Onduleur diphasé MLI pour bobines

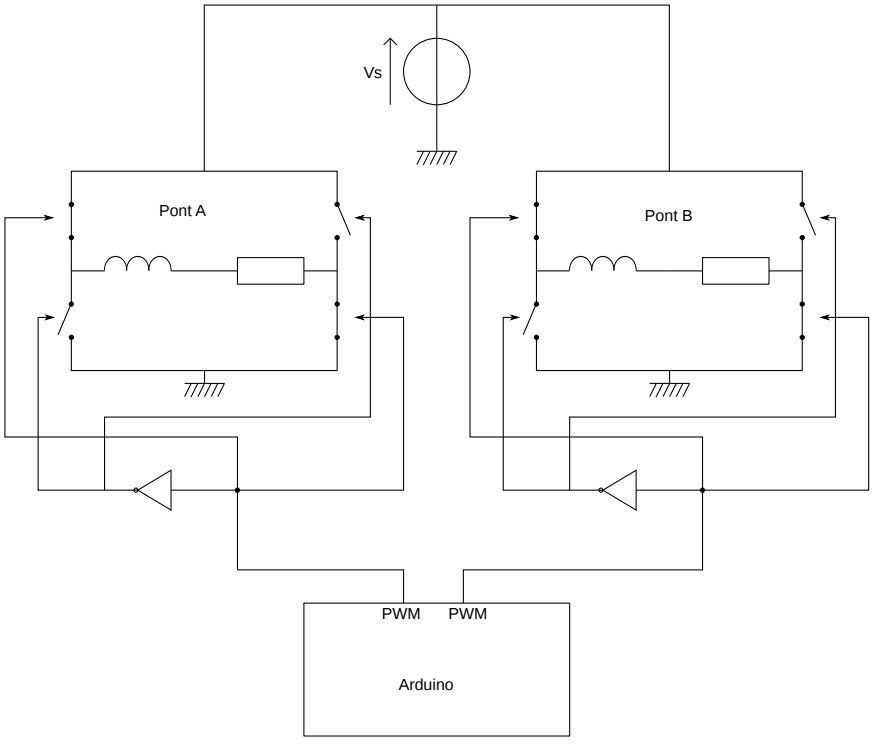
1. Introduction

Ce document présente un onduleur diphasé permettant d'alimenter deux bobines. Les bobines peuvent être utilisées pour produire un champ tournant, superposé éventuellement à un champ stationnaire.

Le système est constitué de deux onduleurs à modulation de largeur d'impulsion pilotés par une carte Arduino. Le programme de l'arduino génère deux signaux de commande MLI, afin d'obtenir par exemple deux courants sinusoïdaux en quadrature, avec éventuellement une composante DC.

2. Principe de l'onduleur

Chaque bobine est alimentée via un pont en H, qui réalise un hacheur à quatre quadrants. Chaque pont est piloté par une sortie PWM de l'arduino. Pour obtenir un champ tournant, les deux bobines sont disposées perpendiculairement. On peut bien sûr utiliser deux couples de bobines (type bobines de Helmholtz). La source d'énergie doit être une source de tension V_s.



3. Réalisation

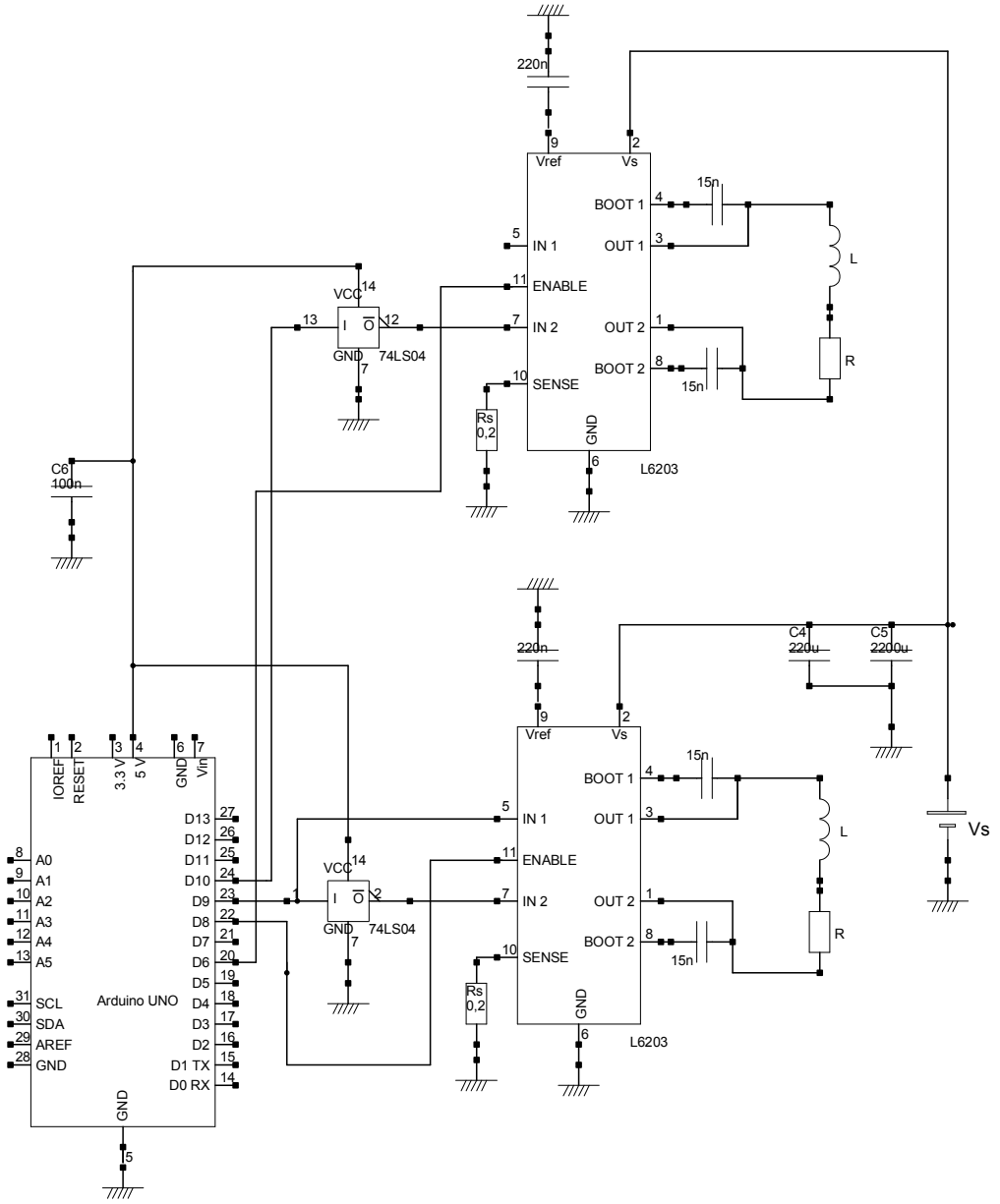
Chaque pont en H est réalisé avec le circuit intégré L6203, qui comporte 4 transistors MOSFET et le circuit de commande. Il peut délivrer jusqu'à $4\ A$ à une fréquence de découpage maximale de $100\ kHz$.

Chaque pont comporte deux entrées IN1 et IN2. La première est pilotée directement par une sortie PWM de l'arduino alors que la seconde est pilotée par la même sortie via une porte inverseuse. Il comporte aussi une entrée ENABLE, qui permet d'activer ou de désactiver le pont, reliée à une sortie de l'arduino. Voici les sorties PWM à utiliser pour les deux ponts :

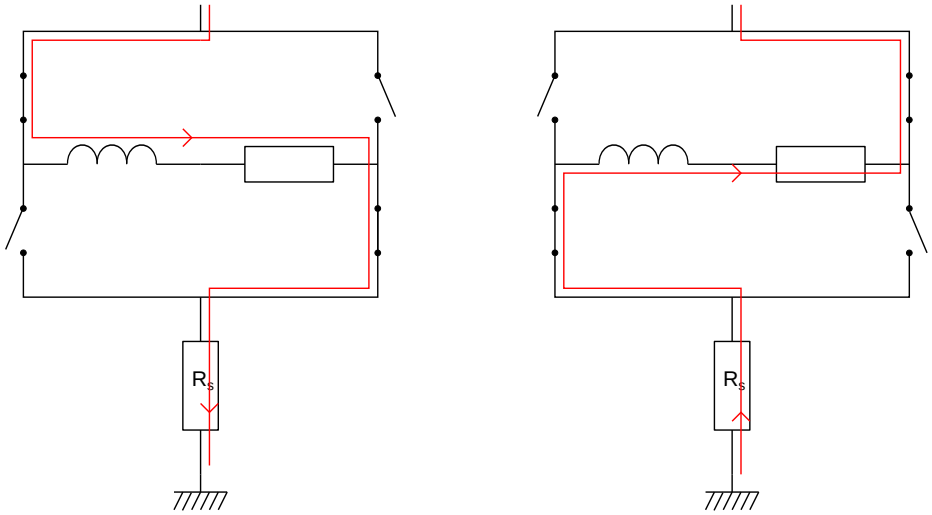
- ▷ Arduino MEGA : D11 pour le pont A, D12 pour le pont B.
- ▷ Arduino UNO ou YUN : D9 pour le pont A, D10 pour le pont B.

Le circuit logique 74LS04 (portes inverseuses) est alimenté par la sortie $+5\ V$ de l'arduino. La source de tension V_s est une alimentation de laboratoire stabilisée en tension. La tension V_s est comprise entre $12\ V$ et $48\ V$.

Voici le schéma complet :



La base de chaque pont est reliée à la masse par une petite résistance $R_s = 0,2\ \Omega$ (borne SENSE), qui permet de mesurer le courant traversant le pont. Ce courant est, au signe près, le courant qui traverse la bobine.

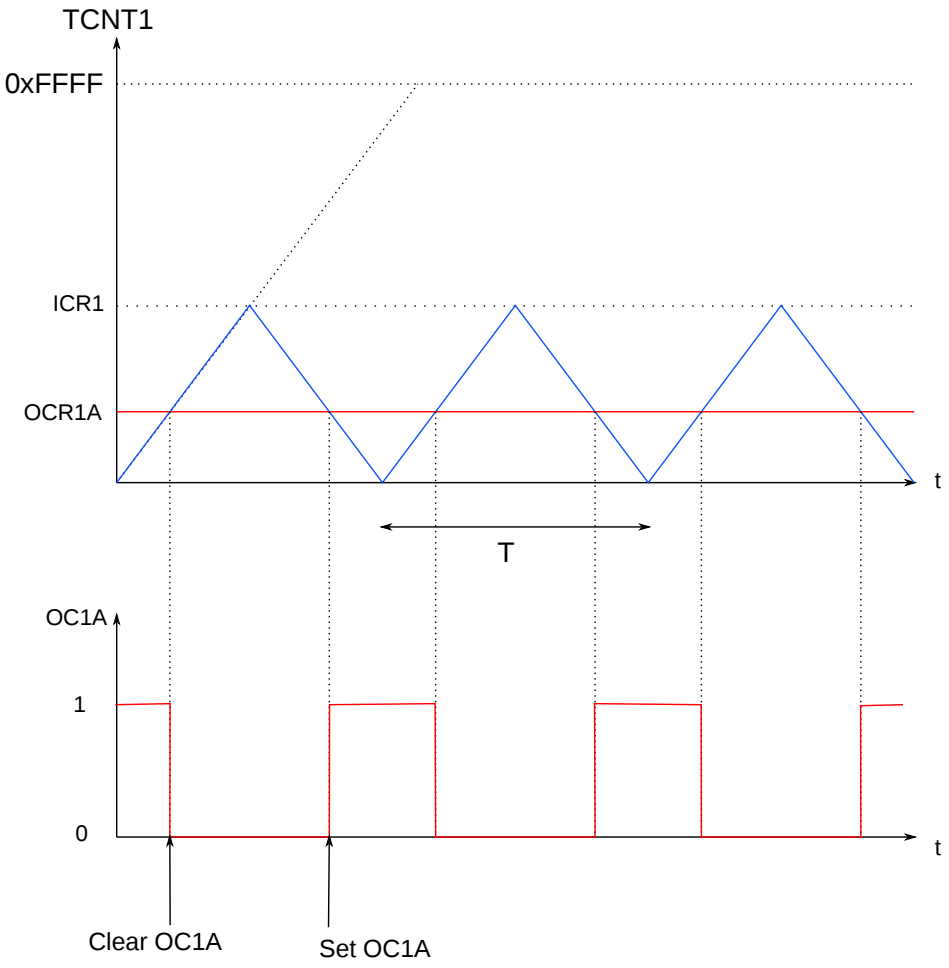


La tension aux bornes de R_s est suivie à l'oscilloscope pour obtenir le courant dans la bobine. Cette tension change de signe à chaque basculement du pont, alors que le courant dans la bobine varie continument.

4. Programme Arduino

Le Timer 1 est utilisé pour générer les deux signaux à modulation de largeur d'impulsion (MLI), comme expliqué en détail dans Génération d'un signal par modulation de largeur d'impulsion. Les sorties A et B du timer (bornes 9 et 10 de l'arduino UNO, 11 et 12 de l'arduino MEGA) sont utilisées.

La figure suivante montre comment un signal PWM est généré avec le compteur TCNT1 du Timer 1 sur la sortie A.



Le compteur a une phase de croissance jusqu'à ICR1 puis une phase de décroissance. La période de découpage T est ajustée à la fois par la durée d'un top d'horloge et par la valeur de ICR1. La sortie OC1A passe à 0 lorsque le compteur passe au dessus de OCR1A et passe à 1 lorsque le compteur passe en dessous de OCR1A. Le rapport de OCR1A par ICR1 fixe donc le rapport cyclique.

Le signal est une sinusoïde avec un éventuel décalage DC, générée avec une table et un accumulateur de phase. À chaque période T, le rapport cyclique est modifié en fonction de la valeur de la sinusoïde à cet instant.

Une impulsion est générée sur la sortie D7 au début de chaque période.

generateurPWM-2Phases.ino

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

uint32_t icr;
uint32_t table_onde_A[NECHANT];
uint32_t table_onde_B[NECHANT];
uint32_t indexA,indexB;
uint32_t accum1,accum2,increm;
uint16_t diviseur[6] = {0,1,8,64,256,1024};
```

```
volatile uint8_t flash;
uint8_t strob = 0;
```

La fonction suivante programme le Timer 1 avec une période en microsecondes (période de découpage de la modulation).

```
void init_pwm_timer1(uint32_t period) {
    char clockBits;
    TCCR1A = 0;
    TCCR1A |= (1 << COM1A1); //Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC
    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
}
```

La fonction suivante est appelée par interruption à chaque fois que le compteur dépasse la valeur maximale (et revient à zéro). Elle incrémente les deux accumulateurs de phase et met à jour les registres OCR1A et OCR1B qui définissent les rapports cycliques des sorties A et B. Par ailleurs, deux signaux numériques sont émis :

- ▷ Sur la sortie D7 : une impulsion de commande d'un flash (stroboscope), à chaque fois que la phase passe par zéro.
- ▷ Sur la sortie D2 : un signal carré synchrone avec le signal de la sortie A.

```
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==64) {
  #if defined(__AVR_ATmega2560__)
     PORTE &= ~(1 << PORTE4);
  #else
     PORTD &= ~(1 << PORTD2);
  #endif
  }
}
```

Les deux fonctions suivantes permettent de remplir les tables avec une sinusoïde dont l'amplitude et le décalage sont donnés. La somme de l'amplitude et du décalage ne doit pas dépasser 1.

```
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 sorties utilisées, programme le timer avec la période choisie et définit les deux sinusoïdes à appliquer aux deux bobines A et B.

```
void setup() {
    pinMode(8,OUTPUT);
    digitalWrite(8,HIGH); // commande ENABLE A
    pinMode(6,OUTPUT);
    digitalWrite(6,HIGH); // commande ENABLE B
#if defined(__AVR_ATmega2560__)
    pinMode(11,OUTPUT);
    pinMode(12,OUTPUT);
```

```
#elif defined(__AVR_ATmega32U4__)
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
#else
    pinMode(9,OUTPUT); // INA
    pinMode(10,OUTPUT); // INB
#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
    uint32_t period_pwm = 100; // en microsecondes (10 kHz)
    float frequence = 2; // en Hz
    accum1 = 0;
    accum2 = ((uint32_t)(NECHANT * 0.25)) << SHIFT_ACCUM;
    flash=0;
    increm = (uint32_t) (((float)(0xFFFFFFFF))*((float)(frequence)*1e-6*(float)(period_p
    init_pwm_timer1(period_pwm);
    set_sinus_table_A(0.5,0.0);
    set_sinus_table_B(0.5,0.0);
}
```

Sur cet exemple, l'accumulateur de la sortie B est initialisé pour donner un signal en quadrature par rapport à la sortie A. On peut bien sûr programmer un autre déphasage.

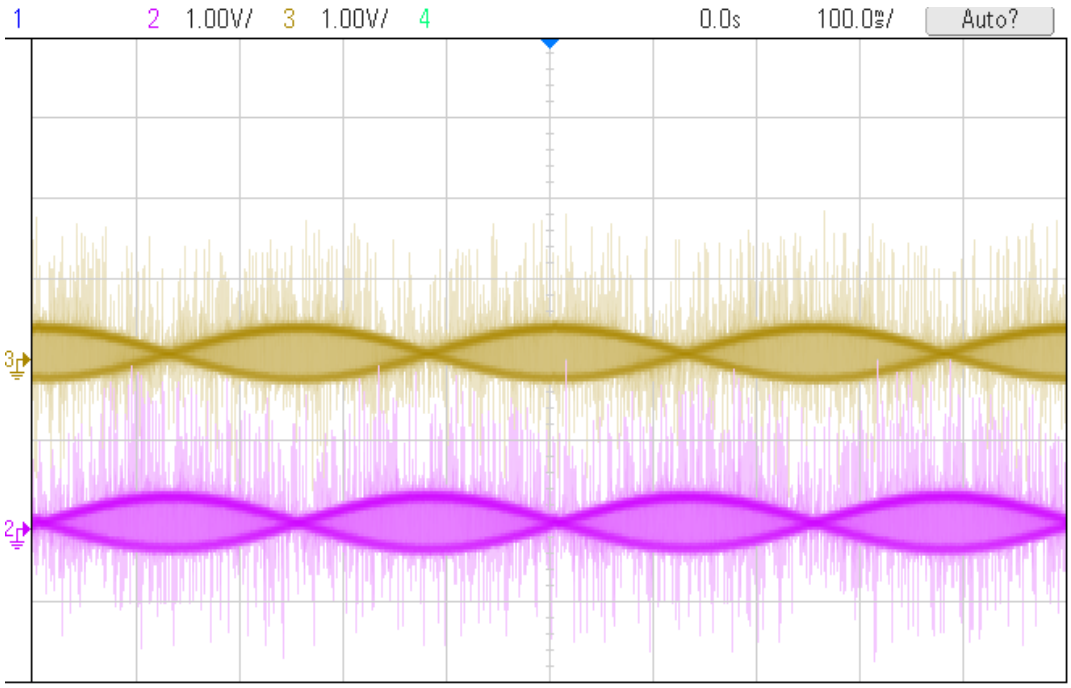
```
void loop() {

}
```

5. Contrôle du courant

Les deux bobines comportent 500 spires, ont une inductance $L = 11\ mH$ et une résistance $R = 3\ \Omega$. Une fréquence de découpage de $10\ kHz$ est suffisante pour obtenir un courant sinusoïdal, car la constante de temps L/R est de $3\ ms$ environ.

Un oscilloscope permet de contrôler la tension aux bornes de la résistance $R_s = 0,2\ \Omega$ de chaque pont. Voici la copie d'écran de l'oscilloscope pour deux sinusoïdes d'amplitude 0.5 en quadrature. La tension d'alimentation est $V_s = 12\ V$.



La trace comporte deux courbes de signes opposés, à cause du changement de signe de la tension à chaque basculement du hacheur. La valeur maximale atteinte par le courant dans chaque bobine est ici de $1,5\ A$ (tension de $0,3\ V$ sur une résistance de $0,2\ \Omega$). Elle serait de $3\ A$ avec des sinusoïdes d'amplitude 1. Pour augmenter le courant, on peut aussi augmenter la tension d'alimentation.