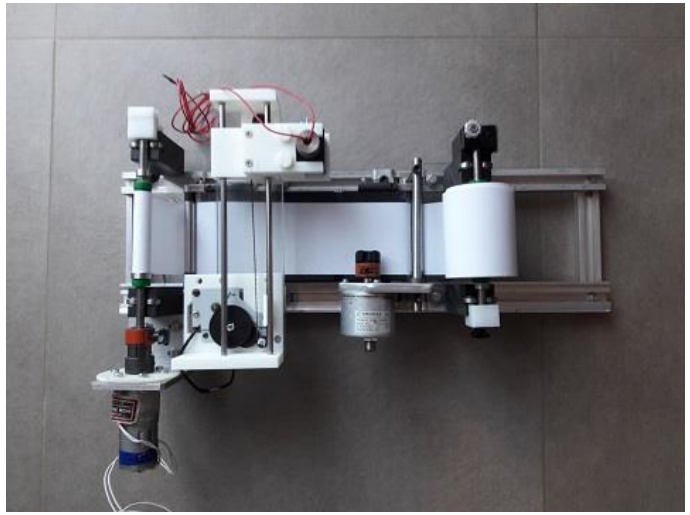
L’orgue de barbarie électronique

1. Découverte du projet
2. Modélisation du système
3. Comportement du système : comparaison théorique et expérimentale
4. Mise en place de l’asservissement
5. Conclusion



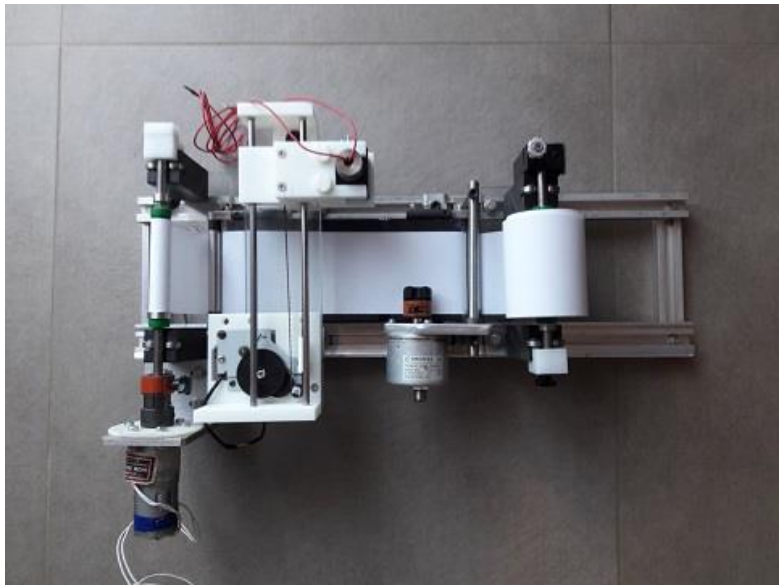
Objectifs & Maquette

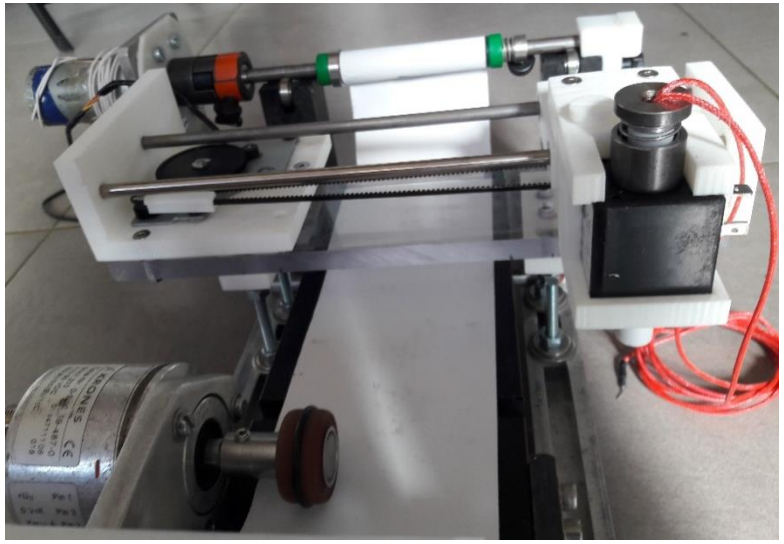
- **Objectifs :**

1. Réaliser une maquette
2. Modéliser la maquette
3. Asservir le déroulement du papier pour la lecture

- **Problématique :**

De quelle manière pouvons-nous modéliser notre système en vue de réaliser un asservissement du déroulement du papier?





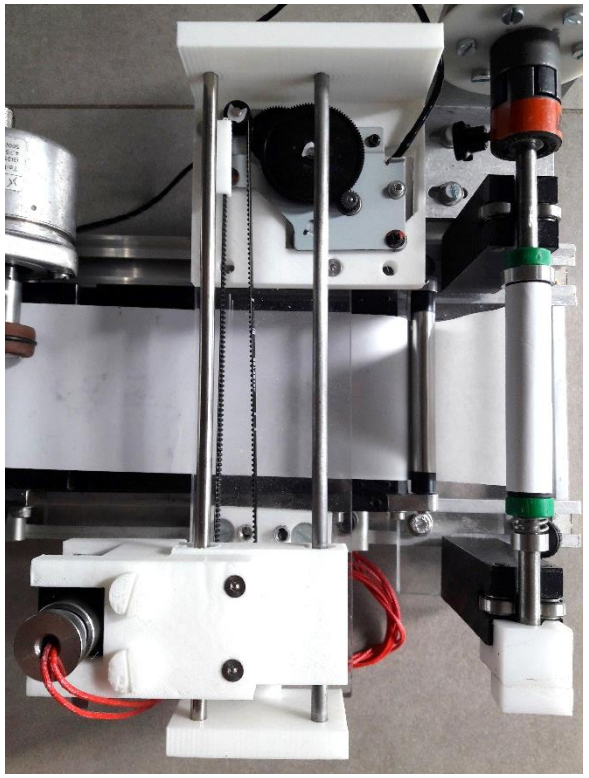


Schéma bloc et équations du MCC

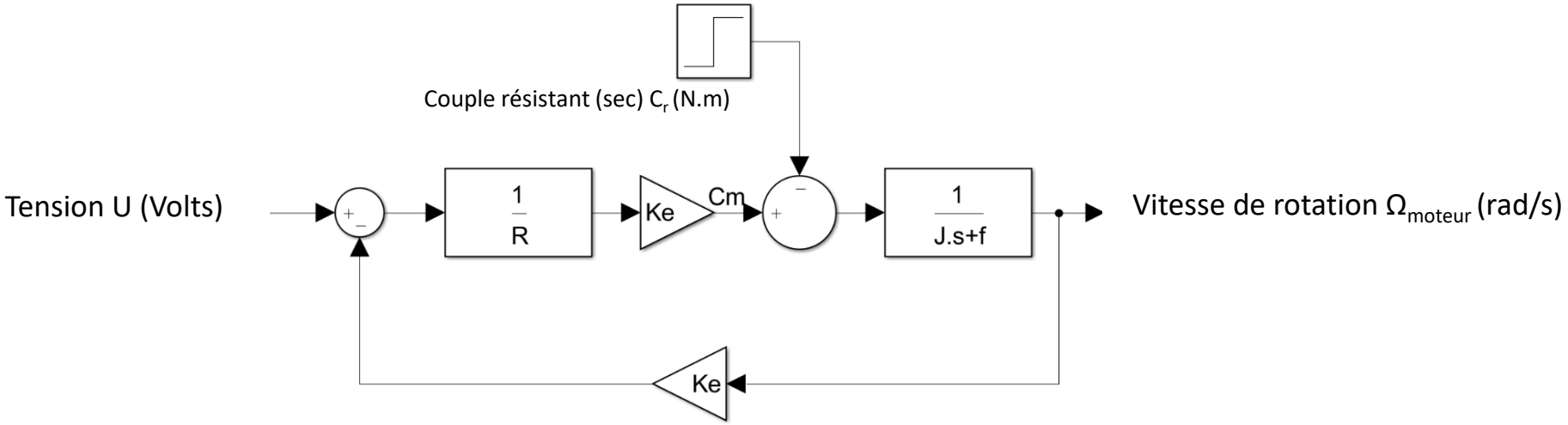
- Equations du moteur à courant continu:

$$U(p) = R \cdot I(p) + L \cdot p \cdot I(p) + K_e \cdot \Omega_m(p)$$

$$J \cdot p \cdot \Omega_m(p) = C_m(p) - C_r(p)$$

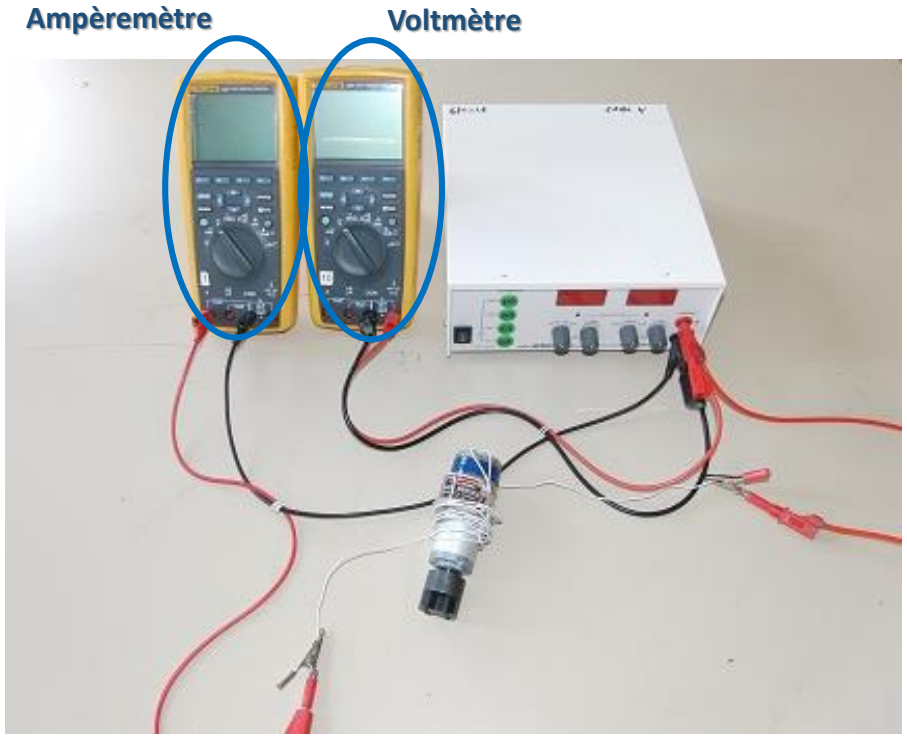
$$C_m(p) = K_c \cdot I(p)$$

- Schéma bloc du moteur à courant continu:



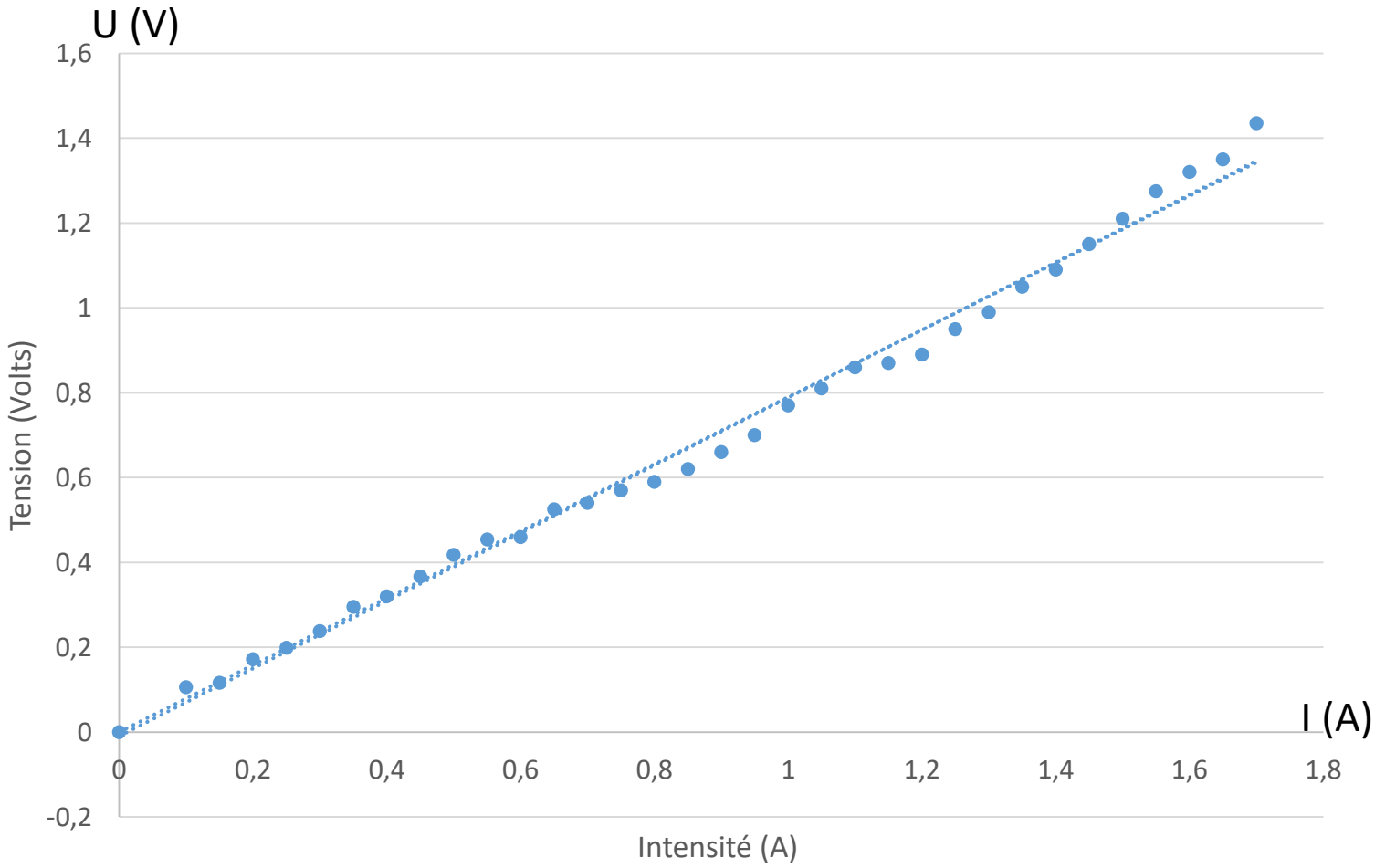
Détermination expérimentale des paramètres du moteur

- Résistance du moteur :
  Essai à rotor bloqué Ω=0



$$U = E + R \cdot i$$

$$U = R \cdot i$$

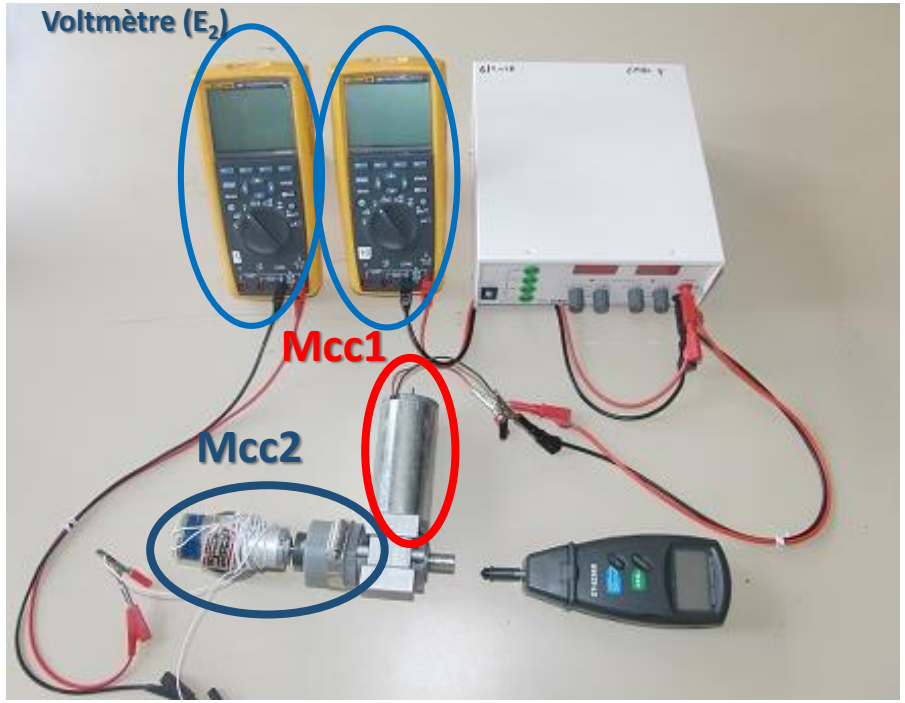


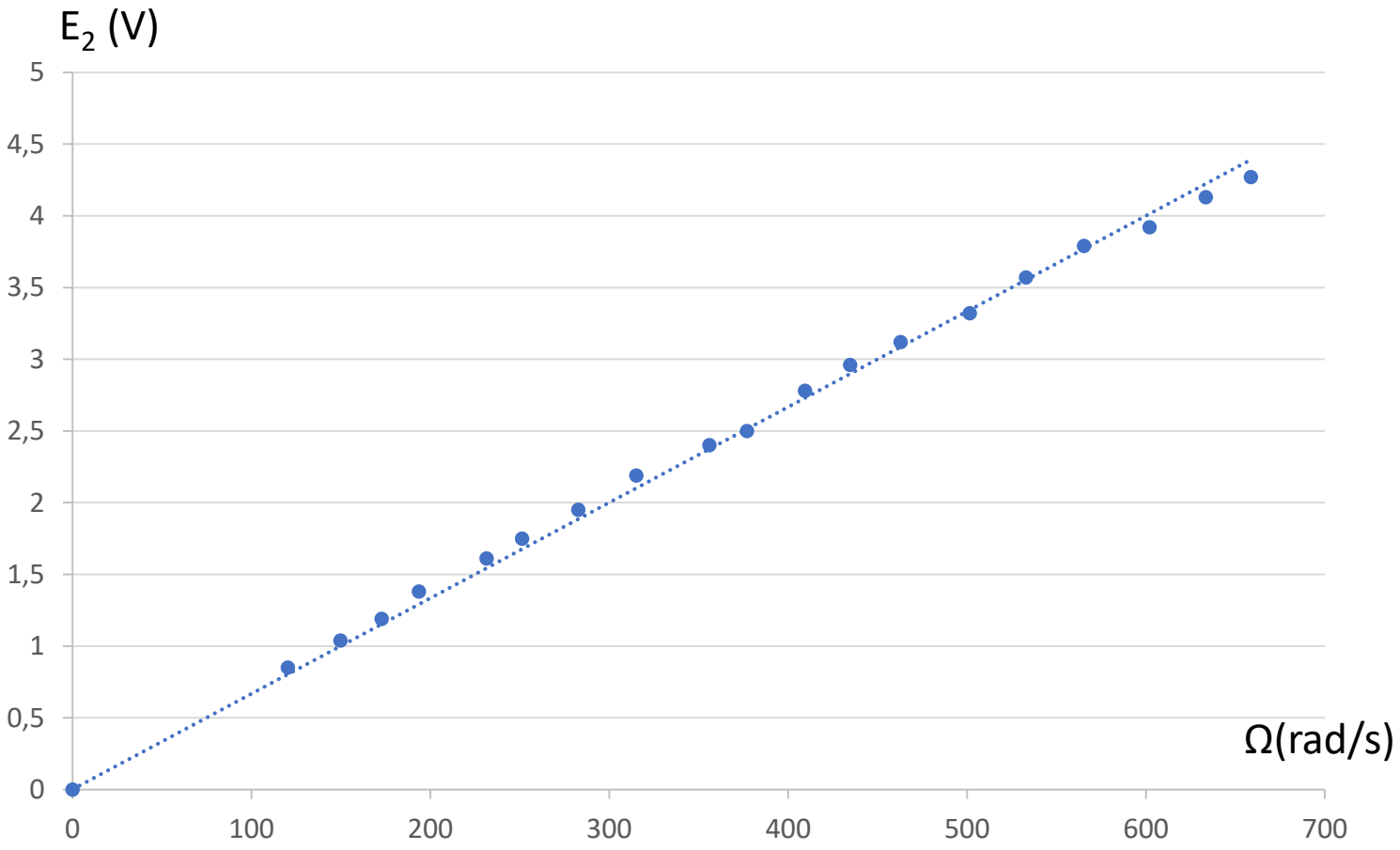
Valeur ohmmètre : $R_{ohm} = 7.3 \cdot 10^{-1}\Omega \pm 0.5 \cdot 10^{-1}\Omega$

Valeur mesurée : $R_{mes} = 7.6 \cdot 10^{-1}\Omega$

Détermination expérimentale des paramètres du moteur

- Constante K_e/K_c: Essai à vide





$$E_{mcc2} = K_{e_2}.\Omega$$

$$U_{mcc2} = E_{mcc2}$$ (Pas de courant R.i=0)

Valeur mesurée : $K_e = 6.67.10^{-3} \quad V/rad/s$

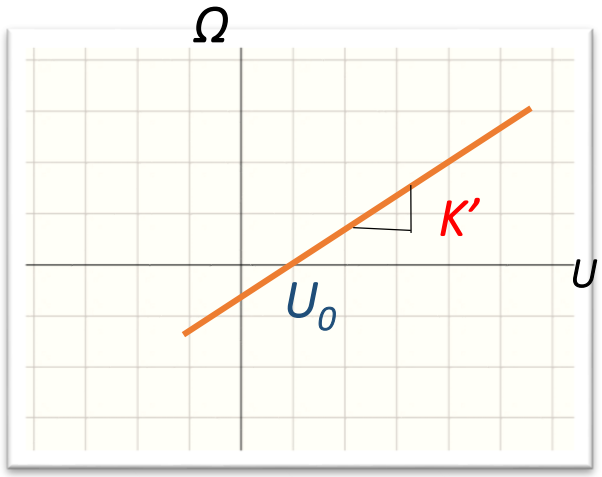
Détermination expérimentale des paramètres du système

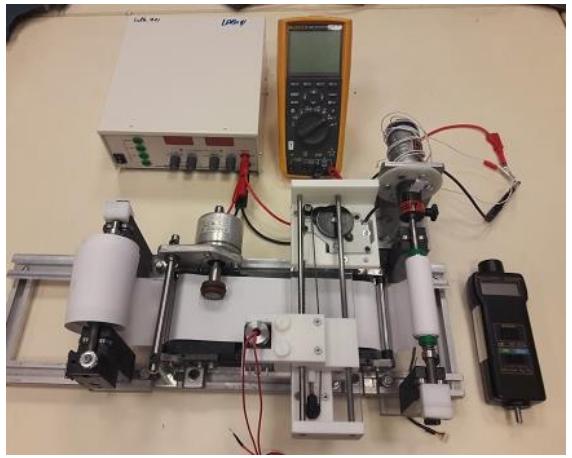
- Frottements fluide et sec:

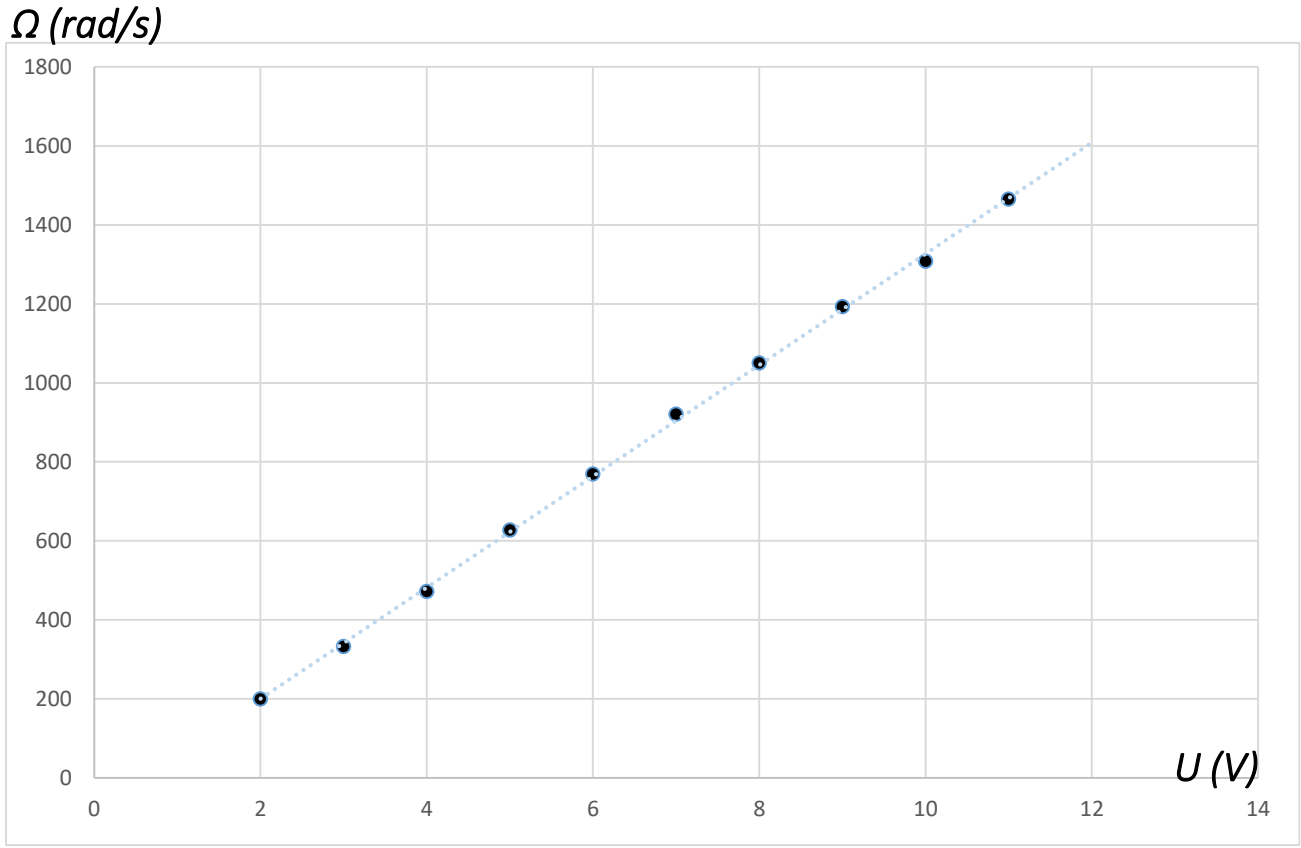
$$U = R.i + K_e.\Omega$$

$$U - \frac{R.C_{r0}}{K_c} = \frac{R.f + K_e.K_c}{K_c}.\Omega$$

$$\Omega = \underbrace{\frac{K_c}{R.f + K_e.K_c}}_{K'}.U - \underbrace{\frac{K_c}{R.f + K_e.K_c}}_{K'}.\underbrace{\frac{R.C_{r0}}{K_c}}_{U_0}$$







$$f = 3.8.10^{-6} \quad N.m/rad/s$$

$$C_{r0} = 5.10^{-3} \quad N.m$$

Calcul théorique du moment d'inertie équivalent

- Energies cinétiques:

$$Ec_{moteur/0} = \frac{1}{2} \cdot J_{moteur} \cdot \Omega_{moteur/0}^{2}$$

$$Ec_{codeur/0} = \frac{1}{2} \cdot J_{codeur} \cdot \Omega_{codeur/0}^{2}$$

$$Ec_{sup/0} = \frac{1}{2} \cdot J_{sup}(t) \cdot \Omega_{moteur/0}^{2}$$

$$Ec_{bobine/0} = \frac{1}{2} \cdot J_{bob}(t) \cdot \Omega_{moteur/0}^{2}$$

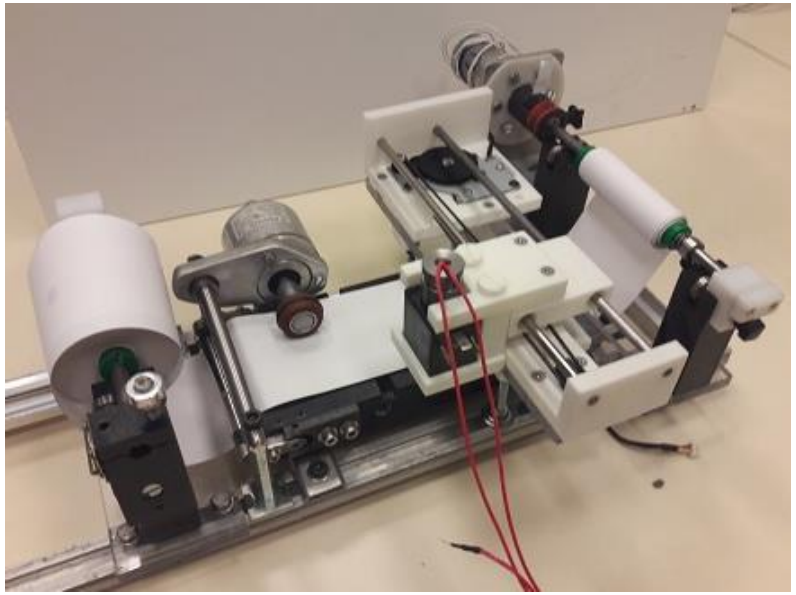
- Energies cinétiques exprimée avec la vitesse du moteur:

$$Ec_{moteur/0} = \frac{1}{2} \cdot J_{moteur} \cdot \Omega_{moteur/0}^{2}$$

$$Ec_{codeur/0} = \frac{1}{2} \cdot J_{codeur} \cdot \left(\frac{R_{sup}(t)}{R_{codeur}}\right)^{2} \cdot \Omega_{moteur/0}^{2}$$

$$Ec_{sup/0} = \frac{1}{2} \cdot (J_{axe} + J_{papier\ enroulé}(t)) \cdot \Omega_{moteur/0}^{2}$$

$$Ec_{bobine/0} = \frac{1}{2} \cdot (J_{axe} + J_{papier\ déroulé}(t)) \cdot \left(\frac{R_{bob}(t)}{R_{sup}(t)}\right)^{2} \cdot \Omega_{moteur/0}^{2}$$



- Moment d'inertie équivalent en fonction du temps :

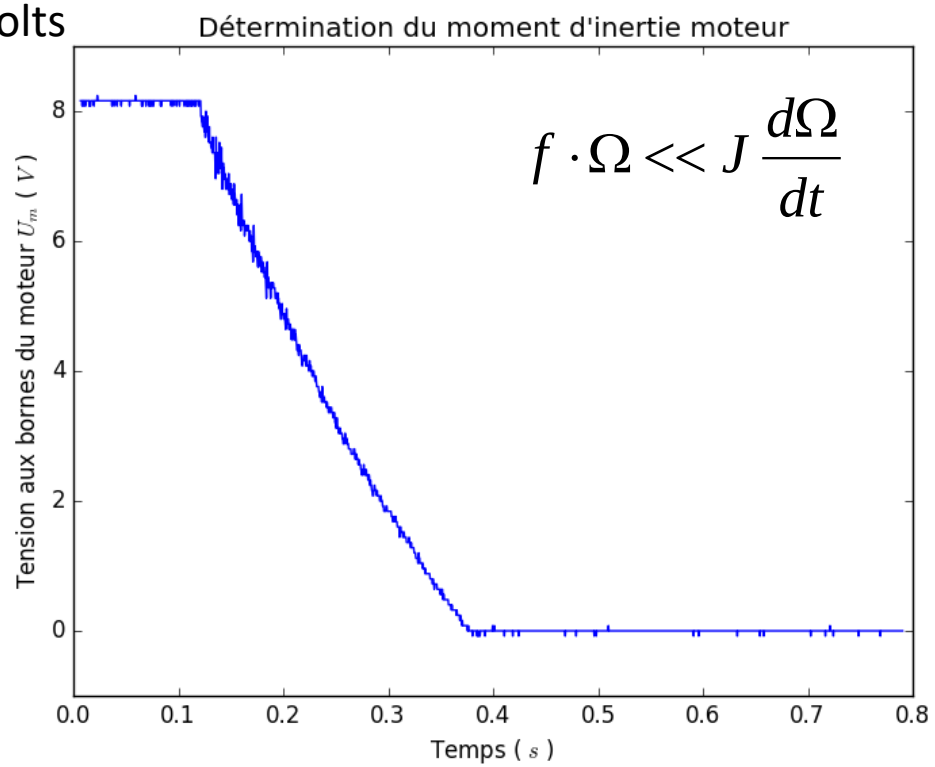
$$J_{eq}(t) = J_{moteur} + J_{codeur} + 2 \cdot J_{axe} + J_{papier\ enroulé}(t) + J_{papier\ déroulé}(t)$$

$$J_{eq}(t=0) = 1.1 \cdot 10^{-5} \quad kg.m^{2}$$

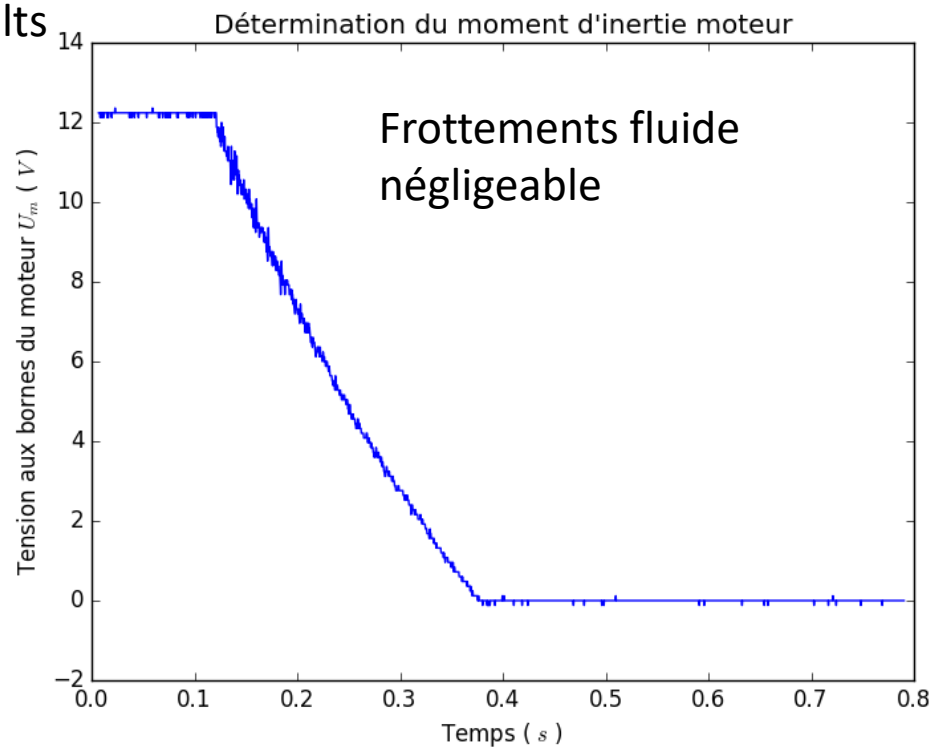
Détermination expérimentale des paramètres du système

- Moment d'inertie expérimental : Essai en lâché

8 Volts



12 Volts

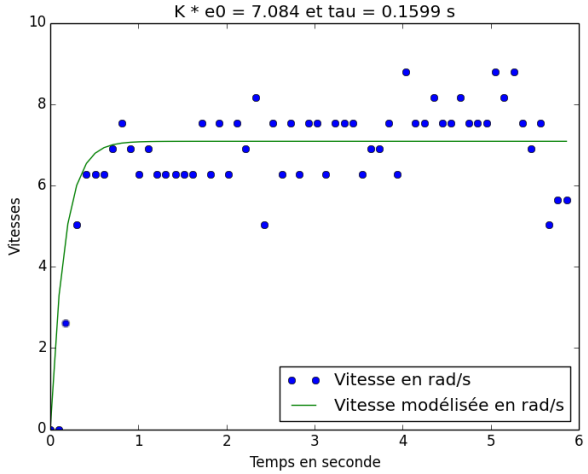


$$\Omega_m = cst - \frac{C_{r0}}{J_{lâcher1}}.t$$

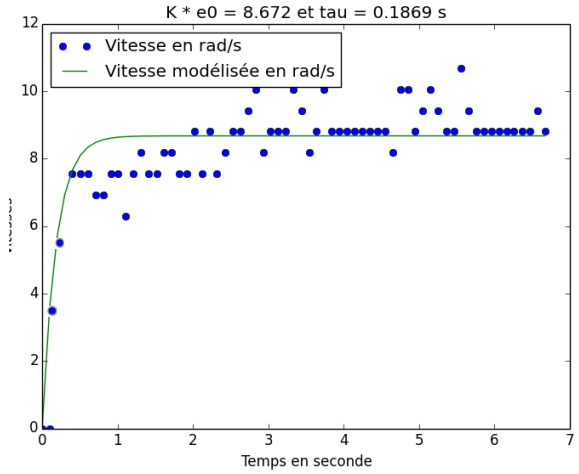
Valeur mesurée (moyenne) : $J_{lâcher} = 6.10^{-4} kg.m^2$

- Moment d'inertie expérimental : Réponse à un échelon de tension

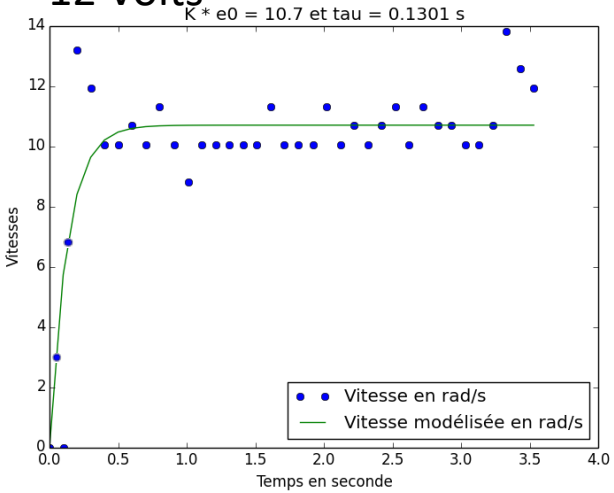
9 Volts

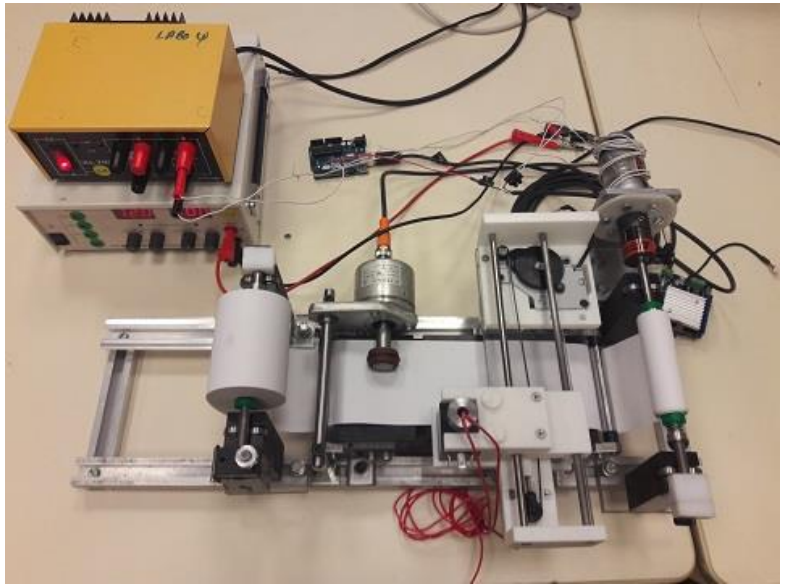


10 Volts



12 Volts





$$\tau = \frac{R.J_{échelon}}{R.f + K_e.K_c} \qquad J_{échelon} = \frac{\tau.(K_e.K_c + R.f)}{R}$$

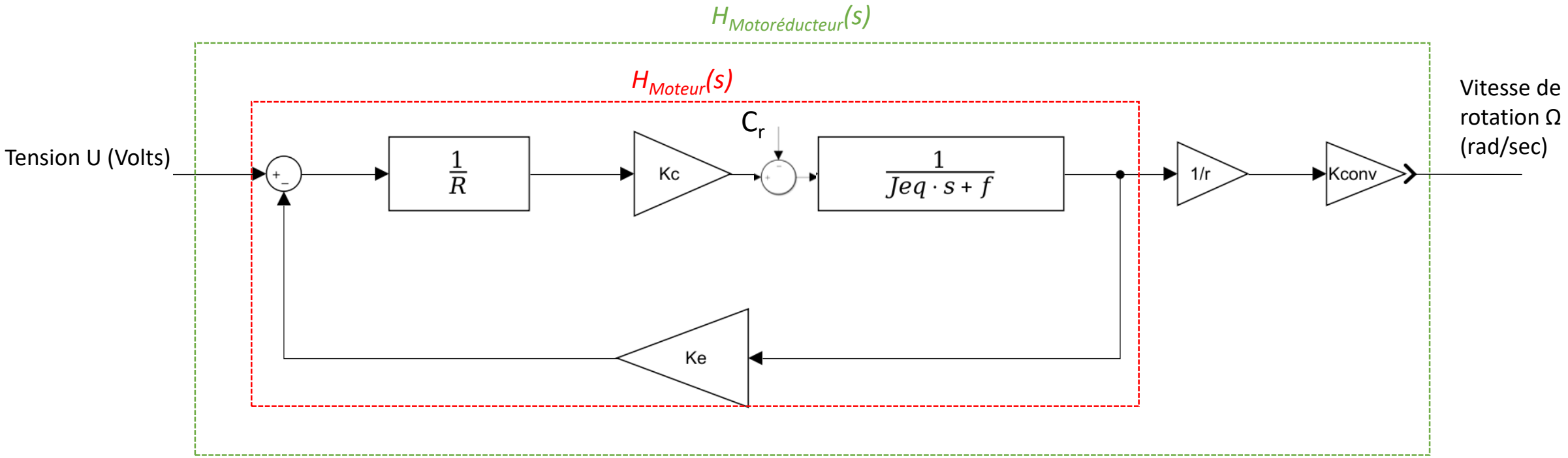
Valeur moyenne : $J_{échelon} = 9.5.10^{-6} \quad kg.m^2$

Confrontation des différents résultats concernant le moment d'inertie

- Essai en lâcher $J_{lâcher} = 6.10^{-4} kg.m^2$
- Réponse à un échelon de tension $J_{échelon} = 9.52.10^{-6} \quad kg.m^2$
- Par le calcul à t=0s $J_{théorique} = 1{,}1.10^{-5} \quad kg.m^2$
- Valeur finalement retenue $J_{eq} = 10^{-5} \quad kg.m^2$

Modélisation du système

Schéma bloc

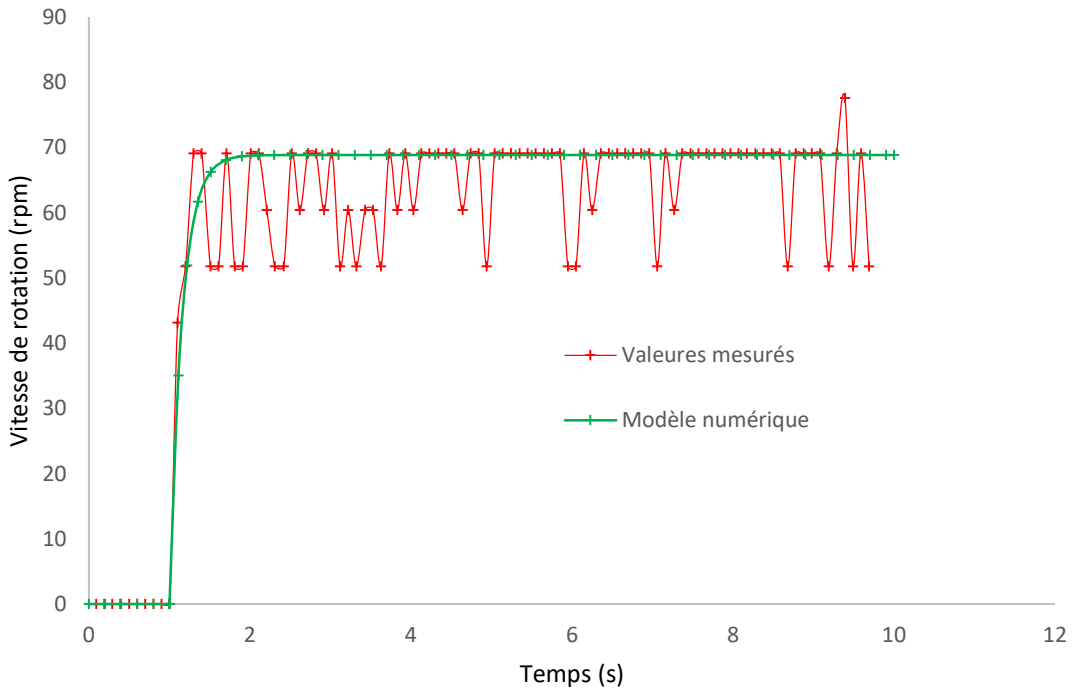


$$H_{Motoréducteur}(s) = H_{Moteur}(s).\frac{Kconv}{r} = \frac{\frac{Kc.Kconv}{(R.f + Ke.Kc).r}}{1 + \frac{R.Jeq}{R.f + Ke.Kc} \times s}$$

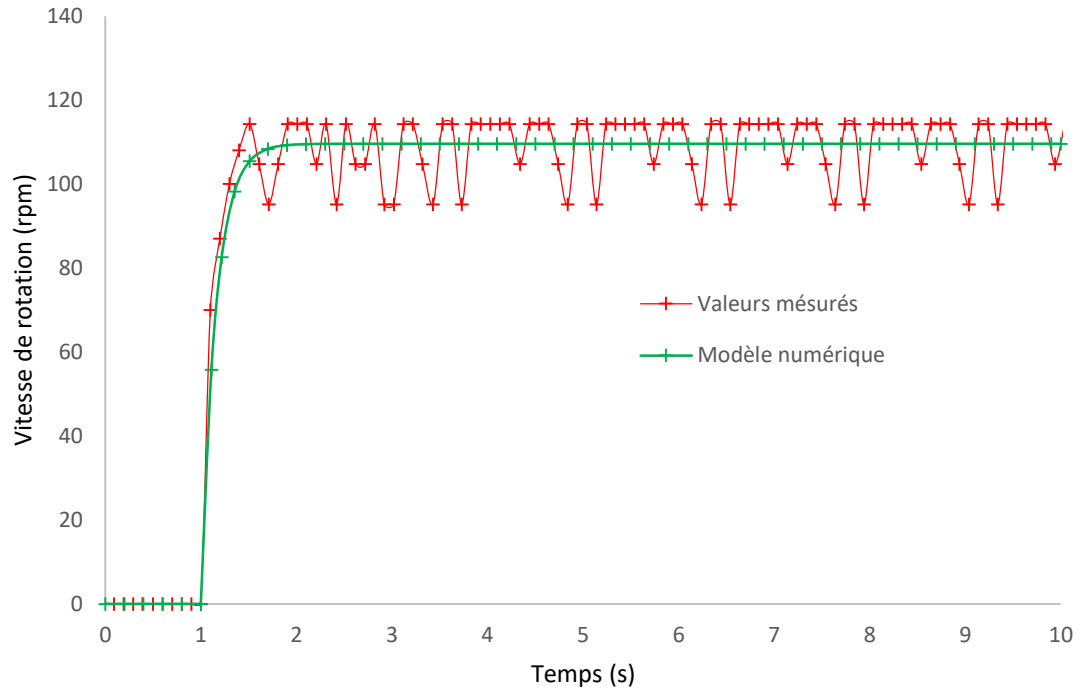
Comparaison théoriques/expérimentale

- Comparaison des réponses théoriques et expérimentales à un échelon de tension :

Réponse a un échelon de 6V

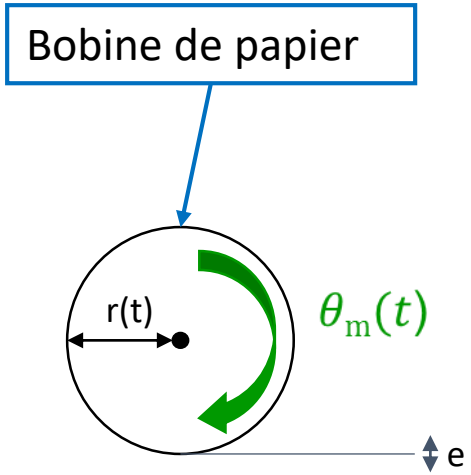


Réponse a un echelon de 9V

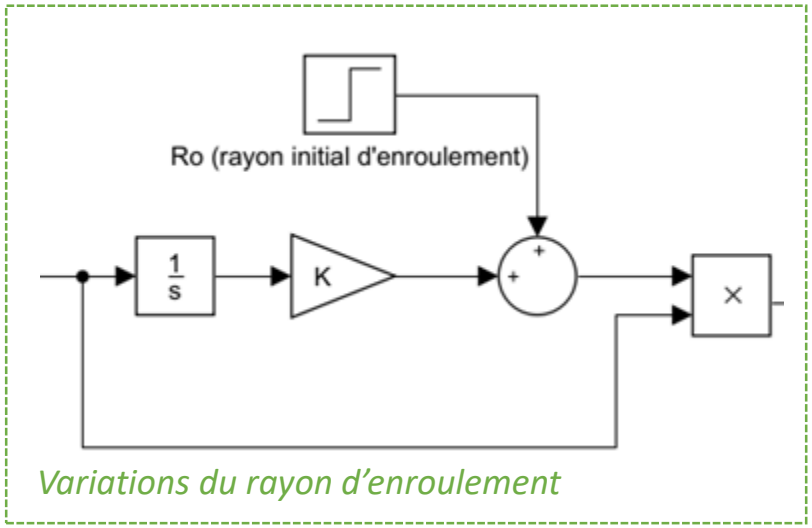


Validation du modèle

Variation du rayon d'enroulement



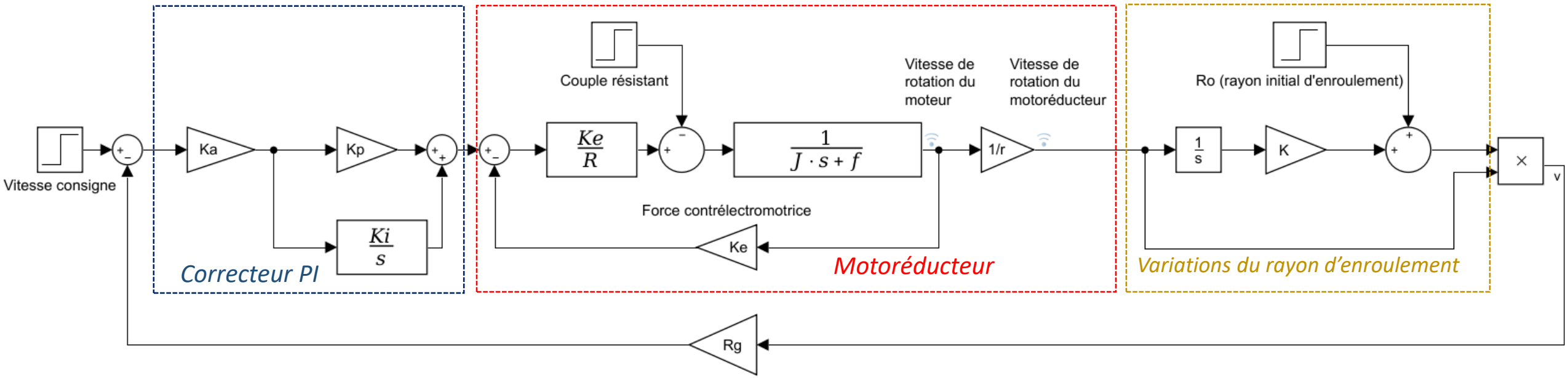
$$r(t) = r_0 + \frac{\theta_m(t)}{2\pi} e$$



Variations du rayon d'enroulement

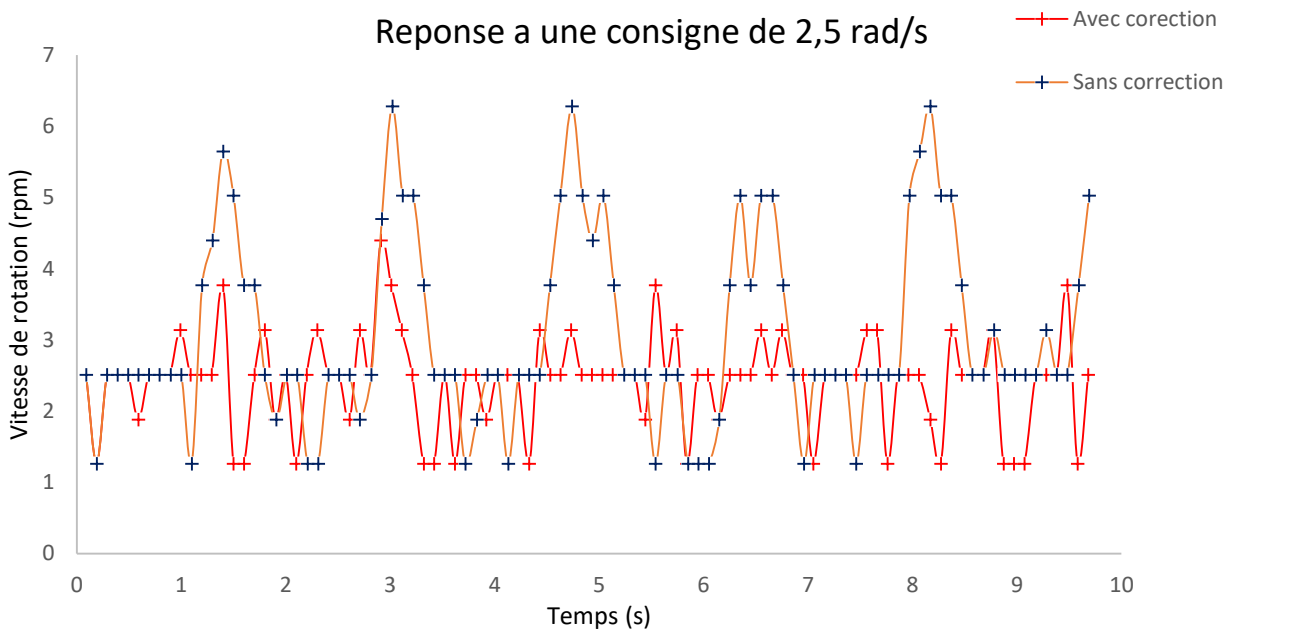
Mise en place de l'asservissement

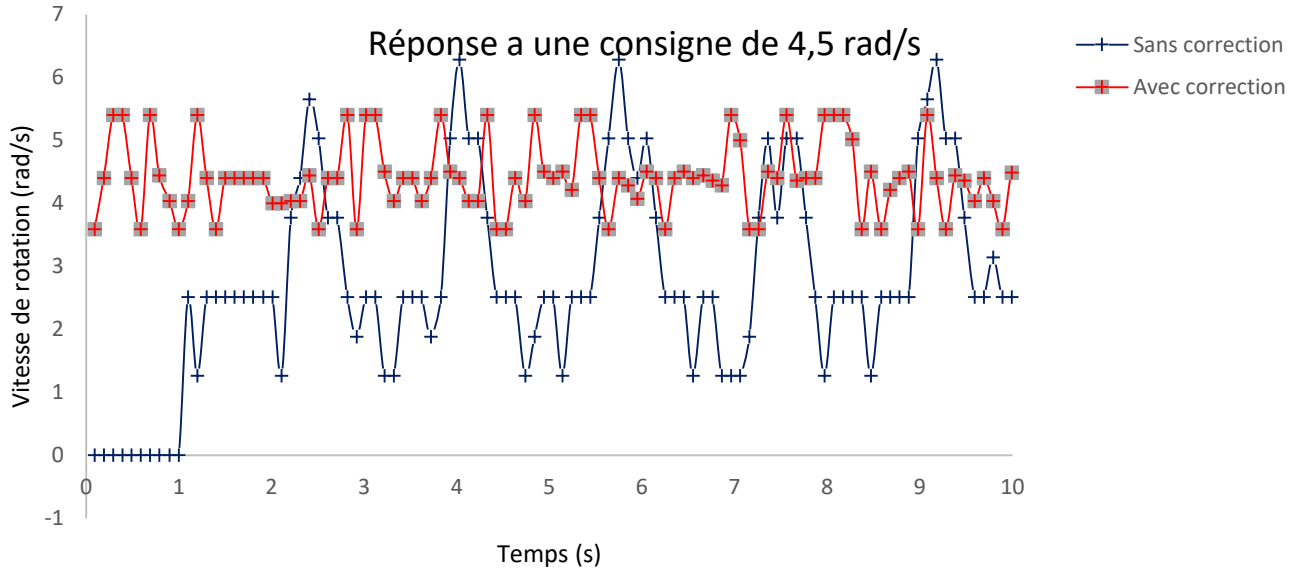
Schéma bloc du système avec asservissement



Mise en place de l'asservissement

Paramètres théoriques de l'asservissement



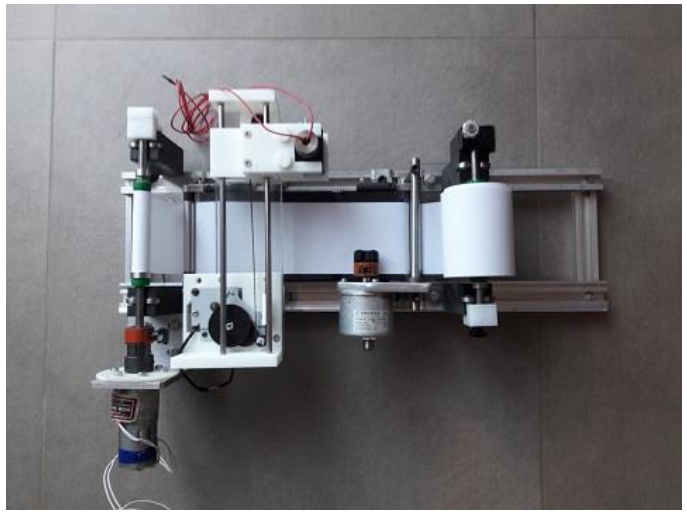


Support d'enroulement non cylindrique

Conclusion

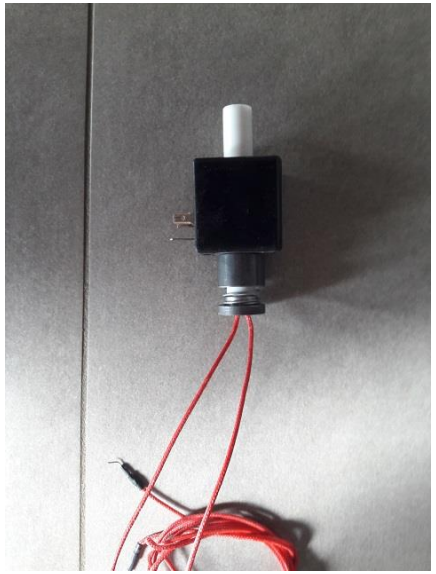
Bilan du TIPE:

- Prototype d'un orgue de barbarie électronique conçu
- Modèle du système réalisé
- Asservissement mis en place



Pour aller plus loin :

- Ecriture des partitions
- Lecture des partitions



Annexe 1 : Programme asservissement

```
#include <FlexiTimer2.h>
#include <digitalWriteFast.h>

// Codeur incrémental
#define codeurPinA 2
//#define codeurPinB 18
#define codeurInterruptionA digitalPinToInterrupt(codeurPinA)
//#define codeurInterruptionB digitalPinToInterrupt(codeurPinB)
volatile long ticksCodeurOld = 0;
volatile long ticksCodeurCumul = 0;
volatile long front = 0;

// Moteur CC
#define directionMoteurA 8
#define pwmMoteurA 9

// Cadence d'envoi des données en ms
#define TSDATA 100
unsigned long tempsDernierEnvoi = 0;
unsigned long tempsCourant = 0;

// Cadence d'échantillonnage en ms
#define CADENCE_MS 10
volatile double dt = CADENCE_MS / 1000.;
volatile double temps = -CADENCE_MS / 1000.;

// Commande
volatile double consigne = 3; // 8 Rad/sec au codeur soit 0.1 m/s sur le papier avec Rg=1.25cm
volatile double tensionAlim = 12.;
volatile double teta;
volatile double commande;
volatile double commandeSat;
volatile double saturation;
volatile double omega;
volatile double omegaPrec;
volatile double P_x = 100.;
volatile double I_x = 0.;

// parametre
volatile int count = 500;
volatile double ratio = 1;
volatile double pi = 3.14;
// radian : pi = 3.141592
// degre  : pi = 180.
```

```
void setup() {
  // Codeur incrémental
  pinMode(codeurPinA, INPUT);      // entrée digitale pin A codeur
  //pinMode(codeurPinB, INPUT);      // entrée digitale pin B codeur
  digitalWrite(codeurPinA, HIGH);  // activation de la résistance de pullup
  //digitalWrite(codeurPinB, HIGH);  // activation de la résistance de pullup
  attachInterrupt(codeurInterruptionA, GestionInterruptionCodeurPinA, CHANGE);
  //attachInterrupt(codeurInterruptionB, GestionInterruptionCodeurPinB, CHANGE);
  // HIGH   : niveau haut (sur certains modeles uniquement)
  // LOW    : niveau bas
  // CHANGE : front montant et descendant
  // RISING : front montant
  // CHUTE  : front descendant
  front = 2;

  // Moteur CC
  pinMode(directionMoteurA, OUTPUT);
  pinMode(pwmMoteurA, OUTPUT);

  // Liaison série
  Serial.begin(9600);
  Serial.flush();

  // Compteur d'impulsions de l'encodeur
  ticksCodeurOld = 0;
  ticksCodeurCumul = 0;

  // La routine isrt est exécutée à cadence fixe
  FlexiTimer2::set(CADENCE_MS, 1 / 1000., isrt); // résolution timer = 1 ms
  FlexiTimer2::start();
}

void loop() {
  // Ecriture des données sur la liaison série
  ecritureData();
}

void isrt() {
  //double Kp = 0.05;
  //double Ki = 0.12;

  double Kp = 4;
  double Ki = 0.1;
```

Annexe 2 : Programme asservissement

```
// Nombre de ticks codeur depuis la dernière fois
int codeurDeltaPos;
codeurDeltaPos = ticksCodeurCumul - ticksCodeurOld;
ticksCodeurOld = ticksCodeurCumul;

// Calcul de l'angle de rotation
teta = (2.*pi * (ticksCodeurCumul / front)) / (count * ratio); // en rad/deg/tr

// Calcul de la vitesse de rotation
omega = ((2.*pi * ((double)codeurDeltaPos / front)) / (count * ratio)) / dt; // en rad/s

temps += dt;

/******* Régulation********/
double Ti;

Ti = Ki / (Kp + 0.01);

// Ecart entre la consigne et la mesure
double ecart;
ecart = consigne - omega;

// Terme proportionnel
P_x = Kp * ecart;

// Terme intégral
I_x = I_x + Ki * dt * ecart;

// Calcul de la commande
commande = P_x + I_x;

// Application de la saturation sur la commande
if (commande > tensionAlim) {
  commandeSat = tensionAlim;
}
else if (commande < -tensionAlim) {
  commandeSat = -tensionAlim;
}
else {
  commandeSat = commande;
}
```

```
void CommandeMoteurA(double tension)
{
  int tensionInt = int(255 * tension / tensionAlim);
  // Saturation par sécurité
  if (tensionInt >= 255) {
    tensionInt = 255;
    saturation = 1;
  }
  else if (tensionInt <= -255) {
    tensionInt = -255;
    saturation = -1;
  }
  else {
    saturation = 0;
  }
  // Commande PWM
  if (tensionInt >= 0) {
    digitalWrite(directionMoteurA, HIGH);
    analogWrite(pwmMoteurA, tensionInt);
  }
  if (tensionInt < 0) {
    digitalWrite(directionMoteurA, LOW);
    analogWrite(pwmMoteurA, -tensionInt);
  }
}

void GestionInterruptionCodeurPinA()
{
  //if (digitalReadFast2(codeurPinA) == digitalReadFast2(codeurPinB)) {
  // ticksCodeurCumul--;
  //}
  //else {
    ticksCodeurCumul++;
  //}
}

/*void GestionInterruptionCodeurPinB()
{
  if (digitalReadFast2(codeurPinA) == digitalReadFast2(codeurPinB)) {
    ticksCodeurCumul++;
  }
  else {
    ticksCodeurCumul--; }}*/
```