

Méthode expérimentale d'Etude de la compression des grains et des poudres en condition presque oedométrique par mesure photo-élastique des contraintes dans le cylindre de confinement

English Title :

A simple cheap experimental method to study boundary effect on oedometric compression of grains and powders using photo-elastic detection

F. Douit & P. Evesque

Lab. MSSMat, UMR 8579 CNRS, CENTRALESUPELEC
92290 CHATENAY-MALABRY, France, e-mails: pierre.evesque@ecp.fr ;
frederic.douit@centralesupelec.fr

Abstract:

It is proposed a new low-expensive device using photoelasticity to analyze the stress variation in a cylindrical oedometer using its elasticity to determine stress at wall. The grains or powders are contained in a cylindrical hole drilled in a photo-elastic material closed by two pistons. Radial stress is measured though photo-elasticity. The compression can be generated by a press or a jack ; it is controlled, monitored and recorded via a smartphone or a pc through an Arduino set-up recording the applied vertical stress and displacements, while the video records the photoelastic patterns, when the material is compressed or decompressed. Such a method may be useful to study local distribution of stress, its sensitivity to boundary conditions and their evolution, to the exact stress direction....

Pacs # : 5.40 ; 45.70 ; 62.20 ; 83.70.Fn

En mécanique, on sait depuis longtemps que les conditions aux limites peuvent jouer un rôle important. C'est comme cela qu'on amarre un paquebot à la bitte d'amarrage du quai : quelques tours de cordes suffisent à bloquer le bateau, grâce au frottement solide. Les effets de rivet, les contre-écrous sont d'autres moyens d'imposer un blocage lié à l'action d'un frottement solide.

La mécanique des milieux granulaires et des poudres elle aussi est dominée en partie par ce frottement solide [1, 2]. L'un d'entre nous (avec de Gennes) [3, 4] a montré il y a quelques temps l'importance de cet effet pour comprendre la compression dans un silo. Mais ce qui se passe dans un silo, peut aussi se passer dans un essai oedométrique, ou dans d'autres confinements.

Par exemple, bien souvent l'essai oedométrique n'est regardé que comme un système simple [5, 6, 7], où la loi rhéologique joue le rôle prédominant. Est-ce si sûr ? Les conditions aux limites, i.e. à l'interface moule-poudre, ne peuvent-elles jamais donner lieu à des comportements anormaux ? Dans ce cas il semble que ceux-ci devraient être rares étant donné que peu d'articles traitent le sujet. Nous avons décidé de nous poser cette question : quelle est l'évolution des conditions aux limites lors du chargement d'un silo, lors d'un essai oedométrique...Et de la résoudre expérimentalement.

De la même façon, le frottement solide peut aussi jouer un rôle dans la fabrication des compacts [8, 9, 10, 11] (pharmaceutiques par exemple) ; ou dans le broyage ou le concassage de matériaux pour lesquels on utilise les poussées entre grains ou avec le bord de la meule ou du concasseur pour briser les grains. On se sert alors des autres grains comme un mors d'étau vis-à-vis du grain qu'on brise...

Par ailleurs, de nombreux matériaux structurés un peu complexe, à texture, du style lamellé, avec des interfaces entre couches différentes, avec des fils, ou des poreux à plusieurs composants, à structures orientées peuvent se comporter différemment suivant l'orientation des lamelles, des interfaces et du système de compression; de même le moule peut avoir une action différente suivant son orientation par rapport à la texture du matériau...

Nous proposons ici une expérience simple à mettre en œuvre pour étudier la zone d'interface (ou de contact) entre le moule et le milieu qu'on comprime. Le moule sera toujours supposé à comportement élastique linéaire (de toutes les façons on s'apercevra très vite au démontage du moule si cette hypothèse est restée vérifiée ou non, suivant l'état final du moule). La forme des contraintes et leur évolution dans le moule dépendra alors directement des conditions aux limites qu'on impose (sans vraiment les connaître en général) au milieu confiné, écroui, plastifié ou... On pourra ainsi montrer si ce matériau est soumis à des forces homogènes ou non, si on peut considérer qu'il est en déformation homogène, ou si une structure apparaît au niveau des conditions aux limites...

L'expérience que l'on propose ici est prévue pour l'étude des milieux granulaires ; mais rien n'empêche de l'utiliser dans d'autres systèmes (comme ceux listés ci-dessus. Nous détecterons les contraintes à l'interface milieu-moule grâce à sa trace dans le moule, en utilisant la photoélasticité pour détecter les contraintes dans le moule, et leurs variations spatiales et temporelles au cours de la compression (décompression).

En général, on considère que les poudres et les milieux granulaires ont des comportements pour lesquels la cohésion, le frottement solide jouent des rôles importants. Grains et poudres sont définis comme étant un ensemble de particules dont le comportement mécanique est quasi similaire à compression faible, de telle sorte que les mécaniciens des sols utilisent les mêmes concepts mécaniques pour les

caractériser [12]. Ceux-ci considèrent en effet que les contraintes stériques et le frottement solide entre les grains sont responsables de leurs propriétés mécaniques particulières (comme la dilatance).

A plus forte compression, les grains peuvent se déformer (fluer) ou se briser. Le système granulaire devient ainsi très cohérent, soit par frittage, soit simplement par adhésion forte entre les grains, dilatance joue alors un rôle probablement négligeable.

L'approche par photoélasticimétrie nous donne un nouveau point de vue. En visualisant les contraintes reportées sur le matériau servant de moule, nous pouvons représenter la cartographie des variations de contraintes aux interfaces avec la poudre. Par cette méthode, nous mesurons plus localement, et pouvons repérer des informations jusque là inaccessibles.

A noter aussi que cette expérience pourra permettre d'étudier le couplage entre le milieu et le moule élastique. Utiliser en science numérique, cela deviendra un outil de base du numéricien pour contraindre les programmes numériques aux interfaces et vérifier la réalité de leurs hypothèses.

La réalisation de cette expérience est simple et peu coûteuse en temps et en argent ; mais il faut quand même un peu de matériel. Il faut pouvoir usiner des moules en plastiques transparents, pouvoir polir leurs interfaces au besoin ; réaliser un banc optique assez stable et de mise en œuvre simple. Définir les capteurs de mesures (pression, déplacement) qui caractérisent l'expérience au niveau macroscopique, comme d'habitude. Leur acquisition se fait ici par une carte électronique de marque Arduino. Un montage électronique a été nécessaire pour adapter les signaux. Pour piloter l'expérience aisément, un programme C++ a été développé.

Les mesures locales du champ de contrainte se feront par photoélasticimétrie en filmant l'évolution des contraintes en cours de compression entre polariseurs croisés. Ces expériences restent du domaine quasi-statique car l'évolution des contraintes est très lente par rapport à la vitesse du son. On pourrait vouloir aller plus vite ; il faudrait utiliser une vidéo rapide. La mesure optique dépend de l'intégrale sur le parcours optique des retards optiques locaux ; on rappellera cela dans la première partie.

Les films permettent d'étudier les phénomènes observés à posteriori ; ils peuvent être enregistrés, voir compressés en temps réel sur le mini pc (Arduino).

Les films obtenus à partir des expériences de photoélasticité sont souvent esthétiquement beaux, car ils révèlent quelque chose de réel qu'on a un peu de mal à appréhender. Pour cela ils font rêver. Nous invitons donc le lecteur à reproduire cette expérience, et peut-être tombera-t-il en transe ou en rêve, comme le dit Gaston Bachelard dans « *la flamme d'une chandelle* » (aux éditions Les Presses universitaires de France, 1961, à relire aussi au besoin).

Avant d'entreprendre la réalisation de cette expérience, il a fallu revoir les fondamentaux théoriques nécessaires à la compréhension de la photoélasticité, comme la biréfringence optique ou la mécanique élastique.

Aussi, nous verrons qu'il est nécessaire d'instrumenter l'expérience de capteurs et d'une carte électronique de commande, complétée par un programme en C++.

Pour finir, une analyse des résultats expérimentaux sera effectuée, tant au niveau des capteurs que du traitement vidéo.

1. Quelques rappels

1.1. Quelques rappels d'élasticité et d'optique

Nous passerons sur le comportement mécanique des milieux granulaires, qui est en partie ce que l'on cherche à étudier, mais dont on n'a pas besoin de connaître. Nous donnons simplement une liste bibliographique non exhaustive. Nous voulons ici remonter aux conditions aux limites imposées sur le moule par le matériau. Ce moule est élastique. Les équations de la mécanique des milieux continus impose donc l'état des contraintes $\sigma_e(x,y,z)$ en tout point de coordonnées (x,y,z) du matériau, imposée par l'élasticité [13]. Ces contraintes élastiques s'ajoutent aux contraintes existantes $\sigma_o(x,y,z)$ (σ_o dépend de x,y,z si le matériau n'est pas parfaitement homogène, mais est précontraint).

Une première chose est donc de caractériser l'état initial des contraintes ; le reste, i.e. $\sigma_e(x,y,z)$, peut être prédit théoriquement par la théorie de l'élasticité, soit de façon théorique, soit de manière numérique en supposant les conditions aux limites. Il existe pour cela des codes par éléments finis qui résolvent facilement ce problème ; nous mêmes, nous avons utilisé le code COMSOL:

On peut donc connaître en tout point $\sigma(x,y,z) = \sigma_e(x,y,z) + \sigma_o(x,y,z)$

Bien entendu, nous voudrions faire le processus inverse : remonter aux contraintes de surfaces connaissant $\sigma(x,y,z)$ en tout point intérieur du moule. Avant de discuter si l'on est capable réellement de remonter aux conditions limites exactes connaissant $\sigma(x,y,z)$; posons-nous d'abord la question de la détection des champs de contrainte par la photoélasticité.

La contrainte $\sigma(x,y,z)$ déforme localement le matériau élastique de sorte que cela modifie les paramètres de propagation des champs électriques et magnétiques dans le matériau. Pour un matériau élastique transparent cela se traduit par une variation de l'indice de réfraction :

$$n(x,y,z) = n(\sigma(x,y,z)) = n_o + \Delta n = n_o + \delta n(\sigma_e(x,y,z)) + \delta n(\sigma_o(x,y,z)),$$

ou n_o est l'indice du matériau et $\delta n(\sigma)$ sa variation en fonction de la contrainte appliquée. $\delta n(\sigma)$ est petit et varie linéairement avec σ au premier ordre ; nous verrons aussi que δn dépend de la direction de propagation et de la polarisation de la lumière, car la polarisabilité du matériau dépend de la direction du champ électrique par rapport aux contraintes appliquées.

Un pinceau lumineux qui traverse le milieu élastique va donc subir/tester en chaque point un indice qui dépendra de la polarisation de l'onde donc de la valeur et de l'orientation de la contrainte. Le retard optique de ce pinceau subira sera la somme des retards locaux infinitésimaux. Il dépendra donc de la polarisation de l'onde, de la distribution des contraintes dans le matériau.

1.2. Quelques rappels Couplage mécanique/optique ou Photoélasticimétrie:

Si l'on fait propager un faisceau lumineux (formés de pinceaux) dans un matériau élastique sous contrainte, chaque pinceau subira aussi en tout point une déflexion, i.e. variation locale de direction de propagation due à la variation de n local ; mais cet effet est mineur car la variation de n est faible et relativement lente à l'intérieur du matériau dès que l'on se trouve un peu loin des sources d'inhomogénéités de contrainte les moyennes. Le système restera très faiblement diffusant, et les plans d'ondes resteront des plans d'onde.

Par exemple, si l'on considère un faisceau se propageant suivant z , son retard optique pourra être dévié légèrement par la traversée du moule, et son retard optique pourra dépendre légèrement de la position (x,y) de sa traversée ; par contre ce faisceau aura un retard optique bien défini, dépendant de la position (x,y) , mais sans grande fluctuation locale car le moule élastique effacera les fluctuations de n dès qu'on loin d'une source hétérogène de contrainte. Ainsi, le champ de contrainte local ne varie vite qu'à proximité des parois, sources des contraintes imposées, beaucoup moins vite plus loin, « homogénéisée » par l'élasticité du moule. Comme le pinceau initial peut être considéré comme homogène sur une bonne largeur (éclairage uniforme), ces effets ne seront pas visibles, et pourront être négligés par la suite.

Par contre cet effet peut devenir important lorsqu'on utilise l'interférence entre deux faisceaux de polarisations différentes. Car chaque polarisation réagira selon son orientation par rapport à la contrainte principale majeure. Ainsi, dans un milieu sous contrainte anisotrope, les polarisations optiques qu'il faut choisir sont directement liées à la direction de propagation utilisée, et aux directions des axes principaux des contraintes ; ainsi les deux indices de réfractions des deux polarisations possibles seront donc différents, car les valeurs des contraintes principales y seront différentes. Il s'en suivra que le déphasage optique de chaque polarisation pourra être différent. Comme ce déphasage joue un rôle important dans la différence des retards optiques lorsqu'on réunit les deux ondes à la sortie, on pourra voir des effets importants, pour un faisceau se dirigeant selon l'axe z le déphasage optique varie notablement par rapport à la longueur d'onde λ de l'onde lumineuse test.

Pour ceux qui aiment les math, cela se traduit par

$$\Delta e_{Pi} = \int_{\text{trajet } i} \delta n_{Pi}(x,y,z) dz$$

Où le trajet est le trajet lumineux considéré à l'intérieur du moule.

Plaçons le système photoélastique entre deux polariseurs croisés, comme sur la Fig. 1-1, l'un, P_0 , appelé polariseur, l'autre, P_1 , analyseur. Ils polarisent l'onde en P_0 et P_f . L'onde polarisée à l'entrée par le polariseur P_0 se propage dans le milieu photoélastique en se décomposant à tout instant sur les axes du milieu biréfringent, et en se recombinant, chaque projection subissant un incrément de décalage optique différent de l'autre puisque l'indice de réfraction est différent suivant les deux polarisations. De part l'élasticité ces axes évoluent continument. A la sortie du milieu biréfringent, l'analyseur remélange ces polarisations en les combinant de manière à garder l'information des deux décalages optiques.

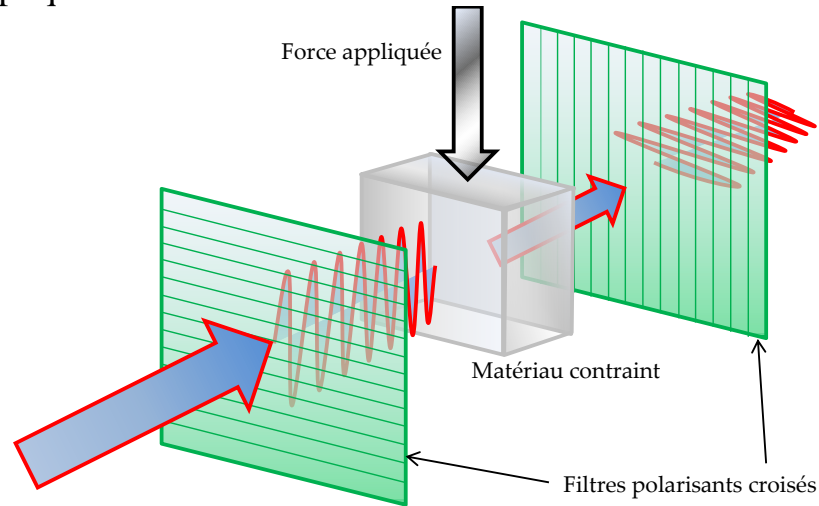


Figure 1-1 : banc photo-élastique.

Notons α l'angle entre le polariseur P_1 et l'axe principal E_1 du matériau dans le plan défini par les vecteurs (P_1, P_2) . Dans la base (E_1, E_2) , les vecteurs P_1, P_2 des polariseurs P_1, P_2 s'écrivent $P_1 = (\cos\alpha, \sin\alpha)$; $P_2 = (-\sin\alpha, \cos\alpha)$. Le polariseur P_1 laisse passer l'onde $P_1 \sin(\omega t)$, qui se transforme à l'entrée du matériau en se projetant sur les axes du matériau :

$$P_1 \sin(\omega t) = \cos(\alpha) E_1 \sin(\omega t) + \sin(\alpha) E_2 \sin(\omega t)$$

Soit $\varphi = \int \pi (n_1 + n_2) dz / \lambda$, intégrale prise sur l'épaisseur du matériau, et $\delta\varphi = \int \pi (n_1 - n_2) dz / \lambda$, le champ électrique de l'onde à la sortie du matériau s'écrit :

$$E = \cos(\alpha) E_1 \sin(\omega t + \varphi + \delta\varphi) + \sin(\alpha) E_2 \sin(\omega t + \varphi - \delta\varphi)$$

Le polariseur P_2 ne laisse passer qu'une partie de ce champ, celle qui est parallèle à P_2 . Soit $P_2 \cdot E = -\sin(\alpha)\cos(\alpha)\sin(\omega t + \varphi + \delta\varphi) + \sin(\alpha)\cos(\alpha)\sin(\omega t + \varphi - \delta\varphi)$ que l'on peut écrire, puisque $\sin(\alpha)\cos(\alpha) = \frac{1}{2} \sin(2\alpha)$ et que $\sin(\omega t + \varphi - \delta\varphi) - \sin(\omega t + \varphi + \delta\varphi) = 2\cos(\omega t + \varphi) \sin(\delta\varphi)$.

L'intensité lumineuse I issue de ce 2ème polariseur est donc proportionnelle $I = (P_2 \cdot E)^2$ au carré du produit scalaire de $P_2 \cdot E$. Soit

$$I = (P_2 \cdot E)^2 = [\sin(2\alpha) \sin(\delta\varphi)]^2$$

avec $\delta\varphi = \int \pi (n_1 - n_2) dz/\lambda$, et $\varphi = \int \pi (n_1 + n_2) dz/\lambda$, intégrales prises sur l'épaisseur e du matériau.

On note que si le matériau n'est pas déformé, I est strictement nulle. C'est normal puisque Les polariseurs sont en position croisé (ou perpendiculaires l'un à l'autre).

2. Mise en place de l'expérience

2.1. Matériel d'instrumentation

2.1.1 Presse mécanique

Le système permettant d'exercer une pression de plus de 100 MPa, est constitué d'une presse mécanique de marque M&O représentée ci-dessous en Figure 2-1 :



Figure 2-1 : Presse mécanique.

Elle utilise un moteur triphasé pour faire tourner une vis sans fin activant un plateau horizontal en translation linéaire verticale. Elle peut fournir au maximum une force de 50 kN. Cette presse est surmontée d'une poutre horizontale sur laquelle sont fixés un piston vertical et son capteur de mesure de force. La hauteur de la poutre est ajustable manuellement. Le diamètre du piston appliquant la force est de 12 mm, sa surface est donc de 1,13 cm². La pression maximum mesurée qu'il peut appliquer est donc de 442 MPa.

Le déplacement du plateau est activé par un moteur triphasé disposant d'une commande basse tension. La montée se fait par une alimentation 0/10V et la descente -10/0V. La tension envoyée règle la vitesse. Pour cela, il a fallu créer un câblage adapté aux commandes souhaitées. Nous avons d'une part introduit un switch qui permet de passer manuellement en « montée » (0/10V) ou en « descente » (-10/0V), d'autre part, la vitesse est ajustée via un potentiomètre 10 tours avec un maximum de 100 micromètres par seconde. Par ailleurs, un câblage parallèle a été ajouté, qui disconnecte cette commande, et pilote la presse par une

carte électronique de pilotage Arduino qui délivre la tension de sortie à appliquer en 8 bits. Par conséquent, la vitesse pourra varier par pas de 0,8 microns par seconde.

Afin de sécuriser son utilisation, le contrôle du relais de puissance a été câblé d'un bouton marche, un bouton arrêt, un bouton d'arrêt d'urgence et d'un voyant de mise sous tension. Deux capteurs fin de course sont placés respectivement en position haute et basse du plateau. L'un coupe l'alimentation en montée tandis que l'autre coupe l'alimentation en descente. Cela laisse la possibilité de débloquer la presse.

A noter qu'il est possible de substituer la presse par un cric, car certains peuvent soulever 2 à 5 tonnes. Il suffira de faire une armature autour pour contenir l'expérience, et la mesure de déplacement du cric pourra se faire par voie optique à l'aide d'un caméscope de bonne résolution.

2.1.2 Les capteurs

Afin de mesurer la pression appliquée, la presse est équipée d'un capteur de force de marque FGP, modèle FN3042 (première page en annexe 1), présenté ci-dessous en Figure 2-2. Il peut mesurer jusqu'à 50 kN, avec une incertitude de mesure de 72 N (cela est suffisant compte tenu de notre but). Il intègre un pont de Wheatstone et est habituellement couplé à un conditionneur d'instrumentation standard. Dans notre cas, afin de réduire les coûts et d'obtenir un meilleur contrôle sur l'instrument, il sera branché directement à la carte Arduino qui servira de conditionneur. Il dispose d'une sortie 4 fils, deux pour l'alimentation du pont et deux pour récupérer le signal de mesure.



Figure 2-2 : Capteur de force FN3042.

Il est fixé en haut de l'armature de la presse, le piston de compression est vissé en son centre.



Figure 2-3 : Capteur de déplacement CLP13.

Le distance parcourue par le piston est mesurée par un capteur de déplacement de type potentiométrique de marque Megatron, modèle CLP13 (première page en annexe 2), présenté en Figure 2-3. Il possède une résolution de 10 micromètres et sa course est de 100 millimètres. Toutefois, il sera utilisé et étalonné sur un déplacement de 10 millimètres au total, mais un décalage de zéro sera possible électroniquement. L'ensemble est aussi amplifié et acquis par la carte Arduino.

Il est fixé sur le côté du plateau mobile et sa pointe est en contact avec l'armature de la base de la presse.

2.2. Electronique

2.2.1 Acquisition

2.2.1.1 Carte Arduino

Dans le domaine de l'acquisition de données et du pilotage d'actionneurs par carte programmable, une évolution remarquable a eu lieu, donnant naissance à différents modèles de cartes polyvalentes, puissantes et peu onéreuses. Elles sont accessibles au grand public, ce qui permet d'avoir une documentation importante, une multitude d'applications éprouvées et un suivi en constante évolution. Les constructeurs développent continuellement de nouveaux modèles ou modules adaptables pour répondre aux attentes des utilisateurs.

En effet, l'une des forces de ces cartes est le fait qu'elles peuvent accueillir une grande variété de modules additionnels (appelés « shield ») permettant de piloter tout type d'appareillage électrique ou électronique. Il est possible de gérer des moteurs continus ou pas à pas, de concevoir de l'éclairage intelligent ; tout ce qui est commandable par une interface électrique est accessible. Au niveau de l'acquisition, n'importe quel capteur peut être branché, le coût de certains composants comme les accéléromètres ou gyroscopes ayant fortement baissé suite à l'émergence de l'électronique embarquée au sein des smartphones, il est maintenant possible de recueillir une multitude de données. En y connectant une caméra, des capteurs de mouvement ainsi que de petits moteurs, des étudiants créent de véritables robots performants et les confrontent lors de compétitions internationales. Cela est devenu plus aisé car la programmation est simplifiée contrairement aux premières puces programmables. De plus, le logiciel de programmation est libre de droit et ne nécessite qu'une connaissance basique de l'informatique. Leur utilisation n'est plus réservée aux professionnels, elles s'intègrent dans des systèmes domotiques d'où leur incroyable succès. Rappelons que leur vitesse de calcul élevée et la possibilité de raccorder toutes sortes de périphériques, leur confèrent une puissance équivalente à un ordinateur standard, avec un encombrement réduit et une consommation électrique faible. Il sera alors possible par exemple, de faire de la compression vidéo depuis une caméra et de stocker les données sur une carte mémoire en toute autonomie et de les piloter via un smartphone.

Il existe plusieurs constructeurs, comme Xbee, Mbed, Adafruit, Raspberry ou encore Arduino... J'utiliserai une carte de ce dernier (cf Figure 2-4) pour mon expérience.

Voici ses principales caractéristiques :

- Horloge de 16 MHz.

- 8 bits en sortie.

- 10 bits en entrée.

- 54 entrées/sorties numériques dont 15 en PWM (PWM = Pulse Width Modulation ou modulation de largeur d'impulsions en français). On joue sur le rapport cyclique du signal pour obtenir la valeur moyenne de la tension désirée (0 à 5 V). De base, la fréquence est de 500 Hz, on peut la monter jusqu'à 16 kHz, ceci est important lorsque l'on pilote un moteur continu).

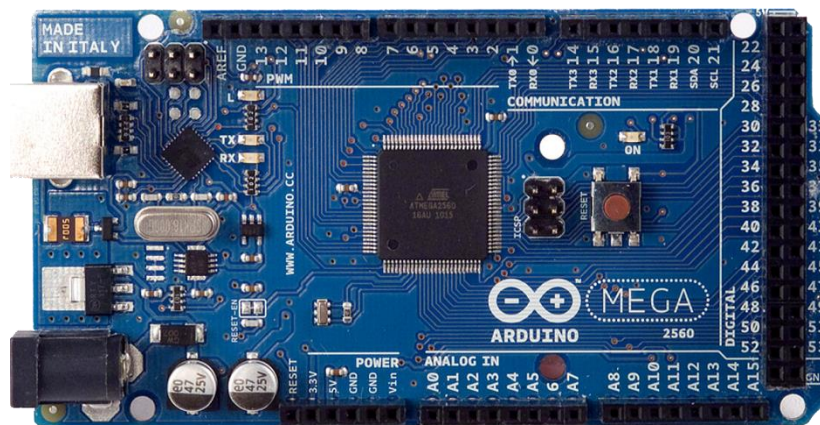


Figure 2-4 : Arduino mega 2560.

- 16 entrées analogiques de 0 à 5 V.

- Fourni une source de 5 V et 3,3 V.

- Courant max de 40 mA.

- Communication avec le pc par connexion usb avec le protocole série ou par bluetooth pour le cas d'un smartphone.

- Alimentation par port usb ou externe.

- Coût indicatif : 30 euros (elle m'a été généreusement offerte par le Laboratoire d'informatique et des systèmes avancés).

Avec une acquisition sur 10 bits, nous avons donc une résolution de $5 \text{ V}/1024 = 4,9 \text{ mV}$.

2.2.1.2 Module Bluetooth



Figure 2-5 : Module bluetooth HC-06.

Afin de s'affranchir du câble usb et de la dépendance d'un ordinateur, l'utilisation d'un module bluetooth s'avère très utile. Il est simple à mettre en œuvre, son protocole de communication reste celui du port série, et le pilotage peut désormais se faire via un smartphone ou tablette tournant sous Android. A noter qu'il est tout à fait possible d'utiliser du matériel Apple ou un système linux.

Le modèle choisi est celui de Figure 2-5 ; c'est le HC-06 de chez Sunfounder, sa norme bluetooth est la 2.1. (coût indicatif : 13 euros).

Il dispose de 4 broches :

- 2 des broches servent pour l'alimentation, connectées au +5V et GND de l'Arduino.
- 2 autres servent pour la communication, connectées aux pins TX et RX de l'Arduino.

Ce module est vendu tel quel et est prêt à l'emploi. Cependant, le reste du matériel qui équipera la carte Arduino doit faire l'objet de sélection de composants et d'un montage particulier sous la forme d'un « shield » connecté sur le dessus de la carte. L'ensemble forme un kit d'expérience pour Arduino.

Sélectionnons les pièces (II.2.2 & II.2.3) avant de s'intéresser à l'intégration globale (II.2.4).

2.2.2 Potentiomètre de commande

La presse se pilotant par une tension comprise entre -10 et +10 V et afin de pouvoir la commander par l'Arduino, il a fallu utiliser un potentiomètre numérique. Il sera intégré avec les composants nécessaires à son bon fonctionnement, sur le « shield » du kit Arduino.

Voici le modèle choisi (Figure 2-6), le MCP41HV51 de chez Microchip (première page en annexe 3) (coût = 1,50 €, fourni gratuitement par Microchip). Il se contrôle en 8 bits et sa gamme de tension va de - 18 V à + 18 V, il est donc parfaitement adapté.

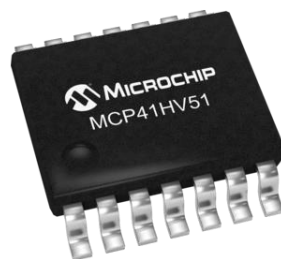


Figure 2-6 : Puce MCP41HV51.

Pour commander cette puce il faut se référer à sa documentation technique. Elle est connectée par 14 broches ; leur raccordement à la carte Arduino est schématisé par la Figure 2-7 et est expliqué par le Tableau 2-1.

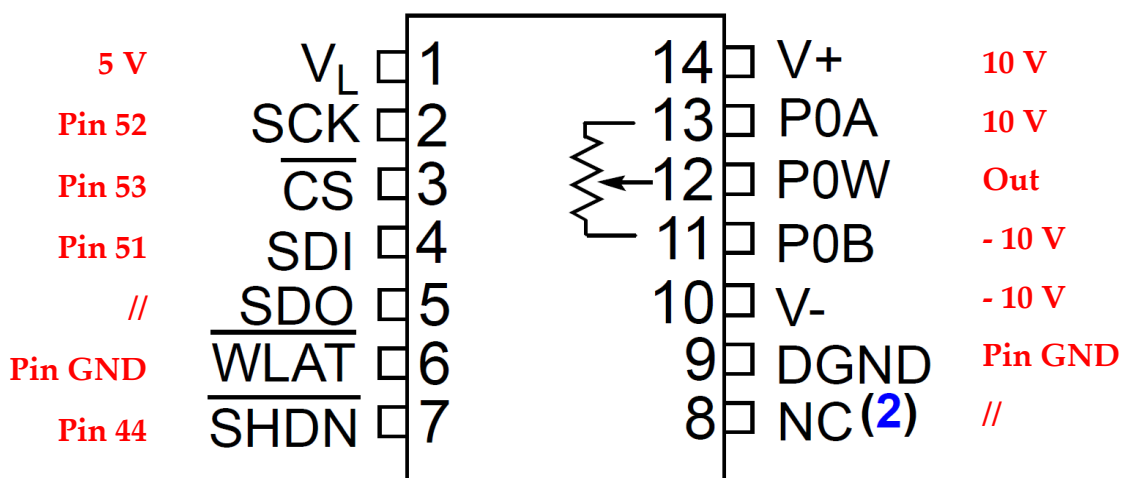


Figure 2-7 : Configuration des broches du MCP41HV51.

La broche 1 (V_L) reçoit une alimentation de commande de 5V de l'arduino.
La broche 2 (SCK) permet de synchroniser l'horloge entre l'arduino et le composant.
La broche 3 (\overline{CS}) active ou désactive l'envoi de changement de position du curseur du potentiomètre.
La broche 4 (SDI) reçoit les données de l'arduino. (bit entre 0 et 254)
La broche 5 (SDO) permet d'avoir un retour du SDI sur l'arduino pour le contrôle d'erreur. Je ne l'utilise pas.
La broche 6 (\overline{WLAT}) bloque la mise à jour de la valeur de position du potentiomètre. Reliée à la masse, elle ne bloquera jamais l'information.
La broche 7 (\overline{SHDN}) est l'équivalent d'un marche/arrêt général. Par défaut le potentiomètre est en position milieu si le composant est en arrêt. En alimentation double ce sera 0 mais pas dans le cas d'une alimentation simple.
La broche 8 (NC) comme son nom l'indique n'est reliée à rien.
La broche 9 (DGND) se réfère à la masse.
La broche 10 (V_-) est l'alimentation de puissance inférieure du composant.
La broche 11 (P0B) est la tension inférieure à faire varier.
La broche 12 (P0W) est la patte de sortie du potentiomètre délivrant la tension désirée en fonction de la valeur du bit reçu sur la broche SDI.
La broche 13 (P0A) est la tension supérieure à faire varier.
La broche 14 (V_+) est l'alimentation de puissance supérieure du composant.

Tableau 2-1 : Correspondance des broches du MCP41HV51.

Ce potentiomètre étant contrôlé en 8 bits, nous aurons 255 valeurs à attribuer, de 0 à 254, 127 étant le point milieu (pour le montage, voir en II.2.4, le « shield » personnel à notre expérience).

2.2.3 Amplification des signaux

Pour récupérer la mesure des capteurs, on utilise généralement un conditionneur d'instrumentation standard. Dans notre cas, afin de réduire les coûts et d'obtenir un meilleur contrôle sur l'instrument, nous avons créé notre propre conditionneur qui sera placé sur le « shield », emboîté sur l'Arduino.

Pour exploiter au mieux les 10 bits d'acquisition de la carte, il faut amplifier les signaux des capteurs pour s'aligner sur la plage d'entrée de 0 à 5 V.

Le choix s'est porté sur deux amplificateurs d'instrumentation de la marque Burr-Brown, racheté par Texas Instruments, le INA122 (première page en annexe 4, coût = 8 € pièce, fournis gratuitement par Texas Instruments).

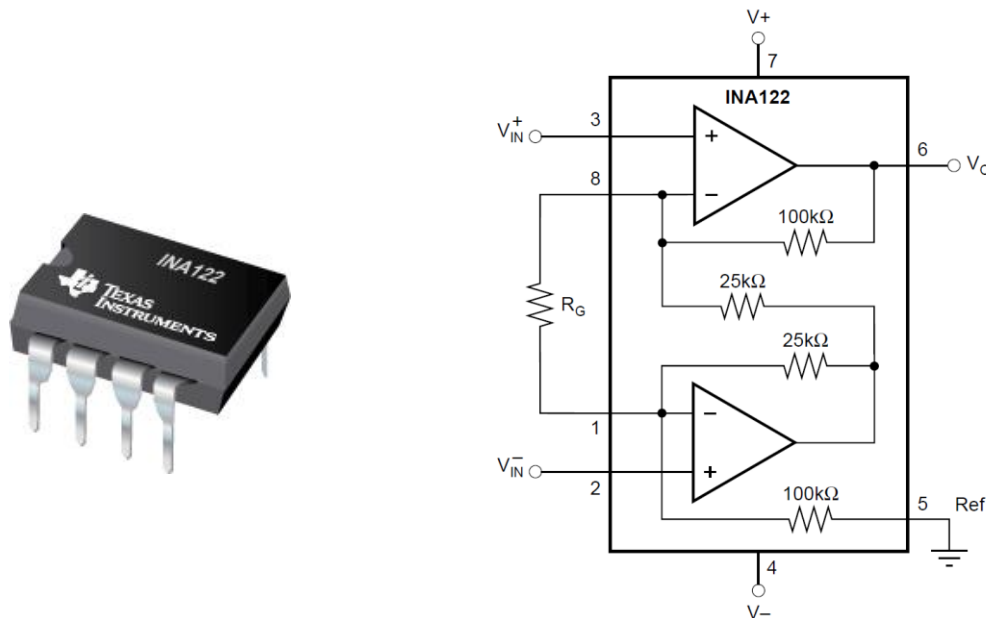


Figure 2-8 : Représentation électronique du INA122.

D'après la Figure 2-8, nous pouvons décrire son fonctionnement.

La tension de sortie, en volts, se calcule de la manière suivante :

$$V_o = (V_{in}^+ - V_{in}^-) [1 + 100/25 + 2(100)/R_g] \quad (1)$$

Son gain se caractérise par l'équation suivante :

$$\text{gain} = 5 + (200 \text{ kohms}/R_g) \quad (2)$$

Avec R_g , la résistance variable choisie.

2.2.3.1 Déplacement

Pour le capteur de déplacement, la plage utilisée est d'environ 10 mm.

D'après ses caractéristiques, la tension de sortie est de 5 V pour 101,6 mm. Il nous faudra donc un gain avoisinant 10 pour que la pleine échelle corresponde aux 10 mm désirés (un décalage de zéro sera ajouté par la suite).

$$10 = 5 + (200 \text{ kohms}/R_g) \Rightarrow R_g = 40 \text{ kohms} \quad (3)$$

Pour plus de praticité, une résistance de 39 kohms sera utilisée.

$$\text{gain} = 5 + (200 \text{ kohms}/39 \text{ kohms}) = 10,13 \quad (4)$$

Etendue de mesure utilisée = $101,6/10,13 = 10,03 \text{ mm}$

Résolution sur 10 bits : $10,03/1024 = 9,8 \text{ micromètres}$

Ce gain sera fixe, néanmoins, si l'on veut modifier cette plage, on pourra remplacer la résistance par une autre ou par une variable.

En fonction des blocs de plexiglas utilisés, la hauteur de départ peut varier considérablement (quelques centimètres). Pour compenser cet écart, il suffit de placer un potentiomètre en entrée V_{in}^- de l'amplificateur, comme indiqué sur la Figure 2-9. En ajustant la tension injectée, on pourra créer un décalage de zéro, afin de nous placer dans la zone de mesure désirée.

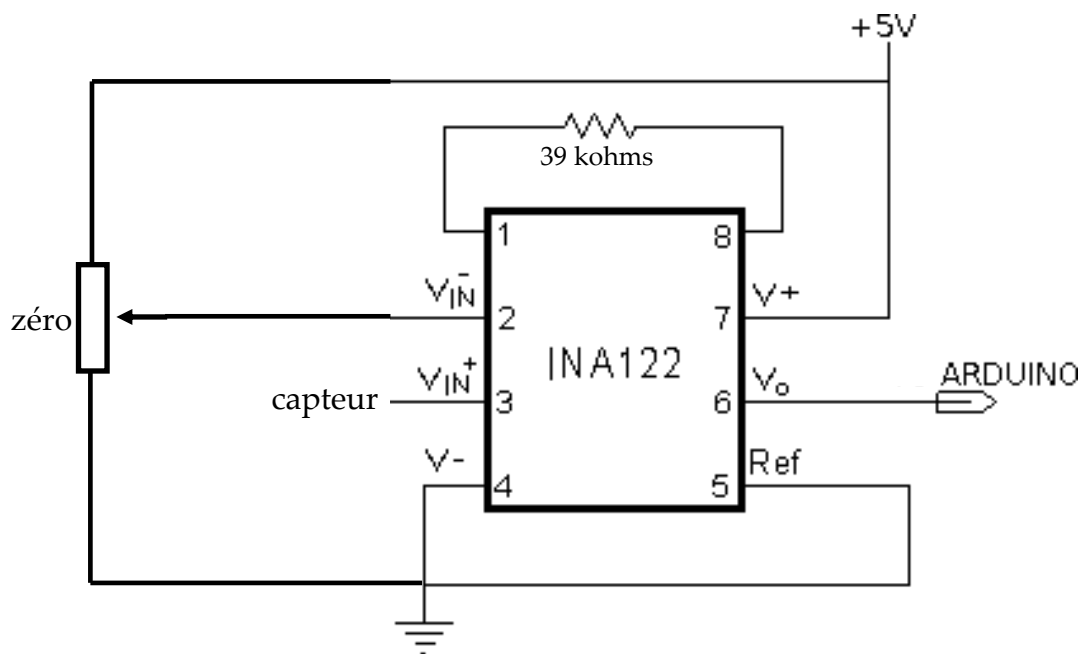


Figure 2-9 : Schéma d'amplification pour le capteur de déplacement.

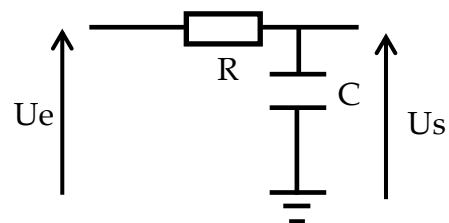
L'amplificateur étant sensible aux bruits parasites, un filtre passe-bas est placé en sortie avant l'Arduino.

Avec $R = 2200 \text{ ohms}$

$C = 0,47 \text{ }\mu\text{F}$

Fréquence de coupure : $1/2\pi RC = 154 \text{ Hz}$

Temps de réponse : 5 ms [14]



L'amplificateur est alimenté par la patte 7 (V^+) reliée aux 5 V de l'arduino.

Les pattes 4 et 5 (V^- et Ref) sont reliées à la masse de l'Arduino.

Les pattes 2 et 3 (V_{in}^- et V_{in}^+) sont les deux entrées.

Les pattes 1 et 8 accueillent la résistance R_G qui fixe le gain de l'amplificateur à 10,13.

Enfin, la patte 6 (V_o) envoie le signal amplifié à l'Arduino après filtrage.

2.2.3.2 Force

La mesure du capteur de force est effectuée par un pont de jauges montées en pont de Wheatstone intégré dans le capteur. En sortie nous aurons de très faibles tensions, d'où l'importance de les amplifier. D'après sa fiche technique, il délivrera 1,5 mV par V d'alimentation pour sa pleine échelle.

Dans notre cas nous aurons 7,5 mV pour 50 kN. Pour le moment il sera étalonné sur une plage 0 – 20 kN, ce qui donne 3 mV pour 20 kN.

Pour utiliser les 10 bits de l'Arduino, nous aurons besoin d'un gain de :

$$\begin{aligned} 5000/3 &= 1667a \\ 1667 &= 5 + (200 \text{ kohms}/R_g) \\ \Rightarrow R_g &= 120 \text{ ohms} \end{aligned} \quad (5)$$

Résolution sur 10 bits : $20000/1024 = 19,5 \text{ N}$

Au cas où il sera nécessaire d'étalonner le capteur sur ses 50 kN, et par souci de contrôle accru, une résistance variable de 1 kohm sera utilisée pour le gain. La Figure 2-10 représente le schéma de câblage.

Ce circuit aura aussi un filtre passe-bas en sortie, cependant, étant donné le gain bien plus élevé que pour le capteur de déplacement, deux filtres supplémentaires seront aussi insérés sur les deux entrées de l'amplificateur, afin d'éviter d'amplifier les bruits entrants.

Le pont est alimenté par l'Arduino, ainsi que l'amplificateur par la patte 7 (V_+).

Les pattes 4 et 5 (V_- et Ref) sont reliées à la masse de l'Arduino.

Les pattes 2 et 3 (V_{in-} et V_{in+}) reçoivent les deux autres branches du pont après filtrage.

Les pattes 1 et 8 accueillent la résistance variable R_G qui fixe le gain de l'amplificateur aux alentours de 1667.

Enfin, la patte 6 (V_o) envoie le signal amplifié à l'Arduino après filtrage.

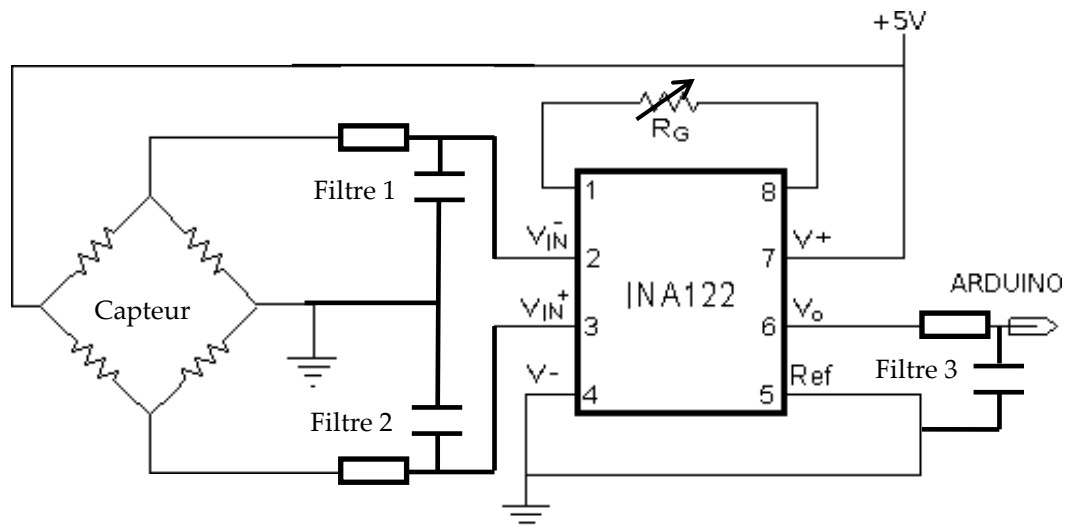


Figure 2-10 : Schéma d'amplification pour le capteur de force.

Filtre 1 et 2 : $R = 2200 \text{ ohms}$; $C = 0,47 \mu\text{F}$

Fréquence de coupure : $1/2\pi RC = 154 \text{ Hz}$; Temps de réponse : 5 ms [14]

Filtre 3 : $R = 2200 \text{ ohms}$; $C = 1 \mu\text{F}$;

Fréquence de coupure : $1/2\pi RC = 72 \text{ Hz}$; Temps de réponse : 10 ms [14]

2.2.4 Logiciel Altium et « shield » Arduino personnel à notre expérience

Pour intégrer cette électronique sur la carte Arduino, le tout est monté sur un petit circuit imprimé (ou PCB de l'anglais Printed circuit board) qui servira de « shield ». Le schéma électrique complet est disponible en annexe 5.

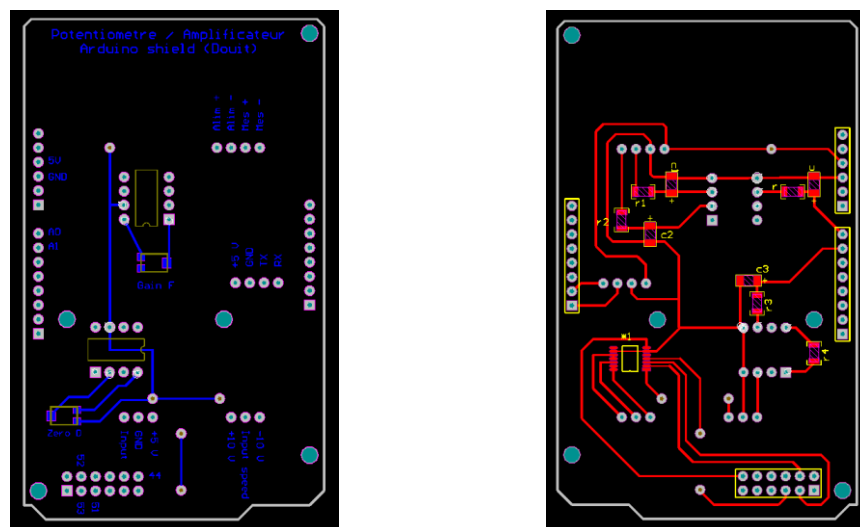


Figure 2-11 : Empreinte du PCB.

Pour développer le circuit à imprimer, j'ai utilisé le logiciel Altium. Il est très complet et permet la réalisation de schémas électriques normés et les empreintes pour imprimer des PCB.

Voici, en Figure 2-11, les deux faces du PCB que j'ai développé :

Une fois imprimé, il faut souder les composants. Certains, comme les amplificateurs, sont dits « traversant » et se soudent de manière classique. D'autres, comme le potentiomètre numérique ou les composants passifs, sont nommés « CMS » pour « composants montés en surface » ; on les dépose sur une fine couche de soudure liquide, on les passe au four, la soudure se solidifie et les composants sont ainsi fixés.

Voici le résultat en Figure 2-12 :

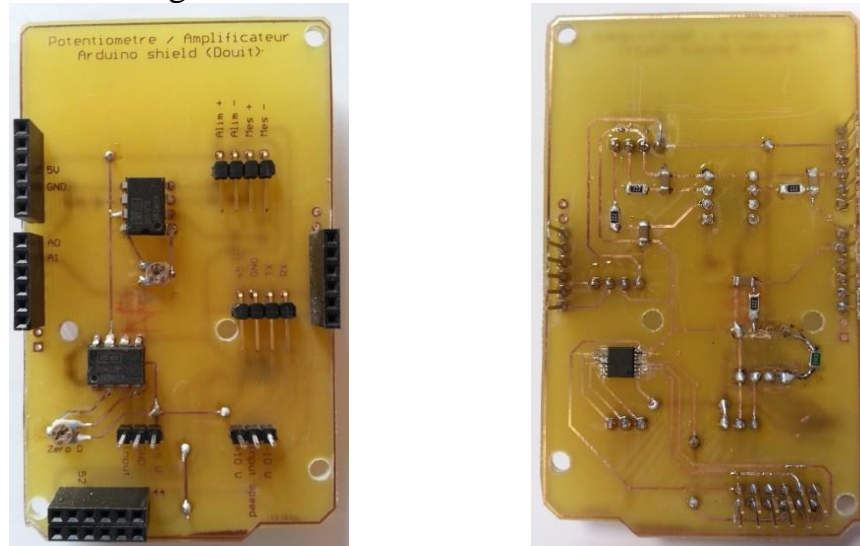


Figure 2-12 : PCB finalisé.

Ce « shield », mesurant 10 cm sur 5 cm, peut désormais être branché sur l'Arduino. Le programme sera chargé, s'en suivra un étalonnage des capteurs, puis enfin, des mesures seront acquises tout en pilotant la presse.

2.3. Programmation

2.3.1 Arduino

Les cartes d'acquisition/pilotage vendues dans le commerce à bas coût, présentent comme nous l'avons déjà vu beaucoup d'avantages. Elles sont très polyvalentes et puissantes. Cependant, toutes cartes se doivent d'être programmées pour lier les commandes aux actions souhaitées. Généralement, pour les cartes dites « professionnelles » il est nécessaire d'une part, d'acheter la licence logiciel onéreuse et d'autre part, de prendre du temps afin d'assimiler correctement la logique du programme. Or, pour les cartes du type Arduino, le logiciel de programmation est non seulement gratuit mais aussi très simple d'utilisation. Il est pensé pour être compris rapidement et ne nécessite que quelques bases de programmation pour lancer une multitude d'applications simples, comme le pilotage de leds ou de moteurs. Une interface épurée (cf. Figure 2-13) permet à l'utilisateur de créer son programme en toute liberté.

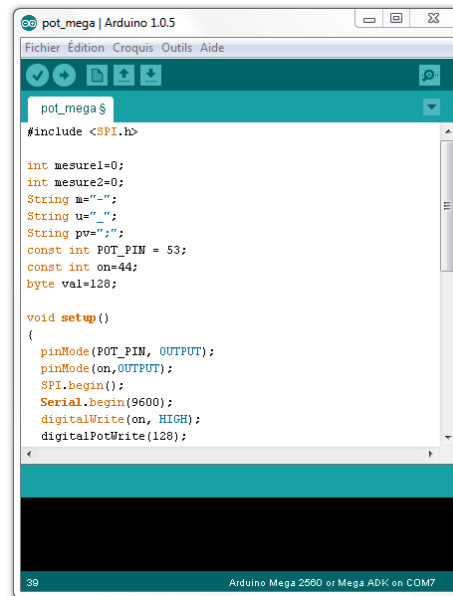


Figure 2-13 : Interface de programmation de l'Arduino.

Le code s'écrit en langage C ; deux fonctions de base sont obligatoires pour initier le plus primitif des programmes. Le Tableau 2-2 explique ces deux boucles de code.

Void Setup () {...}	La fonction setup () est appelée lors du démarrage du programme. Il permet d'initialiser les variables, les modes de broches, de commencer à utiliser les bibliothèques, etc... Cette fonction de configuration ne sera lancée qu'une seule fois, après chaque mise sous tension ou réinitialisation de la carte Arduino.
Void Loop () {...}	Après la création d'une fonction setup (), qui initialise et définit les valeurs initiales, la fonction loop () fait exactement ce que son nom l'indique, elle enchaîne des boucles consécutives, permettant au programme d'évoluer et de répondre aux commandes. Elle fournit un contrôle actif sur la carte Arduino.

Tableau 2-2 : Les deux fonctions de base à coder.

Ainsi dans notre cas, et dans un premier temps (Setup), on doit donc déclarer les variables que l'on va utiliser et il faut initialiser les protocoles :

- Variables pour les mesures à acquérir.
- Les broches connectées.
- Protocole série.
- Fonction générant le pilotage de la presse.

Puis, dans un second temps, au sein de la boucle loop () :

- On écoute les données qu'on reçoit de l'Arduino pour les convertir et les renvoyer par le port série à un programme de gestion de l'acquisition.

- On écrit aussi la valeur de commande du potentiomètre numérique qu'on utilise (et qu'on a envoyée précédemment par le port série depuis le programme de gestion).

A noter que le protocole pour le potentiomètre est constitué de 4 étapes :

- Activation de la broche.
- Choix du potentiomètre (dans le cas de plusieurs composants).
- Valeur à envoyer.
- Désactivation de la broche.

Le programme complet est disponible en Figure 2-14.



```

pot_mega | Arduino 1.0.5
Fichier Edition Croquis Outils Aide

pot_mega
#include <SPI.h>

int mesure1=0;
int mesure2=0;
String m="-";
String u=" ";
String pv="";
const int POT_PIN = 53;
const int on=44;
byte val=128;

void setup()
{
  pinMode(POT_PIN, OUTPUT);
  pinMode(on, OUTPUT);
  SPI.begin();
  Serial.begin(9600);
  digitalWrite(on, HIGH);
  digitalPotWrite(128);
}

void loop()
{
  while (Serial.available())
  {
    val=Serial.read();
    digitalPotWrite(val);
  }
  mesure1=analogRead(0);
  mesure2=analogRead(1);
  Serial.println(m+mesure1+pv+u+mesure2+pv);
  delay(10);
}

int digitalPotWrite(byte value)
{
  digitalWrite(POT_PIN, LOW);
  SPI.transfer(0);
  SPI.transfer(value);
  digitalWrite(POT_PIN, HIGH);
}
23 Arduino Mega 2560 or Mega ADK on COM7

```

Figure 2-14 : Programme de l'Arduino.

Ce programme pourrait se suffire à lui-même ; en effet, une interface de communication série est disponible et permet de recueillir les données envoyées par l'Arduino et de transmettre des commandes à la carte.

Pour plus de simplicité et de liberté de manipulation, j'ai développé un programme en C++ procurant une gestion plus souple. Je le présente maintenant, il est légèrement différent suivant qu'on utilise une liaison direct vers pc (via RS232), ou une connexion bluetooth (smartphone ou tablette Android).

2.3.2 C++ pour pc Windows

Le but de l'expérience étant d'étudier le comportement des poudres sous pression, l'acquisition des valeurs des capteurs donnera une série de mesures de force et de déplacement en fonction du temps. Pour les dépouiller, une lisibilité adaptée aux méthodes utilisées est nécessaire pour être plus efficace. Les séries de valeurs seront traitées par un tableur, et représentées sous forme graphique. Il faut donc organiser les données pour qu'elles soient lues simplement.

Qt Créator est utilisé comme interface de développement C++. Il est gratuit, simple et est très utilisé par la communauté des développeurs. Il est fourni avec une multitude de librairies et est très intuitif. Son aide à l'interface graphique est souple et permet de gagner du temps dans la conception des fenêtres ou des boutons.

Au niveau de la communication entre l'ordinateur et la carte Arduino, le protocole utilisé est celui du port série (ou RS232). Quand le pc et la carte Arduino sont reliés par le biais d'un câble usb, on utilise la librairie « QtSerialPort » qui donne accès aux informations de base à paramétrer pour initier puis entretenir la communication entre les deux appareils.

Nous y trouvons entre autres :

- L'ouverture du port
- La définition de débit de données
- Le nombre de bits utilisés
- La configuration d'un bit de parité ou non
- L'utilisation d'un bit de stop ou non

A cela il faut ajouter les commandes de base d'écriture et de lecture des données à travers le port. Lorsqu'une donnée est disponible, un signal « readyread » est émis, la commande « serial.read » est lancée pour récupérer les informations.

Pour envoyer une information par le port, c'est la commande « serial.write » qui sera utilisée.

A partir de cette base, libre à chacun de développer un programme utilisant les données transmises pour faciliter le pilotage et l'acquisition.

Dans notre cas, l'interface se compose :

- D'une partie d'initialisation de protocole.

- D'une partie pilotage avec le choix de la vitesse de montée ou de descente de la presse.
- D'une partie acquisition des valeurs des capteurs sous formes numériques et graphique
- De la possibilité d'écrire les données dans un fichier texte.

Cette dernière est transparente pour l'utilisateur ; il ne s'agit que d'un bouton, mais le code le constituant organise les données sous une forme lisible et pratique pour être interprétées simplement après coup par le tableur de dépouillement.

De même pour l'affichage des valeurs de force et de déplacement, le calcul de conversion est introduit après l'étalonnage des capteurs. D'ailleurs, on a introduit un mode « étalonnage » pour en simplifier la mise en œuvre.

Voici en Figure 2-15 ce que donne l'interface graphique ; puis les explications relatives détaillées en

	- Connexion ; déconnexion ; paramètres ; nettoyer l'écran.
	- Colonne force : valeur instantanée en kg. - Colonne déplacement : valeur instantanée en mm. - Case val moy P : valeur moyenne en Mpa sur 10 valeurs. - Case val moy D : valeur moyenne en mm sur 10 valeurs.
	- +/-, pas : choix du pas d'incrément (en % de vitesse max) appliqué lors de l'appui sur up ou down. - Up/down/stop : vitesse augmentée ou diminuée d'un pas ou mise à l'arrêt. - Go to 0 : abaisse la presse jusqu'à la position prédéfinie initialement comme « zéro ». - F max : valeur à laquelle la presse s'arrête si atteinte.
	- Active ou désactive la création et le remplissage du fichier texte de stockage des données.

Tableau 2-3 et Tableau 2-4.

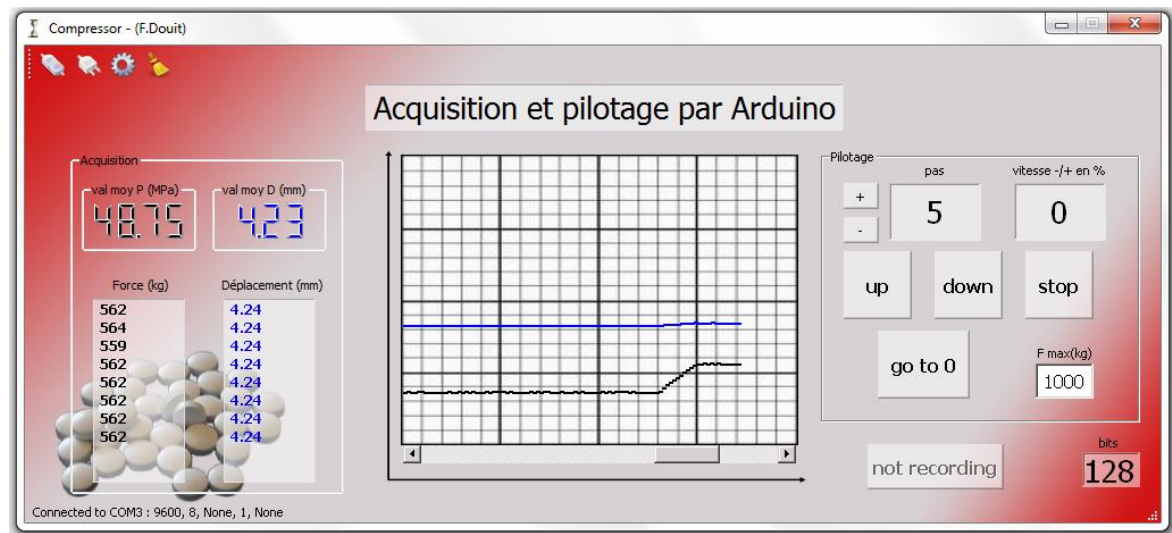
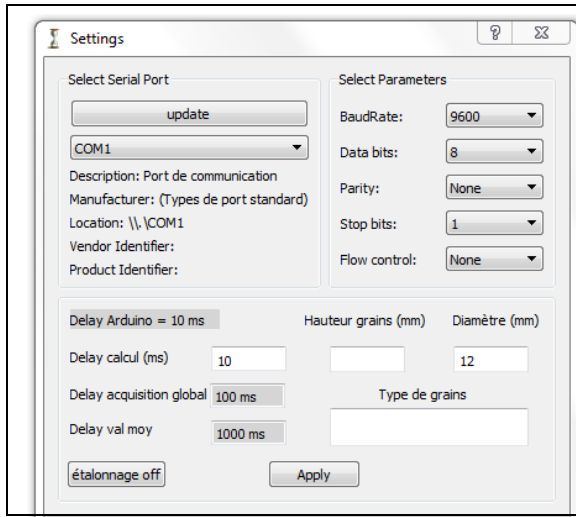


Figure 2-15 : Interface de pilotage/acquisition.

	<ul style="list-style-type: none"> - Connexion ; déconnexion ; paramètres ; nettoyer l'écran.
	<ul style="list-style-type: none"> - Colonne force : valeur instantanée en kg. - Colonne déplacement : valeur instantanée en mm. - Case val moy P : valeur moyenne en Mpa sur 10 valeurs. - Case val moy D : valeur moyenne en mm sur 10 valeurs.
	<ul style="list-style-type: none"> - +/-, pas : choix du pas d'incrément (en % de vitesse max) appliqué lors de l'appui sur up ou down. - Up/down/stop : vitesse augmentée ou diminuée d'un pas ou mise à l'arrêt. - Go to 0 : abaisse la presse jusqu'à la position prédéfinie initialement comme « zéro ». - F max : valeur à laquelle la presse s'arrête si atteinte.
	<ul style="list-style-type: none"> - Active ou désactive la création et le remplissage du fichier texte de stockage des données.

Tableau 2-3 : Explication de l'interface.



Nouvelle fenêtre après l'appui sur le bouton paramètres :


- Sélection des paramètres de protocole. 
- Choix du pas de temps entre deux acquisitions (un délai de 10 ms a été programmé au niveau Arduino, à cela s'ajoute la valeur rentrée pour donner le délai global d'acquisition).
- Renseignement des paramètres de l'expérience (hauteur, diamètre...).
- Activation du mode « étalonnage ».

Tableau 2-4 : Explication de l'interface.

Le graphique de la Figure 2-15 est placé à titre indicatif et décisionnel ; il permet d'avoir un aperçu en temps réel des variations de force et de déplacement, et de repérer toute anomalie. On peut ainsi déceler tout comportement inattendu, indésirable, résultant d'un dysfonctionnement, ou au contraire, permettre de repérer un phénomène intéressant qu'il sera bon d'approfondir par la suite, à l'aide des données recueillies.

Le code complet du programme est disponible en annexe 6.

2.3.3 C++ pour smartphone/tablette Android

L'utilisation d'un ordinateur connecté par câble usb peut devenir contraignant dans certains environnements de travail. Pour obtenir une meilleure ergonomie, j'ai développé un second programme fonctionnant sous Android et utilisant le protocole bluetooth afin de s'affranchir de toute liaison physique et diminuant ainsi l'encombrement.

Développé aussi à l'aide de Qt Créator, l'interfaçage diffère quelque peu pour s'adapter au mode de fonctionnement de ce système d'exploitation.

L'intégration du protocole bluetooth nécessite une approche différente pour initier la connexion et engager les communications entre les appareils. Cependant, l'intégration au niveau Arduino reste sur un protocole série, il ne verra donc pas la différence, son programme restant inchangé.

La librairie utilisée cette fois, s'appelle « QBluetoothSocket » à laquelle on rajoute plusieurs dépendances. L'idée générale est d'installer une communication entre les modules bluetooth, par le biais d'un système serveur/client.

- La connexion s'initie par la commande « socket.connectToService ».
- Pour récupérer une valeur, le signal « readyRead » est connecté à la commande « socket.readLine ».

- Pour envoyer une valeur, la commande « socket.write » sera utilisée.

Notons que les paramètres de communication concernent cette fois les identifiants des modules bluetooth utilisés, comme l'UUID (universal unique identifier ou identifiant unique universel en français, composition alphanumérique codée) ou l'adresse MAC (media access control que l'on peut définir comme l'adresse physique du matériel, composée de 12 caractères alphanumériques).

Une fois cette communication paramétrée, le reste du programme n'est fondamentalement pas différent du premier (II.3.2.). Seuls quelques ajustements sont nécessaires pour correspondre au fonctionnement d'Android. Par exemple, l'écriture dans un fichier texte est gérée différemment.

Aussi, l'interface est un peu simplifiée pour plus d'ergonomie. Le mode étalonnage est supprimé (il n'est disponible que dans la version pc). De même, la valeur du pas d'incrément des vitesses est fixe et est égal à 20% de la vitesse totale, pour plus de simplicité d'utilisation. Ceci limite à 5, le nombre de vitesses possibles en montée ainsi qu'en descente.

Le premier programme pour pc Windows peut être considéré comme un mode de développement (plus d'options comme l'étalonnage...), alors que le deuxième pour Android sera le mode « utilisateur » (prêt à l'emploi et simplifié).

Cette nouvelle interface est décrite dans les Figure 2-16 et Figure 2-17.

Délai calcul (ms)	Hauteur grains (mm)
10	
Diamètre (mm)	Type de grains
12	
Apply	Délai Arduino 10 ms

Figure 2-16 : Interface de configuration.

Nous retrouvons ici les paramètres à rentrer :

- Délai d'acquisition.
- Diamètres du piston.
- Hauteur des grains.
- Type de grains.

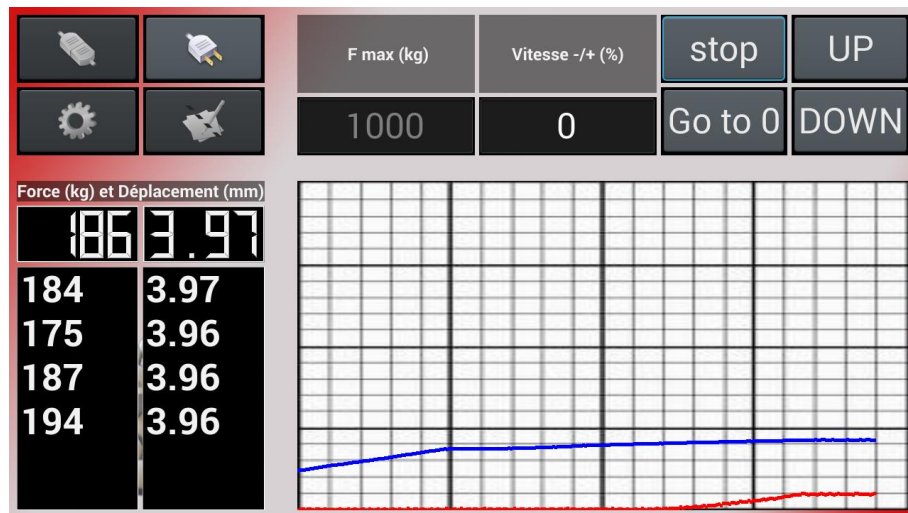


Figure 2-17 : Interface principale.

Sur l'interface principale, nous retrouvons les mêmes commandes de base, à savoir :

- Connexion/déconnexion.
- Ecriture dans fichier texte.
- Montée/descente/stop/retour au zéro.
- Force maximum
- Affichage des valeurs sous formes numérique et graphique.

Le code complet de ce programme est disponible en annexe 7.

L'utilisation du périphérique Android par communication bluetooth apporte un réel avantage en matière d'ergonomie ; il faut cependant prendre en considération que la configuration de base de l'Arduino ne peut se faire que par usb (il faut une connexion physique pour reprogrammer la puce de l'Arduino).

2.4. Matériel pour la photoélasticité

2.4.1 Les pièces usinées

La photoélasticité utilise le fait qu'un matériau transparent soumis à une contrainte locale se voit déformé, induisant ainsi une modification de ses propriétés diélectriques locales. La permittivité locale du matériau dépend alors de la contrainte, ce qui joue sur la vitesse de la lumière et produit des variations de phase de l'onde lumineuse. Ceci s'observe par polarimétrie et permet de remonter aux déformations internes du matériau transparent étudié lorsque celui-ci obéit à une loi simple (de type élastique isotrope si possible). On peut ainsi vouloir utiliser cette méthode pour rechercher et caractériser un peu mieux l'état des sollicitations extérieures imposées à ce milieu. (Par exemple ici, l'état local subit par un lit de poudre fortement comprimée dans un moule).

Pour l'expérience, il a fallu usiner des pièces de plexiglas afin d'accueillir la poudre à comprimer. Deux formes ont été choisies, elles sont représentées en Figure 2-18 et Figure 2-19 :

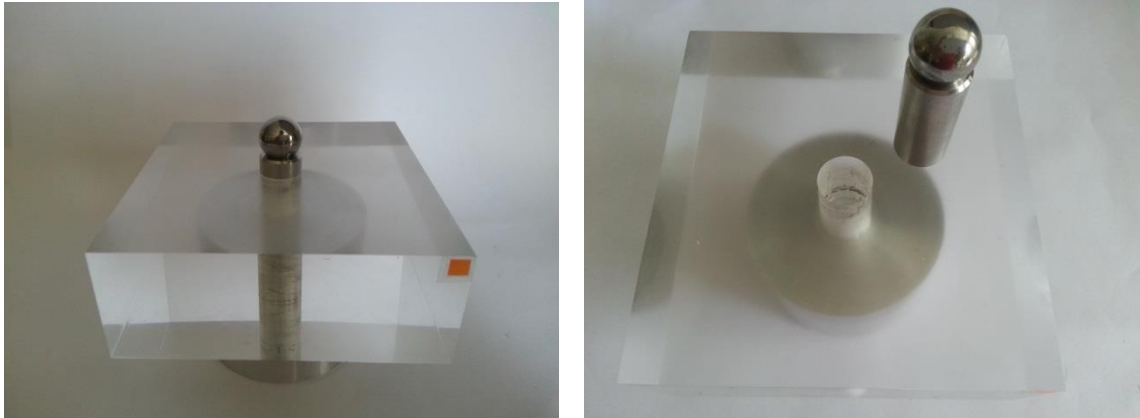


Figure 2-18 : Plexiglas avec trou rond.

- Tout d'abord, un bloc parallélépipédique de 8 cm de côté (longueur et largeur) et de 3 cm de haut. Un trou de 12 mm de diamètre est percé au milieu dans la hauteur. Ce trou forme les parois verticales du moule qu'on va utiliser. Il est fermé en haut et en bas par deux surfaces circulaires horizontales.

- La première, celle du bas, est formée d'une pièce métallique cylindrique (acier) de 4 cm de diamètre et de 2 cm de haut, surmontée d'un téton cylindrique de 12 mm de diamètre et de 1 cm de haut, qui vient s'insérer dans et sous le bloc de plexiglas. Il servira de piston inférieur « fixe ». (Attention, cette partie est « fixe » par rapport à la poudre, mais est mobile dans le repère du laboratoire puisque le plateau de la pièce monte ou descend).

- La deuxième pièce de fermeture est un cylindre métallique de 12 mm de diamètre et 3 cm de haut qui servira de piston supérieur et avancera le long de la paroi interne du plexiglas. Cette pièce est la partie « mobile » du moule, mais elle est en réalité fixe dans le repère du laboratoire, puisqu'elle est reliée directement à la poutre maîtresse de la presse via le piston de compression de la presse (décrit précédemment) et son capteur de force. Tout ceci est sensé être fixe dans le repère de la presse, hors déformations élastiques de la presse elle-même.

- Pour parachever le montage, une bille métallique de 12 mm de diamètre est placée entre le piston supérieur et le capteur de force fixé sur la presse, permettant ainsi de compenser le défaut d'alignement global des pièces.

- Le second moule reprend essentiellement les caractéristiques du premier, mais cette fois il aura un trou carré de 1 cm de côté en son milieu, et non une forme cylindrique. Les pistons seront eux aussi carrés. L'usinage et le polissage d'un trou carré étant difficile, ce second moule est fabriqué en deux parties, puis maintenues ensemble par une armature métallique épaisse de blocage. Une lucarne est usinée

dans deux faces opposées afin de laisser passer les rayons lumineux, pour pouvoir observer les effets/figures photoélastiques et remonter aux déformations.

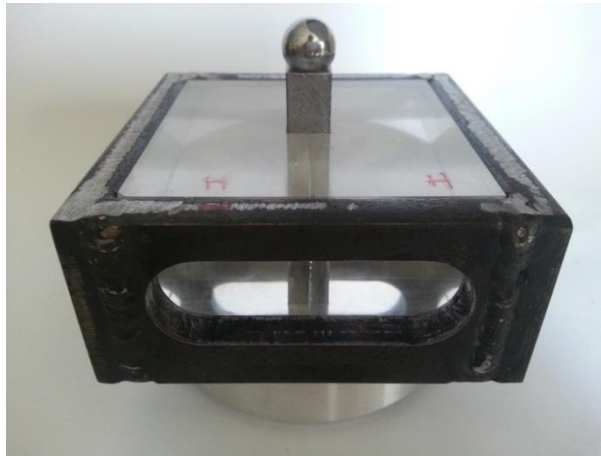


Figure 2-19 : Plexiglas avec trou carré.

L'idée est de voir comment la forme du trou peut influencer la compression de la poudre. Plusieurs types de poudres pharmaceutiques sont utilisés. Ce sont des excipients qui interviennent dans la composition de cachets médicaux. Il y a par exemple du lactose, du saccharose ou de l'amidon. Ils ont une granulométrie différente, allant de 100 à 800 micromètres, et des caractéristiques mécaniques différentes [5, 6, 7, 8]. Nous verrons si leurs comportements diffèrent.

2.4.2 Le matériel optique

Afin d'observer les phénomènes issus de la photoélasticité, il nous faut du matériel adapté, comme une source lumineuse et des polariseurs. Tout d'abord, l'éclairage est constitué de différents éléments :

- Une led de puissance verte de 3 W de type xéon (annexe 8) est utilisée comme source. Sa longueur d'onde est de 525 nm à +/- 5 nm et s'alimente entre 3,3 et 4 V.
- Une lentille convergente est placée devant la led afin d'obtenir des faisceaux parallèles pour la traversée du milieu.
- Afin de garder la tension stable, deux batteries de 3,7 V seront montées en série, et un régulateur ajustable de type LM317 maintiendra la tension d'alimentation à 3,6 V (annexe 9) (coût = 2 €, fourni gratuitement par Texas Instruments). le schéma électrique est décrit en Figure 2-20.

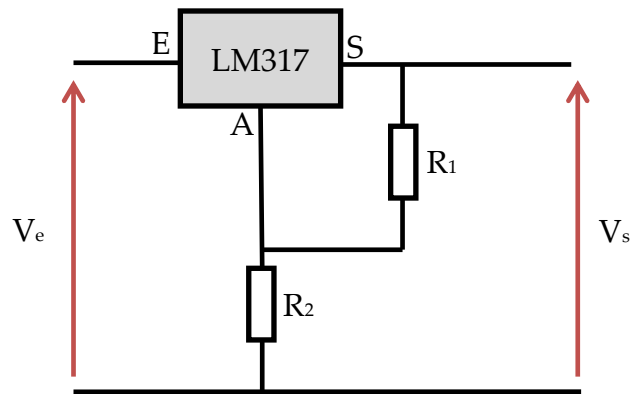


Figure 2-20 : Schéma électrique de régulation.

- E, S, A sont les 3 broches du régulateur et sont respectivement, l'entrée de tension, la sortie de tension et la broche d'ajustage de tension.
- V_e et V_s sont les tensions d'entrée et de sortie.
- Il est recommandé d'utiliser une résistance de 240 ohms pour R_1 . R_2 dépendra de la tension de sortie désirée.

La tension de sortie souhaitée V_s se calcule de la manière suivante :

$$V_s = 1,25 (1 + R_2/R_1) \quad (6)$$

Si l'on veut obtenir 3,6 V en sortie, il faudra donc une résistance de 450 ohms pour R_2 .

Dans un second temps, il faut placer des polariseurs linéaires de part et d'autre du bloc de plexiglas. En effet, pour observer les variations d'indices dues aux contraintes internes, il est nécessaire de les dissocier du reste en filtrant les polarisations émises. En croisant deux polariseurs linéaires de type filtre photographiques, représentés en Figure 2-21, la lumière est bloquée tant qu'il n'y a pas de variation d'indice engendrant une polarisation de direction différente aux filtres.



Figure 2-21 : Filtres polariseurs linéaires.

Pour étudier plus finement les franges qui apparaîtront, on ajoutera deux lames quart d'onde, ou lame à retard de phase (cf. I.2.4), constituées d'un matériau

biréfringent qui filtreront les franges isoclines qui gênent pour l'observation des isochromatiques.

Enfin, un caméscope est placé de l'autre côté du bloc de plexiglas afin de capturer l'évolution des franges en temps réel. Il filme en 30 images par seconde.

La Figure 2-22 représente l'ensemble du dispositif.

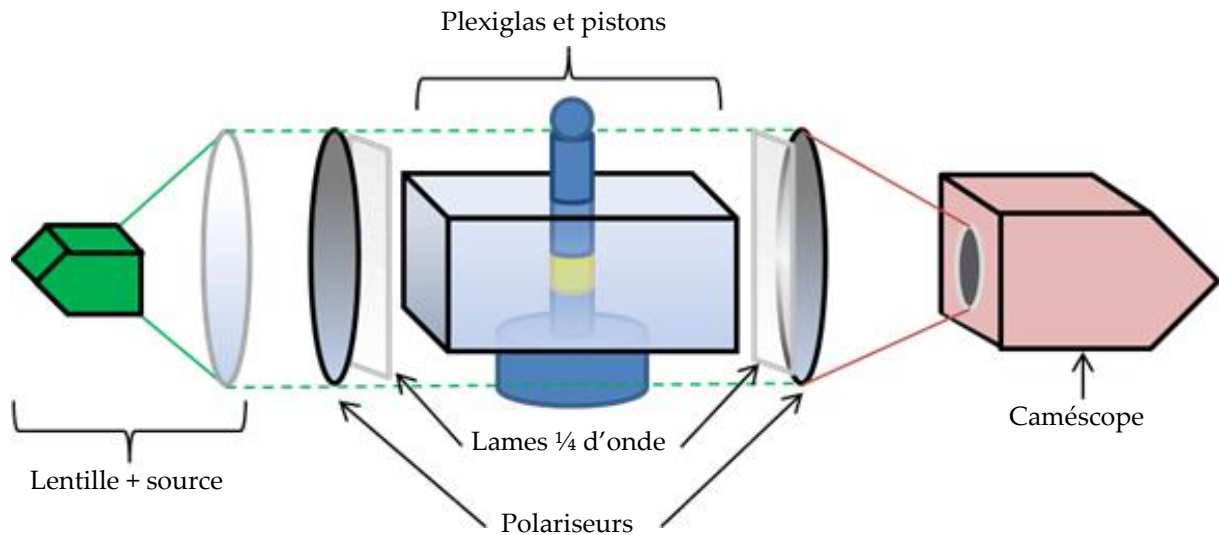


Figure 2-22 : Dispositif de photoélasticité.

3. Résultats/simulation/analyse

3.1. Résultats expérimentaux

Le but de cette étude est de démontrer qu'il est possible de monter une expérience simple et peu coûteuse pour étudier la compression des poudres sous un nouvel angle. Pour étayer notre argumentaire, nous avons donc procédé à différents essais qui peuvent servir d'exemples concernant les possibilités de cette expérience.

Utilisant les deux moules usinés et les 4 poudres pharmaceutiques à notre disposition, nous avons réalisé plusieurs séries de mesures :

- Une série de compressions avec le moule à trou rond pour une hauteur de 8 mm de poudre.
- Une autre série de compressions avec le même moule, mais cette fois avec une hauteur de poudre de 12 mm.
- Enfin, j'ai utilisé le moule à trou carré avec une hauteur de poudre de 8 mm.

Voici les différents paramètres utilisés :

- Diamètre du piston rond = 12 mm.
- Côté du trou carré = 10 mm.
- Pression max appliquée = 150 MPa (A noter qu'on peut monter plus en pression, mais n'ayant pas de plexiglas de secours, nous avons volontairement limité à 150 MPa par sécurité).

- Vitesse de montée en pression = 60% de la vitesse max, soit 60 micromètres par seconde.
- Vitesse de descente en décompression = 100% de la vitesse max, soit 100 micromètres par seconde.
- Vitesse d'éjection = 100% de la vitesse max, soit 100 micromètres par seconde.
- Acquisition des données toutes les 100 ms (ce taux est abaissable à environ 1 ms).
- 4 poudres pharmaceutiques (granulométrie décroissance) :
 - De type « gros grains », granulométrie d'environ 800 micromètres.
 - Granulométrie d'environ 650 micromètres.
 - Lactose
 - Amidon

Pour plus de facilité, lors de la création des graphiques, ces poudres seront nommées par ordre de granulométrie décroissance (calibre) et type de trou. Nous aurons donc p1r, p2r, p3r, p4r pour le trou rond, et p1c, p2c, p3c, p4c pour le trou carré.

La Figure 3-1 représente les comprimés en sortie de moule. De haut en bas et de gauche à droite nous avons :

- Première ligne : de p1r à p4r pour une hauteur initiale de 12 mm.
- Deuxième ligne : de p1r à p4r pour une hauteur initiale de 8 mm.
- Troisième ligne : de p1c à p4c pour une hauteur initiale de 8 mm.

(Notons que le lactose (p3r et p3c) n'a pas gardé sa cohésion en sortie de moule).



Figure 3-1 : Comprimés une fois sortis du moule.

En relevant le déplacement effectué après la compression (dép en mm), nous pouvons déduire le ratio ($h_0 / (h_0 - \text{dép})$) de compression pour chaque essai. Ces résultats sont répertoriés dans le Tableau 3-1 :

	P1		P2		P3		P4	
	Ratio	dép	Ratio	dép	Ratio	dép	Ratio	dép
Rond $h_0 = 12 \text{ mm}$	1,82	5,43	1,82	5,39	1,91	5,72	1,95	5,86
Rond $h_0 = 8 \text{ mm}$	2,12	4,23	1,95	3,9	2,22	4,4	1,9	3,78
Carré $h_0 = 8 \text{ mm}$	2,22	4,4	2,24	4,43	2,08	4,15	2,37	4,63

Tableau 3-1 : Ratio de compression des poudres selon la hauteur et la forme du trou.

3.1.1 Etude graphique

Regardons maintenant ce que donnent les mesures recueillies. Nous allons comparer les 4 poudres en fonction de la hauteur pour le moule avec trou rond. Voici en Figure 3-2 les courbes de compression en force appliquée (kg) par rapport au déplacement :

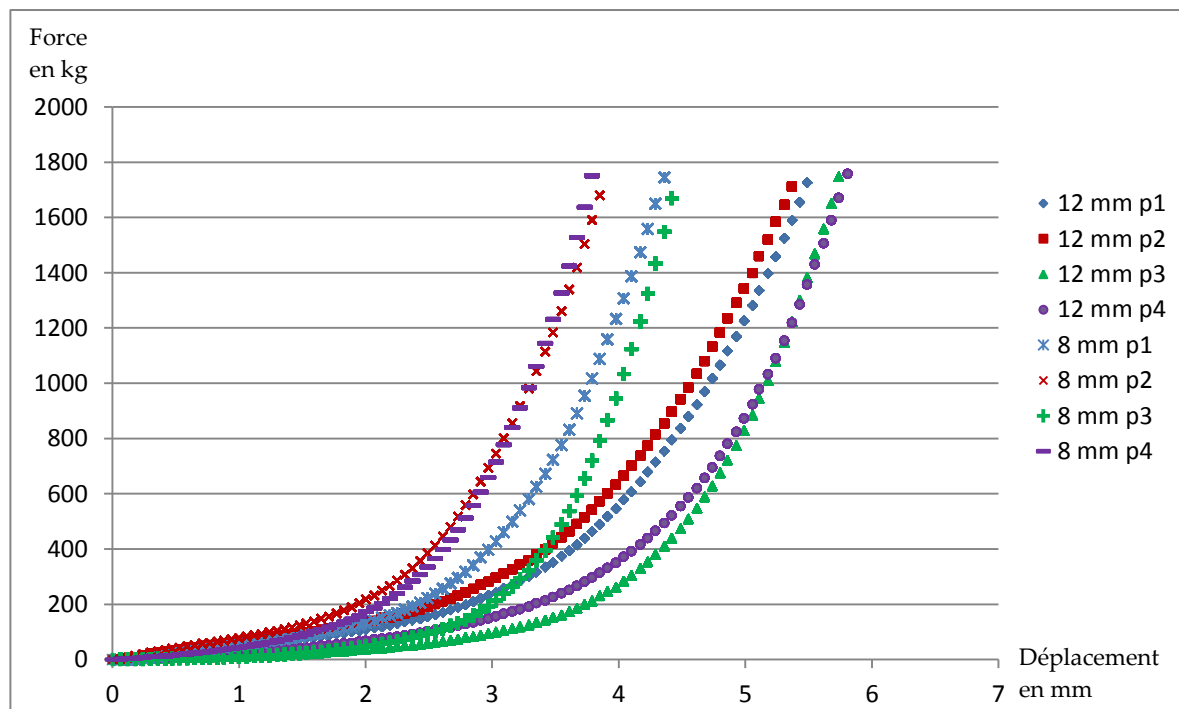


Figure 3-2 : Compression selon la hauteur de poudre et de son type.

Nous pouvons remarquer que la poudre 4 présente un plus grand écart dans la vitesse de compression entre les deux hauteurs.

Comparons maintenant en fonction de la forme du trou. La section d'application de la force étant différente, nous comparerons la pression (MPa) en fonction du déplacement comme montré sur la Figure 3-3 :

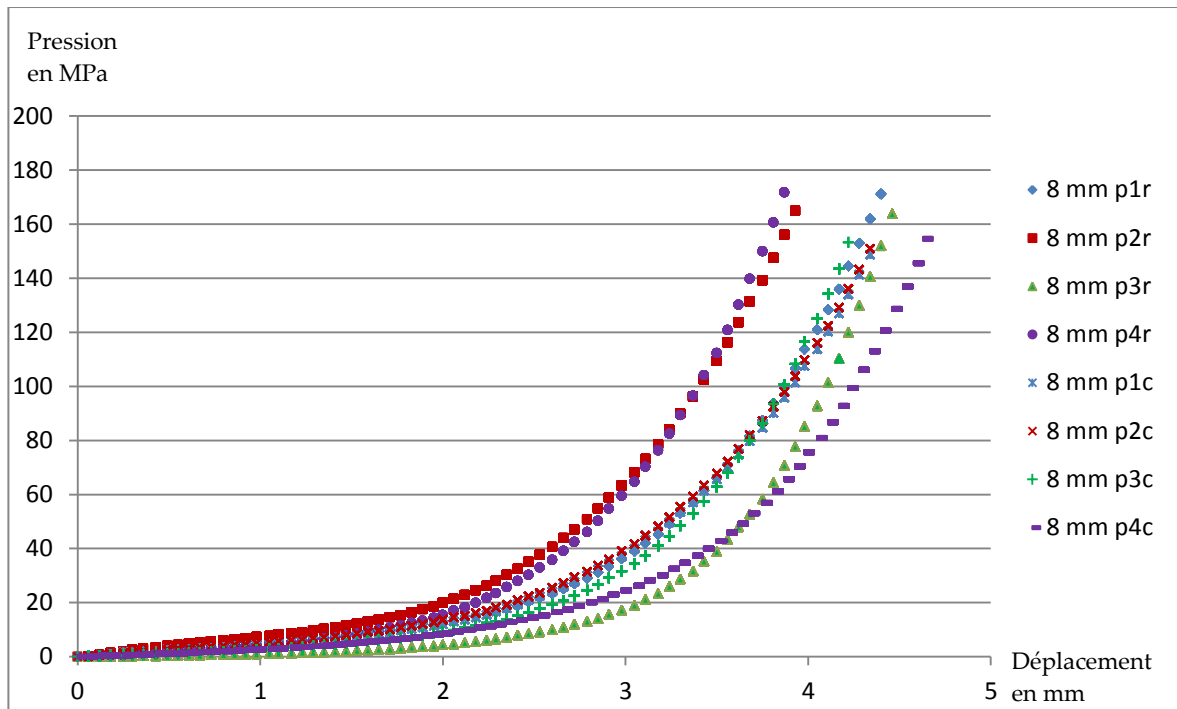


Figure 3-3 : Comparaison en fonction de la forme du trou.

Nous pouvons voir les différences de comportement selon la forme du trou. A noter que la poudre 4 présente le plus grand écart.

Regardons ce qu'il se passe lors de la décompression. La Figure 3-4 montre la chute de force mesurée (en kg) en fonction du déplacement :

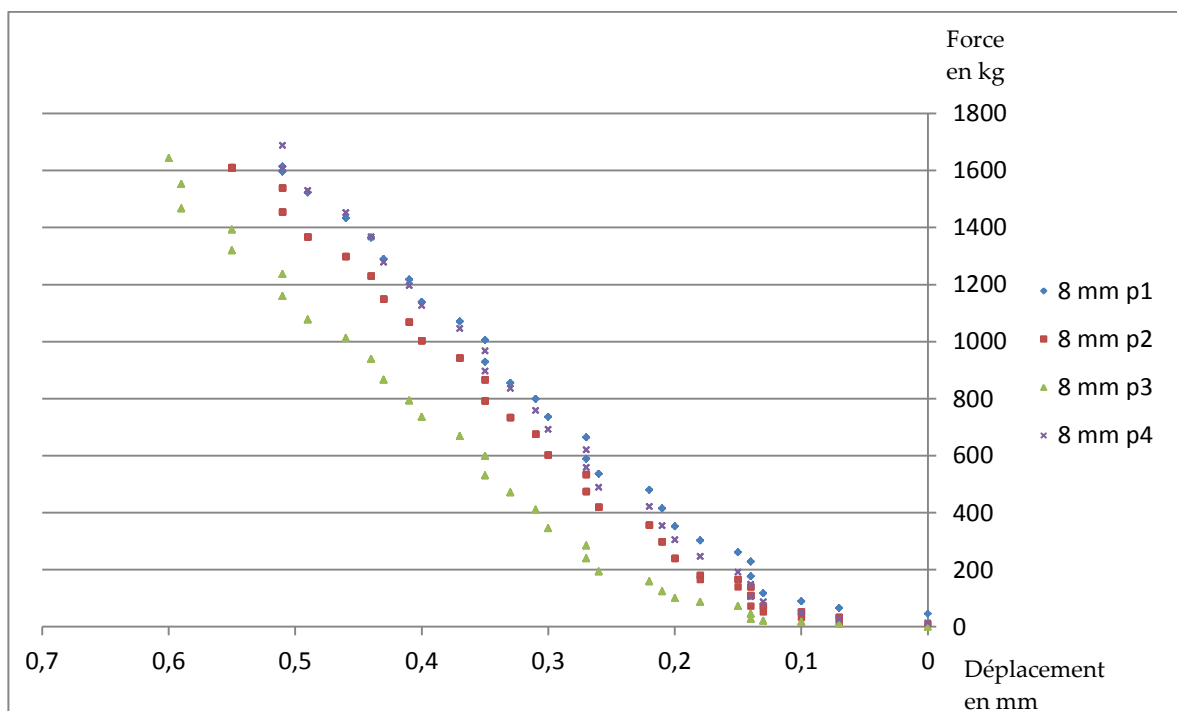


Figure 3-4 : Décompression en fonction du déplacement.

Cette variation de force mesurée jusqu'à zéro en fonction du déplacement permet de visualiser la détente du comprimé. En effet, nous pouvons voir que les comprimés se sont, en moyenne, expansés de 0,5 à 0,6 mm dans la hauteur.

En dernière étape, nous procédons à l'éjection du comprimé. Voyons, en Figure 3-5, comment varie la pression appliquée en fonction du déplacement.

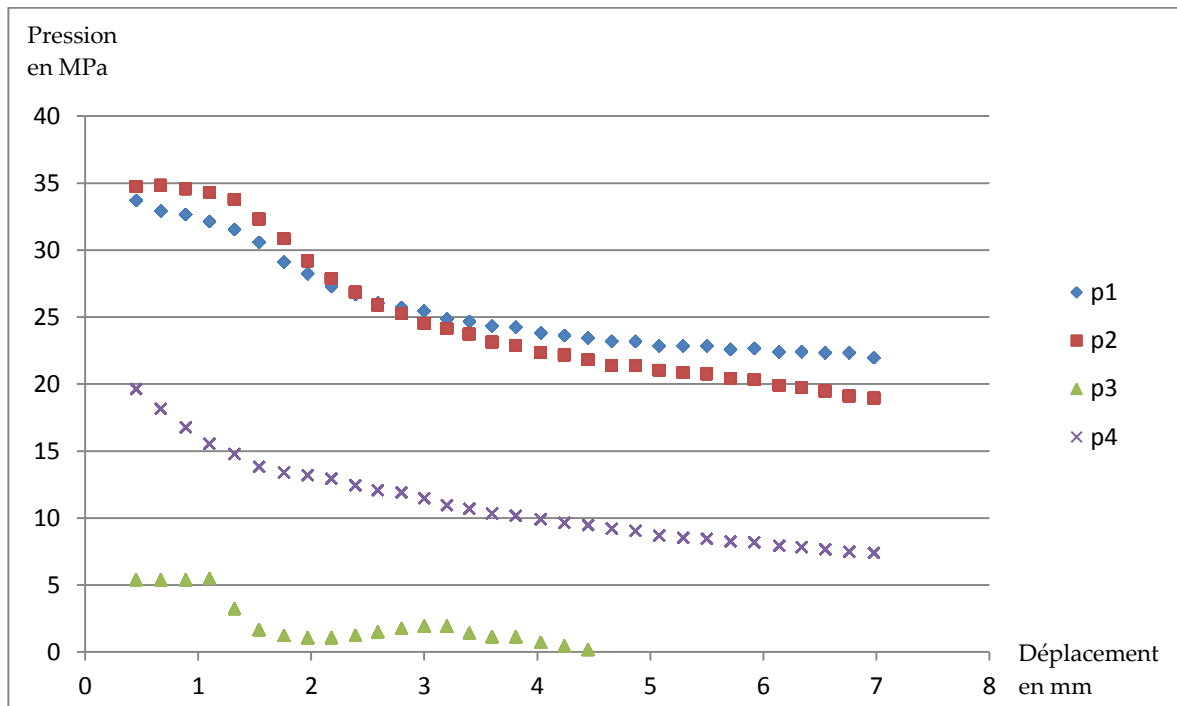


Figure 3-5 : Pression d'éjection en fonction du déplacement.

Cette pression montre comment le comprimé reste contraint au sein du moule, et comment les forces de frottement interagissent. Nous pouvons remarquer que la poudre 3 oppose très peu de résistance ; cela se comprend, car si l'on regarde la Figure 3-1, on voit clairement que la cohésion au sein du comprimé n'a pas tenue, d'où cette faible résistance.

3.1.2 Etude optique

3.1.2.1 Principe de fonctionnement

Les différents capteurs nous ont donné des mesures que nous avons étudiées. Maintenant, essayons d'analyser les résultats optiques.

Tout d'abord, étalonnons le plexiglas afin de déterminer son coefficient photoélastique. Pour cela on appliquera une pression uniaxiale déterminée (épaisseur contrainte traversée de 15 mm), afin d'obtenir une correspondance avec le nombre de franges.

Je fais varier la force appliquée (en kg) et je répertorie les franges :

- 900 kg -> 4 franges (ordre 3).
- 1200 kg -> 5 franges (ordre 4).

- 1500 kg -> 6 franges (ordre 5).
- 1800 kg -> 7 franges (ordre 6), (cf. Figure 3-6).

Nous voyons bien que l'augmentation du nombre de franges varie linéairement avec la charge. Cela prouve que l'on a bien une pression uniaxiale et un matériau de type élastique linéaire.

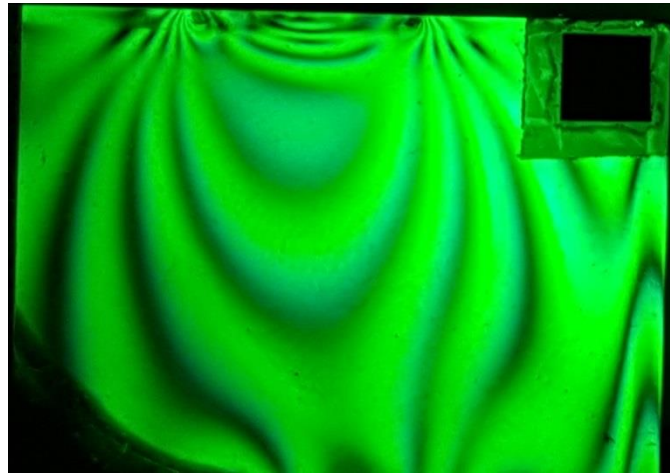


Figure 3-6 : Franges d'étalonnage.

En regardant cette photo, on pense qu'il n'est pas aisé de différencier les franges. Cependant, le comptage s'est fait dynamiquement, pendant la compression continue ; il est alors plus facile de suivre l'apparition de chaque nouvelle frange [15].

En appliquant la formule (14) du chapitre I.2.4., on peut déterminer la constante photoélastique de notre matériau. Elle est de 5,25 brewsters (la valeur moyenne tabulée chez les fabricants est aux alentours de 5 brewsters).

Prenons maintenant comme exemple la capture photo, présentée en Figure 3-7, prise en fin de compression avec la poudre p3 avec 12 mm de hauteur :

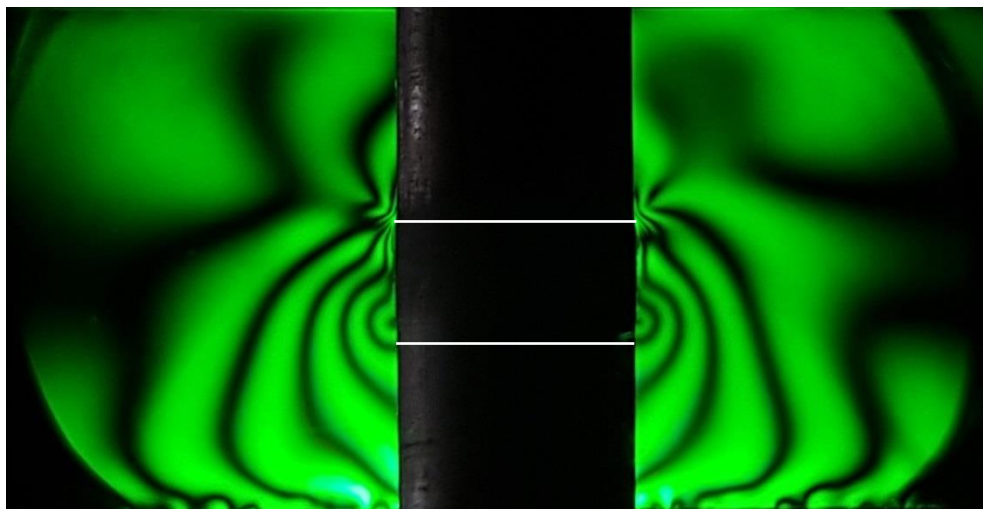


Figure 3-7 : Capture en fin de compression de p3 pour 12 mm de hauteur.

Les traits blancs représentent l'interface piston/poudre à ce stade de compression. Zoomons pour voir ce qu'il se passe au contact de la poudre et des pistons (cf Figure 3-8) :

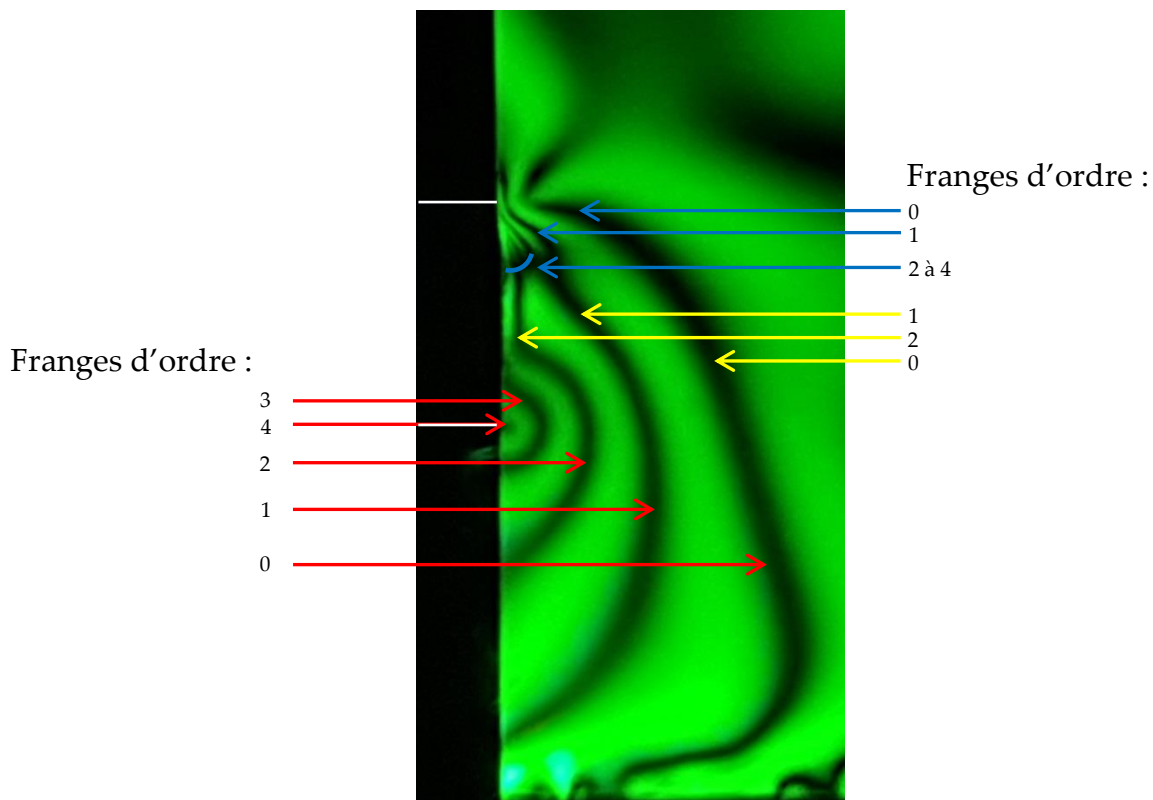


Figure 3-8 : Zoom sur l'interface.

On peut voir 5 franges au niveau de l'interface poudre/piston du bas, mais aussi 5 franges au niveau de l'interface poudre/piston du haut. Entre les deux, seulement 3 franges.

Nous pouvons déjà déduire que la valeur de la contrainte au niveau des interfaces piston/poudre est la même. Cependant, la dispersion des franges indique que la contrainte n'est pas localisée sur la même étendue ; on parle alors de concentration de franges et cela indique que la contrainte varie plus ou moins rapidement. Entre les deux pistons la contrainte est plus faible.

Du point de vue numérique, toujours en appliquant la formule (14) du chapitre I.2.4, nous aurons un $\Delta\sigma = 8,3$ MPa de contrainte par frange ($\Delta\sigma = (\sigma_2 - \sigma_1)$).

Nous avons donc aux interfaces, un $\Delta\sigma$ de 33,2 MPa, alors qu'au milieu ce sera deux fois plus faible, c'est-à-dire 16,6 MPa.

3.1.2.2 Comparaison des résultats

Maintenant que nous avons la méthode pour analyser les franges, regardons comment apparaissent ces franges. La Figure 3-9 montre une série de photos à différents temps, en cours de compression :

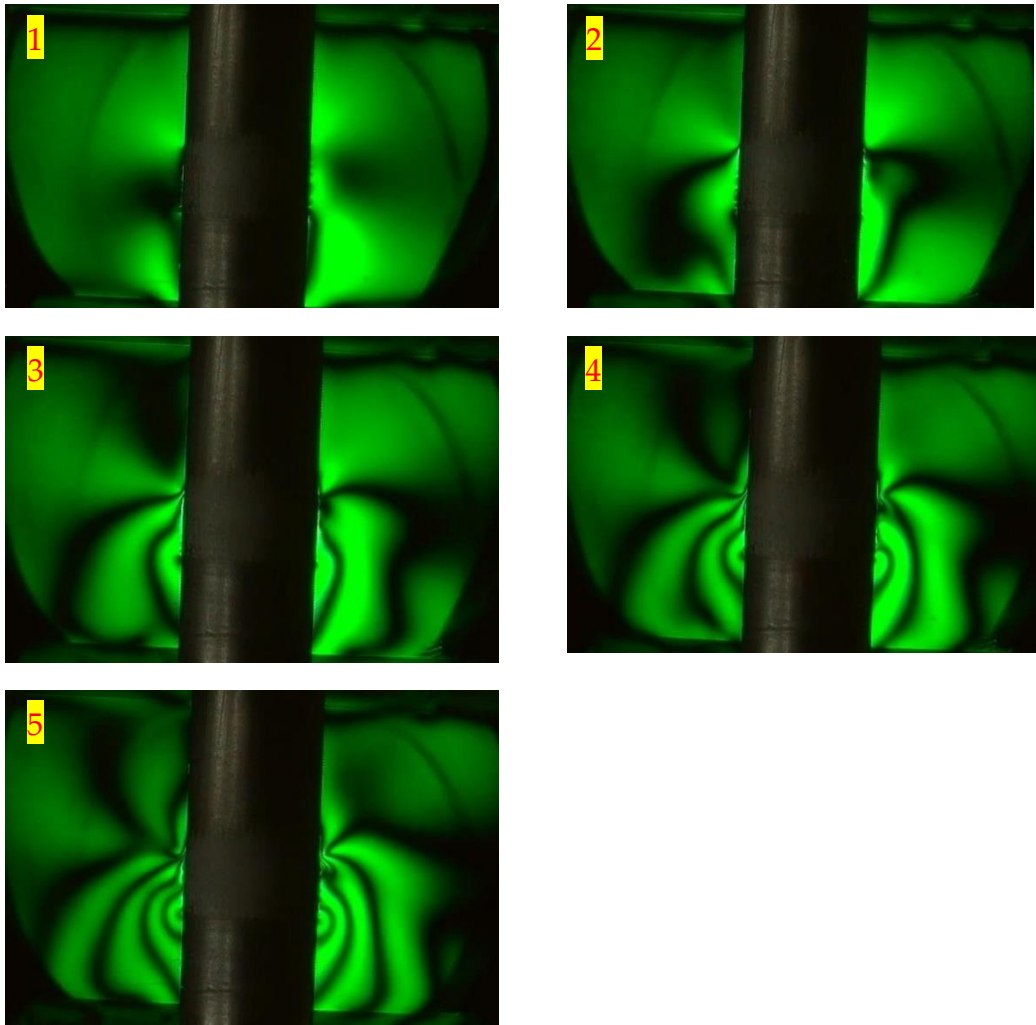


Figure 3-9 : Série de photos lors de la compression.

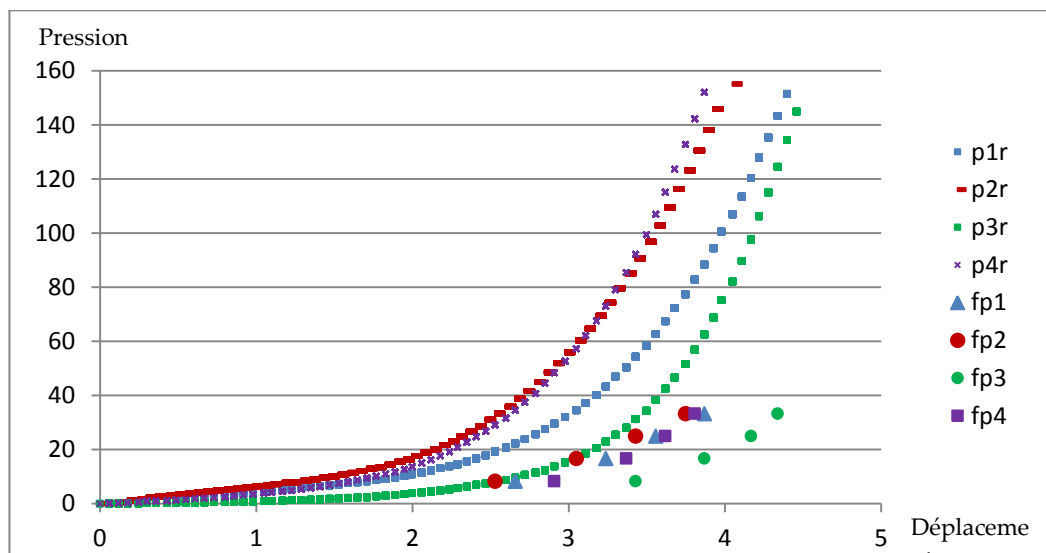


Figure 3-10 : Comparaison entre la pression appliquée et la contrainte aux interfaces

Nous voyons l'évolution et l'apparition des franges à différents endroits. Intéressons-nous à cette évolution par rapport à la pression appliquée.

La Figure 3-10 montre la comparaison entre l'évolution d'apparition des franges au niveau de l'interface poudre/piston du bas et la courbe de montée en pression pour une hauteur initiale de 8 mm de poudre dans le moule avec trou rond.

Il est aussi intéressant de comparer l'évolution d'apparition des franges entre les deux interfaces. La Figure 3-11 nous montre à quel moment apparaissent les franges pour la poudre p2 :

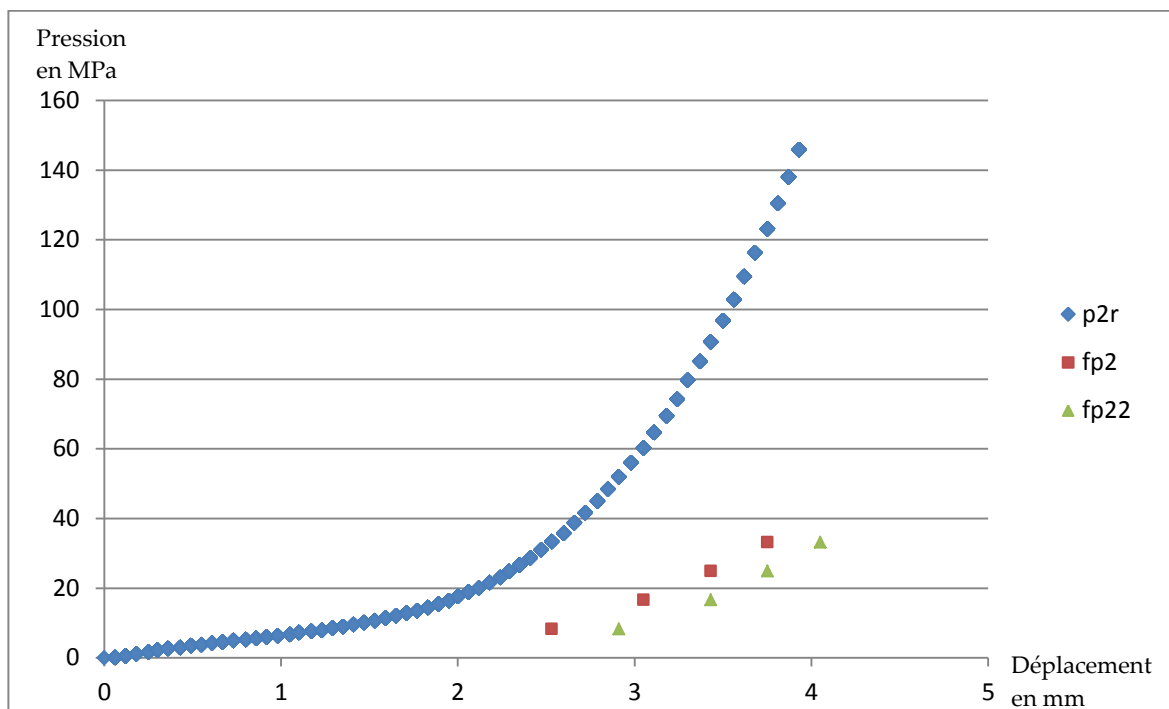


Figure 3-11 : Comparaison entre apparition des franges.

fp2 représente les franges du piston du bas, et fp22 celles du haut.

On peut voir que la contrainte augmente d'abord au niveau du piston du bas avant celle du piston du haut.

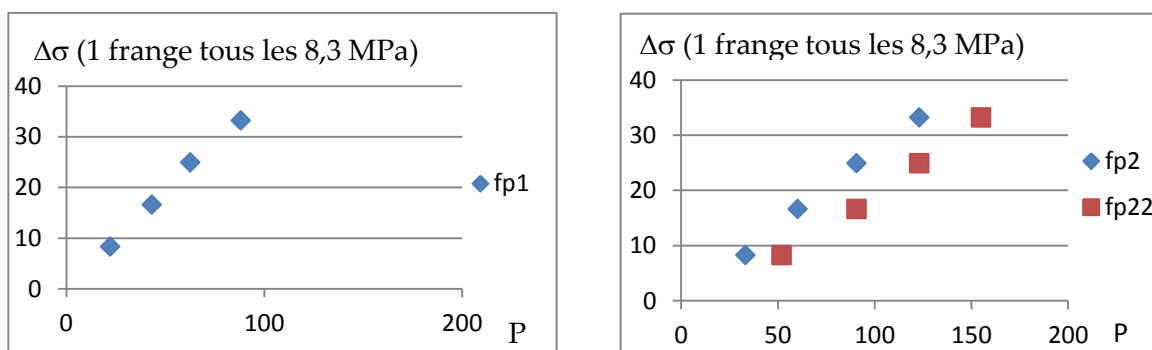


Figure 3-12 A : Contrainte par rapport à la pression appliquée P (MPa).

Afin de vérifier comment la contrainte au niveau de l'interface varie par rapport à la pression P appliquée, regardons l'ensemble de graphiques de la Figure 3-13 :

$\Delta\sigma = 8,3$ MPa par frange \Rightarrow les 4 points correspondent à l'apparition successive des 4 franges.

Le rapport semble linéaire.

Nous avons vu que le comprimé reste sous contrainte après décompression.

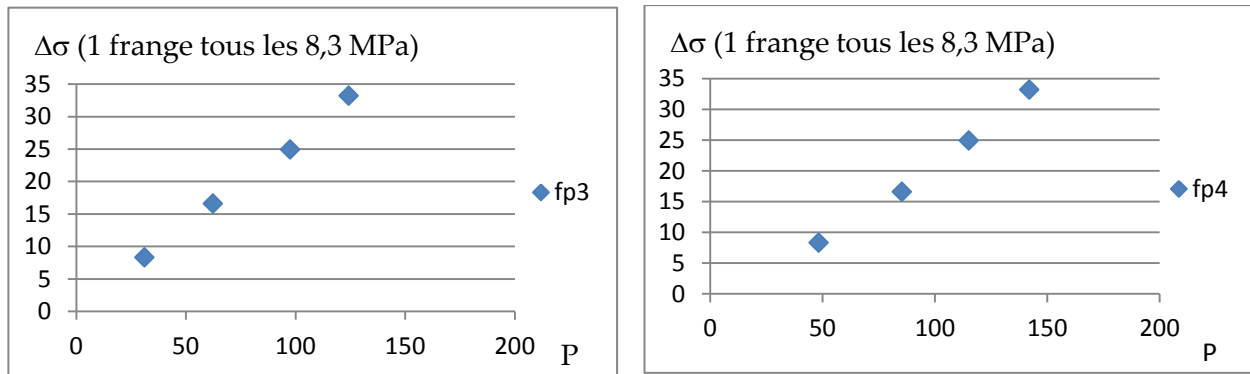


Figure 3-13 B: Contrainte par rapport à la pression appliquée.

Essayons de voir le phénomène au niveau optique. La *Figure 3-14* provient d'une photo prise après la décompression :

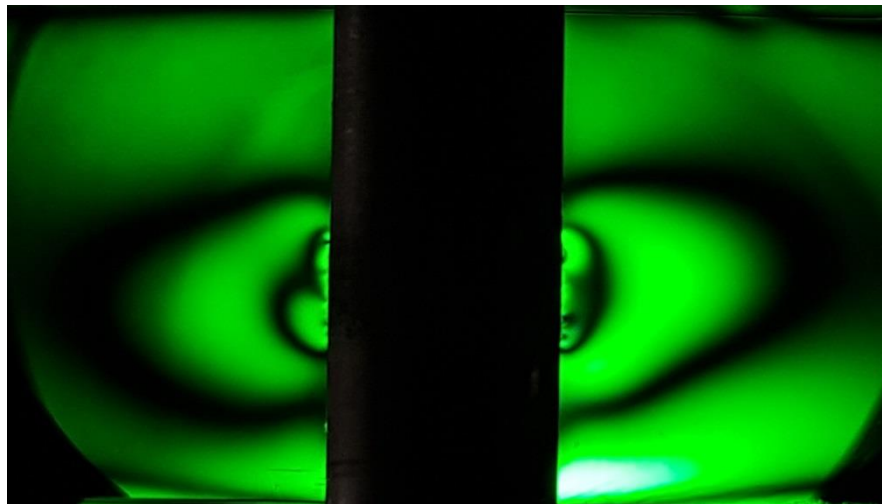


Figure 3-14 : Photo suite à la décompression.

Nous remarquons très nettement 2 franges (ordre 0 et 1). Nous pouvons en déduire qu'il reste un $\Delta\sigma$ résiduel de 8,3 MPa.

Par le biais de ces données issues d'images d'expérience de photoélasticité, nous sommes à même de repérer, d'identifier et de calculer les contraintes internes, ce qui n'est pas accessible par les capteurs extérieurs. De plus, certains recoupements ou certaines comparaisons entre ces deux types de résultats apportent d'avantage de compréhension concernant ce domaine d'étude de compression des poudres.

Nous vous laissons envisager d'autres méthodes de comparaison, d'autres analyses plus détaillées. Ceci n'est qu'un début pour apprendre à se servir de ce nouvel outil.

3.2. Simulation

Pour compléter les résultats obtenus, il est intéressant d'essayer de simuler la répartition des contraintes au sein du moule à l'aide d'un logiciel d'éléments finis (Comsol).

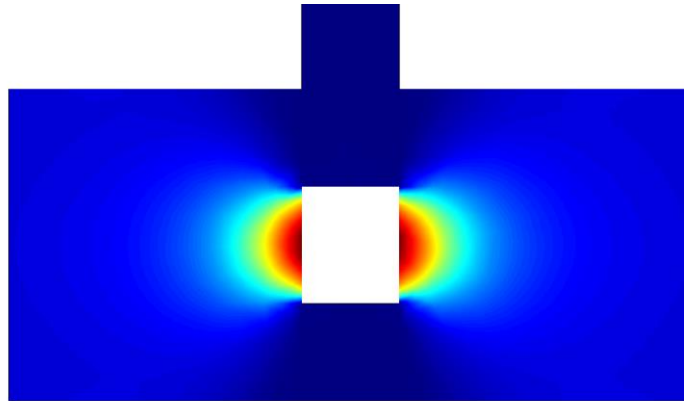


Figure 3-15 : Simulation d'une contrainte homogène.

En premier lieu, on peut simuler une contrainte appliquée uniformément sur les parois d'un trou de hauteur 12 mm (100 MPa), comme illustré en Figure 3-15 :

Ensuite, essayons de répartir arbitrairement les contraintes différemment selon la hauteur. Dans l'exemple de la Figure 3-16, en partant du bas du trou nous avons une contrainte appliquée sur les 2 premiers mm de 125 MPa ; une autre au dessus sur 8 mm, de 25 MPa ; et enfin une dernière de 100 MPa sur les 2 derniers mm :

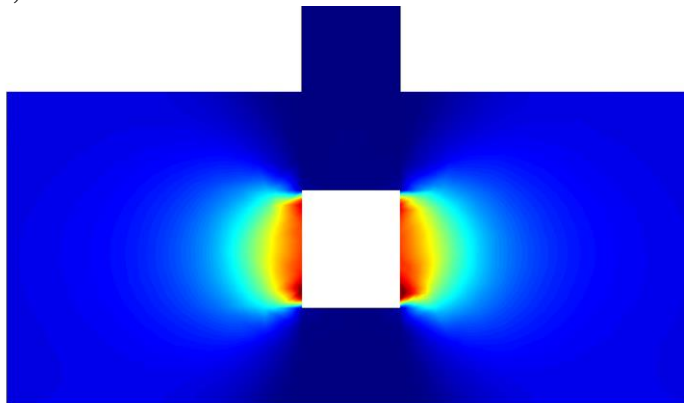


Figure 3-16 : Simulation d'une contrainte inhomogène.

En comparant, on peut voir que la contrainte au milieu est bien plus faible car elle est reportée en haut et en bas. Cela pourra être mis en évidence en simulant les franges photoélastiques. Bien sûr, ces résultats ne sont qu'une ébauche, ils sont à retravailler, différents paramètres sont à étudier et à ajuster. Cette méthode peut

aider à la compréhension des répartitions des contraintes dans le milieu. Une caractérisation très précise localement de cette répartition peut être simulée.

Ils montrent que les caractéristiques locales varient d'autant plus qu'on est près des bords de confinement du matériau, là où les contraintes sont à déterminer...

4. Conclusion

Cet article relate la mise en œuvre d'un nouvel outil de mesure précis des contraintes dans un moule servant à la compaction d'un système granulaire ou d'un système multicouche. Cette méthode peut apporter des renseignements précieux sur le comportement de ces matériaux en compression.

En plus de l'approche classique par l'utilisation de capteurs d'instrumentations (pression, déplacement), nous avons développé un système de mesure par photoélasticimétrie. La poudre est comprimée au sein d'un moule de plexiglas. Les contraintes aux interfaces poudres/moule/pistons sont diffusées dans le plastique. A l'aide de filtres polarisants, nous suivons l'évolution de ces contraintes par l'apparition de franges liées au déphasage de l'onde lumineuse qui traverse le milieu, dont l'indice de réfraction variera en fonction de la contrainte. De nouvelles données ont ainsi pu être récoltées, une première analyse a été faite pour caractériser ces interfaces. Le but de cet article était tout d'abord, de montrer la faisabilité de l'expérience mais aussi d'en déterminer l'efficacité et les limites. La bonne linéarité entre frange et contrainte prouve l'acuité de cette technique. Cette méthode est donc potentiellement très prometteuse ; elle donnera d'autant plus d'informations qu'on saura l'adapter aux différents types d'expérience, aux différents types de questions qu'on peut résoudre avec elle.

Nous pouvons tester différents types de poudre, à grains plus ou moins cassants, plus ou moins plastique ; on pourra étudier l'effet du rapport d'aspect, de la hauteur du matériau, avec différentes vitesses de compression ou encore différentes formes de trou ; tout en sachant que le matériel utilisé est peu coûteux et peut être changé en fonction des besoins. De plus, nous pouvons aussi utiliser cette technique pour résoudre d'autres problèmes liés aux conditions aux limites et à leur évolution.

Cette méthode permet une évaluation réelle des interfaces et propose un test probant du traitement des conditions aux limites, point crucial dans beaucoup de problèmes de simulation numérique.

Le matériel utilisé étant simple, peu onéreux et facile à mettre en œuvre, il est donc tentant de vouloir le transformer en un kit industriel dans le but de le vendre.

Il permet aussi de susciter des vocations à des mécaniciens/physiciens amateur qui voudrait concrétiser/améliorer leur savoir-faire. Ils peuvent apprendre quelques principes d'électroniques, d'optique et de mécanique, en améliorant leur technique de simulation numérique, de programmation... De plus des performances supérieures sont accessibles sans grand coup. La presse mécanique peut être remplacée par un cric...

Nous avons utilisé une carte d'acquisition de type Arduino, mais d'autres modèles sont disponibles, comme par exemple une carte à vitesse d'acquisition plus élevée, ou sur une plus grande résolution. Au niveau de la capture vidéo, nous avons utilisé un caméscope ; on peut trouver maintenant du bon matériel vidéo peu onéreux et embarquable. L'essor des smartphones permet d'allier pilotage et acquisition simplement. Une bonne résolution optique peut aussi aider à la détection de petits déplacements. L'étude des franges peut être affinée (méthode d'interpolation de Tardy). D'autres points expérimentaux ont été détectés dont la compréhension reste encore limitée, par exemple le fait que l'expérience permette de voir les franges localisées dans le matériau, ce qu'il faudra approfondir.

Voici donc une nouvelle approche en matière d'étude de compression des poudres et grains ; elle ne demande qu'à évoluer pour déceler de nouvelles données à exploiter.

Note de Pierre Evesque : Le lecteur pourrait s'étonner de voir un article publié par Pierre Evesque, compte tenu de ses deux éditoriaux publiés dans *Poudres & Grains* cette année, i.e. en 2015. Il répond à cette objection en deux points :

Tout d'abord, il faut noter que ce travail a été réalisé par F. Douit, dans son stage de fin d'étude pour obtenir son diplôme d'ingénieur de recherche CNAM, sous la tutelle de P. Evesque. Ce dernier ne peut donc pas se désengager de ce projet ; ce serait faire une injustice à F.Douit ; ce serait contraire à la déontologie.

Le lecteur comprendra aussi tout l'intérêt de ce travail, qui montre la possibilité de définir des expériences de recherche intéressantes à très faible coût. Nous nous proposons à court terme d'illustrer l'intérêt de ce type d'expériences pour améliorer nos connaissances dans différents domaines.

Par ailleurs, cette expérience est très pédagogique, car elle fait intervenir différents domaines de la physique (élasticité, plasticité, optique, électronique,...) et reflète la nécessité d'avoir une culture générale. Elle permet donc d'apprendre que la recherche doit se fonder sur des connaissances que d'autres avant vous ont acquises, et transmises, et que l'on doit apprendre à contrôler, puis à utiliser. Monter une telle expérience permet à la fois d'être face à un manque de connaissance (c'est donc un apprentissage de la recherche), de proposer des solutions qu'on peut tester, puis de les comparer à celles que d'autres ont proposées. Cela permet d'améliorer notre connaissance de la physique, via la séquence classique question-réponse-test, et à prendre conscience que sans le test de la réalité le physicien n'a pas amélioré sa connaissance scientifique.

Enfin la qualité des images et des films obtenus montre que les phénomènes physiques peuvent être beaux et simples, et peuvent faire rêver, tout en vous poussant à approfondir votre culture bibliographique pour être plus efficace.

On sait faire ce type d'expérience (photoélasticimétrie) depuis longtemps. La particularité de celle-ci est son faible coût et l'ampleur de ses possibilités thématiques d'application.

Cette expérience n'apprendra rien au tricheur, sauf à démontrer à lui-même qu'il se met en défaut quand il triche. Par contre cette expérience peut servir à établir l'existence de mauvaises interprétations. C'est pour cela que certains faux scientifiques n'apprécieront pas cette expérience, car elle leur montrera leur tricherie. Le débat scientifique à propos des retombées scientifiques de cette expérience doit donc rester ouvert.

Enfin, le combat mené par P.Evesque à travers les tribunaux (administratif et de grande instance) est probablement perdu, comme le montre le dernier jugement (du 15/7/2015) du tribunal administratif. Mais qu'a-t-on appris à travers ces dédales (1) le manque de fiabilité des avocats lorsque l'état est en cause, comme le montre la démission sine die de l'avocat de PE à quelques jours de l'audience en Cours administrative d'Appel (CAA), et sans que celui-ci ne le dise à la CAA (voir le blog <http://defense-pierre-evesque.over-blog.com>), (2) le manque réel de crédibilité et de clairvoyance de la justice humaine : la justice refuse de mettre en doute la qualité des procédures administratives, leur respect par l'administration et la fiabilité des conclusions.

Ainsi, que le dit ce dernier jugement : «... que ce dernier (M.Evesque) se trouve dans une situation de grande souffrance psychologique, traduisant un symptôme dépressif, et que son comportement, qui s'est traduit par plusieurs esclandres, a généré une situation très tendue au sein du laboratoire et l'incompréhension de la hiérarchie et de ses collègues ; que le compte rendu de la réunion du comité médical du 15 mai 2013 indique que M. Evesque témoigne d'une souffrance psychologique dont il reconnaît lui-même l'intensité et d'un état d'épuisement intellectuel, confirmant le diagnostic de syndrome dépressif, et que cet état est constitutif d'une pathologie invalidante et de gravité confirmée nécessitant des soins prolongés et le rendant temporairement inapte à l'activité professionnelle ;... »

Donc la souffrance est reconnue par tout le monde, même par la Cour, mais pas le harcèlement par refus de déontologie. Or il est clair que P.Evesque est toujours au combat ; il n'est donc pas en dépression. De plus cette souffrance ne peut se comprendre que par le refus de déontologie. Ainsi comment le Tribunal peut-il croire à un syndrome dépressif caché, quand il voit la volonté de P.Evesque de divulguer les incohérences du système, avec un état cherchant à cacher des faits, à les nier, à les transformer... (toute chose incompatible avec son travail de scientifique).

C'est la preuve que l'Etat (pouvoir exécutif) se déifie, qu'il associe à cette déification la Justice et le Parlement (pouvoirs législatif et judiciaire) pour s'octroyer un quitus inconcevable. A mieux y regarder, ce n'est pas l'Etat, ni la Justice, ni le Parlement, qui doivent tous obéir à une déontologie particulière, et faire en sorte que l'esprit soit en accord avec la lettre, mais l'administration, i.e. les hommes, qui refusent cette déontologie. De même, il est clair que la position des médecins du Comité médical est inqualifiable et incohérente.

Par exemple, comment ce Comité peut-il espérer réduire la souffrance de P.Evesque avec de telles méthodes? Ce comité est tout simplement formé de paons non scientifiques qui veulent faire avaler des couleuvres comme si c'était du caviar (Malheureusement, ils ne sont pas les seuls à avoir le même type d'objectif....).

Reste maintenant à bâtir une science honnête, basée sur une communauté honnête indépendante des pouvoirs humains.

On sait que ceci ne peut se faire sans un contrôle sérieux et réel des revues à comité de lectures, qui ont maintenant dépassé en mal toutes les prévisions. Ceci passe par un débat ouvert et franc, ce que les revues refusent.

Ceci est encore possible via des évaluations a posteriori par des hommes de bonne volonté indépendants de tout pouvoir, ou contrôlés par un système de pouvoir qui se soit réformé.

Peu de personnes voudront me croire maintenant ; c'est pourquoi j'ai laissé un certain nombre de traces dans *Poudres et Grains* [17].

Bibliographie

- [1] Y. Boudiaf, « Etude de l'influence des paramètres physicochimiques du liquide de mouillage sur le procédé de granulation par voie humide. », Faculté de pharmacie, Université Henry Poincaré – Nancy 1, (2009), 82 p.
- [2] M. Kadir, « Compression de poudres pharmaceutiques et interaction avec l'outillage, analyse expérimentale et modélisation numérique. », Institut national Polytechnique de Toulouse, (2004), 182 p.
- [3] P. Evesque, PG. de Gennes, « sur la statique des silos. », Comptes Rendus de l'Académie des Sciences - Series IIB - Mechanics-Physics-Astronomy, Volume 326, Issue 11, (November 1998), p.761–766, doi:10.1016/S1251-8069(98)80011-0.
- [4] H. Riedel, A. Ronzheimer, M. Sitzmann, P. Evesque, PG. de Gennes, « sur la statique des silos. », Comptes Rendus de l'Académie des Sciences - Series IIB - Mechanics-Physics-Astronomy, Volume 326, Issue 11, (November 1998), p.761–766, doi:10.1016/S1251-8069(98)80011-0.
En fait, cela fait référence à l'article [7], mais PG de Gennes a voulu montrer ainsi son intérêt pour le problème en incluant/associant le travail de ses stagiaires, qu'il a embauchés après publication de l'article (note de PE).
- [5] A. Modaressi, S. Boufellouh & P. Evesque, « Modelling of stress distribution in granular pile : comparison with centrifuge experiments. », Chaos 9, (sept 99, reçu le 30/12/2008), (comparer à Vanel et al. PRE60,R5040(Nov.199, reçu mai1999)), p.523-543.
- [6] S. Boufellouh, « calcul des contraintes dans un empilement. », thèse ECP, soutenue le 20/4/2000.
- [7] A. Modaressi & P. Evesque, « Is the friction angle the maximum slope of a free surface of a non cohesive material. », Poudres & Grains 12 (5), (juin 2001), p.83-102.
- [8] V. Busignies, B. Leclerc, P. Porion, P. Evesque, G. Couarraze and P. Tchoreloff, « Compaction behaviour and new predictive approach to the compressibility of binary mixtures of pharmaceutical excipients. », Eur. J. Pharm. Biopharm., 64, (2006), p.66-74.
- [9] V. Busignies, B. Leclerc, P. Porion, P. Evesque, G. Couarraze and P. Tchoreloff, « Investigation and modelling approach of the mechanical properties of compacts made with binary mixtures of pharmaceutical excipients. », Eur. J. Pharm. Biopharm., 64, (2006), p.51-65.
- [10] V. Busignies, B. Leclerc, P. Porion, P. Evesque, G. Couarraze and P. Tchoreloff, « Potential of X-ray microtomography to detect localized variations of density in cylindrical tablets. », Eur. J. Pharm. Biopharm., 64, (2006), p.38-50.

- [11] V. Busignies, B. Leclerc, P. Porion, P. Evesque, G. Couarraze & P. Tchoreloff, « Application of percolation model to the tensile strength and the reduced modulus of elasticity of three compacted pharmaceutical excipients. », *Europ. J. Pharm. Biopharm.*, 67.507-514, (2007), doi:10.1016/j.ejpb.2007.02.005.
- [12] P. Evesque, « Quasi-static mechanics of granular material. », *Poudres & Grains* NS 1, (2000), p.1-155, ISSN 1257-3957, arXiv:cond-mat/0507303.
- [12 bis] P. Evesque, « Eléments de mécanique quasistatique des milieux granulaires mouillés ou secs. », *Poudres & Grains* NS-1, (décembre 2000), p.1-155.
- [13] L. Landau et E. Lifchitz, « Théorie de l'élasticité. », Edition de Moscou, (1967), Tome VII, 206 p.
- [14] Okawa electric design, « RC low-pass filter design tool. », disponible sur : (<http://sim.okawa-denshi.jp/en/CRtool.php>).
- [15] F. Douit, Vidéo de photoélasticité, « <http://www.poudres-et-grains.eu/datas/photoelastic.avi> »
- [16] D. Borza, « Introduction à l'étude des contraintes par photoélasticité. », Laboratoire de photomécanique du département de mécanique de l'INSA de Rouen, U.V. mécanique expérimentale, (2003), 8 p.
- [17] P&G No 22 – 1 (2015), P. Evesque; 1st Editorial, I am Charlie (English), pages 1-3 ; P. Evesque; 1er Editorial, Je suis Charlie (français), pages 4-9
- P&G No 22 – 2 (2015), P. Evesque; 2nd Editorial, Why I am Charly (français-English) , pages 10-15
- P&G No 19 – 3(2011) P. Evesque; Reading notes on: “Les milieux granulaires ; Entre fluide et solide” by B.Andreotti, Y. Forterre et O. Pouliquen, , pages 17-18
- P&G No 20 - 1 (Juin 2012) P. Evesque; Dialogue of the deaf : «Hydrodynamics» with dissipation. Towards mixing or demixing ? , pages 1-28
- Avec sa traduction en anglais: P&G No 20 - 1 (Juin 2012) P. Evesque; Dialogue de sourds : "Hydrodynamique" avec dissipation. Vers un mélange, ou une ségrégation ? , pages 11-28
- Et son analyse : P&G No 20 - 2 (Juillet 2012) ;J. Villain; Shaken sand, stress and test particles , pages 29-36
- P&G No 20 - 4 (Décembre 2012) P. Evesque; A mes pairs : Gaz granulaire et 2nd Principe de thermodynamique: un gaz "dur", un gaz de combat, un gaz de débat manqué , pages 52-67 (version originale); pages 52-69; (version révisée janvier 2013)
- Avec sa traduction en anglais: P&G No 21 - 1 (2 Janvier 2013) P. Evesque; To my peers. Granular gas and the 2nd principle of thermodynamics , pages 1-19
- P&G No 17 - 1 to 18 (2009): R. Liu, M. Hou, P. Evesque ; Simulation of 3d granular dissipative gas under different kinds of excitations & with different number of balls N,
- P&G No 17 - 20 (2009): P. Evesque ; Microgravité et Gaz Granulaire Dissipatif dans un système vibré : un gaz à vitesse dissymétrique, mais à moyenne nulle pages 577-595
- Avec sa traduction en anglais: P&G No 18 - 1(2010) P. Evesque ; Microgravity and Dissipative Granular Gas in a vibrated container: a gas with an asymmetric speed distribution in the vibration direction, but with a null mean speed everywhere , pages 1-19
- P&G No 7 (1999) P. Evesque: Stress propagation in granular media: Breaking of any constitutive state equation relating local stresses together by a change of boundary conditions pages 1-18
- P&G No 11 – 4 (2000); P. Evesque: The jamming surface of granular matter Determined from soil mechanics results pages 58-59
- P&G No 12 – 8 (2001) P. Evesque: Macroscopic Continuous Approach versus Discrete Approach, Fluctuations, criticality and SOC. A state of the question based on articles in *Powders & Grains* 2001 pages 122-150
- Notice pour auteurs (et lecteur)
- P&G No 13 – 8 (2002) P. Evesque: Editorial: Pourquoi publier dans *Poudres & Grains* : Vers une politique de l'évaluation scientifique plus efficace pages 6-11
- Avec sa traduction en anglais: P&G No 13 (2002) P. Evesque: Editorial : Why deciding to publish in *Poudres & Grains* : Towards a more efficient politics of scientific evaluation Translated by Henri-Thierry TOUTOUNJI pages 12-17

Table des annexes

Annexe 1 : Capteur de force FN3042.....	62
Annexe 2 : Capteur de déplacement CLP13	63
Annexe 3 : Potentiomètre MCP41HVX1	64
Annexe 4 : Amplificateur INA122.....	65
Annexe 5 : Schéma électrique.....	66
Annexe 6 : Code C++ pour pc.....	67
Annexe 7 : Code C++ pour Android	75
Annexe 8 : Led de puissance.....	83
Annexe 9 : Régulateur LM317.....	84
Annexe 10 : Quelques rappels complémentaires en photo-élasticimétrie	85

Liste des tableaux

Tableau 2-1 : Correspondance des broches du MCP41HV51.....	27
Tableau 2-2 : Les deux fonctions de base à coder.....	33
Tableau 2-3 : Explication de l'interface.....	37
Tableau 2-4 : Explication de l'interface.....	38
Tableau 3-1 : Ratio de compression des poudres selon la hauteur et la forme du trou.....	46

Liste des figures

Figure 1-1 : banc photo-élastique.....	21
Figure 2-1 : Presse mécanique.....	22
Figure 2-2 : Capteur de force FN3042.....	23
Figure 2-3 : Capteur de déplacement CLP13.....	23
Figure 2-4 : Arduino mega 2560.....	25
Figure 2-5 : Module bluetooth HC-06.....	26
Figure 2-6 : Puce MCP41HV51.....	27
Figure 2-7 : Configuration des broches du MCP41HV51.....	27
Figure 2-8 : Représentation électronique du INA122.....	28
Figure 2-9 : Schéma d'amplification pour le capteur de déplacement.....	29
Figure 2-10 : Schéma d'amplification pour le capteur de force.....	31
Figure 2-11 : Empreinte du PCB.....	31
Figure 2-12 : PCB finalisé.....	32
Figure 2-13 : Interface de programmation de l'Arduino.....	33
Figure 2-14 : Programme de l'Arduino.....	34
Figure 2-15 : Interface de pilotage/acquisition.....	37
Figure 2-16 : Interface de configuration.....	39
Figure 2-17 : Interface principale.....	40
Figure 2-18 : Plexiglas avec trou rond.....	41
Figure 2-19 : Plexiglas avec trou carré.....	42
Figure 2-20 : Schéma électrique de régulation.....	43
Figure 2-21 : Filtres polariseurs linéaires.....	43
Figure 2-22 : Dispositif de photoélasticité.....	44
Figure 3-1 : Comprimés une fois sortis du moule.....	45
Figure 3-2 : Compression selon la hauteur de poudre et de son type.....	46
Figure 3-3 : Comparaison en fonction de la forme du trou.....	47

Figure 3-4 : Décompression en fonction du déplacement.	47
Figure 3-5 : Pression d'éjection en fonction du déplacement.	48
Figure 3-6 : Franges d'étalonnage.	49
Figure 3-7 : Capture en fin de compression de p3 pour 12 mm de hauteur.	49
Figure 3-8 : Zoom sur l'interface.	50
Figure 3-9 : Série de photos lors de la compression.	51
Figure 3-10 : Comparaison entre la pression appliquée et la contrainte aux interfaces	51
Figure 3-11 : Comparaison entre apparition des franges.	52
Figure 3-12 : Contrainte par rapport à la pression appliquée.	52
Figure 3-13 : Contrainte par rapport à la pression appliquée.	53
Figure 3-14 : Photo suite à la décompression.	53
Figure 3-15 : Simulation d'une contrainte homogène.	54
Figure 3-16 : Simulation d'une contrainte inhomogène.	54
Figure A10.1 : Décomposition du vecteur E de l'onde lumineuse se propageant selon l'axe x	885
Figure A10.2 : Polarisation linéaire à 45° par rapport à l'axe y.	896
Figure A10.3 : Séparation des composantes.	87
Figure A10.4 : Angle de 90° entre rayon réfléchi et réfracté.	87
Figure A10.5 : Polarisation linéaire.	888
Figure A10.6 : Phénomène de biréfringence.	898
Figure A10.7 : Construction de Huygens.	89
Figure A10.8 : Répartitions des isostatiques.	90
Figure A10.9 : Expérience de photoélasticité.	91
Figure A10.10 : Position des lames quart d'onde.	92

Annexe 1 : Capteur de force FN3042



Modèle FN3042

Capteur de force pour applications en fatigue



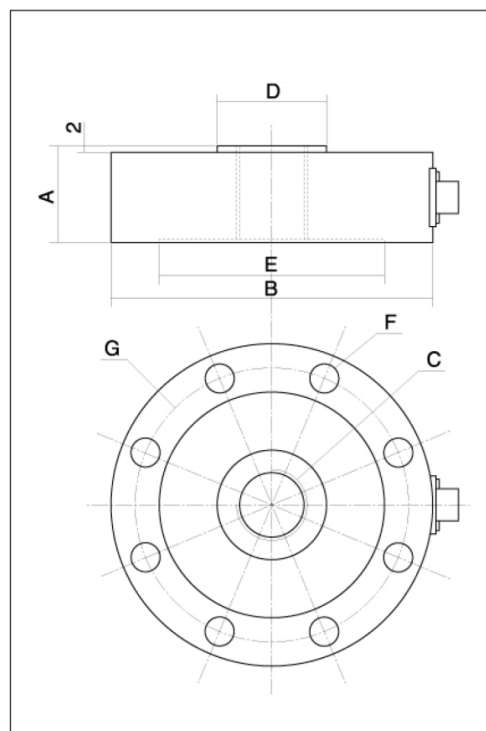
- Etendues de mesure de 0-5 kN à 0-500 kN
- Utilisation traction et compression
- Destiné aux applications en fatigue
- Version sortie haut niveau (amplificateur intégré)

Principalement destinés à être implantés sur des bancs d'essai de fatigue et d'endurance, les capteurs de la série FN3042 offrent, grâce à leur conception originale, une durée de vie largement supérieure à celle d'un capteur traditionnel.

Cette technologie assure une très grande stabilité du zéro dans le temps, principale caractéristique demandée à un capteur en essai d'endurance.

La sortie analogique haut niveau intégrée au capteur lui confère, une grande polyvalence et une facilité d'utilisation et d'exploitation.

Concepteur et producteur de ces capteurs, FGP Sensors dispose d'une grande expérience pour leur adaptation à une utilisation ou un environnement spécifique. De même FGP Sensors propose une vaste gamme d'électroniques de conditionnement et de traitement permettant l'alimentation du capteur, l'amplification du signal et l'affichage de la mesure sur indicateur numérique, pour vous fournir une chaîne de mesure complète, appairée, étalonnée et donc prête à l'emploi.



Caractéristiques mécaniques

E.M. en kN	A	B	C	D	E	F	G	Matière	Raideur en N/m
5	30	101	M16 x 2	34	70	8 x 8,2	85	AU4G	$1,7 \cdot 10^8$
10	30	101	M20 x 1,5	34	70	8 x 8,2	85	Acier Inox	$3 \cdot 10^8$
25	30	101	M20 x 1,5	34	70	8 x 8,2	85	Acier Inox	$6 \cdot 10^8$
50	40	119	M24 x 2	49	83	8 x 10,2	101	Acier Inox	$1,5 \cdot 10^9$
100	50	144	M36 x 3	66	104	8 x 12,2	124	Acier Inox	$2 \cdot 10^9$
200	50	168	M45 x 4	72	118	8 x 16,2	143	Acier Inox	$3,5 \cdot 10^9$
500	70	228	M64 x 4	102	152	16 x 20,2	190	Acier Inox	$6,5 \cdot 10^9$

Photos non contractuelles. Le constructeur se réserve le droit de changer sans préavis les spécifications indiquées. 20/04/2004

www.fgp-instrumentation.com • Tél. +33 (0)1 30 79 65 40 • Fax. +33 (0)1 30 54 01 43 • scom@fgp.tm.fr

Annexe 2 : Capteur de déplacement CLP13

Weg Sensoren

robust · kompakt · präzise

Series CLP13 - Potentiometric Linear Transducer

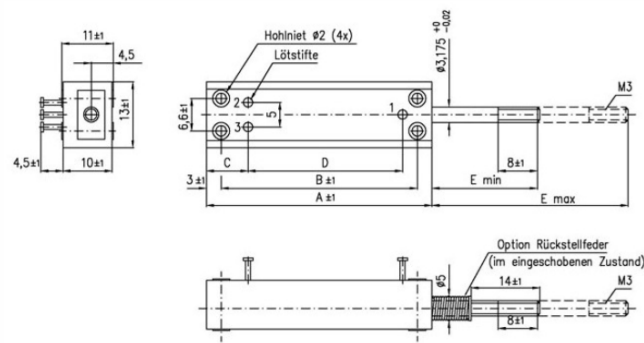
- Resolution quasi infinite
- Electrical travel from 13mm to 100mm
- Small dimensions
- Resistance values 500 Ohm to 20 kOhm
- Optional with spring return

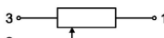


CLP13-100

In that series a high-resolutions conductive plastic element is integrated. Because of this, the transducer offers a long lifetime and a low price.

Drawing



Dimensions in mm					<div></div> <p>Diagram is equivalent to the shaft position in the above drawing</p>
Type	CLP 13-13	CLP 13-25	CLP 13-50	CLP 13-100	
A	38	51	76	127	
B	32	45	70	121	
C	8,5	8,5	8,5	8,5	
D	23,5	36,5	61,5	112,5	
E min.	19 ± 1	19 ± 1	19 ± 1	19 ± 1	
E max.	31,7 +3	44,4 +3	69,8 +3	120,6 +3	
With Spring Return					
E min.	30 ± 1	35 ± 1	40 ± 1	50 ± 1	
E max.	42,7 +3	60,4 +3	90,8 +3	151,6 +3	

Tip: At lowest strokes, and if high resolution and life expectancy are required, we recommend our inductive sensors with an internal electronic. They work with a direct d.c. voltage-input and -output.

Annexe 3 : Potentiomètre MCP41HVX1



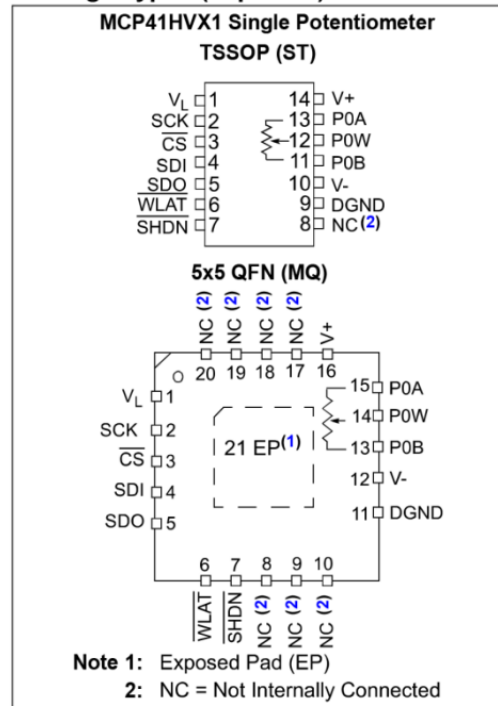
MCP41HVX1

7/8-Bit Single, +36V ($\pm 18V$) Digital POT with SPI Serial Interface and Volatile Memory

Features

- High-Voltage Analog Support:
 - +36V Terminal Voltage Range (DGND = V-)
 - $\pm 18V$ Terminal Voltage Range (DGND = V- + 18V)
- Wide Operating Voltage:
 - Analog: 10V to 36V (specified performance)
 - Digital: 2.7V to 5.5V
1.8V to 5.5V (DGND \geq V- + 0.9V)
- Single Resistor Network
- Potentiometer Configuration Options
- Resistor Network Resolution
 - 7-bit: 127 Resistors (128 Taps)
 - 8-bit: 255 Resistors (256 Taps)
- R_{AB} Resistance Options:
 - 5 k Ω - 10 k Ω
 - 50 k Ω - 100 k Ω
- High Terminal/Wiper Current (I_W) Support:
 - 25 mA (for 5 k Ω)
 - 12.5 mA (for 10 k Ω)
 - 6.5 mA (for 50 k Ω and 100 k Ω)
- Zero-Scale to Full-Scale Wiper Operation
- Low Wiper Resistance: 75 Ω (Typical)
- Low Tempco:
 - Absolute (Rheostat): 50 ppm Typical (0°C to +70°C)
 - Ratiometric (Potentiometer): 15 ppm Typical
- SPI Serial Interface (10 MHz, Modes 0, 0 and 1, 1)
- Resistor Network Terminal Disconnect Via:
 - Shutdown Pin (\overline{SHDN})
 - Terminal Control (TCON) Register
- Write Latch (\overline{WLAT}) Pin to control update of Volatile Wiper Register (such as Zero Crossing)
- Power-On Reset / Brown-Out Reset for both:
 - Digital Supply (V_L /DGND); 1.5V Typical
 - Analog Supply ($V+$ / V-); 3.5V Typical
- Serial Interface Inactive Current (3 μA Typical)
- 500 kHz Typical Bandwidth (-3 dB) Operation (5.0 k Ω Device)
- Extended Temperature Range (-40°C to +125°C)
- Package Types: TSSOP-14 and QFN-20 (5x5)

Package Types (Top View)



Description

The MCP41HVX1 family of devices have dual power rails (analog and digital). The analog power rail allows high voltage on the resistor network terminal pins. The analog voltage range is determined by the V+ and V- voltages. The maximum analog voltage is +36V, while the operating analog output minimum specifications are specified from either 10V or 20V. As the analog supply voltage becomes smaller, the analog switch resistances increase, which effect certain performance specifications. The system can be implemented as dual rail ($\pm 18V$) relative to the digital logic ground (DGND).

The device also has a Write Latch (\overline{WLAT}) function, which will inhibit the volatile wiper register from being updated (latched) with the received data, until the \overline{WLAT} pin is low. This allows the application to specify a condition where the volatile wiper register is updated (such as zero crossing).

Annexe 4 : Amplificateur INA122



INA122

Single Supply, *MicroPower* INSTRUMENTATION AMPLIFIER

FEATURES

- LOW QUIESCENT CURRENT: 60μA
- WIDE POWER SUPPLY RANGE
Single Supply: 2.2V to 36V
Dual Supply: -0.9/+1.3V to ±18V
- COMMON-MODE RANGE TO (V-)-0.1V
- RAIL-TO-RAIL OUTPUT SWING
- LOW OFFSET VOLTAGE: 250μV max
- LOW OFFSET DRIFT: 3μV/°C max
- LOW NOISE: 60nV/√Hz
- LOW INPUT BIAS CURRENT: 25nA max
- 8-PIN DIP AND SO-8 SURFACE-MOUNT

APPLICATIONS

- PORTABLE, BATTERY OPERATED SYSTEMS
- INDUSTRIAL SENSOR AMPLIFIER:
Bridge, RTD, Thermocouple
- PHYSIOLOGICAL AMPLIFIER:
ECG, EEG, EMG
- MULTI-CHANNEL DATA ACQUISITION

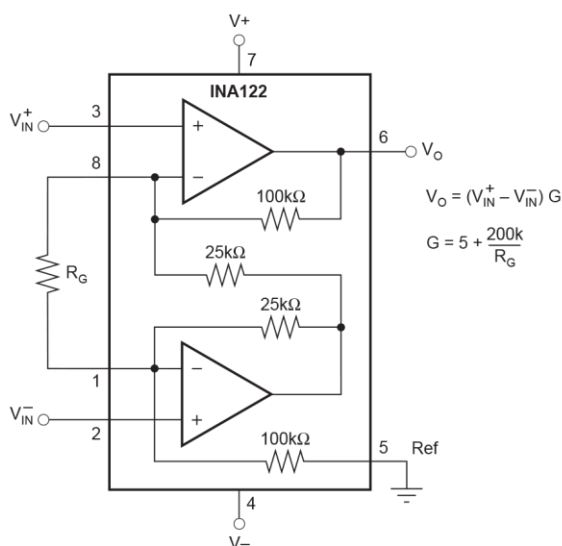
DESCRIPTION

The INA122 is a precision instrumentation amplifier for accurate, low noise differential signal acquisition. Its two-op-amp design provides excellent performance with very low quiescent current, and is ideal for portable instrumentation and data acquisition systems.

The INA122 can be operated with single power supplies from 2.2V to 36V and quiescent current is a mere 60μA. It can also be operated from dual supplies. By utilizing an input level-shift network, input common-mode range extends to 0.1V below negative rail (single supply ground).

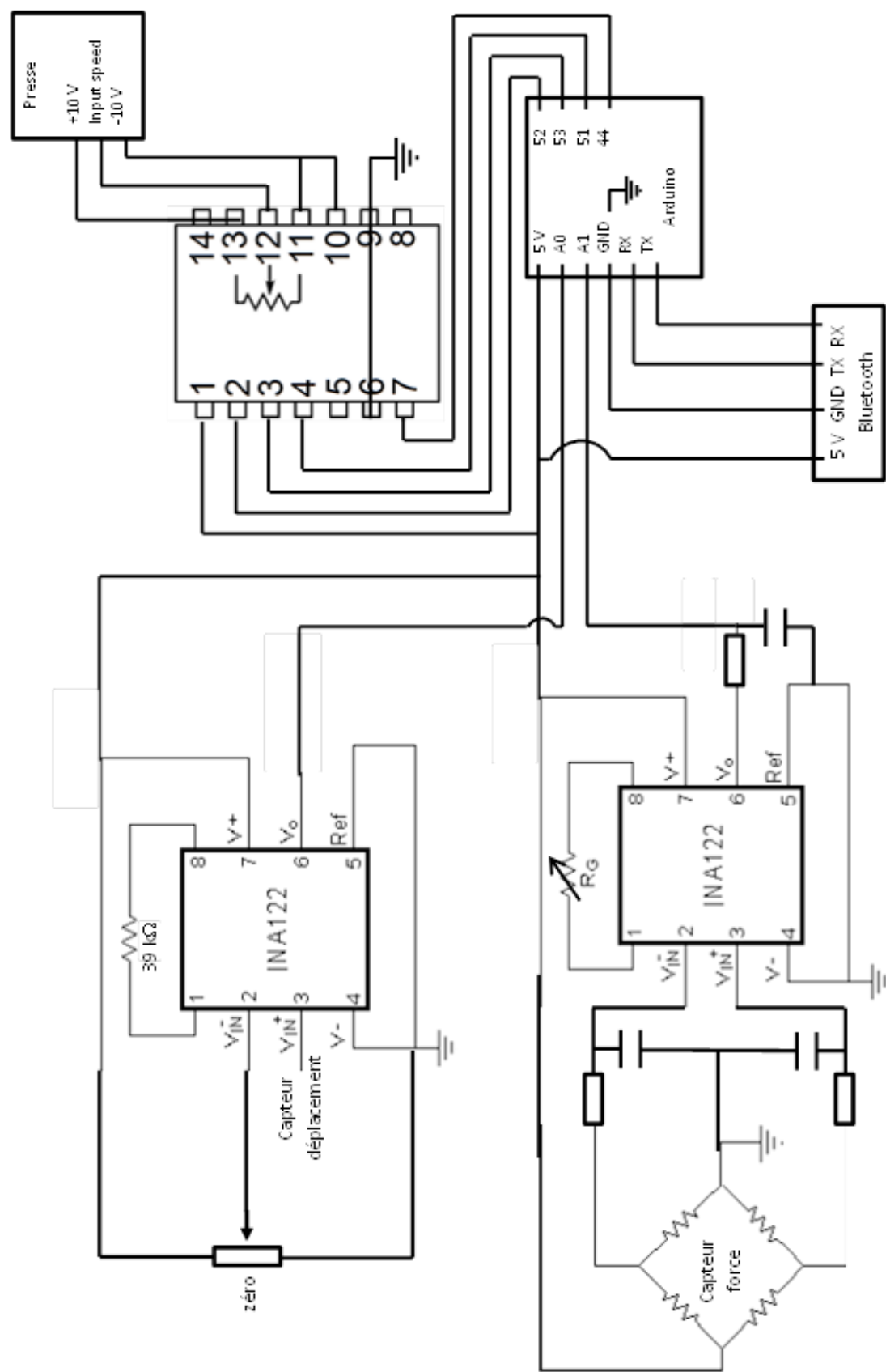
A single external resistor sets gain from 5V/V to 10000V/V. Laser trimming provides very low offset voltage (250μV max), offset voltage drift (3μV/°C max) and excellent common-mode rejection.

Package options include 8-pin plastic DIP and SO-8 surface-mount packages. Both are specified for the -40°C to +85°C extended industrial temperature range.



International Airport Industrial Park • Mailing Address: PO Box 11400, Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706 • Tel: (520) 746-1111 • Twx: 910-952-1111
Internet: <http://www.burr-brown.com/> • FAXLine: (800) 548-6133 (US/Canada Only) • Cable: BBRCORP • Telex: 066-6491 • FAX: (520) 889-1510 • Immediate Product Info: (800) 548-6132

Annexe 5 : Schéma électrique



Annexe 6 : Code C++ pour pc

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QtSerialPort/serialport.h>
#include <QGraphicsScene>

namespace Ui {
class MainWindow;
}

class SettingsDialog;
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent
= 0);
    ~MainWindow();

private slots:
    void open();
    void close();
    void read();
    void on_up_clicked();
    void on_down_clicked();
    void on_stop_clicked();
    void graph();
    void on_record_clicked();
    void lcdzero();
    void on_pplus_clicked();
    void on_pmoins_clicked();
    void on_goto0_clicked();

private:
    Ui::MainWindow *ui;
    SettingsDialog *settings;
    QSerialPort *serial;
    QGraphicsScene *scene;
};

#endif // MAINWINDOW_H

#ifndef SETTINGSDIALOG_H
#define SETTINGSDIALOG_H
#include <QDialog>
#include <QtSerialPort/QSerialPort>
QT_USE_NAMESPACE
QT_BEGIN_NAMESPACE

namespace Ui {
class SettingsDialog;
}

class QIntValidator;

QT_END_NAMESPACE

class SettingsDialog : public QDialog
{
    Q_OBJECT

public:
    struct Settings {
        QString name;
        quint32 baudRate;
        QString stringBaudRate;
        QSerialPort::DataBits dataBits;
        QString stringDataBits;
        QSerialPort::Parity parity;
        QString stringParity;
        QSerialPort::StopBits stopBits;
        QString stringStopBits;
        QSerialPort::FlowControl
flowControl;
        QString stringFlowControl;
        QString coef;
        QString ard;
        QString diam;
        QString haut;
        QString typ;
        int eta;
    };

    explicit SettingsDialog(QWidget
*parent = 0);
    ~SettingsDialog();

    Settings settings() const;

private slots:
    void showPortInfo(int idx);
    void apply();
    void checkCustomBaudRatePolicy(int
idx);
    void on_update_clicked();
    void on_textEdit_textChanged();
    void on_etabut_clicked();

private:
    void fillPortsParameters();
    void fillPortsInfo();
    void updateSettings();

private:
    Ui::SettingsDialog *ui;
    Settings currentSettings;
    QIntValidator *intValidator;
};

#endif // SETTINGSDIALOG_H

```

```

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>
#include "settingsdialog.h"
#include <QFile>
#include <QTextStream>
#include <QDateTime>

QFile ecr;
int i=0;
int j=0;
int l=0;
int m=0;
QString datas;
int sansm;
float sansu;
int got0;
int coe;
int rec=0;
int id;
float moyf[10];
float moyd[10];
float sect;
QString vitesse= "";
int etal=0;
QString sansms;
QString sansus;
float moyed=0;
float moyef=0;
int gt=0;

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setWindowTitle("Compressor - (F.Douit)");
    QApplication::setWindowIcon(QIcon(":/arduino/images/sablier.png"));
    setStyleSheet("QMainWindow {border-image: url(:/arduino/images/fond.jpg)}");

    setFixedSize(1000,420);
    setWindowFlags(Qt::CustomizeWindowHint);
    setWindowFlags(Qt::WindowTitleHint);
    setWindowFlags(Qt::WindowSystemMenuHint);
    setWindowFlags(Qt::WindowMinimizeButtonHint | Qt::WindowCloseButtonHint);

    ui->mainToolBar->setStyleSheet("QToolBar {border: 0px}");

    ui->actionConnect->setEnabled(true);
    ui->actionDisconnect->setEnabled(false);
    ui->up->setEnabled(false);
    ui->down->setEnabled(false);
    ui->stop->setEnabled(false);
    ui->pplus->setEnabled(false);
    ui->pmoins->setEnabled(false);
    ui->lineEdit->setEnabled(false);
    ui->lineEdit_3->setEnabled(false);
    ui->lineEdit_2->setEnabled(false);
    ui->goto0->setEnabled(false);
    ui->actionConfigure->setEnabled(true);
    ui->record->setText("not recording");
    ui->record->setEnabled(true);
    ui->up->setAutoRepeat(true);
    ui->down->setAutoRepeat(true);

    serial = new QSerialPort(this);
    settings = new SettingsDialog;
    scene = new QGraphicsScene;

    connect(ui->actionConnect, SIGNAL(triggered()), this, SLOT(open()));
    connect(ui->actionDisconnect, SIGNAL(triggered()), this, SLOT(close()));
    connect(ui->actionClear, SIGNAL(triggered()), ui->plainTextEdit, SLOT(clear()));
    connect(ui->actionClear, SIGNAL(triggered()), ui->plainTextEdit_2, SLOT(clear()));
    connect(ui->actionClear, SIGNAL(triggered()), this, SLOT(lcdzero()));
    connect(ui->actionClear, SIGNAL(triggered()), scene, SLOT(clear()));
    connect(ui->actionConfigure, SIGNAL(triggered()), settings, SLOT(show()));

    scene->setSceneRect(0,0,ui->graphicsView->size().width(),ui->graphicsView->viewport()-
>size().height());
    ui->graphicsView->setScene(scene);
}

MainWindow::~MainWindow()
{
    delete settings;
    delete ui;
}

void MainWindow::lcdzero()
{
    ui->lcd_read->display(0);
    ui->lcd_read_2->display(0);
}

void MainWindow::open()
{
    SettingsDialog::Settings p = settings->settings();
    serial->setPortName(p.name);
    etal=p.etal;

    if(p.haut =="" || p.typ == "")
    {
        QMessageBox::warning(this,"paramètres","renseigner la hauteur de grains, son type et vérifier le
diamètre dans les paramètres");
    }
    else
    {
        sect= (p.diam.toFloat()/2000)*(p.diam.toFloat()/2000)*3.14159;
    }
}

```

```

if (p.ard=="2341")
{
    serial->open(QIODevice::ReadWrite);
    serial->setBaudRate(p.baudRate);
    serial->setDataBits(p.dataBits);
    serial->setParity(p.parity);
    serial->setStopBits(p.stopBits);
    serial->setFlowControl(p.flowControl);
    ui->lineEdit->setText("0");

    coe= p.coeef.toInt();
    if(rec==1)
    {
        QDateTime dt=QDateTime::currentDateTime();
        QString dt2=dt.toString("dd.MM.yy hh.mm.ss");
        QString dt3=".\\datas\\"+dt2+".txt";
        ecr.setFileName(dt3);
        ecr.open(QIODevice::ReadWrite | QIODevice::Text | QIODevice::Truncate);
        QTextStream ecrId(&ecr);
        vitesse = ui->lineEdit->text();

        ecrId<<"deplacement(mm);force(kg);moydeplacement(mm);moyforce(kg);vitesse: "+vitesse+"%; diam:
"+p.diam+" mm; sect: "+QString::number(sect).replace(".",",")+" m2; delay: 10x"+p.coeef+" ms; hauteur: "+p.haut+"
mm; type: "+p.typ<<endl;
    }

    connect(serial, SIGNAL(readyRead()), this, SLOT(read()));

    ui->actionConnect->setEnabled(false);
    ui->actionDisconnect->setEnabled(true);
    ui->record->setEnabled(false);
    ui->up->setEnabled(true);
    ui->down->setEnabled(true);
    ui->stop->setEnabled(true);
    ui->pplus->setEnabled(true);
    ui->pmoins->setEnabled(true);
    ui->lineEdit->setEnabled(true);
    ui->lineEdit_3->setEnabled(true);
    ui->goto0->setEnabled(true);
    ui->lineEdit_3->setText("5");
    ui->lineEdit_2->setEnabled(true);
    ui->lineEdit_2->setText("128");
    ui->actionConfigure->setEnabled(false);
    ui->statusBar->showMessage(tr("Connected to %1 : %2, %3, %4, %5, %6")
        .arg(p.name)
        .arg(p.stringBaudRate)
        .arg(p.stringDataBits)
        .arg(p.stringParity)
        .arg(p.stringStopBits)
        .arg(p.stringFlowControl));

}
else
{
    QMessageBox::warning(this,"port","vérifier numéro COM");
    ui->statusBar->showMessage(tr("Configure error"));
}
}

void MainWindow::close()
{
    serial->flush();
    disconnect(serial, SIGNAL(readyRead()), this, SLOT(read()));
    serial->close();
    ui->actionConnect->setEnabled(true);
    ui->actionDisconnect->setEnabled(false);
    ui->actionConfigure->setEnabled(true);
    ui->record->setEnabled(true);
    ui->statusBar->showMessage("déconnecté");
    ui->up->setEnabled(false);
    ui->down->setEnabled(false);
    ui->stop->setEnabled(false);
    ui->pplus->setEnabled(false);
    ui->pmoins->setEnabled(false);
    ui->lineEdit->setEnabled(false);
    ui->goto0->setEnabled(false);
    ui->lineEdit->setText("");
    ui->lineEdit_3->setEnabled(false);
    ui->lineEdit_3->setText("");
    ui->lineEdit_2->setEnabled(false);
    ui->lineEdit_2->setText("");
    if(rec==1)
    {
        ecr.close();
    }
}

void MainWindow::read()
{
    if(sansm > ui->lineEdit_fmax->text().toInt())
    {
        on_stop_clicked();
    }

    if(gt==0)
    {
        if(got0 < 5)
        {
            if(ui->lineEdit->text().toInt() !=0)
            {
                on_stop_clicked();
            }
        }
    }

    if(gt==1)
    {
        if(got0 < 15)
        {
            on_stop_clicked();
        }
    }
}

```

```

        }
        gt=0;
    }

    QTextStream ecri(&ecr);
    char c;
    while (serial->bytesAvailable() > 0)
    {
        serial->getChar(&c);
        if(c == '\n')
        {
            i++;
            if(i%coe==0)
            {
                QStringList cher = datas.split(";");
                for (int k=0; k<cher.size(); k++)
                {
                    QString aff = cher.at(k);
                    if(aff.startsWith("-"))
                    {
                        sansms = aff.replace("-", "");
                        if(etal == 1)
                        {
                            ui->plainTextEdit->insertPlainText(sansms+"\n");
                            ui->plainTextEdit->ensureCursorVisible();
                        }
                        else
                        {
                            sansm= sansms.toInt()*2.3279 - 85.121;
                            ui->plainTextEdit->insertPlainText(QString::number(sansm)+"\n");
                            ui->plainTextEdit->ensureCursorVisible();
                            moyf[l]= sansm;
                            l++;
                            if(l==10)
                            {
                                float sommf=0;
                                for(int o=0; o<10; o++)
                                {
                                    sommf += moyf[o];
                                }
                                moyef=sommf/10;
                                ui->lcd_read->display(QString::number(((moyef*9.81)/sect)/1000000));
                                l=0;
                            }
                        }
                    }
                    if(aff.startsWith("_"))
                    {
                        sansus = aff.replace("_", "");
                        got0=sansus.toInt();
                        if(etal == 1)
                        {
                            ui->plainTextEdit_2->insertPlainText(sansus+"\n");
                            ui->plainTextEdit_2->ensureCursorVisible();
                        }
                        else
                        {
                            sansu= sansus.toFloat()*0.0098 - 0.1136;
                            ui->plainTextEdit_2->insertPlainText(QString::number(sansu, 'g', 3)+"\n");
                            ui->plainTextEdit_2->ensureCursorVisible();
                            moyd[m]= sansu;
                            m++;
                            if(m==10)
                            {
                                float sommed=0;
                                for(int oo=0; oo<10; oo++)
                                {
                                    sommed += moyd[oo];
                                }
                                moyed=sommed/10;
                                ui->lcd_read_2->display(QString::number(moyed, 'g', 3));
                                m=0;
                            }
                        }
                    }
                }
            }
        }
    }

    graph();
    if(rec==1)
    {
        if(etal == 1)
        {
            if(vitesse!= ui->lineEdit->text())
            {
                vitesse= ui->lineEdit->text();
                ecri<<sansus.replace(".", ",")+"<<sansms+"<<"<<vitesse<<endl;
            }
            else
            {
                ecri<<sansus.replace(".", ",")+"<<sansms<<endl;
            }
        }
    }
}

```

```

        else
        {
            if(vitesse!= ui->lineEdit->text())
            {
                vitesse= ui->lineEdit->text();
                ecri<<QString::number(sansu,
'g',3).replace(".",",","")+";"<<QString::number(sansm)+";"<<QString::number(moyed,
'g',3).replace(".",",","")+";"<<QString::number(moyef).replace(".",",","")+";"<<vitesse<<endl;
            }
            else
            {
                ecri<<QString::number(sansu,
'g',3).replace(".",",","")+";"<<QString::number(sansm)+";"<<QString::number(moyed,
'g',3).replace(".",",","")+";"<<QString::number(moyef).replace(".",",","")<<endl;
            }
        }
    }
    }
    datas.clear();
}
else
{
    datas.append(c);
}
}

void MainWindow::graph()
{
    int zero = ui->graphicsView->viewport()->size().height()-2;
    j++;

    int val1=sansm*252/2000;
    int val2=sansu*252/10;

    scene->setSceneRect(0,0,ui->graphicsView->size().width()+j,ui->graphicsView->viewport()-
>size().height());

    scene->addEllipse(j,-val1+zero,1,1,QPen(Qt::black));
    scene->addEllipse(j,-val2+zero,1,1,QPen(Qt::blue));
    ui->graphicsView->ensureVisible(j,0,0,0);
}

void MainWindow::on_goto0_clicked()
{
    gt=1;

    QByteArray gotoz;

    QDataStream streamgot (&gotoz, QIODevice::WriteOnly);

    streamgot << 0;

    ui->lineEdit_2->setText(QString::number(0));

    serial->write(gotoz);
    serial->flush();
    ui->lineEdit->setText(QString::number(-100));
}

void MainWindow::on_up_clicked()
{
    QByteArray valeurup;
    QString up= ui->lineEdit->text();
    int vup= up.toInt();
    QString passup= ui->lineEdit_3->text();
    int pasup= passup.toInt();
    if (passup.isEmpty())
    {
        pasup=0;
        ui->lineEdit_3->setText(QString::number(5));
    }

    if (up.isEmpty())
    {
        vup=0;
        ui->lineEdit->setText(QString::number(0));
    }

    if (vup+pasup>100)
    {
        vup=100-pasup;
        ui->lineEdit->setText(QString::number(100));
    }

    qreal valupr= (vup+pasup)*1.27 +128;
    int valup= qRound(valupr);

    QDataStream streamup (&valeurup, QIODevice::WriteOnly);

    streamup << valup;

    ui->lineEdit_2->setText(QString::number(valup));

    serial->write(valeurup);
    serial->flush();
    ui->lineEdit->setText(QString::number(vup+pasup));
}

void MainWindow::on_down_clicked()
{
    QByteArray valeurdwn;
    QString dwn= ui->lineEdit->text();
    int vdwn=dwn.toInt();
    QString passdwn= ui->lineEdit_3->text();
    int pasdwn= passdwn.toInt();
    if (passdwn.isEmpty())
    {
        pasdwn=0;
        ui->lineEdit_3->setText(QString::number(5));
    }
}

```



```

    if (dwn.isEmpty())
    {
        vdown=0;
        ui->lineEdit->setText(QString::number(0));
    }

    if (vdown-pasdown<-100)
    {
        vdown= -100+pasdown;
        ui->lineEdit->setText(QString::number(-100));
    }

    qreal valdownr= (vdown-pasdown)*1.27 +128;
    int valdown= qRound(valdownr);

    QDataStream streamdown (&valeurdown, QIODevice::WriteOnly);
    streamdown << valdown;

    ui->lineEdit_2->setText(QString::number(valdown));
    serial->write(valeurdown);
    serial->flush();
    ui->lineEdit->setText(QString::number(vdown-pasdown));
}

void MainWindow::on_stop_clicked()
{
    QByteArray stop;
    QDataStream streamstop(&stop, QIODevice::WriteOnly);
    streamstop << 128;

    ui->lineEdit_2->setText(QString::number(128));

    serial->write(stop);
    serial->flush();
    ui->lineEdit->setText(QString::number(0));
}

void MainWindow::on_record_clicked()
{
    if(rec==0)
    {
        rec=1;
        ui->record->setText("recording...");
    }

    else
    {
        rec=0;
        ui->record->setText("not recording");
    }
}

void MainWindow::on_pplus_clicked()
{
    QString pasc= ui->lineEdit_3->text();
    int pasp = pasc.toInt()+5;
    if (pasp>50)
    {
        pasp=50;
    }
    ui->lineEdit_3->setText(QString::number(pasp));
}

void MainWindow::on_pmoins_clicked()
{
    QString pasc= ui->lineEdit_3->text();
    int pasm = pasc.toInt()-5;
    if (pasm<0)
    {
        pasm=0;
    }
    ui->lineEdit_3->setText(QString::number(pasm));
}

```

```

#include "settingsdialog.h"
#include "ui_settingsdialog.h"
#include <QtSerialPort/QSerialPortInfo>
#include <QIntValidator>
#include <QLineEdit>
#include <QMessageBox>

QT_USE_NAMESPACE

int eta0=0;

SettingsDialog::SettingsDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::SettingsDialog)
{
    ui->setupUi(this);

    intValidator = new QIntValidator(0, 4000000, this);

    ui->baudRateBox->setInsertPolicy(QComboBox::NoInsert);

    connect(ui->applyButton, SIGNAL(clicked()),this, SLOT(apply()));
    connect(ui->serialPortInfoListBox, SIGNAL(currentIndexChanged(int)),this, SLOT(showPortInfo(int)));
    connect(ui->baudRateBox, SIGNAL(currentIndexChanged(int)),this, SLOT(checkCustomBaudRatePolicy(int)));

    fillPortsParameters();
    fillPortsInfo();
    updateSettings();
}

SettingsDialog::~SettingsDialog()
{
    delete ui;
}

SettingsDialog::Settings SettingsDialog::settings() const
{
    return currentSettings;
}

void SettingsDialog::showPortInfo(int idx)
{
    if (idx != -1)
    {
        QStringList list = ui->serialPortInfoListBox->itemData(idx).toStringList();
        ui->descriptionLabel->setText(tr("Description: %1").arg(list.at(1)));
        ui->manufacturerLabel->setText(tr("Manufacturer: %1").arg(list.at(2)));
        ui->locationLabel->setText(tr("Location: %1").arg(list.at(3)));
        ui->vidLabel->setText(tr("Vendor Identifier: %1").arg(list.at(4)));
        currentSettings.ard=list.at(4);
        ui->pidLabel->setText(tr("Product Identifier: %1").arg(list.at(5)));
    }
}

void SettingsDialog::apply()
{
    if(ui->textEdit_haut->toPlainText()==" " || ui->textEdit_typ->toPlainText()==" ")
    {
        QMessageBox::warning(this,"paramètres","renseigner la hauteur de grains, son type et vérifier le
diamètre");
    }
    else
    {
        updateSettings();
        hide();
    }
}

void SettingsDialog::checkCustomBaudRatePolicy(int idx)
{
    bool isCustomBaudRate = !ui->baudRateBox->itemData(idx).isValid();
    ui->baudRateBox->setEditable(isCustomBaudRate);
    if (isCustomBaudRate)
    {
        ui->baudRateBox->clearEditText();
        QLineEdit *edit = ui->baudRateBox->lineEdit();
        edit->setValidator(intValidator);
    }
}

void SettingsDialog::fillPortsParameters()
{
    ui->baudRateBox->addItem(QLatin1String("9600"), QSerialPort::Baud9600);
    ui->baudRateBox->addItem(QLatin1String("19200"), QSerialPort::Baud19200);
    ui->baudRateBox->addItem(QLatin1String("38400"), QSerialPort::Baud38400);
    ui->baudRateBox->addItem(QLatin1String("115200"), QSerialPort::Baud115200);
    ui->baudRateBox->addItem(QLatin1String("Custom"));

    ui->dataBitsBox->addItem(QLatin1String("5"), QSerialPort::Data5);
    ui->dataBitsBox->addItem(QLatin1String("6"), QSerialPort::Data6);
    ui->dataBitsBox->addItem(QLatin1String("7"), QSerialPort::Data7);
    ui->dataBitsBox->addItem(QLatin1String("8"), QSerialPort::Data8);
    ui->dataBitsBox->setCurrentIndex(3);

    ui->parityBox->addItem(QLatin1String("None"), QSerialPort::NoParity);
    ui->parityBox->addItem(QLatin1String("Even"), QSerialPort::EvenParity);
    ui->parityBox->addItem(QLatin1String("Odd"), QSerialPort::OddParity);
    ui->parityBox->addItem(QLatin1String("Mark"), QSerialPort::MarkParity);
    ui->parityBox->addItem(QLatin1String("Space"), QSerialPort::SpaceParity);

    ui->stopBitsBox->addItem(QLatin1String("1"), QSerialPort::OneStop);
#ifdef Q_OS_WIN
    ui->stopBitsBox->addItem(QLatin1String("1.5"), QSerialPort::OneAndHalfStop);
#endif
    ui->stopBitsBox->addItem(QLatin1String("2"), QSerialPort::TwoStop);

    ui->flowControlBox->addItem(QLatin1String("None"), QSerialPort::NoFlowControl);
    ui->flowControlBox->addItem(QLatin1String("RTS/CTS"), QSerialPort::HardwareControl);
    ui->flowControlBox->addItem(QLatin1String("XON/XOFF"), QSerialPort::SoftwareControl);
}

```

```

        ui->textEdit->setText("10");
        ui->lineEditglob->setText("100 ms");
        ui->lineEditmoy->setText("1000 ms");
        ui->textEdit_diam->setText("12");
    }

    void SettingsDialog::fillPortsInfo()
    {
        ui->serialPortInfoListBox->clear();
        foreach (const QSerialPortInfo &info, QSerialPortInfo::availablePorts())
        {
            QStringList list;
            list << info.portName()
                << info.description()
                << info.manufacturer()
                << info.systemLocation()
                << (info.vendorIdentifier() ? QString::number(info.vendorIdentifier(), 16) : QString())
                << (info.productIdIdentifier() ? QString::number(info.productIdIdentifier(), 16) : QString());

            ui->serialPortInfoListBox->addItem(list.first(), list);
        }
    }

    void SettingsDialog::updateSettings()
    {
        currentSettings.name = ui->serialPortInfoListBox->currentText();

        if (ui->baudRateBox->currentIndex() == 4)
        {
            currentSettings.baudRate = ui->baudRateBox->currentText().toInt();
        }
        else
        {
            currentSettings.baudRate = static_cast<QSerialPort::BaudRate>(ui->baudRateBox->itemData(ui->baudRateBox->currentIndex()).toInt());
        }

        currentSettings.stringBaudRate = QString::number(currentSettings.baudRate);

        currentSettings.dataBits = static_cast<QSerialPort::DataBits>(ui->dataBitsBox->itemData(ui->dataBitsBox->currentIndex()).toInt());
        currentSettings.stringDataBits = ui->dataBitsBox->currentText();

        currentSettings.parity = static_cast<QSerialPort::Parity>(ui->parityBox->itemData(ui->parityBox->currentIndex()).toInt());
        currentSettings.stringParity = ui->parityBox->currentText();

        currentSettings.stopBits = static_cast<QSerialPort::StopBits>(ui->stopBitsBox->itemData(ui->stopBitsBox->currentIndex()).toInt());
        currentSettings.stringStopBits = ui->stopBitsBox->currentText();

        currentSettings.flowControl = static_cast<QSerialPort::FlowControl>(ui->flowControlBox->itemData(ui->flowControlBox->currentIndex()).toInt());
        currentSettings.stringFlowControl = ui->flowControlBox->currentText();

        currentSettings.coef= ui->textEdit->toPlainText();
        currentSettings.diam= ui->textEdit_diam->toPlainText();
        currentSettings.haut= ui->textEdit_haut->toPlainText();
        currentSettings.typ= ui->textEdit_typ->toPlainText();
        currentSettings.eta= eta0;
    }

    void SettingsDialog::on_update_clicked()
    {
        fillPortsInfo();
    }

    void SettingsDialog::on_textEdit_textChanged()
    {
        QString dval=ui->textEdit->toPlainText();
        ui->lineEditglob->setText(QString::number(dval.toInt()*10)+" ms");
        ui->lineEditmoy->setText(QString::number(dval.toInt()*100)+" ms");
    }

    void SettingsDialog::on_etabut_clicked()
    {
        if(eta0==0)
        {
            eta0=1;
            ui->etabut->setText("étalonnage on");
        }
        else
        {
            eta0=0;
            ui->etabut->setText("étalonnage off");
        }
    }
}

```

Annexe 7 : Code C++ pour Android

```

#ifndef CLIENT_H
#define CLIENT_H

#include <qbluetoothserviceinfo.h>

#include <QtCore/QObject>

QT_FORWARD_DECLARE_CLASS(Q
BluetoothSocket)

QT_USE_NAMESPACE

class client : public QObject
{
    Q_OBJECT

public:
    explicit client(QObject *parent = 0);
    ~client();

    void startClient();
    void stopClient();

public slots:
    void sendMessage(const QString
&message);

signals:
    void messageReceived(const QString
&message);
    void disconnected();

private slots:
    void readSocket();

private:
    QBluetoothSocket *socket;
};

#endif // CLIENT_H

```

```

#ifndef SERVER_H
#define SERVER_H

#include <qbluetoothserviceinfo.h>
#include <qbluetoothaddress.h>

#include <QtCore/QObject>
#include <QtCore/QStringList>

QT_FORWARD_DECLARE_CLASS(Q
BluetoothServer)
QT_FORWARD_DECLARE_CLASS(Q
BluetoothSocket)

QT_USE_NAMESPACE

class server : public QObject
{
    Q_OBJECT

public:
    explicit server(QObject *parent = 0);
    ~server();

    void startServer(const
QBluetoothAddress &localAdapter);// =
QBluetoothAddress());
    void stopServer();

public slots:
    void sendMessage(const QString
&message);

signals:
    void messageReceived(const QString
&message);

private slots:
    void clientConnected();
    void clientDisconnected();
    void readSocket();

private:
    QBluetoothServer *rfcommServer;
    QBluetoothServiceInfo serviceInfo;
    QList<QBluetoothSocket *>
clientSockets;
};

#endif // SERVER_H

```

```

#ifndef MAINDIALOG_H
#define MAINDIALOG_H

#include <QDialog>
#include <qbluetoothserviceinfo.h>
#include <qbluetoothsocket.h>
#include <qbluetoothhostinfo.h>
#include <QGraphicsScene>

#include <QDebug>

QT_USE_NAMESPACE

class server;
class client;
class settingsdialog;

namespace Ui {
class maindialog;
}

class maindialog : public QDialog
{
    Q_OBJECT

public:
    maindialog(QWidget *parent = 0);
    ~maindialog();

signals:
    void sendMessage(const QString
&message);

private slots:
    void showMessage(const QString
&message);
    void clientDisconnected();
    void on_pauseButton_clicked();
    void on_connectButton_clicked();
    void on_config_clicked();
    void on_ecrire_clicked();
    void on_up_clicked();
    void on_down_clicked();
    void on_goto0_clicked();
    void on_stop_clicked();
    void recuset(const QString &Coef, const
QString &Diam, const QString &Haut, const QString
&Typ);
    void graph();

private:
    Ui::maindialog *ui;
    server *serv;
    QList<client *> clients;
    client * cli;
    QList<QBluetoothHostInfo> localAdapters;
    QString localName;
    QGraphicsScene *scene;
};

#endif // MAINDIALOG_H

```

```

#ifndef SETTINGSDIALOG_H
#define SETTINGSDIALOG_H

#include <QDialog>

namespace Ui {
class settingsdialog;
}

class settingsdialog : public QDialog
{
    Q_OBJECT

public:
    explicit settingsdialog(QWidget
*parent = 0);
    ~settingsdialog();

private slots:
    void on_apply_clicked();

private:
    Ui::settingsdialog *ui;

signals:
    void sendset(const QString &coef,
const QString &diam, const QString &haut,
const QString &typ);
};

#endif // SETTINGSDIALOG_H

```

```

#include "server.h"
#include <qbluetoothserver.h>
#include <qbluetoothsocket.h>
#include <qbluetoothlocaldevice.h>

static const QLatin1String serviceUuid("e8e10f95-1a70-4b27-9ccf-02010264e9c8");

server::server(QObject *parent)
: QObject(parent), rfcommServer(0)
{
}

server::~~server()
{
    stopServer();
}

void server::startServer(const QBluetoothAddress& localAdapter)
{
    if (rfcommServer)
        return;

    //! [Create the server]
    rfcommServer = new QBluetoothServer(QBluetoothServiceInfo::RfcommProtocol, this);
    connect(rfcommServer, SIGNAL(newConnection()), this, SLOT(clientConnected()));
    bool result = rfcommServer->listen(localAdapter);
    if (!result) {
        qWarning() << "Cannot bind chat server to" << localAdapter.toString();
        return;
    }
    //! [Create the server]

    //serviceInfo.setAttribute(QBluetoothServiceInfo::ServiceRecordHandle, (uint)0x00010010);

    //! [Class Uuid must contain at least 1 entry]
    QBluetoothServiceInfo::Sequence classId;

    classId << QVariant::fromValue(QBluetoothUuid(QBluetoothUuid::SerialPort));
    serviceInfo.setAttribute(QBluetoothServiceInfo::BluetoothProfileDescriptorList,
                            classId);

    classId.prepend(QVariant::fromValue(QBluetoothUuid(serviceUuid)));

    serviceInfo.setAttribute(QBluetoothServiceInfo::ServiceClassIds, classId);
    serviceInfo.setAttribute(QBluetoothServiceInfo::BluetoothProfileDescriptorList, classId);
    //! [Class Uuid must contain at least 1 entry]

    //! [Service name, description and provider]
    serviceInfo.setAttribute(QBluetoothServiceInfo::ServiceName, tr("Bt Chat Server"));
    serviceInfo.setAttribute(QBluetoothServiceInfo::ServiceDescription,
                             tr("Example bluetooth chat server"));
    serviceInfo.setAttribute(QBluetoothServiceInfo::ServiceProvider, tr("qt-project.org"));
    //! [Service name, description and provider]

    //! [Service UUID set]
    serviceInfo.setServiceUuid(QBluetoothUuid(serviceUuid));
    //! [Service UUID set]

    //! [Service Discoverability]
    QBluetoothServiceInfo::Sequence publicBrowse;
    publicBrowse << QVariant::fromValue(QBluetoothUuid(QBluetoothUuid::PublicBrowseGroup));
    serviceInfo.setAttribute(QBluetoothServiceInfo::BrowseGroupList,
                             publicBrowse);
    //! [Service Discoverability]

    //! [Protocol descriptor list]
    QBluetoothServiceInfo::Sequence protocolDescriptorList;
    QBluetoothServiceInfo::Sequence protocol;
    protocol << QVariant::fromValue(QBluetoothUuid(QBluetoothUuid::L2cap));
    protocolDescriptorList.append(QVariant::fromValue(protocol));
    protocol.clear();
    protocol << QVariant::fromValue(QBluetoothUuid(QBluetoothUuid::Rfcomm));
    protocol << QVariant::fromValue(quint8(rfcommServer->serverPort()));
    protocolDescriptorList.append(QVariant::fromValue(protocol));
    serviceInfo.setAttribute(QBluetoothServiceInfo::ProtocolDescriptorList,
                             protocolDescriptorList);
    //! [Protocol descriptor list]

    //! [Register service]
    serviceInfo.registerService(localAdapter);
    //! [Register service]
}

void server::stopServer()
{
    // Unregister service
    serviceInfo.unregisterService();

    // Close sockets
    qDeleteAll(clientSockets);

    // Close server
    delete rfcommServer;
    rfcommServer = 0;
}

void server::readSocket()
{
    QBluetoothSocket *socket = qobject_cast<QBluetoothSocket *>(sender());
    if (!socket)
        return;

    while (socket->canReadLine()) {
        QByteArray line = socket->readLine().trimmed();
        emit messageReceived(QString::fromUtf8(line.constData(), line.length()));
    }
}

void server::sendMessage(const QString &message)
{
    QByteArray text;// = message.toUtf8() + '\n';
    QDataStream streamsend(&text, QIODevice::WriteOnly);
    streamsend << message.toInt();
}

```



```

        foreach (QBluetoothSocket *socket, clientSockets)
            socket->write(text);
    }

void server::clientConnected()
{
    QBluetoothSocket *socket = rfcommServer->nextPendingConnection();
    if (!socket)
        return;

    connect(socket, SIGNAL(readyRead()), this, SLOT(readSocket()));
    connect(socket, SIGNAL(disconnected()), this, SLOT(clientDisconnected()));
    clientSockets.append(socket);
}

void server::clientDisconnected()
{
    QBluetoothSocket *socket = qobject_cast<QBluetoothSocket *>(sender());
    if (!socket)
        return;

    clientSockets.removeOne(socket);

    socket->deleteLater();
}

#include "client.h"
#include <qbluetoothsocket.h>

client::client(QObject *parent)
: QObject(parent), socket(0)
{
}

client::~client()
{
    stopClient();
}

static const QLatin1String sUuid("00001101-0000-1000-8000-00805F9B34FB");

void client::startClient()//const QBluetoothServiceInfo &remoteService)
{
    if (socket)
        return;

    // Connect to service
    socket = new QBluetoothSocket(QBluetoothServiceInfo::RfcommProtocol);
    qDebug() << "Create socket";

    const QBluetoothAddress ad("98:d3:31:20:23:4a" );
    const QBluetoothUuid uid(sUuid);

    socket->connectToService(ad,uid);//remoteService);
    qDebug() << "ConnectToService done";

    connect(socket, SIGNAL(readyRead()), this, SLOT(readSocket()));
    connect(socket, SIGNAL(disconnected()), this, SIGNAL(disconnected()));
}

void client::stopClient()
{
    delete socket;
    socket = 0;
}

void client::readSocket()
{
    if (!socket)
        return;

    while (socket->canReadLine()) {
        QByteArray line = socket->readLine();
        emit messageReceived(QString::fromUtf8(line.constData(), line.length()));
    }
}

void client::sendMessage(const QString &message)
{
    QByteArray text;// = message.toUtf8() + '\n';
    QDataStream streamsend(&text, QIODevice::WriteOnly);
    streamsend << message.toInt();
    socket->write(text);
}

#include "maindialog.h"
#include "ui_maindialog.h"
#include "server.h"
#include "client.h"
#include "settingsdialog.h"

#include <qbluetoothuuid.h>
#include <qbluetoothserver.h>
#include <qbluetoothdeviceinfo.h>
#include <qbluetoothlocaldevice.h>
#include <QtAndroidExtras>
#include <QTimer>
#include <qbluetoothaddress.h>
#include <QFile>
#include <QTextStream>
#include <QDateTime>

```

```

#include <QMessageBox>
#include <QDir>

static const QLatin1String serviceUuid("e8e10f95-1a70-4b27-9ccf-02010264e9c8");
int p;
int e;
int gt=0;
int coe;
int i=0;
int l=0;
int m=0;
int got0;
int j=0;
QFile ecr;
QString vitesse= "";
QString folder=QStandardPaths::writableLocation(QStandardPaths::DocumentsLocation);
float sect;
QString coef="";
QString diam="";
QString haut="";
QString typ="";
QString sansms;
QString sansus;
int sansm;
float sansu;
int moyfrr;
int sansmrr;
float moyf[10];
float moyd[10];
float moyed=0;
float moyef=0;

maindialog::maindialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::maindialog)
{
    ui->setupUi(this);
    showFullScreen();
    setStyleSheet("QDialog {border-image: url(:/rc/images/fond.jpg)}");

    QAndroidJniObject activity = QtAndroid::androidActivity();
    if (activity.isValid()) {
        QAndroidJniObject window = activity.callObjectMethod("getWindow", "()Landroid/view/Window;");
        if (window.isValid()) {
            const int FLAG_KEEP_SCREEN_ON = 128;
            window.callObjectMethod("addFlags", "(I)V", FLAG_KEEP_SCREEN_ON);
        }
        p=0;
        e=0;
        gt=0;
        ui->pauseButton->setEnabled(false);
        ui->lineEdit_fmax->setText("1000");
        ui->speed->setText("0");
        ui->down->setEnabled(false);
        ui->up->setEnabled(false);
        ui->stop->setEnabled(false);
        ui->goto0->setEnabled(false);
        scene = new QGraphicsScene;
        scene->setSceneRect(0,0,ui->graphicsView->size().width(),ui->graphicsView->viewport()-
>size().height());
        ui->graphicsView->setScene(scene);
        settingsdialog *settings=new settingsdialog(this);

        connect(settings,SIGNAL(sendset(QString,QString,QString,QString)),this,SLOT(recuset(QString,QString,QString,QStri
ng)));
    }

    maindialog::~maindialog()
    {
        settingsdialog *settings=new settingsdialog(this);

        disconnect(settings,SIGNAL(sendset(QString,QString,QString,QString)),this,SLOT(recuset(QString,QString,QString,QS
tring)));
        qDeleteAll(clients);
        delete serv;
    }

    void maindialog::recuset(const QString &Coef, const QString &Diam, const QString &Haut, const QString
&Typ)
    {
        coef=Coef;
        diam=Diam;
        haut=Haut;
        typ=Typ;
    }

    void maindialog::on_connectButton_clicked()
    {
        if(haut==" " || typ=="")
        {
            QMessageBox::warning(this,"paramètres","renseigner la hauteur de grains, son type et vérifier le
diamètre dans les paramètres");
        }
        else
        {
            coe= coef.toInt();

            ui->connectButton->setEnabled(false);
            ui->config->setEnabled(false);
            ui->ecrire->setEnabled(false);
            ui->lineEdit_fmax->setEnabled(false);
            ui->down->setEnabled(true);
            ui->up->setEnabled(true);
            ui->stop->setEnabled(true);
            ui->goto0->setEnabled(true);

            localAdapters = QBluetoothLocalDevice::allDevices();
            QBluetoothLocalDevice adapter(localAdapters.at(0).address());
            QBluetoothAddress adadap = adapter.address();

            serv = new server(this);

```

```

connect(serv, SIGNAL(messageReceived(QString)), this, SLOT(showMessage(QString)));
connect(this, SIGNAL(sendMessage(QString)), serv, SLOT(sendMessage(QString)));
serv->startServer(adadap);

localName = QBluetoothLocalDevice().name();

client *clien = new client(this);

connect(clien, SIGNAL(messageReceived(QString)), this, SLOT(showMessage(QString)));
connect(clien, SIGNAL(disconnected()), this, SLOT(clientDisconnected()));
connect(this, SIGNAL(sendMessage(QString)), clien, SLOT(sendMessage(QString)));

clien->startClient();

clients.append(clien);

ui->pauseButton->setEnabled(true);
}

}

void maindialog::clientDisconnected()
{
    client *clien = qobject_cast<client*>(sender());
    if (clien) {
        clients.removeOne(clien);
        clien->deleteLater();
    }
}

void maindialog::on_config_clicked()
{
    settingsdialog *settings=new settingsdialog(this);

connect(settings,SIGNAL(sendset(QString,QString,QString,QString)),this,SLOT(recuset(QString,QString,QString,QString)));
}

void maindialog::on_pauseButton_clicked()
{
    if(p==0)
    {
        emit sendMessage("128");
        ui->speed->setText("0");
        p=1;
        ui->pauseButton->setIcon(QIcon(":/rc/images/disconnectpp.png"));
        ui->ecrire->setEnabled(true);
        ui->lineEdit_fmax->setEnabled(true);
    }
    else if(p==1)
    {
        p=0;
        ui->pauseButton->setIcon(QIcon(":/rc/images/disconnect.png"));
        ui->ecrire->setEnabled(false);
        ui->lineEdit_fmax->setEnabled(false);
    }
}

void maindialog::on_up_clicked()
{
    QString up = ui->speed->text();
    int vup = up.toInt();
    if(vup==100)
    {
        vup=80;
    }
    qreal vupr = (vup+20)*1.27 +127;
    int valup = qRound(vupr);
    ui->speed->setText(QString::number(vup+20));
    emit sendMessage(QString::number(valup));
}

void maindialog::on_down_clicked()
{
    QString dn = ui->speed->text();
    int vdn = dn.toInt();
    if(vdn==80)
    {
        vdn=-80;
    }
    qreal vdown = (vdn-20)*1.27 +127;
    int valdn = qRound(vdown);
    ui->speed->setText(QString::number(vdn-20));
    emit sendMessage(QString::number(valdn));
}

void maindialog::on_stop_clicked()
{
    emit sendMessage("128");
    ui->speed->setText("0");
}

void maindialog::on_goto0_clicked()
{
    gt=1;
    emit sendMessage("0");
    ui->speed->setText("-100");
}

void maindialog::on_ecrire_clicked()
{
    if(e==0)
    {
        if(haut =="" || typ == "")
        {
            QMessageBox::warning(this,"paramètres","renseigner la hauteur de grains, son type et vérifier le diamètre dans les paramètres");
        }
        else
        {

```

```

sect= (diam.toFloat()/2000)*(diam.toFloat()/2000)*3.14159;
ui->ecrire->setIcon(QIcon(":/rc/images/ecrire.png"));
e=1;

QDateTime dt=QDateTime::currentDateTime();
QString dt2=dt.toString("dd.MM.yy hh.mm.ss");
QString dt3=folder+"/"+dt2+".txt";
ecr.setFileName(dt3);
ecr.open(QIODevice::ReadWrite | QIODevice::Text | QIODevice::Truncate);
QTextStream ecr(&ecr);
vitesse = ui->speed->text();
ecrid<<"deplacement(mm);force(kg);moydeplacement(mm);moyforce(kg);vitesse: "+vitesse+"%; diam:
"+diam+" mm; sect: "+QString::number(sect).replace(".",",")+" m2; delay: 10x"+coef+" ms; hauteur: "+haut+" mm;
type: "+typ<<endl;
}
}
else if(e==1)
{
    ui->ecrire->setIcon(QIcon(":/rc/images/ecrireb.png"));
    e=0;
    ecr.close();
}
}

void maindialog::showMessage(const QString &message)
{
    if(p==0)
    {
        if(sansm > ui->lineEdit_fmax->text().toInt())
        {
            if(ui->speed->text().toInt()!=0)
            {
                on_stop_clicked();
            }
        }

        if(gt==0)
        {
            if(got0 < 5)
            {
                if(ui->speed->text().toInt()!=0)
                {
                    on_stop_clicked();
                }
            }
        }

        if(gt==1)
        {
            if(got0 < 15)
            {
                on_stop_clicked();
                gt=0;
            }
        }

        QTextStream ecr(&ecr);
        i++;
        if(i%coe==0)
        {
            QStringList cher = message.split(";");
            for (int k=0; k<cher.size(); k++)
            {
                QString aff = cher.at(k);
                if(aff.startsWith("-"))
                {
                    sansms = aff.replace("-", "");
                    sansm= sansms.toInt()*2.3279 - 85.121;

                    qreal sansmr = sansm;
                    sansmrr = qRound(sansmr);

                    ui->plainTextEdit->insertPlainText(QString::number(sansmrr)+"\n");
                    ui->plainTextEdit->ensureCursorVisible();

                    moyf[l]= sansm;
                    l++;

                    if(l==10)
                    {
                        float sommf=0;
                        for(int o=0; o<10; o++)
                        {
                            sommf += moyf[o];
                        }
                        moyef=sommf/10;

                        qreal moyfr = moyef;
                        moyfrr = qRound(moyfr);

                        ui->lcd_read->display(QString::number(moyfrr));
                        l=0;
                    }
                }
                if(aff.startsWith("_"))
                {
                    sansus = aff.replace("_", "");
                    got0=sansus.toInt();
                    sansu= sansus.toFloat()*0.0098 - 0.1136;
                    ui->plainTextEdit_2->insertPlainText(QString::number(sansu, 'g', 3)+"\n");
                    ui->plainTextEdit_2->ensureCursorVisible();

                    moyd[m]= sansu;
                    m++;

                    if(m==10)
                    {
                        float sommed=0;
                        for(int oo=0; oo<10; oo++)
                        {
                            sommed += moyd[oo];
                        }
                    }
                }
            }
        }
    }
}

```


Annexe 8 : Led de puissance

Xeon 3 Power Pure Green LED

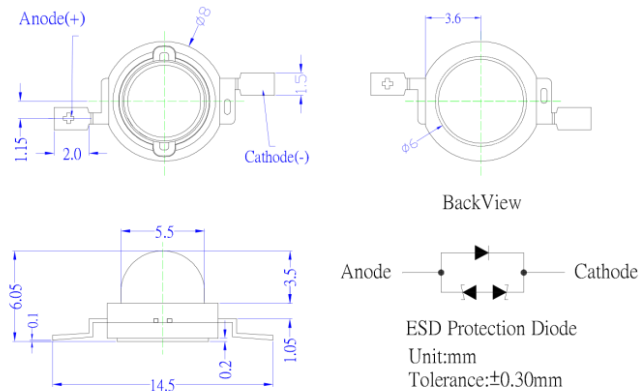
OSG5XME3C1E

VER C.0

■Features

- Highest Luminous Flux
- Super Energy Efficiency
- Long Lifetime Operation
- Superior ESD protection
- Superior UV Resistance

■Outline Dimension



■Applications

- Read lights (car, bus, aircraft)
- Portable (flashlight, bicycle)
- Bollards / Security / Garden
- Traffic signaling / Beacons
- In door / Out door Commercial lights
- Automotive Ext

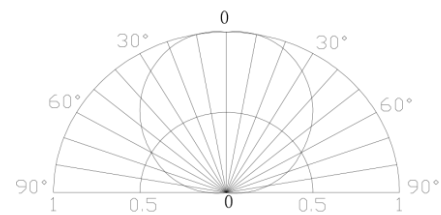
■Absolute Maximum Rating

(Ta=25°C)

Item	Symbol	Value	Unit
DC Forward Current	I_F	800	mA
Pulse Forward Current*	I_{FP}	1000	mA
Reverse Voltage	V_R	5	V
Power Dissipation	P_D	3200	mW
Operating Temperature	T_{opr}	-30 ~ +85	°C
Storage Temperature	T_{stg}	-40 ~ +100	°C
Lead Soldering Temperature	T_{sol}	260°C/5sec	-

*Pulse width Max.10ms Duty ratio max 1/10

■Directivity



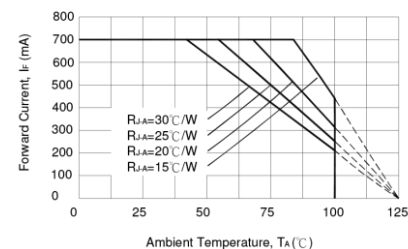
■Electrical -Optical Characteristics

(Ta=25°C)

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
DC Forward Voltage	V_F	$I_F=350mA$	3.0	3.3	4.0	V
		$I_F=700mA$	3.5	3.8	4.5	V
DC Reverse Current	I_R	$V_R=5V$	-	-	10	μA
Domi. Wavelength	λ_D	$I_F=700mA$	520	525	530	nm
Luminous Flux	Φ_v	$I_F=700mA$	120	130	-	lm
50% Power Angle	$2\theta_{1/2}$	$I_F=700mA$	-	120	-	deg

Note: Don't drive at rated current more than 5s without heat sink for Xeon 3 emitter series.

■Forward Operating Current (DC)



LED & Application Technologies



VER C.0

Annexe 9 : Régulateur LM317



LM317

SLVS044W – SEPTEMBER 1997 – REVISED OCTOBER 2014

LM317 3-Terminal Adjustable Regulator

1 Features

- Output Voltage Range Adjustable
From 1.25 V to 37 V
- Output Current Greater Than 1.5 A
- Internal Short-Circuit Current Limiting
- Thermal Overload Protection
- Output Safe-Area Compensation

2 Applications

- ATCA Solutions
- DLP: 3D Biometrics, Hyperspectral Imaging, Optical Networking, and Spectroscopy
- DVR and DVS
- Desktop PC
- Digital Signage and Still Camera
- ECG Electrocardiogram
- EV HEV Charger: Level 1, 2, and 3
- Electronic Shelf Label
- Energy Harvesting
- Ethernet Switch
- Femto Base Station
- Fingerprint and Iris Biometrics
- HVAC: Heating, Ventilating, and Air Conditioning
- High-Speed Data Acquisition and Generation
- Hydraulic Valve
- IP Phone: Wired and Wireless
- Infusion Pump
- Intelligent Occupancy Sensing
- Motor Control: Brushed DC, Brushless DC, Low-Voltage, Permanent Magnet, and Stepper Motor
- Point-to-Point Microwave Backhaul
- Power Bank Solutions
- Power Line Communication Modem
- Power Over Ethernet (PoE)
- Power Quality Meter
- Power Substation Control
- Private Branch Exchange (PBX)
- Programmable Logic Controller
- RFID Reader
- Refrigerator
- Signal or Waveform Generator
- Software Defined Radio (SDR)
- Washing Machine: High-End and Low-End
- X-ray: Baggage Scanner, Medical, and Dental

3 Description

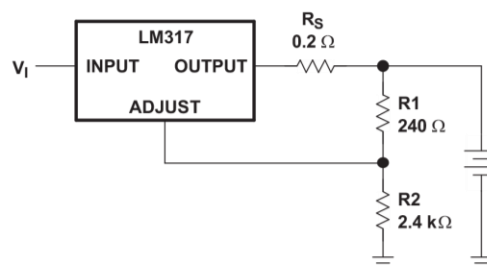
The LM317 device is an adjustable three-terminal positive-voltage regulator capable of supplying more than 1.5 A over an output-voltage range of 1.25 V to 37 V. It requires only two external resistors to set the output voltage. The device features a typical line regulation of 0.01% and typical load regulation of 0.1%. It includes current limiting, thermal overload protection, and safe operating area protection. Overload protection remains functional even if the ADJUST terminal is disconnected.

Device Information⁽¹⁾

PART NUMBER	PACKAGE (PIN)	BODY SIZE (NOM)
LM317	SOT (4)	6.50 mm × 3.50 mm
	TO-220 (3)	10.16 mm × 8.70 mm
	TO-220 (3)	10.16 mm × 8.59 mm
	TO-263 (3)	10.18 mm × 8.41 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

4 Battery-Charger Circuit



An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.

Annexe 10 : Quelques rappels complémentaires en photo-élasticimétrie

L'étude exposée ici montre comment observer l'évolution de l'état de contrainte d'un moule contenant une poudre (ou un milieu granulaire) en train d'être comprimée, via une analyse photoélastique permettant de remonter aux conditions aux limites à l'interface moule-poudre.

Notre moule est supposé être peu déformable, et force donc le milieu granulaire à évoluer sous l'effet du tassement dans un volume spécifique presque oedométrique (cellule cylindrique à rayon constant). Néanmoins, il est fait d'un matériau transparent classique utilisé dans de bonnes conditions (c'est à dire qu'il est supposé ne subir que de petites déformations élastiques). La compression de la poudre va appuyer sur ses parois, qui vont répondre de façon élastique. De ce fait son état de contrainte va évoluer. L'écart entre la pression interne sur sa paroi et la pression extérieure va augmenter rapidement. Les contraintes à l'intérieur du moule vont devenir inhomogènes.

La compression des poudres va faire croître la pression appliquée sur le moule. De façon réciproque cette pression de surface va le déformer. On espère pouvoir remonter à la variation de la pression locale ainsi que sur la paroi, par une analyse photoélastique des contraintes.

La photoélasticimétrie fait intervenir différents concepts de physique, tant mécanique qu'optique. Il va nous falloir donc revenir sur des notions essentielles, tel que la polarisation de la lumière et la biréfringence afin de lier les phénomènes à la mécanique pour aboutir à la base de la photoélasticité.

Tout d'abord, rappelons qu'en physique, la lumière peut être traitée de façon duale comme une onde ou comme un ensemble de particules (photons). Dans le premier cas, il s'agit d'une onde électromagnétique vectorielle transverse, à deux composantes perpendiculaires à la direction de propagation. Elle est composée d'un champ électrique noté \vec{E} et d'un champ magnétique noté \vec{B} (perpendiculaire à \vec{E}). Ces deux champs sont contenus dans le plan d'onde perpendiculaire à la direction de propagation de l'onde. Les variations de \vec{E} et \vec{B} sont liées par les lois de l'électromagnétisme (équations de Maxwell). Il n'est donc pas nécessaire d'étudier les deux champs à la fois, car l'un est déductible de l'autre. Le choix dépendra des conditions de l'expérience. Dans notre cas, le moule est un matériau photoélastique dont l'indice de réfraction n dépend de l'état de contrainte. Nous omettrons donc de décrire ce qui se passe pour le champ magnétique et nous ne parlerons que du champ électrique \vec{E} .

Si on considère une direction de propagation de la lumière (disons l'axe x), on constate que le champ électrique \vec{E} peut être orienté de 2 manières différentes (et perpendiculaires l'un à l'autre) suivant l'axe y ou l'axe z . Ce champ \vec{E} forme un espace vectoriel à deux dimensions (une fois la direction de propagation donnée).

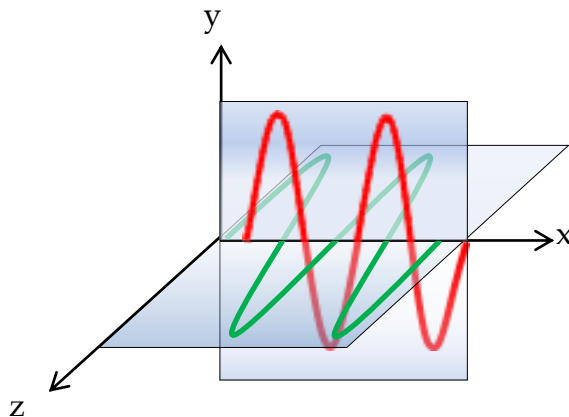


Figure A10.1 : Décomposition du vecteur \vec{E} de l'onde lumineuse se propageant selon l'axe x .

A10.1. Polarisation

L'onde du champ électrique peut être rapportée sur les deux plans perpendiculaires au plan d'onde. Avec une propagation le long de l'axe x , la figure A10.1 illustre cette décomposition.

Son équation de champ sera alors :

$$\begin{aligned} \text{A10.1} \quad E_y &= E_{0,y} \sin(kx - \omega t + \varphi_y) \\ E_z &= E_{0,z} \sin(kx - \omega t + \varphi_z) \end{aligned}$$

Avec : $\omega = 2\pi\nu$: fréquence angulaire
 $k = 2\pi/\lambda$: nombre d'onde
 x : direction de propagation de la lumière
 E_0 : amplitude du champ électrique
 $E_{0,y}$, $E_{0,z}$: projection de E_0 sur les axes y et z
 φ_y , φ_z : phase du champ électrique selon y et z

Par conséquent, toute solution du champ \vec{E} sous la forme écrite plus haut est une solution possible des équations de Maxwell. λ et ν sont respectivement la longueur d'onde et la fréquence de l'onde lumineuse dans le vide (indice $n = 1$). Elles sont liées par la relation $\lambda = c/\nu$ avec c , vitesse de la lumière dans le vide. Lors des essais photoélastiques, l'indice variera ; cette différence de permittivité du milieu fera varier la vitesse de phase de l'onde.

L'état de polarisation de la lumière est défini par le déphasage entre $E_{0,y}$ et $E_{0,z}$. Lors du passage dans un dioptre, cet état peut prendre plusieurs formes :

- La plus commune est la polarisation elliptique, le déphasage φ prend une valeur quelconque.
- La polarisation linéaire, φ vaut 0 ou un multiple de π .
- La polarisation circulaire, φ vaut $\pi/2 + 2k\pi$ (k entier).

Notons qu'il y a deux formes de polarisation circulaire, la droite et la gauche, selon la direction de rotation du champ.

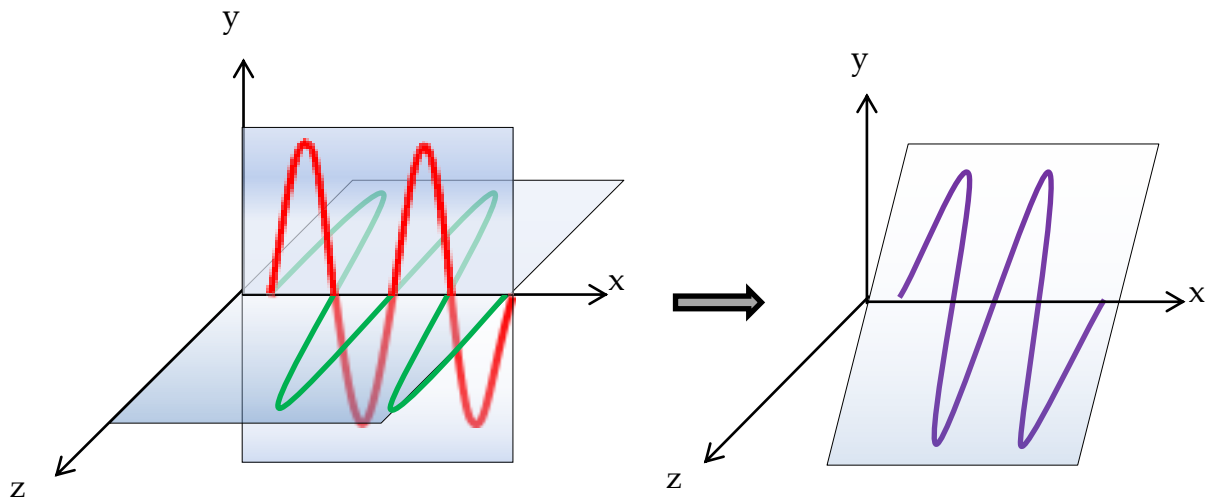


Figure A10.2 : Polarisation linéaire à 45° par rapport à l'axe y.

Aussi, la somme de deux polarisations circulaires de même intensité, une gauche et l'autre droite, donnera une polarisation linéaire (ce champ \vec{E} appartient bien à un espace vectoriel à deux dimensions).

D'un point de vue pratique, on choisira comme vecteur de base, le type de polarisation qui se prêtera au mieux aux besoins de l'expérience, qu'il soit circulaire (gauche et droite) ou linéaire (deux axes orthogonaux), sachant que l'on peut repasser d'une représentation à l'autre à tout instant.

Pour notre étude, on commencera par utiliser une polarisation linéaire, représentée en figure A10.2, pour visualiser les contraintes. Dans un second temps, on passera en polarisation circulaire pour isoler certains phénomènes optiques.

Rappelons qu'en franchissant un dioptre (une surface matériel), des conditions de raccordement du champ électrique existent. De ce fait, une partie du faisceau est réfléchi et l'autre est transmise. S'il est renvoyé, on parlera de réflexion, avec un angle de réflexion égal à l'angle d'incidence. S'il pénètre, on

parlera d'absorption et de coefficient d'absorption. En revanche, s'il traverse, on parlera alors de transmission et d'indice de réfraction.

Il est possible d'obtenir une polarisation linéaire par réflexion vitreuse, elle sera plus ou moins importante en fonction de l'angle d'incidence.

Prenons un rayon incident avec une polarisation elliptique. C'est la réflexion de la composante perpendiculaire au plan d'incidence qui sera privilégiée par rapport à la composante contenue dans le plan d'incidence qui elle, sera transmise. La figure A10.3 illustre ce phénomène.

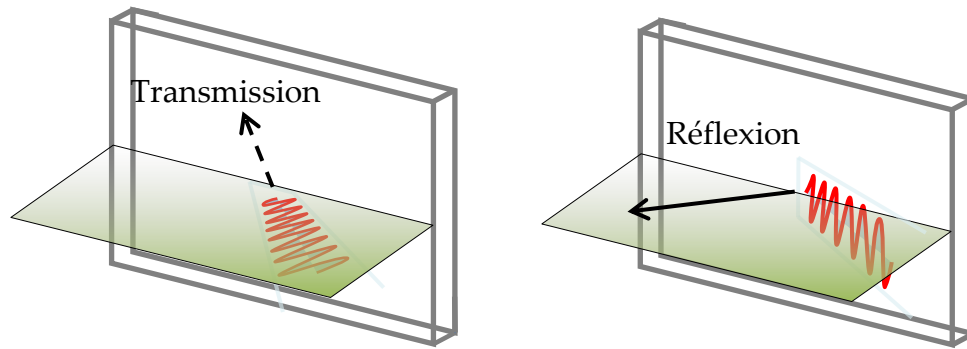


Figure A10.3: Séparation des composantes.

Dans le cas particulier où l'angle entre le rayon réfléchi et le rayon réfracté est égal à 90° , comme le montre la figure A10.4, le rayon réfléchi sera totalement polarisé perpendiculairement au plan d'incidence et le rayon réfracté sera totalement polarisé parallèlement au plan d'incidence.

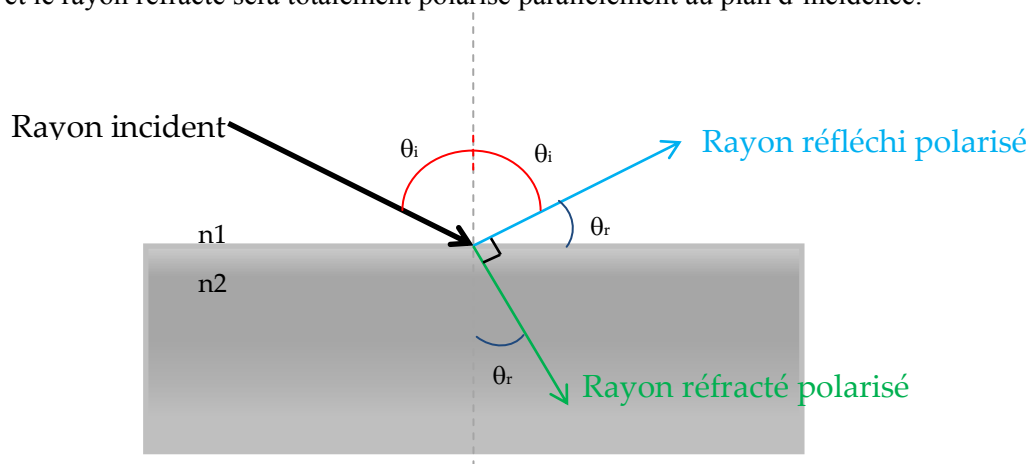


Figure A10.4 : Angle de 90° entre rayon réfléchi et réfracté.

Calculons cet angle d'incidence particulier θ_i :

Nous savons que l'angle d'incidence θ_i est égal à l'angle de réflexion. θ_r est l'angle de réfraction. n_1 et n_2 sont les indices des milieux traversés.

D'après la loi de Snell-Descartes,

$$(A10.2) \quad n_1 \cdot \sin(\theta_i) = n_2 \cdot \sin(\theta_r)$$

$$\begin{aligned} \text{avec} \quad n_1 \cdot \sin(\theta_i) &= n_2 \cdot \sin(90 - \theta_i) \\ n_1 \cdot \sin(\theta_i) &= n_2 \cdot \cos(\theta_i) \\ \theta_i &= \arctan(n_2/n_1) \end{aligned}$$

Dans le cas de l'interaction air/verre, ayant pour indices respectifs 1 et 1,5, θ_i est égal à 56° . Cet angle d'incidence particulier est appelé angle de Brewster.

Il est aussi possible d'utiliser des filtres polarisants pour ne sélectionner qu'une composante de l'onde en fonction de l'axe du polariseur. Les filtres dichroïques, qui ont une absorption de la lumière différente selon sa polarisation, existent sous forme naturelle comme la tourmaline par exemple. De nos jours on utilise des filtres artificiels comme le polaroïd, il est composé de longues molécules étirées qui forment une grille ne laissant passer que les ondes dont le champ électrique est perpendiculaire à l'orientation de la grille. Voici, ci-dessous, une représentation en Fig. A10.5.

Afin de déterminer le pourcentage de lumière filtrée en fonction de l'angle d'entrée, il nous suffit de calculer l'intensité I en sortie de filtre. Pour cela, la loi de Malus est utilisée :

$$(A10.3) \quad I = I_0 \cdot \cos^2(\theta)$$

Avec I_0 , intensité à l'entrée du polariseur et θ , angle d'entrée de la source.

Dans le cas d'une source non polarisée, il faut appliquer le théorème de la moyenne à la loi de Malus car θ prend plusieurs valeurs :

$$(A10.4) \quad I = I_0/2 \quad \text{car} \quad \frac{\int_0^{\pi/2} \cos^2(\theta) d\theta}{\frac{\pi}{2} - 0} = \frac{1}{2}$$

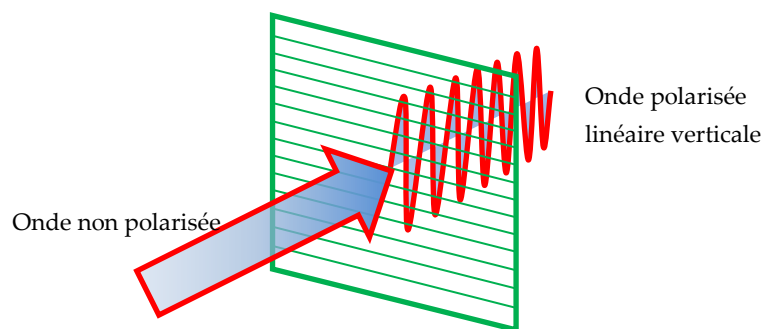


Figure A10.5 : Polariseur linéaire.

Pour le cas d'une source polarisée linéairement et avec un angle de $\pi/2$ par rapport à l'axe du polariseur, l'intensité de sortie sera donc nulle. C'est le cas pour deux polariseurs croisés.

A10.3. Biréfringence

Certains cristaux sont dits anisotropes uniaxes (un seul axe « optique » différent des deux autres, qui ont eux les mêmes caractéristiques), d'autres sont encore plus anisotropes et sont dits biaxes (c'est le cas des tricliniques, avec un degré de symétrie très faible). Dans ce dernier cas, chaque direction du champ \vec{E} a des vitesses variant en fonction de la direction de propagation (elle n'a lieu que dans une direction appartenant au plan perpendiculaire au champ \vec{E} considéré). Mais pour un uniaxe, comme la calcite, le tenseur de permittivité a une symétrie axiale de telle sorte que le matériau a la particularité de posséder deux indices de réfraction dépendants de la direction de polarisation lumineuse, l'un pour une direction privilégiée, et l'autre commune aux deux directions perpendiculaires.

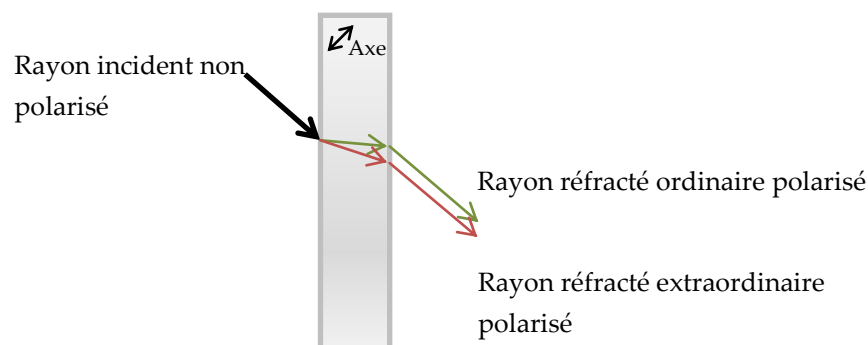


Figure A10.6 : Phénomène de biréfringence.

Comme le montre la figure A10.6, lorsque le rayon incident entre dans le cristal, il se divise en deux rayons. L'un est appelé « rayon ordinaire O » qui suit les lois normales de réfraction optique et l'autre « rayon extraordinaire E » qui « échappe à la loi du précédent » et diverge du précédent.

Ces deux rayons sont polarisés linéairement et sont perpendiculaires entre eux. Le rayon extraordinaire sera polarisé dans le plan de l'axe optique. Leurs vitesses respectives seront différentes, ce qui induit des indices différents, appelés n_o et n_e . L'indice du rayon O sera fixe alors que celui du rayon E variera en fonction de la direction de propagation. La différence entre les indices est appelée biréfringence et peut être négative ou positive.

Afin de calculer la trajectoire du rayon E en fonction de la direction du rayon incident, il faut utiliser la construction de Huygens (cf. figure A10.7) pour déterminer la surface d'onde de l'ellipsoïde des indices.

Elle a pour équation :

$$(A10.5) \quad \frac{x^2}{n_x^2} + \frac{y^2}{n_y^2} + \frac{z^2}{n_z^2} = 1$$

Où x , y et z sont les coordonnées des points appartenant à l'ellipsoïde des indices et n_x , n_y et n_z sont leurs indices respectifs.

Dans notre cas d'un milieu uniaxe, nous pouvons écrire :

$$(A10.6) \quad \frac{x^2}{n_o^2} + \frac{y^2}{n_e^2} = 1$$

Avec nos indices ordinaire et extraordinaire.

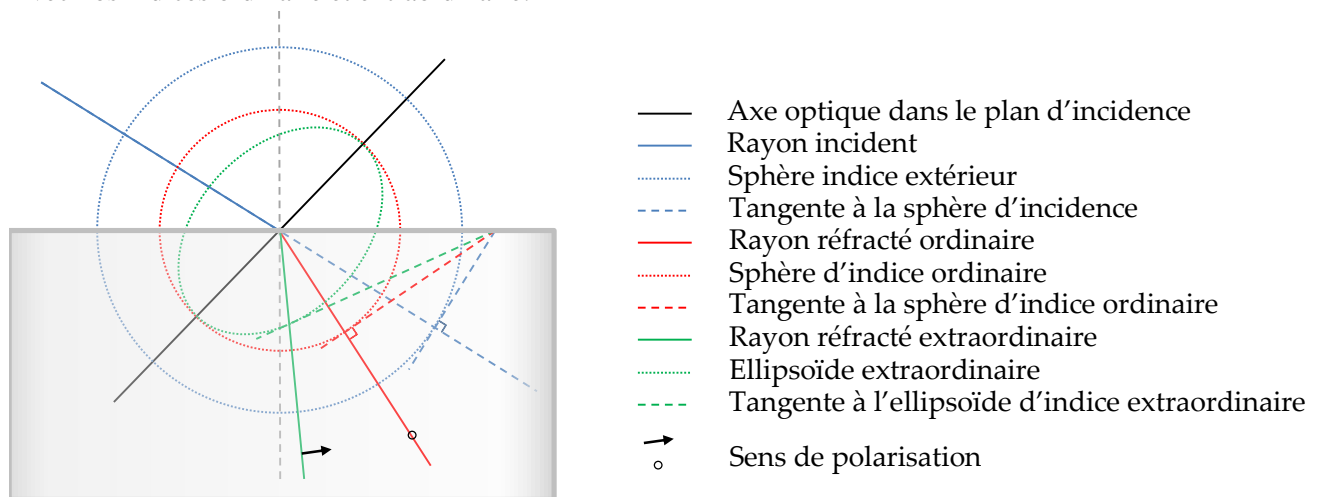


Figure A10.7 : Construction de Huygens.

L'indice ordinaire donne une sphère de rayon $1/n_o$ et l'indice extraordinaire donne un ellipsoïde de révolution dont l'axe vaut $1/n_o$ et la section $1/n_e$.

Si $n_e > n_o$, alors l'ellipsoïde sera contenu dans la sphère.

Si $n_e < n_o$, alors la sphère sera contenue dans l'ellipsoïde.

Ils seront toujours tangents sur l'axe optique.

A10.4. Mécanique

Intéressons-nous maintenant aux phénomènes mécaniques. Prenons un matériau transparent, homogène et isotrope. En lui soumettant des sollicitations mécaniques extérieures, on va créer un état de contrainte tridimensionnel en chaque point, caractérisé par un tenseur de contraintes σ [13]. En se plaçant dans un repère $(o, 1, 2, 3)$, nous pouvons définir les trois directions principales des contraintes, σ_1 , σ_2 , σ_3 . Dans le cadre de l'expérience, les contraintes observées seront selon deux axes ; les contraintes selon le troisième axe seront intégrées dans le plan des deux autres axes.

Le tenseur devient alors comme suit :

$$(A10.7) \quad \begin{vmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

Les intensités et les directions des contraintes varient de manière continue d'un point à un autre du matériau (hypothèse de base de la mécanique des milieux continus). On peut ainsi tracer une courbe dont la tangente en chaque point est dans la direction de la contrainte principale. Comme les contraintes principales sont perpendiculaires entre elles, on peut aussi tracer une seconde courbe perpendiculaire en tout point de la première. Nous avons donc une infinité de courbes perpendiculaires entre elles en chaque point. Elles sont appelées lignes isostatiques, elles donnent la répartition globale des contraintes principales dans le matériau comme le montre la figure A10.8.

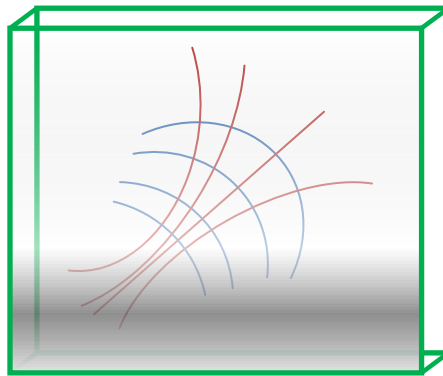


Figure A10.8 : Répartition des isostatiques.

Voyons maintenant quelles informations nous donnent ces isostatiques :

- Les axes de symétrie et les bords libres du matériau sont des isostatiques.
- Si une isostatique a une forte courbure, alors la contrainte principale qui lui est perpendiculaire subit une forte variation.
- Lorsqu'elles sont rectilignes, elles sont appelées « isoclines » (elles seront noires si parallèles aux axes des polariseurs de l'expérience de photoélasticité).
- Les équations d'équilibre mécanique de Lamé et Maxwell permettent de les étudier et s'écrivent ainsi :

$$\frac{\partial \sigma_1}{\partial S_1} + \frac{\sigma_1 - \sigma_2}{\rho_2} = 0$$

$$\frac{\partial \sigma_2}{\partial S_2} + \frac{\sigma_1 - \sigma_2}{\rho_1} = 0$$

Avec S_1 et S_2 , les coordonnées curvilignes des lignes isostatiques et ρ_1 et ρ_2 , leurs rayons de courbure.

Par le biais d'une expérience de photoélasticité, on peut déduire la valeur de σ_1 et de σ_2 et tracer les isostatiques.

A10.5. Photoélasticité

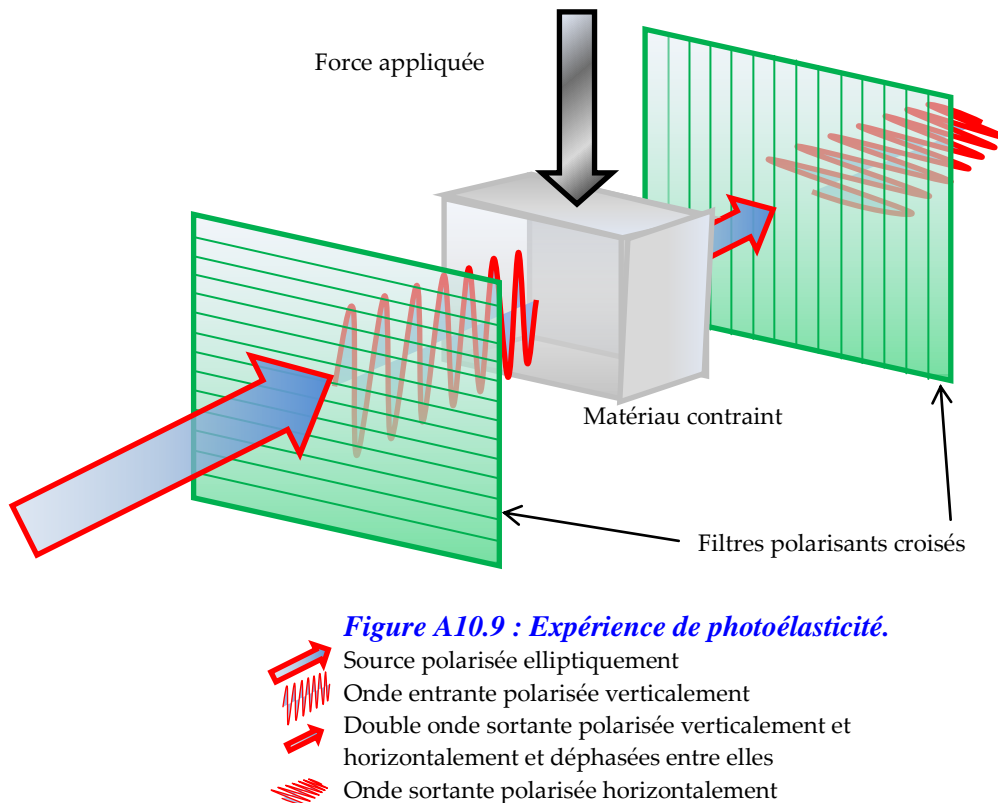
La photoélasticité utilise le phénomène de biréfringence accidentelle. Un corps transparent isotrope devient biréfringent sous la contrainte. Les rayons réfractés subissent les variations d'indice du milieu provoquées par la sollicitation extérieure. Les modifications d'indice présentent alors les mêmes symétries que le

champ des contraintes. En définissant 1 et 2 comme étant les deux directions principales des contraintes, et pour un rayon lumineux traversant le plan 1-2, les deux lignes neutres sont alors ces deux directions 1 et 2. L'indice devient n_1 selon l'axe 1 et n_2 selon l'axe 2.

Maxwell a déterminé les lois liant les indices principaux aux contraintes principales. Dans le cas de 2 dimensions :

$$(A10.8) \quad n_2 - n_1 = C(\sigma_2 - \sigma_1)$$

Avec n_2, n_1 les indices le long des axes de contrainte, C étant la constante photoélastique du matériau utilisé en brewster (10^{-12} Pa^{-1}) et σ_2, σ_1 sont les contraintes principales. Dans le cas du plexiglas, la valeur typique de C est de 5 brewsters.



Un rayon lumineux de longueur d'onde λ se verra décomposé en deux composantes vibrant chacune selon les directions 1 et 2 avec une vitesse respective $v_1 = c/n_1$ et $v_2 = c/n_2$ où c est la vitesse de la lumière dans le vide. Après un parcours e , ces composantes présentent donc un déphasage φ :

$$(A10.9) \quad \varphi = 2\pi e (n_2 - n_1)/\lambda$$

λ étant la longueur d'onde de la source utilisée.

Prenons une expérience de photoélasticité complète, comme illustré en figure A10.9.

Un rayon de longueur d'onde λ traverse un polariseur linéaire et se retrouve donc polarisé verticalement. Il traverse ensuite le milieu contraint et parcourt un chemin optique différent suivant les deux polarisations du champ électrique. A la sortie, il en résulte un déphasage optique entre les deux polarisations. La somme des deux composantes forme une polarisation généralement elliptique dont on analyse l'excentricité par un deuxième polariseur (analyseur) en position croisée avec le premier. Cet analyseur, mis à la sortie, ne laisse donc passer que la composante polarisée horizontalement.

Il faut aussi prendre en compte l'angle α formé entre les lignes de contraintes principales du milieu et la direction des polariseurs. L'intensité lumineuse I en sortie est donnée par :

$$(A10.10) \quad I = I_0 \sin^2(2\alpha) \sin^2(\varphi/2)$$

Pour avoir extinction en sortie, il y a deux possibilités :

- Tout d'abord il faut que :

$$(A10.11) \quad \sin^2(2\alpha) = 0$$

Avec $\alpha = 0$ ou un multiple de $\pi/2$

On comprend ainsi que si les lignes de contraintes principales se retrouvent parallèles aux axes des polariseurs, il y a donc extinction de la lumière. Si on tourne simultanément les deux polariseurs, on verra ces lignes se déplacer. En effet, les contraintes principales changent d'orientation graduellement dans le milieu. Ces lignes sont appelées « isoclines » ; en repérant leurs positions selon différents angles, nous pourrions tracer les isostatiques.

- La seconde possibilité pour avoir extinction est que :

$$(A10.12) \quad \sin^2(\varphi/2) = 0$$

Avec $\varphi = 2\pi N$ (N entier).

En combinant les équations (A10.11) et (A10.12), on peut aussi écrire :

$$(A10.13) \quad \sigma_2 - \sigma_1 = \lambda N / eC = fN/e$$

Avec f, coefficient de frange du matériau pour une longueur d'onde donnée de la source lumineuse (en kPa.m/frange).

N est aussi appelé ordre de frange.

Notons que si $\sigma_2 - \sigma_1 = 0$, alors $\varphi = 0$, il y aura donc extinction.

Les contraintes principales variant progressivement, nous verrons donc une série de franges noires appelées isochromatiques. Ceci est valable pour une source monochromatique. Dans le cas d'une source blanche, nous verrons des franges en dégradé de couleur, le retard optique variant avec la longueur d'onde (d'où la dénomination « isochromatique »).

Afin de mieux les observer, il est préférable de supprimer les isoclines. Pour cela il faut intercaler deux lames quart d'onde, une après le polariseur et l'autre avant l'analyseur.

Ces lames sont taillées dans un matériau biréfringent et ont la particularité d'induire un déphasage de $\pi/2$. En reprenant l'équation (A10.), cela nous donne :

$$(A10.14) \quad (n_2 - n_1)e = \lambda/4$$

D'où la dénomination « lame quart d'onde ».

Comme le montre la figure A10.10, il faut positionner l'axe principal de la première lame à 45° de l'axe du polariseur. Puis, placer la deuxième lame à 90° de la première, elle sera donc à 45° de l'analyseur [16]. Cela a pour conséquence de transformer la polarisation linéaire en circulaire et inversement. En polarisation circulaire il n'y a plus de directions optiques privilégiées, ce qui empêche la formation d'isoclines.

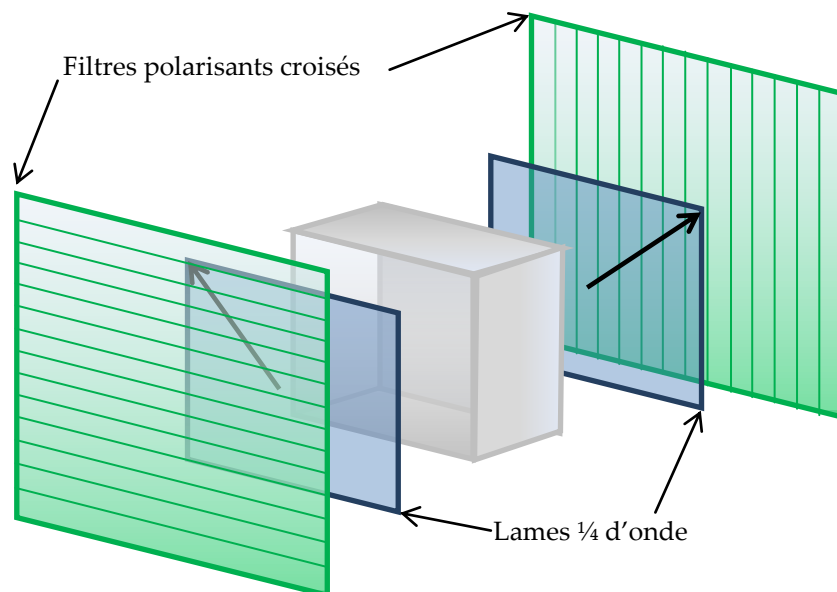


Figure A10.10 : Position des lames quart d'onde.