grandeur physique (vitesse, pression, direction ...). Cette variation est alors traduite en une grandeur électrique, analysée ensuite par la partie électronique du MEMS. Il possède parfois un micro-actionneur intégré qui, à partir d'un signal électrique, va agir sur la partie mécanique[16].

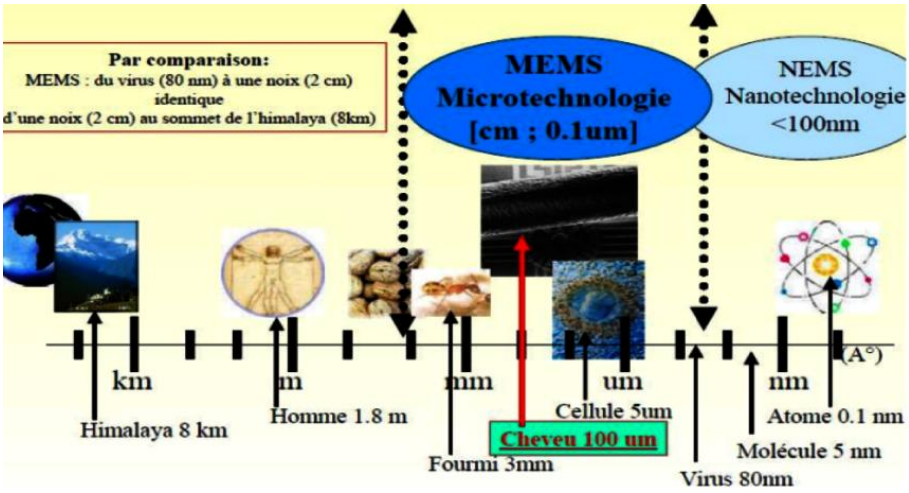



**Figure (I.14) :** Ordre de grandeur des microsystèmes

Les MEMS sont des systèmes intégrant des composants mécaniques et électroniques à l'échelle microscopique, typiquement dans une plage de taille allant de 0,1 µm à quelques centimètres. Cela est illustré par le diamètre d'un cheveu humain (~100 µm), situé dans cette gamme. En comparaison, les NEMS fonctionnent à des dimensions inférieures à 100 nanomètres, c'est-à-dire à l'échelle des virus, molécules et atomes [10].

Ces technologies sont essentielles dans de nombreux domaines comme les capteurs, la biomédecine, l'aérospatiale et l'électronique de grand public

### I.6.2 Les différents domaines d'application des MEMS :

Les avancées autour de la micro-technologie de réalisations des MEMS (basées sur le principe de la photolithographie) ont engendré une explosion des applications et une segmentation du domaine. On trouve quatre familles associées à leurs cadres applicatifs Figure (I.15)

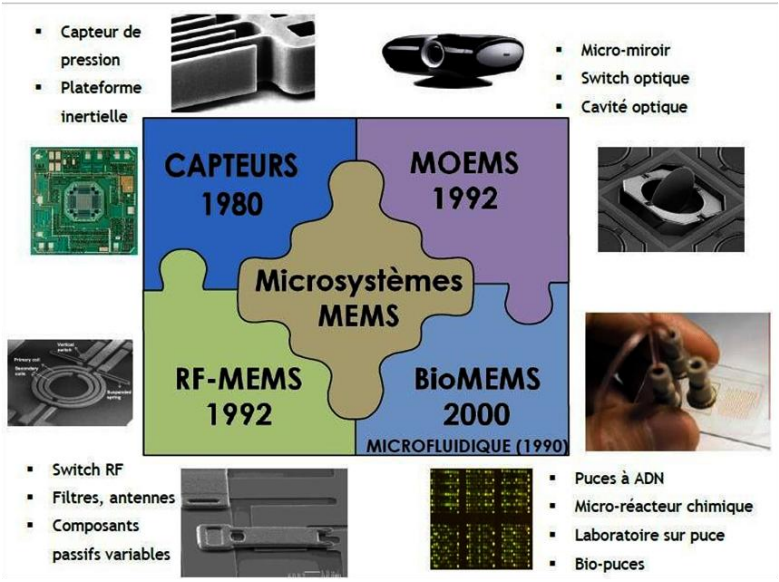



**Figure (I.15) :** Les domaines d'application des MEMS [17].

### I.6.2.1 Exemple d'une chaîne de mesure d'un capteur de pression :

En général une mesure de pression utilise une membrane déformable comme corps d'épreuve. Dans le cadre des MEMS, celle-ci est gravée dans le substrat de silicium.


**Figure (I.16):** Production collective de MEMS qui seront séparés en toute fin du précédé Visualisation de la cavité réalisée pour obtenir la membrane en silicium [18].

La membrane du capteur de pression subit une déformation en présence d'une différence de pression :

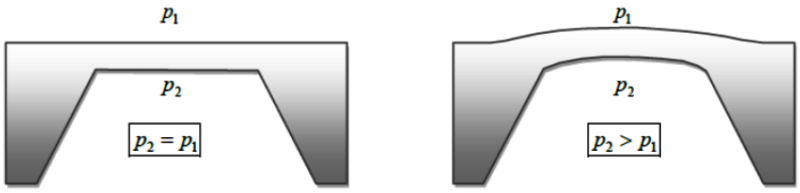



**Figure (I.17)** : Schéma de la déformation de la membrane d'un capteur de pression

Afin d'avoir une mesure de déformation, les micro-technologies permettent l'intégration de jauges piézorésistives directement implantées dans la structure des MEMS au cours de la production. La jauge de contraintes voit sa longueur relative (ΔL/L) varier lors de la déformation, entrainant la variation relative de sa résistance (ΔR /R) [19]:

$$\frac{\Delta R}{R} = K \frac{\Delta l}{l} \qquad \text{(I.5)}$$

Avec ; K facteur de jauge

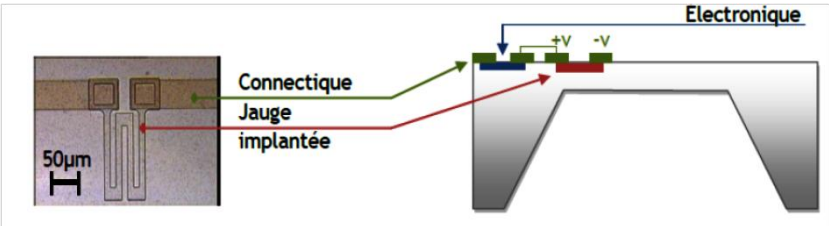



**Figure (I.18)** : Schéma de l'implantation intégrée d'une jauge piézorésistive

L'électronique de mesure associée est le pont de Wheatstone. Une variation de valeur d'une des résistances (jauge piézorésistive) du montage en pont, fait varier la mesure de sortie de la tension. La piézorésistivité des matériaux constituants les jauges de contrainte est la propriété de variation de conductivité sous l'effet d'une déformation mécanique [20].

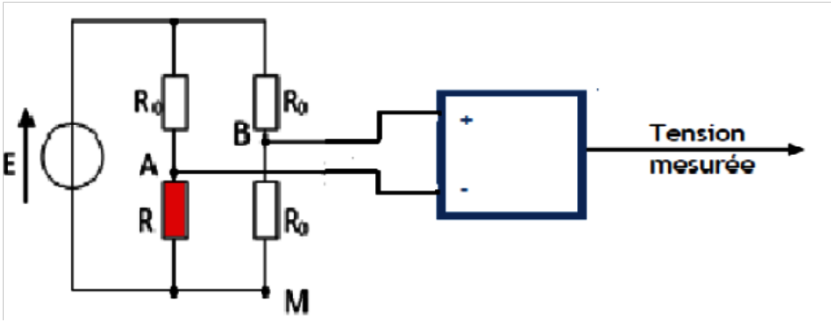



**Figure (I.19)** : Schéma du pont de Wheatstone associée au capteur.

## I.7 Définition du capteur intelligent de pression :

Le capteur intelligent de pression est un dispositif électronique intégré conçu pour mesurer la pression physique des gaz ou des liquides, tout en intégrant des fonctionnalités avancées de traitement du signal. Contrairement aux capteurs classiques qui se contentent de générer un signal analogique brut, le capteur intelligent est capable de traiter le signal en interne, de le corriger, de le compenser face aux variations environnementales (comme la température), et de fournir une sortie numérique fiable et précise [21].

Ce type de capteur représente une évolution technologique majeure, car il combine plusieurs fonctions dans un seul composant, notamment : l'auto-étalonnage, la conversion analogique-numérique, la détection des erreurs, la compensation thermique, ainsi que la communication avec des systèmes externes via des interfaces numériques ($I^2C$, SPI) ou sans fil (Bluetooth, ZigBee, LoRa). Il est devenu un élément clé dans les systèmes de mesure intelligents, en particulier dans les domaines industriel, médical, automobile et aéronautique [13].

### I.7.1Composants d'un capteur intelligent de pression :

Un capteur intelligent de pression est généralement constitué des éléments suivants :

1) Élément de détection primaire : capteur piézorésistif, capacitif ou MEMS, qui convertit la variation de pression en un signal électrique.
2) Unité de traitement interne (Microcontrôleur ou ASIC) : elle filtre, corrige et numérise le signal tout en appliquant des algorithmes de compensation thermique.
3) Interface de communication : permet l'échange de données avec d'autres systèmes via des protocoles numériques (SPI, $I^2C$, UART) ou sans fil.
4) Système d'alimentation : alimente les circuits internes de manière stable.

- Intégration facile dans les systèmes embarqués et les réseaux IoT.

### I.7.2 Inconvénients du capteur intelligent de pression :

- Coût élevé : En raison de l'intégration de composants électroniques avancés (microcontrôleur, interface de communication, algorithmes de traitement), ces capteurs sont souvent plus coûteux que les capteurs traditionnels.
- Complexité technique : L'installation, la configuration et le diagnostic d'un capteur intelligent nécessitent des compétences techniques spécifiques, notamment en électronique et en programmation embarquée.
- Consommation d'énergie accrue : La présence de circuits de traitement et de communication numériques augmente la consommation énergétique, ce qui peut être un désavantage dans les systèmes à alimentation limitée (batterie).
- Dépendance au logiciel intégré : Le fonctionnement du capteur repose en partie sur un firmware interne. Une erreur logicielle ou un bug peut affecter la précision des mesures ou même entraîner une panne.
- Sensibilité aux interférences électromagnétiques : Dans des environnements industriels fortement perturbés, les composants électroniques sensibles peuvent être influencés par des interférences, même si des filtres sont souvent intégrés.
- Maintenance plus complexe : En cas de dysfonctionnement, l'analyse du problème peut nécessiter des outils spécialisés ou des interfaces de diagnostic, ce qui rend la maintenance plus difficile et coûteuse.
- Incompatibilité potentielle: Certains capteurs intelligents peuvent ne pas être immédiatement compatibles avec toutes les architectures matérielles ou protocoles utilisés dans un système existant.

### I.7.3Avantages du capteur intelligent de pression :

- Haute précision de mesure grâce au traitement embarqué.
- Stabilité à long terme avec réduction de la nécessité d'un étalonnage fréquent.
- Résistance au bruit et aux interférences électromagnétiques.
- Capacité de diagnostic automatique en cas de dysfonctionnement.
- Intégration facile dans les systèmes embarqués et les réseaux IoT.

### I.7.4Applications typiques:

- Industrie : surveillance de la pression dans les tuyauteries, réservoirs et systèmes hydrauliques.
- Médecine : capteurs de pression artérielle, respirateurs, dispositifs de surveillance.
- Automobile : systèmes TPMS (pression des pneus), capteurs de pression d'huile ou de carburant.
- Aéronautique et spatial : mesure de l'altitude, contrôle de la pression en cabine.

**Figure (I.20) :** capteur intelligent de pression [7].

## I.8 Conclusion

Dans ce chapitre, nous avons présenté des notions théoriques et une compréhension approfondie sur les capteurs de pression, ainsi des méthodes de mesure associées qui s'impose comme une étape préalable essentielle pour aborder de manière rigoureuse assavoir : l'étude des différents types de pression et des technologies de détection disponibles, l'analyse des notions fondamentales de pression, à sa classification selon le contexte de mesure, puis à l'exploration détaillée des capteurs avancés utilisés pour sa détection, en mettant en lumière leurs principes de fonctionnement, leurs avantages et leurs limites.

# Chapitre II:

# Dispositif programmable Arduino et le protocole de communication Firmata

# Chapitre II:Dispositif programmable Arduino et le protocole de communication Firmata

## II.1 Introduction

Le dispositif programmable Arduino est une plateforme matérielle open-source conçue pour simplifier le développement et le contrôle de systèmes électroniques embarqués. Composée d'une carte microcontrôleur programmable et d'un environnement de développement dédié, Arduino permet aux utilisateurs, qu'ils soient débutants ou experts, de concevoir rapidement des applications intégrant des capteurs, des actionneurs et diverses interfaces physiques. Sa modularité et son accessibilité ont fait d'Arduino un outil incontournable pour l'éducation, le prototypage et les projets industriels légers.

Le protocole de communication Firmata complète cette plateforme en offrant un standard permettant à un ordinateur hôte de contrôler à distance les fonctions d'un microcontrôleur Arduino via une liaison série. Firmata établit une couche d'abstraction qui facilite la gestion des entrées/sorties numériques et analogiques, la modulation de largeur d'impulsion, et la communication avec des périphériques comme les servomoteurs ou les bus I2C et SPI. Cette approche simplifie le développement logiciel en déchargeant le programmeur des détails bas niveau du matériel, rendant ainsi l'interaction entre logiciel et matériel plus directe et efficace.

En associant Arduino et Firmata, il devient possible de concevoir et de piloter des systèmes électroniques complexes avec une grande flexibilité, tout en réduisant considérablement la complexité technique nécessaire au contrôle matériel.

Dans ce chapitre, nous montrons qu'Arduino et les protocoles Firmata permettent une communication sans effort entre le logiciel et le matériel, permettant aux utilisateurs de construire et de gérer un large éventail de projets électriques avec simplicité et adaptabilité.

## II.2 Arduino : Définition et modèles principaux

Arduino est une plateforme matérielle open-source basée sur des cartes microcontrôleurs programmables, conçue pour simplifier le prototypage et le contrôle de systèmes électroniques. Depuis sa création, Arduino s'est imposé comme un standard dans le domaine de l'électronique embarquée, grâce à son architecture accessible et modulaire, ainsi qu'à son environnement de développement intégré (IDE) convivial [22].

Parmi les modèles les plus utilisés, on distingue :

- Arduino Uno : équipé d'un microcontrôleur ATmega328P fonctionnant à 16 MHz, il dispose de 14 broches d'entrée/sortie numériques (dont 6 capables de PWM), 6 entrées

analogiques, 32 Ko de mémoire flash et fonctionne sous une tension de 5 V. Ce modèle est largement utilisé pour les projets éducatifs et les applications générales [23].

- Arduino Mega 2560 : doté d'un ATmega2560 avec une fréquence similaire, il offre un grand nombre de broches (54 numériques, 16 analogiques) et une mémoire importante (256 Ko flash). Il est adapté aux projets complexes nécessitant de nombreuses interfaces matérielles [24].
- Arduino Leonardo : basé sur l'ATmega32U4, il permet la communication USB native, autorisant le contrôle en tant que périphérique HID (clavier, souris), ce qui ouvre des possibilités spécifiques en interfaces utilisateur [25].
- Arduino Due : équipé d'un microcontrôleur ARM Cortex-M3 (Atmel SAM3X8E) à 84 MHz, fonctionnant en 3.3 V, il offre une puissance de calcul accrue et un nombre élevé d'entrées/sorties, adapté aux applications avancées [26].
- Arduino Nano 33 BLE : microcontrôleur nRF52840 Cortex-M4 cadencé à 64 MHz, avec connectivité Bluetooth Low Energy intégrée, adapté aux projets IoT et sans fil [27].

### II.2.1 Les gammes de cartes Arduino

Arduino propose plusieurs modèles de cartes microcontrôleurs adaptées à une large variété d'applications, allant des projets simples aux systèmes embarqués complexes. Ces cartes se distinguent principalement par leur microcontrôleur, leur nombre d'entrées/sorties, leur mémoire et leurs fonctionnalités spécifiques [28].

#### II.2.1.1 Arduino Uno

- Microcontrôleur : ATmega328P
- Fréquence : 16 MHz
- Entrées/Sorties : 14 numériques (6 PWM), 6 analogiques
- Mémoire : 32 Ko Flash, 2 Ko SRAM
- Tension de fonctionnement : 5 V
- Usage typique : apprentissage, prototypage basique
- Remarque : carte la plus populaire et polyvalente, idéale pour débutants

#### II.2.1.2 Arduino Mega 2560

- Microcontrôleur : ATmega2560
- Fréquence : 16 MHz
- Entrées/Sorties : 54 numériques (15 PWM), 16 analogiques
- Mémoire : 256 Ko Flash, 8 Ko SRAM
- Tension de fonctionnement : 5 V

- Usage typique : projets complexes nécessitant beaucoup d'E/S

### II.2.1.3 Arduino Leonardo

- Microcontrôleur : ATmega32U4
- Fréquence : 16 MHz
- Entrées/Sorties : 20 numériques (7 PWM), 12 analogiques
- Mémoire : 32 Ko Flash, 2.5 Ko SRAM
- Tension de fonctionnement : 5 V
- Usage typique : interfaces HID (clavier, souris)

### II.2.14 Arduino Due

- Microcontrôleur : ARM Cortex-M3 (Atmel SAM3X8E)
- Fréquence : 84 MHz
- Entrées/Sorties : 54 numériques (12 PWM), 12 analogiques
- Mémoire : 512 Ko Flash, 96 Ko SRAM
- Tension de fonctionnement : 3.3 V
- Usage typique : applications avancées nécessitant plus de puissance de calcul

### II.2.1.5 Arduino Nano

- Microcontrôleur : ATmega328P
- Fréquence : 16 MHz
- Entrées/Sorties : 14 numériques (6 PWM), 8 analogiques
- Mémoire : 32 Ko Flash, 2 Ko SRAM
- Tension de fonctionnement : 5 V
- Usage typique : projets compacts, intégration sur breadboard

### II.2.1.6 Arduino Nano 33 BLE

- Microcontrôleur : nRF52840 Cortex-M4
- Fréquence : 64 MHz
- Entrées/Sorties : 14 numériques, 8 analogiques
- Mémoire : 1 Mo Flash, 256 Ko SRAM
- Tension de fonctionnement : 3.3 V
- Usage typique : applications IoT avec connectivité Bluetooth Low Energy

### II.2.2 Avantages de la carte Arduino UNO

- Facilité d'apprentissage et d'utilisation : Arduino UNO se distingue par son architecture simple et un environnement de développement (IDE) intuitif, idéal pour les novices en électronique et programmation. Sa prise en main rapide permet de se concentrer sur l'expérimentation plutôt que sur la configuration complexe.
- Écosystème riche et communauté active : La popularité mondiale de l'Arduino UNO garantit une abondance de ressources, tutoriels, bibliothèques et forums d'entraide. Cette dynamique communautaire accélère la résolution des problèmes et favorise le partage de projets innovants.
- Grande compatibilité matérielle : Dotée de nombreuses broches d'E/S (14 numériques, 6 analogiques), la carte supporte un large éventail de capteurs, moteurs, écrans et autres modules. Les shields facilitent l'extension des fonctionnalités sans soudures ni complications.
- Coût réduit et disponibilité : Accessible financièrement, l'Arduino UNO est idéal pour l'éducation et le prototypage rapide. Sa large diffusion commerciale assure une disponibilité facile et un approvisionnement continu.
- Robustesse et fiabilité éprouvées : Conçue autour du microcontrôleur ATmega328P, la carte offre une stabilité opérationnelle adaptée aux environnements pédagogiques et projets expérimentaux, même avec une utilisation intensive.

### II.2.3 Constitution de la carte Arduino UNO


**Figure (II.1) :** Les différentes parties de la carte Arduino UNO.

**Tableau (II.1) :** Les différentes parties de la carte Arduino UNO et leurs fonctions.

| **Composant** | **Fonction** |
|---|---|
| Microcontrôleur ATmega328P | Traitement et exécution du programme |
| Régulateur de tension | Alimentation stable en 5 V |
| Broches E/S numériques | Contrôle des périphériques |
| Broches analogiques | Lecture de signaux analogiques |
| Convertisseur USB-série | Communication PC via USB |
| Quartz 16 MHz | Synchronisation des opérations |
| Bouton Reset | Réinitialisation manuelle |
| LEDs intégrées | Indications visuelles |

### II.2.3.1 Microcontrôleur principal :

L'élément central est un microcontrôleur ATmega328P de chez Atmel (désormais Microchip), un circuit intégré 8 bits reposant sur l'architecture AVR RISC [29].

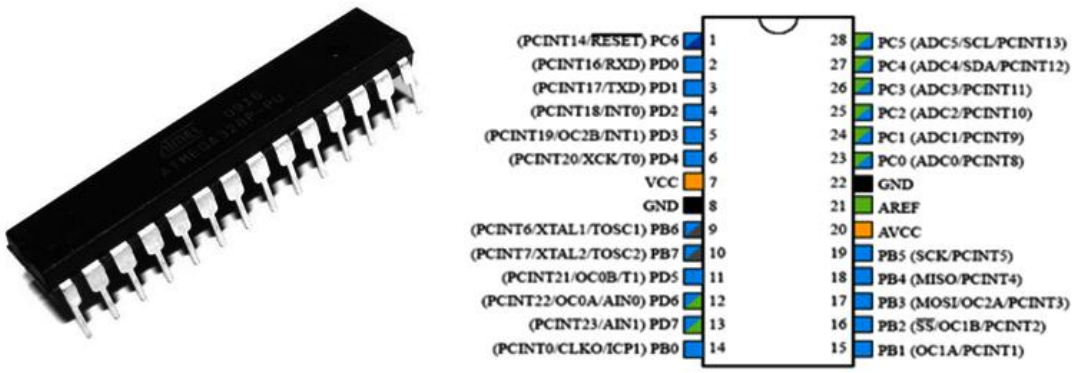



**Figure (II.2) :** Microcontrôleur ATMEGA328p.

Ce microcontrôleur fonctionne à une fréquence de 16 MHz et possède une mémoire flash de 32 Ko pour stocker le programme, 2 Ko de SRAM pour les données temporaires, et 1 Ko d'EEPROM pour le stockage non volatile.

### II.2.3.2 Alimentation :

La carte peut être alimentée via un câble USB ou par une source externe (7 à 12 V recommandé). Un régulateur de tension intégré stabilise l'alimentation à 5 V nécessaire

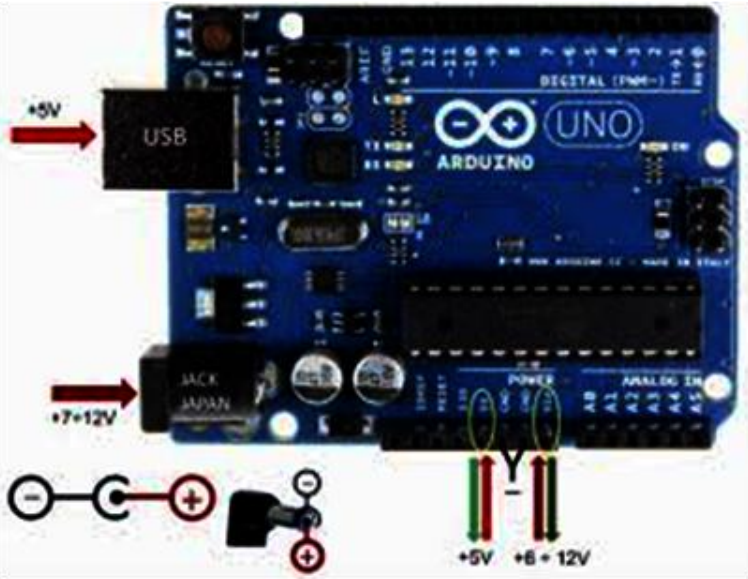



**Figure (II.3):** L'alimentation de la carte arduino uno.

Pour le fonctionnement des composants. La carte dispose aussi de broches dédiées à l'alimentation (5 V, 3.3 V, GND) pour alimenter des modules externes.

1. Entrées/Sorties numériques : Elle offre 14 broches numériques configurables en entrée ou sortie, dont 6 peuvent générer des signaux PWM (modulation de largeur d'impulsion), utiles pour le contrôle de moteurs, LED, etc.
2. Entrées analogiques : doté de six broches analogiques convertissent des signaux continus en valeurs numériques (10 bits de résolution), permettant la lecture de capteurs analogiques comme les potentiomètres ou capteurs de température.

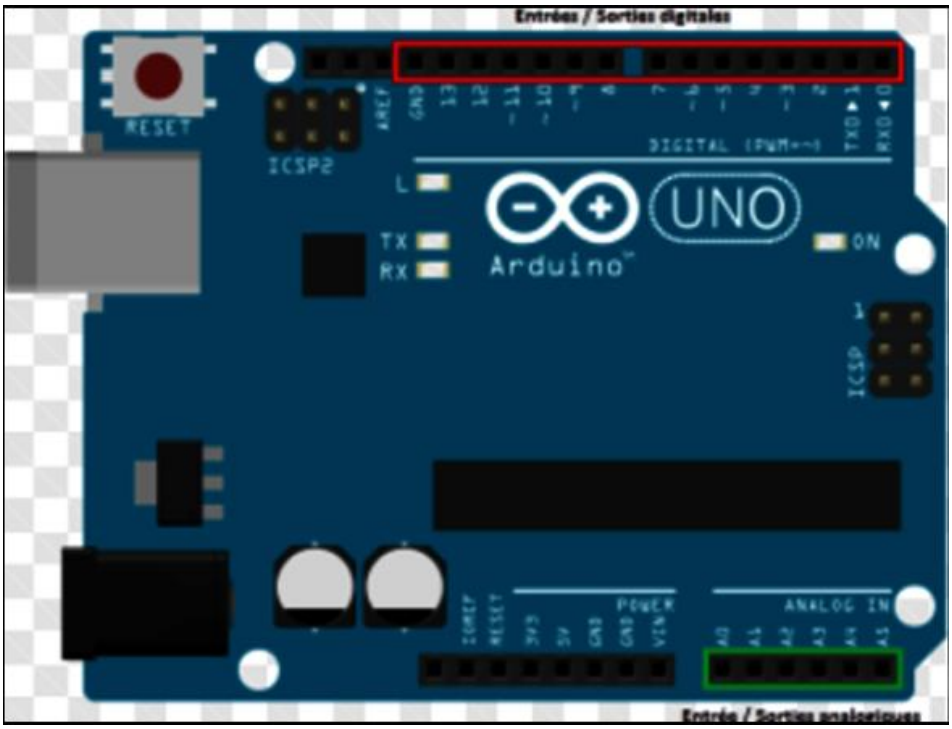



**Figure (II.4) :** Entrées/Sorties numériques et les entrées analogiques.

3. Communication série : Un port USB permet la communication entre le microcontrôleur et un ordinateur via un convertisseur USB-série (souvent un ATmega16U2 sur les versions récentes). La carte supporte également des protocoles de communication standards tels que UART, SPI et I2C [30].

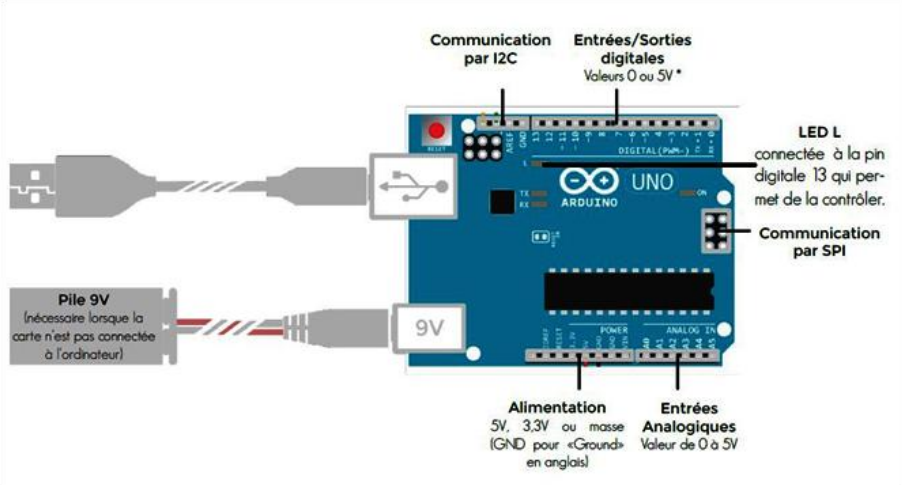



**Figure (II.5) :** Communication série.

4. Quartz oscillateur : Un oscillateur à quartz de 16 MHz fournit l'horloge système garantissant la synchronisation précise des opérations du microcontrôleur.
5. Bouton Reset : Un bouton physique permet de réinitialiser manuellement le microcontrôleur, relançant le programme chargé.

6. LEDs intégrées : On distingue les LED « ON » indiquant la mise sous tension, les LED connectée à la broche 13 pour des tests rapides et LED de communication RX/TX indiquant l'activité série.

### II.3 Programmation de la carte Arduino UNO

La programmation de la carte Arduino UNO repose principalement sur son environnement de développement intégré (IDE) et sur le langage C/C++ simplifié par des bibliothèques dédiées. Cette approche permet de rendre accessible la programmation des microcontrôleurs même aux débutants, tout en conservant la puissance et la flexibilité nécessaire aux projets avancés.

#### II.3.1 Environnement de développement (IDE Arduino)

L'IDE Arduino (Integrated Development Environment) est un logiciel gratuit et multiplateforme (Windows, macOS, Linux) conçu pour écrire, compiler et téléverser des programmes sur les cartes Arduino, comme l'Arduino UNO. Il comprend [31]:

- Un éditeur de texte où l'on écrit le code source (appelé *sketch*).
- Un compilateur qui traduit ce code en langage machine compréhensible par le microcontrôleur.
- Un outil de téléversement qui transfère le programme compilé vers la mémoire flash de la carte via le câble USB.
- Un moniteur série permettant de communiquer en temps réel avec la carte pour afficher des données ou envoyer des commandes.

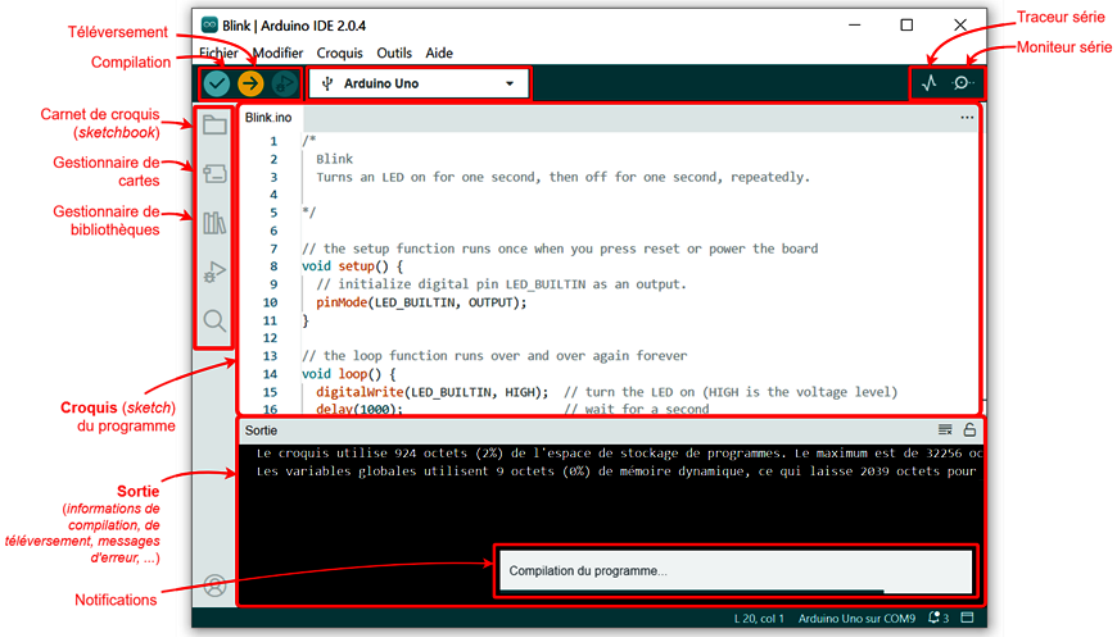



**Figure (II.6) :** Environnement de développement de l'arduino IDE

### II.3.1.1 Installation

L'IDE peut être téléchargé depuis le site officiel : https://www.arduino.cc/en/software Le logiciel s'installe facilement, avec un assistant guidant la configuration initiale (choix de la carte, du port série, etc.).

### II.3.1.2 Utilisation basique

- ✓ Écrire du code dans l'éditeur.
- ✓ Vérifier/Compiler le code pour détecter les erreurs de syntaxe (bouton « Vérifier »).
- ✓ Téléverser le code sur la carte Arduino (bouton « Téléverser »).
- ✓ Ouvrir le moniteur série pour voir les données envoyées depuis la carte ou envoyer des instructions (via le port USB).

### II.3.2 Structure d'un programme Arduino (Sketch)

Un programme Arduino, appelé sketch, est écrit dans un langage proche du C/C++. Sa structure est simple et repose sur deux fonctions essentielles qui organisent l'exécution du code :

- setup() : Cette fonction est exécutée une seule fois au démarrage de la carte. Elle sert à initialiser les paramètres : configurer les broches en entrée ou sortie, démarrer la communication série, initialiser des variables ou des capteurs.
- loop() : Cette fonction contient le cœur du programme et s'exécute en boucle infinie. C'est ici que se trouve la logique principale, par exemple lire des capteurs, contrôler des actionneurs ou gérer des communications.

Exemple classique d'une LED intégrée à la broche 13 clignote toutes les secondes.

```
void setup() {
  pinMode(13, OUTPUT); // Configure la broche 13 comme sortie
}
void loop() {
  digitalWrite(13, HIGH); // Allume la LED connectée à la broche 13
  delay(1000);            // Attend 1 seconde
  digitalWrite(13, LOW);  // Éteint la LED
  delay(1000);            // Attend 1 seconde
}
```

Ce code montre que :

- pinMode(13, OUTPUT); : définit la broche 13 en mode sortie.
- digitalWrite(13, HIGH); : met la broche 13 à l'état haut (allumé).
- delay(1000); : pause de 1000 millisecondes (1 seconde).
- digitalWrite(13, LOW); : met la broche 13 à l'état bas (éteint).

Cette organisation simplifie la programmation embarquée en distinguant clairement la phase d'initialisation (setup) et la phase d'exécution répétée (loop). Elle est adaptée à la nature cyclique des systèmes embarqués qui surveillent et contrôlent en continu leur environnement.

### II.3.3 Bibliothèques et fonctions intégrées

#### II.3.3 3.1 Bibliothèques Arduino

Arduino propose un ensemble de bibliothèques (libraries) qui facilitent la programmation en fournissant des fonctions préécrites pour gérer des composants matériels ou des protocoles complexes. Ces bibliothèques permettent d'abstraire les détails matériels et de simplifier le développement [32].

- Firmata : Un protocole qui permet de contrôler un Arduino depuis un ordinateur sans avoir à écrire un nouveau programme à chaque fois. Il est souvent utilisé avec des bibliothèques comme StandardFirmata ou FirmataExpress pour une meilleure performance.
- La bibliothèque Wire facilite la communication I2C.
- La bibliothèque SPI gère la communication SPI.

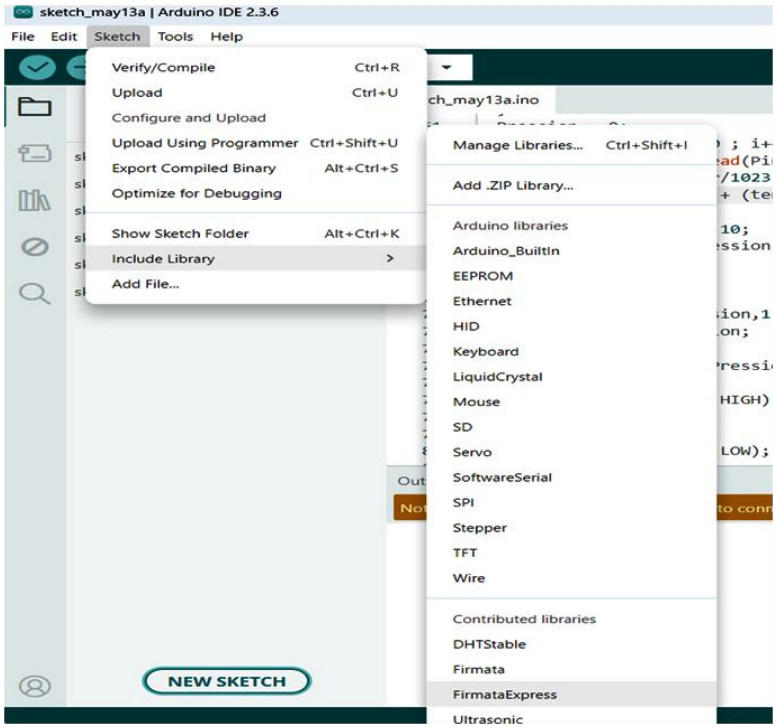



Figure (II.7) : Les Bibliothèques d'Arduino

Ces bibliothèques sont accessibles via l'IDE Arduino et peuvent être ajoutées à un projet en quelques clics.

### II.3.3 3.2 Fonctions de base

Arduino met à disposition un jeu de fonctions simples et standardisées pour gérer les opérations courantes [32]:

- Entrées/Sorties numériques
  - pinMode(pin, mode) : configure la broche pin en entrée (INPUT) ou sortie (OUTPUT).
  - digitalWrite(pin, value) : met la broche à l'état haut (HIGH) ou bas (LOW).
  - digitalRead(pin) : lit l'état haut ou bas d'une broche configurée en entrée.
- Entrées analogiques et sorties PWM
  - analogRead(pin) : lit une valeur analogique sur une broche (valeur entre 0 et 1023).
  - analogWrite(pin, value) : génère un signal PWM sur une broche (valeur entre 0 et 255).
- Communication série
  - Serial.begin(baudrate) : initialise la communication série à la vitesse spécifiée (par ex. 9600 bauds).

- Serial.print(data) / Serial.println(data) : envoie des données vers le moniteur série.
- Serial.read() : lit les données reçues via la liaison série.

### II.3.4 Téléversement et exécution du programme

Le fabricant a suggéré qu'une telle carte devrait être équipée de plusieurs ports de communication ; certains types peuvent être précisés pour le moment.

#### II.3.4.1 Téléversement du code

Une fois le code (sketch) écrit et compilé dans l'IDE Arduino, la prochaine étape est le téléversement vers la carte Arduino UNO. Ce processus consiste à transférer le programme binaire dans la mémoire flash du microcontrôleur ATmega328P.

- Le transfert s'effectue via une liaison USB grâce à un convertisseur USB-série intégré (souvent un ATmega16U2 sur Arduino UNO).
- L'IDE gère automatiquement la sélection du port série et la communication avec la carte pour le transfert.

#### II.3.4.2 Phase d'exécution

Dès que le téléversement est terminé avec succès :

- Le microcontrôleur démarre automatiquement l'exécution du programme depuis la mémoire flash.
- Il commence par exécuter la fonction setup() une seule fois pour initialiser les ressources.
- Ensuite, il entre dans une boucle infinie, exécutant sans cesse la fonction loop().

Le programme fonctionne de manière autonome, contrôlant les entrées/sorties et réagissant selon la logique définie.

#### II.3.4.3 Moniteur série

L'IDE Arduino inclut un moniteur série qui permet une communication bidirectionnelle entre l'ordinateur et la carte pendant l'exécution :

- La carte peut envoyer des données via la liaison série, visibles en temps réel dans le moniteur.
- L'utilisateur peut aussi envoyer des commandes ou des données depuis le moniteur vers la carte.

Cet outil est précieux pour le débogage et l'analyse du comportement du programme.

### II.3.4.4 Reprogrammation et mise à jour

L'utilisateur peut modifier le code et téléverser une nouvelle version aussi souvent que nécessaire. Le microcontrôleur écrase l'ancien programme et exécute le nouveau à chaque mise à jour.

## II.4 Scripts Python

Python est un langage de script orienté objet, publié en 1991. Il a été développé par Guido van Rossum de l'Institut national de recherche en mathématiques et en informatique d'Amsterdam. Python est rapidement devenu l'un des langages de programmation les plus populaires au monde. Il est aujourd'hui particulièrement utilisé pour le calcul éducatif et scientifique, et comme langage de programmation le plus populaire en science des données [32].

Les raisons pour lesquelles Python est populaire et que tout le monde devrait envisager de l'apprendre :

- Il est open source, gratuit et largement disponible, bénéficiant d'une vaste communauté open source.
- Il est plus facile à apprendre que des langages comme C, C++ et Java, ce qui permet aux développeurs débutants comme professionnels de se familiariser rapidement avec le langage.
- Il est plus facile à lire que de nombreux autres langages de programmation populaires.
- Il est populaire en intelligence artificielle, qui connaît une croissance fulgurante, en partie grâce à sa relation privilégiée avec la science des données.
- Il améliore la productivité des développeurs grâce à de vastes bibliothèques standard et des bibliothèques open source, permettant aux programmeurs d'écrire du code plus rapidement et d'effectuer des tâches complexes avec un minimum de code.

En mode interactif, les lignes d'instructions ne sont plus accessibles une fois exécutées. Mais il est possible d'écrire et de conserver un programme (un script) à l'aide d'un éditeur, afin de pouvoir l'exécuter à volonté ou le modifier ultérieurement. Il existe de nombreux éditeurs de scripts Python qui incluent également un interpréteur pour exécuter les programmes. On parle alors d'IDE (environnement de développement intégré). Il s'agit d'un environnement de programmation complet se présentant sous la forme d'une application. Il se compose généralement d'un éditeur de code, d'un interpréteur, d'un débogueur… On peut citer Pycharm, Spider… Mais pour une introduction à la programmation en Python, l'utilisation de l'interpréteur IDLE, qui permet également l'édition de scripts, est tout à fait adaptable.

## II.5 Programmation avancée : Firmata, Firmata Express et PySerial

### II.5.1 protocole de contrôle à distance

Firmata est un protocole de communication série standardisé qui permet à un ordinateur hôte de contrôler à distance une carte Arduino sans nécessiter la programmation embarquée classique. Il fonctionne en transmettant des commandes structurées via USB (ou autre liaison série), permettant d'activer ou de lire les entrées/sorties numériques et analogiques, gérer la modulation PWM, les servomoteurs, et même les bus I2C et SPI [33].

Grâce à Firmata, les développeurs peuvent piloter la carte Arduino depuis des langages de haut niveau comme Python, JavaScript, ou Processing, accélérant le développement et simplifiant l'interfaçage matériel-logiciel [34].

### II.5.2 Firmata Express : version allégée pour le prototypage

Firmata Express est une variante simplifiée du protocole Firmata. Cette version réduit le nombre d'instructions et limite les fonctionnalités aux plus couramment utilisées, comme les lectures et écritures analogiques et numériques.

L'objectif est d'améliorer la performance en limitant la surcharge et de rendre le protocole plus accessible aux débutants et aux environnements éducatifs, où la simplicité prime sur la richesse fonctionnelle complète.

### II.5.3 PySerial : interface Python pour la communication série

Pour exploiter Firmata (et Firmata Express) depuis Python, la bibliothèque **PySerial** est essentielle. Elle offre une interface simple pour ouvrir, lire et écrire des données sur un port série (USB) [35].

L'association PySerial + Firmata permet de :

- Établir une connexion avec la carte Arduino via USB.
- Envoyer des commandes Firmata pour contrôler les broches (digitales/analogiques).
- Recevoir des données de capteurs en temps réel.

Cette approche est idéale pour développer des applications interactives, de visualisation de données, ou des systèmes de contrôle embarqués pilotés par ordinateur.

### II.5.4 Communication Arduino – Python

Les méthodes de communications entre un arduino et un programme Python sont résumées à l'aide des schémas suivant :

### II.5.4.1 Communication Arduino - Python via le port série avec Pyserial :

L'Arduino envoie des données sur le port série par la fonction ”Serial.print()”

- ❖ Le programme Python réceptionne les données de la liaison série, envoyées par l'Arduino, à l'aidede la fonction ”readline()”
- ❖ Le programme Python envoie des données sur le port série avec la fonction ”write()”
- ❖ L'arduino réceptionne les données envoyées par le programme Python sur la liaison série grâce à lafonction ”Serial.read()”

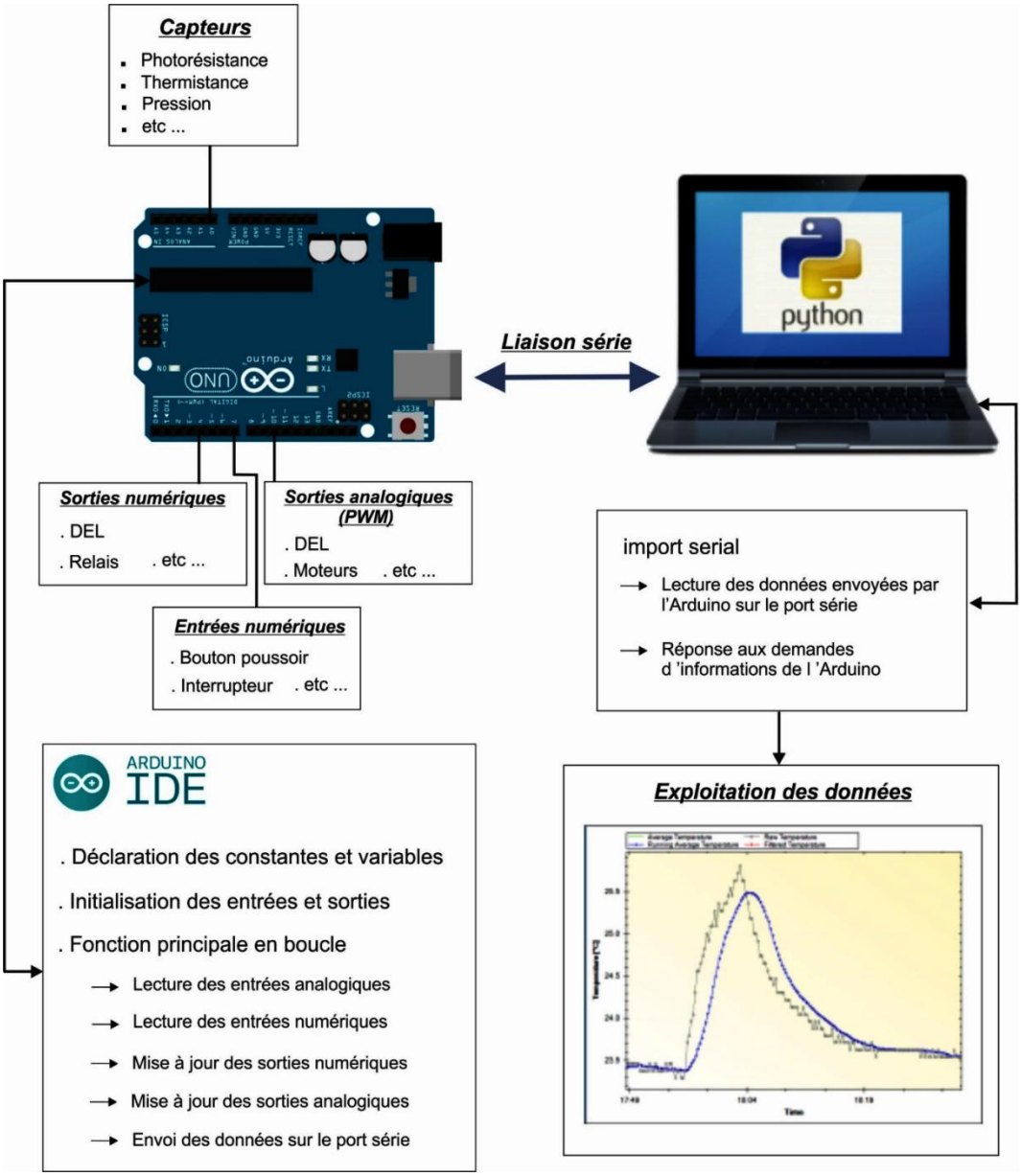



**Figure (II.8) :** Communication Arduino - Python via le port série avec Pyserial [36].

### II.5.4.1 Communication Arduino-Python via le protocole de communication Firmata

Pour contrôler l'Arduino via le protocole de communication Firmata standard, le programme Python utilise la bibliothèque "PyFirmata" (le sketch "Firmata standard" doit être téléversé dans la mémoire de l'arduino au préalable) [37].

Toutes les fonctions utiles à l'utilisation de "PyFirmata" (fonction de déclaration des entrées et sorties, d'itérateur, d'écritures…) que nous avons définies jusqu'à présent ont été regroupées dans un fichier Python, nommé "PyFirmataDef.Py" que l'on peut importer dans tous les programmes, à condition que le fichier des fonctions soit dans le même dossier que le fichier du programme, avec l'instruction : from PyFirmataDef import *

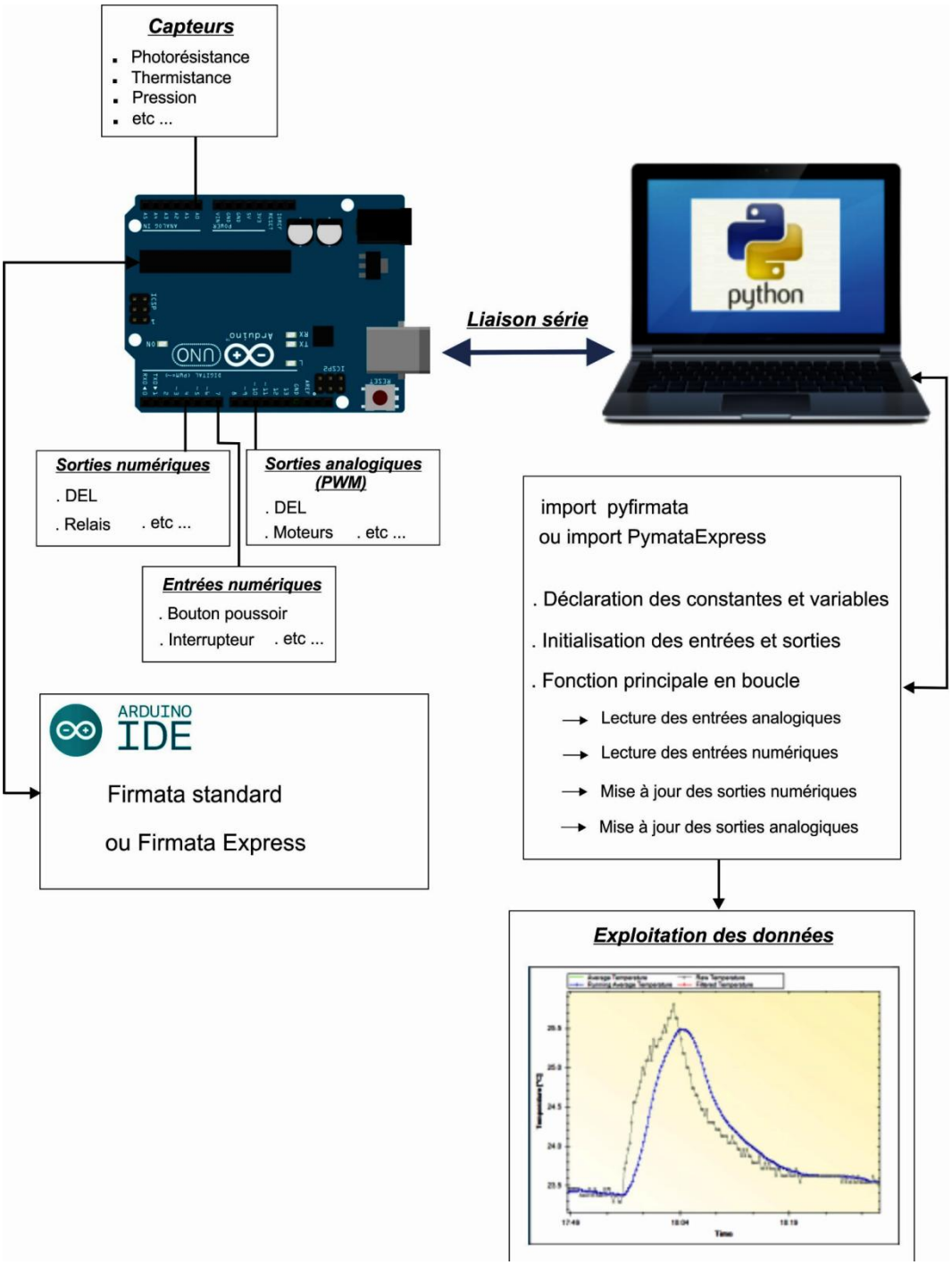



**Figure (II.9) :** Communication Arduino-Python via le protocole de communication Firmata [38].

## II.6 Conclusion

Ce chapitre présente une présentation complète de l'Arduino Uno, une carte d'acquisition fréquemment utilisée. Nous avons expliqué clairement le processus de décision et examiné en détail les options disponibles. Nous avons ensuite clarifié les éléments essentiels d'Arduino, notamment les composants matériels et la programmation, avec plus de précision, ainsi le principe de fonctionnement de la carte Arduino tout en intégrant ses caractéristiques uniques. Nous avons également étudié le protocole Firmata, qui améliore l'environnement Arduino en permettant les connexions entre les microcontrôleurs et les ordinateurs hôtes, permettant ainsi un contrôle avancé du matériel via d'autres langages de programmation. Nous avons également analysé Firmata Express, une version modifiée de Firmata qui réduit le jeu de commandes pour une utilisation plus simple, la rendant particulièrement adaptée aux applications pratiques et au prototypage rapide.

# Chapitre III

# Etude et simulation du capteur de pression MPX2200AP

## III.1 Introduction :

Dans ce chapitre, nous présentons les résultats expérimentaux réalisés à l'aide d'un système électronique basé sur une carte Arduino et le capteur de pression MPX2200AP. Un montage électronique a été conçu et mis en œuvre afin de mesurer la pression et de la convertir en signaux électriques exploitables par des outils numériques. Ce système nous a permis d'étudier la pression en fonction de différentes grandeurs physiques : la tension électrique, le volume et la profondeur.

Le dispositif utilisé comprend les composants essentiels suivants : le capteur MPX2200AP pour la détection de la pression, une carte Arduino UNO pour la collecte des données, une LED pour l'indication visuelle, ainsi qu'un bouton-poussoir pour le contrôle manuel de l'expérience. Une seringue et une tubulure ont été utilisées pour appliquer une pression contrôlée sur le capteur de manière sécurisée.

Les données ont été collectées via un code Arduino spécifique, puis analysées et modélisées en Python, ce qui nous a permis d'étudier et de visualiser les relations entre la pression et les autres variables physiques comme le volume ou la profondeur.

Ce chapitre présentera en détail les résultats obtenus, accompagnés, en mettant l'accent sur la précision, la réactivité et la fiabilité du système dans différents contextes expérimentaux.

## III.2 Capteur de pression MPX2200AP

Pour mesurer cette grandeur avec un Arduino, on utilisera le capteur de pression MPX2200AP dont la sensibilité est de 20 mv/kPa pour une plage de mesure allant de 20 à 250 kPa (soit, une tension de sortie du capteur entre 0,2 V et 4,8 V). Ce capteur est donc parfaitement adapté à une utilisation avec un Arduino.

Le capteur de pression absolue MPX4250AP dispose de 6 broches, mais seules trois bornes sont utilisées pour le connecter à une carte Arduino (Figure III.1):

$V_{CC}$ : +5 V

GROUND : masse

$V_{out}$ : signal de sortie sur une entrée analogique de la carte.

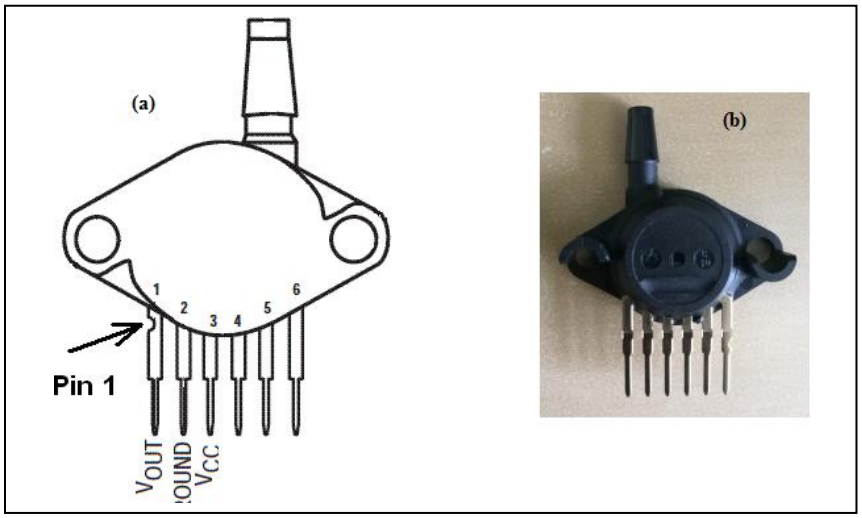



**Figure (III.1) :** Capteur de pression MPX2200AP (a) ;Vue de dessous : (b) :Vue de dessue.

### III.2.1 Caractéristiques du capteur de pression MPX2200AP :

- Tension de fonctionnement : de 4,85V à 5,35V.
- Courant circuit alimentation maximal : 10 mA.
- Sensibilité : 20 mV / kPa.
- Plage de mesure : 20 à 250 kPa.
- Tension de sortie (Vout) : 0,2 V à 4,8 V.
- Précision : ± 1,5% sur Vout.
- Courant de charge maximal (signal de sortie) : 0,1 mA.
- Temps de réponse : 1 ms.
- Température de fonctionnement : 0 - 85 °C

La fiche technique du capteur, fourni par le constructeur, donne la courbe de transfert reliant la pression absolue P en kPa à la tension de sortie $V_{out}$ en V [40] :

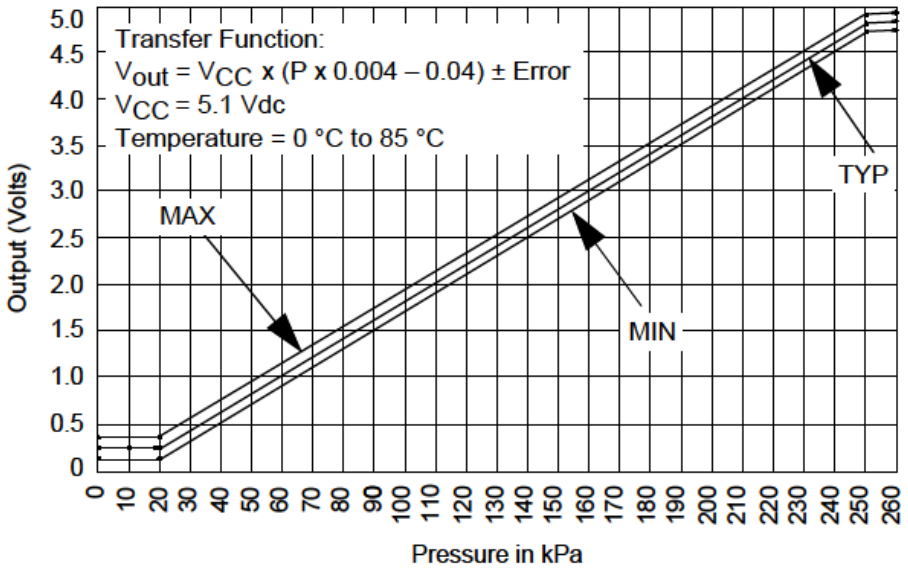



**Figure (III.2) :** courbe de transfert des caractéristiques techniques principales du capteur MPX2200AP [40].

On a donc :

$V_{out} = 5{,}0\ x\ (P\ x\ 0{,}004 - 0{,}04)$ *et* $P(Kpa) = (V_{out}/0.02) + 10$

### III.2.2 Schéma électrique interne équivalent

Le schéma fonctionnel du circuit interne du MPX2200AP est représenter dans la figure III.3 montrant la cellule piézorésistive (pont de Wheatstone), l'amplificateur d'instrumentation, et le circuit de compensation de température. Le signal analogique linéaire est généré en sortie ($V_{out}$).

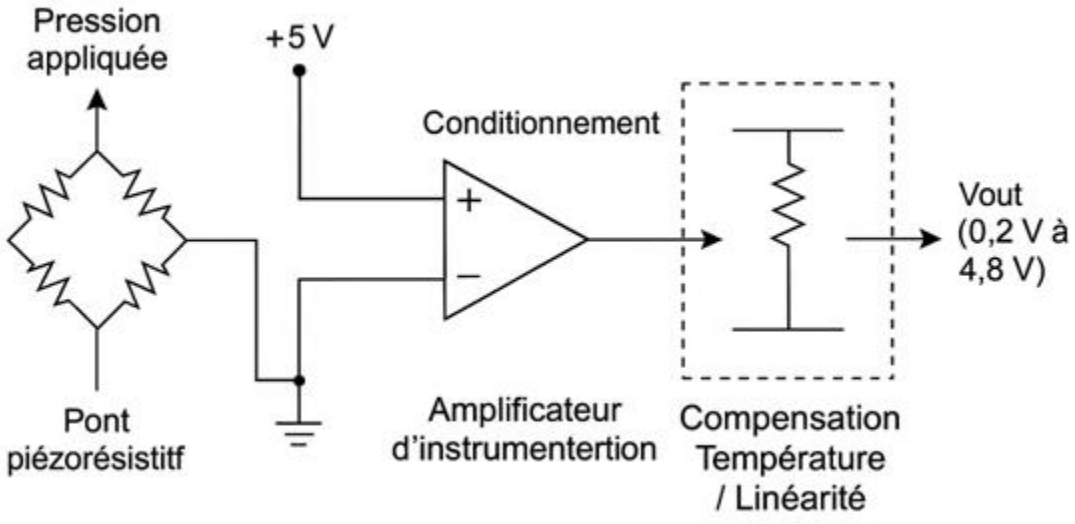



**Figure (III.3) :** Schéma fonctionnel du circuit interne du MPX2200AP [40].

### III.3 Dispositif de mesure de pression

Les différentes taches de mesure de pression et d'exploitation seront réalisées avec le circuit suivant :

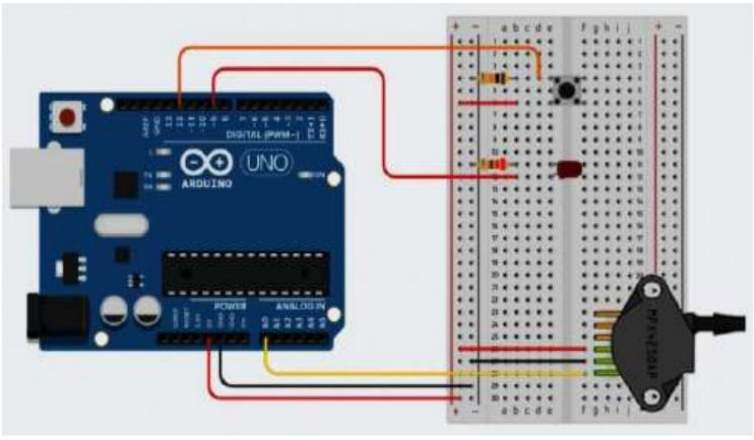



**Figure )III.4) : Circuit de mesure de pression et de fonctionnement**

Le matériel associer comprend :

- ❖ Matériel informatique :
- Un micro-ordinateur (PC)
- Câble USB pour la connexion Arduino
- Logiciel de simulation
- ❖ Composants électroniques :

- Une carte Arduino UNO
- Une plaque d'essais
- Des fils de liaison
- Un capteur de pression MPX2200AP
- Une LED rouge (diode électroluminescente)
- Unr résistance de 220 Ω (pour protéger la DEL)
- Une résistance de 10 kΩ (pour le bouton poussoir)
- Un bouton poussoir
- Une seringue
- Un tuyau

### III.3.1 Mesure d'une pression absolue avec un capteur MPX2200AP:

Nous allons simplement mettre en œuvre l'utilisation d'un capteur de pression dont la sortie est reliée à l'entrée A0 de l'Arduino (figure III.5),

Pour afficher dans la fenêtre Python Shell ou le moniteur série, la tension mesurée et la pression correspondante après un appui sur le bouton poussoir. Un nouvel appui sur le bouton poussoir arrête les mesures. La pression mesurée sera modifiée avec une seringue d'un volume utile de 60 mL fixée au capteur par l'intermédiaire d'un tuyau. En déplaçant le piston, initialement placé sur la graduation 30 mL, on fera varier le volume de l'air enfermé dans le corps de la seringue et donc la pression appliquée sur le capteur.

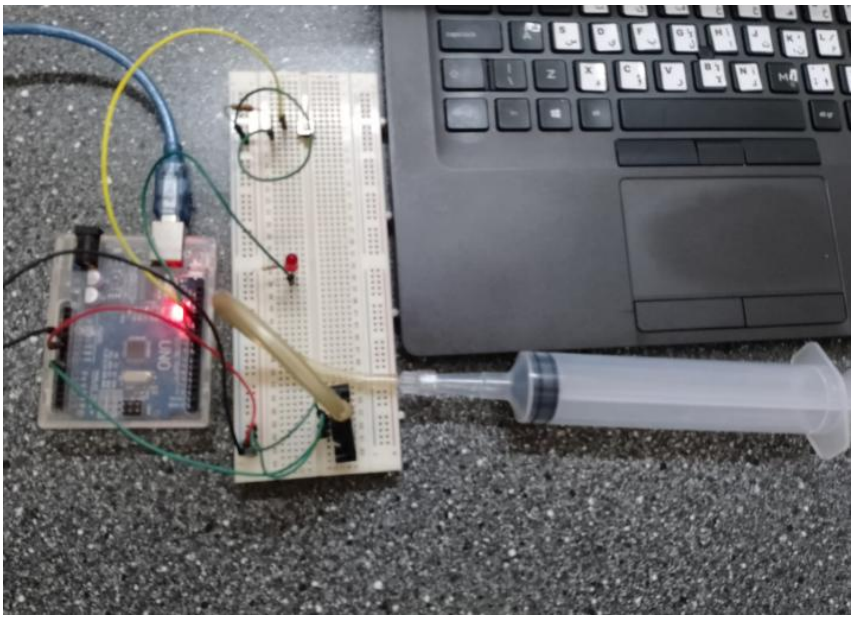


**Figure (III .5) :** Montage expérimentale utilisé pour la mesure de pression absolue.

#### III.3.1.1 Programme en langage Arduino:

Ce code programmé, avec le logiciel ARDUINO IDE, permet de mettre en œuvre l'utilisation d'un capteur de pression MPX2200AP.

```
// Déclaration des constantes et variables
const int PinSensor=0;
const int PinButton= 12;
int ValSensor = 0;
float tension = 0.0;
float Pression = 0.0;
float OldPression = 0.0;
int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;
// Initialisation des entrées et sorties
void setup() {
Serial.begin(9600);
pinMode(PinButton,INPUT);
Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}
// Fonction principale en boucle
void loop() {
ValButton = digitalRead(PinButton);
delay(10);
if ((ValButton == HIGH)&&(OldValButton == LOW))
{
State=1-State;
}
OldValButton = ValButton;
if (State==1)
{
if (OldState == 0)
{
// Fonction principale en boucle
void loop() {
  ValButton = digitalRead(PinButton);
delay(10);
if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;
if (State==1)
  {
if (OldState == 0)
  {
   Serial.println("Mesure de la pression en cours.");
   Serial.println("");
   Serial.println ("Tension Entree A0 (en V) ; Pression (en kPa):");
   OldState=1;
```

```
  }
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;
  Pression = (tension / 0.02) + 10;
  if (OldPression != Pression)
  {
    Serial.print(tension);
    Serial.print(" ; ");
    Serial.println(Pression,1);
    OldPression = Pression;
  }
  delay(500);
  }
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    OldState = 0;}
  }
}
```

### III.3.1.2 Résultats dans le moniteur série

Pour visualiser les résultats la pression (en KPa) en fonction de la tension appliquée (en V) à l'entrée A0 il faut appuyer sur le bouton poussoir

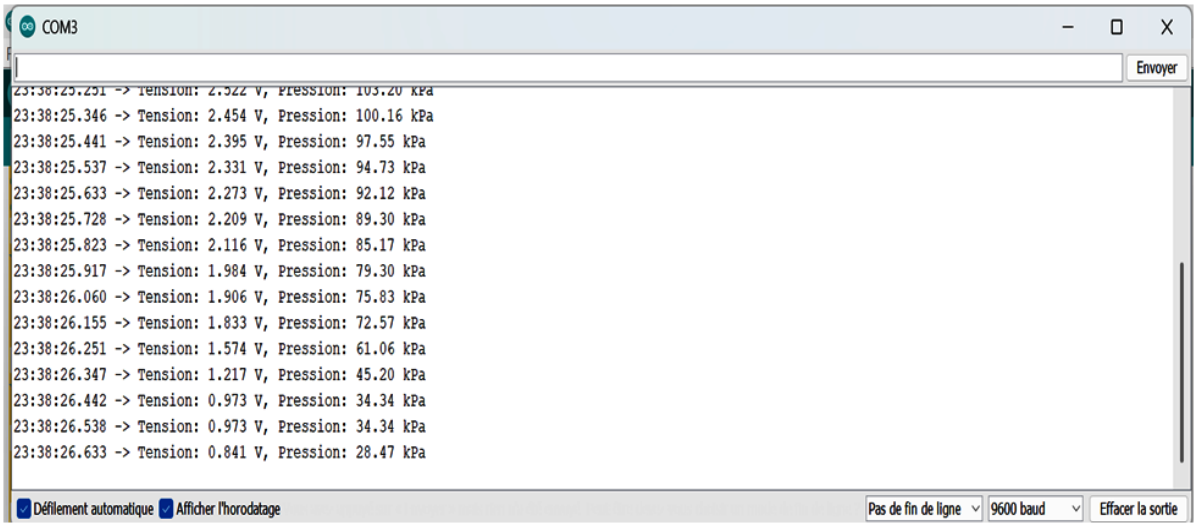



**Figure (III.6):** résultats de mesure de la pression absolue dans l'Arduino.

### III.3.1.3 Déroulement du programme :

- ❖ Déclaration des constantes et variables :
- *const int PinSensor = 0* (broche du capteur de pression)
- *const int PinButton = 12* (broche du bouton poussoir)
- *int ValSensor = 0* (variable nombre entier valeur broche du capteur)
- *float tension = 0.0* (variable nombre décimal calcul tension broche capteur)
- *float Pression = 0.0* (variable nombre décimal calcul pression)
- *float OldTPression = 0.0* (variable nombre décimal ancien calcul pression)
- *int ValButton = 0* (variable nombre entier valeur broche bouton poussoir)
- *int OldValButton = 0* (variable nombre entier ancienne valeur broche bouton poussoir)
- *int State* = (variable nombre entier pour action à effectuer)
- *int OldState = 0* (variable nombre entier pour action effectuée précédemment)
- ❖ Initialisation des entrées et sorties :
- Initialisation de la liaison série à un débit de 9600 bauds,
- Initialisation de la broche du bouton poussoir en entrée numérique.
- Fonction principale en boucle

### III.3.1.4 Programme en Python :

Pour contrôler l'Arduino via le port série avec Pyserial ou le protocole de communication Firmata standard, le programme Python utilise la bibliothèque "PyFirmata" (le sketch "Firmata standard" doit être téléversé dans la mémoire de l'Arduino au préalable).

Le code en Python élaborer est le suivant :

```
import serial
import time
import matplotlib.pyplot as plt
# --- Connexion série ---
port = 'COM3'  # Remplacez par le port réel (COMx)
baudrate = 9600
try:
    arduino = serial.Serial(port, baudrate, timeout=1)
    time.sleep(2)
    print(" Connexion à l'Arduino réussie.\n")
except:
    print(" Erreur de connexion au port série.")
    exit()
# --- Données ---
tensions = []
pressions = []
# --- Fonction pour LED ---
def envoyer_commande_led(etat):
    if etat == 1:
        arduino.write(b'1')
    else:
        arduino.write(b'0')
# --- Graphique interactif ---
plt.ion()
fig, ax = plt.subplots()
line, = ax.plot([], [], 'bo-', label="Pression (kPa)")
ax.set_xlabel("Tension (V)")
ax.set_ylabel("Pression (kPa)")
ax.set_title("Pression en fonction de la tension")
ax.grid(True)
ax.legend(loc="upper left")
# --- Boucle principale ---
try:
    while True:
        ligne = arduino.readline().decode('utf-8', errors='ignore').strip()
        if ligne:
            print(ligne)
            if "Tension:" in ligne and "Pression:" in ligne:
                try:
                    parts = ligne.split(',')
                    tension = float(parts[0].split(":")[1].replace("V", "").strip())
                    pression = float(parts[1].split(":")[1].replace("kPa", "").strip())
```

```
                tensions.append(tension)
                pressions.append(pression)
                # Mise à jour du graphique
                line.set_xdata(tensions)
                line.set_ydata(pressions)
                ax.relim()
                ax.autoscale_view()
                plt.draw()
                plt.pause(0.01)
                # Contrôle LED
                if pression > 100:
                    envoyer_commande_led(1)
                else:
                    envoyer_commande_led(0)
            except Exception as e:
                print(" Erreur de parsing:", e)
        time.sleep(0.1)
# --- Sortie ---
except KeyboardInterrupt:
    print("\n Programme arrêté par l'utilisateur.")
    plt.ioff()
    # Tracer final et sauvegarde
    plt.figure(figsize=(8,5))
    plt.plot(tensions, pressions, 'r.-', linewidth=2, markersize=5, label="Pression
mesurée")
    plt.xlabel("Tension (V)")
    plt.ylabel("Pression (kPa)")
    plt.title("Graphique final : Pression en fonction de la tension")
    plt.grid(True)
    plt.legend()
    plt.savefig("graphique.png")
    plt.show()
    arduino.close()
```

### III.3.1.5 Résultats dans la fenêtre Python Shell :

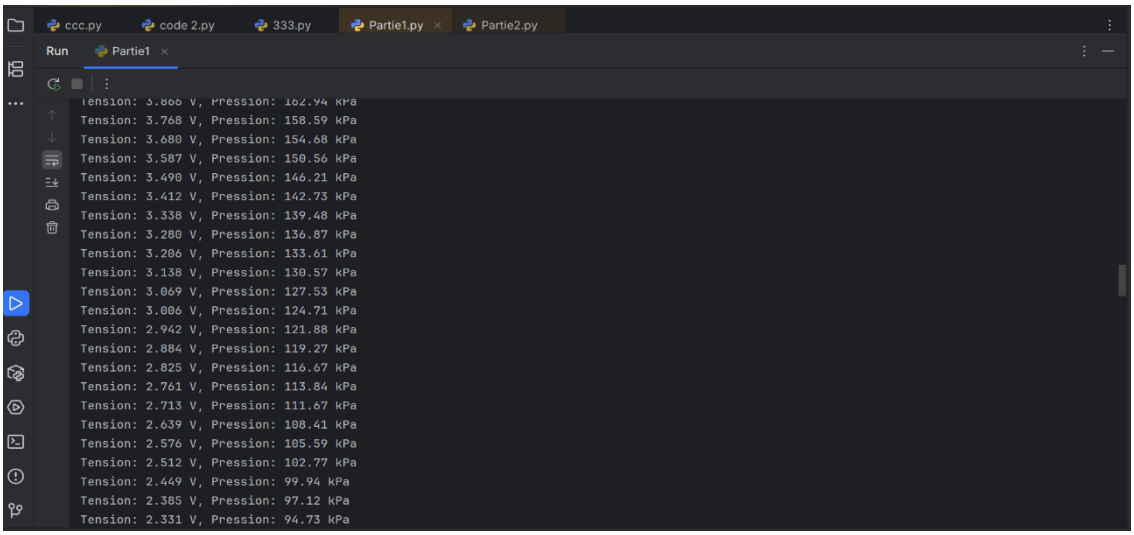



**Figure ( III. 7):** Résultats de mesure de la pression absolue dans Python.

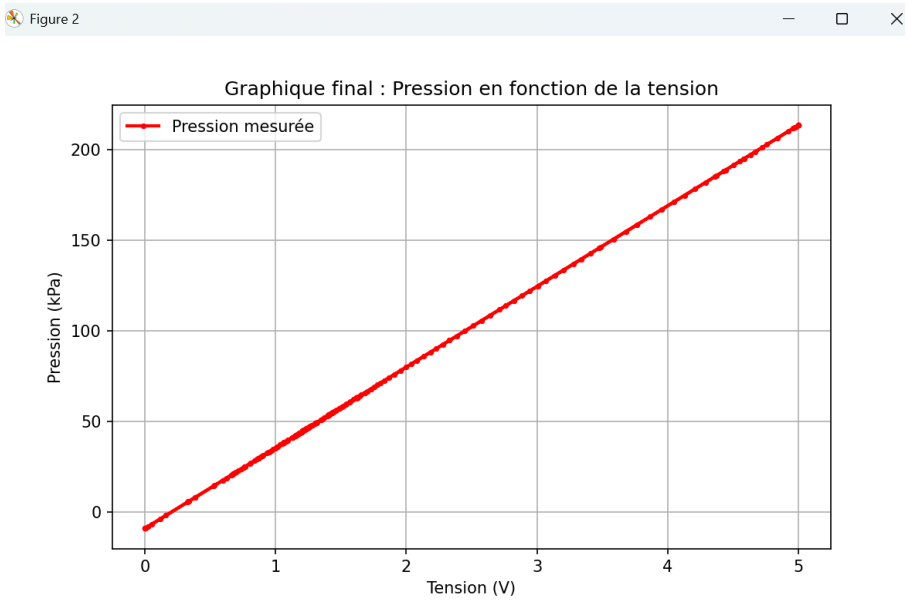



**Figure (III.8) :** L'évolution de la pression mesurée en fonction de la tension

À partir des mesures obtenues, on observe une relation quasi linéaire entre la tension et la pression, ce qui est cohérent avec les caractéristiques du capteur MPX2200AP, qui génère une tension proportionnelle à la pression appliquée.

Les mesures montrent que l'augmentation de la tension entraîne une augmentation de la pression,

ce qui indique que la mesure et le traitement du signal ont été réalisés avec succès et avec une précision acceptable.

### III.3.2 Indicateur de pression

De façon à s'assurer que la pression mesurée par notre capteur MPX2200AP ne soit pas supérieure ou inférieure à la pression maximale (250 kPa) ou minimale (20 kPa) admissible, nous avons utilisé une DEL rouge qui sera allumée quand la pression est supérieure ou inférieure à des seuils à définir afin de prévenir de leurs dépassements. La pression mesurée sera modifiée avec une seringue d'un volume utile de 60 mL fixée au capteur par l'intermédiaire d'un tuyau. En déplaçant le piston, initialement placé sur la graduation 30 mL, on fera varier le volume de l'air enfermé dans le corps de la seringue et donc la pression appliquée sur le capteur.

La pression est mesurée après un appui sur le bouton poussoir. Un nouvel appui sur le bouton poussoir arrête les mesures.

#### III.3.2.1 Programme en langage Arduino:

```
// Déclaration des constantes et variables
const int PinSensor = 0;
const int PinButton = 12;
const int PinLed = 9;
const int PMax = 200;
const int PMin = 55;
int ValSensor = 0;
float tension = 0.0;
float Pression = 0.0;
float OldPression = 0.0;
int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;
// Initialisation des entrées et sorties
void setup() {
  Serial.begin(9600);
 pinMode(PinButton,INPUT);
pinMode(PinLed,OUTPUT);
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}
```

```
// Fonction principale en boucle
void loop() {
  ValButton = digitalRead(PinButton);
delay(10);
if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;
if (State==1)
{
  if (OldState == 0)
  {
   Serial.println("Mesure de la pression en cours.");
   Serial.println("");
   Serial.println ("Pression (en kPa):");
   OldState=1;
  }
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;
  Pression = tension / 0.02 + 10;
  if (OldPression != Pression)
  {
    Serial.println(Pression,1);
    OldPression = Pression;
  }
  if (Pression>PMax or Pression<PMin)
  {
    digitalWrite(PinLed,HIGH);
  }
  else {
    digitalWrite(PinLed,LOW);
  }
  delay(500);
}
else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    digitalWrite(PinLed,LOW);
    OldState = 0;}
  }
}
```

### III.3.2.2 Programme en Python :

Pour contrôler l'Arduino via le port série avec Pyserial**.** Le code en Python élaborer est le suivant :

```
# Importations des librairies et définition de fonctions
import serial
import time
# Déclaration des constantes et variables
PinSensor = 0
PinButton = 12
PinLed = 9
PMax = 200
PMin = 55
ValSensor = 0
tension = 0
Pression = 0
OldPression = 0
ValButton = 0
OldValButton = 0
State = 0
OldState = 0
# Connexion à l'Arduino
print("\nConnexion à l'Arduino en cours...")
PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)
InputPinSensor = AnalogInput(board, PinSensor)
InputPinButton = DigitalInput(board, PinButton)
ArduinoIterateur = Iterateur(board)
time.sleep(0.5)
print("\nConnexion à l'Arduino établie - Appuyez sur Ctrl-C pour quitter\n")
print("\nAppuyez sur le bouton poussoir pour commencer les mesures.\n");

# Boucle principale du programme
while True:
    try:
        ValButton = InputPinButton.read()
        time.sleep(0.01)
        if ValButton == 1 and OldValButton == 0:
            State=1-State
        OldValButton = ValButton;
        if State==1:
            if OldState == 0:
                print("\nMesure de la pression en cours.\n")
                print("Tension Entree A0 (en V) ; Pression (en kPa):")
                OldState=1
```

```
ValSensor = InputPinSensor.read()
            tension = ValSensor*5.0
            Pression = (tension / 0.02) + 10
            if OldPression != Pression:
               print(round(tension,2), " ; ", round(Pression,1))
               OldPression = Pression
            if Pression>PMax or Pression<PMin:
               DigitalWrite(board,PinLed,1)
            else:
               DigitalWrite(board,PinLed,0)
            time.sleep(0.5)
         else:
            if OldState == 1:
               print("\nFin des mesures.\n")
               DigitalWrite(board,PinLed,0)
               OldState = 0
      except KeyboardInterrupt:
         DigitalWrite(board,PinLed,0)
         board.exit()
         sys.exit(0)
```

### III.3.3 Vérification de la loi de Boyle-Mariotte:

Le but est de vérifier la loi de Boyle-Mariotte à l'aide de notre capteur de pression MPX2200AP et de la seringue d'un volume utile de 60 mL.

En appliquant la loi de Boyle-Mariotte, qui montre que :

- À température constante, le produit de la pression d'un gaz par son volume est constant (P × V = constante).
- Si un gaz subit une transformation qui l'amène d'un état n°1 (son volume est V1 et sa pression P1) à un état n°2 (son volume est V2 et sa pression P2) alors la loi de Boyle-Mariotte peut s'écrire (P1.V1 = P2.V2).

#### III.3.3.1 Descriptif de la manipulation

En déplaçant le piston, initialement placé sur la graduation 30 mL, on fait varier le volume de l'air enfermé dans le corps de la seringue reliée au capteur de pression.

Après avoir appuyé sur le bouton poussoir, le volume lu (entre 10 et 60 mL) sur le corps de la seringue est saisi au clavier dans la fenêtre Python Shell ou le moniteur série. La pression est ensuite mesurée par le capteur qui délivre une tension électrique proportionnelle à la pression.

Une moyenne est réalisée sur dix mesures et le résultat de la pression moyenne (p en kPa) pour le volume (V en mL) est affiché dans la fenêtre Python Shell ou le moniteur série. Les mesures pour le volume V sont arrêtées en appuyant sur le bouton poussoir.

Une nouvelle mesure de la pression pour un volume différent est réalisable en appuyant de nouveau sur le bouton poussoir.

Il est donc possible d'acquérir des couples de données (P, V) afin de vérifier la loi de Boyle-Mariotte. La LED rouge est allumée si la pression est inférieure ou supérieure aux seuils de pression correspondant à un volume de 10 ou de 60 mL.

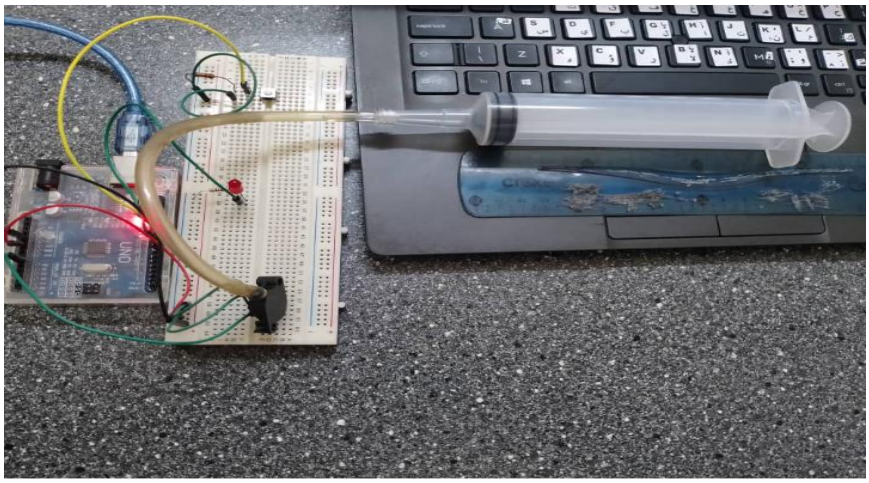


**Figure (III .9) :** Montage expérimentale utilisé pour vérifier la loi de Boyle-Mariotte.

### III.3.3.2 Programme en langage Arduino:

Ce code programmé, avec le logiciel Arduino IDE afin de vérifier la loi de Boyle-Mariotte.

```
// Déclaration des constantes et variables
const int PinSensor = 0;
const int PinButton = 12;
const int PinLed = 9;
const int PMax = 240;
const int PMin = 50;
int ValSensor = 0;
float tension = 0.0;
float Pression = 0.0;
float OldPression = 0.0;
int Vol = 0;
int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;
// Initialisation des entrées et sorties
void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  pinMode(PinLed,OUTPUT);
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}
// Fonction principale en boucle
void loop() {
ValButton = digitalRead(PinButton);
delay(10);
if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;
if (State==1)
{
  if (OldState == 0)
  {
Serial.print("Veuillez saisir le volume V de la seringue en mL ");
    Serial.println("(valeur entre 10 et 60 mL).");
    Vol = 0;
    while(Vol<10 || Vol>60)
    {
```

```
    Vol=Serial.parseInt();
   }
   Serial.println("Mesure de la pression en cours.");
   Serial.println("");
   Serial.println ("Volume (mL);Pression (kPa):");
   OldState=1;
  }
  Pression = 0;
  for(int i = 0 ; i < 10 ; i++){
   ValSensor = analogRead(PinSensor);
   tension = (ValSensor/1023.0)*5.0;
   Pression = Pression + (tension / 0.02 + 10);
  }
  Pression = Pression / 10;
  if (OldPression != Pression)
  {
   Serial.print(Vol);
   Serial.print(";");
   Serial.println(Pression,1);
   OldPression = Pression;
  }
  if (Pression>PMax or Pression<PMin)
  {
   digitalWrite(PinLed,HIGH);
  }
  else {
   digitalWrite(PinLed,LOW);
  }
  delay(500);
}
  else
  {
   if (OldState == 1){
   Serial.println("Fin des mesures.");
   Serial.println("");
   digitalWrite(PinLed,LOW);
   OldState = 0;}
  }
}
```

### III.3.3.3 Résultats dans le moniteur série :

Pour commencer les mesures il faut appuyer sur le bouton poussoir, puis il faut saisir le Volume V de la seringue en mL (Valeur entre 10 et 60 mL)

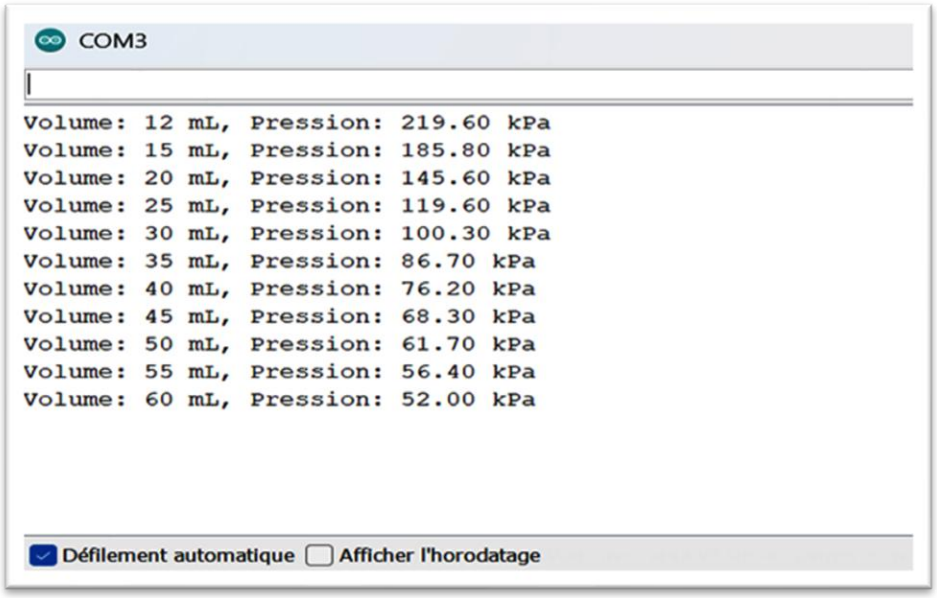



**Figure (III.10):** résultats de mesure dans l' Arduino a fin de vérifier la loi de Boyle-Mariotte.

### III.3 .3.4 Code python :

```
import serial
import matplotlib.pyplot as plt
import time
def moyenne_mobile(data, window_size=5):
    """Calcule une moyenne mobile simple."""
    if len(data) < window_size:
        return sum(data) / len(data)
    else:
        return sum(data[-window_size:]) / window_size
# Configuration du port série
port = 'COM3'  #  Remplacez selon votre cas
baudrate = 9600
ser = serial.Serial(port, baudrate, timeout=1)
time.sleep(2)
# Stockage des données
pressions_brutes = []
volumes_bruts = []
pressions_filtrees = []
volumes_filtrees = []
# Initialisation du graphique
plt.ion()
fig, ax = plt.subplots()
line, = ax.plot([], [], 'bo-', label="P = f(V)")
ax.set_xlabel("Volume (mL)")
ax.set_ylabel("Pression (kPa)")
ax.set_title("Courbe améliorée : Pression vs Volume")
ax.grid(True)
ax.legend()
derniere_pression = None
dernier_volume = None
while True:
    try:
        ligne = ser.readline().decode().strip()
        if ligne and "Pression" in ligne and "Volume" in ligne:
            parties = ligne.split('|')
            pression = float(parties[0].split(':')[1].strip())
            volume = float(parties[1].split(':')[1].strip())
```

```
            # Stockage brut
            pressions_brutes.append(pression)
            volumes_bruts.append(volume)
            # Moyenne mobile (filtrage simple)
            pression_f = moyenne_mobile(pressions_brutes)
            volume_f = moyenne_mobile(volumes_bruts)
            # Ne tracer que si la valeur change significativement (filtrage simple)
            if (derniere_pression is None or abs(pression_f - derniere_pression) > 0.5
or
                    abs(volume_f - dernier_volume) > 0.5):
                pressions_filtrees.append(pression_f)
                volumes_filtrees.append(volume_f)
                # Mise à jour du tracé
                line.set_xdata(volumes_filtrees)
                line.set_ydata(pressions_filtrees)
                ax.relim()
                ax.autoscale_view()
                plt.pause(0.05)
                derniere_pression = pression_f
                dernier_volume = volume_f
                print(f"P={pression_f:.1f} kPa | V={volume_f:.1f} mL")
        except KeyboardInterrupt:
            print("Arrêt par l'utilisateur.")
            break
        except Exception as e:
            print("Erreur:", e)
            continue
ser.close()
```

Les résultats affichés dans la fenêtre Python Shell ont été présenté sous forme d'un graphe montre l'évolution de la pression en fonction du volume corrigé Vc.

D'après la figure (III.11), il est clair que la pression augmente lorsque le volume diminue, et inversement, pour une température reste constante.

Pour les calculs, il faut ajouter le volume d'air contenu dans le tuyau, qui relie la seringue et le capteur, au volume d'air de la seringue

Le tuyau a un diamètre de 5 mm pour une longueur de 15 cm : Vair tuyau ≅ 3 mL

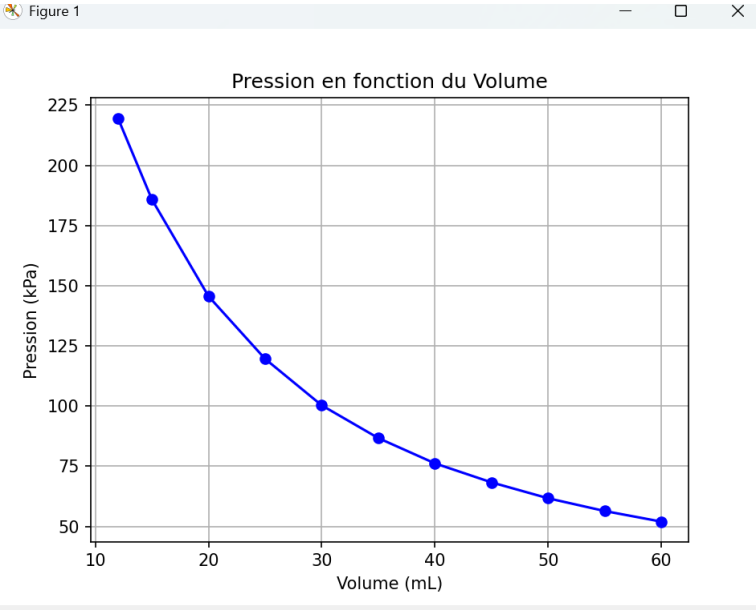



**Figure (III.11) :** la variation de la pression en fonction du volume corrigé.

Mais comme d'après la loi de Boyle-Mariotte : P = k / V (où k est une constante en kPa.L),

Il est préférable de tracer le graphe représentant P = f (1/Vc).

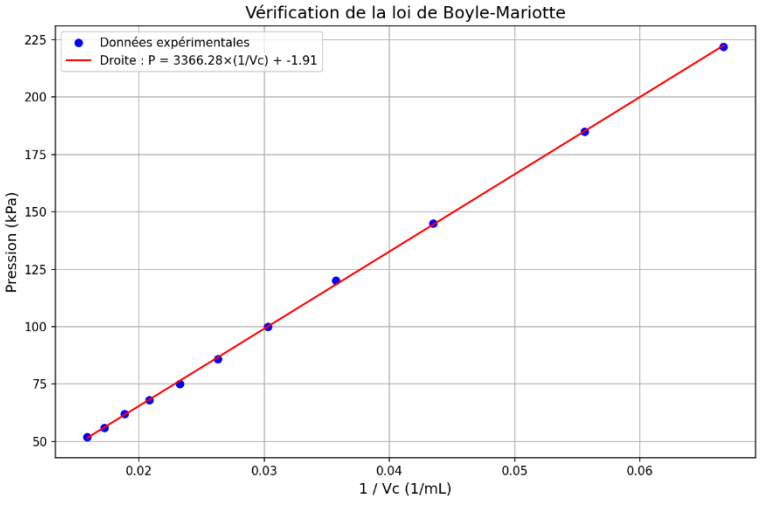



**Figure (III.12) :** la variation de la pression en fonction de l'inverse du volume corrigé Vc.

La représentation graphique de P = f(1/Vc) peut être modélisée par une fonction linéaire (Figure III.12).

On a trouvé que la valeur de la constante de la loi de Boyle-Mariotte d'environ 3,3 kPa.L approximée précédemment.

Cette expérience confirme que, dans un système fermé à température constante, le produit de la pression et du volume reste constant, ce qui est caractéristique d'un gaz parfait en transformation isotherme.

### III.3.4 Principe fondamental de la statique des fluides

Le but de cette partie est de déterminer la profondeur d'immersion dans une colonne d'eau d'un tuyau relié à un capteur de pression MPX2200AP en appliquant le principe fondamental de la statique des fluides au montage suivant :

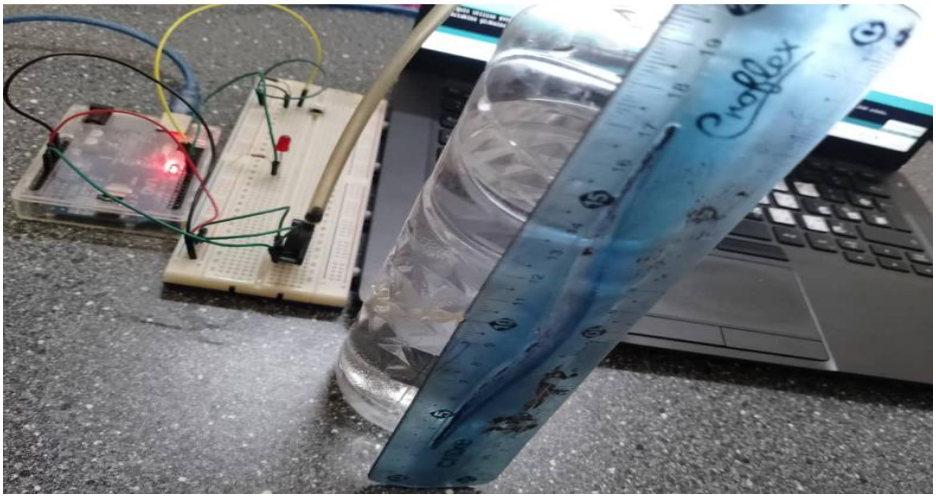


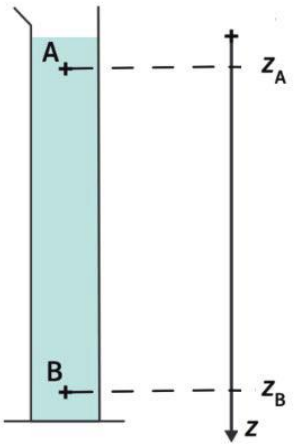
**Figure (III.13):** Montage utilisé pour mesurer de la pression en fonction de la profondeur d'immersion.

#### III.3.4.1 Description de la manipulation

Après avoir positionné le tuyau au niveau du point A, constituant l'origine du repère ($Z_A$=0), et mesuré la pression PA de ce point, on déplacera l'extrémité du tuyau (Point B) dans la colonne d'eau. Le microcontrôleur mesurera en continu la pression du point B et calculera la distance (en m) entre les deux points :

La différence de pression (en Pa) entre deux points A et B (Cf. schéma ci-contre) d'un fluide homogène au repos est égale à :

$$P_B - P_A = \rho\, g\, (Z_B - Z_A)$$




Où :

B : est en-dessous de A,

$\rho$ : masse volumique du fluide en $kg.m^{-3}$, Avec $\rho_{eau} = 1000\ kg/m^3$

$g$ : champ de pesanteur ($g = 9{,}81\ m.s^{-2}$),

$Z_B - Z_A$ : la distance entre les plans horizontaux passant par A et B en m.

### III.3.4.2 Programme en langage Arduino:

Ce code programmé, avec le logiciel Arduino IDE

```
// Déclaration des constantes et variables
const int PinSensor=0;
const int PinButton= 12;
int ValSensor = 0;
float tension = 0.0;
float Pression = 0.0;
float PRef = 0.0;
float Dz=0.0;
float OldDz = 0.0;
int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;
boolean BtnAppui = false;
// Initialisation des entrées et sorties
void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  Serial.print("Placez le tuyau a la position de reference ");
  Serial.println("et appuyez sur le bouton poussoir.");
while (BtnAppui==false){
    ValButton = digitalRead(PinButton);
    if ((ValButton == HIGH)&&(OldValButton == LOW)){
BtnAppui=true;
    delay (200);
    }
  }
```

```
   OldValButton = ValButton;
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;
  PRef = (tension / 0.02) + 10;
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}
// Fonction principale en boucle
void loop() {
  ValButton = digitalRead(PinButton);
delay(10);
if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;
if (State==1)
  {
  if (OldState == 0)
  {
   Serial.println("Mesure de la pression en cours.");
   Serial.println("");
   Serial.println ("Pression (en kPa), Profondeur (cm):");
   OldState=1;
  }
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;
  Pression = (tension / 0.02) + 10;
  Dz = 100*(Pression-PRef)/9.81;
  if (abs(OldDz - Dz)>2)
  {
    Serial.print(Pression,1);
    Serial.print(" ; ");
    Serial.println(Dz,0);
    OldDz = Dz;
  }
  delay(500);
}
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    OldState = 0;}
  }
}
```

### III.3.4.3 Résultats dans le moniteur série :

Pour commencer les mesures il faut qu'on place le tuyau à la position de référence et appuyons sur le bouton poussoir. Puis on déplaçons le tuyau et on appuyons de nouveau sur le bouton poussoir.

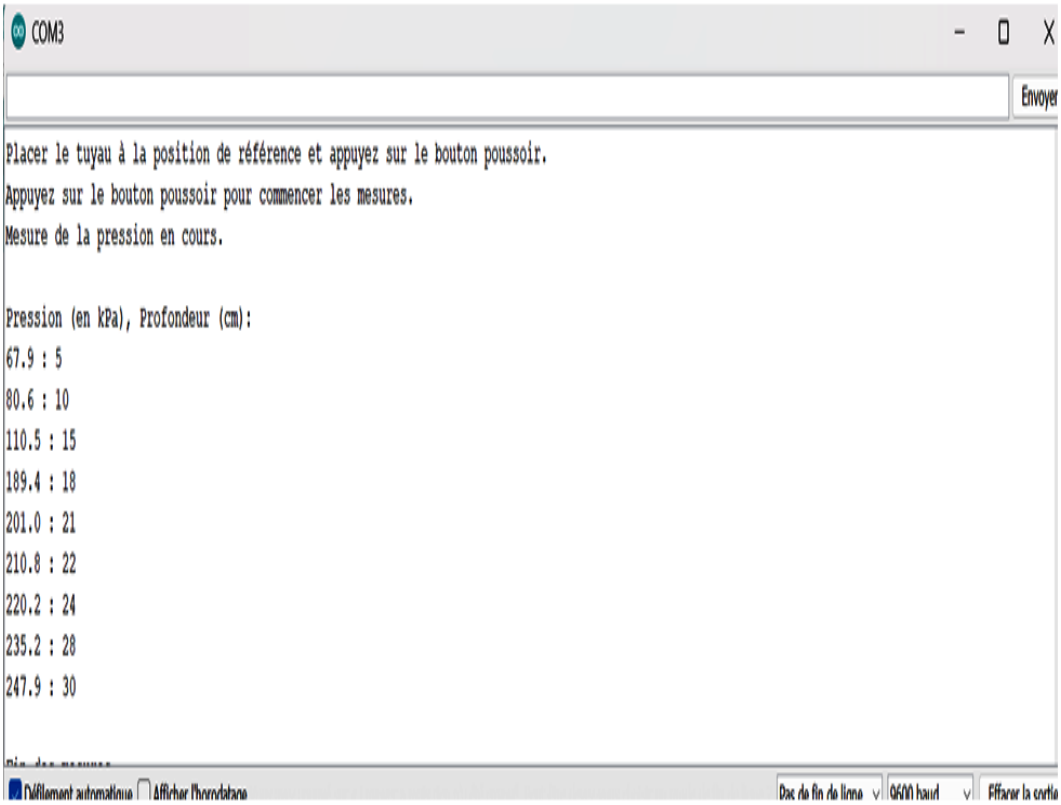


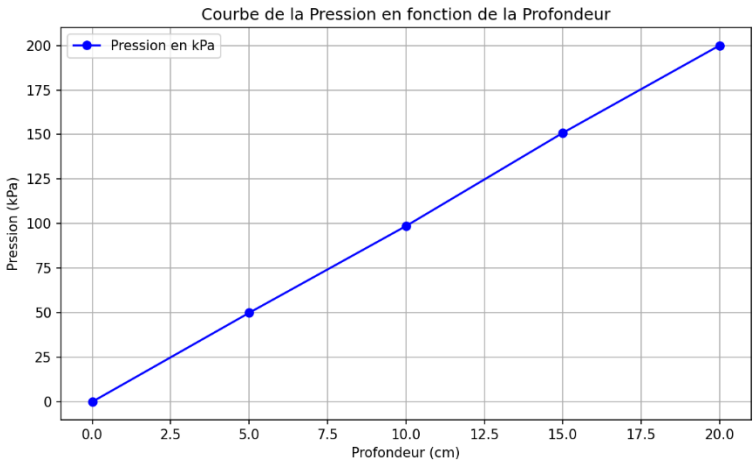

**Figure (III.14):** Résultat de mesure de la pression en fonction de la profondeur d'immersion dans l'Arduino.

### III.3.4.4 Programme en Python :

```
import serial
import matplotlib.pyplot as plt
# Configuration du port série
port = 'COM3'       # Remplacez par le port utilisé par votre Arduino
baud_rate = 9600
# Listes pour stocker les données
pressions = []
profondeurs = []
# Connexion au port série
try:
    ser = serial.Serial(port, baud_rate, timeout=2)
    print(" Connexion établie avec Arduino. Lecture des données...\n")
except:
    print(" Échec de la connexion. Vérifiez le port COM.")
    exit()
# Lecture des données depuis le port série
try:
    while True:
        ligne = ser.readline().decode('utf-8').strip()
        if ligne.startswith("Pression"):
            try:
         # Exemple de ligne: "Pression (kPa) : 120.50 | Profondeur (cm) : 12.30"
                parties = ligne.split('|')
                pression = float(parties[0].split(':')[1].strip())
                profondeur = float(parties[1].split(':')[1].strip())
                pressions.append(pression)
                profondeurs.append(profondeur)
                print(f" Pression : {pression} kPa, Profondeur : {profondeur} cm")
                # On arrête après 20 mesures
                if len(pressions) >= 20:
                    break
            except:
                print(f" Donnée invalide : {ligne}")
except KeyboardInterrupt:
    print("\n Arrêt manuel de la lecture.")
finally:
    ser.close()
# Tracé du graphe
plt.plot(profondeurs, pressions, marker='o', color='blue')
plt.title(' Pression en fonction de la profondeur')
plt.xlabel('Profondeur (cm)')
plt.ylabel('Pression (kPa)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Les résultats affichés dans la fenêtre Python l ont été présenté sous forme d'un graphe montre l'évolution de la pression en fonction de la profondeur d'immersion.

D'après la figure (III.15), il est clair que la pression augmente lorsque le volume diminue, et inversement, pour une température reste constante.




**Figure(III.15):** variation de la pression en fonction de la profondeur d'immersion.

À partir de la figure (III.15), On observe que ;

La pression augmente globalement avec la profondeur, ce qui est conforme au principe fondamental de la statique des fluides :

La pression augmente de 50 kPa tous les 5 cm de profondeur. Cela signifie que la relation entre la pression (P) et la profondeur (h) est linéaire.

La relation entre la pression et la tension de sortie du capteur est également linéaire.

Le capteur MPX2200AP fournit une tension proportionnelle à la pression mesurée. Par conséquent, la relation entre la profondeur et la tension de sortie est aussi linéaire. Cela signifie qu'il est possible de déduire directement la profondeur à partir de la tension mesurée, ce qui simplifie le traitement et la conversion des données dans un système embarqué.

## III.4 Conclusion :

Ce chapitre propose une analyse approfondie de l'utilisation d'Arduino pour la mesure de pression et les applications associées, soulignant les avantages de capteurs de pression tels que le MPX2200AP. Il guide les utilisateurs dans la création de circuits, l'écriture de code en Python et en langage Arduino, et la prise de mesures grâce à une série d'applications approfondis.

Plusieurs tâches on tété effectuées illustrent la configuration des circuits de mesure de pression, la configuration de l'Arduino pour la lecture et l'interprétation des données des capteurs, et la configuration d'avertissements visuels pour les seuils de pression. Nous abordons également des sujets plus avancés, tels que l'étalonnage des capteurs à l'aide de la loi de Boyle-Mariotte pour améliorer la précision des mesures de pression et de vérifier le principe fondamental de la statique des fluides.

Les résultats expérimentaux obtenus à l'aide du système électronique basé sur Arduino et le capteur MPX2200AP confirment la pertinence et l'efficacité de ce dispositif pour l'étude de la pression dans divers contextes.

L'ensemble des mesures démontre non seulement la fiabilité du capteur utilisé, mais aussi la capacité du montage à produire des données exploitables et cohérentes.

.

# Conclusion Générale

**Conclusion générale :**

Le travail réalisé dans le cadre de ce projet nous a permis d'explorer en profondeur le comportement des capteurs de pression, notamment le MPX2200AP, ainsi que leur interaction avec les systèmes numériques. Ces capteurs sont de plus en plus utilisés dans le domaine industriel, médical et environnemental, grâce à leur capacité à convertir une pression mécanique en signaux électriques exploitables.

Ce projet a offert l'opportunité d'étudier les variations de pression dans différentes conditions physiques, telles que le volume et la profondeur, et de comprendre comment ces variations peuvent être mesurées et analysées numériquement.

Grâce à la plateforme Arduino UNO, une solution électronique open source et polyvalente, nous avons pu concevoir un système fonctionnel capable d'acquérir des données de pression et de les transmettre à un ordinateur en temps réel. La communication entre l'Arduino et l'ordinateur hôte a été assurée par la bibliothèque PySerial en Python, ce qui nous a permis de gérer le flux de données série de manière flexible et directe. Cette interaction illustre la puissance de l'intégration entre le matériel embarqué et la programmation de haut niveau pour la mesure et l'analyse physique.

Les concepts théoriques ont été renforcés par une partie pratique dans laquelle nous avons conçu et réalisé des expériences concrètes. Celles-ci comprenaient la mesure de la pression en fonction de la tension électrique, du volume (selon la loi de Boyle-Mariotte), et de la profondeur (pression hydrostatique). Des scripts Python ont été développés pour acquérir, lustrer et représenter graphiquement les données collectées, ce qui a permis d'observer les lois physiques à l'œuvre et de valider expérimentalement le comportement du capteur.

Les résultats issus des simulations et des expériences ont démontré la linéarité et la sensibilité du capteur MPX2200AP dans divers contextes, confirmant son aptitude à être utilisé dans des environnements pédagogiques ou de recherche à faible coût. Les expériences menées ont permis non seulement de vérifier des lois physiques fondamentales, mais aussi de développer des compétences pratiques en étalonnage de capteurs, en programmation Arduino et en visualisation de données en temps réel.

La mesure précise de la pression est cruciale dans de nombreux domaines tels que la mécanique des fluides, les systèmes hydrauliques ou encore le suivi des infrastructures géotechniques. Grâce à des outils robustes comme Arduino, Python et des capteurs de précision

tels que le MPX2200AP, il est désormais possible de concevoir des systèmes efficaces et abordables pour la surveillance et l'analyse de la pression en temps réel.

Enfin, plusieurs perspectives peuvent être envisagées à l'issue de cette étude, telles que l'intégration de la transmission de données sans fil (Bluetooth, Wi-Fi), la mise en œuvre de solutions avancées de journalisation et d'analyse, ou encore l'exploration de relations physiques complémentaires, comme le couplage pression-température. Ces pistes ouvrent des perspectives prometteuses pour de futures recherches dans le domaine des systèmes de capteurs intelligents et des instruments embarqués.

**Bibliographie:**

[1] Lutgens, F.K. and Tarbuck, E.J., "The Atmosphere: An Introduction to Meteorology", 14th ed. New York: Pearson, 2019.

[2] NXP Semiconductors, "MPX2200 Series: Silicon Piezoresistive Pressure Sensors – Technical Data", Rev. 13, Eindhoven: NXP Semiconductors, Jun. 2008.

[3] G. Asch et D. George, "Les capteurs en instrumentation industrielle". Paris: Dunod, 1982.

[4] A. Diligent, "Mesures et capteurs". Paris: Dunod, 1999.

[5] J.-P. Moulin, "Mesures physiques et instrumentation". Paris: Dunod, 2008.

[6] J. Morel, "Capteurs et systèmes de mesure". Paris: Lavoisier, 2005.

[7] G. Deniélou, "Les capteurs pour l'automatisme". Paris: Dunod, 2006.

[8] M. Ledoux, "Les capteurs en automatique". Toulouse: Casteilla, 2001.

[9] J.-P. Varlan, "Capteurs et Instrumentation". Paris: Dunod, 2003.

[10] H. Fanet, "Microcapteurs et MEMS". Paris: Dunod, 2004.

[11] G. Asch et D. George, "Les capteurs en instrumentation industrielle". Paris: Dunod, 1982.

[12] R. Boukrouh, "Instrumentation industrielle et capteurs de pression". Mémoire de Master, Université de Constantine 1, Constantine, 2018.

[13] N. Belkheir, "Étude d'un capteur de pression intelligent". Mémoire de Master, Université de Blida, Blida, 2019.

[15] A. Labani, "Simulation de capteurs piézorésistifs appliqués au domaine biomédical". Mémoire de Master, Université de Sétif, Sétif, 2020.

[16] W. Meziani, "Étude comparative des capteurs MEMS (capacitif, piézorésistif...)". Mémoire de Master, Université de Tlemcen, Tlemcen, 2020.

[17] M. Khelladi, "Capteurs MEMS pour systèmes embarqués". Mémoire de Master, Université de Boumerdes, Boumerdes, 2020.

[18] K. Fekih, "Technologie MEMS et leurs applications industrielles". Mémoire de Master, Université de Sétif, Sétif, 2021 .

[19] Y. Bouacida, "Développement de microsystèmes MEMS pour la mesure de pression". Mémoire de Master, Université de Batna, Batna, 2022.

[20] S. Mekki, "Capteurs intelligents et traitement du signal". Mémoire de Master, Université de Sétif, Sétif, 2019.

[21] A. Guerroua, "Capteurs piézorésistifs MEMS et techniques de compensation". Mémoire de Master, Université de Boumerdes, Boumerdes, 2019.

[22] ARDUINO. Arduino Uno – Spécifications techniques. Disponible sur : https://www.arduino.cc/en/Guide/ArduinoUno .

[23] Arduino Uno – Spécifications techniques. Université des Sciences et de la Technologie d'Oran (USTO), Algérie, 2024.

[24] BENHAMOU, K., Les microcontrôleurs dans les systèmes embarqués, Faculté des Sciences, Université de Tlemcen, 2023.

[25] ALI, M., Programmation et communication USB sur les microcontrôleurs, Revue Algérienne des Technologies et des Systèmes Embarqués, Vol. 15, No. 2, 2022

[26] ZERROUKI, M., Systèmes embarqués avancés : Les microcontrôleurs ARM Cortex, Ed. Université d'Alger, 2021.

[27] BOUHLOU, T., Applications des systèmes IoT avec Arduino, Revue des Innovations Technologiques, Vol. 10, No. 3, 2023.

[28] Guide pratique des cartes Arduino, Centre de Recherche en Informatique et Automatique d'Alger, 2022.

[29] HADJADJ, K., Introduction aux microcontrôleurs et architectures RISC, Alger : Presses Universitaires d'Alger, 2021.

[30] LAMARI, A., Les protocoles de communication dans les systèmes embarqués, Université de Constantine, 2020.

[31] TAHAR, F., Développement logiciel pour microcontrôleurs Arduino, Éditions Universitaires Algériennes, 2022.

[32] BELHADJ, R., Les bibliothèques Arduino et la gestion des composants, Revue Algérienne des Sciences Appliquées.

[33] ARDUINO. Fonctions standardisées pour la gestion des opérations courantes. Disponible sur : https://www.arduino.cc/en/Reference.

[34] Protocole Firmata et intégration avec Arduino, Journal Algérien des Sciences Informatique, 2022.

[35] YAHIAOUI, D., Communication série et bibliothèques en Python, Université de Boumerdès, 2020.

[36] PYFIRMATA. Communication Arduino-Python via Firmata. Disponible sur : https://pyfirmata.readthedocs.io/ .

[37] PYFIRMATA. Communication Arduino-Python via Firmata. Disponible sur : https://pyfirmata.readthedocs.io/.

[38] Paul Deitel, Python for Programmers, Copyright © 2019 Pearson Education, Inc.

[39] Python Software Foundation, Python Programming Language, https://www.python.org/ (accessed 2024).

[40] M Fraden, J. Handbook of Modern Sensors: Physics, Designs, and Applications. 5th ed., Springer (2016).

.

## ملخص :

في هذا العمل، قمنا بقياس الضغط المطلق باستخدام لوحة أردوينو، بالاعتماد على المستشعرMPX4250AP ، الذي يتميز بحساسية تبلغ 20 ميلي فولت/كيلو باسكال، وضمن مجال قياس يتراوح بين 20 و250 كيلو باسكال )أي أن الجهد الخارج يتراوح بين 2.0 و8.4 فولت(. وهذا ما يجعله مناسبًا تمامًا للتكامل مع بيئة أردوينو.ونظرًا لمحدودية شاشة العرض التسلسلية، التي تتيح فقط عرضًا لحظيًا للقيم، فقد اعتمدنا على برمجية Python تقوم باستقبال البيانات المرسلة من طرف الأردوينو عبر المنفذ التسلسلي، ثم عرضها على شكل رسم بياني لتمكين تحليل أعمق.كما تم إجراء عدة تطبيقات عملية باستخدام المستشعر MPX2200AP، منها قياس الضغط المطلق، والتحقق من ضغط العتبة باستخدام مؤشر ضغط، بالإضافة إلى التحقق من قانون بويل-ماريوت، والمبدأ الأساسي لميكانيكا الموائع الساكنة.وقد أظهرت نتائج المحاكاة استجابة خطية وحساسية عالية لمستشعر MPX2200APفي سياقات متنوعة، مما يثبت صلاحيته للاستخدام في البيئات التعليمية والبحثية منخفضة التكلفة.



**الكلمات المفتاحية** :مستشعر الضغط المطلق؛ أردوينو؛ بايثون؛MPX2200AP ؛ محاكاة.



## Résumé

Dans le cadre de ce travail, nous avons mesuré la pression absolue à l'aide d'une carte Arduino, en utilisant le capteur MPX4250AP, dont la sensibilité est de 20 mV/kPa pour une plage de mesure comprise entre 20 et 250 kPa (correspondant à une tension de sortie entre 0,2 V et 4,8 V). Ce capteur s'avère parfaitement adapté à une utilisation avec Arduino.Étant donné les limitations du moniteur série, qui ne permet qu'un affichage en temps réel des données, nous avons eu recours à un script Python chargé de récupérer les données transmises par l'Arduino via le port série, puis Plusieurs expérimentations de les représenter graphiquement pour une analyse approfondie pratiques ont été menées à l'aide du capteur MPX2200AP, notamment la mesure de la pression absolue, la vérification du seuil de pression à l'aide d'un indicateur, ainsi que la validation de la loi de Boyle-Mariotte et du principe fondamental de la statique des fluides.Les résultats obtenus par simulation ont confirmé la linéarité et la sensibilité du capteur MPX2200AP dans divers contextes, démontrant son adéquation pour des applications pédagogiques ou de recherche à faible coût.



**Mots-clés** : capteur de pression absolue ; Arduino ; Python ; MPX2200AP ; simulation.



## Abstract

In this work, we measured absolute pressure using an Arduino board and the MPX4250AP pressure sensor, which offers a sensitivity of 20 mV/kPa over a measurement range from 20 to 250 kPa (i.e., an output voltage between 0.2 V and 4.8 V). This makes it particularly well-suited for integration with Arduino platforms.

Due to the limitations of the serial monitor, which only allows real-time visualization, we employed a Python script to retrieve data transmitted from the Arduino via the serial port and graphically display it for enhanced analysis.Several practical applications were implemented using the MPX2200AP sensor, including absolute pressure measurement, threshold pressure verification using a pressure indicator, and validation of Boyle-Mariotte's law and the fundamental principle of fluid statics.

Simulation results demonstrated the linear response and sensitivity of the MPX2200AP sensor across various contexts, confirming its suitability for low-cost educational and research environments.



Keywords: absolute pressure sensor; Arduino; Python; MPX2200AP; simulation.