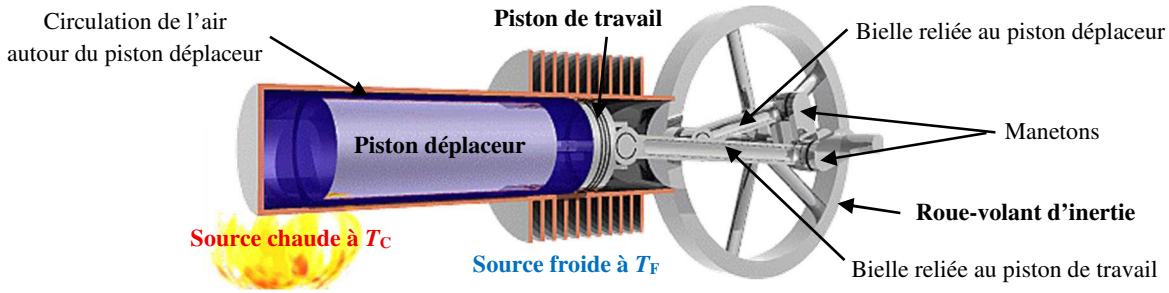
Moteur de Stirling – Tracé d'un cycle (P, V)

Principe du moteur Stirling (Robert et James Stirling 1816)

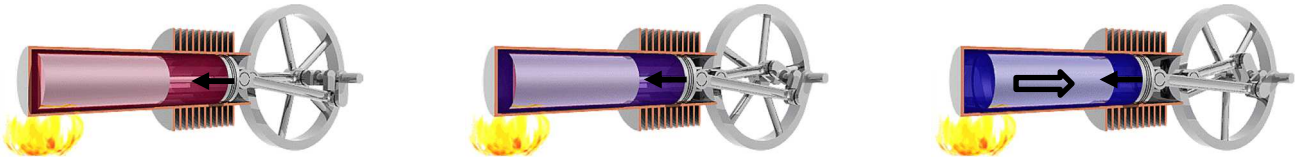
Le moteur Stirling est un moteur à ***combustion externe*** (toute source de chaleur peut être utilisée : énergie solaire, énergie géothermique, énergie nucléaire…) et à fluide de travail en ***cycle fermé*** (l'air interne au moteur ne sort jamais de celui-ci).
Les moteurs thermiques usuels sont à *combustion interne* en *cycle ouvert*.
La spécificité de ce moteur réside dans un ***régénérateur*** (échangeur thermique interne) qui améliore son efficacité.

Moteur Stirling de type bêta

Le moteur comporte deux pistons, un ***piston de travail ou piston moteur*** et un ***piston « déplaceur »*** reliés à la ***roue d'inertie*** via deux bielles fixées à la roue grâce à deux manetons décentrés par rapport à l'axe de rotation de la roue (cf. schéma ci-dessous).
Cet embiellage crée pour le piston moteur un retard de phase de $\pi/2$ (un quart de tour) par rapport au piston déplaceur.
Le piston déplaceur (de plus petit diamètre que le cylindre) possède un double rôle :
- faire passer alternativement le gaz de la source chaude à la source froide (ailettes de refroidissement dans l'air ambiant) ;
- contribuer à réchauffer ou refroidir l'air au cours de ses transformations.

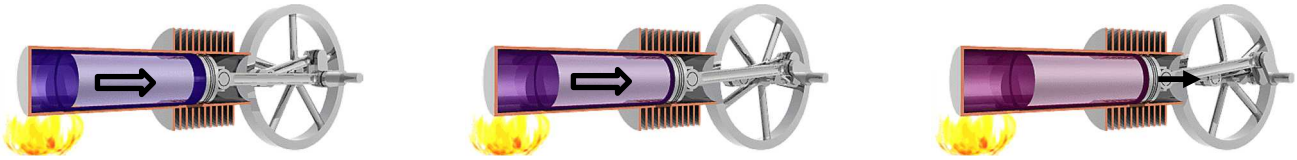


Bien distinguer l'air du côté piston moteur-source froide de l'air du côté source chaude pour analyser le fonctionnement.



Piston déplaceur en position basse

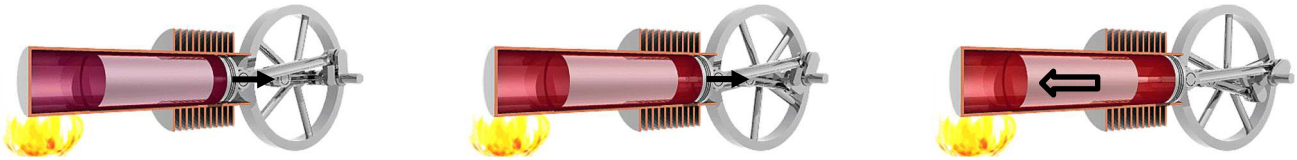
La totalité de l'air, encore chaud, est au contact de la source froide : sa température baisse ainsi que sa pression. Puis il commence à être refoulé vers la source chaude par les deux pistons qui se rapprochent tandis que le piston déplaceur se refroidit au contact de la source froide.
Le volume entre les deux pistons diminue : la compression du volume « moteur » s'effectue au contact du déplaceur refroidi, la ***compression*** est ***isotherme*** ($V\downarrow$, $P\uparrow$ à T = constante).



Point mort bas

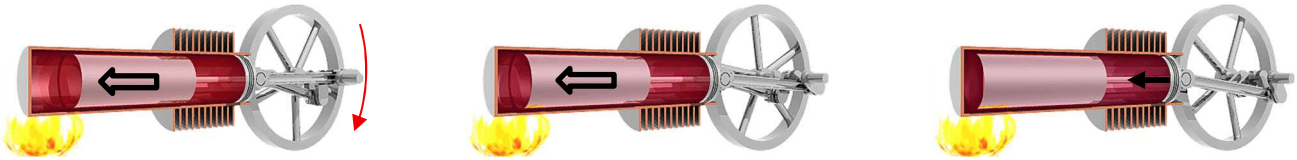
Piston de travail en position basse

Au voisinage du point mort bas, le volume « moteur » varie peu autour de son minimum : la ***compression*** est ***isochore***. L'air est progressivement transféré du côté de la source chaude par le piston déplaceur qui se rapproche de sa position la plus haute.



Piston déplaceur en position haute

L'air réchauffé dont la pression a augmenté commence à repousser le piston moteur tandis que le déplaceur contribue à le transférer vers le piston moteur réchauffant au passage le piston déplaceur. Les pistons s'éloignent, le piston moteur est refoulé par l'air chaud dont la pression va diminuer. Le piston déplaceur est alors au contact de la source chaude : la phase motrice est une ***détente isotherme*** ($V\uparrow$, $P\downarrow$ à T = constante).



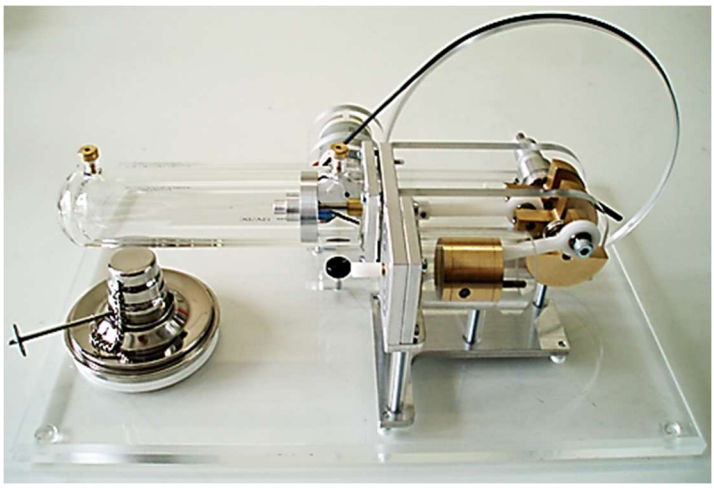
Point mort haut (PMH)

Piston de travail en position haute

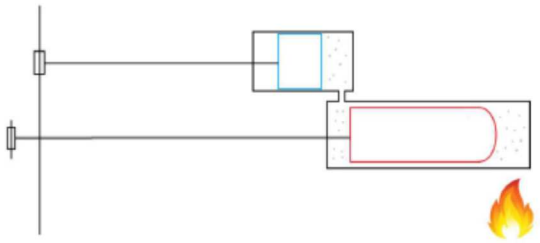
Au voisinage du point mort haut, le volume « moteur » varie peu autour de son maximum et l'air du côté « moteur » est au contact de la source froide, il se refroidit et sa pression diminue : la détente est isochore. Le piston déplaceur transfère progressivement l'air vers la source froide.

Moteur Stirling de type gamma

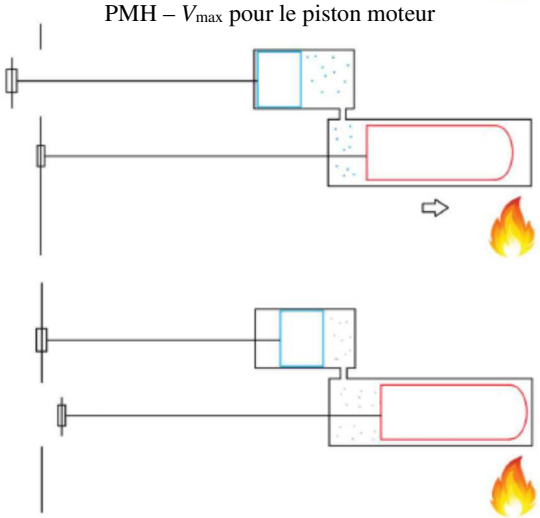
Dans un moteur de type gamma, le fonctionnement est plus facile à appréhender car les deux pistons évoluent dans deux cylindres séparés mais en communication l'un avec l'autre (schéma ci-contre et photo ci-dessous).



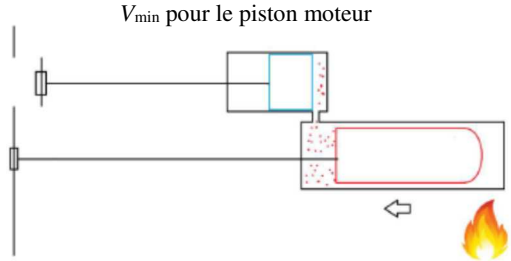
Dans ce moteur, le régénérateur est la plaque en métal entre les deux cylindres.



PMH – V_{max} pour le piston moteur



V_{min} pour le piston moteur



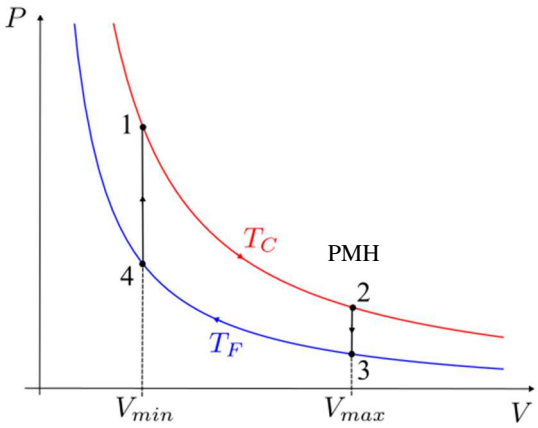
Modélisation du cycle de Stirling

On note a le taux de compression défini par : $a = \dfrac{V_{max}}{V_{min}}$.

On suppose que l'air est un gaz parfait.

En l'absence de régénérateur, le rendement de ce moteur est :

$$\eta_{sr} = 1 - \frac{T_F \ln a + \dfrac{T_C - T_F}{\gamma - 1}}{T_C \ln a + \dfrac{T_C - T_F}{\gamma - 1}}.$$



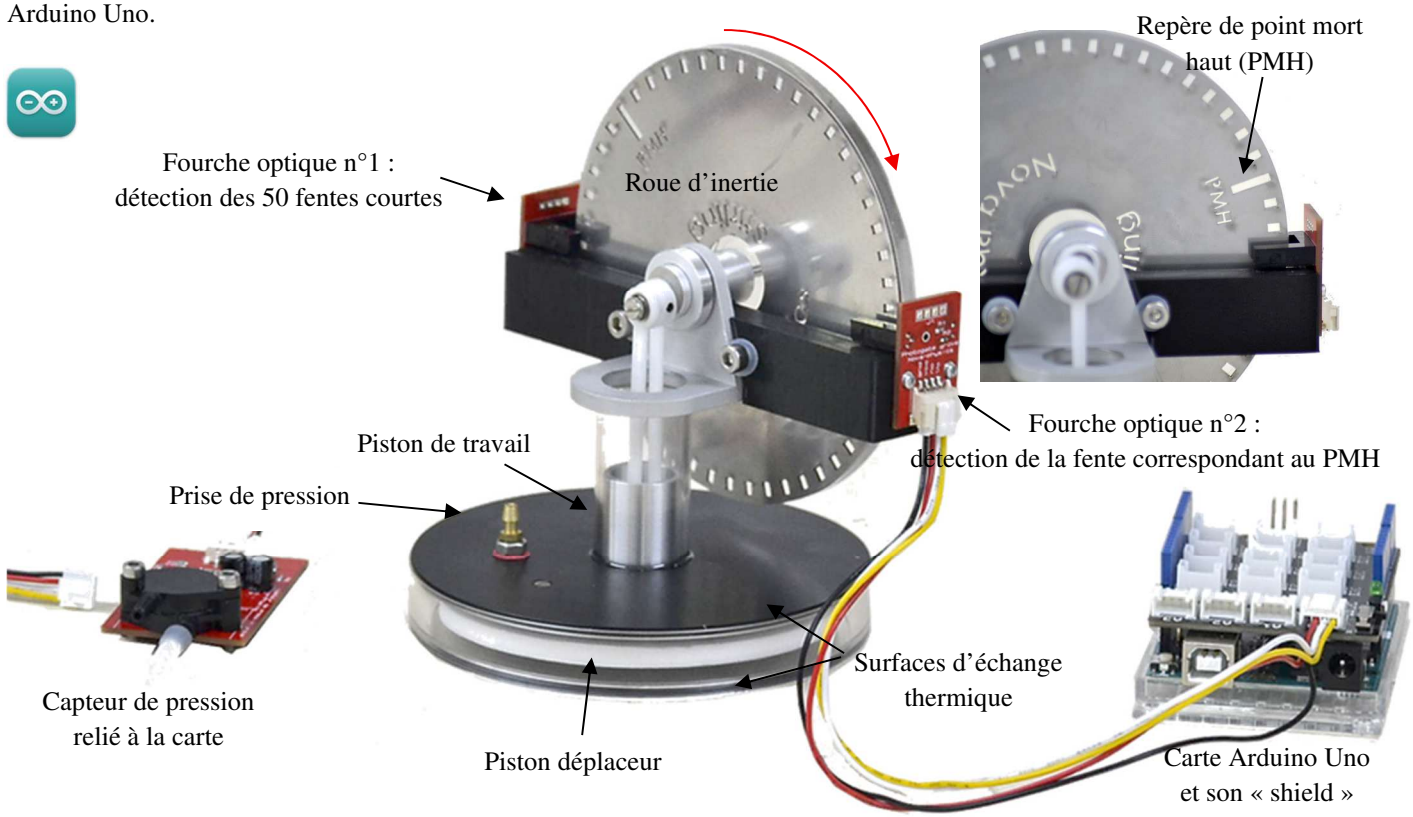
En présence d'un régénérateur parfait, on suppose que celui-ci récupère entièrement l'énergie nécessaire au réchauffage isochore (de 4 à 1) au cours du refroidissement isochore (de 2 à 3) : le piston déplaceur est effectivement au voisinage de la source chaude de 2 à 4 de façon à restituer l'énergie emmagasinée de 4 à 2.

L'efficacité devient alors : $\eta_{ar} = 1 - \dfrac{T_F}{T_C}$.

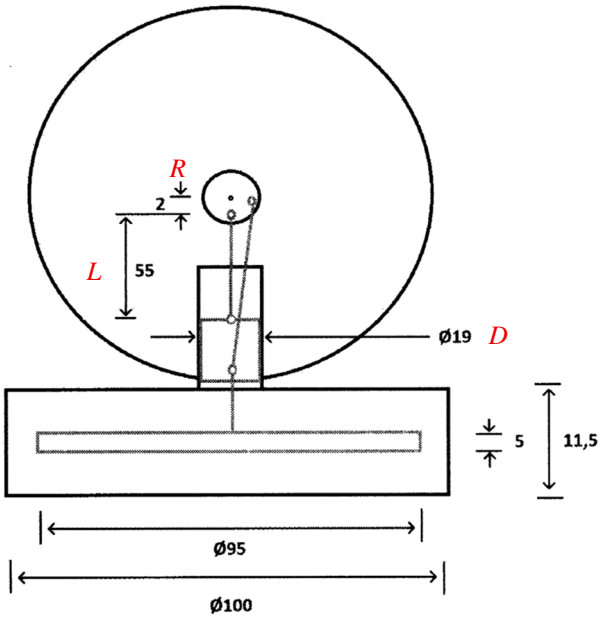
On reconnaît l'efficacité de Carnot valable pour un cycle ditherme réversible appelé cycle de Carnot et constitué de deux isothermes et de deux isentropiques. Cependant le cycle de Stirling n'est pas un cycle de Carnot car constitué de deux isothermes et de deux isochores et n'est pas ditherme en raison des échanges avec le régénérateur.
Néanmoins ce résultat permet de comprendre que le rendement est amélioré en présence du régénérateur (qui ne saurait être parfait en pratique).

Moteur Stirling instrumenté

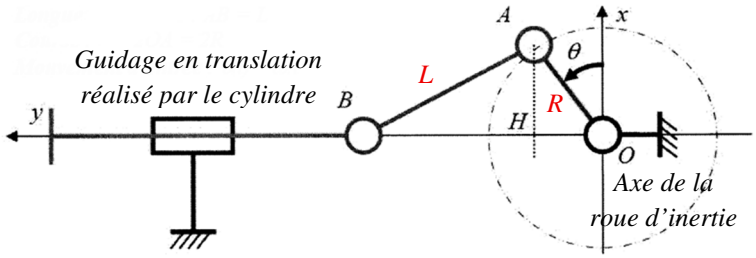
Deux fourches optiques permettant de déterminer la position de la roue d'inertie et un capteur de pression sont reliés à une carte Arduino Uno.



Données techniques



Distances / longueurs en mm



Mouvement d'entrée : $\theta(t) = \omega t$ (sens + = sens horaire de rotation)

L'angle θ est connu grâce aux deux fourches optiques (cf. ci-dessous).

On en déduit : $\theta = \frac{\pi}{2} + i \times \delta\theta$ (observer le passage du PMH la fourche 2)

Le volume du cylindre moteur est alors :

$$V_{cyl} = \left[OB - (L-R)\right]\frac{\pi D^2}{4} = \left[R\sin\theta + \sqrt{L^2 - R^2\cos^2\theta} - (L-R)\right]\frac{\pi D^2}{4}$$

Longueur de la bielle du piston de travail : $L = 55$ mm

Ecart OA du maneton de bielle moteur à l'axe de la roue d'inertie : $R = 2$ mm

Diamètre du piston de travail : $D = 19$ mm

Nombre de fentes découpées dans la roue d'inertie : $N = 50 \Rightarrow$ Angle entre deux fentes successives : $\delta\theta = 2\pi / N$

Parmi ces fentes, une est plus longue et permet de détecter le passage par le point mort haut (PMH) (position du piston de travail la plus haute donc volume maximum). Cette fente est détectée par la fourche optique n°2 (et par la fourche n°1).

A chaque passage d'une fente devant la fourche optique n°1, une mesure de pression est enregistrée.

Au cours d'un tour de la roue d'inertie, on dispose de 50 triplets de la forme $(i, k_{PMH}, \delta P_{discr})$ où :

- i est le numéro de la fente comptée à partir du PMH (celui-ci correspond à $i = 1$) ;
- k_{PMH} est le nombre de tours effectués par la roue (i.e. le nombre de passage de la fente la plus longue devant la fourche 2) ;
- δP_{discr} est la pression différentielle (différence entre la pression atmosphérique et la pression dans le cylindre).

Une surpression nulle correspond à 2,5 V pour une alimentation entre 0 et 5V.

La tension de sortie $U_{capteur}$ est discrétisée par le CAN 10 bits de la carte Arduino et fournit la valeur δP_{discr}.

La valeur de la pression absolue en **Pa** est alors : $P = P_0 + \delta P = P_0 + \frac{P_{discr} - 512}{1023} \times 5 / Sb$ où Sb est la sensibilité du capteur.

$Sb = \frac{\delta U_{capteur}}{\delta P_{réelle}}$ = 1,03 V/**k**Pa = 1,03/1000 V/Pa.

Par ailleurs, le volume minimal du cylindre moteur est V_{min} = 55 660 mm^3 hors volume mort (volume ajouté par la prise de pression).

Le volume réel du cylindre est donc : $V = V_{min} + \left[R \sin\theta + \sqrt{L^2 - R^2 \cos^2\theta} - (L - R) \right] \frac{\pi D^2}{4}$.

Sketch

Hypothèse : la fréquence des mesures est supérieure à la fréquence de rotation, i.e. tous les changements d'état des fourches optiques sont détectés.

Le programme ne commence à écrire les données qu'à partir d'une détection du PMH de façon à disposer des valeurs correspondant à un cycle entier et on définit par ailleurs un nombre maximum de cycles à enregistrer.

```
1  #define capteurPression A0  // Capteur de pression sur la broche A0
2  #define fourche1 A1         // Fourche optique n°1 sur la broche A1
3  #define fourche2 A2         // Fourche optique n°2 sur la broche A2 (détection PMH)
4  #define sepCol ";"          // Séparateur de colonnes -> identique dans pgm python
5
6  int compteur1 = 0;     // Compteur fourche optique n°1 (permet de calculer le volume)
7  int compteur2 = 0;     // Compteur fourche optique n°2 (nombre de passages par le PMH)
8  int etatFourche1;      // Etat de la fourche optique n°1
9  int etatFourche2;      // Etat de la fourche optique n°2
10 int memoire1 = LOW;    // Mémoire de l'état de la fourche optique n°1
11 int memoire2 = LOW;    // Mémoire de l'état de la fourche optique n°2
12 unsigned long t;       // Instant de mesure
13 int cpt2Max = 11;      // Nbre de tours à enregistrer (nombre de lignes à lire par python)
14
15 void setup() {
16   Serial.begin(115200);
17   pinMode(fourche1,INPUT);            // Configure la broche de la fourche n°1 en lecture
18   pinMode(fourche2,INPUT);            // Configure la broche de la fourche n°2 en lecture
19   while (!Serial) {;}                 // Ne rien faire tant que port série non disponible
20 }
21
22 void loop() {
23     while (compteur2 < cpt2Max) {
24       // Détection du passage de la fenêtre du PMH devant la fourche optique n°2
25       etatFourche2 = digitalRead(fourche2);   // Lecture de l'état de la fourche n°2
26       // Si état fourche 2 différent de état enregistré ET état "haut" (faisceau détecté)
27       if((etatFourche2 != memoire2) && (etatFourche2 == HIGH)) {
28             compteur2++;       // On incrémente le compteur2 (nbre de passages par le PMH)
29             compteur1 = 0;     // On remet le compteur1 à zero (angle dans [0, 2*pi])
30       }
31       // Détection du passage d'une des 50 fenêtres devant la fourche optique n°1
32       etatFourche1 = digitalRead(fourche1);   // Lecture de l'état de la fourche n°1
33       // Si état fourche 1 différent de état enregistré ET état "haut" (faisceau détecté)
34       // La condition compteur2 > 0 permet de démarrer la transmission sur un PMH
35       if((etatFourche1 != memoire1) && (etatFourche1 == HIGH) && (compteur2 > 0)) {
36         compteur1++;           // On incrémente le compteur1
37         t = micros();          // Instant de mesure en µs
38         // Affichage des quadruplets de valeurs de compteur1, compteur2, pression, t
39         Serial.print(compteur1); Serial.print(sepCol);
40         Serial.print(compteur2); Serial.print(sepCol);
41         Serial.print(analogRead(capteurPression)); Serial.print(sepCol);
42         Serial.println(t);
43       }
44       // Mémorisation des états pour détecter les changements à la lecture suivante
45       memoire1 = etatFourche1;
46       memoire2 = etatFourche2;
47     }
48     Serial.println("Stop");
49     while (true) {;}
50 }
```

⚠ Ligne 11 : pour les tests, utiliser une valeur « raisonnable » pour la variable `cpt2Max` (3 ou 4).
Noter que la numérotation débute à 0 mais que le cycle n°0 sera ignoré car incomplet donc le premier cycle enregistré aura le n°1 et que le dernier aura pour n° cpt2Max-1 (donc 10 cycles complets avec `cpt2Max = 11`) .

Programme Python

Ressources :

💻 Le sketch Arduino et le programme python à compléter sont disponibles en téléchargement via le QRcode (lien vers la rubrique TP de gilles.beharelle.fr) : Tutoriels Python – Arduino, à la rubrique Arduino – Découvrir en expérimentant : Moteur de Stirling.

✏ Compléter le code de la fonction calcul_PV (directement dans Pyzo ou Spyder)

```python
import serial
import serial.tools.list_ports
import matplotlib.pyplot as plt
from matplotlib import animation
import matplotlib.colors as mcolors
import numpy as np

# calcul_PV permet de tracer :
#    - tous les cycles avec num_cycle=None
#    - un unique cycle avec num_cycle= numéro cycle souhaité
def calcul_PV(dataN, num_cycle=None, Po=1e5, unite='mm3'):
    """ dataN = tableau de lignes de la forme [i, k, Pdiscr, t]
        i = n° de la fente comptée à partir du PMH
        k = nombre de tours (nombre de passages au PMH)
        Pdiscr = pression différentielle renvoyée par CAN Arduino (bits)
        t = instant de mesure (µs)
        num_cycle = n° du cycle souhaité = l'une des valeurs de k dans data
        Po  : pression atmosphérique en Pa
        unite : 'mm3' ou 'm3' ou 'SI'
    """
    assert unite in ['mm3', 'm3', 'SI'], "Unité incorrecte ('mm3', 'm3' ou 'SI)"
    if num_cycle is None:   # Calculs pour tous les cycles
        data = dataN
    else:                   # Calculs pour le cycle spécifié par num_cycle
        assert float(num_cycle) in dataN[:,1], "n° cycle incorrect"
        data = np.array([l for l in dataN if l[1]==float(num_cycle)])

    # Caractéristiques mécaniques du moteur
    L =                         # Longueur bielle piston moteur (mm)
    R =                         # Rayon maneton bielle moteur (mm)
    D =                         # Diamètre piston moteur (mm)
    S =                         # Surface piston moteur (mm**2)
    dtheta =                    # Angle entre 2 fentes (rad)
    Vmin =                      # Volume minimum du cylindre moteur (mm**3)

    # Calcul du volume du gaz dans le cylindre moteur
    i =                         # n° fente à partir du PMH
    theta =                     # Angle correspondant à la fente n°i
    OB =                        # Position piston
    V =                         # Volume (mm**3)
    if unite in ['m3', 'SI']:
        V *= 1e-9

    # Capteur de pression différentielle
    Sb =                        # Sensibilité = dV/dPréelle = 1.03/1000 V/Pa

    # Pression
    dPd =                       # Pression différentielle discrétisée en bits
    dPv =                       # Pression différentielle en V
    dPr =                       # Pression différentielle réelle en Pa
    P =                         # Pression absolue en Pa

    # Instant de mesure
    t =                         # t en µs (cf. sketch Arduino)

    return P, V, t
```

✏ En lisant le code du programme fourni, déterminer les noms des tableaux/listes contenant les données :
- au format texte (tableau 1D de lignes) ;
- au format numpy (tableau 2D).

💡 Des informations supplémentaire concernant la fonction portArduino sont disponibles via le lien précédent (Tutoriels Python – Arduino, à la rubrique Python – Tutoriels détaillés : Acquisition de données via le port série).

La dernière cellule du programme python fourni permet de sauvegarder dans un fichier texte les données volatiles (perdues lors d'un redémarrage du noyau python par exemple).

Acquisition des données

1. Brancher la carte Arduino sur un port USB du PC (à l'arrière du PC et non en façade).
2. Téléverser le sketch sur la carte, prendre `cpt2Max = 3` .
3. Afficher le moniteur série et faire tourner délicatement le moteur dans le sens indiqué page 3 de ce document : les données apparaissent dans le moniteur série (relancer le moteur si nécessaire tant que l'affichage n'est pas interrompu).
4. Fermer le moniteur série.
5. Identifier la variable en relation avec `cpt2Max` dans le programme python et ajuster sa valeur.
6. Exécuter le programme python et relancer le moteur à la main comme à l'étape 3 : les données sont affichées dans le shell puis un cycle est tracé.
7. Modifier la variable `cpt2Max` du sketch (prendre 11 par exemple) et la variable python correspondante.
8. Poser le moteur sur une tasse d'eau chaude, attendre quelques secondes puis le lancer très doucement comme à l'étape 3 s'il ne démarre pas seul.
9. Exécuter le programme python.
10. ⚠ Lorsque le moteur s'arrête poser le moteur sur un **chiffon** et non sur la ~~paillasse~~.
11. Sauvegarder l'acquisition dans un fichier (effectuer une nouvelle acquisition si nécessaire).
12. Ouvrir le fichier (possible depuis l'explorateur de fichiers de Pyzo) et vérifier que les données sont correctes : pas de lignes tronquées ou sautées, nombre entier de cycles complets (la première colonne doit varier de 1 à 50 de façon cyclique).

Ecrire un programme permettant de lire, traiter les données du fichier de sauvegarde et les sauvegarder au format texte dans une liste nommée dataFichier (cf. Tutoriels Python – Arduino, à la rubrique Python – Tutoriels détaillés : Fichiers texte).
Transformer cette liste en tableau numpy nommé dataFichierN.
Vérifier que la fonction calcul_PV permet de tracer le(s) cycle(s).

Traitement des données – Aspects thermodynamiques

13. Définir « à la main » (en analysant les données d'un cycle) ou écrire une fonction permettant de déterminer les index des volumes minimum et maximum dans la liste des volumes au cours d'un unique cycle.
14. Ecrire une fonction aire(x, y) permettant de calculer l'aire sous la courbe $y(x)$ par la méthode des trapèzes.
15. Ecrire une fonction puissance(numCycle) permettant de déterminer la puissance fournie par le moteur au cours du cycle n°numCycle (cette fonction appellera la fonction calcul_PV de façon à disposer des données pour ce cycle en précisant que l'unité doit être m^3).
16. Tester cette fonction.
17. Ecrire une fonction puissanceMoyenne(tab) permettant d'obtenir la puissance moyenne calculée sur tous les cycles disponibles dans le tableau de données *tab*.

Prolongements

18. Imaginer un protocole permettant d'estimer le rendement de ce moteur.