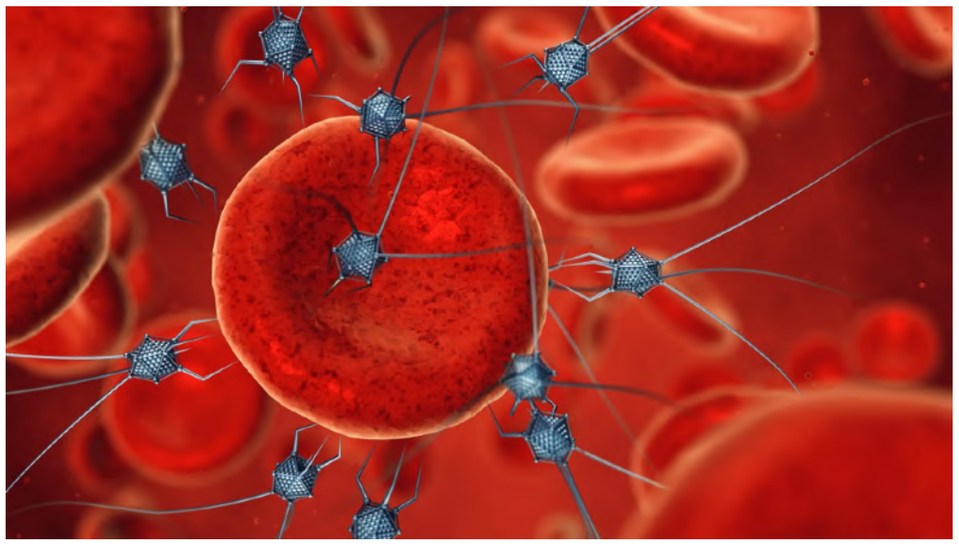
Course de robots dans un fluide visqueux

Alexis NOÉ 11318

Mise en contexte

Besoins

- Besoin de traitements spécifiques chez les sportifs
- Substances à administrer dans des zones précises du corps
- Utilisation de nanorobots



Source: iatranshumanisme.com

Exemples de traitements

- Traitement d'une inflammation
- Administration de compléments nutritionnels ciblés
- Traitement des microdéchirures musculaires

Problème : le sang est visqueux à l'échelle d'un nanorobot

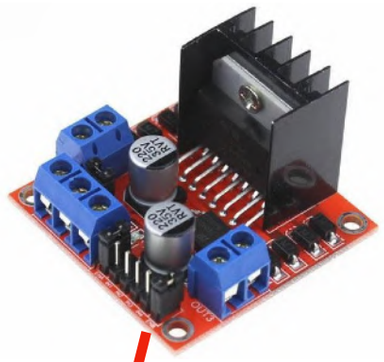
Problématique

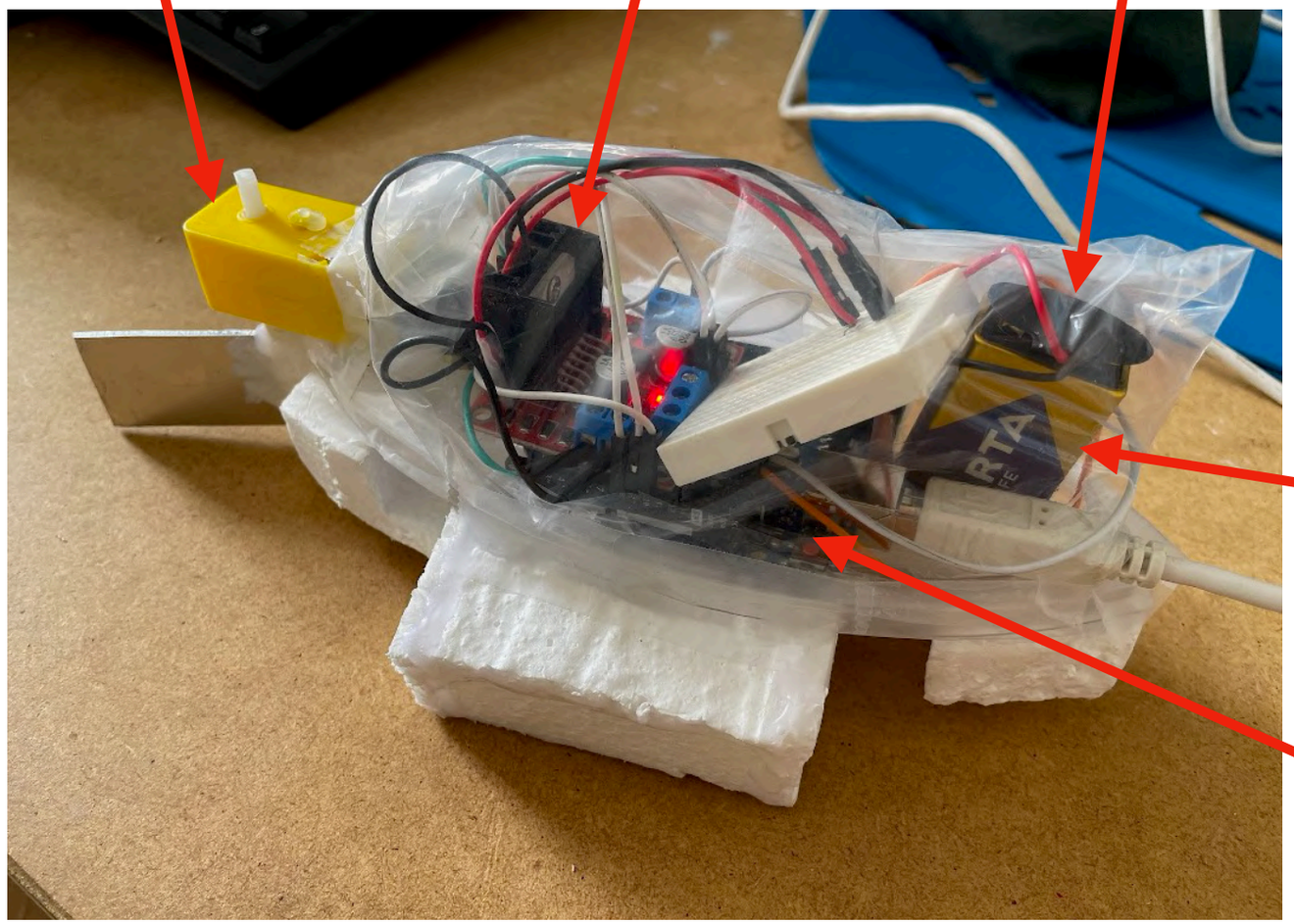
Comment concevoir un robot capable de progresser dans un fluide visqueux ?

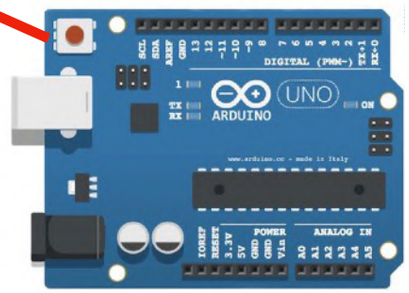
Plan

1. **Introduction et contexte**
2. **Difficulté du déplacement**
3. **Explication théorique**
4. **Fabrication des robots**
5. **Conclusion**

Robot 1 - Le bateau poisson





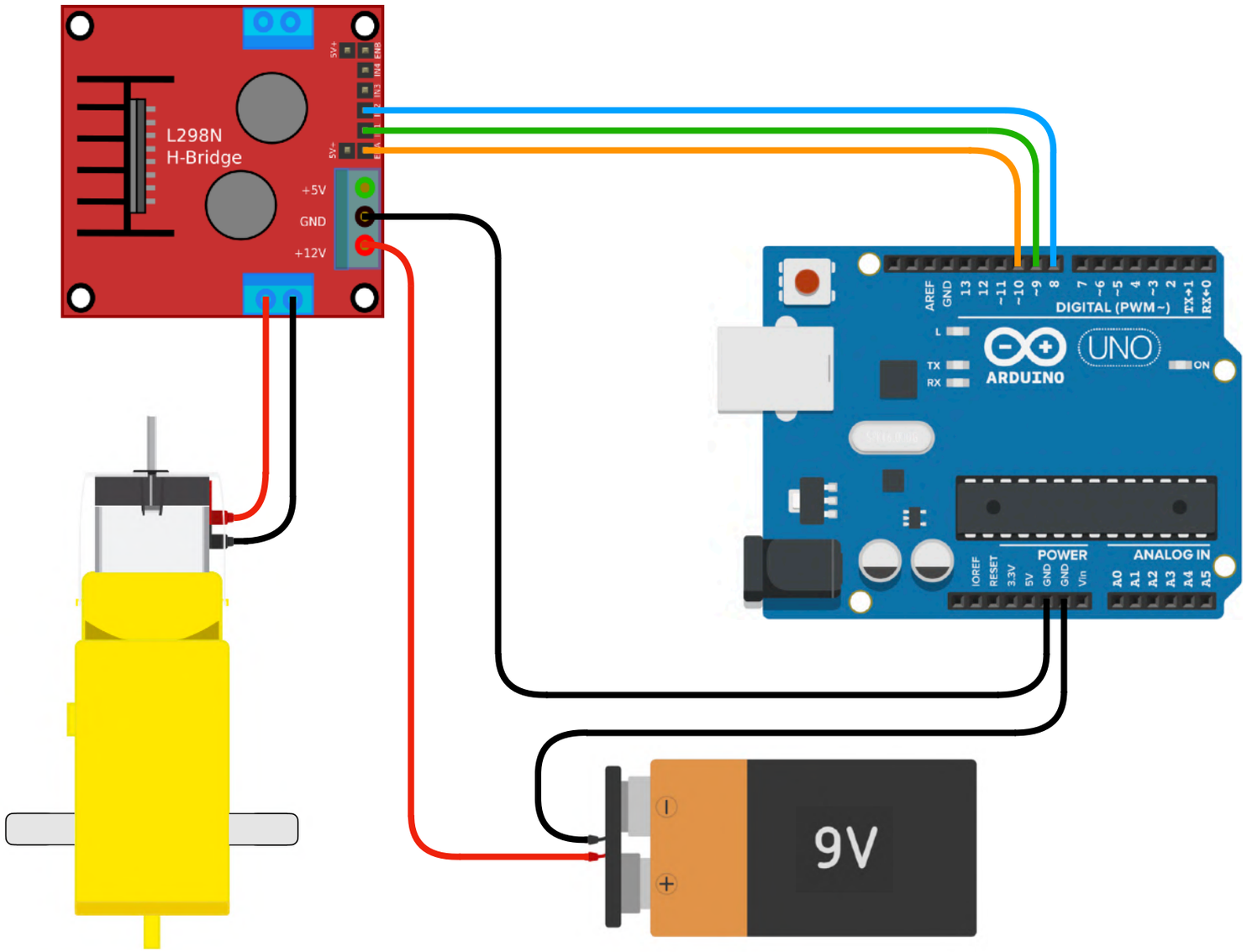


Fiche technique

- masse : $m = 245\ g$
- coque en plastique
- flotteurs en polystyrène
- étanche
- utilisation de Arduino
- taille caractéristique :
  $L = 10\ cm$ (largeur)

Robot 1 - Le bateau poisson

Schéma du montage Arduino



Fabrication du robot

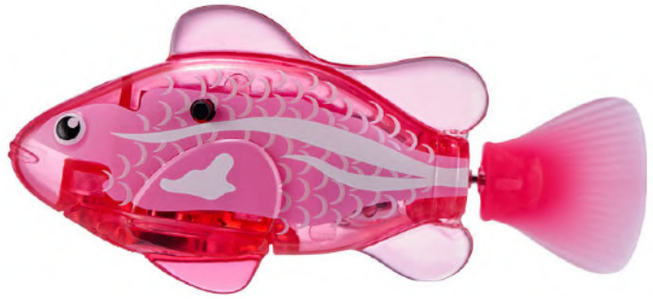
Robot 1 - Le bateau poisson

Eau	Liquide vaisselle
	
- Progression sans difficulté - Vitesse moyenne : $v = 0{,}05\ m/s = 5\ cm/s$ - Nombre de Reynolds : $Re \approx 5000$	- Oscillations sur place - Vitesse moyenne nulle

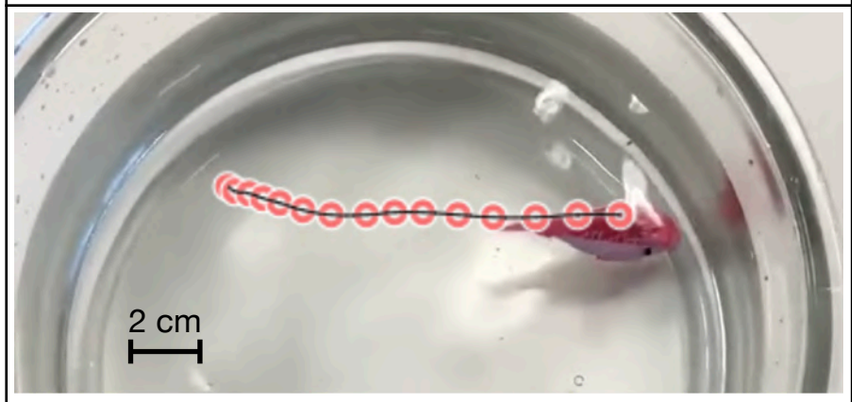
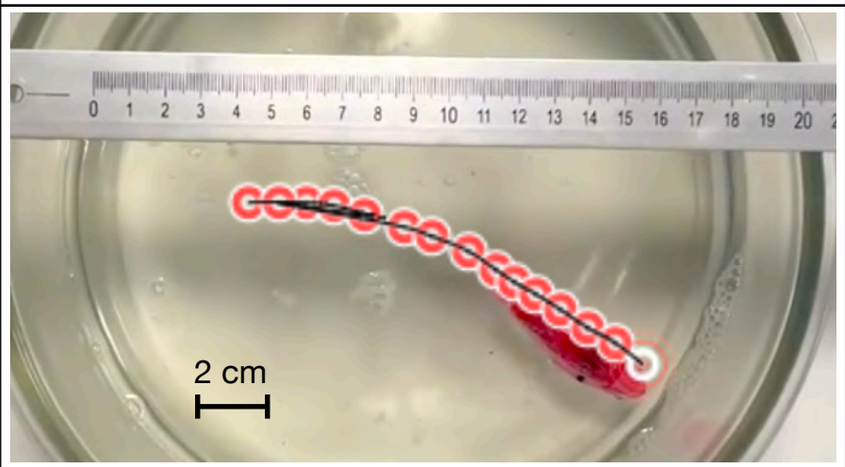
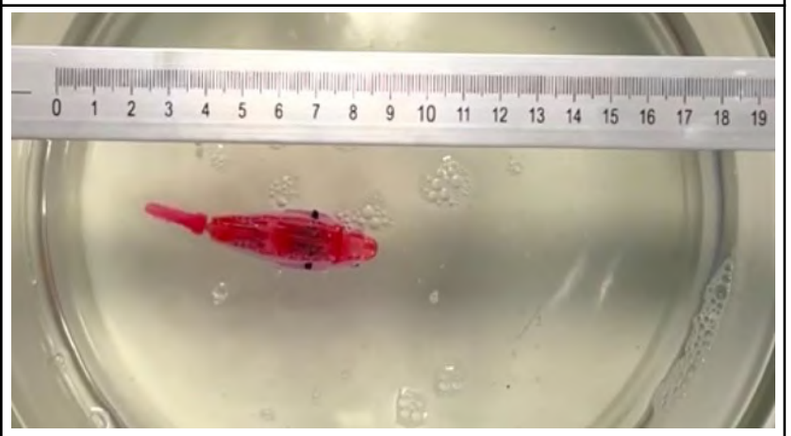
Aucun déplacement dans le liquide vaisselle

Mise en évidence

Expérience : Poisson robotisé RoboFish



Chronophotographie du déplacement dans différentes conditions

Eau	Liquide vaisselle avec nageoire souple	Liquide vaisselle avec nageoire rigidifiée
		 Oscillations « sur place »
Durée de l'acquisition : 2 s Vitesse moyenne : 0,11 m/s = 0,4 km/h Nombre de Reynolds : Re = 2200	Durée de l'acquisition : 2 min 30 s Vitesse moyenne : 0,8 mm/s = 2,9 m/h Nombre de Reynolds : Re = 0,04	Vitesse moyenne nulle Nombre de Reynolds : Re→0

Le poisson avance dans le liquide vaisselle et oscille sur place si on rigidifie sa nageoire

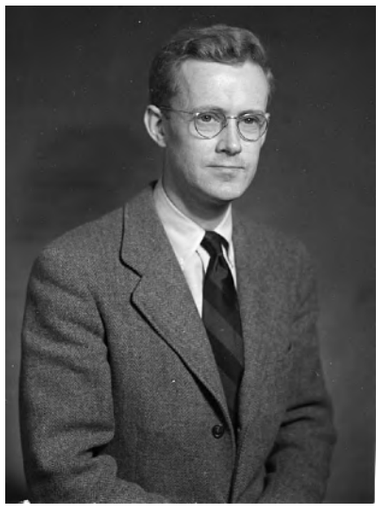
Théorème de la coquille Saint-Jacques et mouvements réciproques

Mouvement réciproque :

- Simple
- Un seul degré de liberté
- Retour à la position initiale par le mouvement opposé

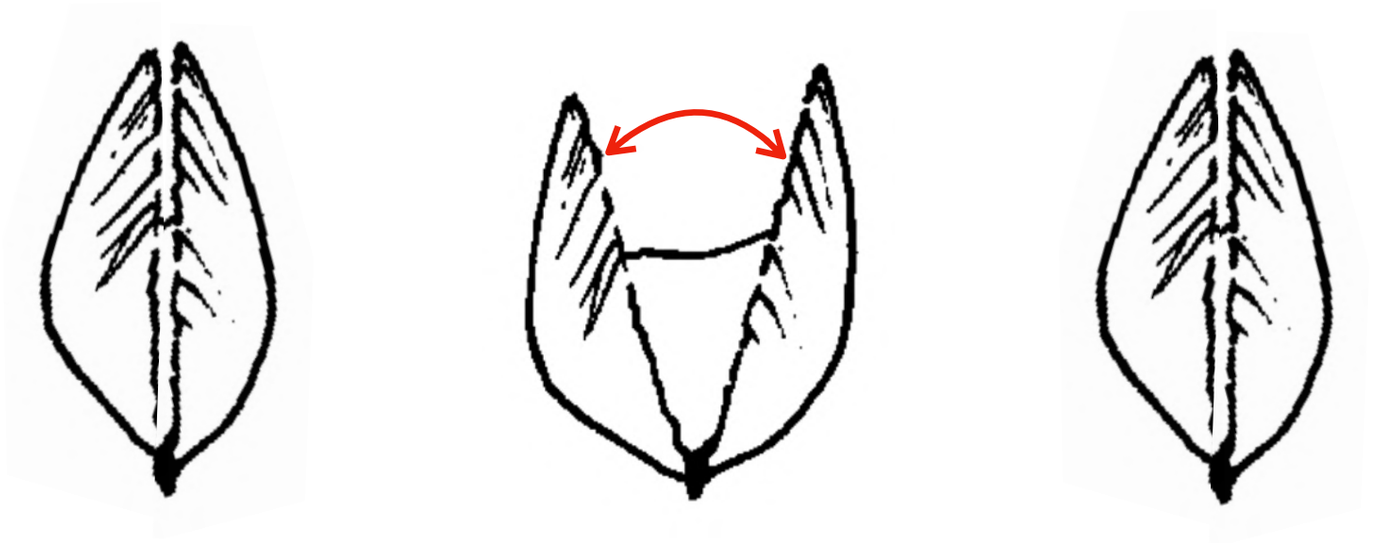
Théorème de la coquille Saint-Jacques (1977)

Un mouvement réciproque n'induit aucun déplacement net dans un fluide visqueux ($Re < 2000$)



Edward Mills Purcell

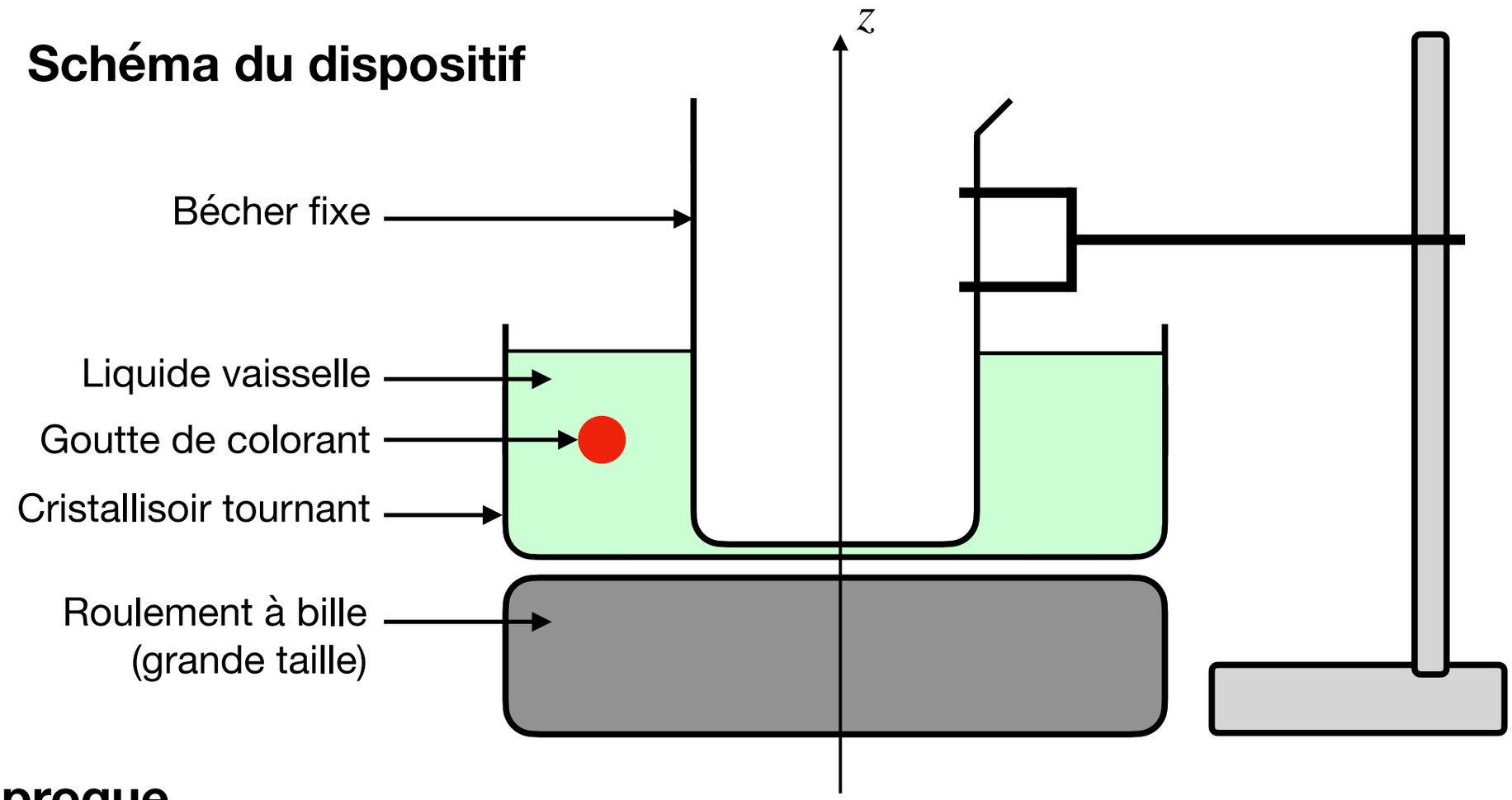
Exemple de mouvement réciproque



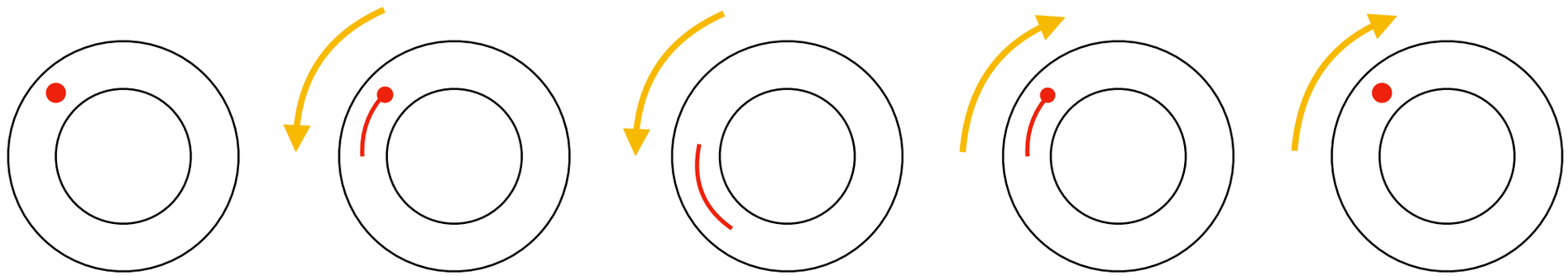
Mise en évidence

Expérience : Écoulement de Couette cylindrique

Schéma du dispositif



Mouvement réciproque



Mise en évidence

Expérience : Écoulement de Couette cylindrique

La goutte se reforme et retrouve sa position initiale

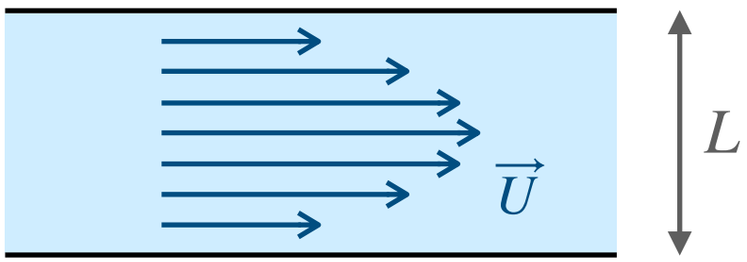
Nombre de Reynolds et viscosimètre à bille

Nombre de Reynolds

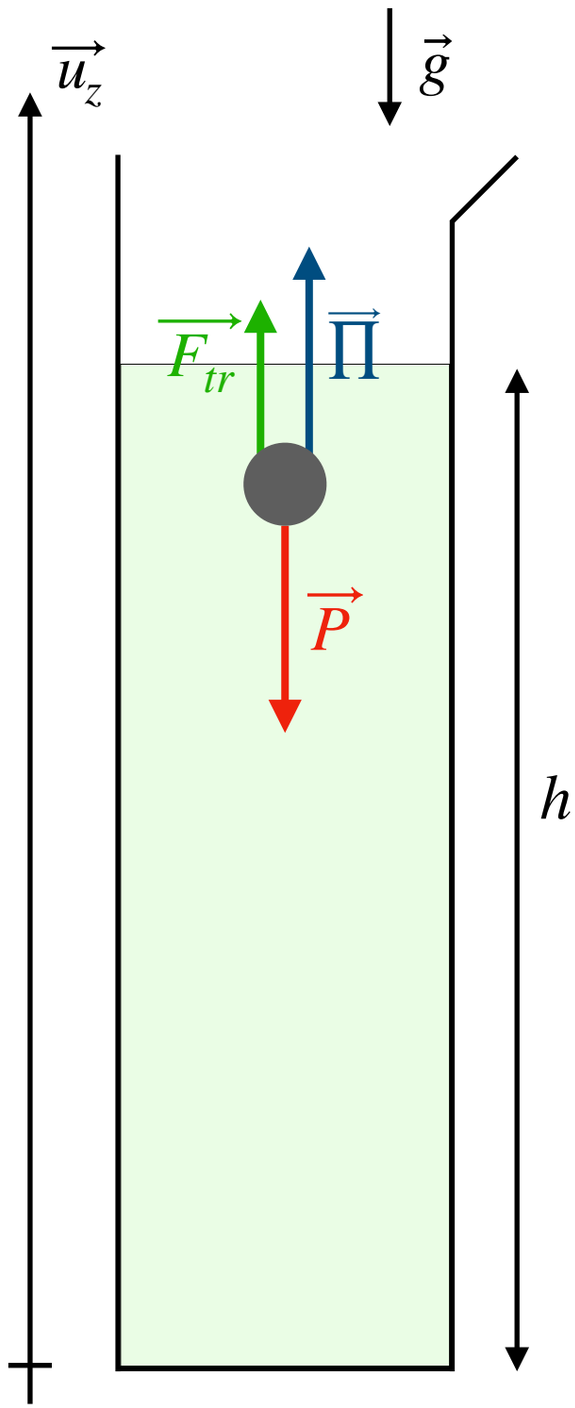
$$Re = \frac{\rho_f U L}{\eta}$$

η : viscosité dynamique du fluide ($Pl = Pa \cdot s$)

ρ_f : masse volumique du fluide (m^3/s)



- **Intérêt des écoulements similaires**



Formule de Stokes : trainée pour un écoulement laminaire

$$\overrightarrow{F_{tr}} = -6\,\pi\,\eta\,r\,\overrightarrow{v_\infty}$$

η : viscosité dynamique du fluide

r : rayon de la bille

Expression de la viscosité dynamique η :

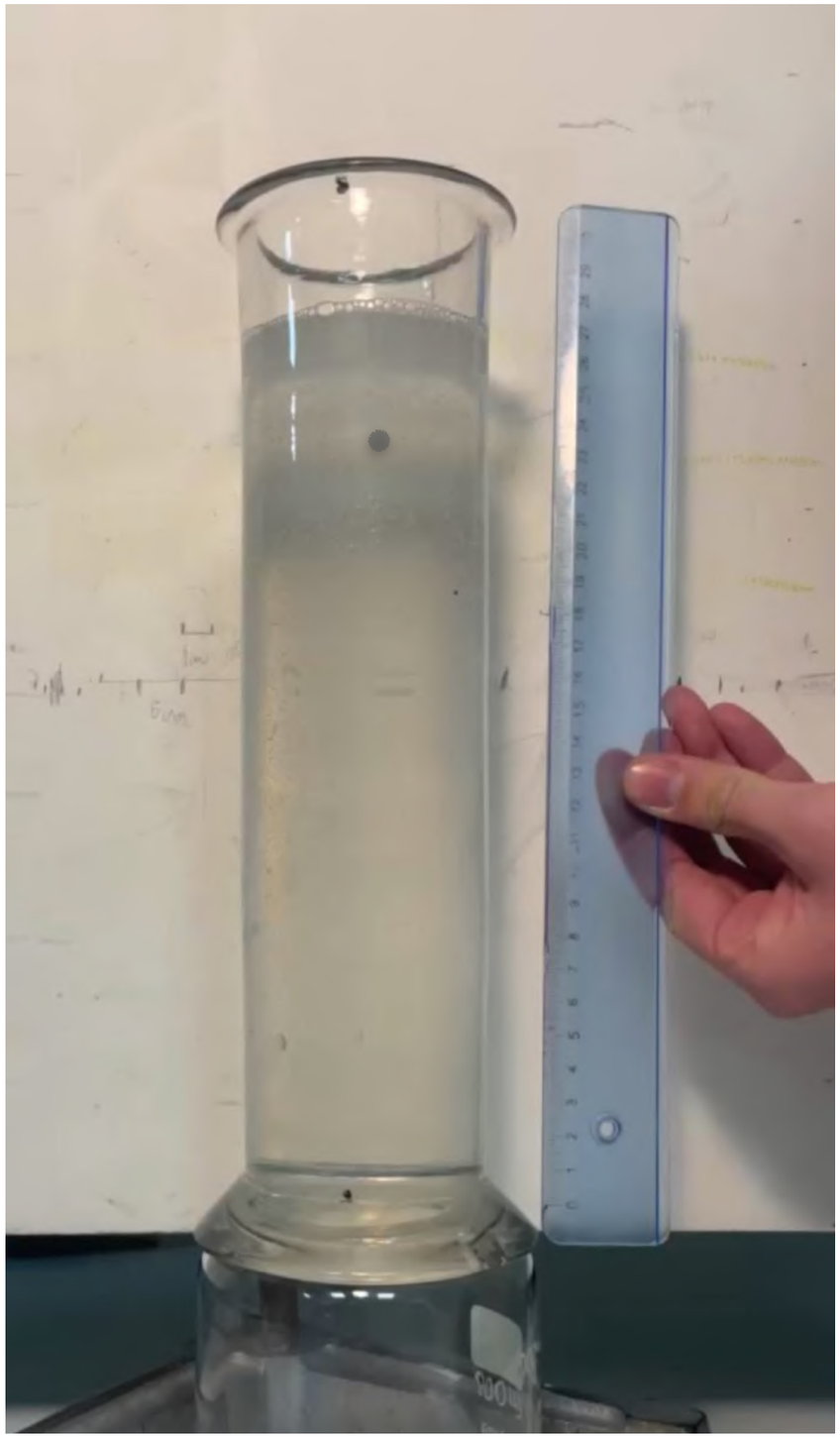
$$\eta = \frac{(m - \rho_f V_b)\,g}{6\,\pi\,r\,v_\infty} = \frac{2}{9}\frac{(\rho_b - \rho_f)\,r^2\,g}{v_\infty}$$

ρ_b : masse volumique de la bille

Viscosimètre à chute de bille - Expérimentation

Protocole :

- mesure de m et de r
- calcul de p_b et p_f
- pointage vidéo pour obtenir v_∞
- calcul de la viscosité η

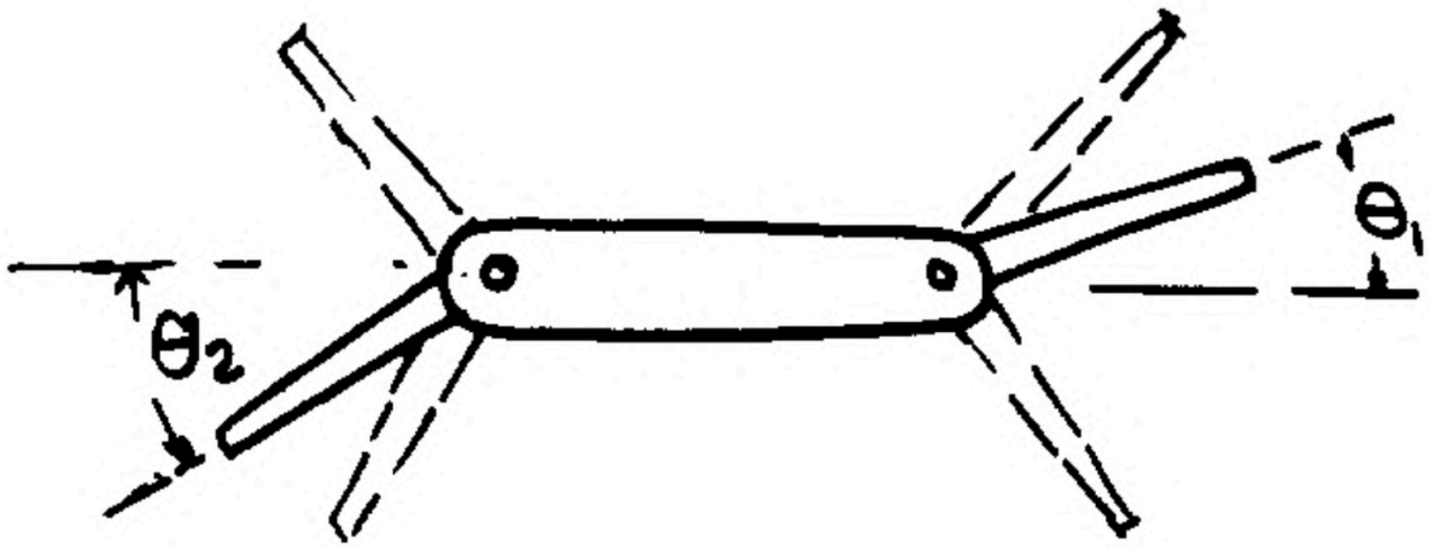


Résultats pour ma bille et le liquide vaisselle

- rayon $r = 4{,}10 \pm 0{,}03\ mm$
- masse de 100 billes $M = 16{,}60 \pm 0{,}06\ g$
- masse d'une bille $m = 0{,}170 \pm 0{,}006\ g$
- vitesse $v_\infty = 20{,}40 \pm 1{,}68\ mm/s$
- **viscosité dynamique mesurée** $\eta = 0{,}29 \pm 0{,}03\ Pl$
- **valeur théorique tabulée** $\eta = 0{,}4\ Pl$

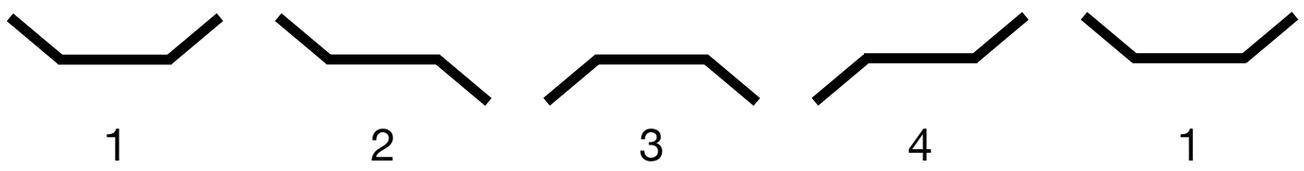
Robot 2 - Nageur à palettes

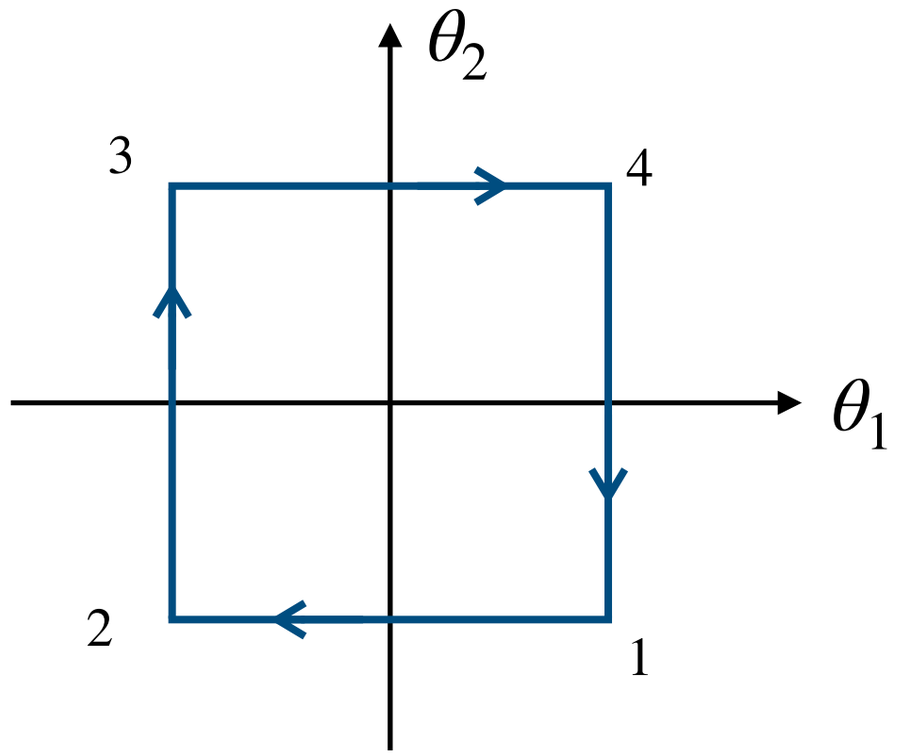
Principe du modèle de Purcell



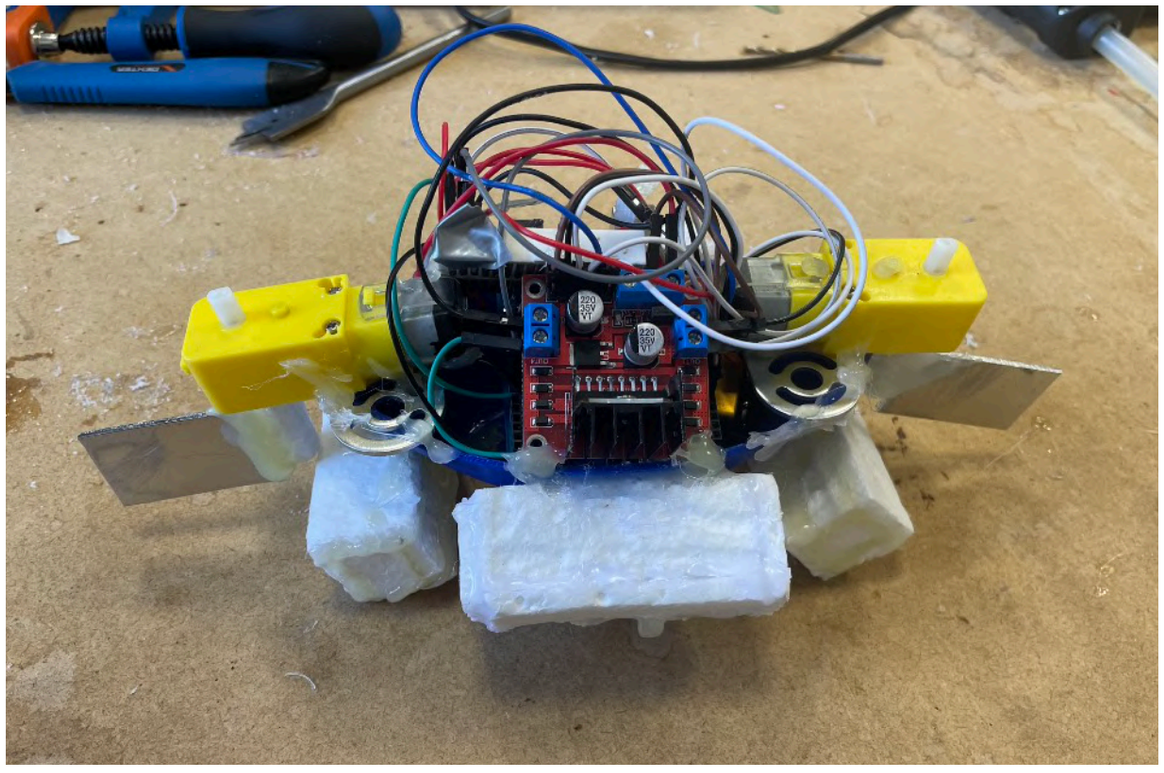
Dessin de Edward Mills Purcell tiré de sa publication *Life at Low Reynolds Number*

Mouvement cyclique :





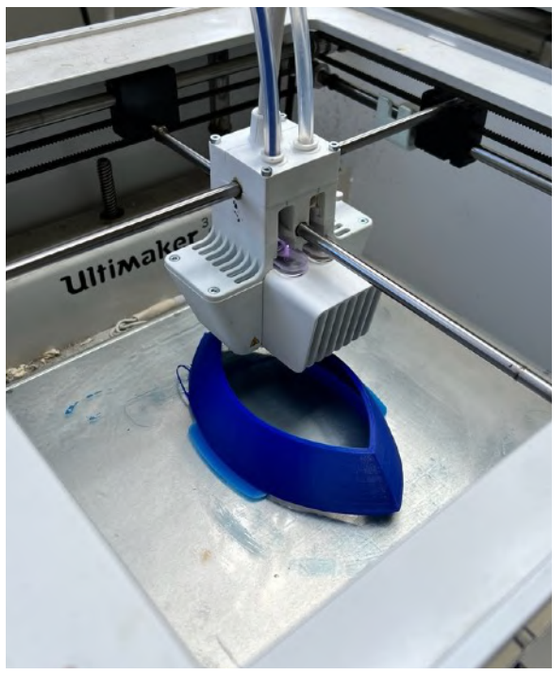
Robot 2 - Nageur à palettes



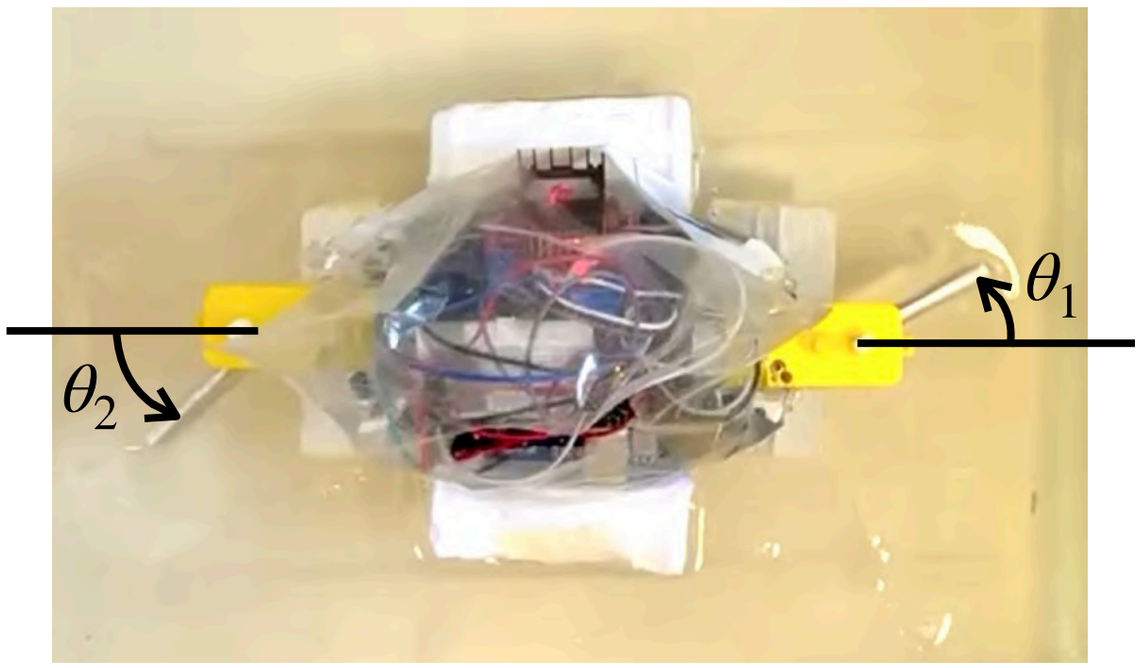
Fiche technique

- Masse : $m = 270\ g$
- Coque conçue sur onshape.com et imprimée en 3D
- Flotteurs en polystyrène
- Étanche
- Utilisation de Arduino
- Taille caractéristique :

$$L = 12\ cm \quad \text{(largeur)}$$

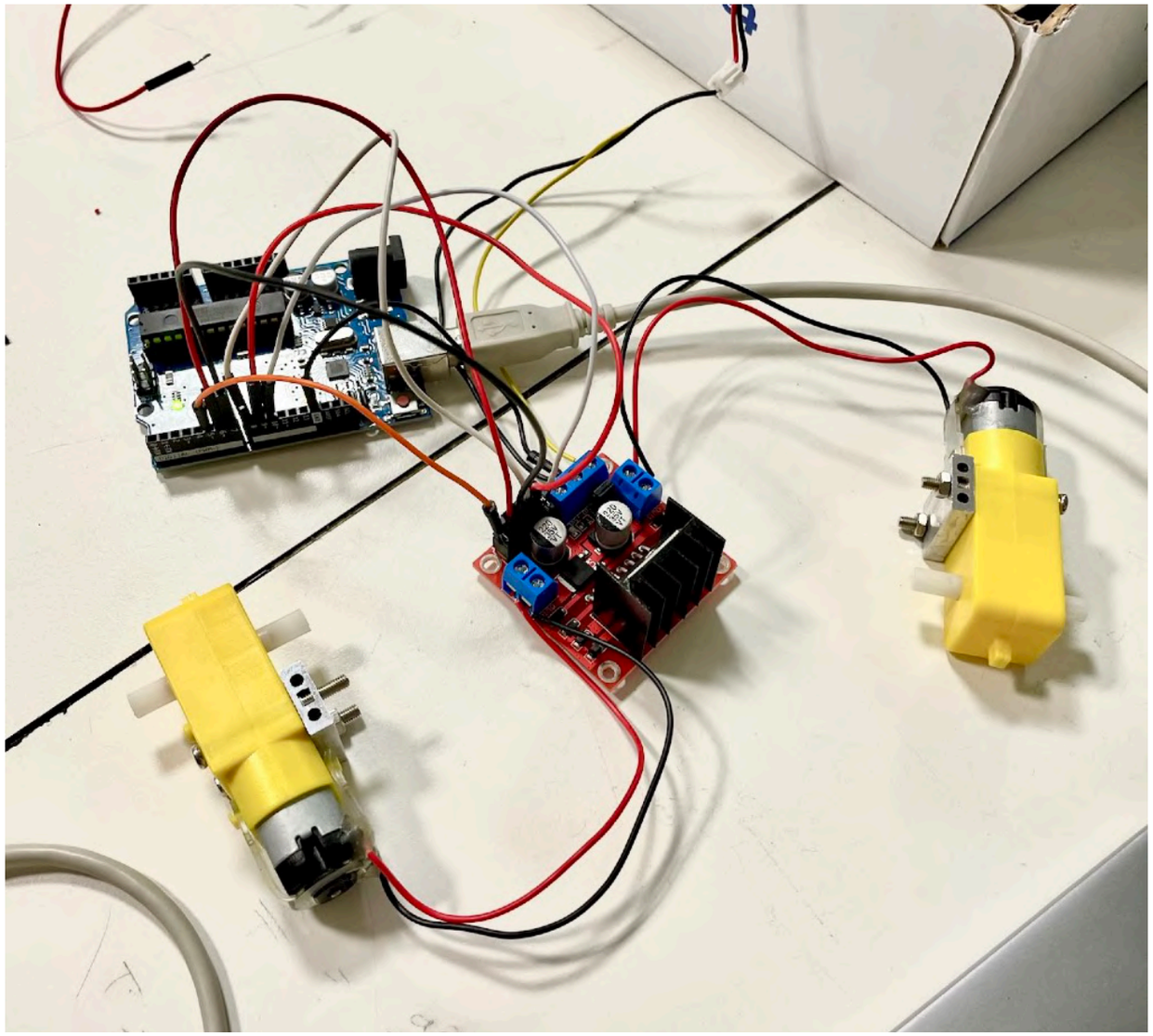


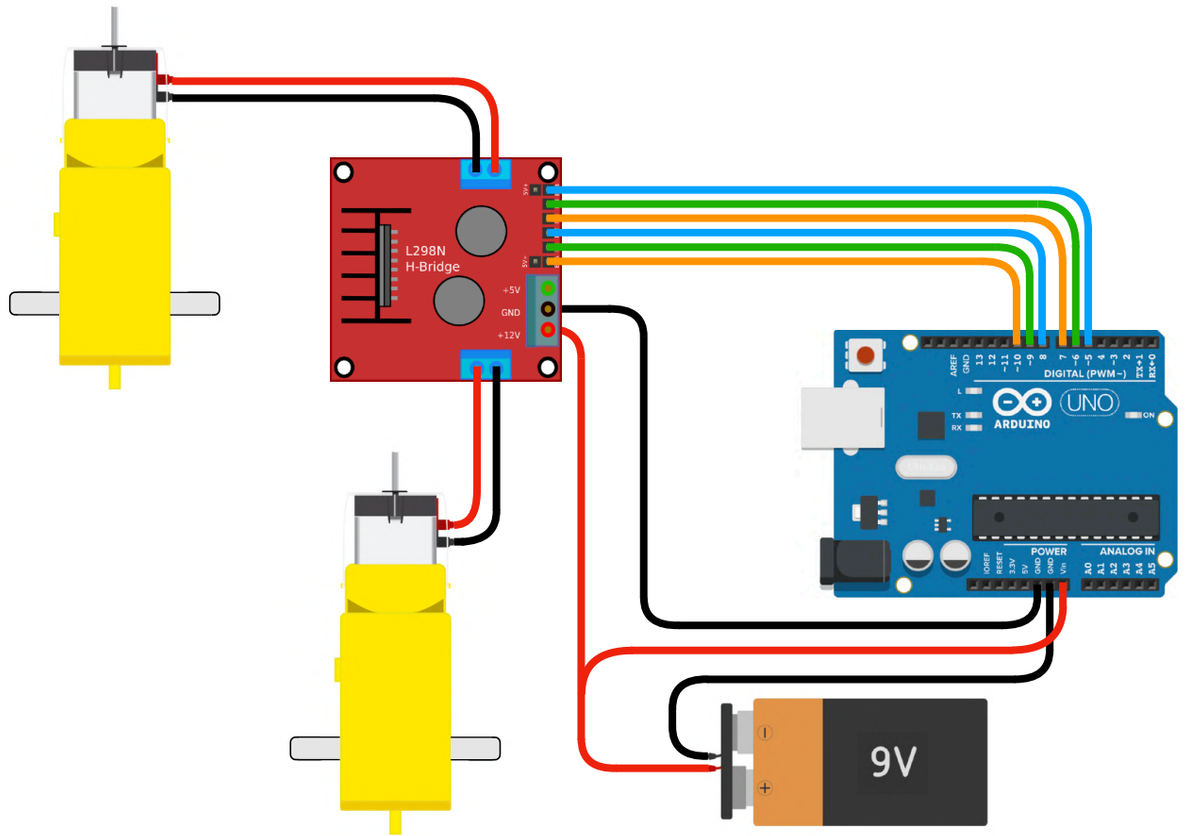
Impression de la coque du bateau en 3D



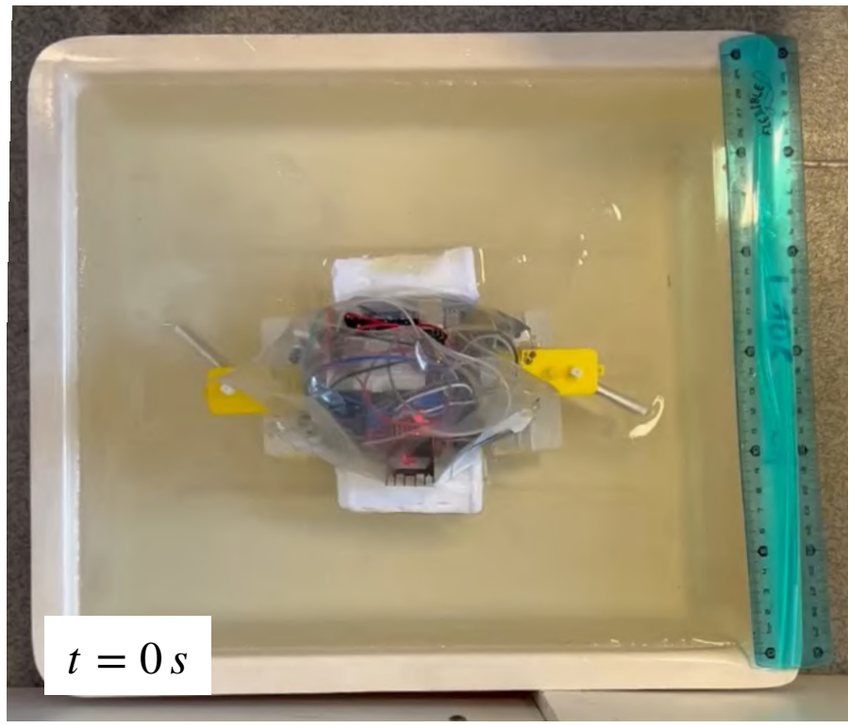
Robot 2 - Nageur à palettes

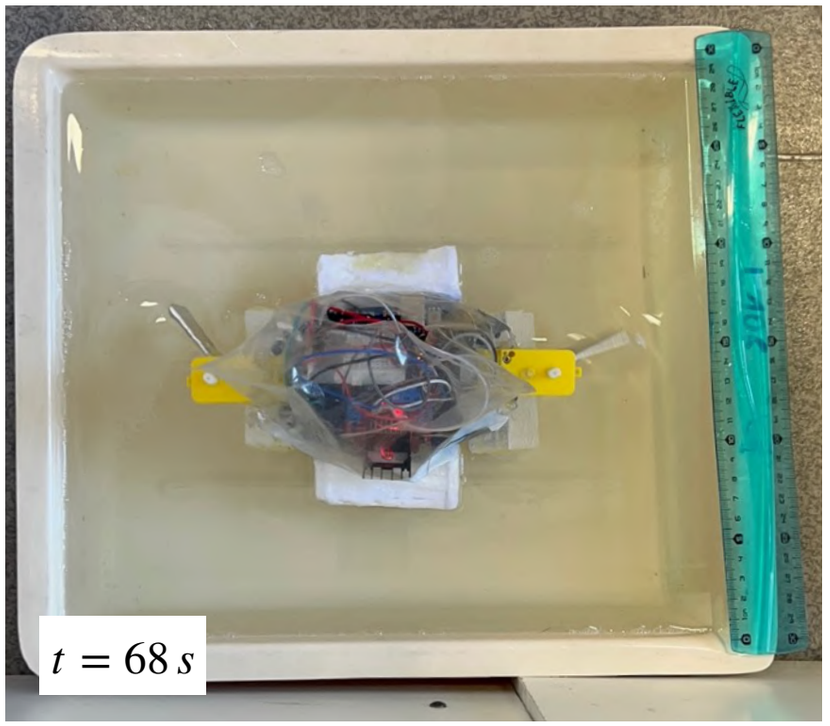
Schéma du montage Arduino

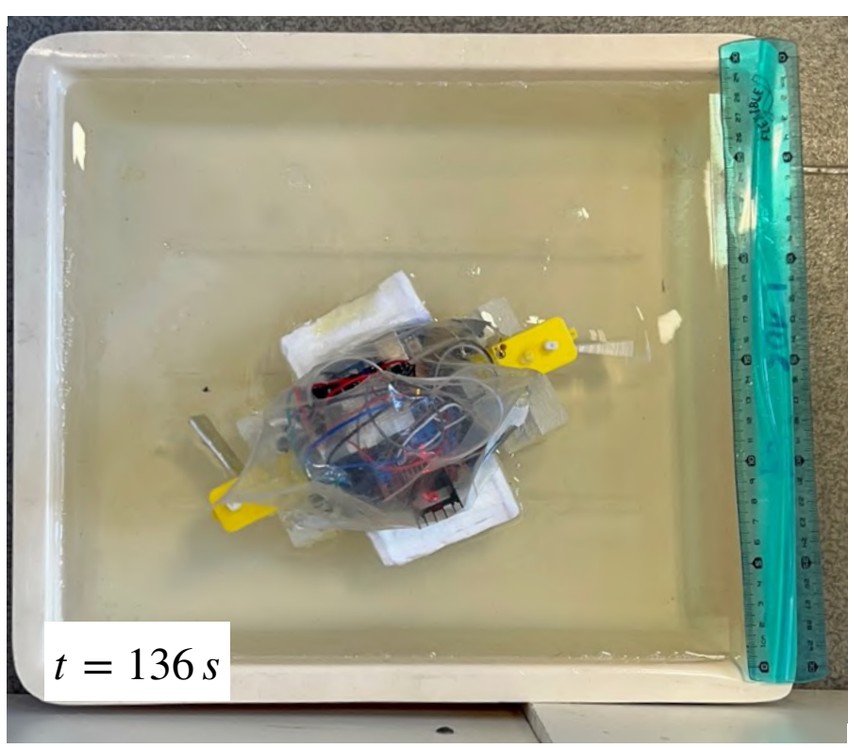


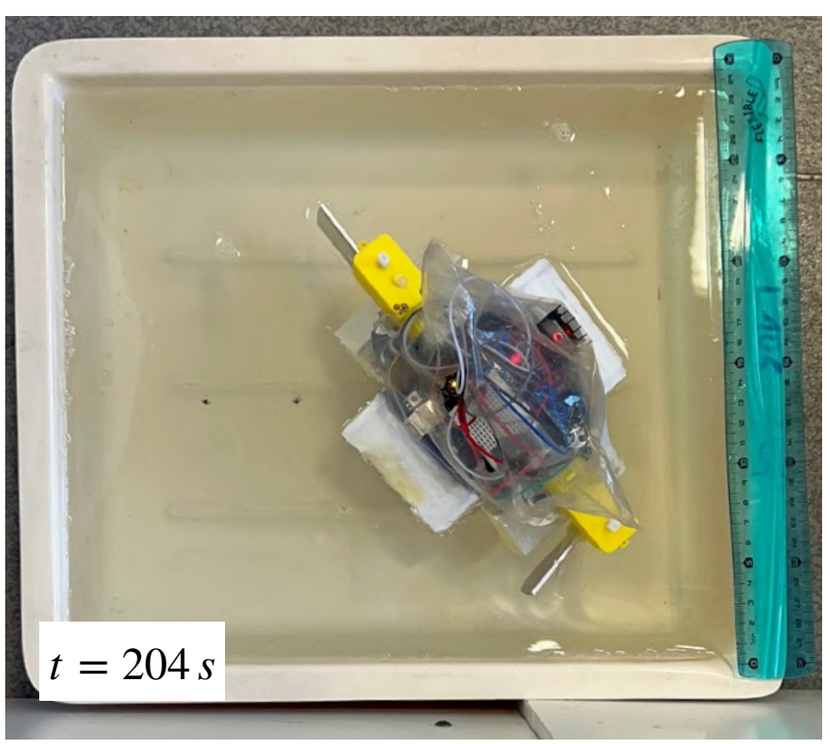


Robot 2 - Nageur à palettes







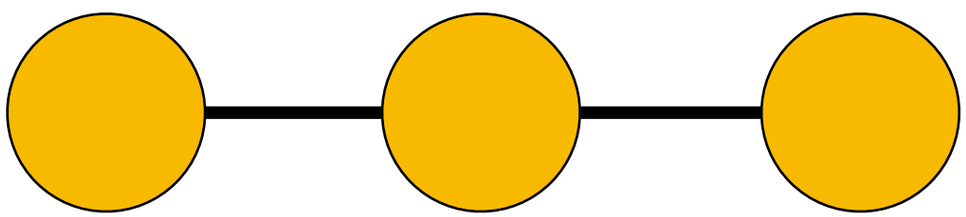


Observation expérimentale :

- Progression légère
- Rotations
- Essai avec de nombreux réglages différents
- **Mouvement imprévisible**
- **Pas de translation**

Robot 3 - Nageur de Najafi et Golestanian

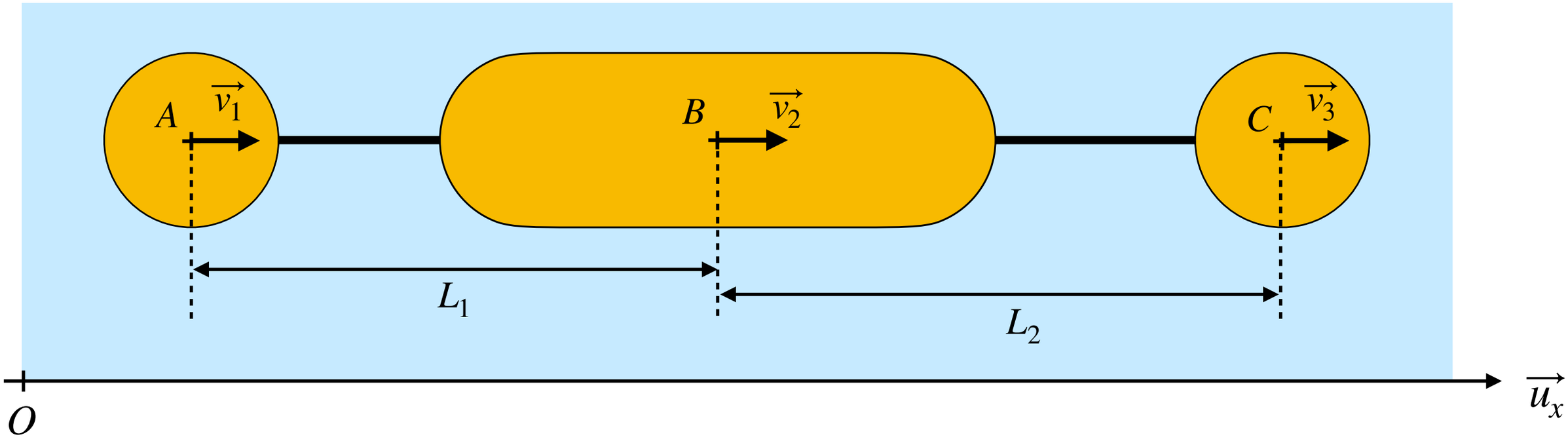
Principe



- 3 boules
- Boules extérieures en translation par rapport à la boule centrale
- Repose sur la force de trainée de Stokes

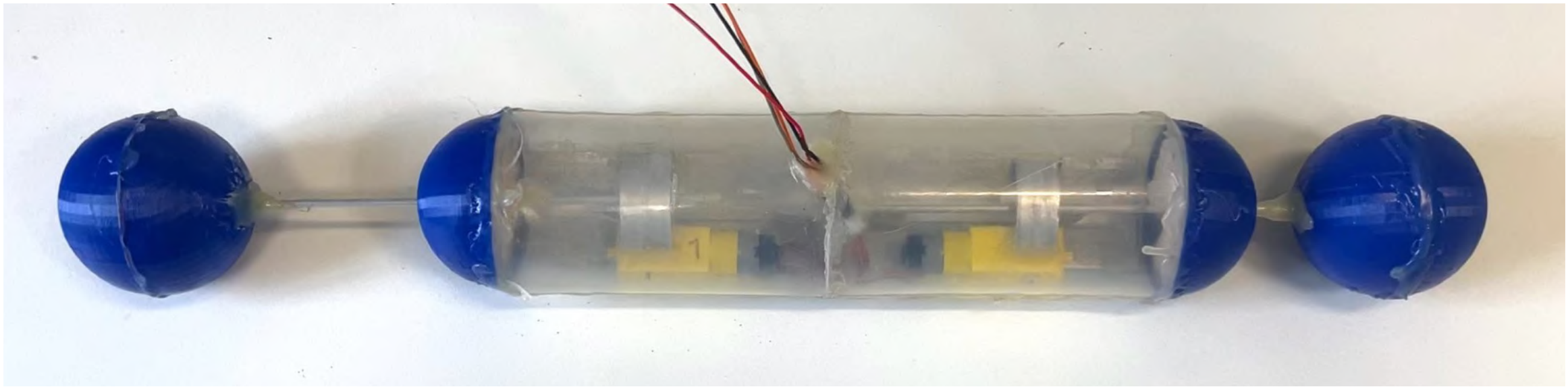
$$\overrightarrow{F_{tr}} = -6\,\pi\,\eta\,r\,\vec{v}$$

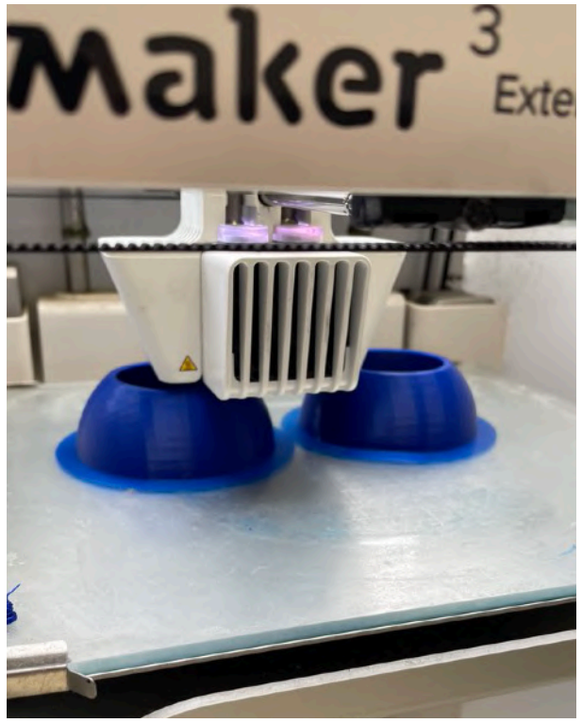
- Les distances entre les boules évoluent, avec des vitesses réglables



Robot 3 - Nageur de Najafi et Golestanian

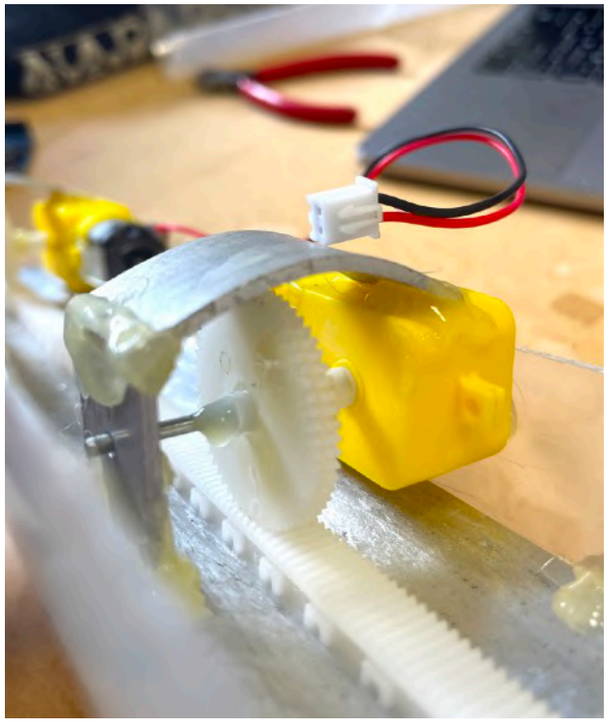
Fabrication du nageur





Impression des boules en 3D

Pignon crémaillère



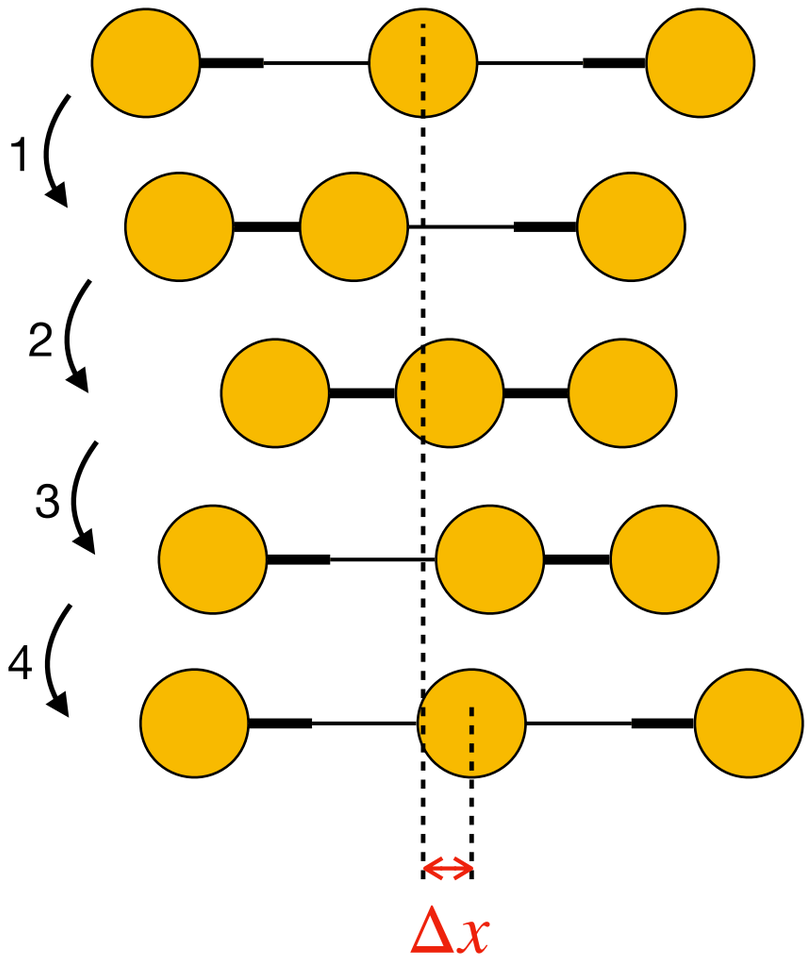
Fiche technique

- Longueur totale : $64\ cm$
- Masse (avec sable) : $m = 710\ g$
- Translation des boules
- Taille caractéristique : $L = 7\ cm$ (diamètre)
- Circuit électronique à l'extérieur

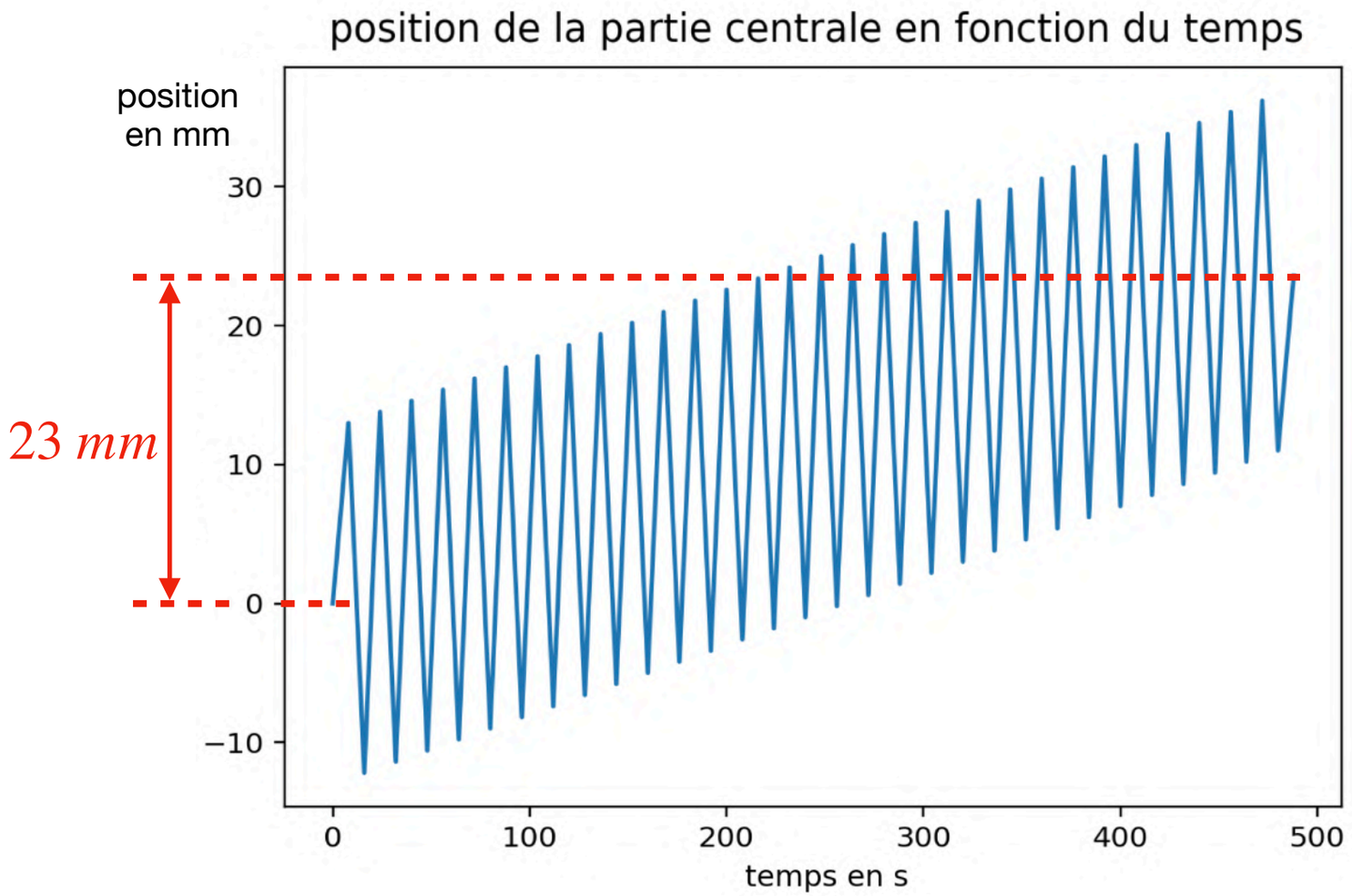
Robot 3 - Nageur de Najafi et Golestanian

Approche théorique

On répète le cycle :



Simulation numérique



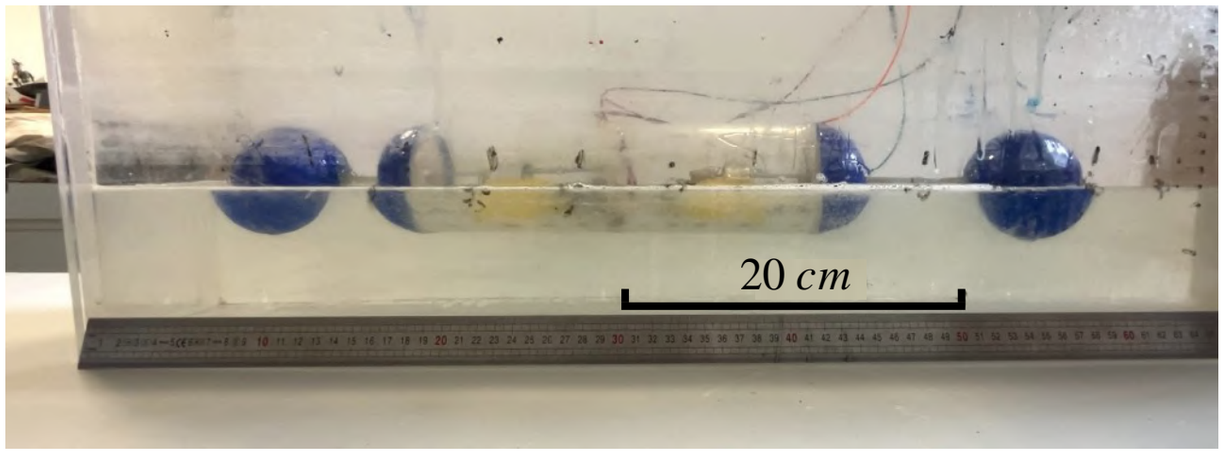
En théorie : Mouvement de translation

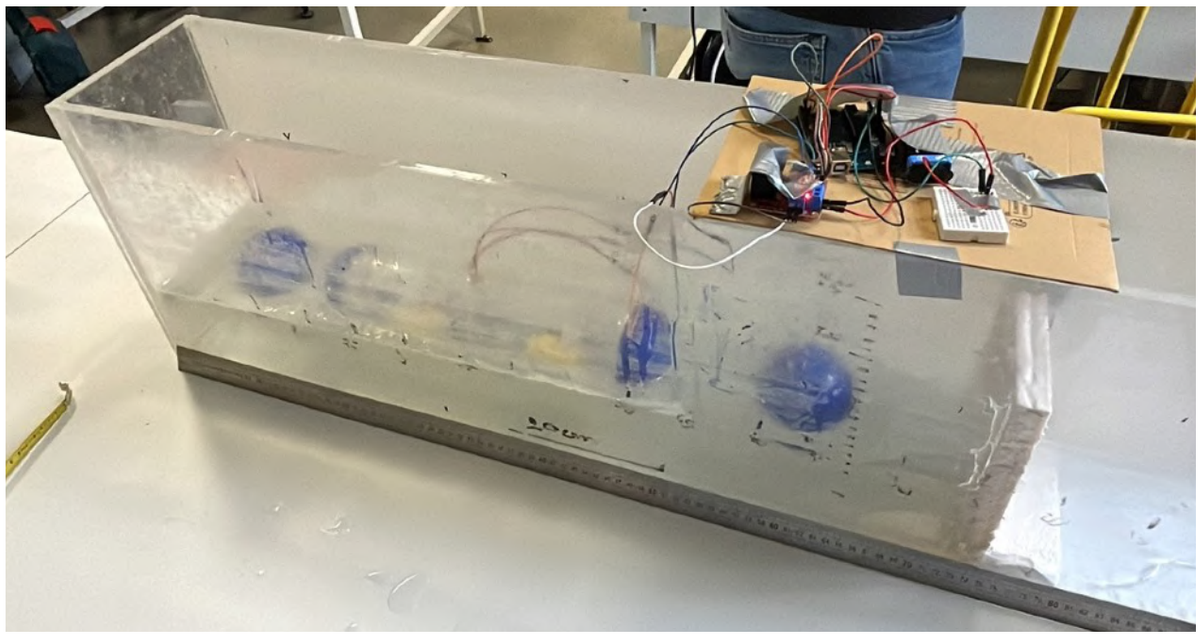
Avancée totale de 23 mm en 8 min pour 30 cycles

Robot 3 - Nageur de Najafi et Golestanian

Expérimentation

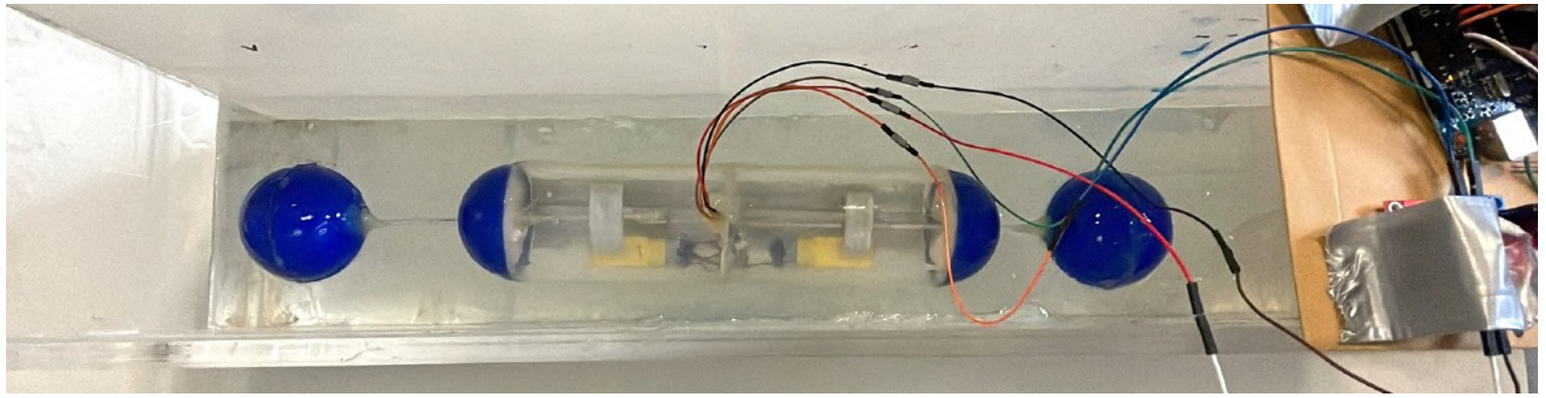
Dispositif

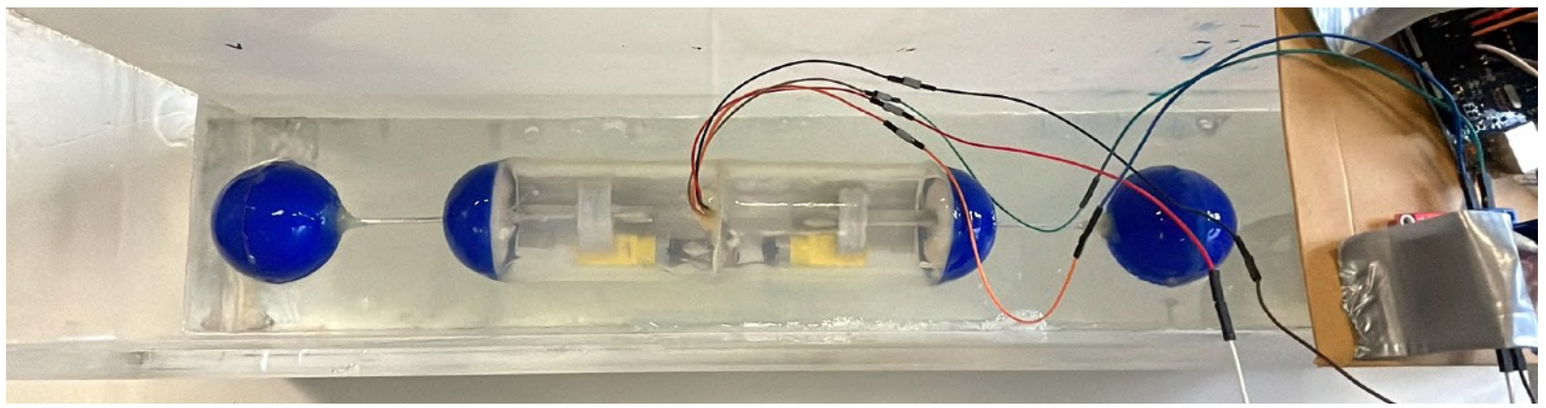




Observation expérimentale :

- Très lent
- Problème : pas entièrement immergé
- Déplacement global de environ $1\ cm$ en $12\ min$
- Contrôlable car translation rectiligne





Robot 3 - Nageur de Najafi et Golestanian

Comparaison théorie et expérimental

En théorie :	Expérimentalement :
- Avancée totale de $23\ mm$ en $8\ min$ - 30 cycles	- Déplacement global de $10\ mm$ environ en $12\ min$ - Environ 45 cycles
Vitesse moyenne : $v_{théo} = 4{,}8 \cdot 10^{-5}\ m/s = 2{,}9\ mm/min$ Nombre de Reynolds : $Re_{théo} = 8{,}7 \cdot 10^{-3} \approx 0{,}01$	Vitesse moyenne : $v_{exp} = 1{,}4 \cdot 10^{-5}\ m/s = 0{,}8\ mm/min$ Nombre de Reynolds : $Re_{exp} = 2{,}5 \cdot 10^{-3} \approx 0{,}001$

Écart relatif :

$$ER = \frac{v_{théo} - v_{exp}}{v_{théo}} \cdot 100 = 71\ \%$$

Conclusion

Annexes

Équation de Navier-Stokes

$$\rho\left[\frac{\partial\vec{v}}{\partial t}+(\vec{v}\cdot\overrightarrow{grad})(\vec{v})\right]=\rho\vec{g}-\overrightarrow{grad}\,(P)+\eta\,\overrightarrow{\Delta}(\vec{v})$$

- $\frac{\partial\vec{v}}{\partial t}$: accélération locale
- $(\vec{v}\cdot\overrightarrow{grad})(\vec{v})$: accélération convective
- $\rho\vec{g}$: pesanteur
- $-\overrightarrow{grad}\,(P)$: résultante volumique des forces de pression
- $\eta\,\overrightarrow{\Delta}(\vec{v})$: force volumique de viscosité

Simplification dans le cas du régime laminaire $Re < 2000$

Équation de Stokes

$$\rho\left[\frac{\partial\vec{v}}{\partial t}+\cancel{(\vec{v}\cdot\overrightarrow{grad})(\vec{v})}\right]=\rho\vec{g}-\overrightarrow{grad}\,(P)+\eta\,\overrightarrow{\Delta}(\vec{v})$$

P : pression

$\vec{v}$: vitesse d'écoulement

ρ : masse volumique du fluide

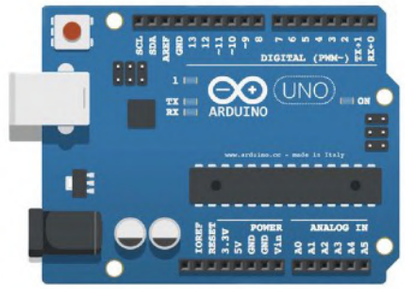
η : viscosité dynamique ($Pa\cdot s$)

$\vec{g}$: pesanteur

Chaîne d'information

Ordinateur

Carte Arduino UNO



Ordres

Palettes sans mouvement

Chaîne d'énergie

Stocker	Alimenter	Distribuer	Convertir	Transmettre et agir
Pile 9V 	**Adaptateur et fils** 	**Hacheur** (module L298N) 	**Moteur** (courant continu) 	**Palettes** 

Agitation des palettes

Calcul de la viscosité

Principe fondamental de la dynamique

$$m\frac{d\vec{v}}{dt} = \vec{P} + \vec{\Pi} + \vec{F_{tr}}$$

$$\vec{P} = m\vec{g}$$

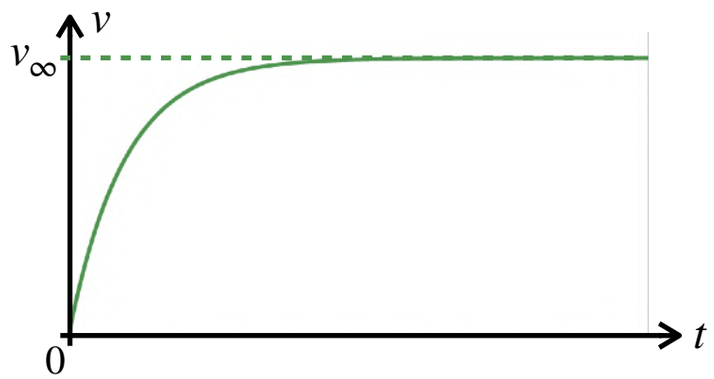
$$\vec{F_{tr}} = -6\pi\eta r\vec{v}$$

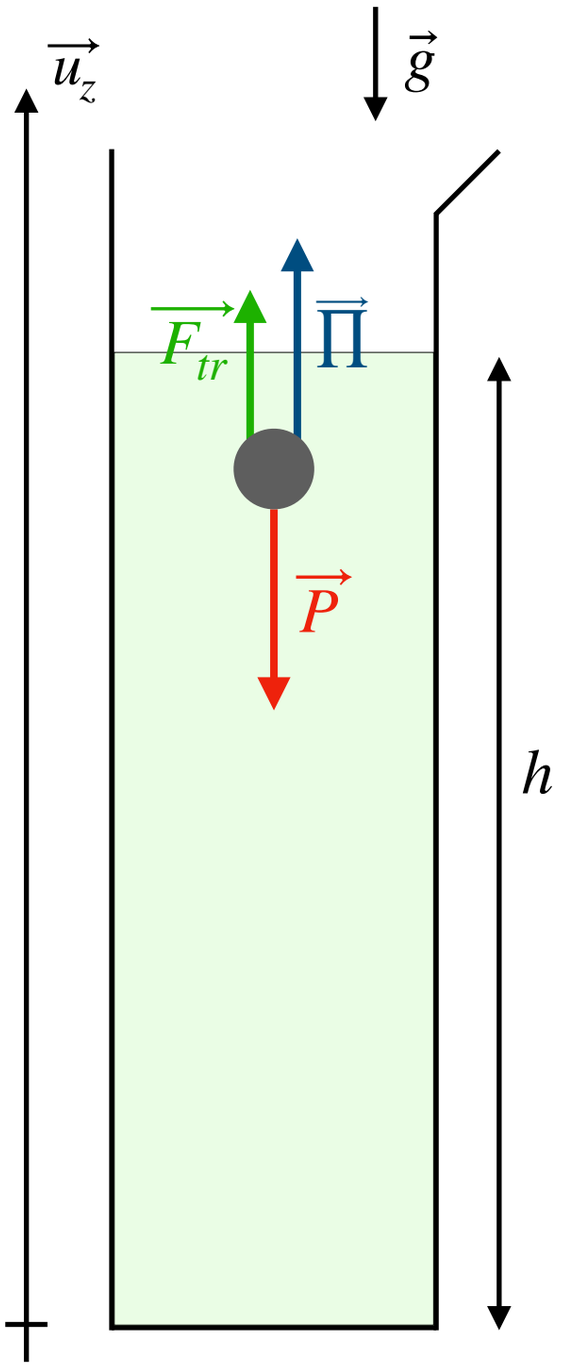
$$\vec{\Pi} = -\rho_f V_b \vec{g}$$

$$\rho_b V_b \frac{d\vec{v}}{dt} = \rho_b V_b \vec{g} - \rho_f V_b \vec{g} - 6\pi\eta r\vec{v}$$

$$\rho_b V_b \frac{dv}{dt} = -\rho_b V_b g + \rho_f V_b g + 6\pi\eta r v$$

$$\vec{v} = -v\,\vec{u_z} \text{ avec } v > 0$$





Au bout de quelques instants : $\frac{dv}{dt} = 0$ et $v = v_\infty = cte$

$$6\pi\eta r v_\infty = (\rho_b - \rho_f) V_b g$$

$$\eta = \frac{(\rho_b - \rho_f) V_b g}{6\pi r v_\infty} = \frac{(\rho_b - \rho_f) 4\pi r^3 g}{3 \cdot 6\pi r v_\infty}$$

$$\eta = \frac{2(\rho_b - \rho_f) r^2 g}{9 v_\infty}$$

v : vitesse de la bille

ρ_f : masse volumique du fluide

ρ_b : masse volumique de la bille

m : masse de la bille

V_b : volume de la bille

g : pesanteur

r : rayon de la bille

Annexe 4 : Calcul de l'incertitude sur la mesure de la viscosité

Principe du calcul - Incertitudes de type B

$$\eta = \frac{2\,(\rho_b - \rho_f)\,r^2\,g}{9\,v_\infty}$$

- Calcul des incertitudes sur chacune des grandeurs qui interviennent dans η
- Utilisation des formules pour l'incertitude d'une somme, d'un produit et d'un quotient : $u(x+y) = \sqrt{u(x)^2 + u(y)^2}$ $\quad u(x \cdot y) = \sqrt{y^2 u(x)^2 + x^2 u(y)^2}$ $\quad u\left(\frac{x}{y}\right) = \sqrt{\frac{u(x)^2}{y^2} + \frac{x^2 u(y)^2}{y^4}}$

Incertitude sur la masse volumique

- Méthode de mesure : burette graduée de volume $100\ ml$ et mesure de la masse de fluide

$$\rho = \frac{m}{V}$$

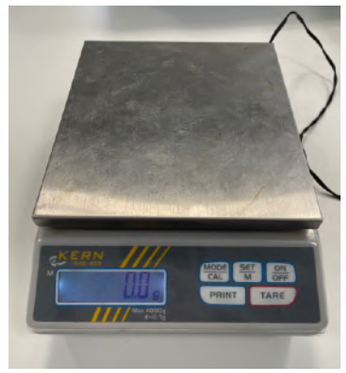
$\Delta_V = 1\ ml$ donc $u(V) = 0{,}58\ ml = 0{,}58 \cdot 10^{-3}\ l$

$\Delta_m = 0{,}1\ g$ donc $u(m) = 0{,}058\ g = 5{,}8 \cdot 10^{-5}\ kg$

Mesures pour le liquide vaisselle : $V = 100{,}00 \pm 0{,}58\ ml$ $\quad m = 102{,}700 \pm 0{,}058\ g$

D'où $\rho_f = 1027{,}00 \pm 5{,}98\ g/l$

De même $\rho_b = 1190{,}00 \pm 7{,}89\ kg/m^3$



Incertitude sur r

$\Delta_r = 0{,}05\ mm$ donc $u(r) = 0{,}03\ mm$

Mesures : $r = 4{,}10 \pm 0{,}03\ mm$ $\quad r^2 = 16{,}81 \pm 0{,}17\ mm^2$

Pied à coulisse à vernier

Incertitude sur position z et vitesse v_∞ sur logiciel de pointage

$\Delta_z = 0{,}5\ mm$ donc $u(z) = 0{,}3\ mm$

$u(dz) = 0{,}42\ mm$ $\quad u(v_\infty) = 1{,}68\ mm/s$ $\quad v_\infty = \frac{dz}{dt}$

Mesures : $dz = 5{,}10 \pm 0{,}42\ mm$ $\quad dt = 0{,}25\ s$

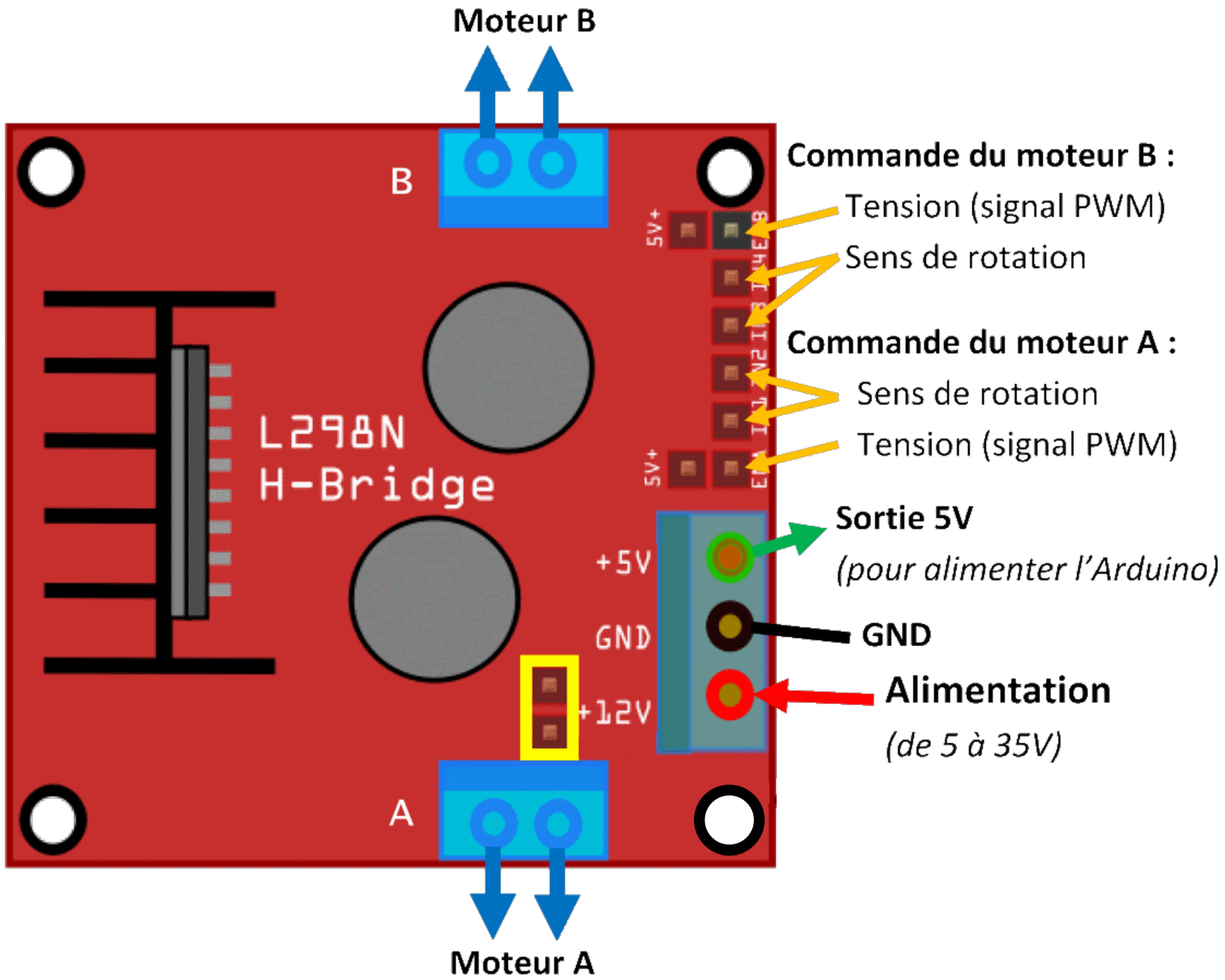
$v_\infty = 20{,}40 \pm 1{,}68\ mm/s$

Incertitude sur la viscosité

$$\eta = 0{,}29 \pm 0{,}03\ Pl$$

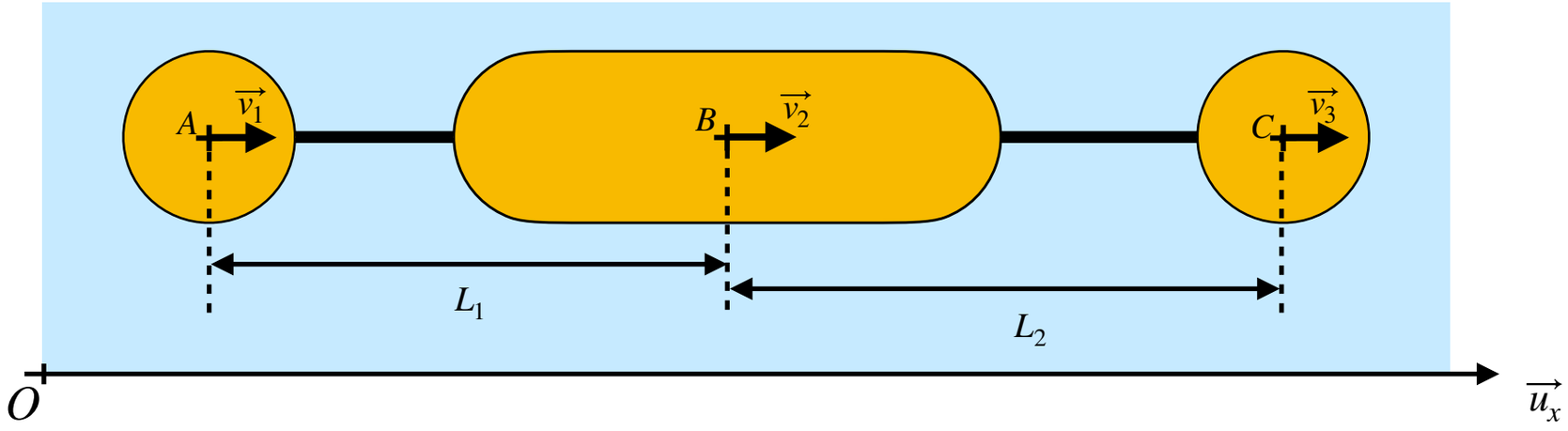
```
from math import *

def incertitude_produit(x,ux,y,uy):
    return (x*y, sqrt((y*ux)**2+(x*uy)**2))

def incertitude_quotient(x,ux,y,uy):
    return (x/y, (x/y)*sqrt((ux/x)**2+(uy/y)**2))

def incertitude_somme(x,ux,y,uy):
    return (x+y,sqrt(ux**2+uy**2))
```

Annexe 5 : Fonctionnement du hacheur (module L298N) pour deux moteurs



https://arduino.blaisepascal.fr/pont-en-h-l298n/

Annexe 6 : Simulation numérique en Python, nageur de Najafi et Golestanian



Utilisation des tenseurs de mobilité M_{ij}

Prise en compte de la mobilité des sphères individuelles (formule de Stokes)

Prise en compte d'un terme d'interaction (qui caractérise comment la force qui s'exerce sur une sphère influence la vitesse d'une autre)

Formulation matricielle des mobilités et interactions entre sphères

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix}$$

M_{ii} : tenseurs de mobilité des sphères individuelles

M_{ij} : tenseurs d'interaction

$$M_{ii} = -\frac{1}{6\pi\eta r} \quad M_{ij} = -\frac{1}{3\pi\eta d_{ij}}$$ avec d_{ij} la distance entre sphères i et j

Système équivalent :

$$\begin{cases} v_1 = -\frac{1}{6\pi\eta r}F_1 - \frac{1}{3\pi\eta L_1}F_2 - \frac{1}{3\pi\eta(L_1+L_2)}F_3 \\ v_2 = -\frac{1}{3\pi\eta L_1}F_1 - \frac{1}{6\pi\eta r}F_2 - \frac{1}{3\pi\eta L_2}F_3 \\ v_2 = -\frac{1}{3\pi\eta(L_1+L_2)}F_1 - \frac{1}{3\pi\eta L_2}F_2 - \frac{1}{6\pi\eta r}F_3 \end{cases}$$

Ajout de 3 équations pour trouver les 6 inconnues

$$\begin{cases} F_1 + F_2 + F_3 = 0 & \text{Inertie totale négligée} \\ -v_1 + v_2 = \dot{L}_1 & \text{Vitesse d'élongation de la tige gauche} \\ -v_2 + v_3 = \dot{L}_2 & \text{Vitesse d'élongation de la tige droite} \end{cases}$$

Problème matriciel final :

$$\begin{pmatrix} 1 & 0 & 0 & \frac{1}{6\pi\eta r} & \frac{1}{3\pi\eta L_1} & \frac{1}{3\pi\eta(L_1+L_2)} \\ 0 & 1 & 0 & \frac{1}{3\pi\eta L_1} & \frac{1}{6\pi\eta r} & \frac{1}{3\pi\eta L_2} \\ 0 & 0 & 1 & \frac{1}{3\pi\eta(L_1+L_2)} & \frac{1}{3\pi\eta L_2} & \frac{1}{6\pi\eta r} \\ 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ F_1 \\ F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dot{L}_1 \\ \dot{L}_2 \end{pmatrix}$$

Annexe 7 : Programme simulation numérique en Python

```
def matriceA (L1,L2):
    matA = numpy.array([[1,0,0,1/(6*pi*eta*r),1/(3*pi*eta*L1),1/(3*pi*eta*(L1+L2))],
                        [0,1,0,1/(3*pi*eta*L1),1/(6*pi*eta*r),1/(3*pi*eta*L2)],
                        [0,0,1,1/(3*pi*eta*(L1+L2)),1/(3*pi*eta*L2),1/(6*pi*eta*r)],
                        [0,0,0,1,1,1],
                        [-1,1,0,0,0,0],
                        [0,-1,1,0,0,0]])
    return matA
```

```
def matriceB (dL1,dL2):
    matB = numpy.array([[0],[0],[0],[dL1],[dL2]])
    return matB
```

```
def phase_1(L1,L1max,dL1,L2,L2max,dL2,a,b,c):  #a, b et c les abscisses des points A, B et C
    tab_a=[]
    tab_b=[]
    tab_c=[]
    for j in range (N):
        matA = matriceA(L1,L2)
        matB = matriceB(-dL1,0)
        X=npl.solve(matA,matB)
        L1 += dL1*delta_t
        v1 = X[0,0]
        v2 = X[1,0]
        v3 = X[2,0]
        a += v1*delta_t
        b += v2*delta_t
        c += v3*delta_t
        tab_a += [a]
        tab_b += [b]
        tab_c += [c]
    return tab_a, tab_b, tab_c, L1, L2
```

```
def phase_2(L1,L1max,dL1,L2,L2max,dL2,a,b,c):
    tab_a=[]
    tab_b=[]
    tab_c=[]
    for j in range (N):
        matA = matriceA(L1,L2)
        matB = matriceB(0,-dL2)
        X=npl.solve(matA,matB)
        L2 += dL2*delta_t
        v1 = X[0,0]
        v2 = X[1,0]
        v3 = X[2,0]
        a += v1*delta_t
        b += v2*delta_t
        c += v3*delta_t
        tab_a += [a]
        tab_b += [b]
        tab_c += [c]
    return tab_a, tab_b, tab_c, L1, L2
```

```
def phase_3(L1,L1max,dL1,L2,L2max,dL2,a,b,c):
    tab_a=[]
    tab_b=[]
    tab_c=[]
    for j in range (N):
        matA = matriceA(L1,L2)
        matB = matriceB(dL1,0)
        X=npl.solve(matA,matB)
        L1 -= dL1*delta_t
        v1 = X[0,0]
        v2 = X[1,0]
        v3 = X[2,0]
        a += v1*delta_t
        b += v2*delta_t
        c += v3*delta_t
        tab_a += [a]
        tab_b += [b]
        tab_c += [c]
    return tab_a, tab_b, tab_c, L1, L2
```

```
def phase_4(L1,L1max,dL1,L2,L2max,dL2,a,b,c):
    tab_a=[]
    tab_b=[]
    tab_c=[]
    for j in range (N):
        matA = matriceA(L1,L2)
        matB = matriceB(0,dL2)
        X=npl.solve(matA,matB)
        L2 -= dL2*delta_t
        v1 = X[0,0]
        v2 = X[1,0]
        v3 = X[2,0]
        a += v1*delta_t
        b += v2*delta_t
        c += v3*delta_t
        tab_a += [a]
        tab_b += [b]
        tab_c += [c]
    return tab_a, tab_b, tab_c, L1, L2
```

```
def modélisation (L1min, L1max, dL1, L2min, L2max, dL2, a1, a2, a3, b, delta_t, N, T):
    L1 = L1min
    L2 = L2min
    a = b - L1
    c = b + L1
    tab1 = []
    tab2 = []
    tab3 = []
    tab_a = []
    tab_b = []
    tab_c = []
    tab_t = []
    for i in range (M): # M cycles
        tab1, tab2, tab3, L1 , L2 = phase_1(L1, L1max, dL1, L2, L2max, dL2, a, b, c)
        tab_a += tab1
        tab_b += tab2
        tab_c += tab3
        a = tab_a[-1]
        b = tab_b[-1]
        c = tab_c[-1]
        tab1, tab2, tab3, L1, L2 = phase_2(L1, L1max, dL1, L2, L2max, dL2, a , b ,c)
        tab_a += tab1
        tab_b += tab2
        tab_c += tab3
        a = tab_a[-1]
        b = tab_b[-1]
        c = tab_c[-1]
        tab1, tab2, tab3, L1, L2 = phase_3(L1, L1max, dL1, L2, L2max, dL2, a , b ,c)
        tab_a += tab1
        tab_b += tab2
        tab_c += tab3
        a = tab_a[-1]
        b = tab_b[-1]
        c = tab_c[-1]
        tab1, tab2, tab3, L1, L2 = phase_3(L1, L1max, dL1, L2, L2max, dL2, a , b ,c)
        tab_a += tab1
        tab_b += tab2
        tab_c += tab3
        a = tab_a[-1]
        b = tab_b[-1]
        c = tab_c[-1]
        tab_t+= [i]
    print("avancée totale", b, 'm')
    x = [i*delta_t for i in range (len(tab_b))]
    y = [i for i in tab_b]
    plt.plot(x,y)
    plt.title('position de la partie centrale en fonction du temps')
    plt.xlabel('temps en s')
    plt.ylabel('position en mm')
    plt.show()
    plt.close()
```

```
modélisation(L1min, L1max, dL1, L2min, L2max, dL2, a1, a2, a3, b, delta_t, N, T)
```

Annexe 8 : Programme Arduino robot 1

```
// Moteur à courant continu avec hacheur L298N à deux sens de rotations .
// Alternance du sens de rotation du moteur

// Variables :
int enA = 10;
// variable qui gère la vitesse du moteur.
int in1 = 9;
// variable qui gère un des sens de rotation du moteur.
int in2 = 8;
// variable qui gère l'autre sens de rotation du moteur.

void setup () {
pinMode (enA,OUTPUT);     // Les variables utilisées sont toutes des sorties
pinMode (in1, OUTPUT) ;
pinMode (in2, OUTPUT) ;


}
void loop()  {
digitalWrite (in1, HIGH) ;  // Arduino envoie un courant dans la broche in1 (PIN 9). Le moteur tourne dans le sens de rotation de in1.
digitalWrite (in2, LOW) ;
analogWrite (enA, 255) ;    // vitesse du moteur, valeur maximale : 255 (échélle comprise entre 0 et 255)
delay (200) ;               // durée de la phase (200 ms = 0,2 s)

digitalWrite (in1, LOW) ;   // le moteur est à l'arrêt pendant 10 ms
digitalWrite (in2, LOW) ;
delay (10) ;

digitalWrite (in1, LOW) ;
digitalWrite (in2, HIGH) ;  // Arduino envoie un courant dans la broche in2 (PIN 8). Le moteur tourne dans le sens de rotation de in2.
analogWrite (enA, 255);
delay (200) ;

}
```

Annexe 9 : Programme Arduino robots 2 et 3

```
// Avec 2 moteurs

int enA = 10;
int enB = 5;

int in1 = 9;
int in2 = 8;
int in3 = 7;
int in4 = 6;


void setup () {
pinMode (enA,OUTPUT);
pinMode (enB,OUTPUT);

pinMode (in1, OUTPUT) ;
pinMode (in2, OUTPUT) ;
pinMode (in3, OUTPUT) ;
pinMode (in4, OUTPUT) ;
}

void loop()  {
digitalWrite (in1, HIGH) ;
digitalWrite (in2, LOW) ;
digitalWrite (in3, LOW) ;
digitalWrite (in4, LOW) ;
analogWrite (enA, 30) ;
analogWrite (enB, 0) ;
delay (200) ;


digitalWrite (in1, LOW) ;
digitalWrite (in2, LOW) ;
digitalWrite (in3, LOW) ;
digitalWrite (in4, LOW) ;
delay (200) ;


digitalWrite (in1, LOW) ;
digitalWrite (in2, LOW) ;
digitalWrite (in3, HIGH) ;
digitalWrite (in4, LOW) ;
analogWrite (enA, 0);
analogWrite (enB, 30) ;
delay (200) ;

digitalWrite (in1, LOW) ;
digitalWrite (in2, LOW) ;
digitalWrite (in3, LOW) ;
digitalWrite (in4, LOW) ;
delay (200) ;


digitalWrite (in1, LOW) ;
digitalWrite (in2, HIGH) ;
digitalWrite (in3, LOW) ;
digitalWrite (in4, LOW) ;
analogWrite (enA, 30);
analogWrite (enB, 0) ;
delay (200) ;

digitalWrite (in1, LOW) ;
digitalWrite (in2, LOW) ;
digitalWrite (in3, LOW) ;
digitalWrite (in4, LOW) ;
delay (200) ;

digitalWrite (in1, LOW) ;
digitalWrite (in2, LOW) ;
digitalWrite (in3, LOW) ;
digitalWrite (in4, HIGH) ;
analogWrite (enA, 0) ;
analogWrite (enB, 30) ;
delay (200) ;

digitalWrite (in1, LOW) ;
digitalWrite (in2, LOW) ;
digitalWrite (in3, LOW) ;
digitalWrite (in4, LOW) ;
delay (200) ;

}
```