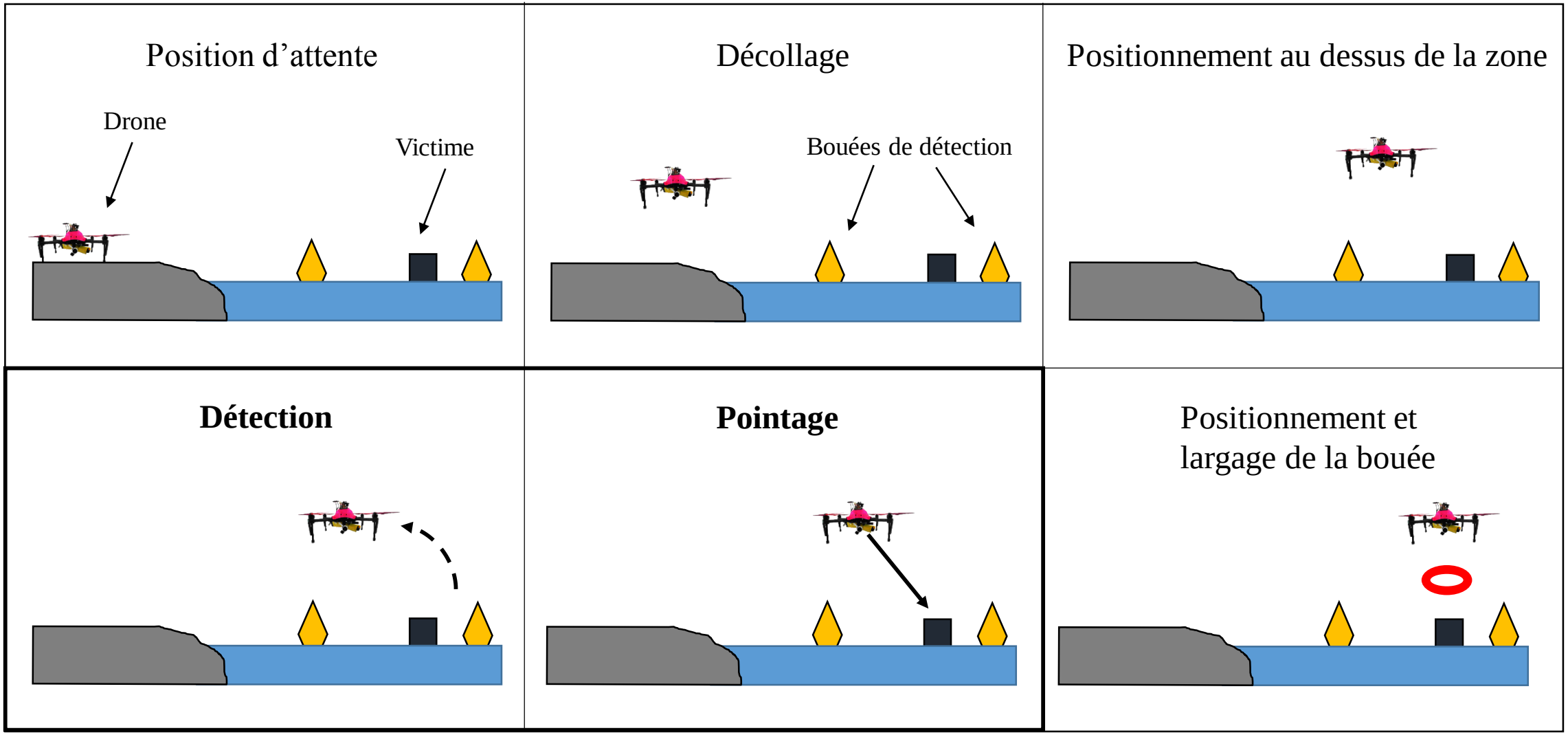
Localisation et suivi de la source d’un signal vibratoire

Introduction et cadre de l’étude
I) Détection d’un signal vibratoire
II) Pointage de la position
Conclusion

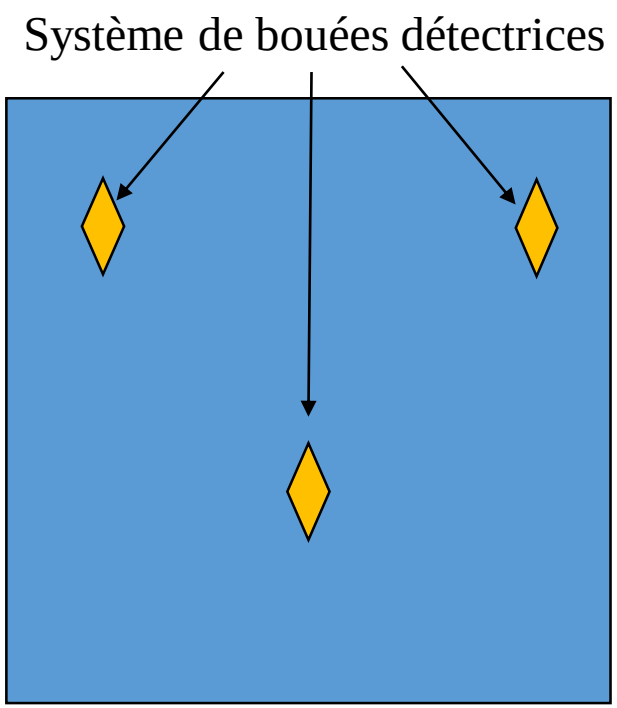
Problématique : *Comment localiser la source d’un signal et la pointer avec une précision optimale ?*

Introduction et cadre de l'étude : Schéma de principe

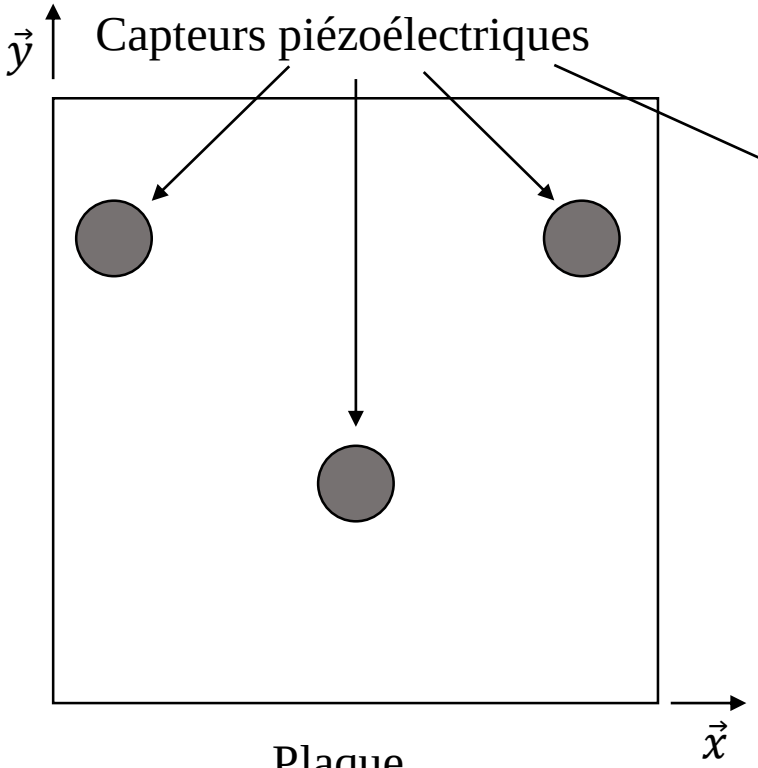


Cadre de l'étude

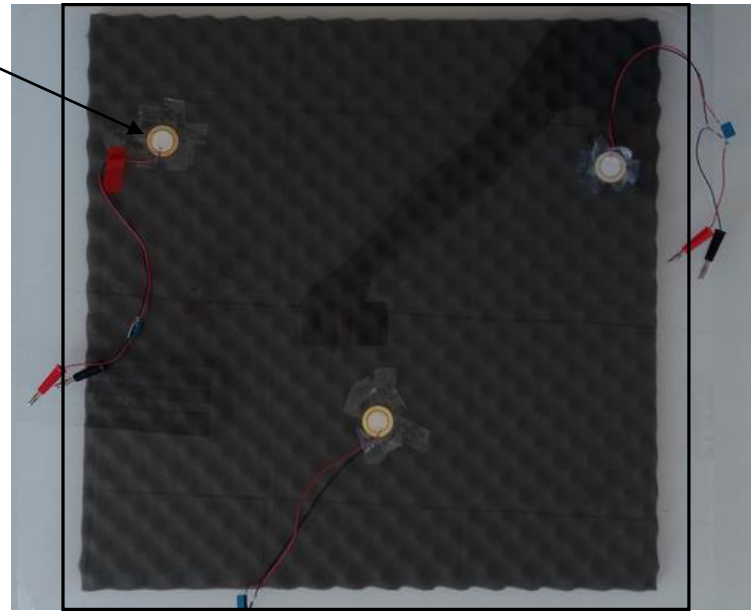
I) Détection d'un signal vibratoire : Modélisation



Zone de baignade

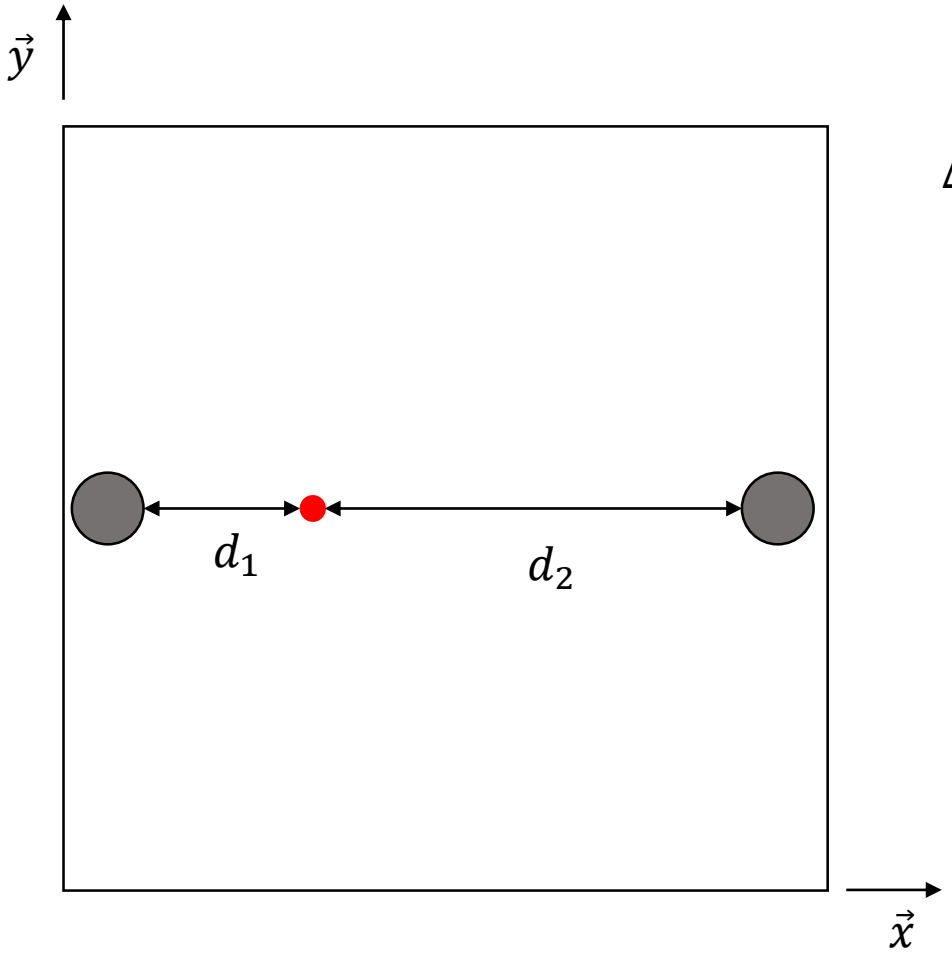


Plaque



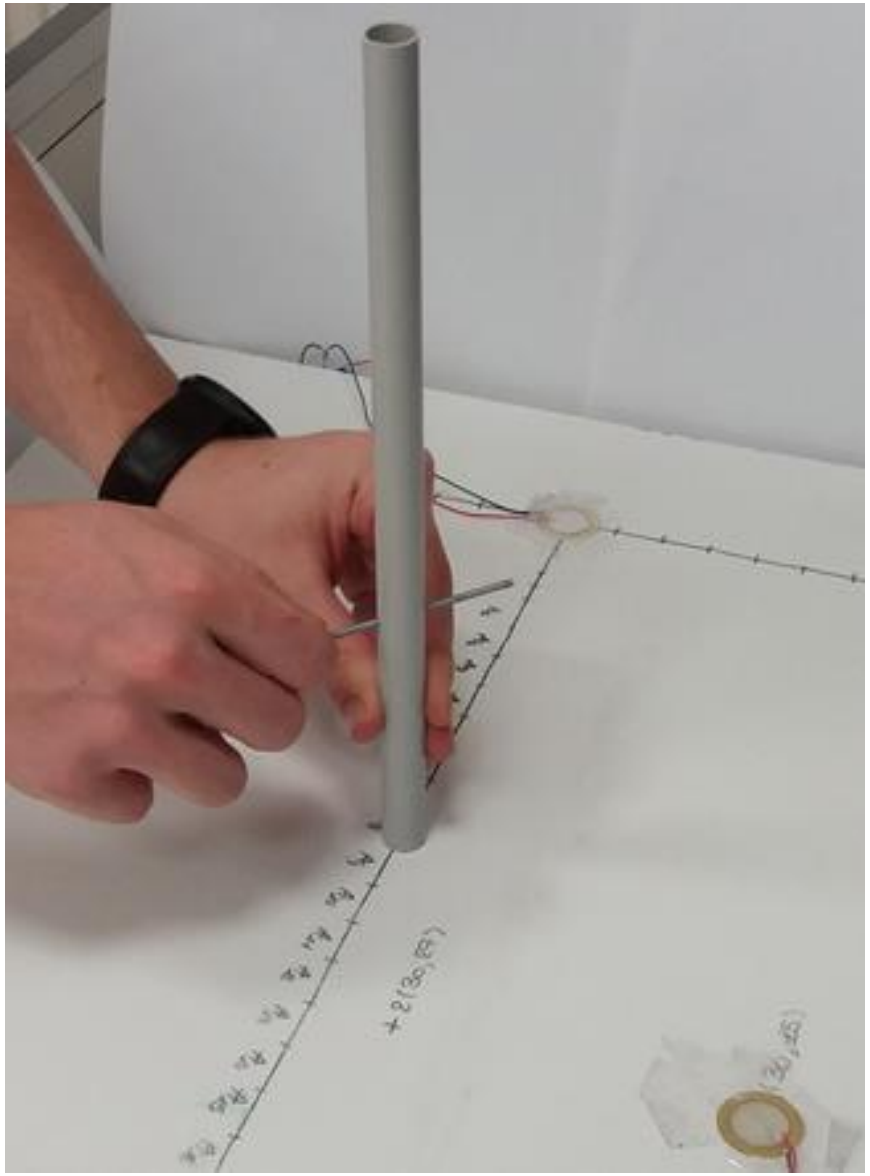
Signal de détresse $\xrightarrow{\textit{Traitement}}$ Localisation

I) Détection d'un signal vibratoire : Vitesse dans le solide (étude 1D)

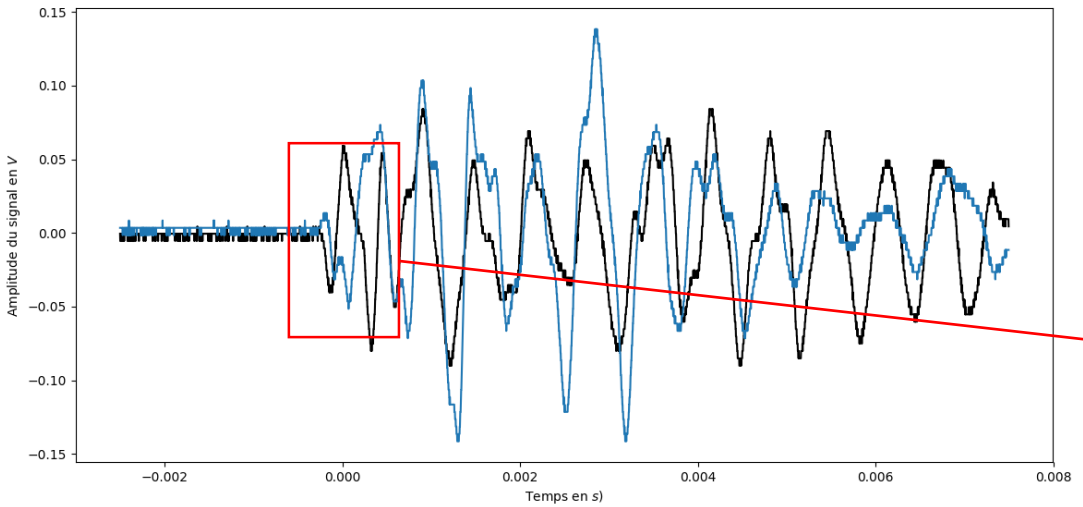


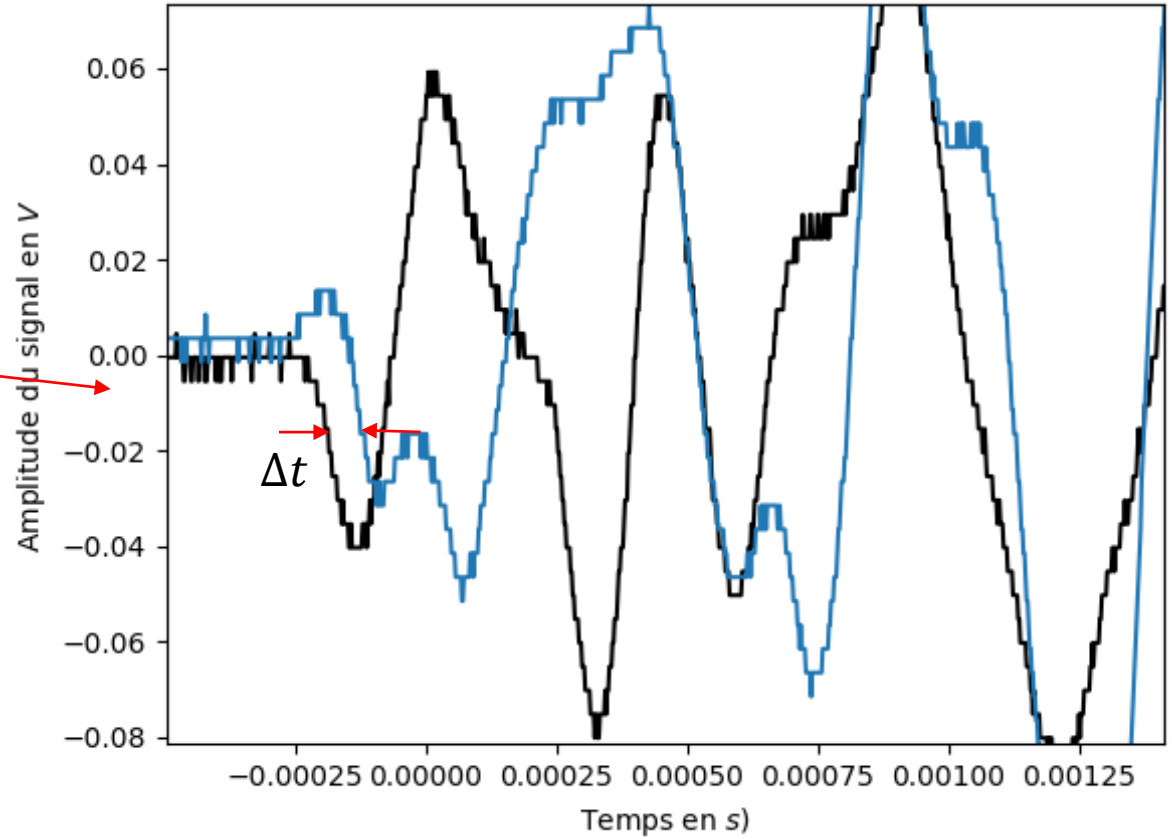
$$\Delta x = d_1 - d_2$$

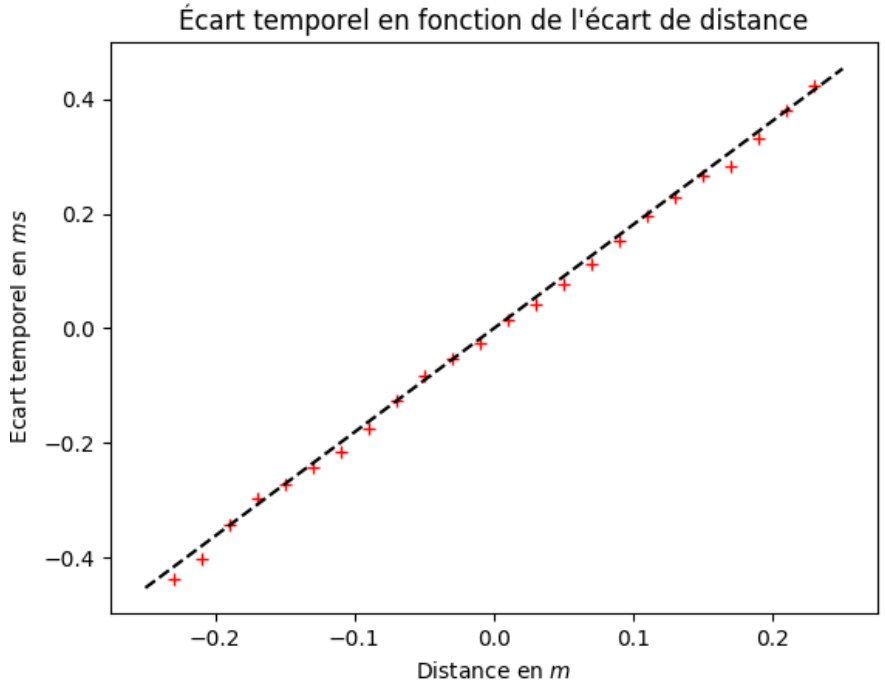
$$V = \frac{\Delta x}{\Delta t}$$



I) Détection d'un signal vibratoire : Vitesse dans le solide (étude 1D)



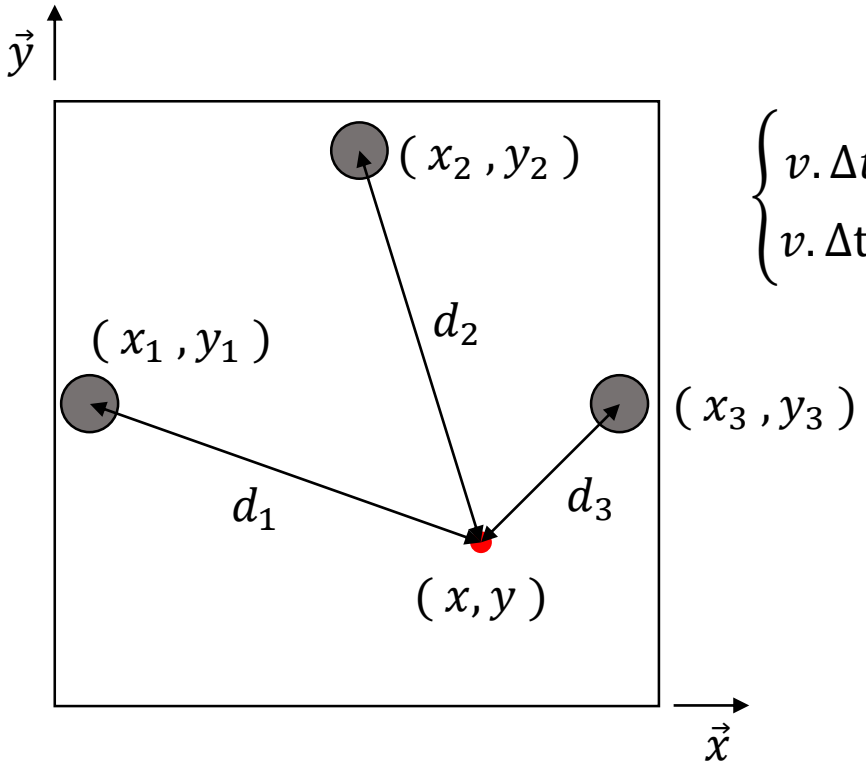




$$V = \frac{\Delta x}{\Delta t}$$

	Bois	**Plexiglas**
Vitesse en m.s^{-1}	552 ± 19	368 ± 12

I) Détection d'un signal vibratoire : Étude en 2D

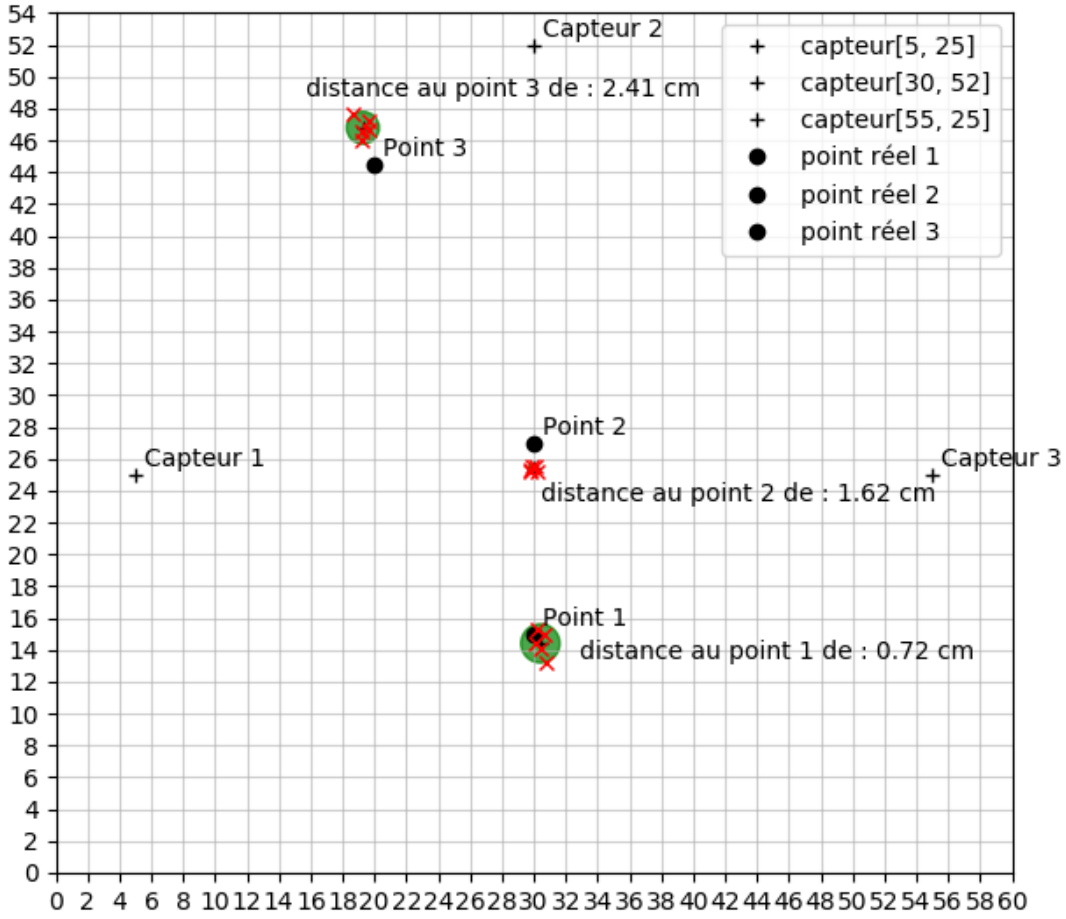


$$\begin{cases} v.\Delta t_{21} = \Delta d_{21} = d_2 - d_1 = \sqrt{(x-x_2)^2 + (y-y_2)^2} - \sqrt{(x-x_1)^2 + (y-y_1)^2} \\ v.\Delta t_{31} = \Delta d_{31} = d_3 - d_1 = \sqrt{(x-x_3)^2 + (y-y_3)^2} - \sqrt{(x-x_1)^2 + (y-y_1)^2} \end{cases}$$

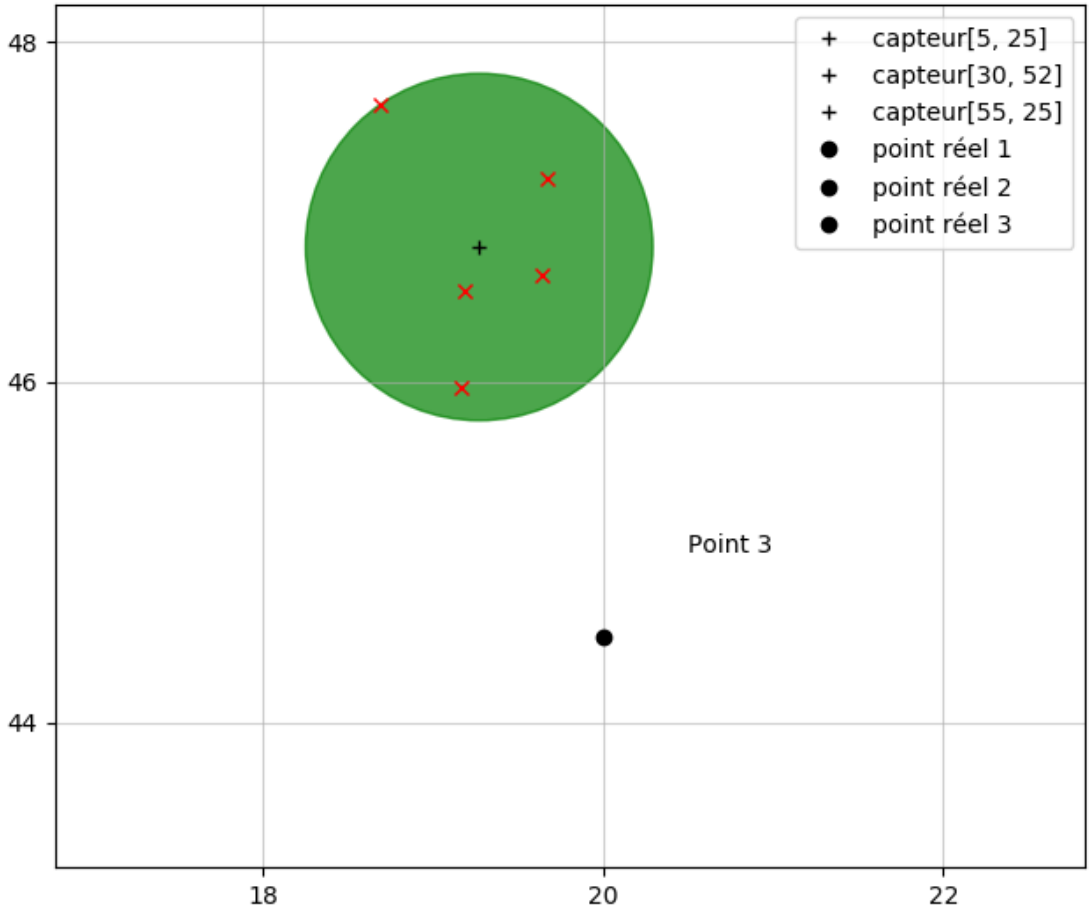
Programme Python: Résolution numérique
Utilisation de la fonction **scipy.optimize.fsolve()**

$[x, y]$

I) Détection d'un signal vibratoire : Résultats et limites de l'étude

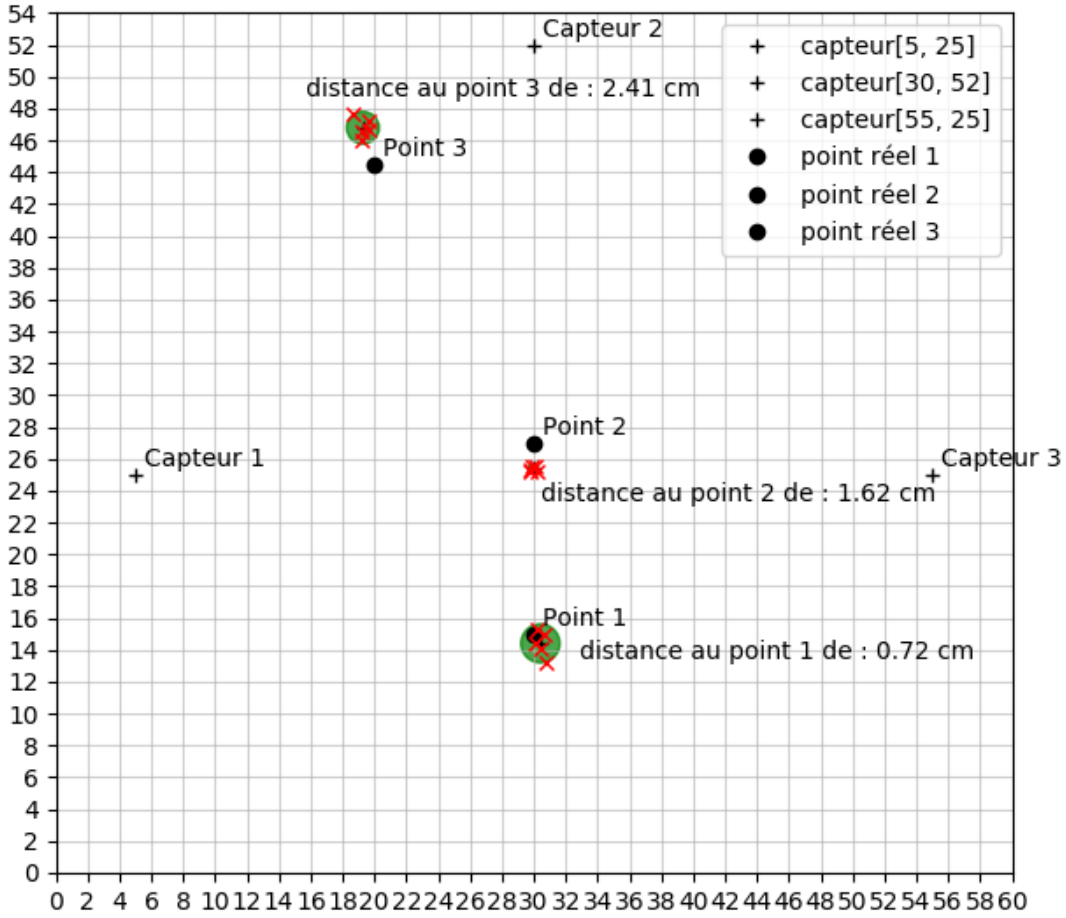


Résultat détection pour le bois

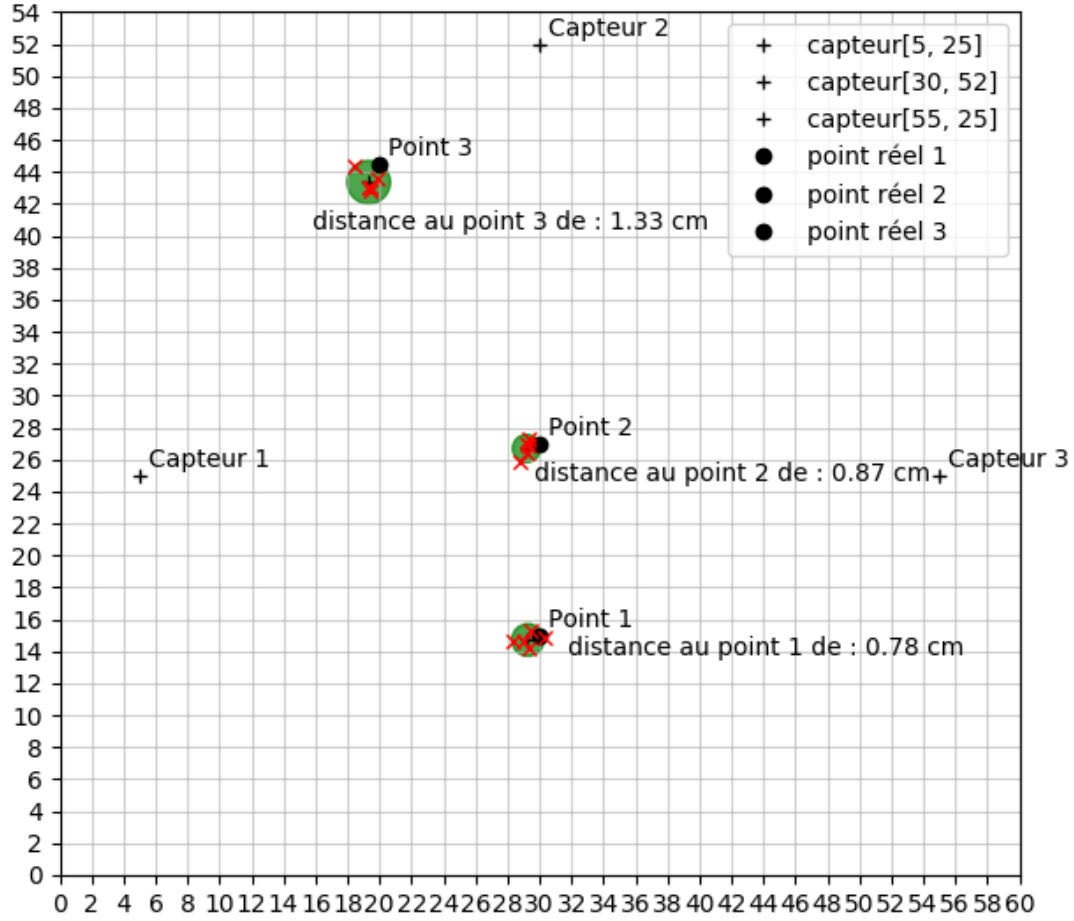


Résultat détection pour le point 3

I) Détection d'un signal vibratoire : Résultats et limites de l'étude

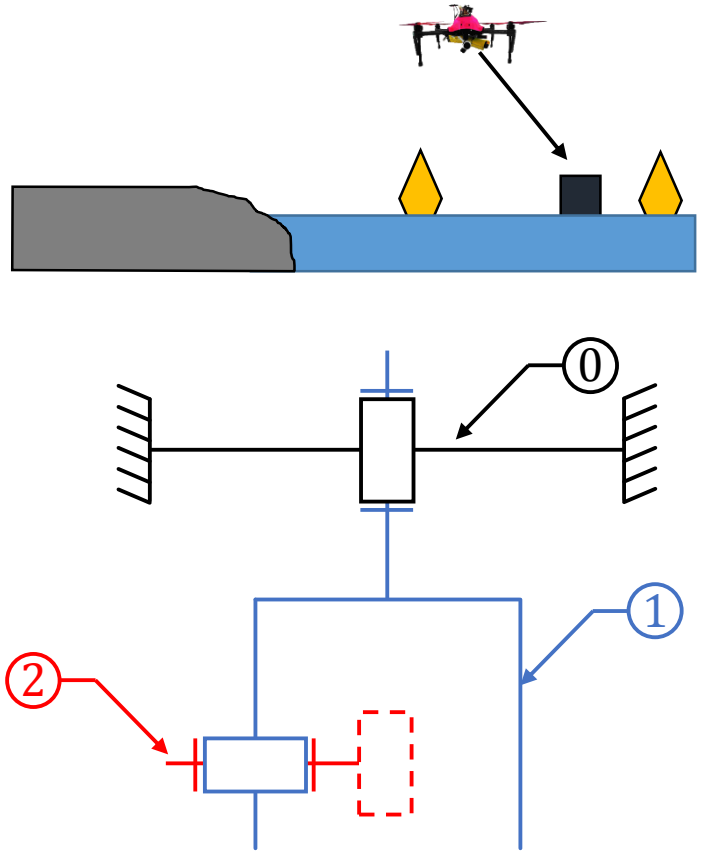


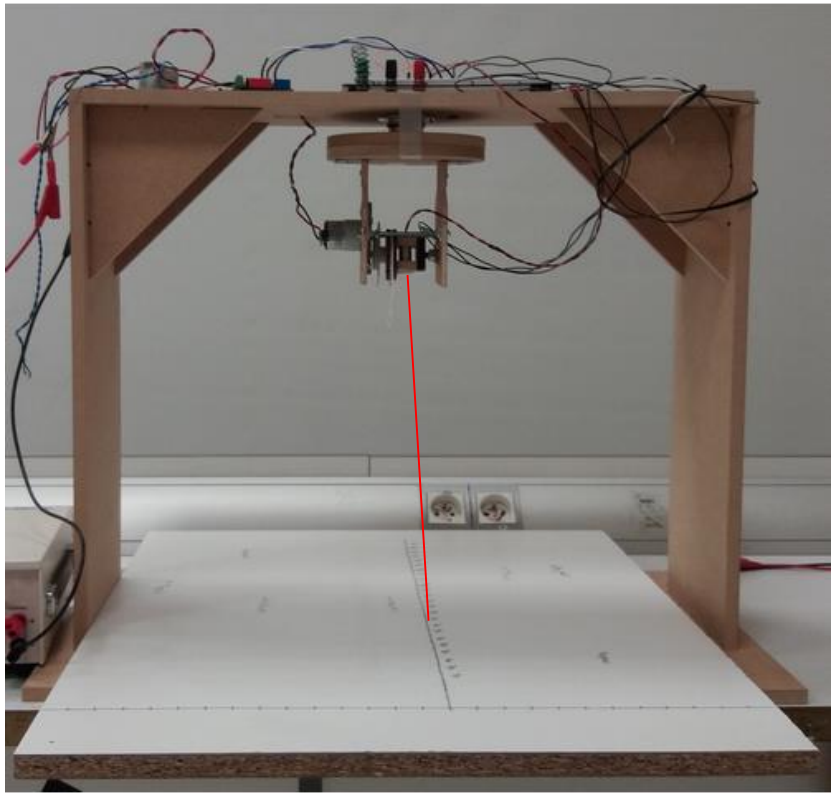
Résultat détection pour le bois

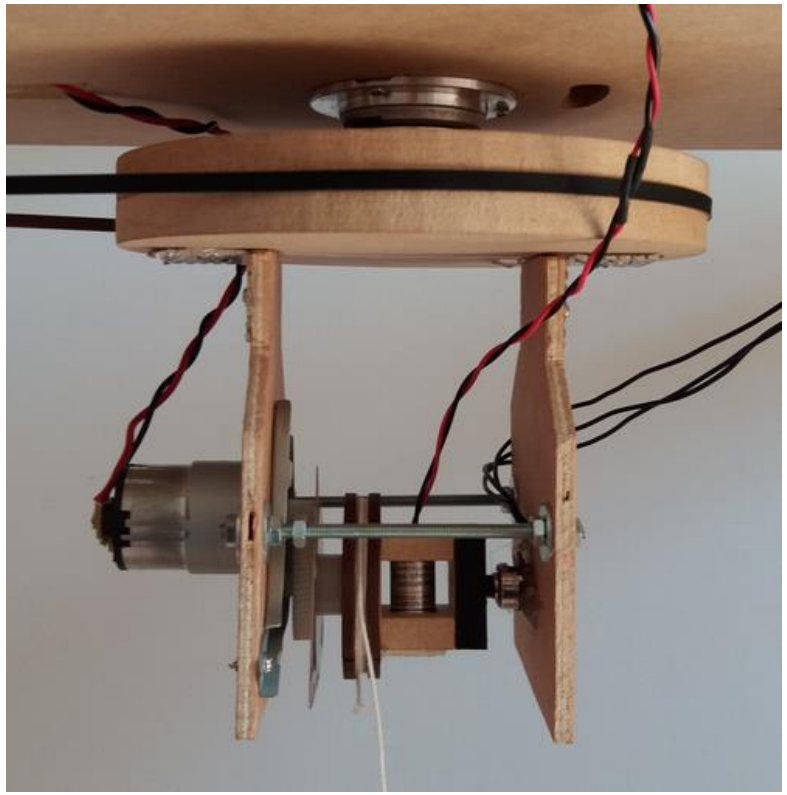


Résultat détection pour le Plexiglas

Pointage de la position: Modélisation

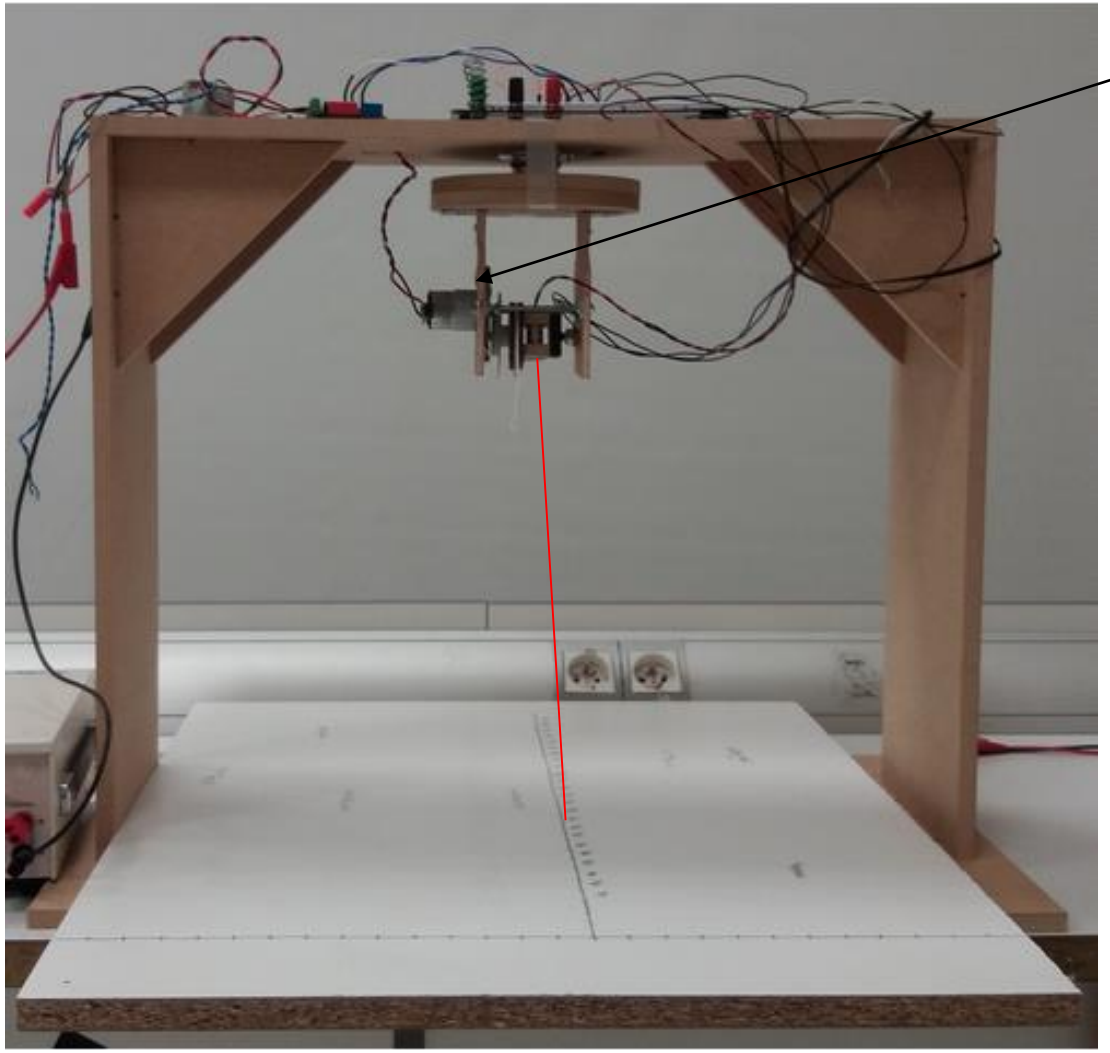


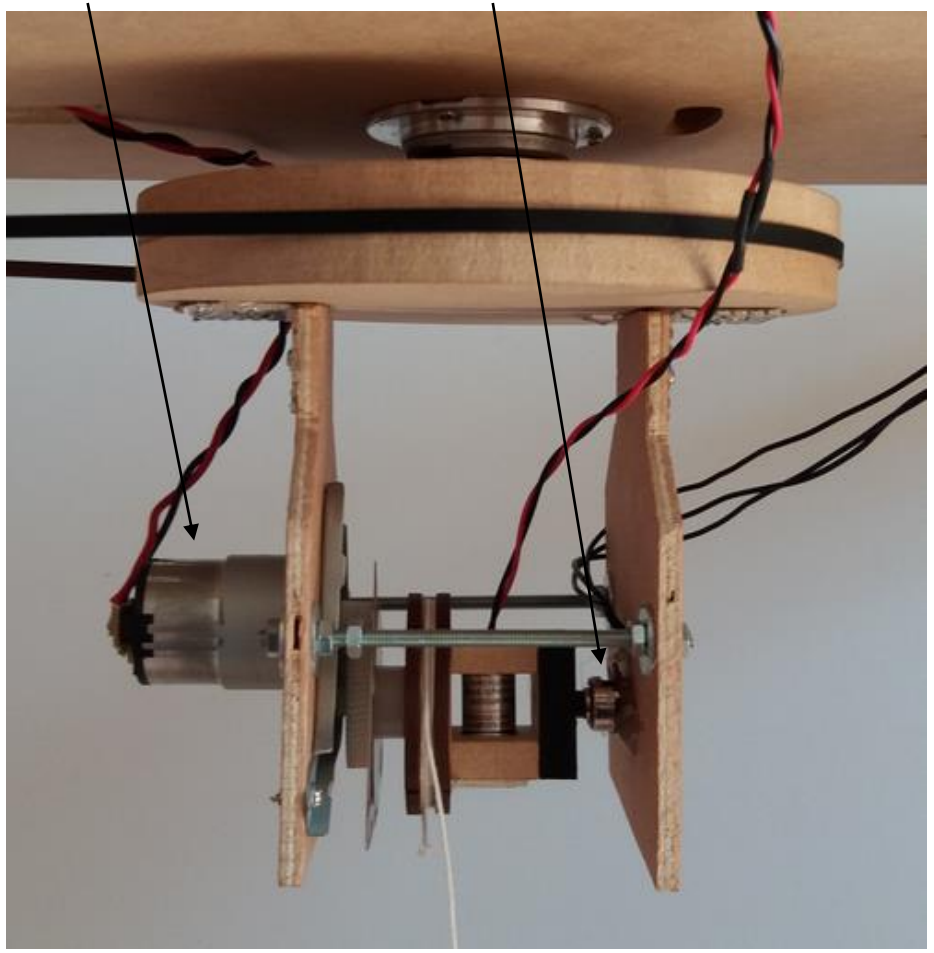




Signal de détresse —*Traitement*→ Localisation ⇢ Position calculée —*Asservissement*→ Point de largage

II) Pointage de la position : Principe





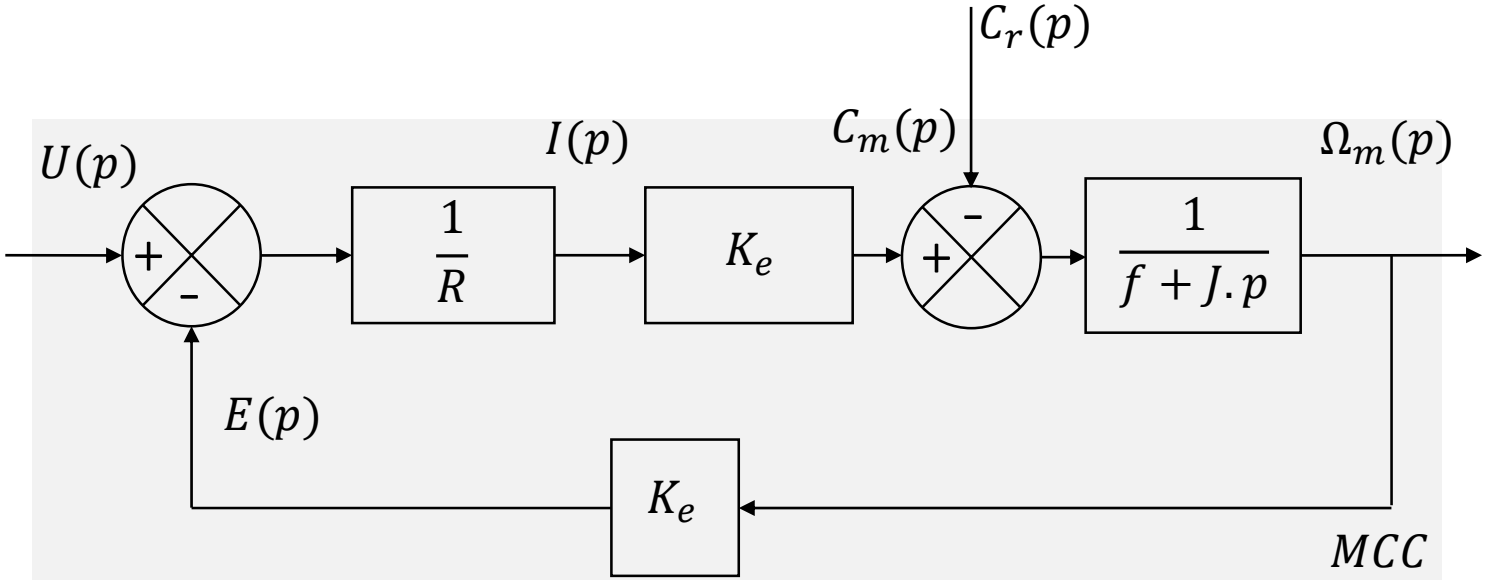
II) Pointage de la position : Détermination des caractéristiques du moteur

Équations du moteur à courant continu

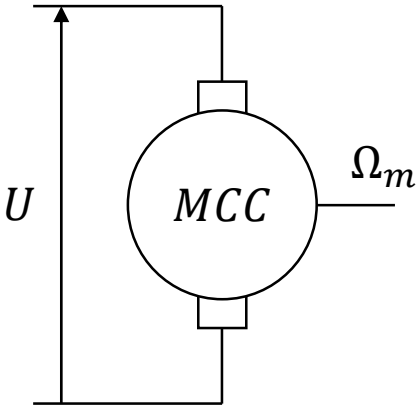
Négligé

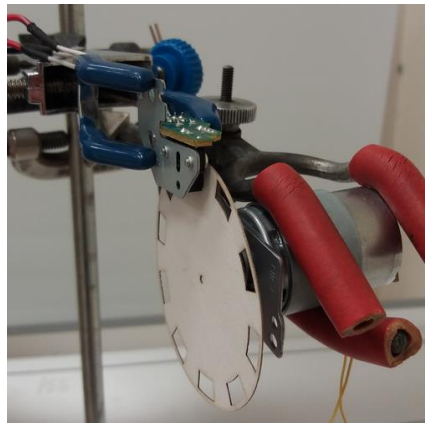
$$\begin{cases} U(p) = E(p) + R.I(p) + L.p.I(p) \\ (Jp + f).\Omega_m(p) = C_m(p) - C_r(p) \\ E(p) = K_e.\ \Omega_m(p) \\ C_m(p) = K_e.I(p) \end{cases}$$

Schéma bloc du moteur à courant continu



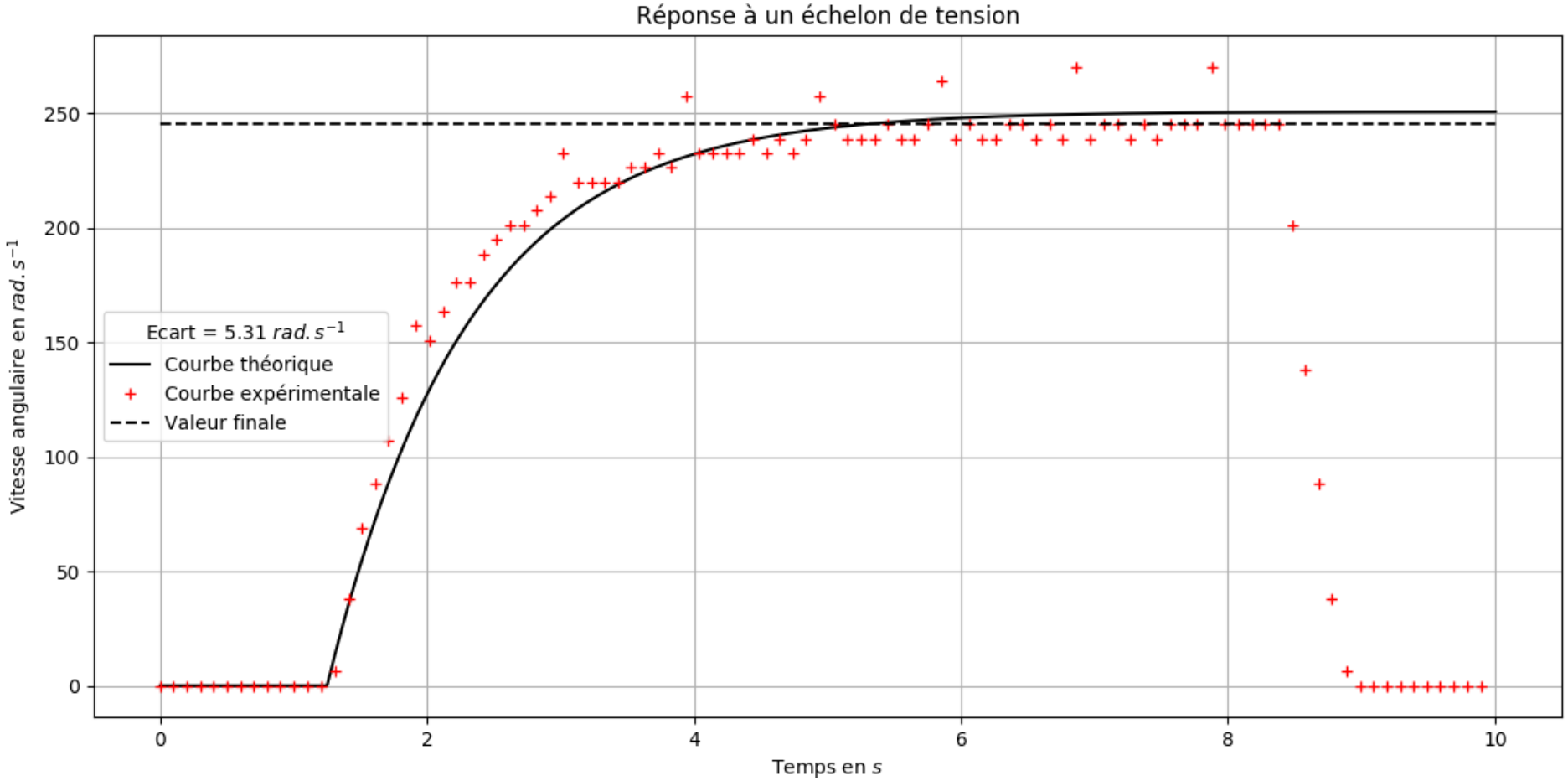
$$\begin{cases} \boldsymbol{K_e = 7,68.10^{-3} \pm 0,2.10^{-3}\ V.s.rad^{-1}} & \textit{Essai en génératrice} \\ \boldsymbol{R = 8.46 \pm 0.02\ \Omega} & \textit{Essai rotor bloqué} \\ \boldsymbol{f = 1.74.10^{-5} \pm 0.05.10^{-5}\ N.m.rad^{-1}.s} & \textit{Essai moteur} \\ \boldsymbol{J = 2.57.10^{-5} \pm 0,06.10^{-5}\ kg.m^2} & \textit{Essai moteur (échelon)} \\ \boldsymbol{C_r = ?} \end{cases}$$





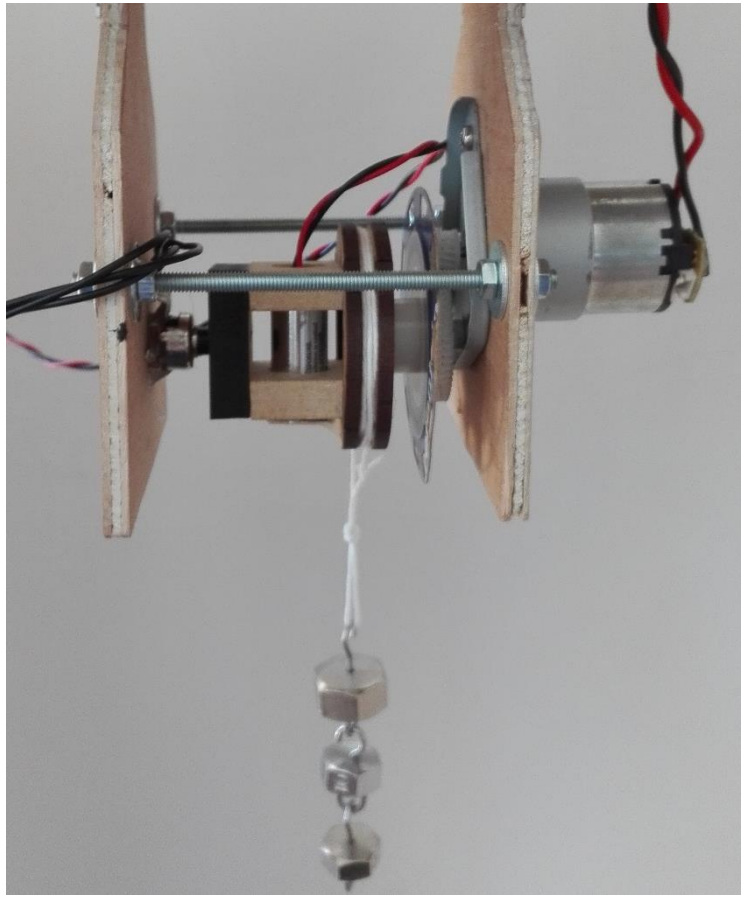
II) Pointage de la position : Détermination des caractéristiques du moteur

Détermination de C_r

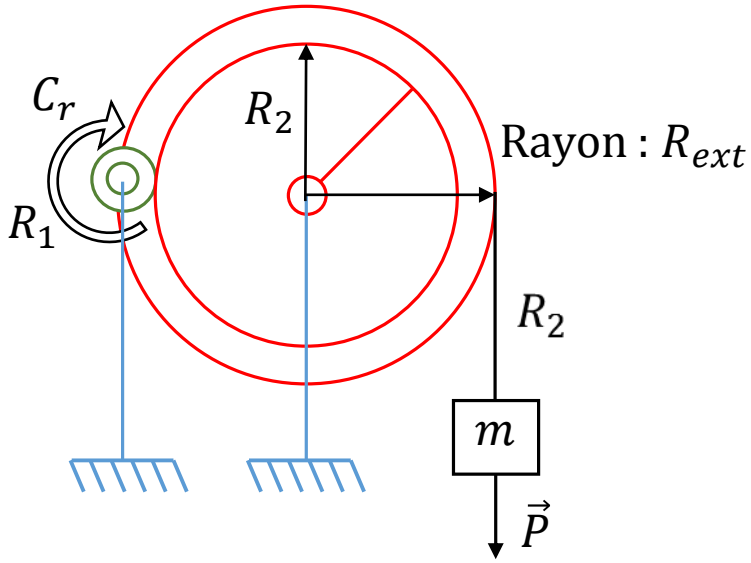


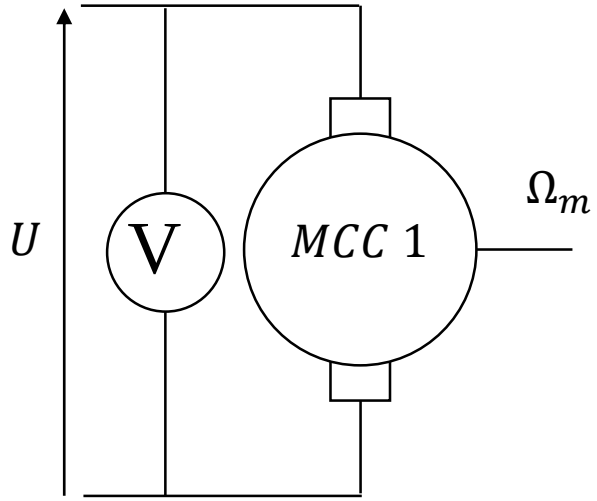
$$C_r(p) = \left(\Omega_m(p)\right)_{C_r(p)=0} - \Omega_m(p)\Big).\frac{R.f + K_e^2}{R} = 1{,}29.10^{-4}\ N.m$$

II) Pointage de la position : Détermination des caractéristiques du moteur



Détermination de C_r





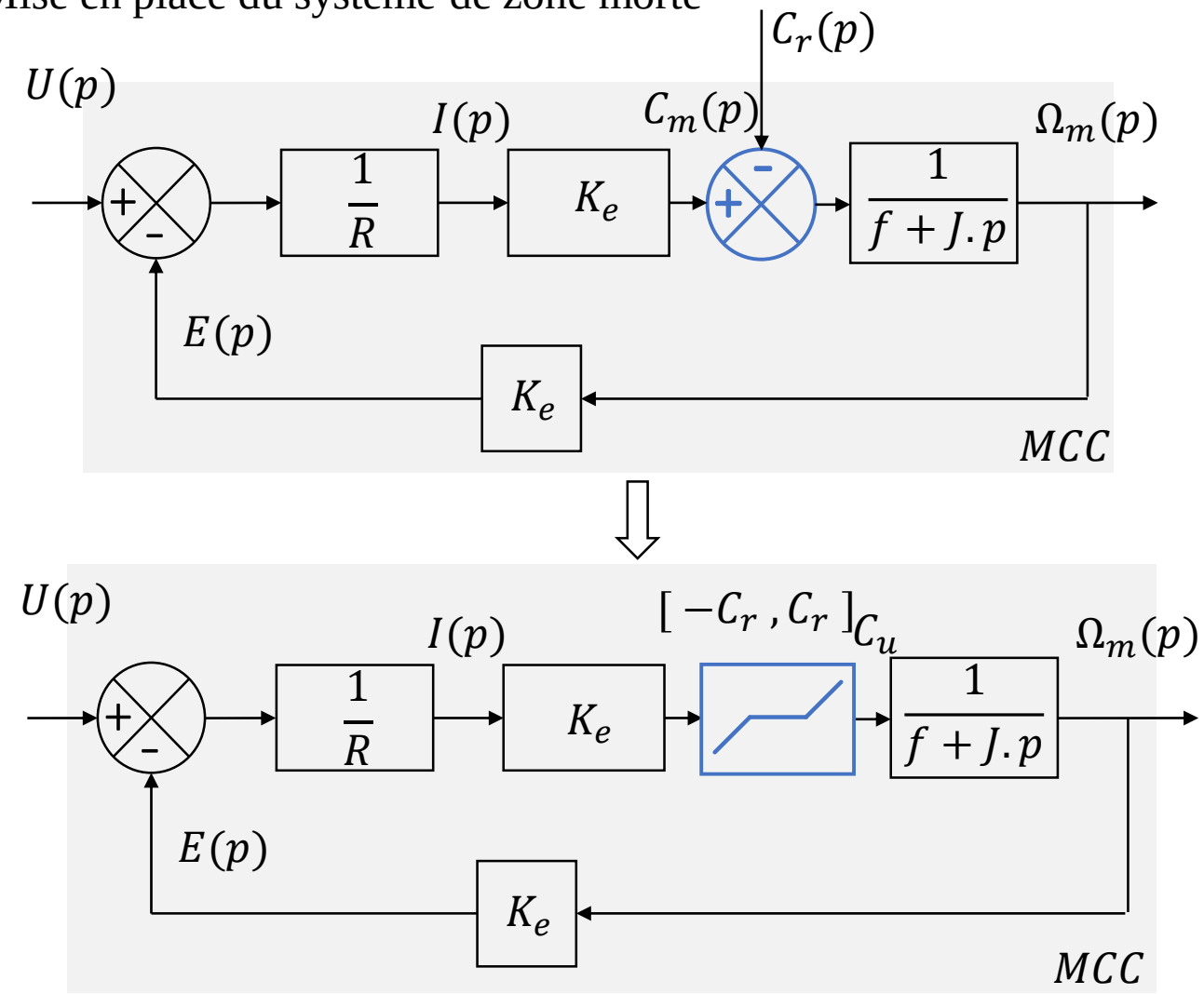
$$C_r = K_{red}.R_{ext}.m.g = 1{,}17.10^{-3}\ N.m$$

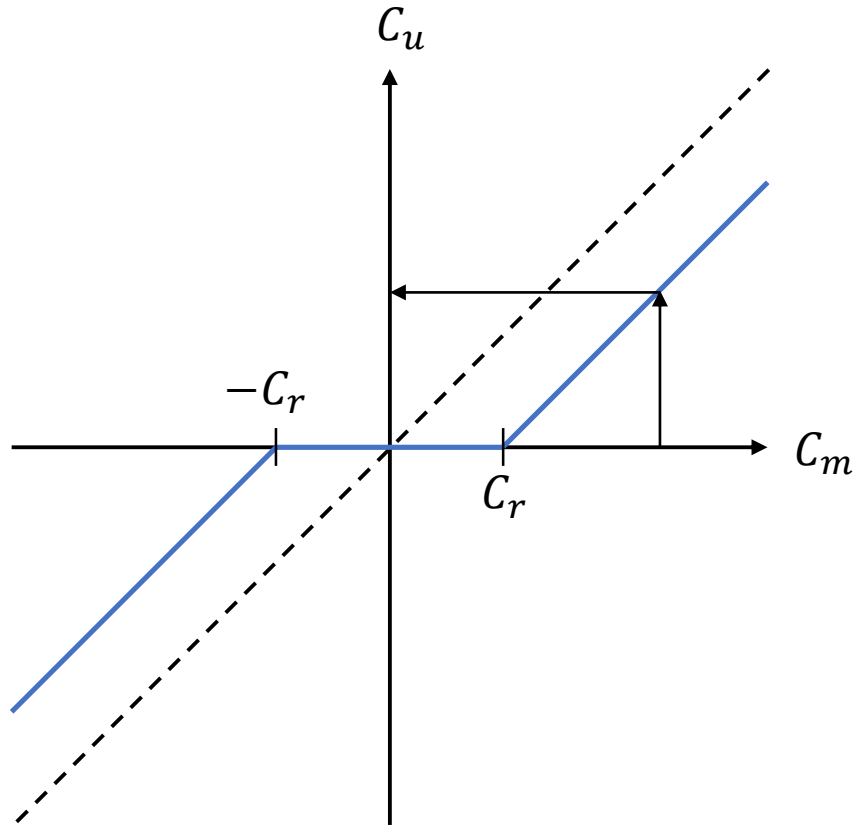
$$\boldsymbol{C_r = \frac{K_e}{R}.U = 1{,}044.10^{-3}\ N.m}$$

$avec\ U = 1{,}15\ V$

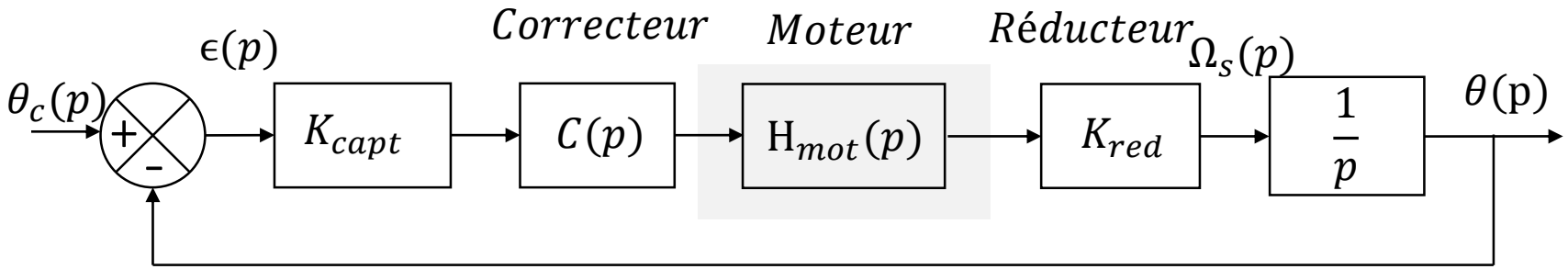
II) Pointage de la position : Détermination des caractéristiques du moteur

Mise en place du système de zone morte

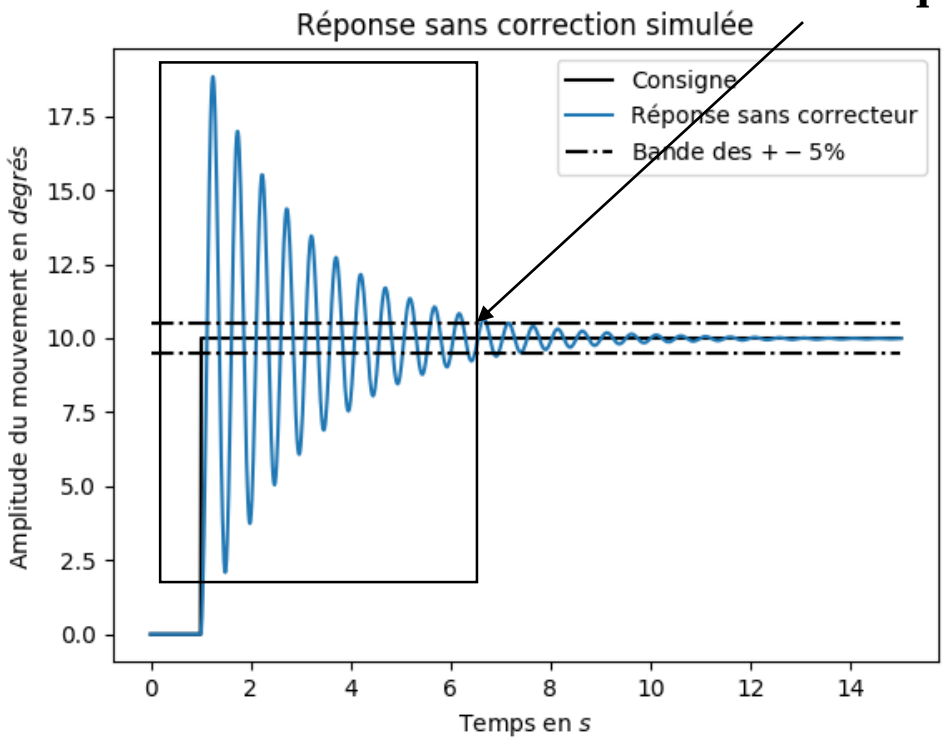


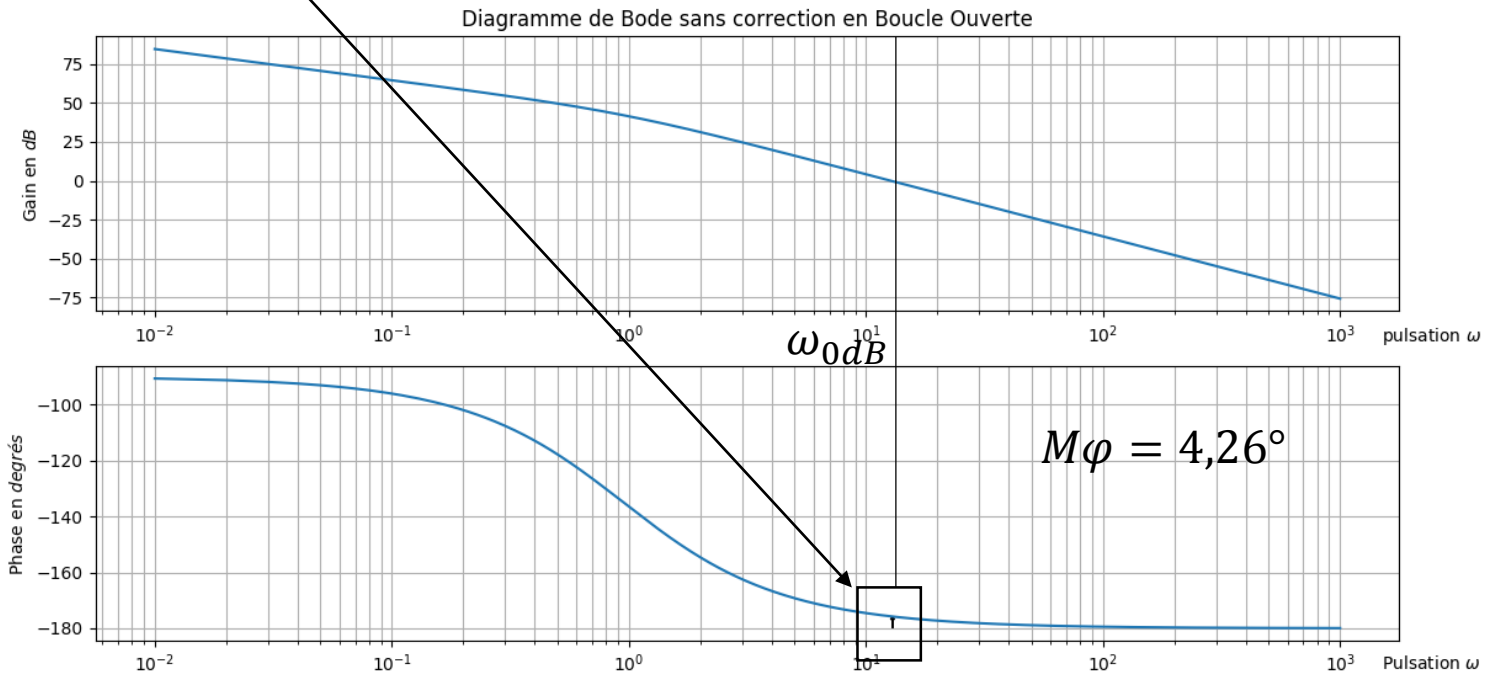


II) Pointage de la position : Asservissement

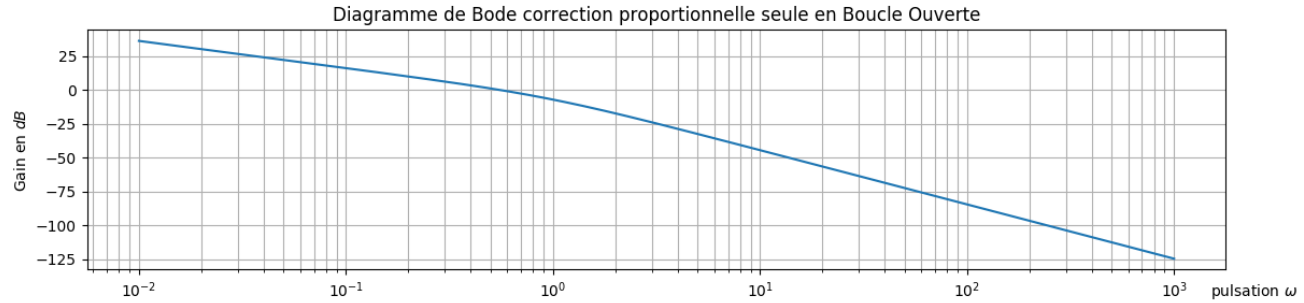


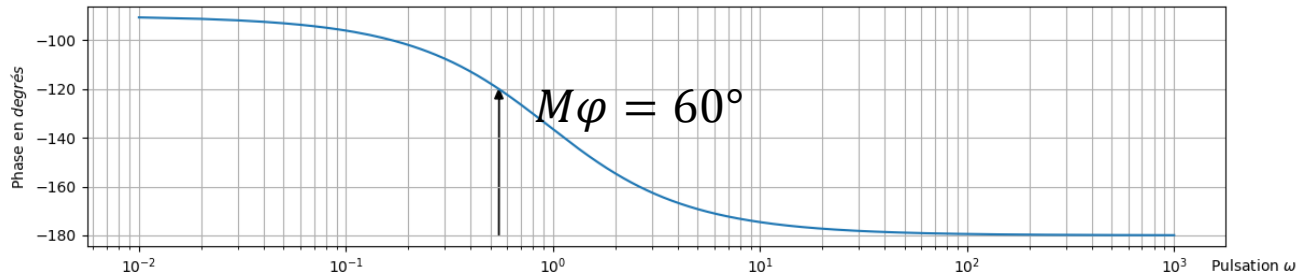
Manque de robustesse

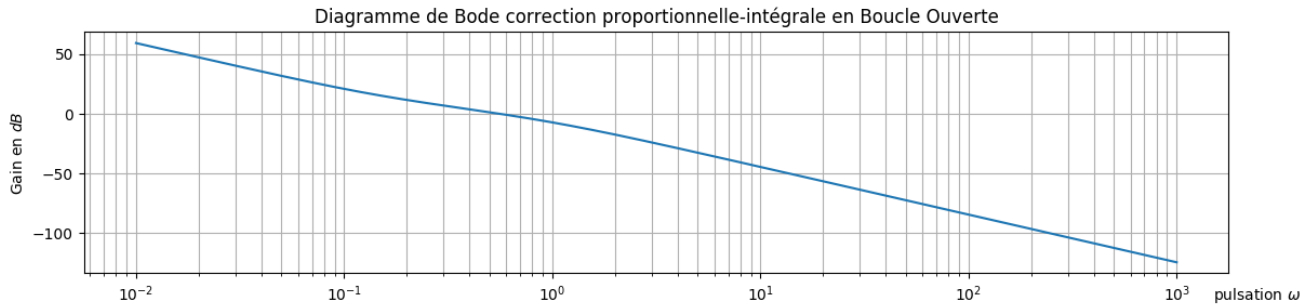


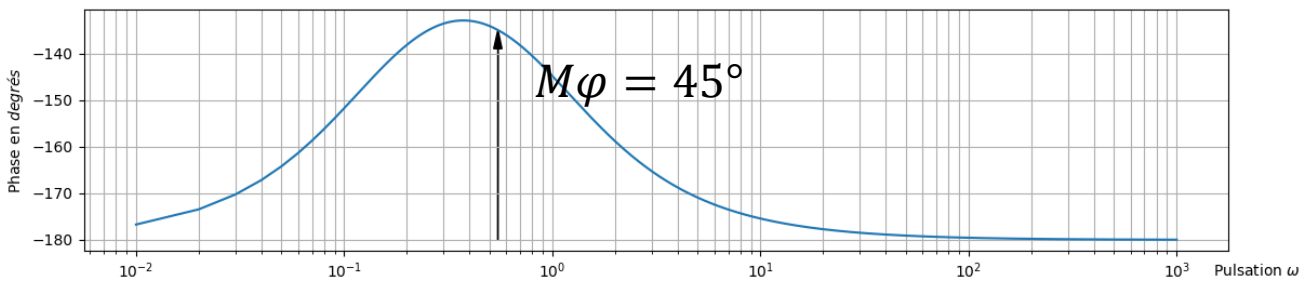


II) Pointage de la position : Mise en place de la correction









Choix d'un correcteur PI:

$$C(p) = K_i.(1 + \frac{1}{\tau_i.p})$$

$$M\varphi_{pc} = \arg\left(H_{nc}(j\omega_{0dB})\right) + 180°$$

$$M\varphi_{pc} \in [\,50° ; 75°\,]$$

$$\omega_{0db} \in [\,0{,}254 \;;\; 0{,}797\,]\ rad.s^{-1}$$

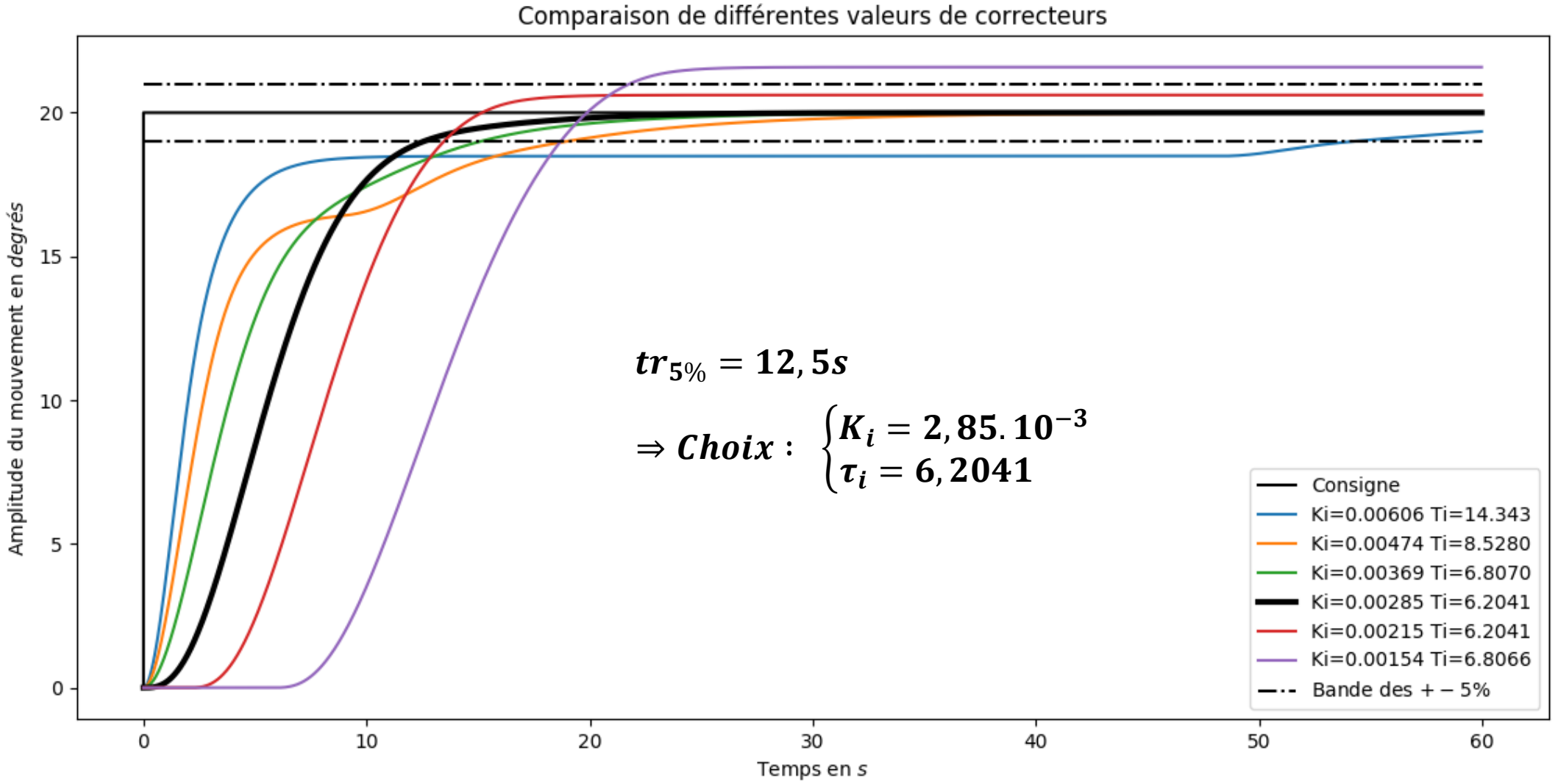
$$K_i = 10^{-\frac{GdB\left(H_{nc}(j\omega_{0dB,pc})\right)}{20}}$$

$$K_i \in [1{,}54.10^{-3} \;;\; 6{,}07.10^{-3}]$$

$$M\varphi_c = \arctan(\tau_i\omega) - \arctan(\frac{R.J}{K_e^2 + R.f}.\omega)$$

$$\tau_i \in [6{,}41 ; 14{,}34]\ s$$

II) Pointage de la position

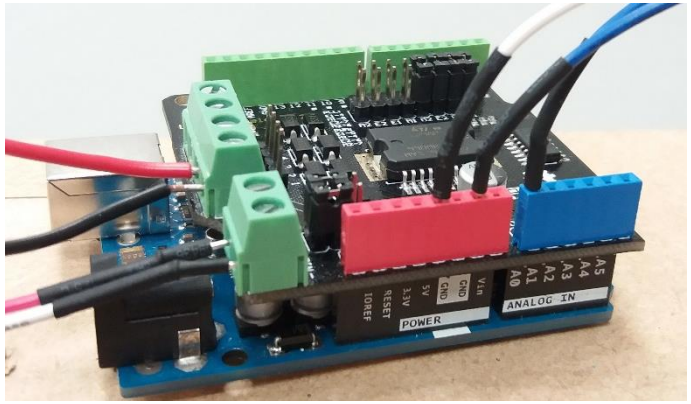


$$tr_{5\%} = 12,5s$$

$$\Rightarrow Choix : \begin{cases} K_i = 2,85.10^{-3} \\ \tau_i = 6,2041 \end{cases}$$

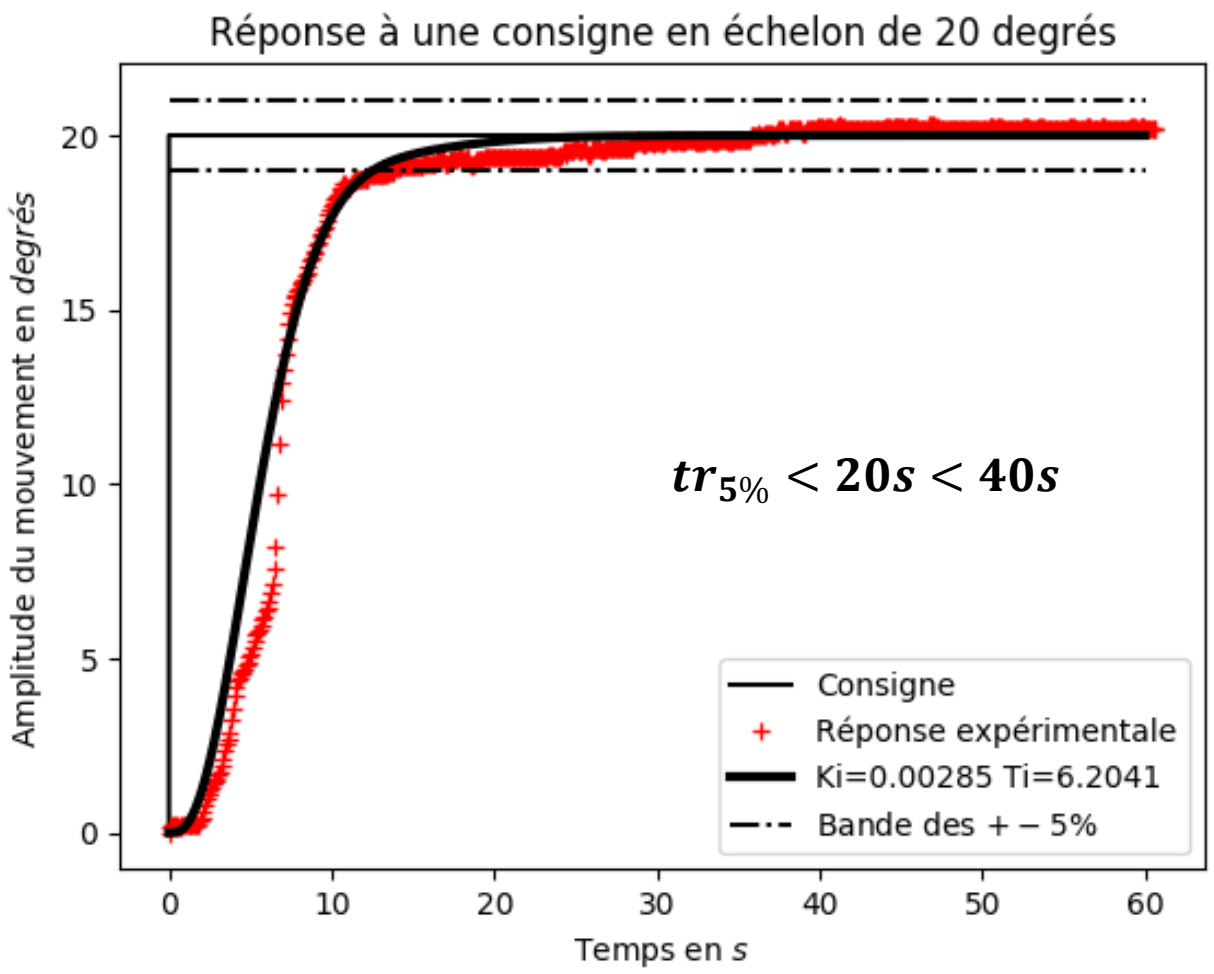
II) Pointage de la position : Implémentation dans l'Arduino

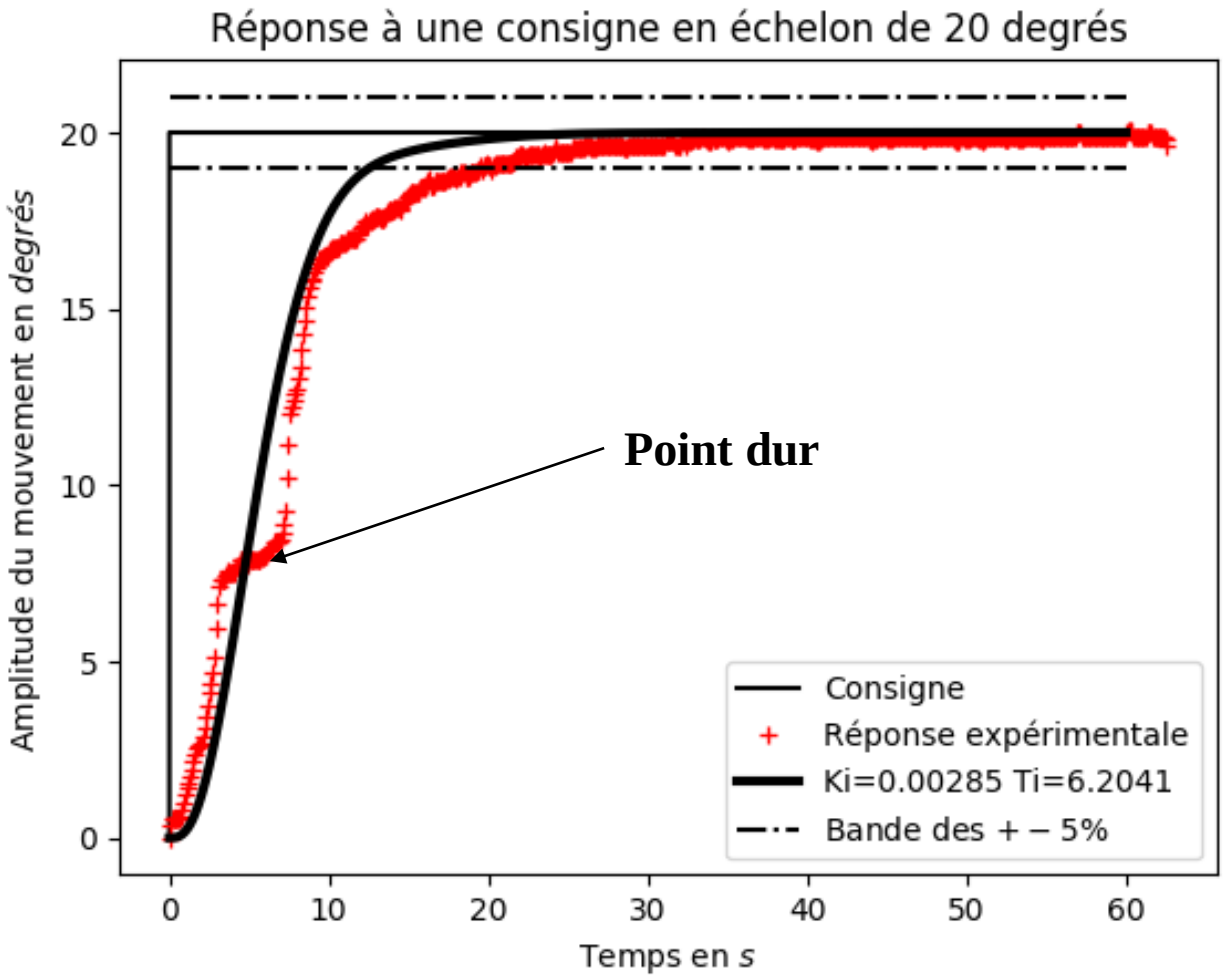
```
//Définition des paramètres du correcteur
float Ki=0.00285;
float Ti=6.2041;

void asser(){
  angle=calcule_angle();
  float ecart=consigne-angle;
//Correcteur PI numérique
  P = Ki*ecart*Kadapt;
  I = I + Ki *Kadapt* dt * ecart/Ti
  commande= P + I;
//Saturation
//Commande du moteur
  Moteur(commande, Valim);
  temps += dt;
}
```



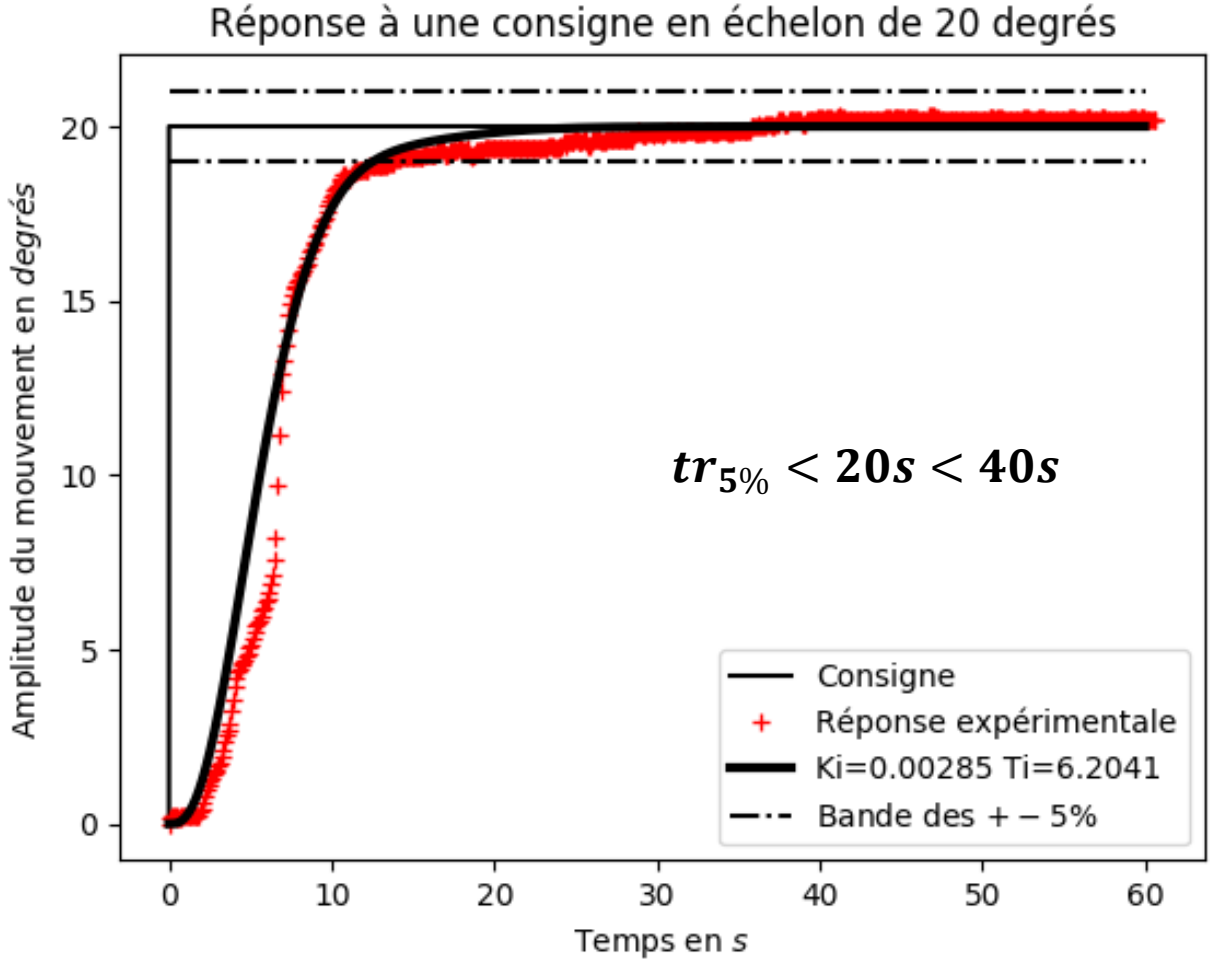

```
void Moteur(double commande, double Valim) {
int tension_int;
 //0/12V-->0/255
  tension_int = (int)(255*(commande/Valim));
//PWM
  if (tension_int>=0) {
    digitalWrite(4, LOW);//Un sens
    analogWrite(5, tension_int);
  }
  if (tension_int<0) {
    digitalWrite(4, HIGH);//L'autre sens
    analogWrite(5, -tension_int);
  }
}
float calcule_angle(){
int ValeurPotent = zero - analogRead(pinpotent);
  float tension = (float(ValeurPotent) * 5.0 )/ 1023;
  float R = (Rtot * tension) / utot;
  float angle = (R /18.09);
  return angle;
}
```

II) Pointage de la position : Résultats et limites de l'étude

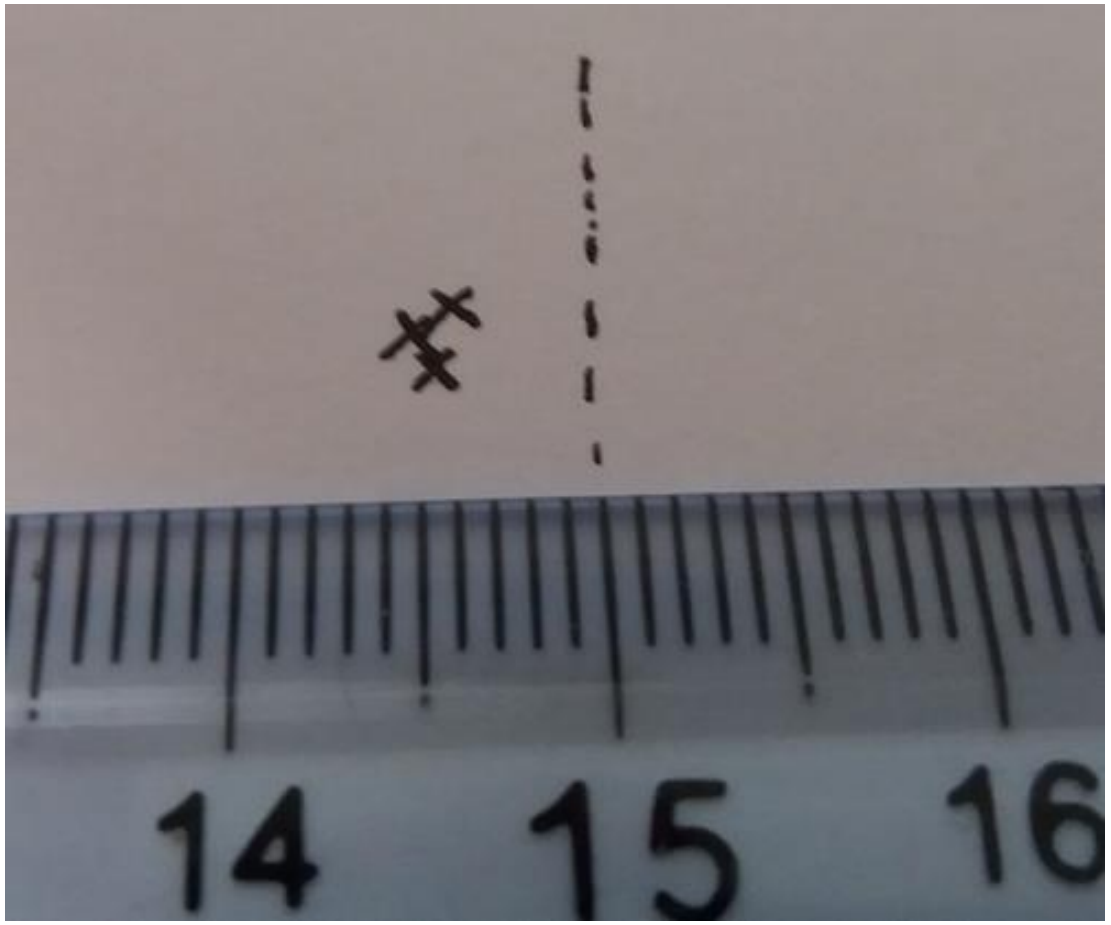




II) Pointage de la position : Résultats et limites de l'étude

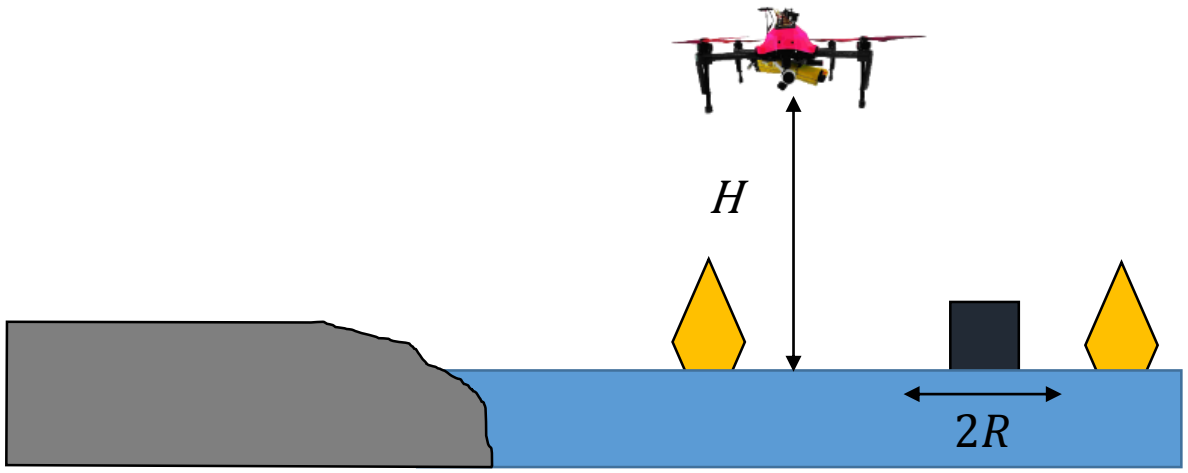


$$tr_{5\%} < 20s < 40s$$

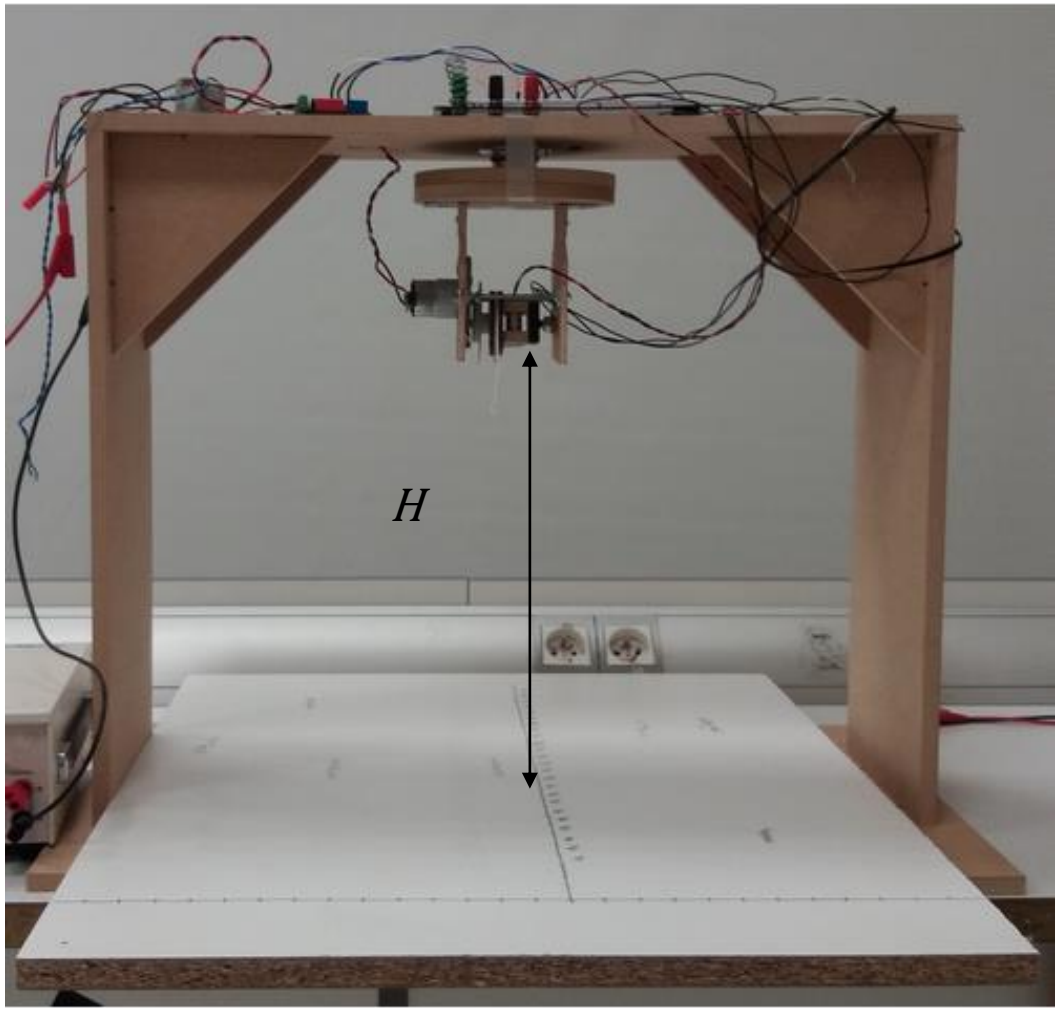


$$Ecart \leq 0,5\ cm$$

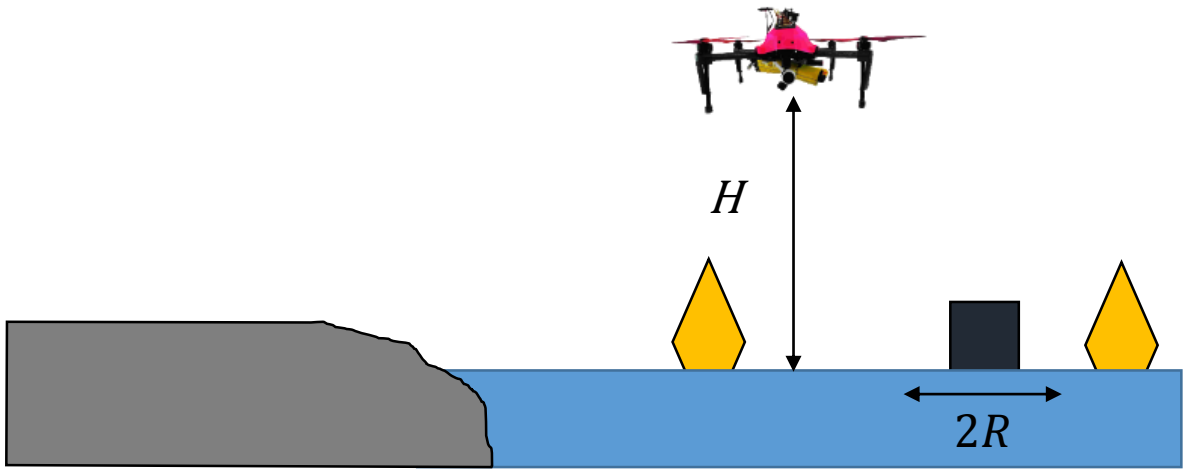
Conclusion: Objectif



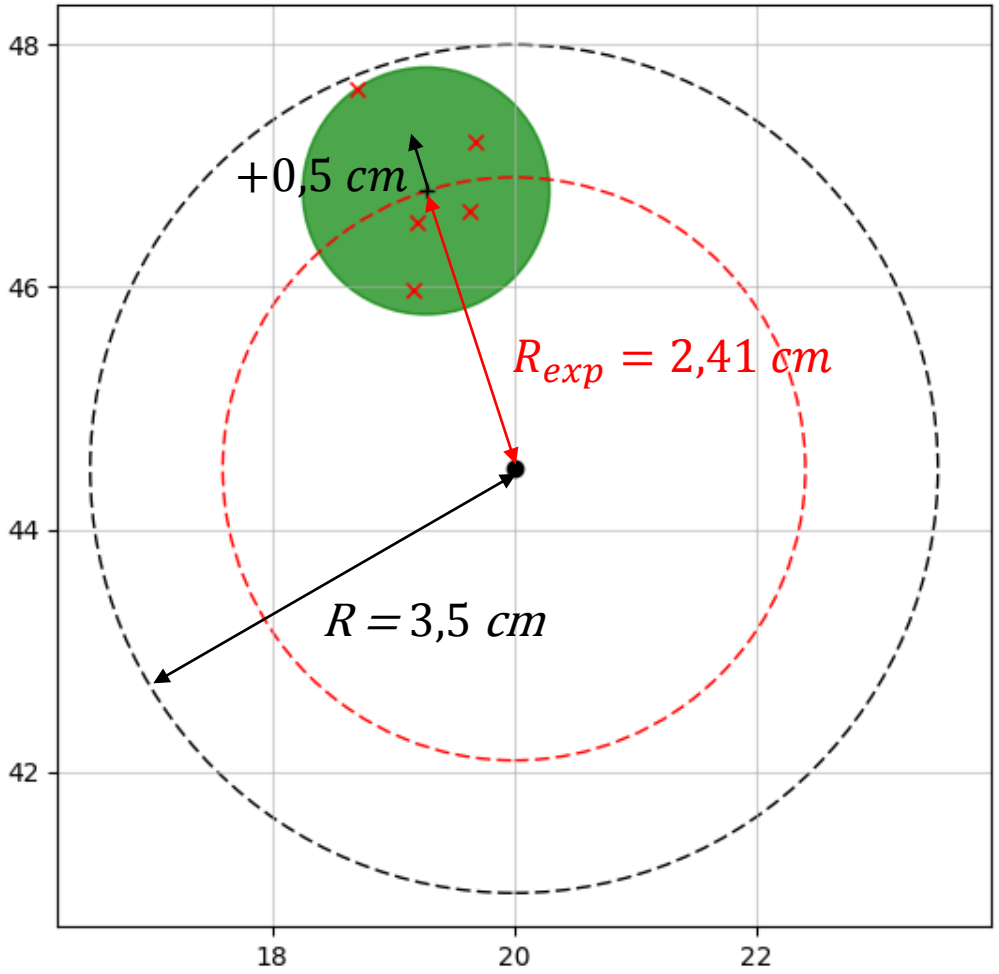
Situation réelle	Modélisation
H = 5 m	H = 35 cm
R = 50 cm	R = 3,5 cm



Conclusion: Objectif



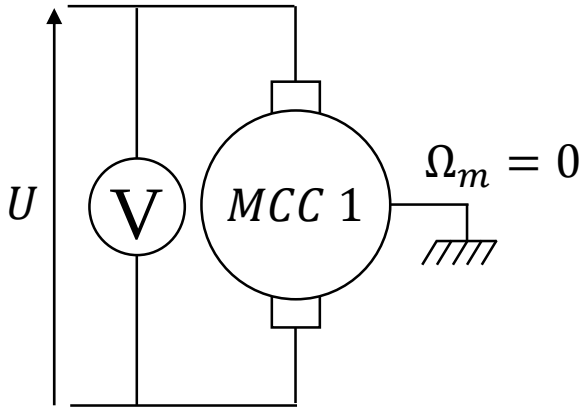
Situation réelle	Modélisation
H = 5 m	H = 35 cm
R = 50 cm	R = 3,5 cm



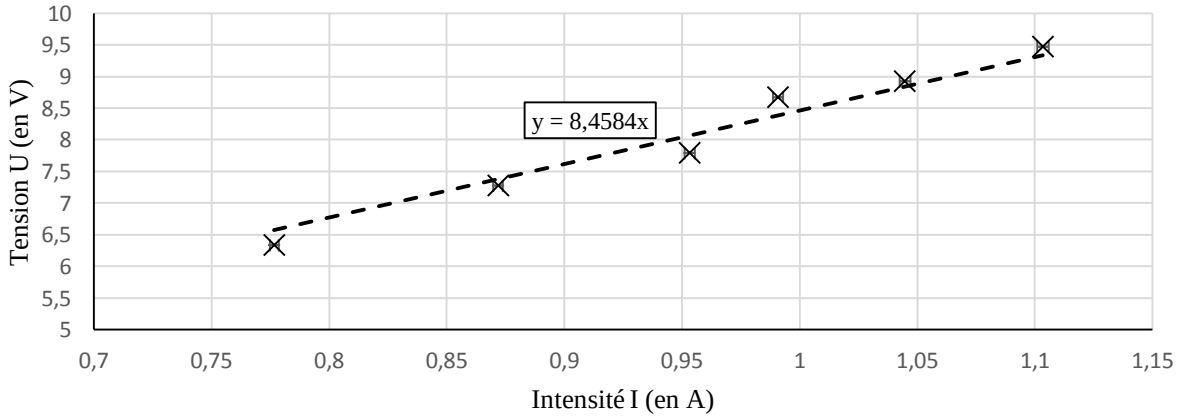
Annexe 1

Détermination de R

$$U(t) = R.I(t)$$

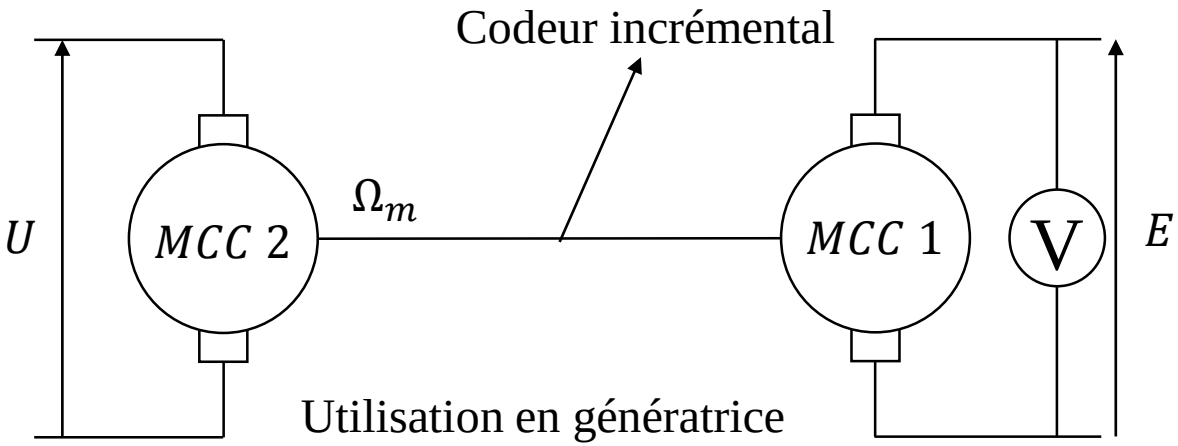


Tension en fonction de l'intensité

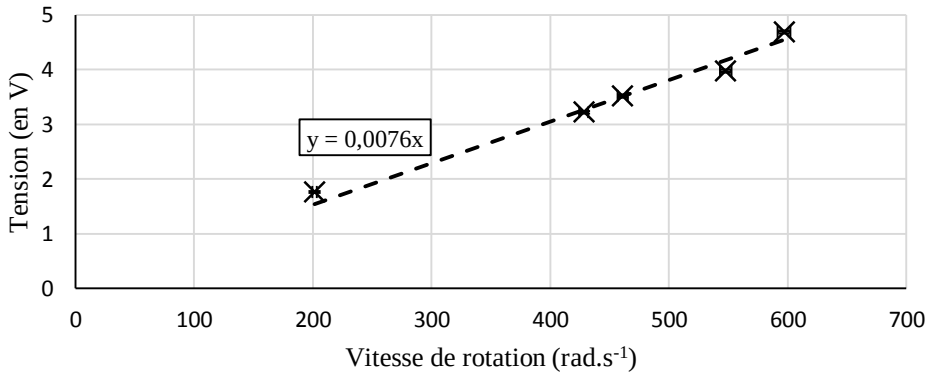


Détermination de K_e

$$U(t) = E(t) = K_e.\Omega_m(t)$$



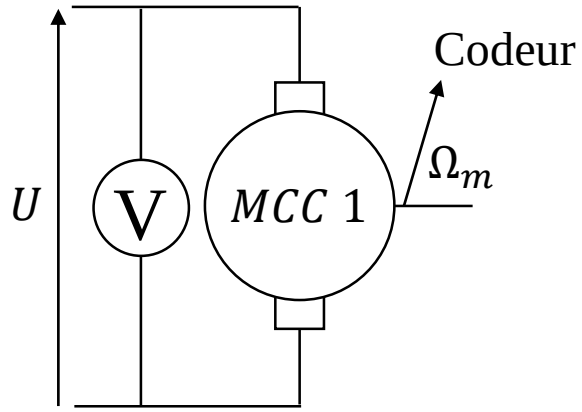
Tension en fonction de la vitesse de rotation au niveau de l'arbre moteur



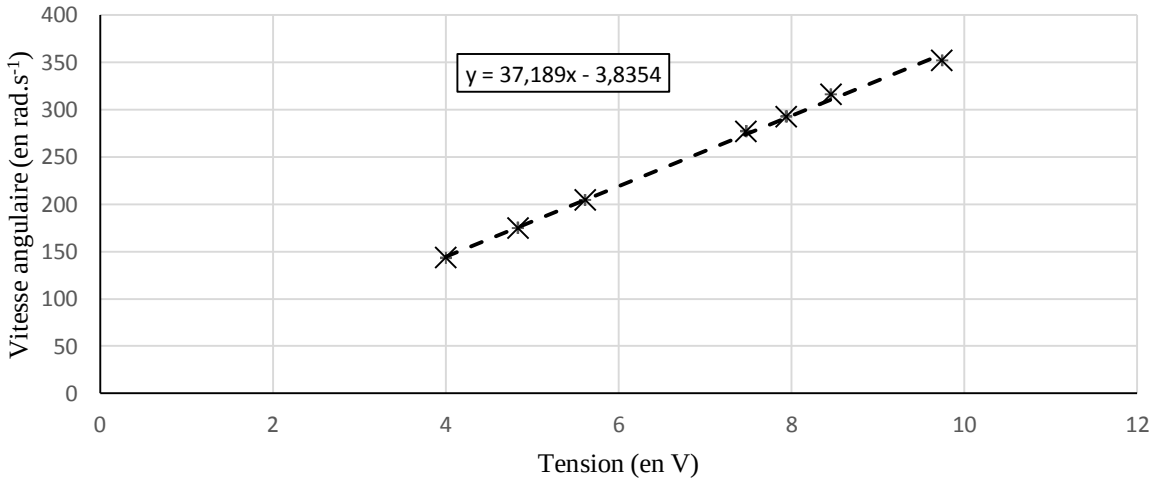
Annexe 1

Détermination de *f*

$$\frac{U(t)}{\Omega_m(p)} = \frac{\frac{K_e}{K_e^2 + R.f}}{1 + \frac{R.J}{K_e^2 + R.f}p}$$



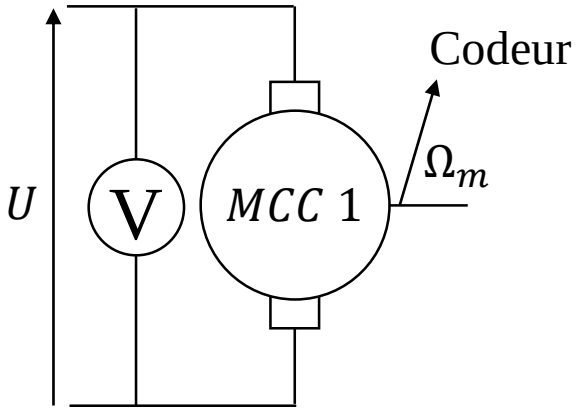
Vitesse angulaire en fonction de la tension au bornes du moteur (utilisation en moteur)

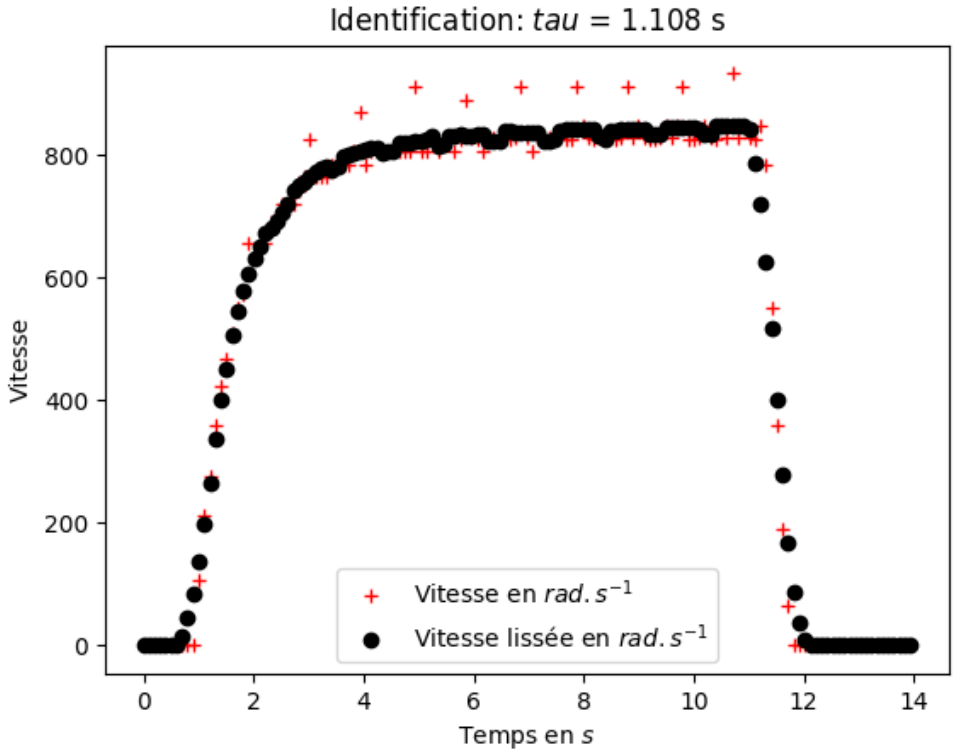


Détermination de *J*

$$\tau = \frac{R.J}{K_e^2 + R.f}$$

$$J = \frac{\tau.(K_e^2 + R.f)}{R}$$





Annexe 2: Résolution du système

```
import numpy as np
from scipy.optimize import fsolve
from math import sqrt

vitesse = 55151.9075366 #Valeur en cm/s de la vitesse des ondes dans le solide

X1,y1 = 5,25
X2,y2 = 30,52
X3,y3 = 55,25
#Coordonnées des différents capteurs (capteur 1 = EA0, capteur 2 = EA1, capteur 3 = EA2)

def temps_seuil(liste) : #Retourne l'indice correspondant au temps de début d'acquisition du signal
    j = 0
    while abs(liste[j]) < 0.01 :
        j = j+1
    return j
```

```
def recupere_donnees() :
    global temps_1_recu, temps_2_recu, temps_3_recu, amplitude_1_recu, amplitude_2_recu, amplitude_3_recu
    donnees = np.loadtxt("Point2_1.txt", skiprows = 1)
    temps_1_recu = donnees[:,0] #Liste associée au temps du capteur 1
    temps_2_recu = donnees[:,2] #Liste associée au temps du capteur 2
    temps_3_recu = donnees[:,4] #Liste associée au temps du capteur 3
    amplitude_1_recu = donnees[:,1] #Liste associée à l'amplitude reçue par le capteur 1
    amplitude_2_recu = donnees[:,3] #Liste associée à l'amplitude reçue par le capteur 2
    amplitude_3_recu = donnees[:,5] #Liste associée à l'amplitude reçue par le capteur 3

    temps_deb_1 = temps_1_recu[temps_seuil(amplitude_1_recu)]
    temps_deb_2 = temps_2_recu[temps_seuil(amplitude_2_recu)]
    temps_deb_3 = temps_3_recu[temps_seuil(amplitude_3_recu)]
    return temps_deb_1, temps_deb_2, temps_deb_3

T1, T2, T3 = recupere_donnees()

D1 = vitesse*T1
D2 = vitesse*T2
D3 = vitesse*T3

def f(L) : #L est la liste contenant les coordonnées du point recherché : L élément de R**2
    x,y = L[0],L[1]
    return [ sqrt((x-x3)**2 + (y-y3)**2) - sqrt((x-x1)**2 + (y-y1)**2) - (D3-D1),
             sqrt((x-x3)**2 + (y-y3)**2) - sqrt((x-x2)**2 + (y-y2)**2) - (D3-D2) ]


coordonnées = fsolve(f,[25,25])
print(coordonnées)
```

Annexe 3: Asservissement

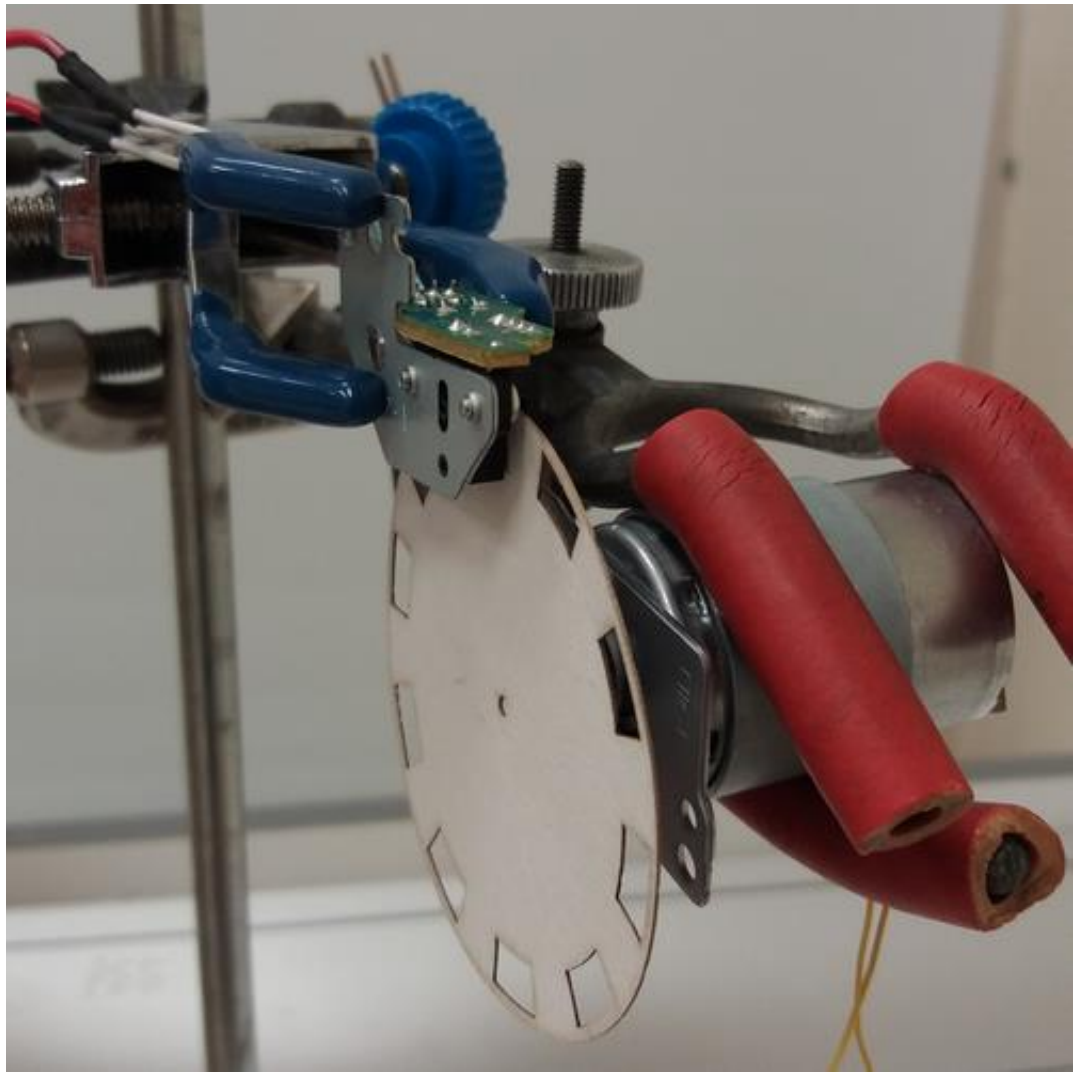
```
#include <digitalWriteFast.h>
#include <FlexiTimer2.h>
/*****************Définition des pins****************/
#define pinpotent A0
#define PWMA 5
#define DIRA 4

/*******************Consigne*********************/
float consigne = 20;//en degrés

/*******Définition des paramètres du correcteur********/
float Ki=0.00285;
float Ti=6.2041;

/***********  Définition des constantes  **********/
static double Umax=11.0;
static double Umin=-11.0;
double Valim = 11.0;// Tension d'alimentation
float zero;
float Kadapt=15.55;
float Rtot=4743;
float utot=4.893;
static double angle; // Angle du laser par rapport à la position de référence
static double commande = 0.; // commande en tension calculée par le PID
static double commande_avant_sat = 0.;
static double P = 0.; // Contient la valeur du terme proportionnel
static double I = 0.; // Valeur du terme intégral

/*******    Définition de la cadence    *******/
#define Cad 10
volatile double dt = Cad/1000.;
volatile double temps = -Cad/1000.;

#define TpsDonnees 100//Temps entre deux envois de données
unsigned long tempsEnvoi = 0;
unsigned long tempsReel= 0;

void setup() {//S'exécute une seule fois
  delay(2000);//délai de 2s permettant de positionner la référence/ouvrir le moniteur série...
  Serial.begin(9600);//Début de la communication Arduino --> Ordinateur
  pinMode(DIRA, OUTPUT);//DIRA --> voir Datasheet L298P
  pinMode(PWMA, OUTPUT);//PWMA

  zero=analogRead(pinpotent);//Pose la référence de position du potentiomètre

  FlexiTimer2::set(Cad, 1/1000., asser); // Exécution de la fonction asser à une cadence très
régulière
  FlexiTimer2::start();
}

void loop() {
  // Effectuée en boucle
  ecritureDonnees();//Envoie les données à l'ordinateur dès que possible (voir fonction)
}
```

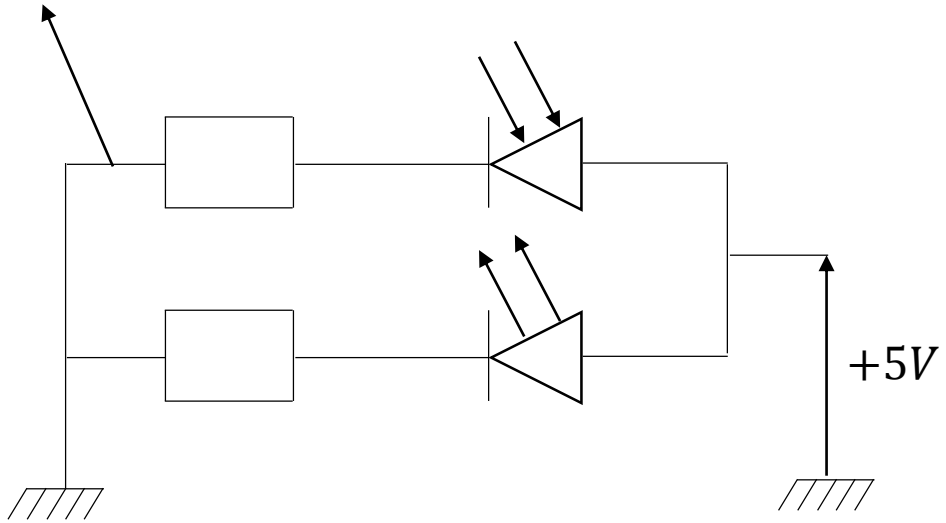
Annexe 3: Asservissement

```
float calcule_angle(){
  // calcule l'angle à partir des données lues sur le potentiomètre
  int ValeurPotent = zero - analogRead(pinpotent);
  float tension = (float(ValeurPotent) * 5.0 )/ 1023;//Tension en V
  float R = (Rtot * tension) / utot;//Pont diviseur de tension
  float angle = (R /18.09);//angle en degrés
  return angle;
}
void asser(){
  angle=calcule_angle();//calcul de l'angle
  float ecart=consigne-angle;//calcul de l'écart:

/******* Correcteur PI *******/
  P = Ki*ecart*Kadapt;// Terme proportionnel
  commande_avant_sat = P + I;
  if (commande_avant_sat > Umax) {
    commande = Umax;
  }
  else if (commande_avant_sat < Umin) {
    commande = Umin;
  }
  else {
    commande = commande_avant_sat;
  }

 I = I + Ki *Kadapt* dt * ecart/Ti
/******* Fin du PI *******/
  Moteur(commande, Valim);// Commande du moteur
  temps += dt;// Augmentation de la variable temps de dt
}
```

```
void Moteur(double commande, double Valim) {
  //Gère le contrôle du moteur
  int tension_int;
  //Normalisation de la tension d'alimentation par rapport à la tension d'alimentation :0/12V--
>0/255
  tension_int = (int)(255*(commande*5/Valim));
  // Saturation
  if (tension_int>255) {
    tension_int = 255;
  }
  if (tension_int<-255) {
    tension_int = -255;
  }
  //Commande du moteur en réglant le rapport cyclique (Pulse Width Modulation)
  if (tension_int>=0) {
    digitalWrite(4, LOW);//Un sens
    analogWrite(5, tension_int);
  }
  if (tension_int<0) {
    digitalWrite(4, HIGH);//L'autre sens
    analogWrite(5, -tension_int);
  }
}

void ecritureDonnees(void) {
// Ecriture des données si la durée depuis la dernière écriture est supérieure à TpsDonnees
// pour limiter l'utilisation de ressource.
  tempsReel= millis();//fonction millis donne le temps écoulé depuis le démarrage de la carte
  if (tempsReel-tempsEnvoi > TpsDonnees) {//Envoi seulement si un intervalle de temps est
dépassé
    Serial.println(String(temps)+";"+String(commande)+";"+String(angle));
    tempsEnvoi = tempsReel
}
```

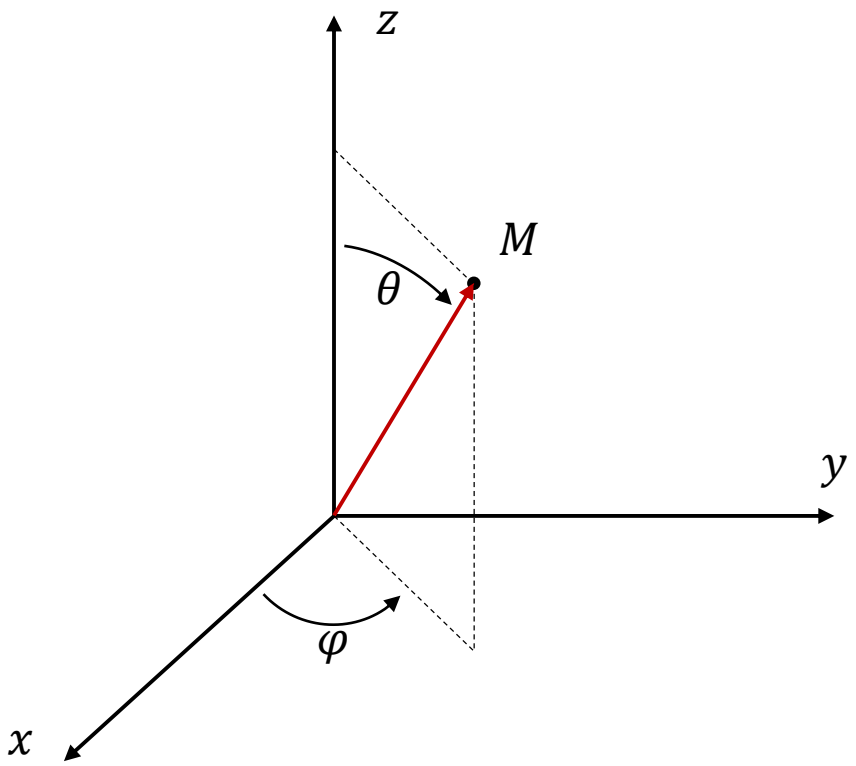
Annexe 4: Codeur incrémental





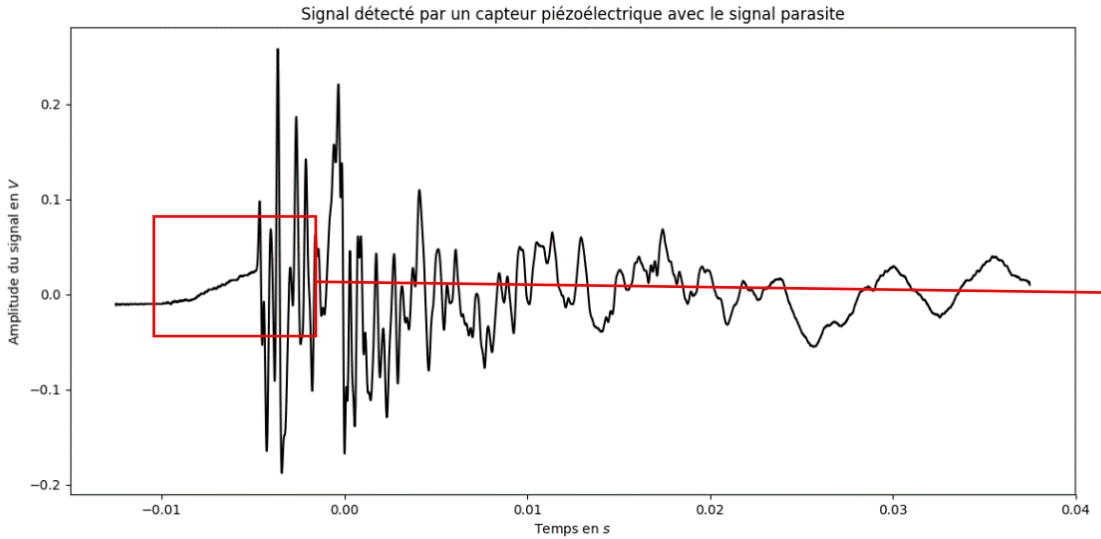
Annexe 5: Changement de base

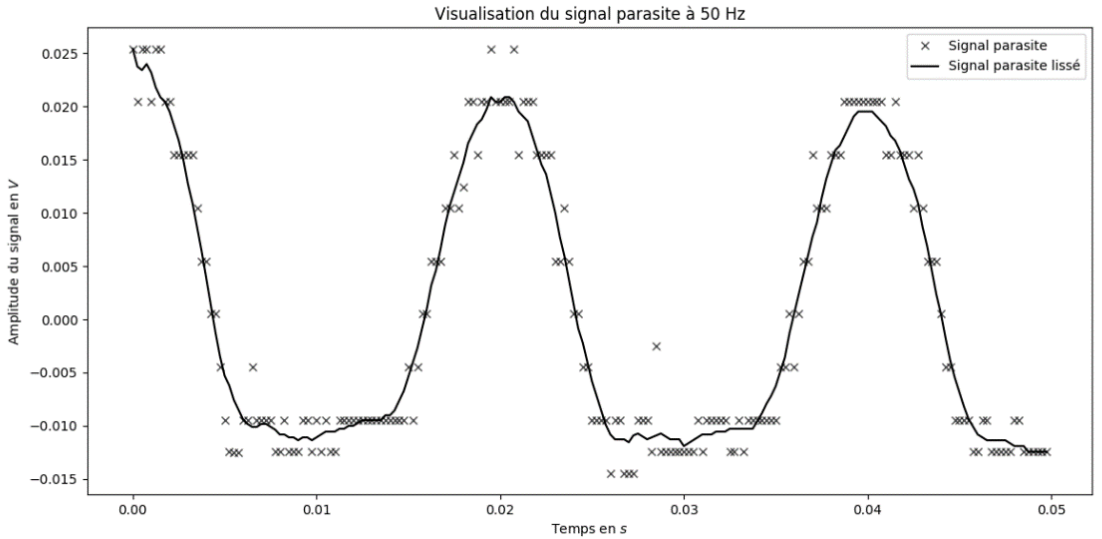
```
from math import atan , acos
from numpy import rad2deg

def transformation( x , y , z = 35 ):
    r = (x**2+y**2+z**2)**0.5
    phi = atan(y/x)
    theta = acos(z/r)
    return  r , phi , rad2deg(theta)
```



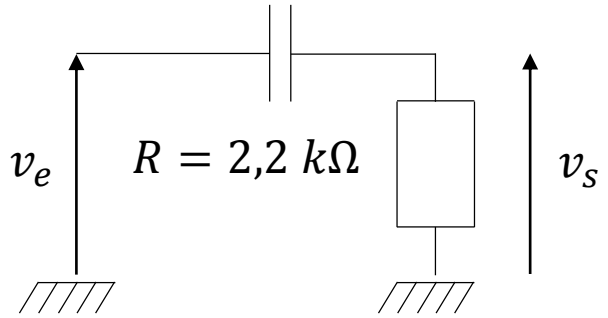
$$Coordonnées\ d'un\ point\ M : \begin{cases} x = r\sin(\theta)\cos(\varphi) \\ y = r\ sin(\theta)\sin(\varphi) \\ z = r\cos(\theta) \end{cases} \Rightarrow \begin{cases} r = \sqrt{x^2+y^2+z^2} \\ \varphi = Arctan(\frac{y}{x}) \\ \theta = Arcos(\frac{z}{\sqrt{x^2+y^2+z^2}}) \end{cases}$$

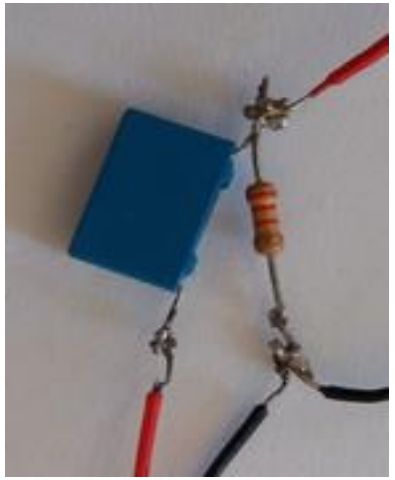
Annexe 6: Bruit à 50 Hz



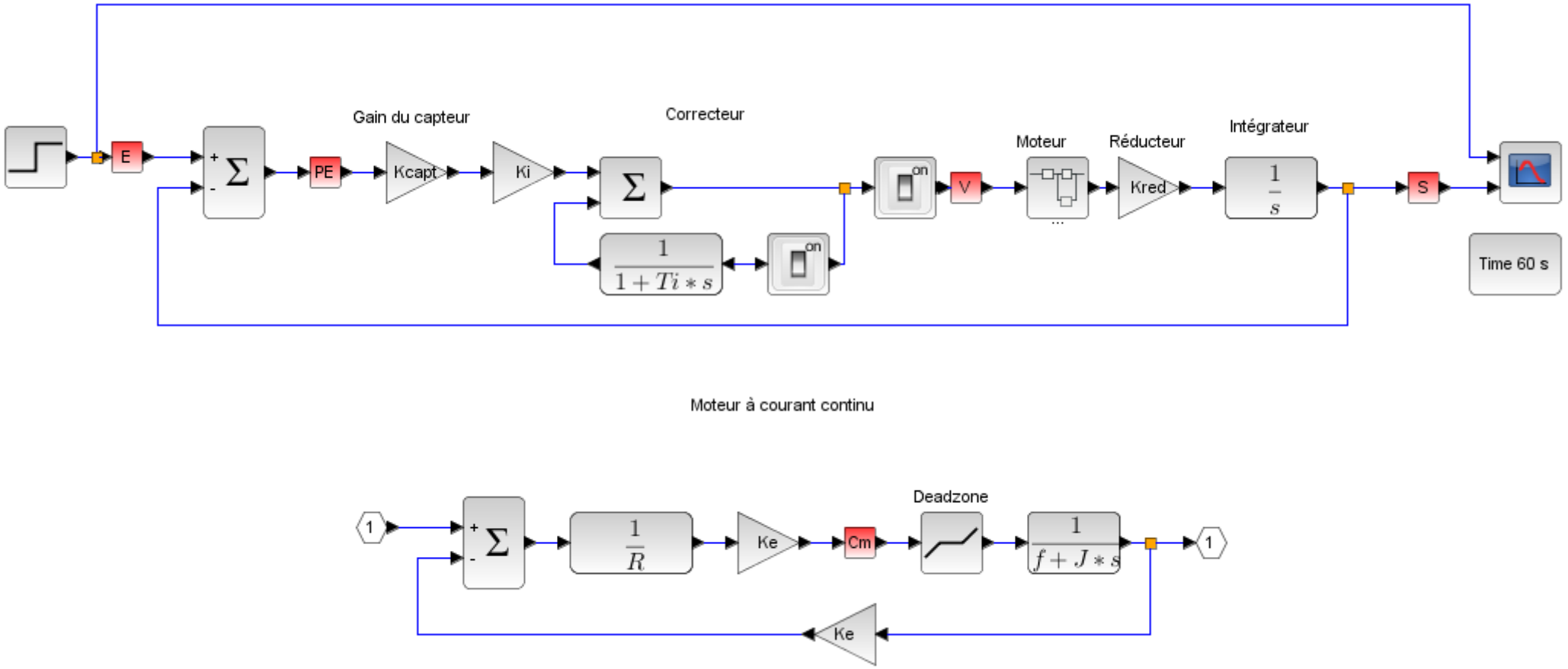


$C = 1\mu F$ $\quad f_c = 72{,}4\ Hz$





Annexe 7: Modélisation Scilab



Annexe 8: Réponse C_r sous-estimé