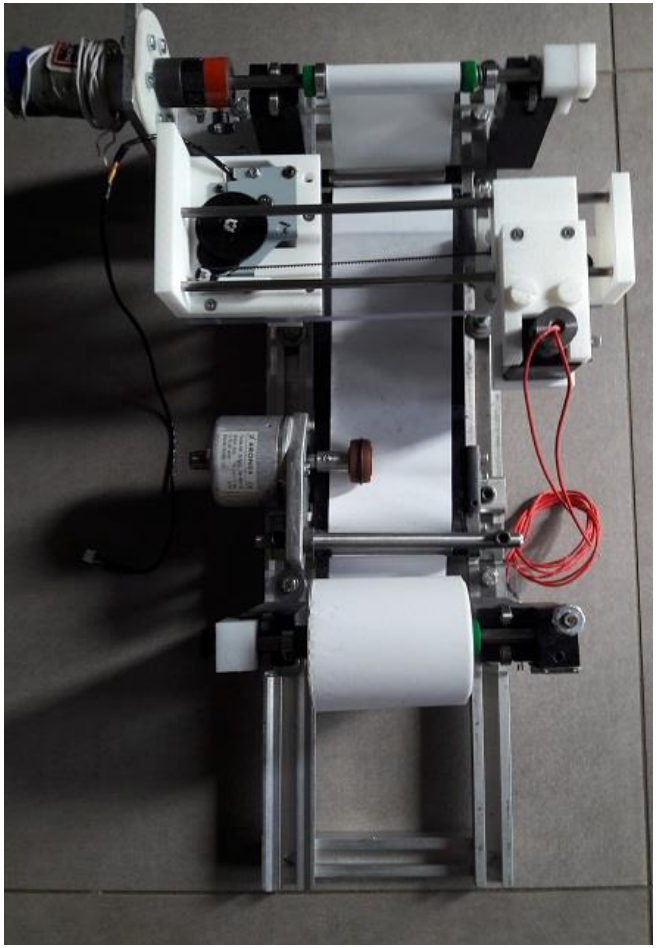
Orgue de barbarie électronique



I. Présentation du prototype

II. Identification des grandeurs du moteur:

III. Modèle du système

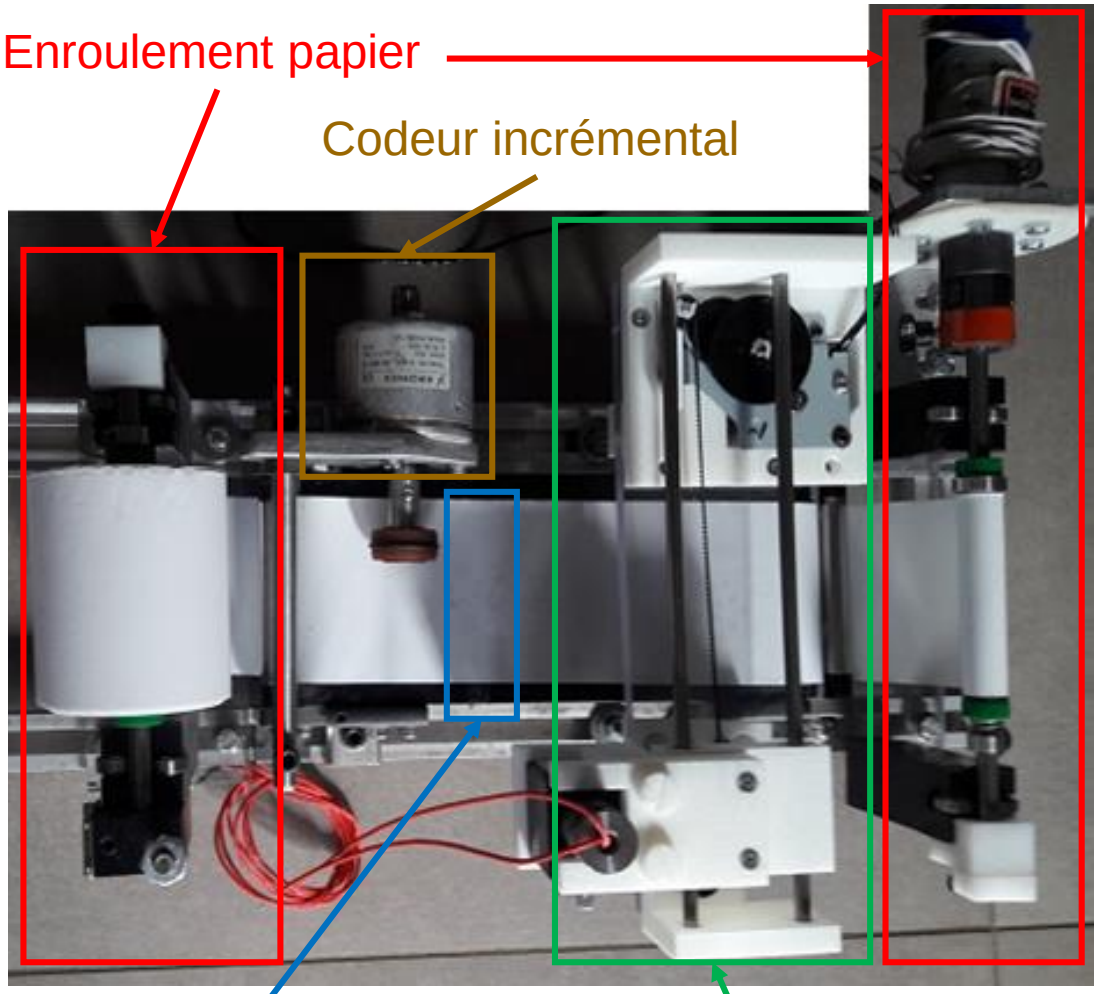
IV. Asservissement en vitesse du moteur

Problématique

- Créer un orgue de barbarie électronique

- Moderniser l'appareil
- Réduire son encombrement
- Réduire le coût

Présentation du prototype



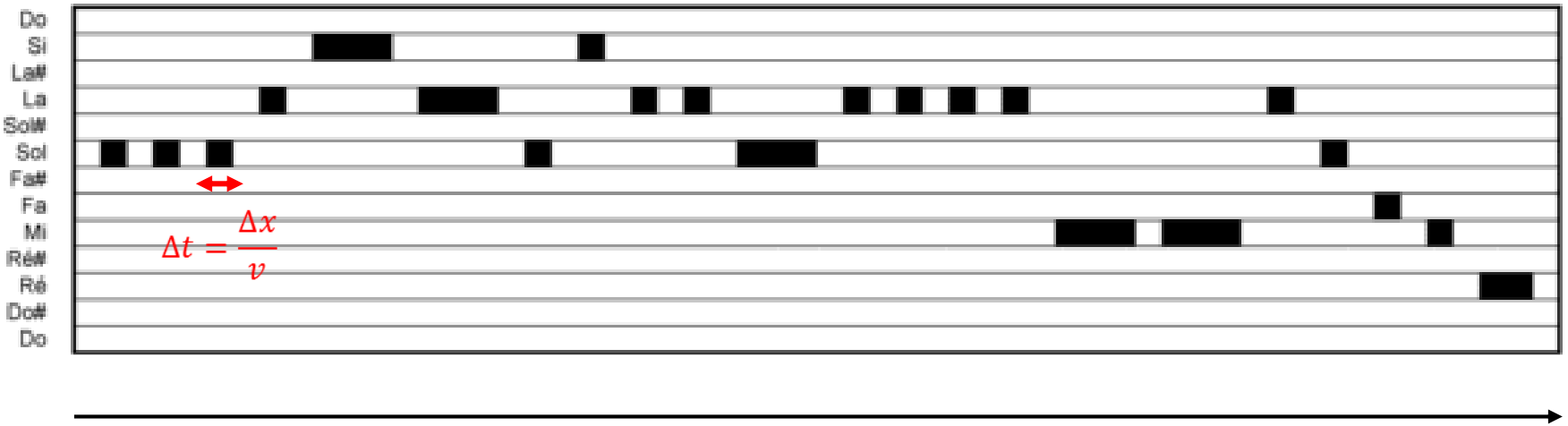
Le défilement du papier

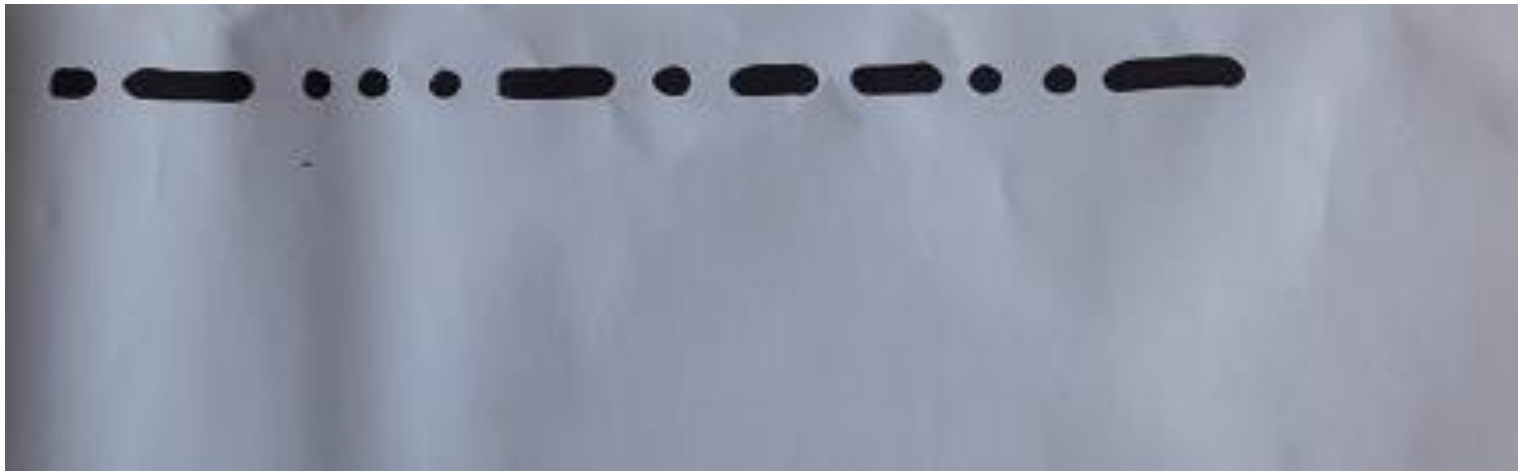
Rouleau de papier thermique

Moteur d'enroulement

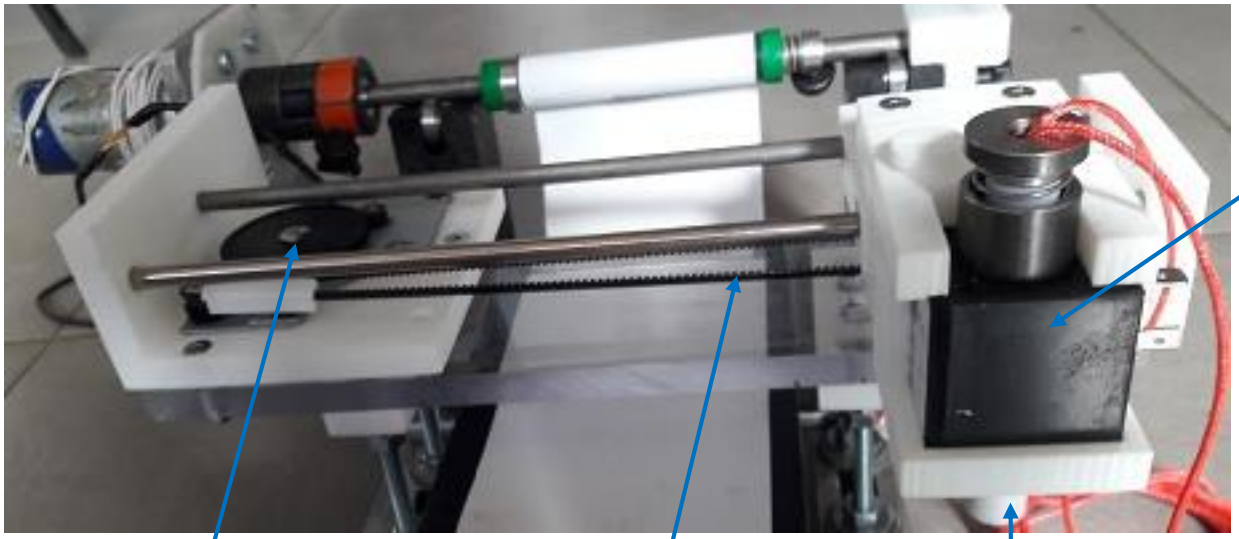
Support d'enroulement

La partition





L'écriture de la partition



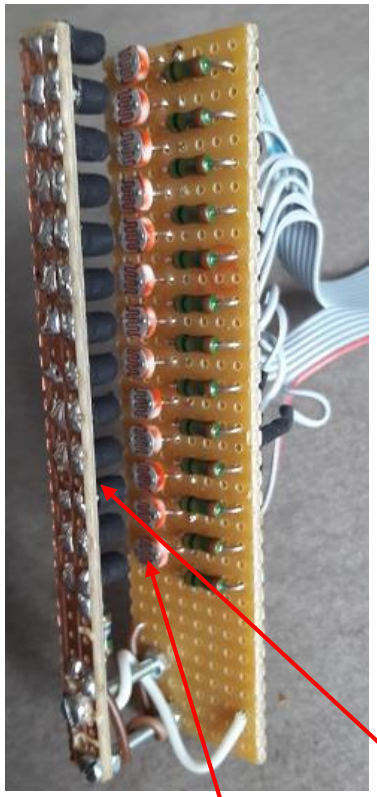
Bobine

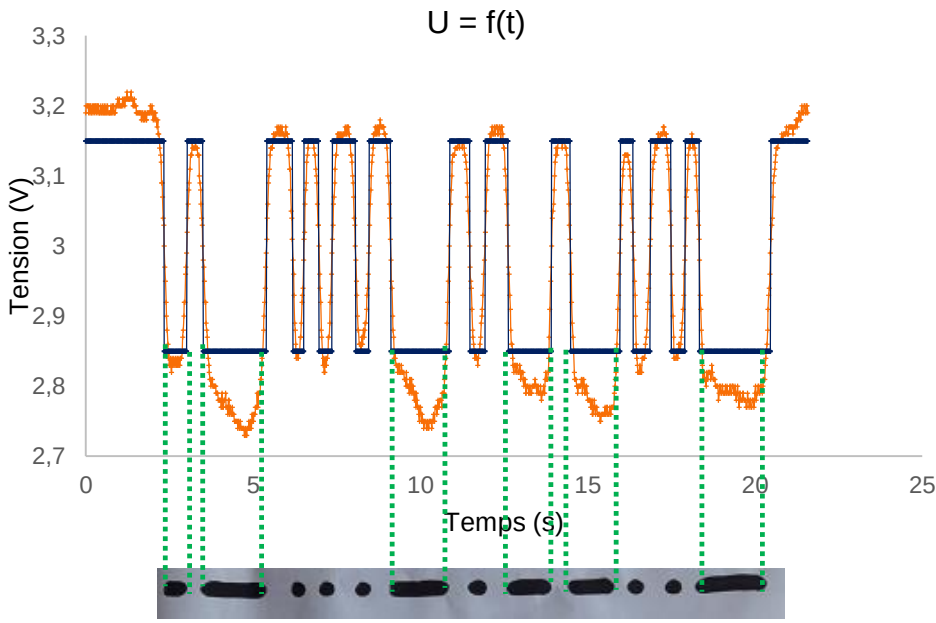
Moteur de déplacement transversal

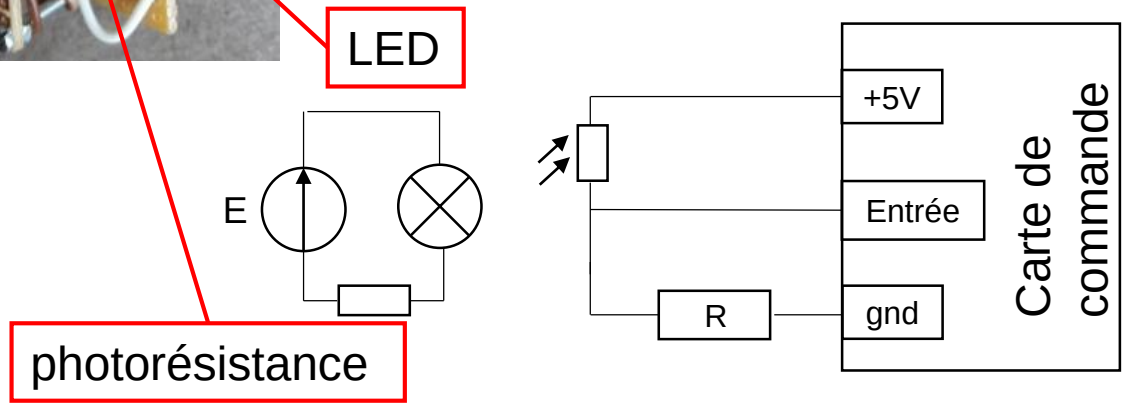
courroie

Résistance chauffante

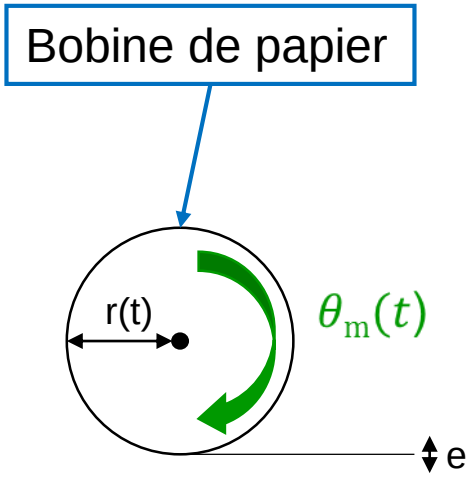
La lecture de la partition



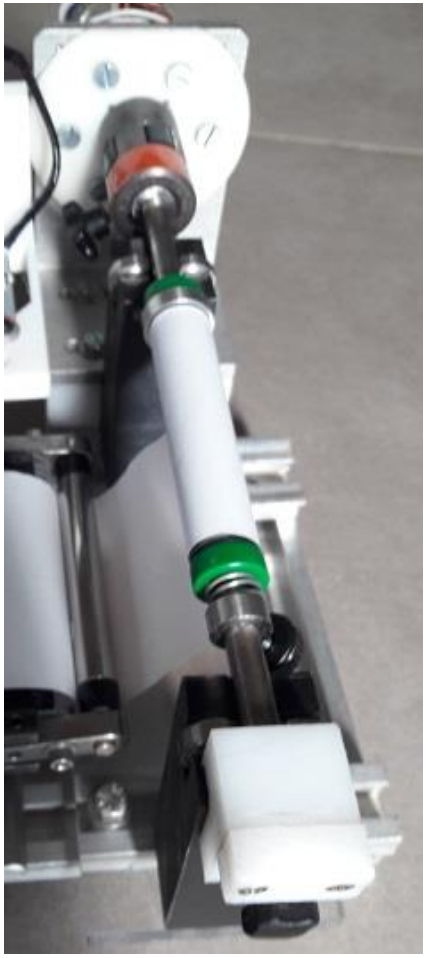


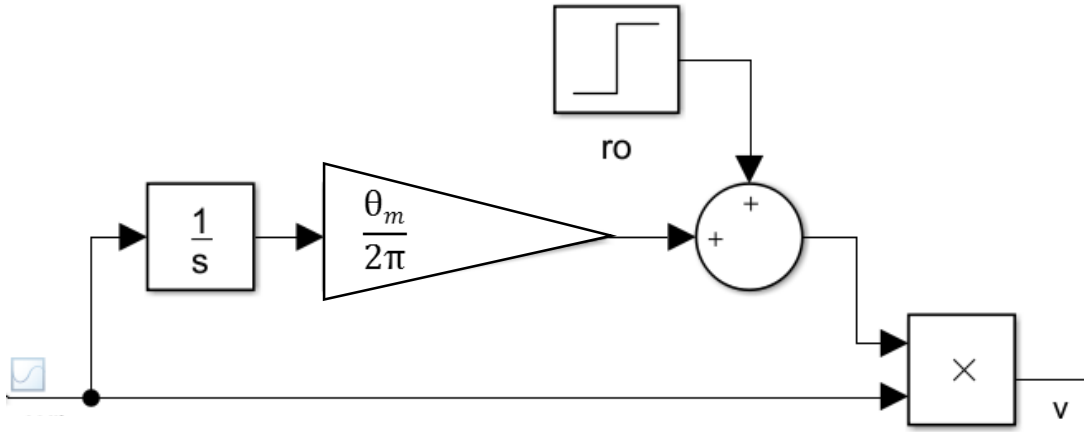


Variation du rayon



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



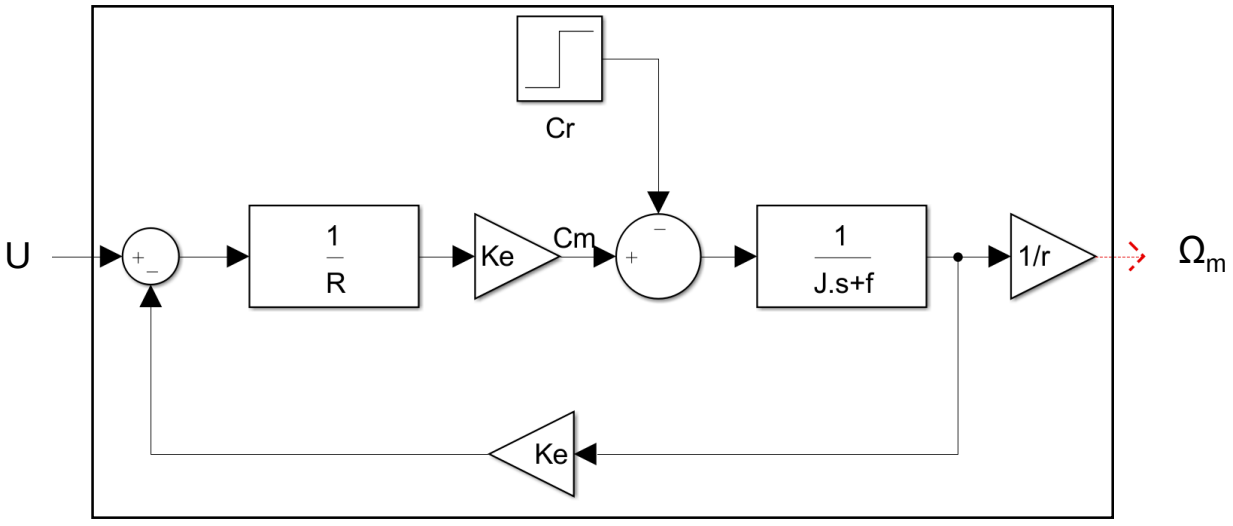


Moteur a courant continu

- Equation électrique: $U = E + R.I$
- Equation mécanique: $J\frac{d\,\Omega}{dt} = C_m - C_p$
- Fonction de transfert:

En régime permanant:

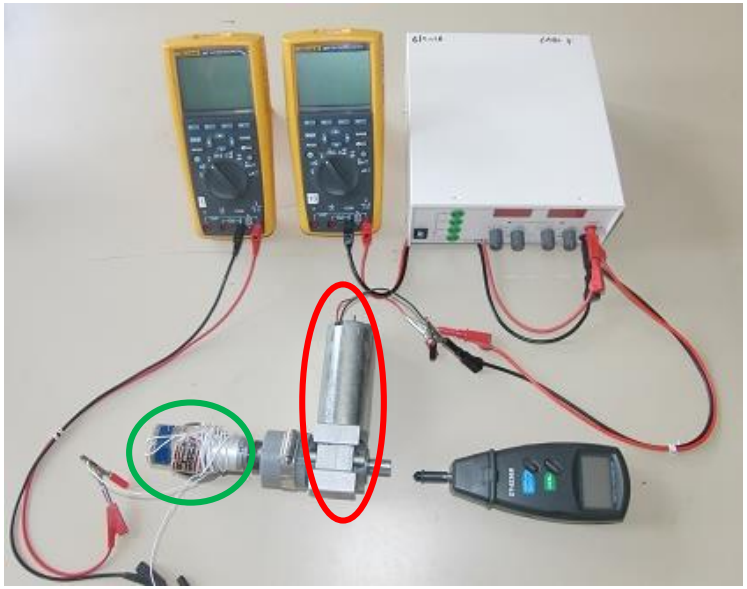
$$\Omega_m = \frac{K_e}{R.f + {K_e}^2}U - \frac{R}{R.f + {K_e}^2}C_r$$

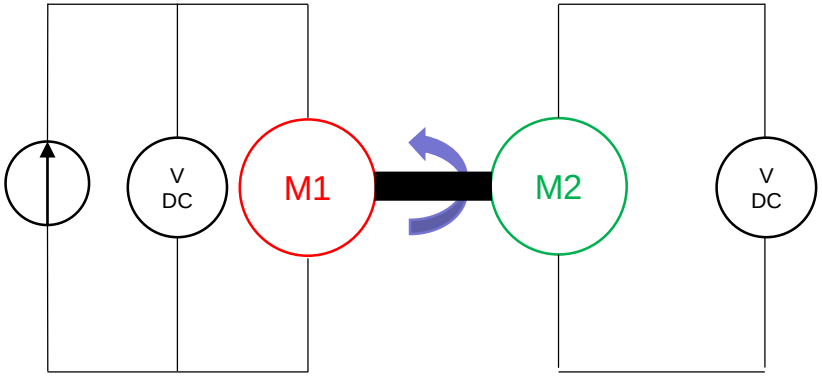


Modèle du moteur

Détermination expérimentale de k_e

Essai à vide

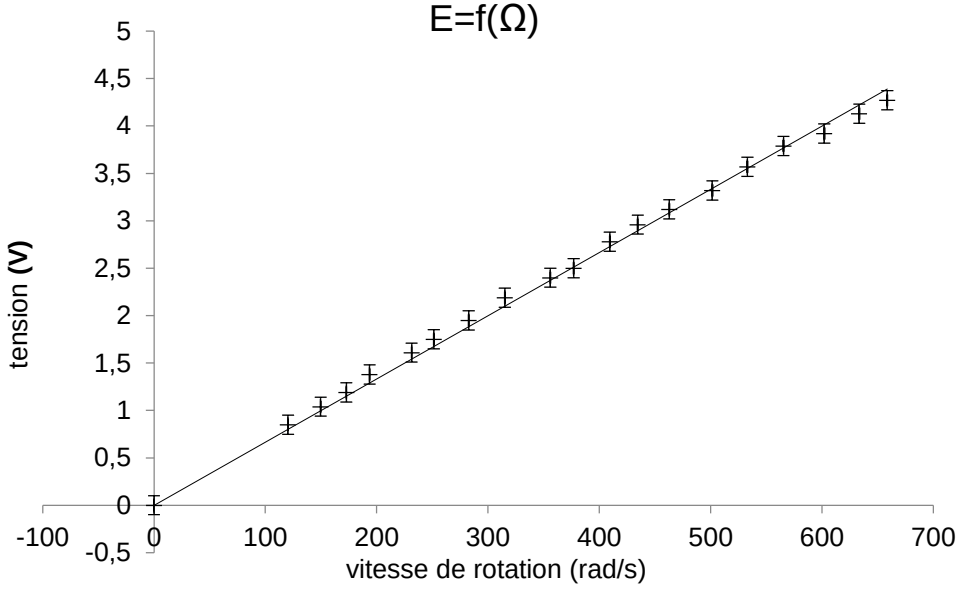




$E = K_e.\Omega$

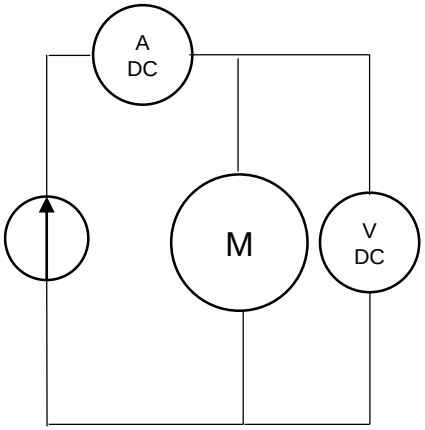
$K_e = 6{,}6.10^{-3}$ *V/rad/s*

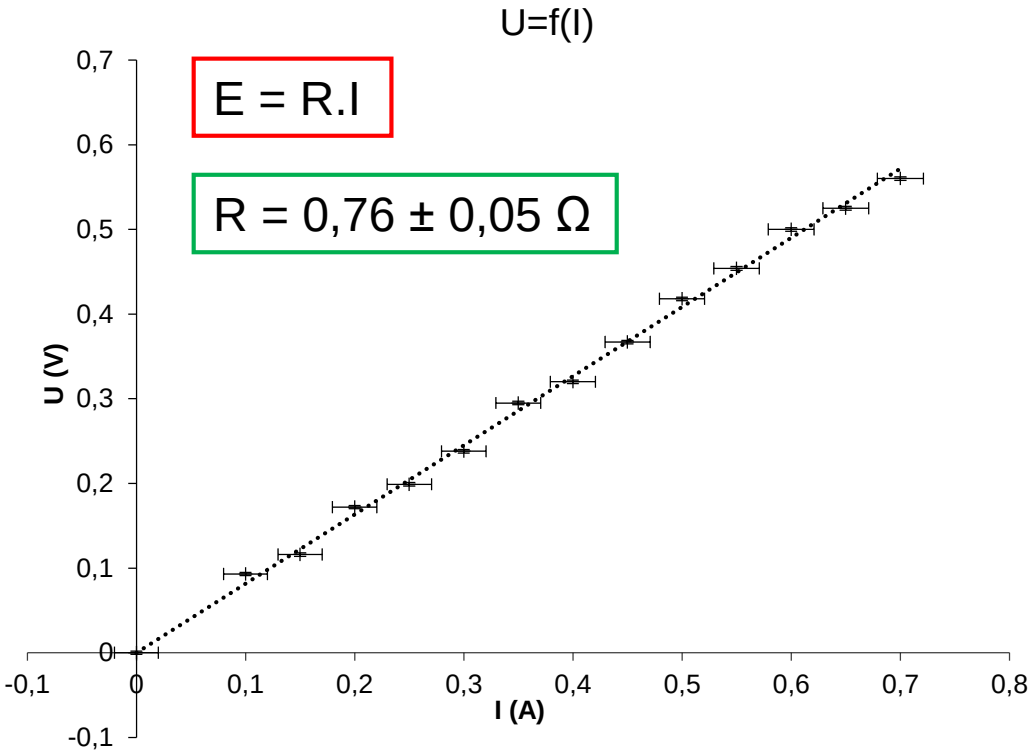
E=f(Ω)



Détermination expérimentale de la résistance de l'induit R

Essai à rotor bloqué

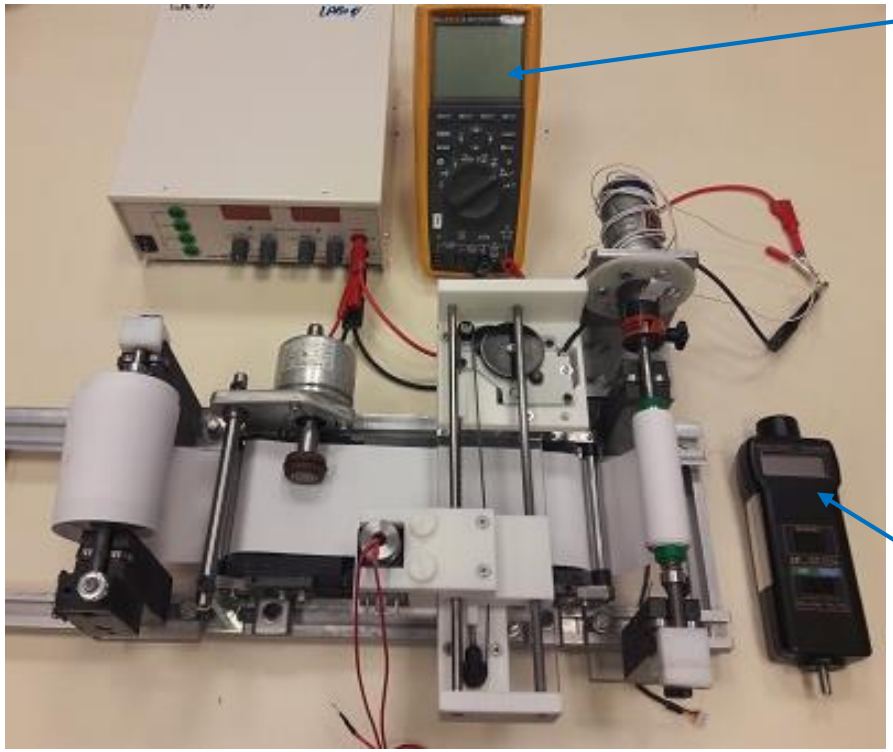


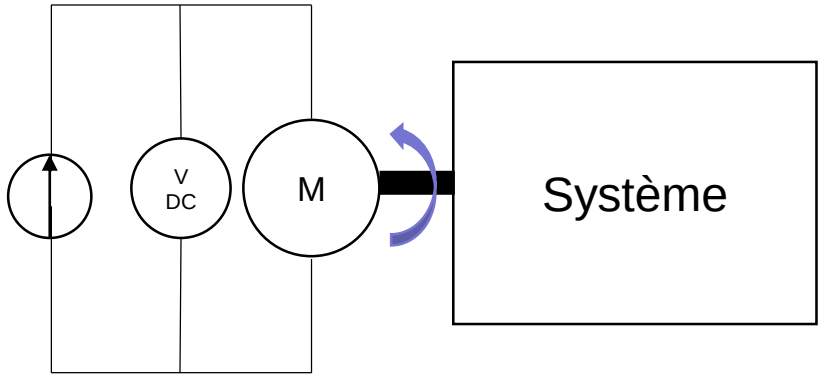


Mesure au ohmmètre:

Valeur min	Valeur Max	Valeur moyenne
R = 0,52 Ω	R = 1,16 Ω	R = 0,72 Ω

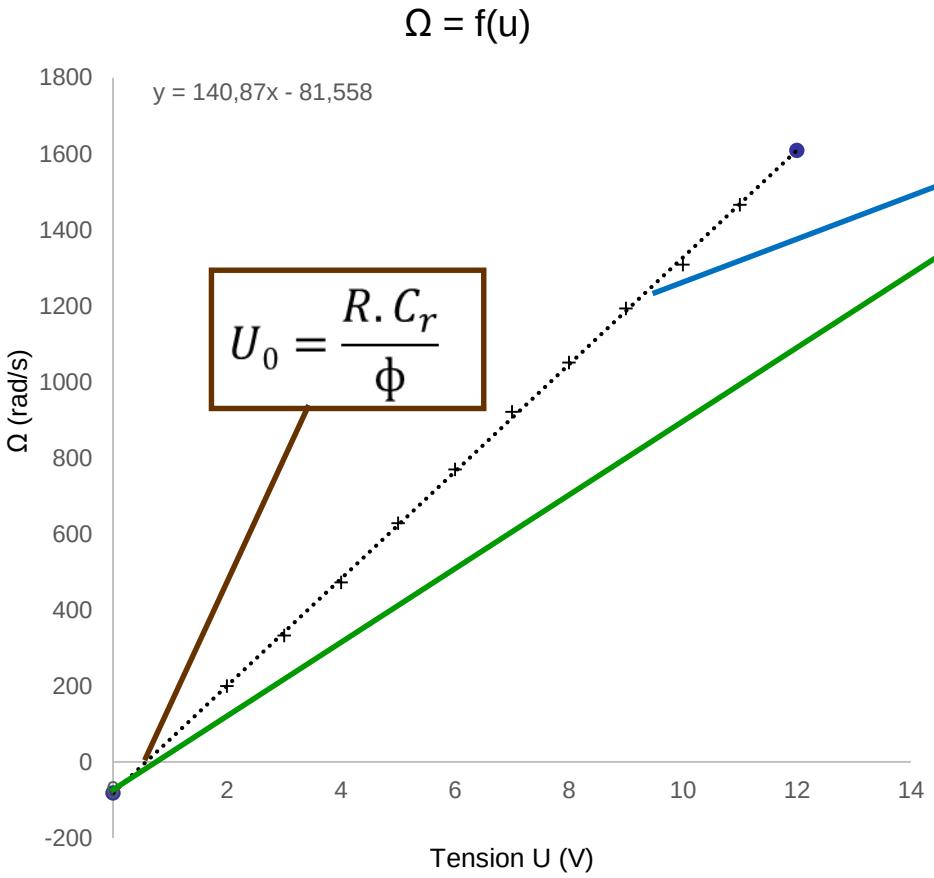
Détermination expérimentale du couple de frottement





Mesure de la vitesse en Régime permanant

Détermination expérimentale du couple de frottement



$$U_0 = \frac{R.C_r}{\phi}$$

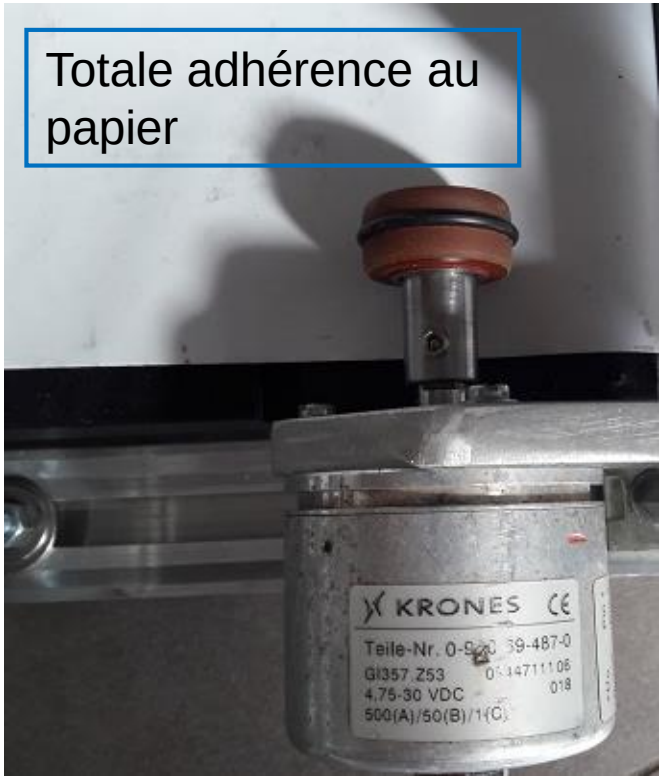
$$\Omega_m = \frac{K_e}{R.f + {K_e}^2} U - \frac{R}{R.f + {K_e}^2} C_r$$

Couple de frottements sec:
$C_r = 5.10^{-3}$ N.m

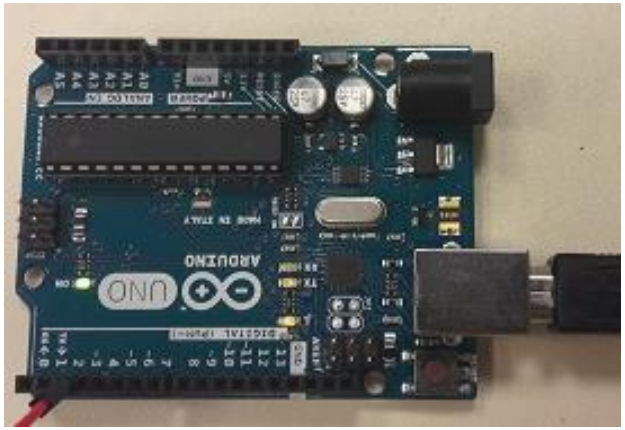
Frottements fluide:
$f = 3{,}8.10^{-6}$ kg/s

Graphiquement	Expérimentalement
$U_0 = 0{,}57$ V	$U_0 = 0{,}8$ V

Mesure de la vitesse



Codeur incrémental



Carte d'acquisition

Tachymètre

Détermination théorique du moment d'inertie J

Energies cinétiques:

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

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

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

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

Moment d'inertie équivalent ramené au moteur:

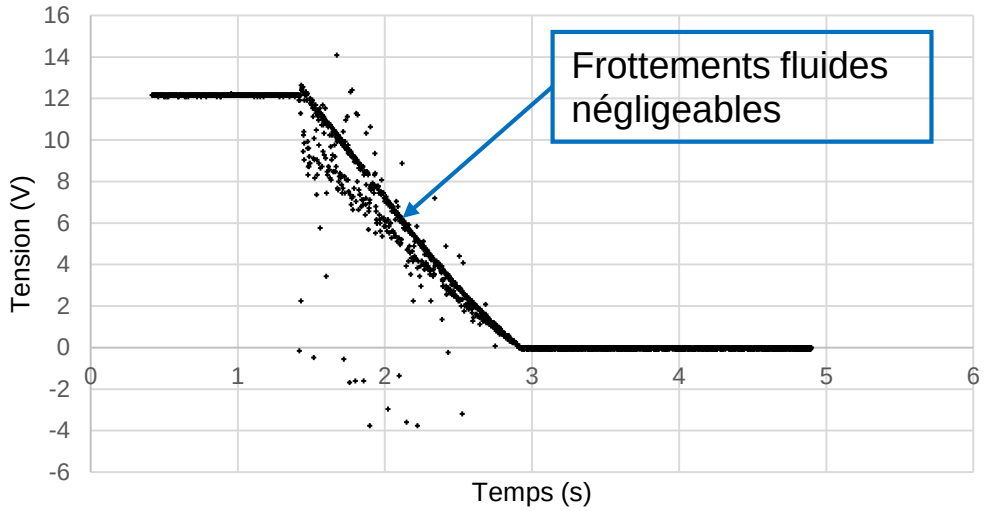
$$J_{eq} = J_{moteur} + J_{codeur} + 2J_{axe} + J_{papier\_enroulé}(t) + J_{papier\_deroulé}(t)$$

$J_{eq\ th} = 1{,}1.10^{-5}$ kg.m^2

Détermination expérimentale du moment d'inertie

Essai en lâché

U = f(t)



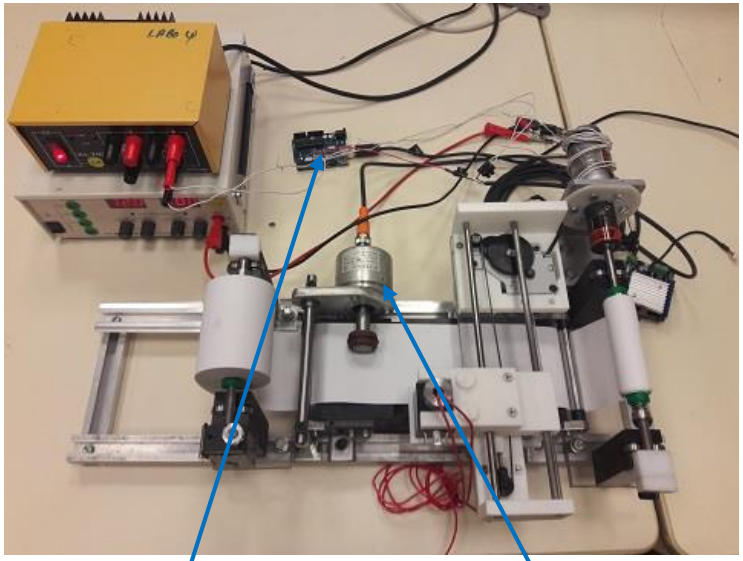
$$J\frac{d\,\Omega}{dt} = -Cr$$

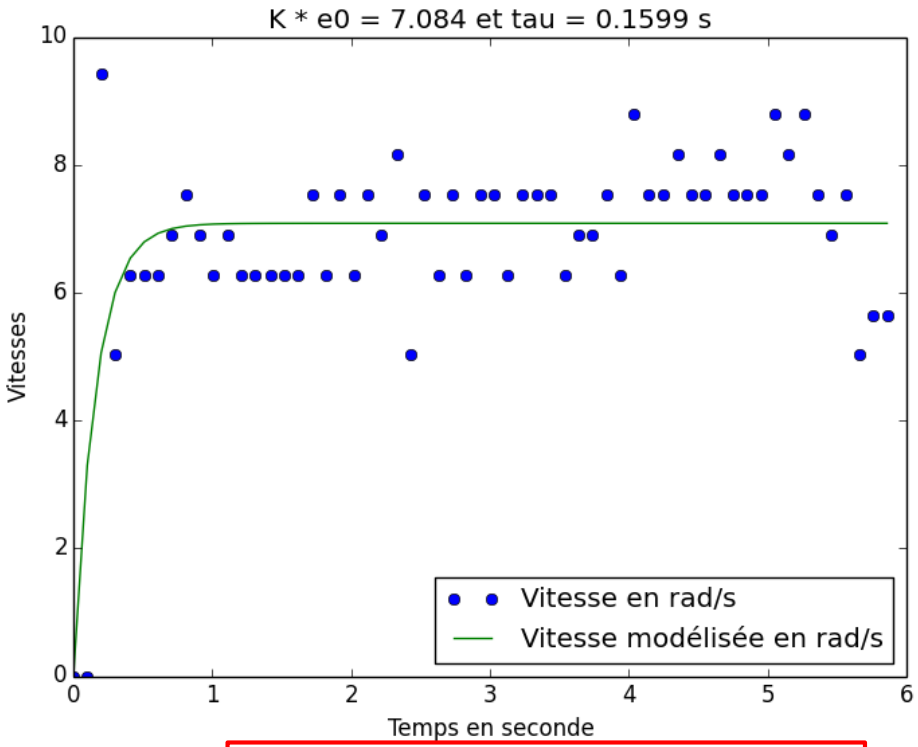
$$U = E = K_e.\Omega$$

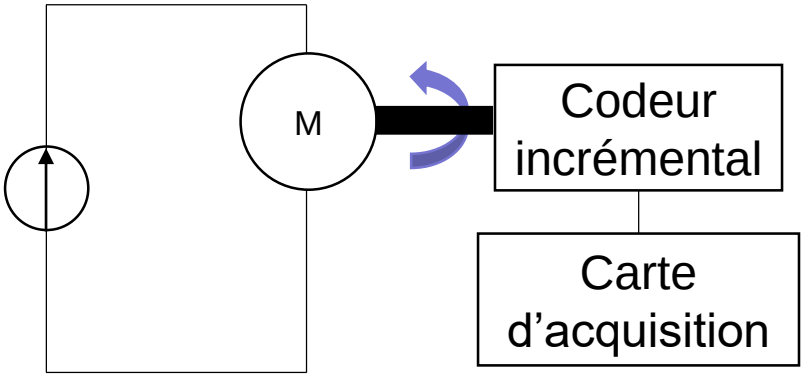
$J = 6{,}5.10^{-4}$ kg.m^2

Détermination expérimentale du moment d'inertie

Réponse du système à un échelon de tension







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

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

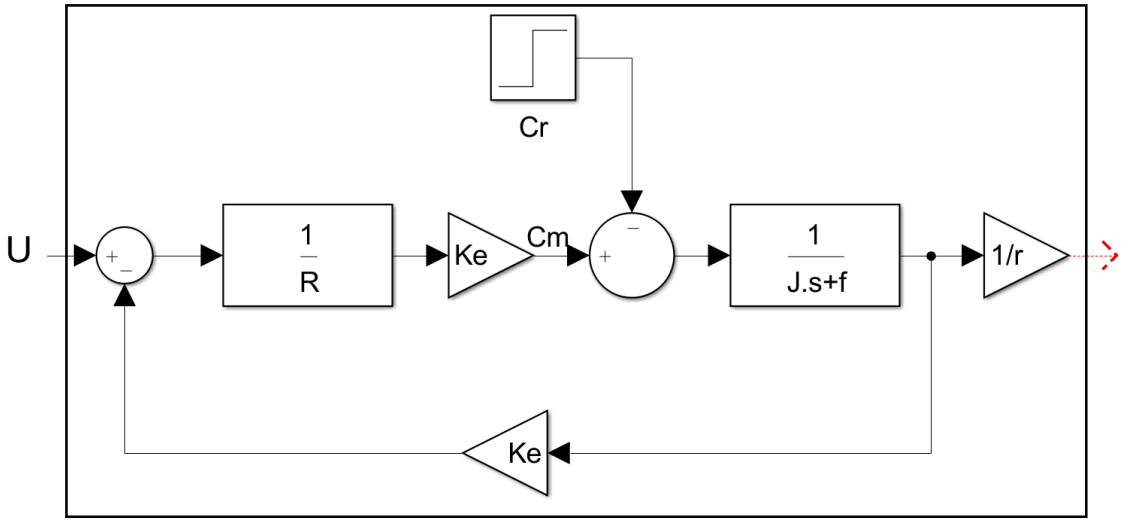
$$J = 9{,}5.10^{-6} Kg.m^2$$

Synthèse sur le moment d'inertie

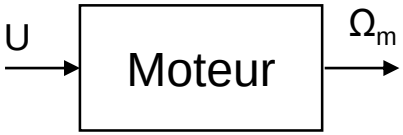
Théorie	$J_{eq} = 1{,}1.10^{-5}$ Kg.m^2
Réponse à un échelon de tension	$J_{eq} = 9{,}5.10^{-6}$ Kg.m^2
Essai en lâché	$J_{eq} = 6{,}5.10^{-4}$ Kg.m^2

Valeur retenue: $J_{eq} = 10^{-5}$ kg.m^2

Modèle du système {Moteur, Réducteur}



Modèle du moteur

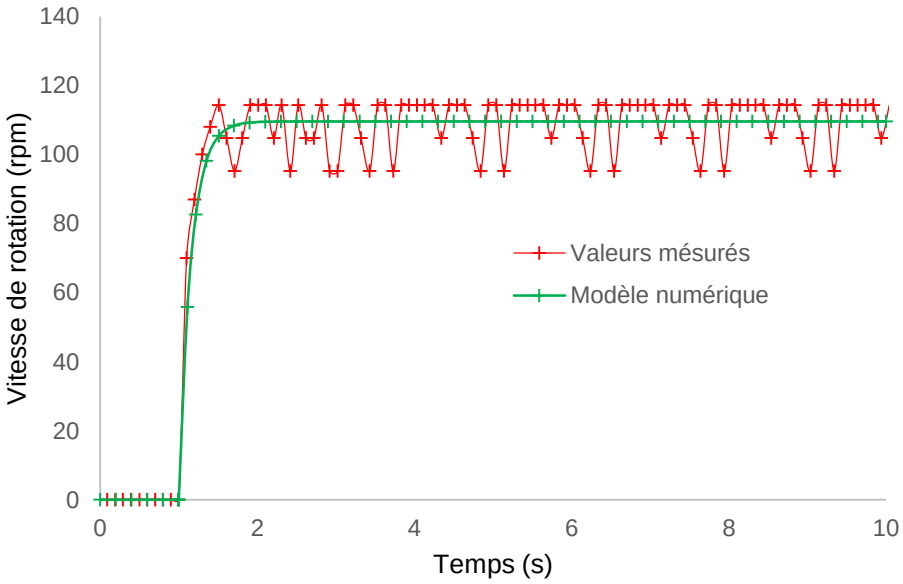


$$\Omega(p) = \frac{\frac{K_e}{Rf + {K_e}^2}}{1 + \frac{Rf}{Rf + {K_e}^2}p} U(p) - \frac{\frac{R}{Rf + {K_e}^2}}{1 + \frac{Rf}{Rf + {K_e}^2}p} Cr(p)$$

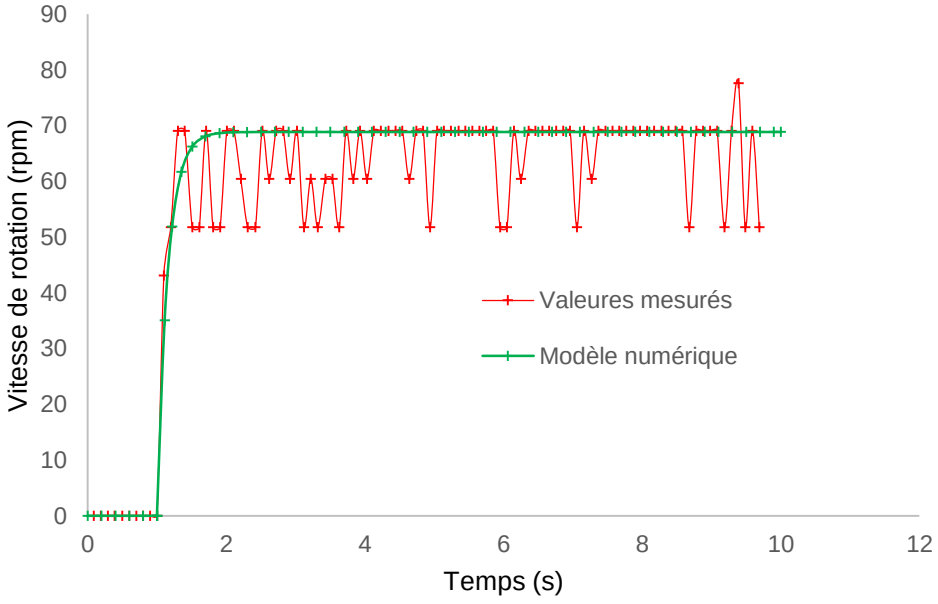
R = 0,76 Ω
$K_e = 6{,}6.10^{-3}$ V/rad/s
$Cr = 5.10^{-3}$ N.m
$f = 3{,}8.10^{-6}$ Kg/s
$J = 10^{-5}$ Kg.m²

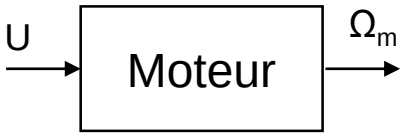
Validation du Modèle

Réponse a un echelon de 9V



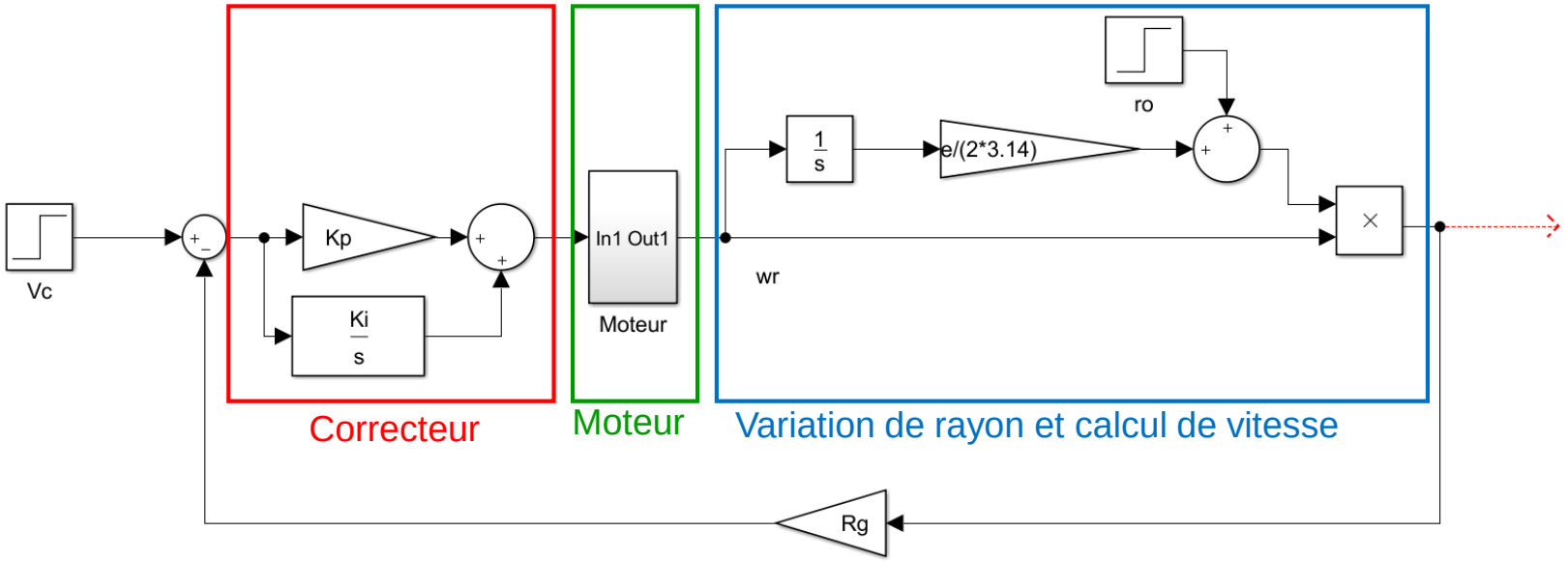
Réponse a un échelon de 6V



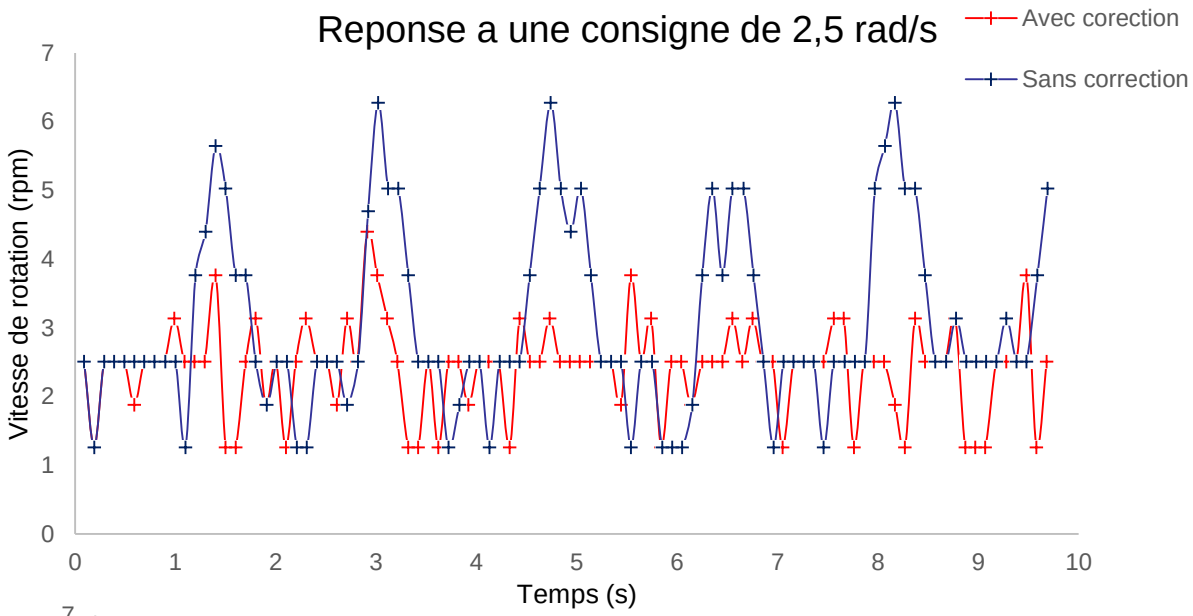


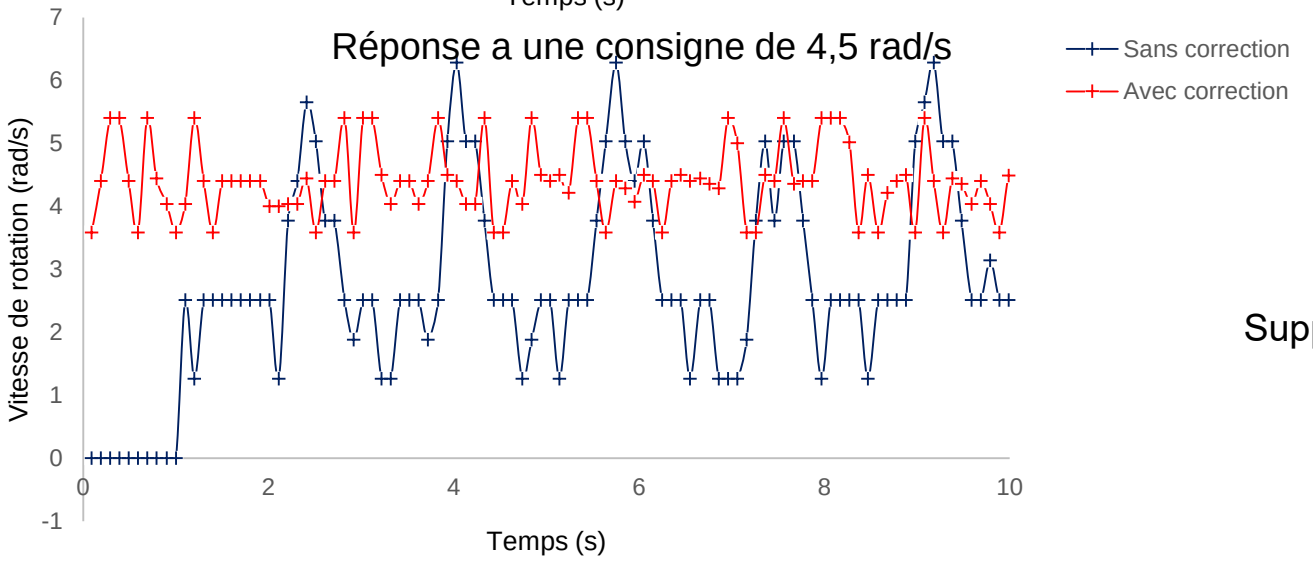
Modèle Validé

Asservissement du moteur en vitesse



Vérification de l'asservissement





Support d'enroulement non cylindrique

conclusion

- Asservissement en vitesse du moteur réussi.
- Projet ambitieux au départ.
- Intension de finir le prototype.

Moment d'inertie

- Energies cinétiques:

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

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

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

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

- Energies cinétiques exprimée avec la vitesse du moteur:

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

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

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

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

- Moment d'inertie équivalent en fonction du temps :

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

Annexe

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

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

// Moteur CC
#define directionMoteurA 9 // M1
#define pwmMoteurA 11

// 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 = 0.08; // m/s
volatile double tensionAlim = 12.;  //12V
volatile double teta;
volatile double commande;
volatile double commandeSat;
volatile double saturation;
volatile double omega;
volatile double omegaPrec;
volatile double P_x = 0.;
volatile double I_x = 0.;

// parametre
volatile int count = 500;
int Rg = 0.0125;  //m
volatile double ratio = 1/Rg;
volatile double pi = 3.14;}

void setup() {
  // Codeur incrémental
  pinMode(codeurPinA, INPUT);      // entrée digitale pin A codeur
  digitalWrite(codeurPinA, HIGH);  // activation de la résistance de pullup
  attachInterrupt(codeurInterruptionA, GestionInterruptionCodeurPinA,
CHANGE);
  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 = 1;
  double Ki = 0;

  // 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
```

Annexe

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

  temps += dt;

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

  Ti = Ki / (Kp + 0.01);

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

  // 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;
  }

  /******* Fin régulation PID ********/

  // Envoi de la commande au moteur
  CommandeMoteurA(commandeSat);}
void ecritureData(void) {
  // Ecriture des données en sortie tous les TSDATA millisecondes
  tempsCourant = millis();
  if (tempsCourant - tempsDernierEnvoi > TSDATA) {
    Serial.print(temps);
    Serial.print("\t");
    Serial.print(consigne);
    Serial.print("\t");
    Serial.print(teta);
    Serial.print("\t");
    Serial.print(commande);
    Serial.print("\t");
    Serial.print(saturation);
    Serial.print("\t");
    Serial.print(omega);
    Serial.print("\r\n");

    tempsDernierEnvoi = tempsCourant;}}

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); }}

void GestionInterruptionCodeurPinA()
{ ticksCodeurCumul++;}
```