Rotation d'un aimant dans un champ magnétique oscillant

1. Introduction

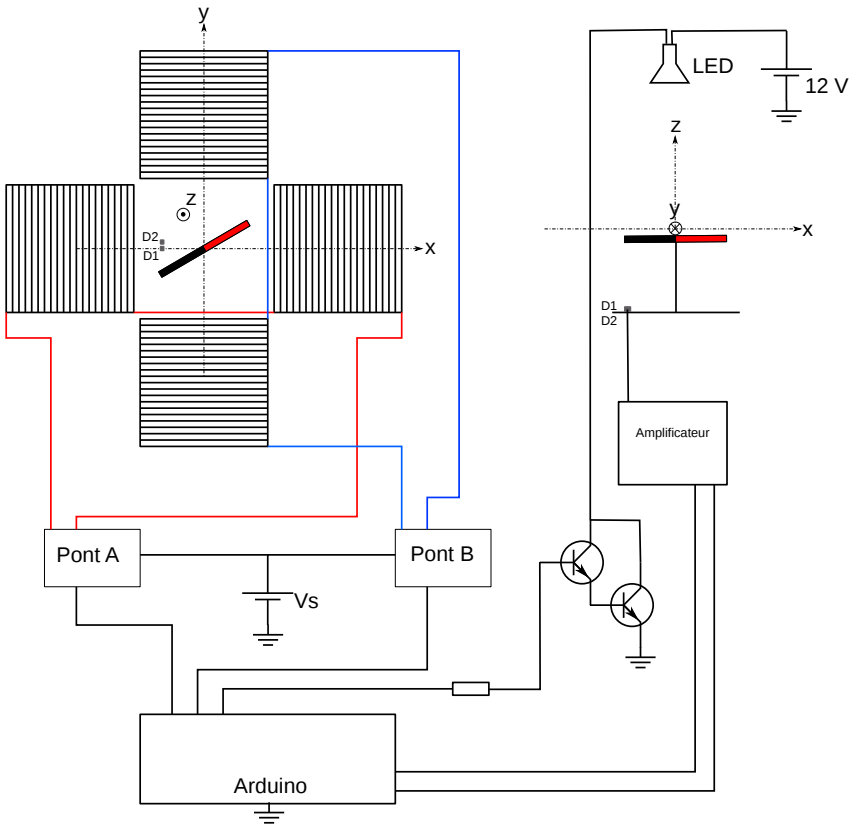
Ce document présente un dispositif expérimental permettant d'étudier le mouvement de rotation d'un barreau aimanté soumis à un champ magnétique. Le champ peut présenter différents modes d'oscillation : champ tournant, champ stationnaire superposé à un champ oscillant, etc.

Il permet d'étudier expérimentalement le problème du pendule forcé, du pendule forcé paramétrique, du moteur synchrone. Par ailleurs, le problème d'un aimant en rotation dans un champ tournant superposé à un champ stationnaire constitue un système dynamique non linéaire à trois degrés de liberté.

2. Dispositif expérimental

2.a. Vue d'ensemble

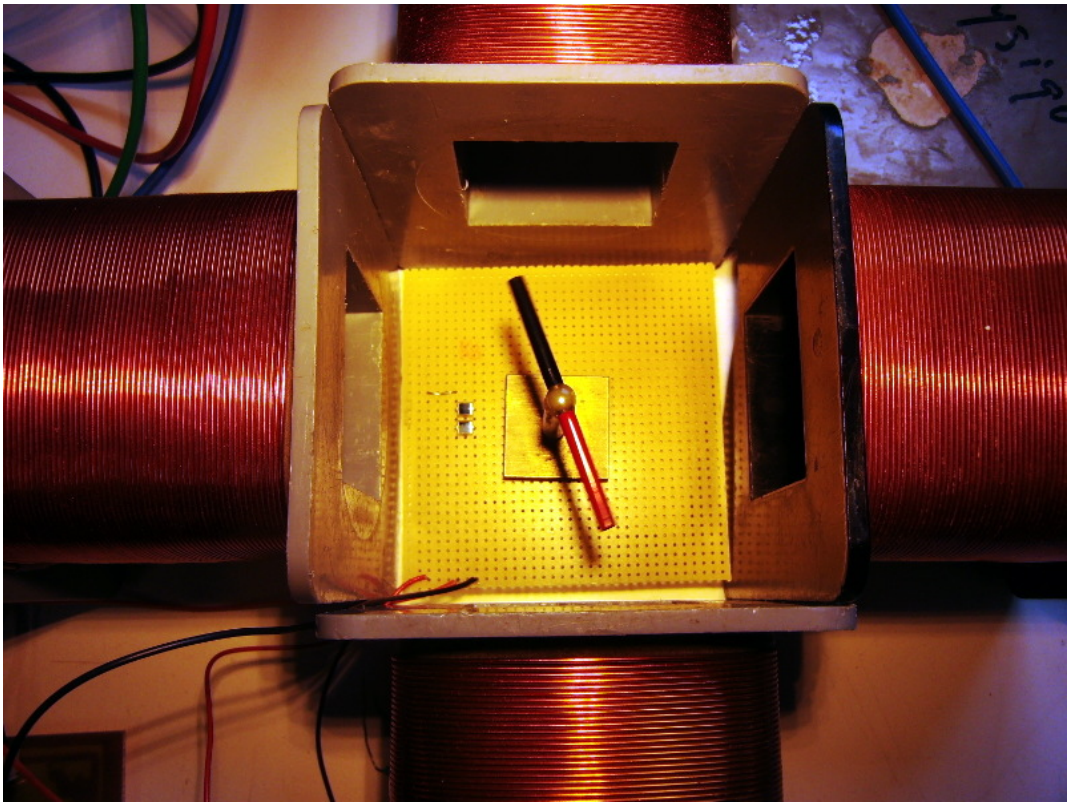
Le barreau aimanté (de longueur 6 cm) est monté sur un pivot vertical, constitué d'une aiguille sur laquelle le barreau repose. Le champ magnétique est produit par deux paires de bobines, chacune comportant 500 spires et d'auto-inductance $L = 11\ mH$. La zone comprise entre ces quatre bobines a une largeur de 9 cm. Le barreau est situé au centre de cette zone, son axe de rotation à l'intersection des deux axes des bobines. Au point central, la première paire produit la composante B_x du champ, la seconde paire produit la composante B_y.



Les deux bobines d'axe x sont alimentées par un pont en H à transistors MOSFET (pont A). Les deux bobines d'axe y sont alimentées par un second pont (pont B). Les deux ponts sont alimentés par une alimentation de laboratoire stabilisée en tension (Vs de 12 à 24 V), et pilotés par la carte arduino avec deux sorties PWM qui délivrent un signal à modulation de largeur d'impulsion (MLI). Ce dispositif d'alimentation des bobines est décrit en détail dans Onduleur diphasé MLI pour bobines.

Le barreau est éclairé par dessus par une lampe à LED fonctionnant sous 12 V et de puissance 5 W. Cette lampe est pilotée par une sortie de l'arduino via un transistor Darlington. L'éclairage peut être continu ou stroboscopique.

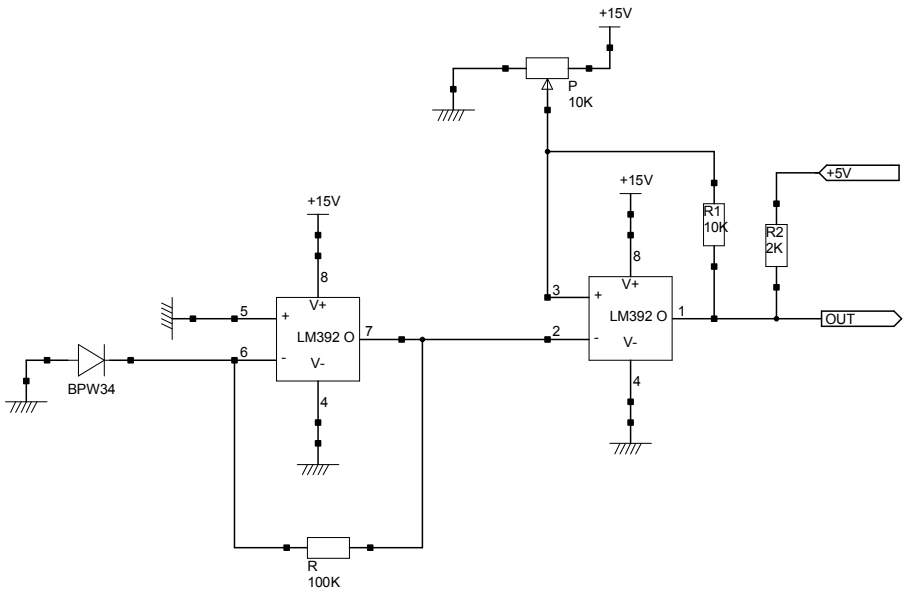
Deux photodiodes sont diposées sur le socle, à environ 5 cm en dessous du barreau. La première (D1) est située sur l'axe x et sert à détecter le passage de l'ombre du barreau sur cet axe. La seconde (D2) se trouve à 4 mm de la première, dans la direction y. Les diodes sont reliées à un amplificateur comportant un comparateur, ce qui permet d'obtenir deux signaux numériques transmis à l'arduino. Le circuit d'amplification est décrit plus loin. Le programme de l'arduino détermine l'intervalle de temps entre deux passages du barreau sur l'axe et sa vitesse lors du passage.



2.b. Amplificateur des diodes

L'amplificateur d'une photodiode est constitué d'un convertisseur courant-tension suivi d'un comparateur à hystérésis. Le circuit est alimenté avec une alimentation stabilisée de 15 V (10 V fonctionne aussi). La résistance $R = 100\ k\Omega$ du convertisseur est choisie pour donner une tension en sortie du convertisseur variant entre une valeur proche de la saturation et un niveau beaucoup plus faible lorsque l'ombre du barreau passe sur la photodiode. Le comparateur permet d'obtenir en sortie une tension nulle lorsque la photodiode est pleinement éclairée, et 5 V lorsqu'elle est sous l'ombre. Pour ce faire, la sortie en collecteur ouvert du comparateur est reliée à l'alimentation 5 V de l'arduino

via une résistance de $R_2 = 2\ k\Omega$. La largeur de l'impulsion ainsi obtenue permet de déterminer la vitesse du barreau. Pour obtenir un temps de réponse assez court, le circuit doit être alimenté avec une tension d'au moins 10 V. L'ampli-op et le comparateur sont intégrés dans un même circuit (LM392).



Le potentiomètre permet de régler la tension de seuil du comparateur. Pour éliminer les basculements multiples à la sortie du comparateur (à cause du bruit dans le signal d'entrée), une résistance R1 relie la sortie à l'entrée non inverseuse, ce qui permet d'obtenir un hystérésis.

3. Programme arduino

Pour détecter les fronts générés par les deux sorties de l'amplificateur à photodiodes, on utilise le mode Input Capture et les entrées ICP (Input Capture Pin), disponibles seulement sur l'arduino MEGA (pas sur le UNO). La technique utilisée est décrite en détail dans mesures de fréquence et de temps.

Voici les bornes de la carte Arduino MEGA utilisées :

- ▷ D8 : activation du pont A.
- ▷ D11 : commande MLI du pont A.
- ▷ D6 : activation du pont B.
- ▷ D12 : commande MLI du pont B.
- ▷ D48 : ICP5, entrée Input Capture du Timer 5.
- ▷ D49 : ICP4, entrée Input Capture du Timer 4.
- ▷ D7 : commande de la lampe LED.
- ▷ D2 : signal de contrôle pour la MLI périodique, synchrone avec le champ oscillant

▷ D3 : alimentation +5 V pour la sortie des comparateurs de l'amplificateur à photodiodes.

Le Timer 1 est utilisé pour générer les deux signaux MLI, sur ses sorties A et B.

Les tensions générées par les deux ponts par modulation MLI sont stockées dans deux tables, et générées avec un accumulateur de phase 32 bits, ce qui permet de faire varier la fréquence très finement. Chaque table comporte 256 échantillons (adressage 8 bits).

bobinesAimant-mega.ino

```
#include "Arduino.h"
#define NECHANT 256
#define NECHANTD2 128
#define SHIFT_ACCUM 24

uint32_t icr;
uint32_t table_onde_A[NECHANT];
uint32_t table_onde_B[NECHANT];
uint32_t accum1,accum2,increm;
uint8_t indexA,indexB;
uint16_t diviseur[6] = {0,1,8,64,256,1024};
volatile uint8_t flash;
uint8_t strob = 0;
uint8_t synchro = 0;
float clock_period;
uint8_t front1 = 0;
uint8_t front2 = 0;
float duree_impulsion_4,duree_impulsion_5;
```

La fonction `init_pwm_timer1` programme le timer 1 pour la générations des signaux MLI sur ses sorties A et B. La période est donnée en microsecondes. Tant que la période est inférieure à environ 4 ms, l'horloge utilisée est à 16 MHz. Si la période est de 100 μs, la valeur maximale du compteur (icr) est 800. Dans ce cas, il y a 800 valeurs possibles pour le rapport cyclique, ce qui donne une bonne finesse pour la modulation MLI. Si la période est réduite, la précision de la modulation diminue.

```
void init_pwm_timer1(uint32_t period) {
    char clockBits;
    TCCR1A = 0;
    TCCR1A |= (1 << COM1A1); //Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/O
    TCCR1A |= (1 << COM1B1);
#if defined(__AVR_ATmega2560__) || defined(__AVR_ATmega32U4__)
    TCCR1A |= (1 << COM1C1);
#endif
    TCCR1B = 1 << WGM13; // phase and frequency correct pwm mode, top = ICR1
    int d = 1;
    icr = (F_CPU/1000000*period/2);
    while ((icr>0xFFFF)&&(d<6)) { // choix du diviseur d'horloge
        d++;
```

```
        icr = (F_CPU/1000000*period/2/diviseur[d]);
   }
   clockBits = d;
   ICR1 = icr; // valeur maximale du compteur
   TIMSK1 = 1 << TOIE1; // overflow interrupt enable
   sei(); // activation des interruptions
   TCNT1 = 0; // mise à zéro du compteur
   TCCR1B |= clockBits; // déclenchement du compteur
}
```

Lorsque le compteur du timer 1 arrive à sa valeur maximale (icr), l'interruption suivante est déclenchée, au cours de laquelle les rapports cycliques des sorties A et B sont mis à jour en fonction des valeurs lues dans les deux tables avec les accumulateurs de phase.

```
ISR(TIMER1_OVF_vect) { // Timer 1 Overflow interrupt
  accum1 += increm;
  accum2 += increm;
  indexA = accum1 >> SHIFT_ACCUM;
  OCR1A = table_onde_A[indexA];
  indexB = accum2 >> SHIFT_ACCUM;
  OCR1B = table_onde_B[indexB];
  if (indexA==0) { // début période
 #if defined(__AVR_ATmega2560__)
      PORTH |= (1 << PORTH4); // flash ON
      PORTE |= (1 << PORTE4);
 #else
      PORTD |= (1 << PORTD7); // flash ON
      PORTD |= (1 << PORTD2);

 #endif
      flash = 1;
      }

  if ((flash)&&(indexA==2)) {
  #if defined(__AVR_ATmega2560__)
      if (strob) PORTH &= ~(1 << PORTH4); // flash OFF
  #else
      if (strob) PORTD &= ~(1 << PORTD7); // flash OFF
  #endif
      flash = 0;
  }
  if (indexA==NECHANTD2) {
  #if defined(__AVR_ATmega2560__)
     PORTE &= ~(1 << PORTE4);
  #else
     PORTD &= ~(1 << PORTD2);
  #endif
```

```
  }
}
```

Lorsque la période de la sortie A débute, la lampe LED est déclenchée. Si la variable `strob` a une valeur non nulle, la lampe est étteinte lorsque l'indice `indexA` atteint la valeur 2. On peut augmenter cette valeur si l'on veut que l'éclair dure plus longtemps. Par ailleurs, un signal est généré sur la sortie D2, vallant 0 pendant la première demi-période, 1 pendant la seconde.

La fonction `start_capture` programme les Timers 4 et 5 pour la capture sur les entrées ICP4 et ICP5. L'argument `clock` est un indice dans le tableau `diviseur`, qui contient les différents diviseurs d'horloge. Par exemple, pour `clock=4`, l'horloge des timers est à la fréquence 16/256 $MHz = 62,5\ kHz$, ce qui permet de faire des mesures jusqu'à environ 1 secondes avec ces timers 16 bits, tout en ayant une bonne précision pour des durées de l'ordre de la milliseconde. Au début, les deux timers sont programmés pour détecter un front montant.

```
void start_capture(uint8_t clock) {
    cli();
    TCCR5A = 0; // normal counting
    TCCR4A = 0;
    TCCR5B = (1 << ICES5); // détection d'un front montant
    TIMSK5 = (1 << ICIE5); // input capture interrupt enable interrupt
    TCCR4B = (1 << ICES4);
    TIMSK4 = (1 << ICIE4);
    sei();
    TCCR5B |= clock;
    TCCR4B |= clock;
    clock_period = 1.0/F_CPU*diviseur[clock];
}
```

Voici la fonction d'interruption exécutée lorsqu'un front est détecté sur l'entrée ICP5 du timer 5. La valeur du compteur permet de calculer la durée écoulée depuis le dernier front détecté. Lorsque le front détecté est le front montant (début du passage de l'ombre sur la photodiode), on modifie le registre TCCR5B pour que le front suivant soit descendant, ce qui permet de mesurer la durée de l'impulsion. Inversement, si le front détecté est descendant, on programme la détection suivante sur le front montant, ce qui permet de mesurer l'intervalle de temps entre la fin de l'impulsion et le début de la suivante. La durée de l'impulsion est mémorisée dans une variable globale, de manière à calculer l'intervalle de temps entre le début de deux impulsions.

```
ISR(TIMER5_CAPT_vect) {
   TCNT5 = 0;
   uint16_t count = ICR5; // input capture register
   float duree = count*clock_period*1000;
   if (front1==0) {
    TCCR5B &= ~(1 << ICES5);
    front1=1;
```

```
    Serial.println("Diode 1 : Phase = "+String(indexA)+" Duree = "+String(duree_impuls
   }
   else {
      TCCR5B |= (1 << ICES5);
      front1=0;
      duree_impulsion_5 = duree;
    }
}
```

Voici la fonction d'interruption qui fait la même chose pour le timer 4 (entrée ICP4) :

```
ISR(TIMER4_CAPT_vect) {
   TCNT4 = 0;
   uint16_t count = ICR4; // input capture register
   float duree = count*clock_period*1000;
   if (front2==0) {
    TCCR4B &= ~(1 << ICES4);
    front2=1;
    Serial.println("Diode 2 : Phase = "+String(indexA)+" Duree = "+String(duree_impuls
   }
   else {
      TCCR4B |= (1 << ICES4);
      front2=0;
      duree_impulsion_4 = duree;

    }
}
```

Les deux fonctions suivantes remplissent les tables pour les sorties A et B, avec une fonction sinusoïdale dont l'amplitude (comprise entre 0 et 1) et le décalage (entre 0 et 1) sont donnés. La somme de l'amplitude et du décalage ne doit pas dépasser 1. Il est même préférable d'éviter les valeurs trop proche de 1. Les valeurs stockées dans ces tables sont les valeurs attribuées aux registres de comparaison OCR1A et OCR1B lors de la modulation. Ces valeurs dépendent de la valeur de `icr`, c'est pourquoi la fonction `init_pwm_timer1` doit être appelée auparavant.

```
void set_sinus_table_A(float amp, float offset) {
  int i;
  float dt = 2*3.1415926/NECHANT;
  for(i=0; i<NECHANT; i++) {
    table_onde_A[i] = icr*0.5*(1.0+offset+amp*sin(i*dt));
  }
}

void set_sinus_table_B(float amp, float offset) {
  int i;
  float dt = 2*3.1415926/NECHANT;
```

```
  for(i=0; i<NECHANT; i++) {
    table_onde_B[i] = icr*0.5*(1.0+offset+amp*sin(i*dt));
  }
}
```

La fonction `setup` configure les entrées-sorties puis définit la période de découpage. On génère sur les sorties A et B deux sinusoïdes en quadrature afin d'obtenir deux champs magnétiques perpendiculaires en quadrature. La variable `frequence` définit la fréquence de ces sinusoïdes.

```
void setup() {
    Serial.begin(115200);
    pinMode(48,INPUT);
    pinMode(49,INPUT);
    pinMode(8,OUTPUT);
    digitalWrite(8,HIGH); // commande ENABLE A
    pinMode(6,OUTPUT);
    digitalWrite(6,HIGH); // commande ENABLE B
    pinMode(3,OUTPUT); // alim sortie comparateur
    digitalWrite(3,HIGH);
#if defined(__AVR_ATmega2560__)
    pinMode(11,OUTPUT);
    pinMode(12,OUTPUT);
#elif defined(__AVR_ATmega32U4__)
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
#else
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
#endif
    pinMode(7,OUTPUT);
    digitalWrite(7,HIGH); // PD7 sur UNO, PH4 sur MEGA
    pinMode(2,OUTPUT);
    digitalWrite(2,HIGH); // PD2 sur UNO, PE4 sur MEGA
    pinMode(4,OUTPUT);
    digitalWrite(4,HIGH);
    uint32_t period_pwm = 100; // en microsecondes
    float frequence = 4; // en Hz
    accum1 = 0;
    accum2 = ((uint32_t)(NECHANT * 0.25)) << SHIFT_ACCUM;
    flash=0;
    synchro=0;
    increm = (uint32_t) (((float)(0xFFFFFFFF))*((float)(frequence)*1e-6*(float)(perio
    init_pwm_timer1(period_pwm);
    set_sinus_table_A(0.5,0.5);
    set_sinus_table_B(0.5,0.0);
    front1 = front2 = 0;
    start_capture(4);
```

```
}


void loop() {

}
```

4. Exemples d'expérience

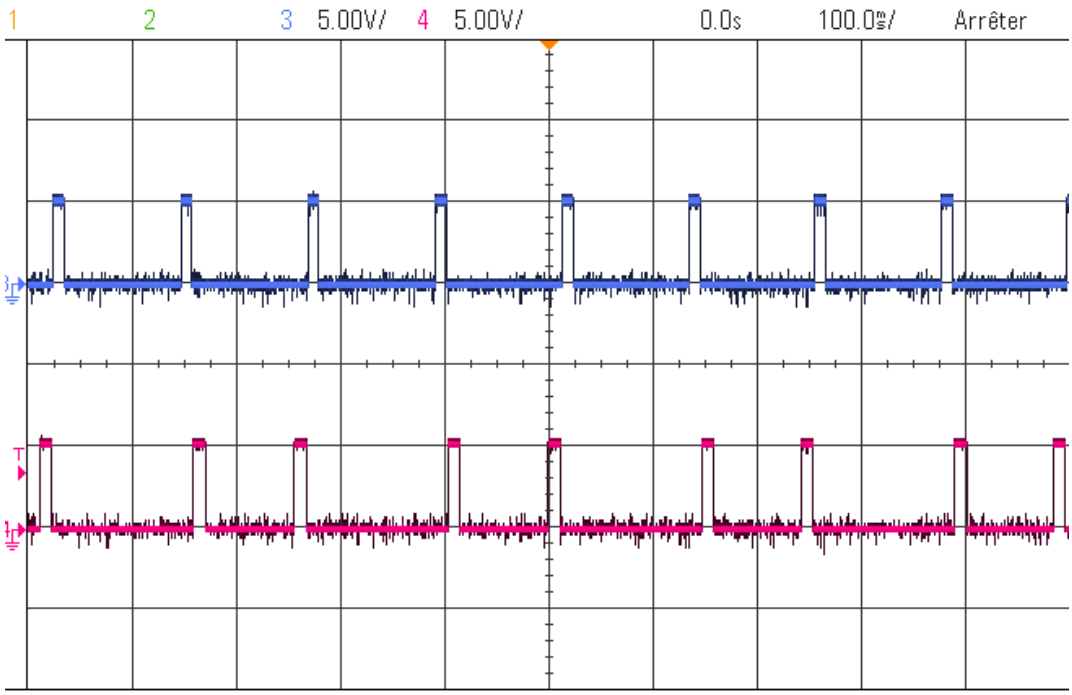
4.a. Oscillation dans un champ stationnaire

On applique un champ stationnaire B_x avec les appels suivants :

```
set_sinus_table_A(0.0,0.5);
set_sinus_table_B(0.0,0.0);
```

Pour une tension d'alimentation $V_s = 12\ V$, le champ magnétique au centre (mesuré avec une sonde à effet Hall) est $B_x = 1,1\ mT$. Ces variations sur la zone sur laquelle se trouve le barreau aimanté est de l'ordre de 10 pour cent. En première approximation, on a donc un champ stationnaire uniforme. On observe des oscillations du barreau autour de ce champ. Pour un champ uniforme, l'équation du mouvement de rotation d'un barreau aimanté est en principe l'équation du pendule.

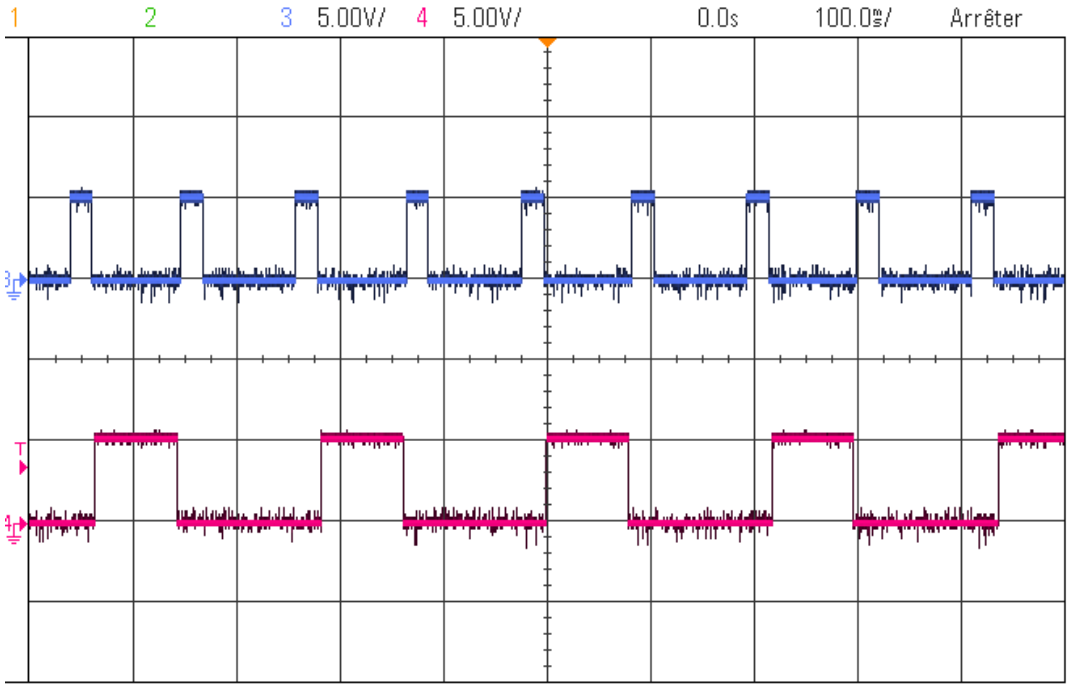
Voici les signaux des photodiodes au début de l'expérience, lorsque l'amplitude d'oscillation est grande (plusieurs dizaines de degrés) :



Le signal de la photodiode D1 située sur l'axe est sur la voie 3 (en bleu). Sa période est la moitié de la période d'oscillation du barreau, qui est ici d'environ 240 ms. Le signal

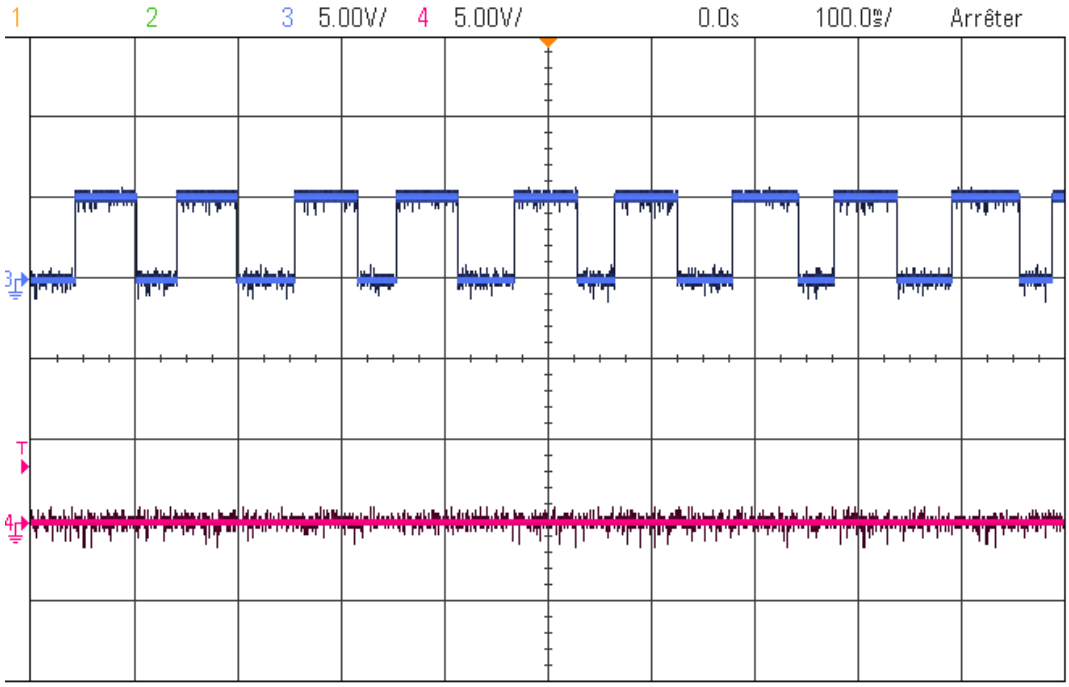
fourni par la diode D2 est décalée dans un sens qui dépend du sens du mouvement. La comparaison des deux permet donc de déterminer le sens du mouvement.

Voici les signaux un peu plus tard, lorsque l'amplitude de l'oscillation est plus faible :



La période est plus courte et la vitesse est plus faible, ce qui est le comportement attendu pour l'équation du pendule. Le signal de la diode D2 ne comporte plus qu'une seule impulsion à chaque demi-période car l'ombre ne dépasse pas la diode D2.

Voici les signaux à la fin du mouvement :



L'ombre ne parvient pas à la diode D2, mais les oscillations sont encore détectées sur la diode D1.

Voyons à présent la sortie console fournie par l'arduino, au début du mouvement (les temps sont en millisecondes) :

```
Diode 2 : Phase = 89 Duree = 4.48 Intervalle = 240.42
Diode 2 : Phase = 52 Duree = 4.56 Intervalle = 215.04
Diode 1 : Phase = 57 Duree = 4.30 Intervalle = 226.30
Diode 1 : Phase = 33 Duree = 4.38 Intervalle = 227.20
Diode 2 : Phase = 38 Duree = 4.61 Intervalle = 238.54
Diode 2 : Phase = 255 Duree = 4.67 Intervalle = 213.18
Diode 1 : Phase = 4 Duree = 4.45 Intervalle = 224.66
Diode 1 : Phase = 234 Duree = 4.48 Intervalle = 224.90
Diode 2 : Phase = 239 Duree = 4.70 Intervalle = 236.48
Diode 2 : Phase = 198 Duree = 4.80 Intervalle = 211.10
Diode 1 : Phase = 203 Duree = 4.48 Intervalle = 222.78
Diode 1 : Phase = 174 Duree = 4.59 Intervalle = 222.21
Diode 2 : Phase = 179 Duree = 4.78 Intervalle = 234.06
Diode 2 : Phase = 136 Duree = 4.82 Intervalle = 208.75
Diode 1 : Phase = 141 Duree = 4.59 Intervalle = 220.70
Diode 1 : Phase = 109 Duree = 4.66 Intervalle = 219.28
Diode 2 : Phase = 114 Duree = 4.90 Intervalle = 231.30
Diode 2 : Phase = 68 Duree = 4.96 Intervalle = 206.27
Diode 1 : Phase = 74 Duree = 4.66 Intervalle = 218.34
Diode 1 : Phase = 38 Duree = 4.70 Intervalle = 216.10
Diode 2 : Phase = 43 Duree = 4.90 Intervalle = 228.29
Diode 2 : Phase = 251 Duree = 4.99 Intervalle = 203.47
Diode 1 : Phase = 0 Duree = 4.75 Intervalle = 215.76
Diode 1 : Phase = 217 Duree = 4.72 Intervalle = 212.69
Diode 2 : Phase = 223 Duree = 4.99 Intervalle = 225.09
Diode 2 : Phase = 171 Duree = 5.09 Intervalle = 200.19
Diode 1 : Phase = 176 Duree = 4.80 Intervalle = 212.67
Diode 1 : Phase = 134 Duree = 4.80 Intervalle = 209.23
Diode 2 : Phase = 139 Duree = 5.10 Intervalle = 221.82
Diode 2 : Phase = 84 Duree = 5.20 Intervalle = 197.17
Diode 1 : Phase = 90 Duree = 4.90 Intervalle = 209.87
Diode 1 : Phase = 43 Duree = 4.93 Intervalle = 205.60
```

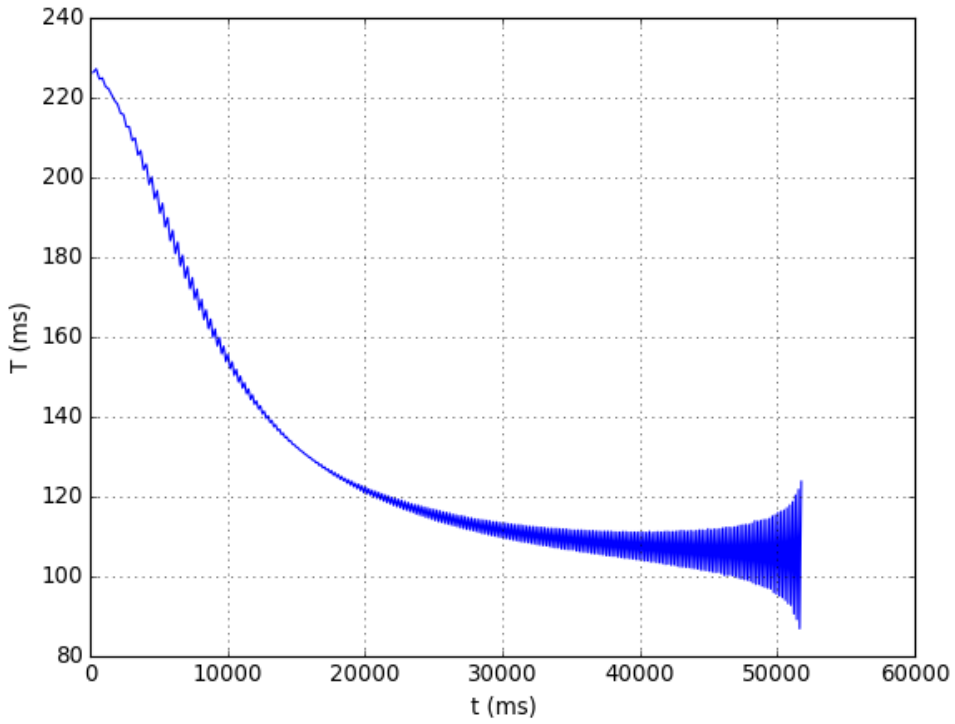
La phase n'a ici aucune signification car il n'y a pas de champ oscillant. Conformément à l'oscillogramme montré ci-dessus, il y a une succession de deux impulsions pour D1 suivies de deux impulsions pour D2. Pour chaque ligne, la durée est celle de l'impulsion qui précède l'intervalle. La pseudo-période d'oscillation est la somme de deux intervalles consécutifs fournis par la diode 1. La somme des intervalles donne le temps depuis le début.

Voici le tracé de l'intervalle pour la diode 1 en fonction du temps :

```
from matplotlib.pyplot import *
def extraction(filename):
    f = open(filename,'r')
    t = 0.0
```

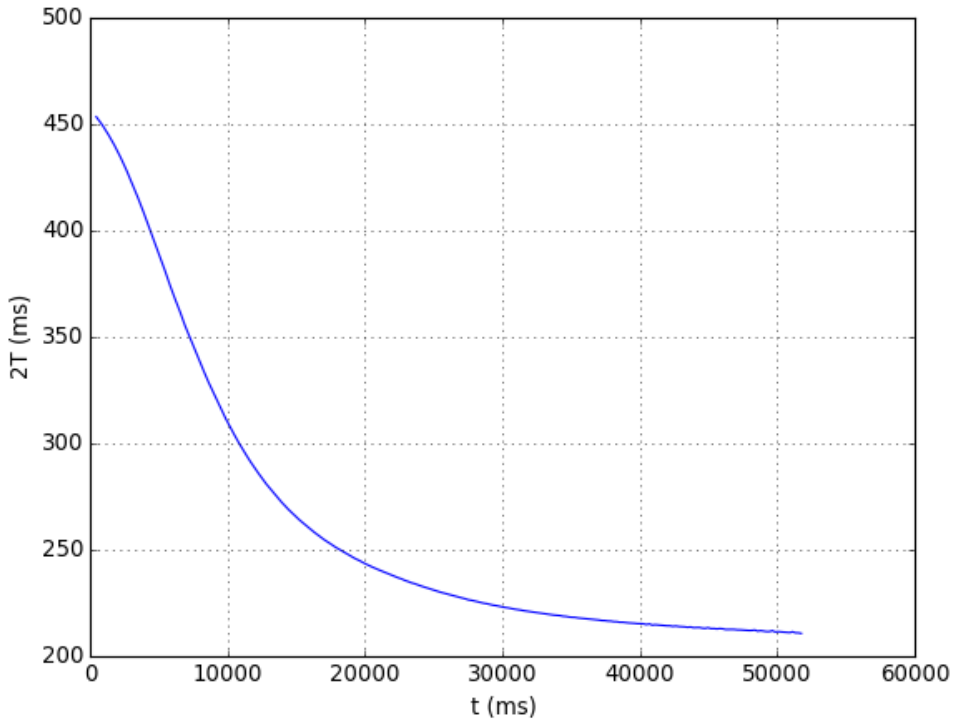
```
    liste_durees = []
    liste_intervalles = []
    liste_temps = []
    for ligne in f:
        L = ligne.strip().split(" ")
        diode = int(L[1])
        phase = float(L[5])
        duree = float(L[8])
        intervalle = float(L[11])
        if diode==1:
            t += intervalle
            liste_temps.append(t)
            liste_durees.append(duree)
            liste_intervalles.append(intervalle)
    f.close()
    liste_periodes = []
    liste_temps2 = []
    n=0
    T = 0
    for i,intervalle in enumerate(liste_intervalles):
        if n==0:
            n+=1
            T = intervalle
        else:
            n = 0
            T += intervalle
            liste_periodes.append(T)
            liste_temps2.append(liste_temps[i])
    return (liste_temps,liste_durees,liste_intervalles,liste_temps2,liste_periodes)

(liste_temps,liste_durees,liste_intervalles,liste_temps2,liste_periodes) = extraction
figure()
plot(liste_temps,liste_intervalles)
xlabel("t (ms)")
ylabel("T (ms)")
grid()
```



Vers la fin du mouvement, le défaut de centrage de la photodiode se manifeste par un écart important entre les intervalles des deux demi-périodes. Voici le tracé de la période :

```
figure()
plot(liste_temps2,liste_periodes)
xlabel("t (ms)")
ylabel("2T (ms)")
grid()
```



4.b. Pendule forcé

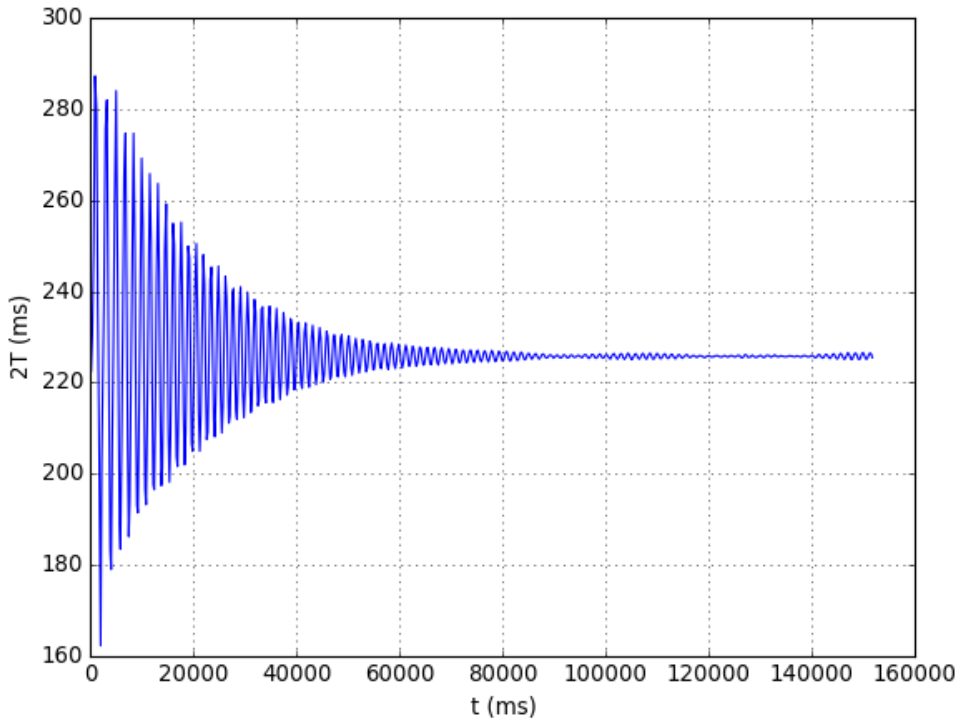
On ajoute au champ stationnaire B_x un champ oscillant B_y, avec une fréquence proche de la fréquence propre en petites oscillations. La courbe ci-dessus permet de déterminer cette fréquence à environ $4,5\ Hz$. Voici la configuration :

```
frequence = 4.5;

set_sinus_table_A(0.0,0.5);
set_sinus_table_B(0.1,0.0);
```

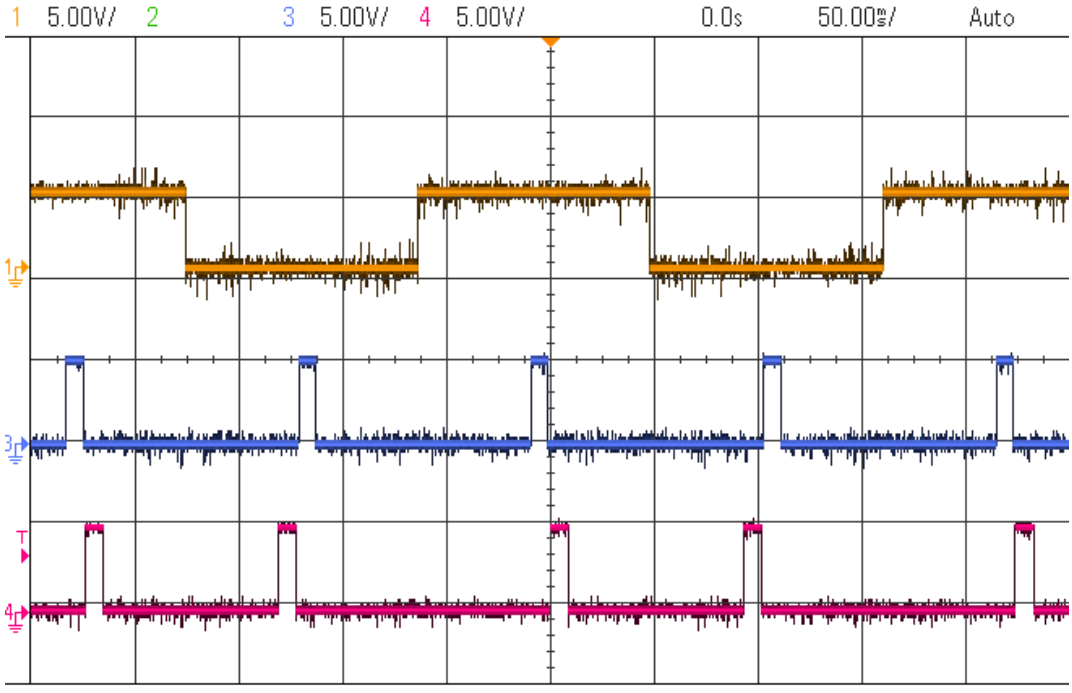
Voici le tracé de la période :

```
(liste_temps,liste_durees,liste_intervalles,liste_temps2,liste_periodes) = extraction
figure()
plot(liste_temps2,liste_periodes)
xlabel("t (ms)")
ylabel("2T (ms)")
grid()
```



Après environ 60 s, un régime permanent est atteint, avec une fréquence égale à celle du champ oscillant ($4,5\ Hz$).

Voici, en régime permanent, le signal numérique A synchrone à l'oscillation sinusoïdale du pont A (voie 1) et les signaux des deux photodiodes :



Le champ oscillant B_y est en quadrature par rapport au signal A. On en déduit que le barreau passe sur l'axe x lorsque le champ $B_y = 0$. La période d'oscillation du barreau est égale à celle du champ.

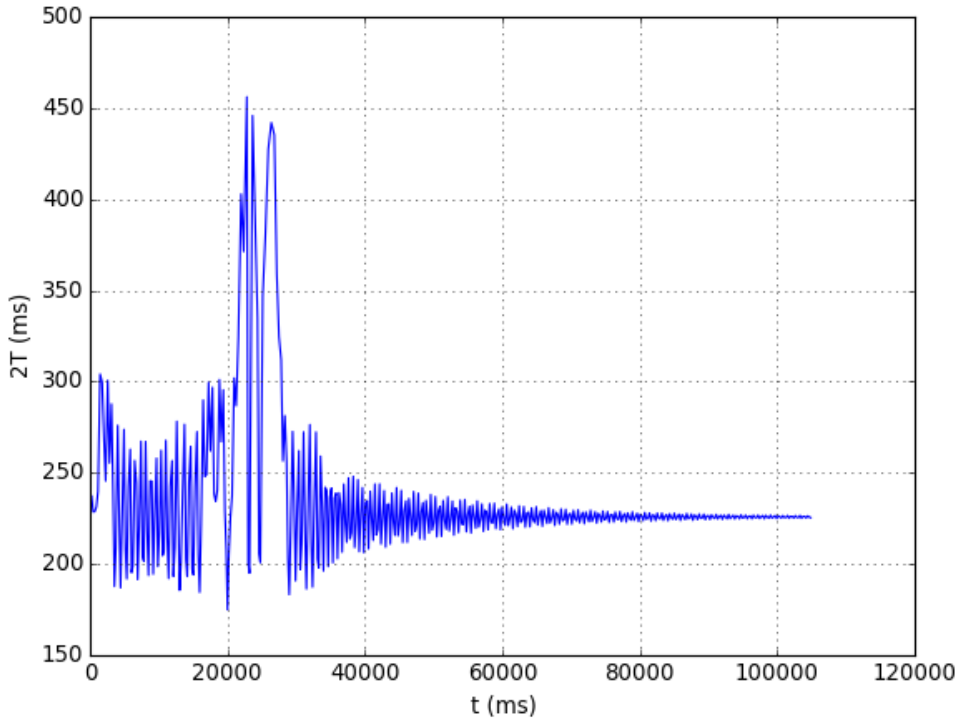
4.c. Pendule forcé paramétrique

Pour obtenir une résonance paramétrique, on applique un champ oscillant dans la même direction que le champ stationnaire, avec une fréquence double :

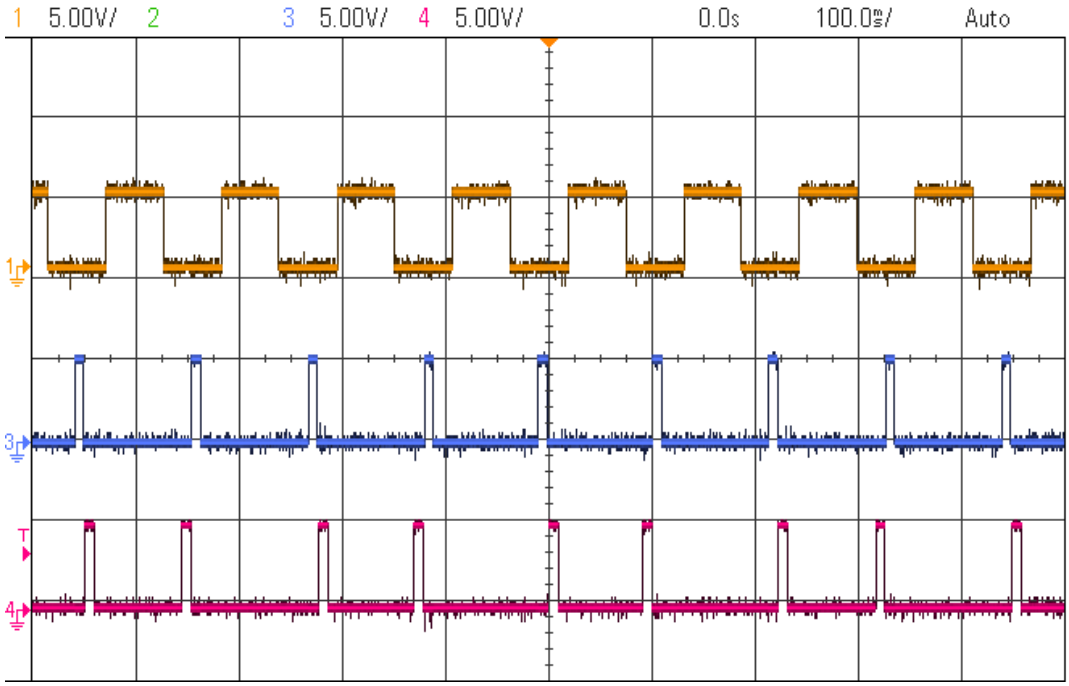
```
frequence = 9.0;

set_sinus_table_A(0.22,0.5);
set_sinus_table_B(0.0,0.0);


(liste_temps,liste_durees,liste_intervalles,liste_temps2,liste_periodes) = extraction
figure()
plot(liste_temps2,liste_periodes)
xlabel("t (ms)")
ylabel("2T (ms)")
grid()
```



Voici les signaux en régime permanent :

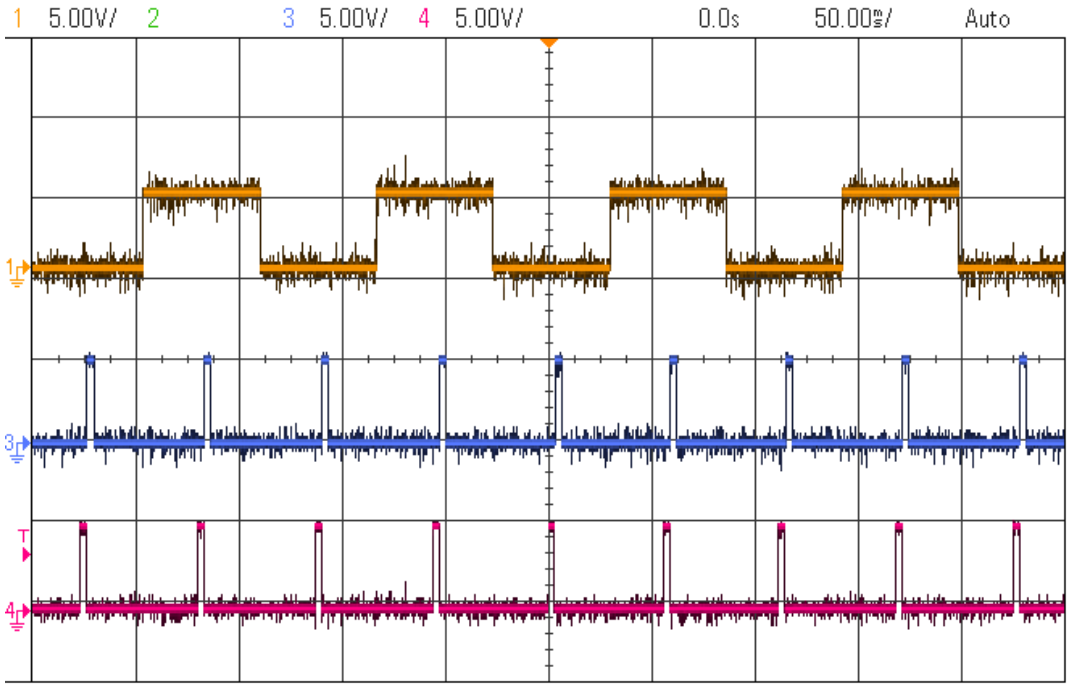


L'oscillation du barreau est synchrone avec le champ Bx oscillant, mais avec une période deux fois plus longue.

4.d. Rotation dans un champ tournant

Voici un exemple de champ tournant :

```
frequence = 9;
set_sinus_table_A(0.3,0.0);
set_sinus_table_B(0.3,0.0);
```



La rotation du barreau est synchrone avec celle du champ, et les deux passent par l'axe x au même moment.