Onduleur monophasé MLI

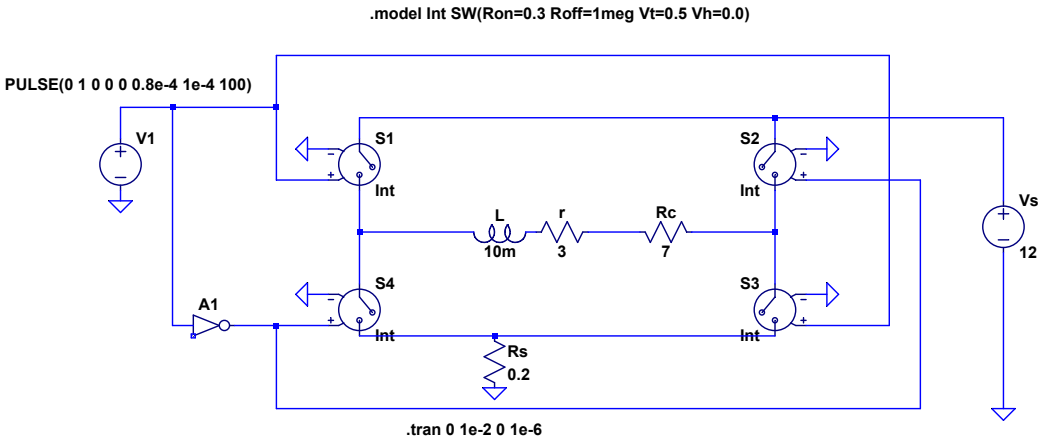
1. Introduction

Ce document présente la réalisation d'un onduleur monophasé piloté par modulation de largeur d'impulsion (MLI), appelée aussi PWM (pulse width modulation). Le signal MLI est généré par un Arduino, avec la méthode expliquée dans Génération d'un signal par modulation de largeur d'impulsion.

L'objectif est d'alimenter une bobine (afin de créer un champ magnétique pour des applications électromécaniques) avec un signal basse fréquence (inférieure à 1000 Hz) et un courant allant jusqu'à 4 A.

2. Principe du convertisseur DC/AC

La source d'énergie est une alimentation stabilisée en tension. Le convertisseur DC/AC à découpage repose sur l'utilisation d'un pont en H (hacheur à quatre quadrants), dont voici une simulation spice (LTSpice) réalisée au moyen de quatre interrupteurs :



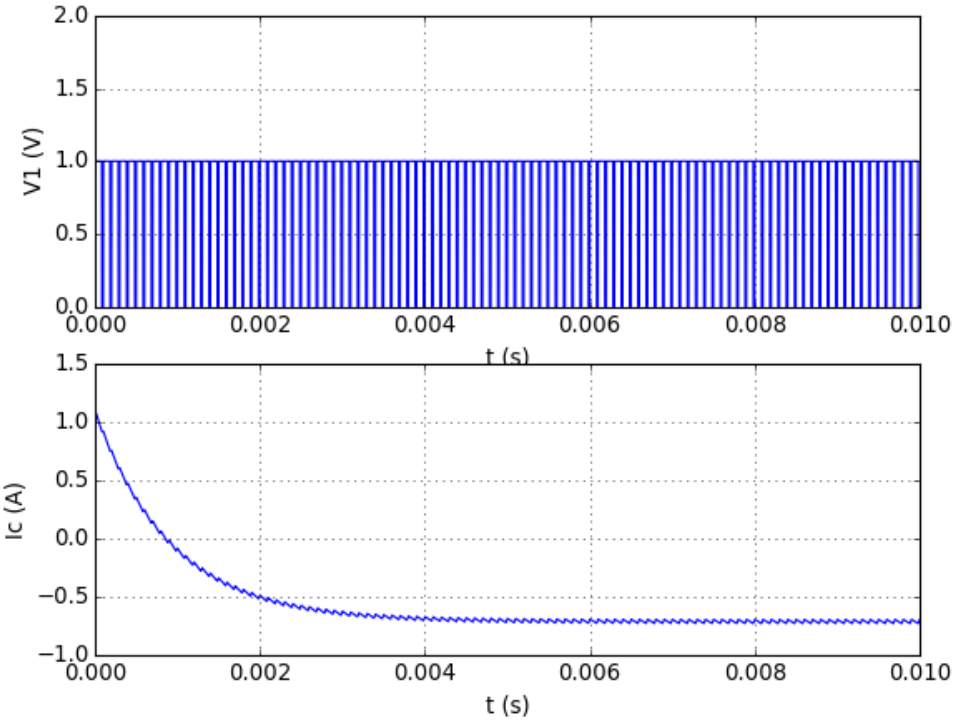
La charge est ici constituée d'une bobine d'auto-inductance 10 mH et de résistance 3 Ω à laquelle on a ajouté une résistance de 7 Ω pour mesurer le courant. L'alimentation est modélisée par une source de tension de 12 V.

Le signal de commande V1 est un signal carré dont le rapport cyclique est fixé. Lorsque la commande est à 5 V, les interrupteurs S1 et S3 sont fermés alors que les interrupteurs S2 et S4 sont ouverts (la porte inverseuse A1 commande S2 et S4). La charge voit alors une force électromotrice positive (potentiel à gauche plus élevé). Lorsque la commande est à 0 V, les interrupteurs S1 et S3 sont ouverts alors que S2 et S4 sont fermés, et la charge voit une force électromotrice négative.

Voici le résultat de la simulation. La période du signal est de 100 μs. On trace la tension de commande et le courant dans la charge :

```
import numpy
from matplotlib.pyplot import *
```

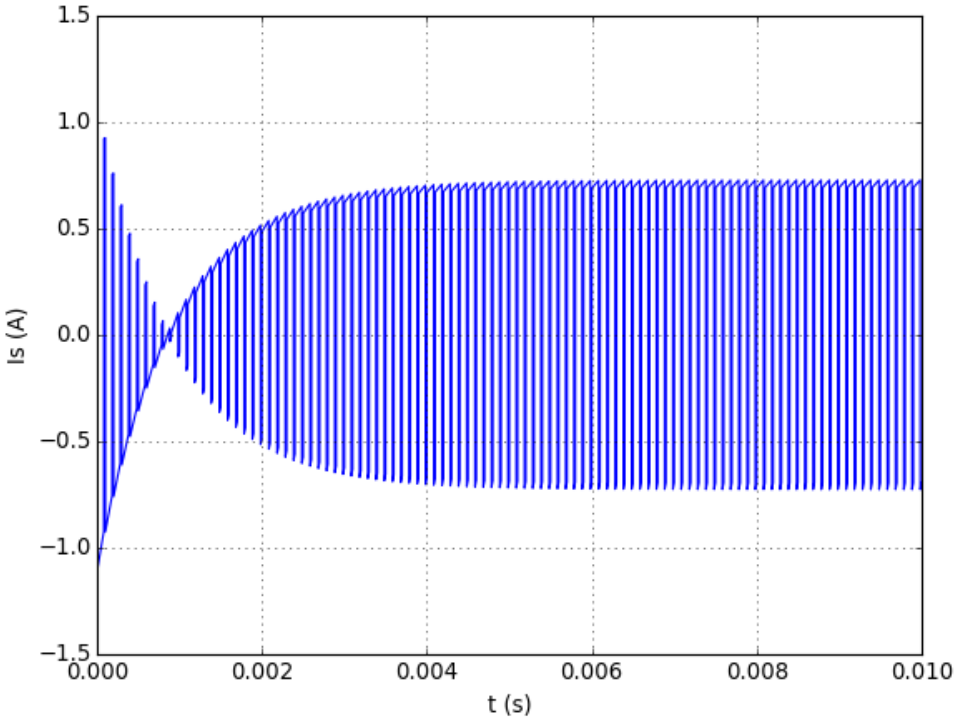
```
data = numpy.loadtxt("pontH.txt",skiprows=1,unpack=True)
t=data[0]
V2 = data[1]
Ic = data[9]
Is = data[12]

figure()
subplot(211)
plot(t,V2)
xlabel("t (s)")
ylabel("V1 (V)")
axis([0,t[-1],0,2])
grid()
subplot(212)
plot(t,Ic)
xlabel("t (s)")
ylabel("Ic (A)")
grid()
```



La constante de temps de la charge est $\tau = L/R = 1\ ms$, ce qui permet une bonne atténuation des oscillations à 10 kHz. Il reste des ondulations du courant très faibles. On remarque aussi le temps d'établissement du régime permanent (temps de réponse), d'environ 4 ms. Pour la technique de modulation de largeur d'impulsion, il faudra procéder à une période de modulation grande devant ce temps, soit une fréquence environ inférieure à 10 Hz.

La résistance Rs entre la base du pont et la masse n'est pas indispensable, mais elle est souvent ajoutée pour obtenir le courant traversant la charge :

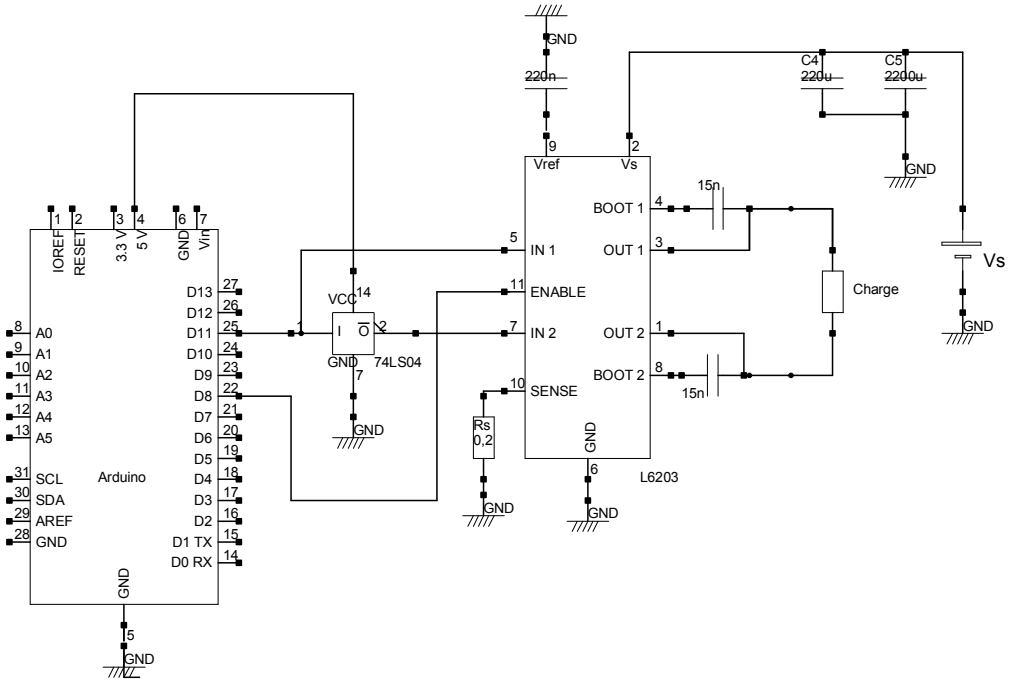
```
figure()
plot(t,Is)
xlabel("t (s)")
ylabel("Is (A)")
grid()
```



Le courant dans cette résistance a la particularité de changer de signe lorsque le pont commute.

3. Réalisation du pont en H

On utilise le pont en H à transistors DMOS L6203, qui peut délivrer jusqu'à 4 A. La fréquence de découpage peut aller jusqu'à 100 kHz.



L'entrée IN1 commande deux transistors (interrupteurs S1 et S3) alors que IN2 commande les deux transistors opposés (S2 et S4). On utilise un inverseur 74LS04 pour piloter ces deux entrées à partir d'une seule sortie de l'Arduino MEGA (D11). L'entrée ENABLE permet d'activer ou de désactiver les commandes des transistors. Lorsqu'on utilise le pont pour commander un moteur DC, le signal MLI est envoyé sur ENABLE alors que IN1 et IN2 permettent de choisir le sens du courant dans le moteur. Dans le cas présent, on fixe ENABLE au niveau haut (avec la sortie D8 de l'arduino), et on applique le signal MLI sur IN1 et IN2. La résistance de 0.2 Ω placée entre l'entrée SENSE et la masse permet de mesurer le courant traversant la charge.

La charge est constituée d'une bobine à aire de 500 spires, d'auto-inductance $L = 10\ mH$ et de résistance interne 3 Ω en série avec une résistance de 7 Ω servant à mesurer le courant. La bobine a pour fonction de produire un champ magnétique variable.

4. Programme Arduino

On reprend le programme Arduino décrit dans Génération d'un signal par modulation de largeur d'impulsion qui réalise un générateur MLI triphasé mais on le simplifie pour piloter une seule phase (ce qui permet d'augmenter la fréquence).

generateurPWM-1P.ino

```
#include "Arduino.h"
#define NECHANT 128
#define SHIFT_ACCUM 25

uint32_t icr;
uint32_t table_onde[NECHANT];
uint32_t accum1,accum2,accum3,increm;
uint16_t diviseur[6] = {0,1,8,64,256,1024};
```

```
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
}

ISR(TIMER1_OVF_vect) { // Timer 1 Overflow interrupt
  accum1 += increm;
  OCR1A = table_onde[accum1 >> SHIFT_ACCUM];
}

void set_sinus_table(float amp, float offset) {
  int i;
  float dt = 2*3.1415926/NECHANT;
  for(i=0; i<NECHANT; i++) {
    table_onde[i] = icr*0.5*(1.0-offset+amp*sin(i*dt));
  }
}
void set_square_table(float amp, float offset) {
  float i0 = NECHANT/2;
  int i;
  for (i=0; i<i0; i++) table_onde[i] = icr*0.5*(1.0-offset+amp);
  for (i=i0; i<NECHANT; i++) table_onde[i] = icr*0.5*(1.0-offset-amp);
}

void setup() {
    pinMode(8,OUTPUT);
    digitalWrite(8,HIGH); // commande ENABLE
#if defined(__AVR_ATmega2560__)
    pinMode(11,OUTPUT);
```

```
#elif defined(__AVR_ATmega32U4__)
    pinMode(9,OUTPUT);
#else
    pinMode(9,OUTPUT);
#endif
    uint32_t period_pwm = 100; // en microsecondes (50 kHz)
    uint32_t frequence = 10; // en Hz
    accum1 = 0;
    increm = (uint32_t) (((float)(0xFFFFFFFF))*((float)(frequence)*1e-6*(float)(perio
    init_pwm_timer1(period_pwm);
    set_sinus_table(0.5,0);
}
void loop() {

}
```

Le signal est une sinusoïde générée dans la fonction `set_sinus_table`. On peut facilement modifier cette fonction pour générer d'autres formes d'onde. L'argument `amp` de cette fonction (compris entre 0 et 1) permet de faire varier l'amplitude de variation du rapport cyclique, et donc l'amplitude d'oscillation du courant dans la charge. Cette fonction doit être appelée après la fonction `init_pwm_timer1`, car elle utilise la variable `icr`, qui fixe la plus grande valeur atteinte par le compteur.

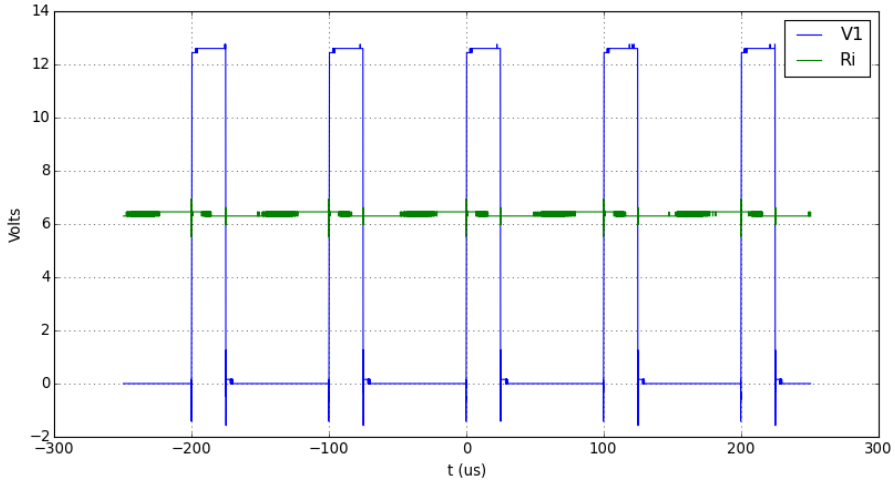
Pour l'arduino MEGA, la sortie D11 doit être utilisée pour délivrer le signal MLI. Sur les arduinos UNO et YUN, on utilise la sortie D9. La sortie D8 est mise au niveau haut pour activer le pont (ENABLE).

Ce programme peut fonctionner avec une porteuse jusqu'à 50 kHz avec un arduino MEGA. La fréquence de la sinusoïde est ici fixée à 10 Hz. Pour augmenter la fréquence de la porteuse, ou pour fonctionner à cette fréquence avec plusieurs phases, il faut utiliser l'arduino Due (plus rapide avec son microcontrôleur 32 bits).

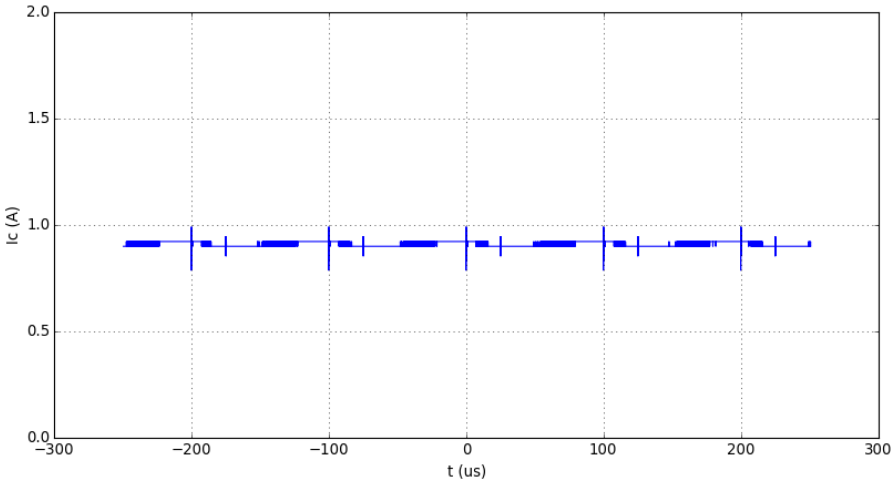
5. Tests de l'onduleur

Afin d'obtenir le courant dans la charge, on fait un test avec la moitié du pont : la charge est branchée entre la sortie OUT1 et la masse. La résistance de 7 Ω a une borne à la masse ; on peut donc obtenir directement la tension à ses bornes à l'oscilloscope.

On commence par un test avec un rapport cyclique constant (sinusoïde d'amplitude nulle avec un décalage). Voici, pour une fréquence de découpage de 10 kHz, la tension V1 (sortie OUT1) et la tension Ri aux bornes de la résistance :

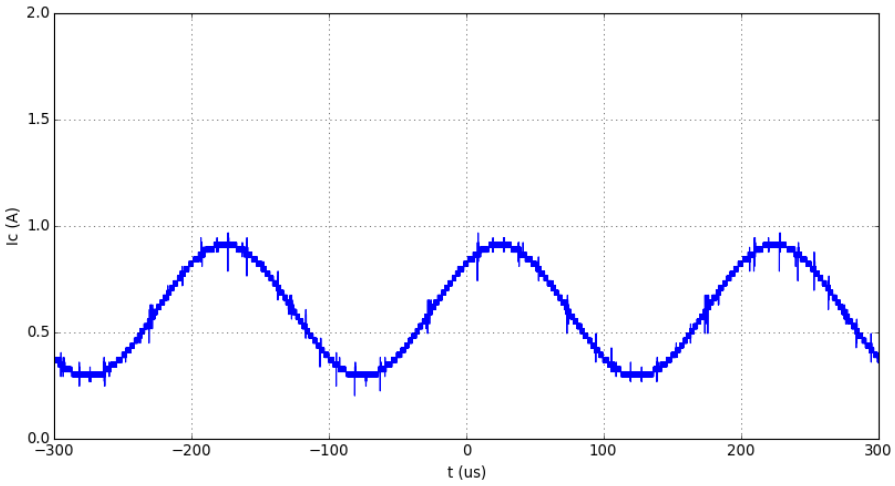


Voici le courant dans la charge :



Le taux d'ondulation est très faible, ce qui montre que la fréquence de découpage de 10 kHz est suffisante. Le bruit de quantification est visible (oscilloscope 8 bits). On voit aussi des pics de courant à la commutation des transistors (mais ceux-ci sont peut-être injectés dans le système de mesure par perturbation électromagnétique).

Voici le courant dans le cas d'une sinusoïde d'amplitude 0.5 (sans décalage) à une fréquence de 10 Hz :

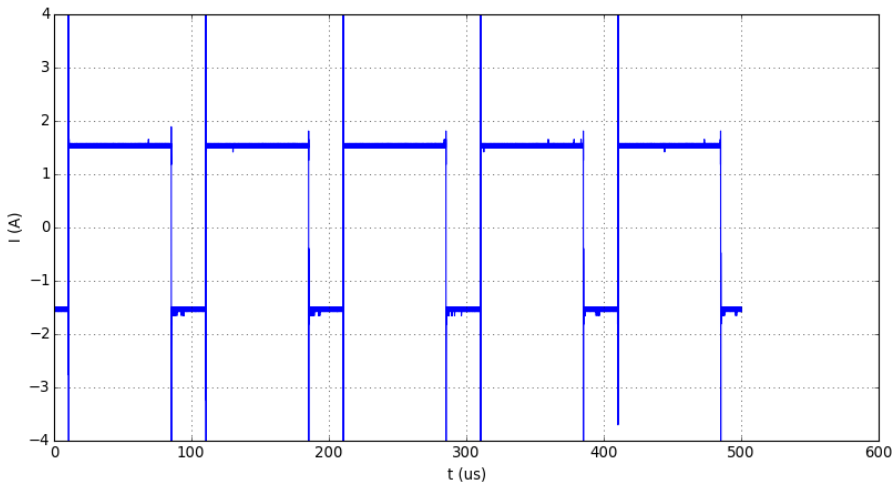


Le bruit visible sur ce signal vient d'une part du bruit de quantification, d'autre part des pics de courant aux commutations. Le courant a une composante DC car on utilise seulement la moitié du pont. Si l'on utilise le pont complet en branchant la charge entre les sorties OUT1 et OUT2, on obtient un courant alternatif sans composante DC, avec une amplitude double.

6. Utilisation de l'onduleur

La bobine est branchée directement sur les sorties OUT1 et OUT2 du pont. Pour contrôler le courant, on enregistre la tension aux bornes de la résistance $R_s = 0,2\ \Omega$. Le courant qui la traverse est, au signe près, le courant dans la bobine.

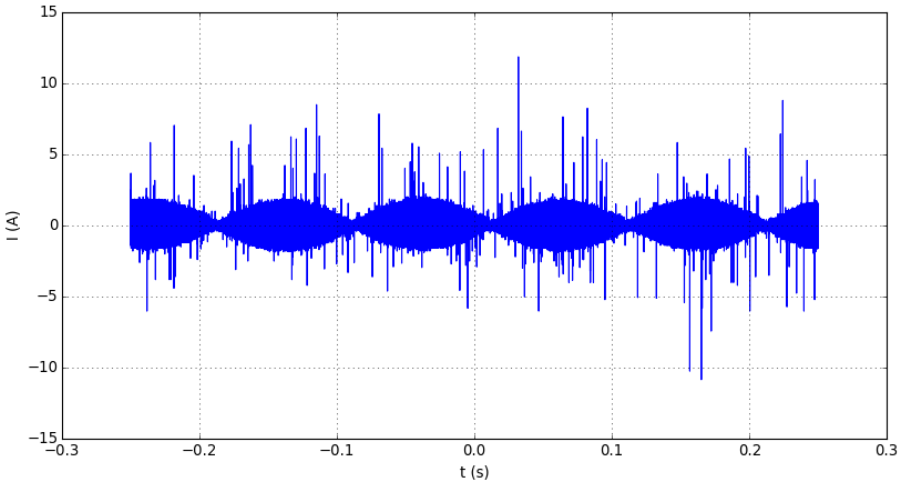
Pour une fréquence de découpage de 10 kHz, une tension constante d'amplitude $0,5$ (`set_sinus_table(0,0.5)`), voici le courant dans la résistance R_s pour la bobine de 10 mH et de résistance interne 3 Ω.



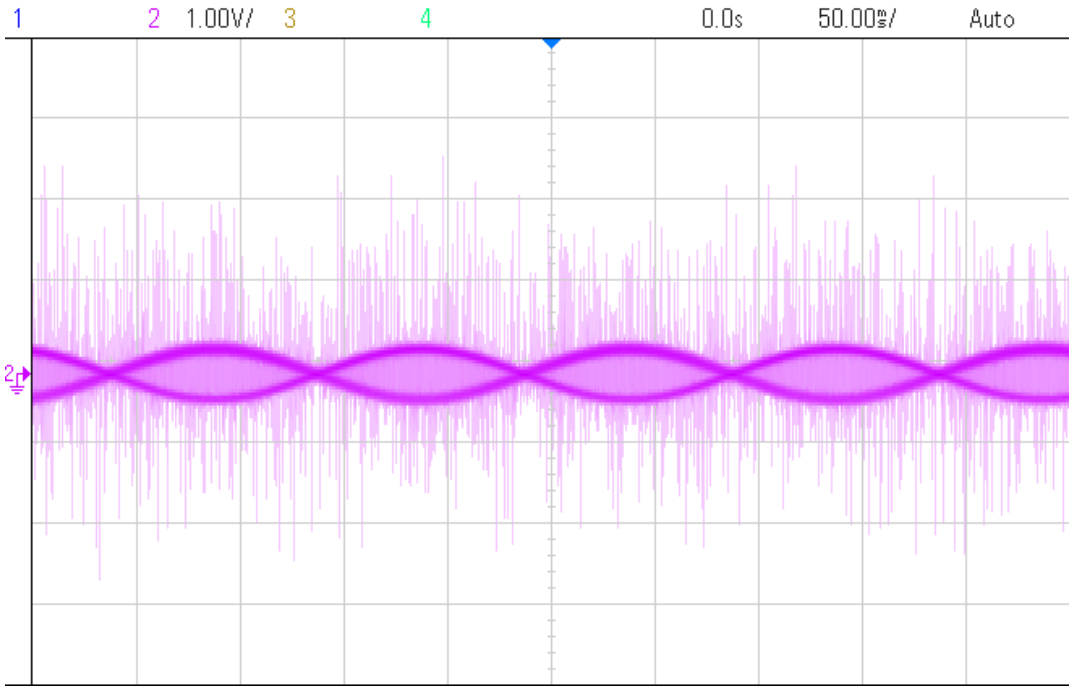
Le courant dans la bobine a une intensité de $1,5\ A$. On peut se servir de cette mesure pour étalonner l'onduleur, c'est-à-dire établir la correspondance entre le rapport cyclique de la tension de commande et le courant dans la bobine. Sur cet exemple, le rapport cyclique

est 3/4, qui correspond à un offset de 0, 5 porgrammé avec la fonction `set_sinus_table`. L'intensité du courant en fonction de l'amplitude est donc (pour cette bobine) $I = 3a$.

Voici le courant dans R_s pour une sinusoïde de 5 Hz programmée avec `set_sinus_table(0.5,0.0)` :



Voici une copie d'écran de l'oscilloscope :



L'image sur l'écran de l'oscilloscope est obtenue par superposition de plusieurs traces, ce qui renforce les parties importantes du signal, alors que la représentation statique est plutôt confuse.