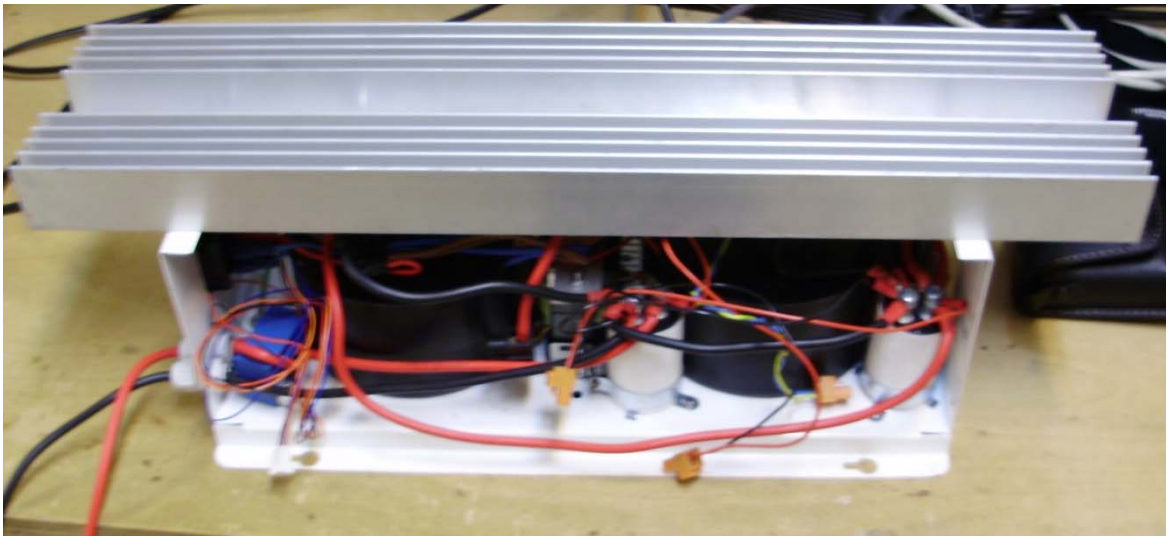
Université François Rabelais Tours
Institut Universitaire Technologique Tours
Dpt. Génie Electrique et Informatique Industrielle
http://www.iut.univ-tours.fr

Chargeur 12V/20A pour batterie plomb OPTIMA 12V 48AH



Elèves :
Benoist Alexandre
De Oliveira Benjamin-Quentin
Année 2008/2009 Groupe P2

Enseignants :
M. Lequeu
M. Brun

Sommaire

Introduction 3
1. Cahier des Charges 4
  1. Synoptiques 4
    1. Synoptique de niveau 1 4
    2. Synoptique de niveau 2 4
  2. Changement du cahier des charges 5
  3. Plannings 5
    1. Planning de prévision 5
    2. Planning réel 6
    3. Comparaison 6
2. La carte génératrice MLI 6
  1. Présentation 6
    1. But de la carte 6
    2. La batterie 6
    3. Cycle de charge 7
    4. Outils utilisés 7
  2. Schéma de la carte 8
  3. Le typon 12
  4. Le programme final de l'ATMEGA 12
    1. Déclarations préalables 13
    2. La fonction Main 15
    3. La fonction Ecrire 17
    4. La fonction Lecture_Tens 19
    5. La fonction LectureTemper 20
    6. La fonction DetCourantMax 20
    7. La fonction DetTensMax 21
    8. La fonction Traitement 22
    9. La fonction Securite 25
3. Caractéristiques électriques de la partie puissance 26
  1. Théoriques 26
    1. Schéma 27
    2. Explications 27

3. Etude théorique du hacheur Buck 27
4. Dimensionnement des composants du hacheur Buck 29
2. Pratiques 31
1. Dimensionnement des composants 31
2. Mise en Boîte finale 31
3. Tests 33
4. Relevés 34
4. Changements à apporter pour charger tout type de batterie 36
5. Etude financière 37
Conclusion 40
Tableaux 41
Bibliographie 41
Illustrations 42
Annexes 44

Introduction

Dans le cadre de l'étude et réalisation de deuxième année de formation à l'IUT GEII de Tours, nous avons réalisé un projet électronique. Notre projet de départ était de concevoir une carte électronique permettant à partir de données relevées par des capteurs de température, de courant et de tension de fabriquer un signal MLI (Modulé en Largeur d'Impulsion). Ce signal MLI permettra de contrôler un hacheur Buck (ou dévolteur) afin de pouvoir charger deux batteries mises en série de 12V 48AH chacune, batterie utilisée pour alimenter les karts électriques du Club Kart de l'IUT. Il nous a été demandé de réaliser le chargeur complet en nous aidant des projets d'études et réalisations des années passées. Nous partons sur un chargeur 24V 20A.

C'est pourquoi nous allons d'abord présenter le cahier des charges du projet puis le travail de départ c'est-à-dire la réalisation de la carte électronique générant le signal MLI en fournissant les caractéristiques du montage (composants utilisés, tensions à ne pas dépasser etc.) ; puis dans une troisième partie nous allons expliciter le schéma de la partie puissance ainsi que les problèmes rencontrés lors des tests et faire l'étude théorique de la partie puissance ainsi que l'étude pratique. Et enfin nous allons faire l'étude financière du projet réalisé.

1. Cahier des Charges

Notre travail a été de réaliser le chargeur de batterie plomb OPTIMA 12V 48AH à partir du secteur 230V. Nous sommes d'abord partis d'un chargeur de deux batteries (mises en série). Nous avons du changer notre projet à cause de problèmes de surchauffe survenant lors des tests (précisés plus loin dans le rapport).

1. Synoptiques

1. Synoptique de niveau 1

Figure 1 : Synoptique niveau 1

2. Synoptique de niveau 2

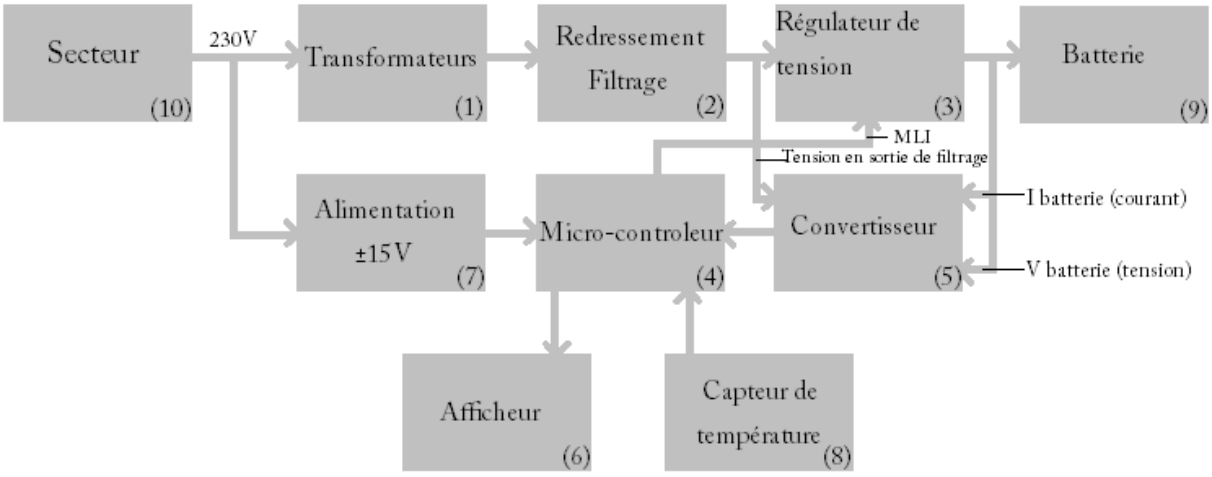


Figure 2 : Synoptique niveau 2

(1). Les transformateurs fournissent une tension efficace de 4x30V et une puissance de 600W totale.
(2). Le filtrage et le redressement sont déjà faits mais nous vérifierons si le filtrage reste de bonne qualité en charge.
(3). Une carte hacheur Buck a déjà été faite mais le courant max n'est pas assez important, nous recalculerons les composants et nous referons la partie puissance.
(4). La gestion sera faite par un microcontrôleur ATMEGA 8535. Voici les entrées et sorties :

Entrées :

- Capteur de courant
- Capteur de tension de la batterie
- Capteur de tension en sortie de redressement

Sorties :

- Signal MLI
- Afficheur LCD

(5). Pour la lecture du courant nous utiliserons un capteur spécialisé (HAS 50). Pour mesurer les deux tensions d'entrée, nous utiliserons un pont diviseur de tension avec filtrage et protection pour l'ATMEGA.
(6). Un afficheur LCD sera utilisé pour pouvoir lire les principales valeurs (courant, tensions, température). Cet afficheur sera un 4x16 (4 lignes, 16 caractères par ligne).
(7). Une carte d'alimentation sera faite pour alimenter la commande. L'alimentation sortira des tensions de ±15V, +5V et 0V.
(8). Un capteur de température sera utilisé pour régler la valeur max de charge. Les informations seront échangées grâce à un bus I2C.
(9). La batterie est de type plomb OPTIMA 12V 48AH, on en rechargera 2 en série. La tension minimale aux bornes d'une batterie est de 10V et la tension maximale est de 16V.
(10). La tension secteur est de 230V 50Hz, celle-ci varie entre -20 % et + 10%.

2. Changement du cahier des charges

Suite aux tests nous avons rencontré des impossibilités matérielles, dues à la commande du hacheur Buck et des pics de courants dus au filtrage d'entrée, nous nous sommes aperçus que le courant de sortie était limité à 10A et une tension de sortie inférieure à 32V donc cela implique que nous ne pouvions pas recharger les batteries correctement.

Nous avons décidé, avec les recommandations de M.Lequeu, de réaliser un chargeur de batterie 12V 20A.

3. Plannings

1. Planning de prévision

Semaine	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	01	02
Réflexion	X	X	X															
Choix composants			X	X														
Conception typon				X	X	X					X							
Gravure soudure						X	X		X		X							
Essais et dépannage							X		X		X	X						
Programmation				X	X	X	X											
Compte rendu				X	X								X	X				
Installation des cartes													X	X				
Oral															X			X

Tableau 1 : Planning de prévision

2. Planning réel

Semaine	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	01	02
Réflexion	X	X	X															
Choix composants			X	X														
Conception typon				X	X	X	X		X		X	X						
Gravure soudure						X	X		X		X	X						
Essais et dépannage							X		X		X	X	X					
Programmation				X	X	X	X		X		X	X	X					
Compte rendu				X	X							X	X	X				
Installation des cartes											X	X	X	X				
Oral															X			X

Tableau 2 : Planning réel

3. Comparaison

Ce qui nous a pris plus de temps que prévu a été la conception de typon et par conséquent la gravure et la soudure car il a fallu modifier la première carte. Les essais et dépannages ont pris une part plus importante du fait des tests réalisés sur les deux cartes produites. L'installation de la carte et de la partie puissance ont été elles aussi plus longues dues aux dépannages réalisés. La programmation a souvent du être refaite pour que la carte de commande puisse s'adapter aux alimentations et aux charges de tests utilisés.

2. La carte génératrice MLI

1. Présentation

1. But de la carte

On doit générer un signal MLI pour contrôler le transistor d'un hacheur Buck. Un hacheur Buck permet d'obtenir une tension en sortie (de hacheur) plus petite que la tension en entrée. La tension de sortie du hacheur a pour valeur :

$V_s = \alpha V_e$ avec V_s : tension de sortie
V_e : tension de d'entrée
$0<\alpha<1$ ---> $V_s \leq V_e$

Le but final de la carte est de gérer ce signal pour recharger une batterie.

2. La batterie

Il faut donc connaître le cycle de charge adapté à la batterie.

Référence : OPTIMA® YellowTop R 3,7.
Modèle : Y T R 3,7.
Type : plomb.
Tension nominale : 12V.

Capacité : 48AH.

Figure 3 : Batterie plomb OPTIMA 12V 48AH

3. Cycle de charge

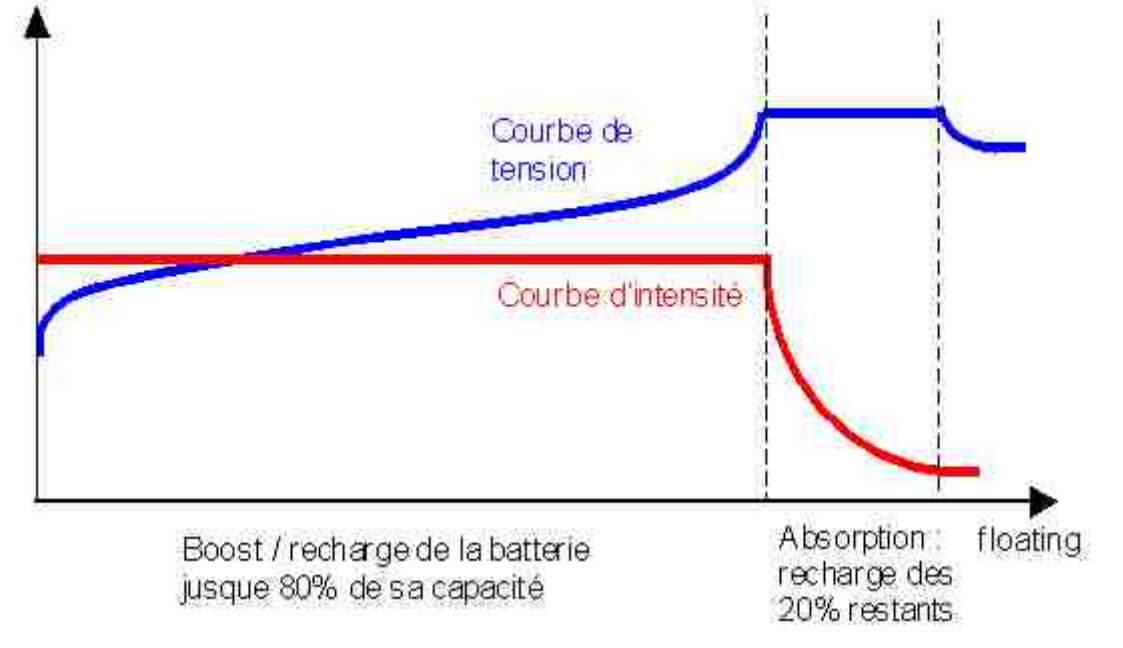


Figure 4 : Courbes tension, courant de charge

En première phase, nous devons avoir en sortie de hacheur Buck un courant constant avec une tension de sortie qui augmente. Il faut donc dans cette phase augmenter le α jusqu'à ce que la tension de sortie vale 16V, étant la tension maximale de charge de la batterie.

En seconde phase on doit conserver la tension constante jusqu'à ce que le courant circulant dans la batterie chute sous 1A.

En troisième et dernière phase on doit appliquer une tension plus basse que 13.8V et un courant inférieur à 1A : c'est le cycle de maintien.

4. Outils utilisés

Pour pouvoir charger la batterie nous devrons manipuler ces grandeurs: de tension d'entrée (du hacheur), de la tension de sortie soit la tension aux bornes de la batterie, du courant et enfin de la température. En effet, pour pouvoir charger la batterie il faut connaître l'état de la tension à ses bornes à tout moment mais aussi le courant la traversant. De plus, la batterie se charge différemment selon la température c'est pour cela qu'on utilise un capteur de température. On utilise un hacheur qui abaissera notre tension. La consigne de cette tension sera appelée par la suite alpha qui variera entre 0 et 0,95.

Les capteurs de tensions sont de simples ponts diviseur de tension. Le capteur de courant utilisé est un HAS 50. Celui-ci fournit une tension image du courant mesuré, quant au capteur de température, on utilisera un LM75 transférant ses données sur BUS I2C.

Pour réaliser le calcul de α (pour fixer la tension aux bornes de la batterie) en fonction des tensions, courant et température relevés et donc pour générer un signal Modulé en Largeur d'Impulsion, nous utiliserons un ATMEGA 8535 qui possède des convertisseurs analogiques numériques, un BUS I2C et enfin une sortie MLI.

Le signal MLI généré, il faut ensuite pouvoir contrôler le transistor du hacheur Buck. On utilisera un driver (un montage pilotant le transistor) car le signal MLI tel quel n'est pas encore adapté.

Pour alimenter tous ces composants il faudra créer des alimentations. Pour le capteur de courant il faudra une alimentation ±15V, pour l'ATMEGA et le capteur de température une alimentation +5V et enfin pour le driver une alimentation +15V.

2. Schéma de la carte

Le schéma complet se trouve en annexe. Nous allons maintenant expliquer chaque élément de la carte électronique générant le signal MLI.

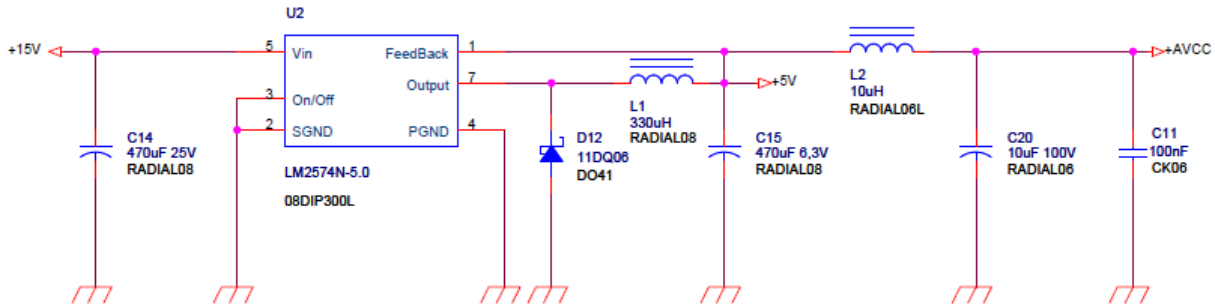


Figure 5 : Alimentation 5V

Ce schéma correspond au circuit d'alimentation 5V. Nous utilisons comme composant principal un régulateur de tension LM2574N5.0 (8 pattes). Il utilise le principe du hacheur Buck ce qui diminue les pertes dues aux pertes joules par rapport à une régulation linéaire. Les composants passifs associés au LM2574N5.0 ont été tirés de la documentation technique (voir annexes). En plus on utilisera un autre filtrage (L2 C20 C11) pour avoir une tension, encore plus stable, pour la référence du convertisseur analogique numérique de l'ATMEGA.

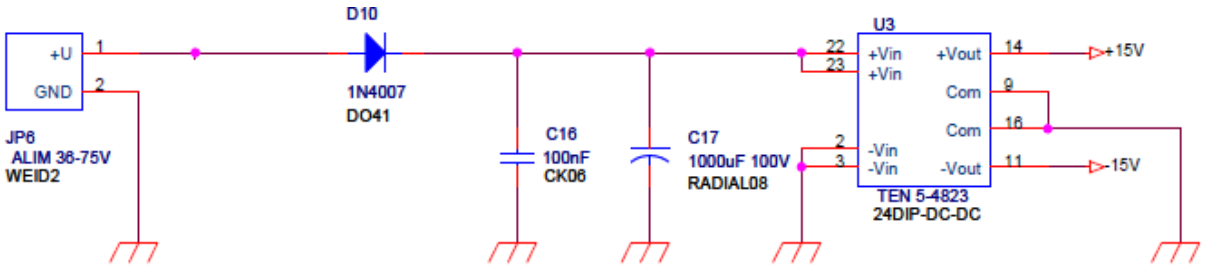


Figure 6 : Alimentation ±15V

Ce schéma nous permet d'obtenir deux tensions symétriques +15V -15V à partir de la tension redressée 42V. En effet le régulateur de tension TEN5-4823 doit avoir une tension d'entrée comprise entre 36 et 75V. La diode D10 (type 1N4007) permet de protéger le composant en cas d'inversion de la tension d'alimentation. En effet si une tension négative est appliquée en entrée du TEN5-4823 celui-ci pourrait être détruit. Le condensateur C16 (100nF) sert au découplage et à filtrer les hautes

fréquences. Le C17 (1000µF) quant à lui sert à maintenir la tension supérieure à la tension minimale du circuit en cas d'oscillation du circuit. La tension est apportée par un connecteur 2 broches JP6 (+42V, 0V).

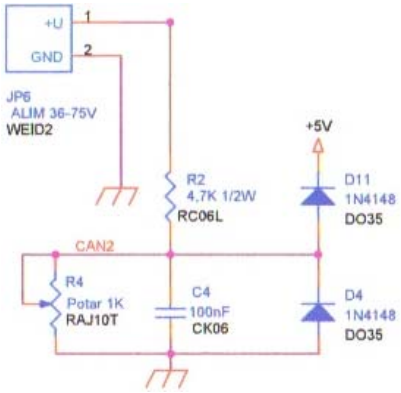


Figure 7 : Capteur de tension d'entrée d'hacheur Buck

Ce montage permet de mesurer la tension redressée. La tension d'entrée redressée est apportée par le connecteur 2 broches JP6. La tension attaque un pont diviseur de tension qui abaissera la tension qui pourra être convertie en numérique par le CAN2 de l'ATMEGA et être donc mesurée. On utilise un pont diviseur de tension car il faut avoir une tension comprise entre 0 et 5V à l'entrée des CAN de l'ATMEGA. Pour une tension d'entrée redressée de 42V on obtient une tension de 3.8V (42*R4/(R4+R2)), soit environ 10 fois moins la tension d'entrée redressée de 42V , en entrée du CAN2.Le condensateur C4 (100nF) sert à filtrer les ondulations de tension. Les diodes D4 et D11 (type 1N4148) reliées au 5V et au 0 V permettent de s'assurer que la tension à l'entrée du CAN2 soit bien comprise entre 0 et 5v pour éviter toute destruction du microcontrôleur. Le CAN a une précision de 5/1024=4,88mV/bits ainsi que tous les autres CAN suivants. La définition de la tension mesurée sera de 10 fois la précision du CAN soit 48,8mV.

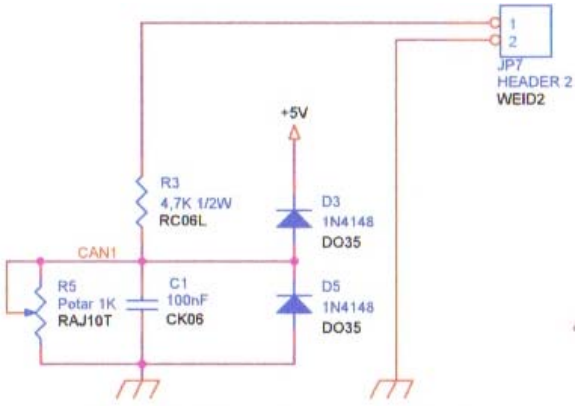


Figure 8 : Capteur de tension de la batterie

Comme le montage précédent celui-ci nous permet la lecture de la tension de sortie (aux bornes de la batterie). La tension maximale de la batterie étant de 16V. Ici le pont diviseur de tension divise la tension par 7 on obtient donc une tension maximale en entrée du CAN1 d'environ 4V: 16*672/(4700+672). Ce changement de multiple nous permet d'améliorer la précision. La définition de la tension mesurée sera de 7 fois la précision du CAN soit 34, 16mV.

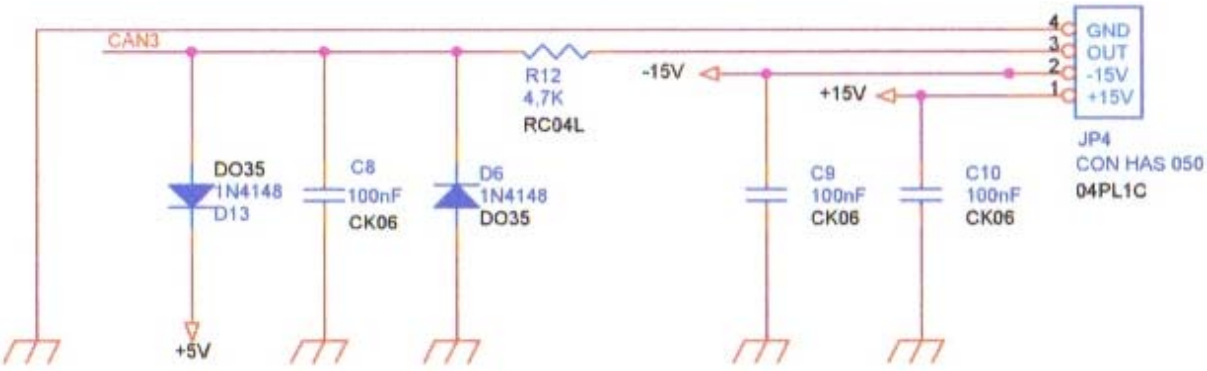


Figure 9 : Capteur de courant

Ce schéma permet de mesurer le courant de sortie (traversant la batterie). On utilise un capteur de courant alimenté en +15 -15 V. Le capteur fournit une tension Vout image de la valeur du courant mesuré. Les condensateurs, C9 et C10 sont des condensateurs de découplage. Le condensateur C8 nous permet d'éliminer les ondulations de courant. La valeur mesurée devra être multipliée par 6 pour avoir la vraie valeur de courant. La précision de la valeur de courant est de 30/1024=29,3mA.

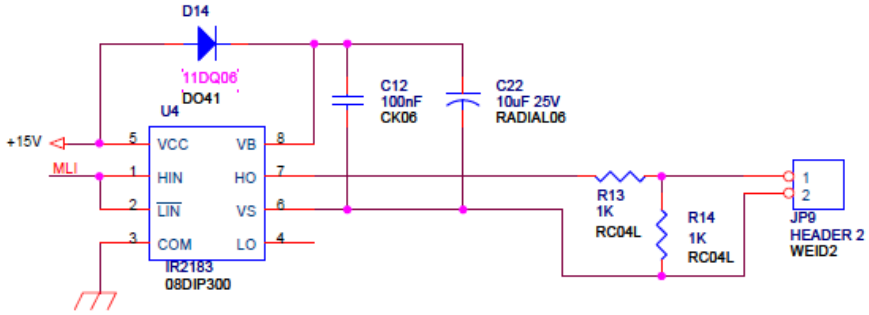


Figure 10 : Driver pilotant le transistor

Ce montage permet de contrôler le transistor du hacheur Buck. On considère le transistor comme un interrupteur, ainsi ce montage permet de le fermer et de l'ouvrir. C'est le signal MLI généré par l'ATMEGA qui permet de régler l'ouverture et la fermeture du transistor. Ce montage est alimenté par du +15V. Le montage a été tiré du rapport du groupe Kamel Agharbi et Vivien Jamet 2005/2007 qui rassemblait les informations du projet sur le hacheur Buck. Néanmoins des changements ont été apportés pour le rendre compatible avec notre transistor. On a changé la résistance R13 en série pour augmenter la tension de grille pour diminuer le temps de commutation.

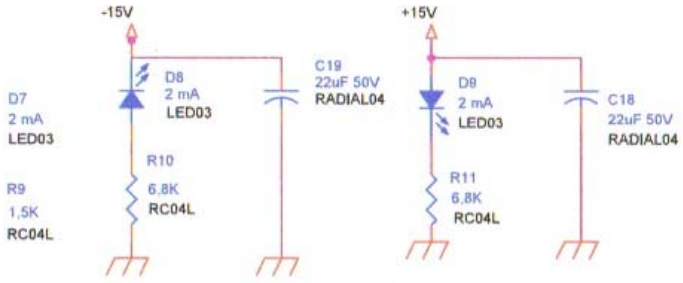


Figure 11 : DEL ±15V

Ces deux schémas permettent de voir grâce à l'allumage de DELs si on a bien les tensions +15 et -15V. Les condensateurs C18 et C19 permettent le filtrage des tensions.

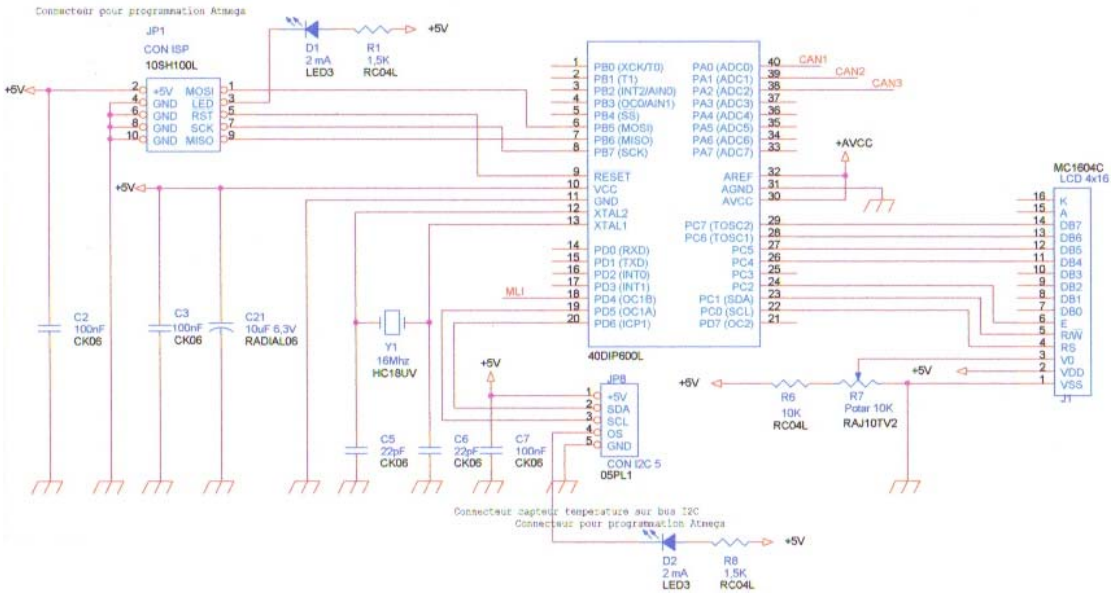


Figure 12 : ATMEGA, LCD, Capteur de température, Connecteur programmation

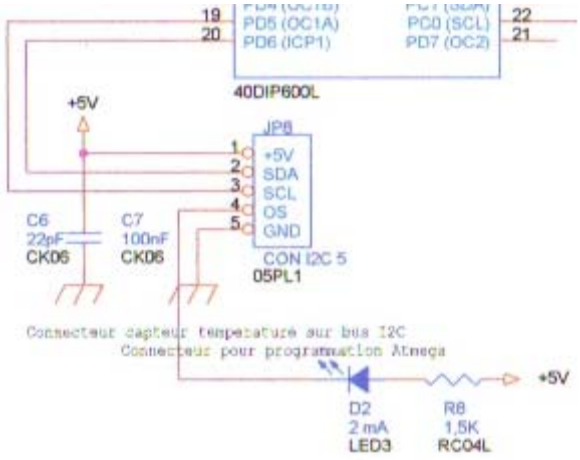


Figure 13 : Capteur de température

Le connecteur CON I2C 5 permet de brancher le capteur de courant. La broche 1 du connecteur est reliée au 5V avec un condensateur de découplage C6. Les broches 2 et 3 sont reliées respectivement aux broches 20 et 19 : ils correspondent physiquement au BUS I2C (BUS série, un fil de données et l'autre pour l'horloge), il y aura transfert de données entre le capteur de température branché sur le connecteur CON I2C 5 et l'ATMEGA. La broche 4 est reliée à une DEL en série avec une résistance (pour limiter le courant traversant la DEL soit environ 5/1500=3.33mA au +5V. Cette DEL s'allumera quand La température n'est pas bonne pour la recharge de la batterie.

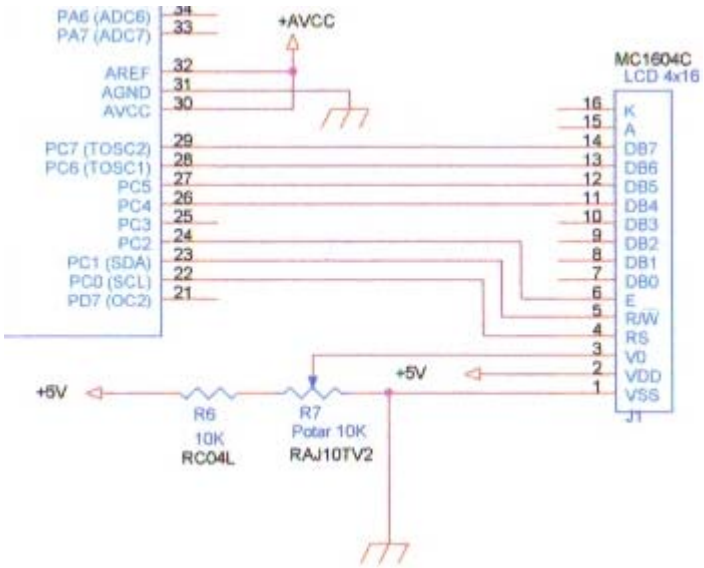


Figure 14 : LCD

La broche 31 AGND, permet de fixer la valeur de la masse des convertisseurs soit 0V. Les broches 32 AREF et 30 AVCC permettent de fixer les valeurs max en tensions des entrées ADC et autres ; on les fixe à 5V.

On a relié ensuite les broches de l'afficheur LCD à l'ATMEGA. La résistance variable permet de régler le contraste de l'afficheur LCD.

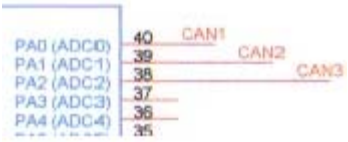


Figure 15 : Entrées CANs

Les broches 40,39 et 38 de l'ATMEGA sont réglées comme étant les entrées des 3 CANs que nous utiliserons. Le CAN 1 servira à convertir la tension aux bornes de la batterie en valeur numérique. Le CAN 2 servira à convertir la tension d'entrée en valeur numérique. Le CAN3 servira à convertir la valeur du courant passant dans la batterie en valeur numérique.

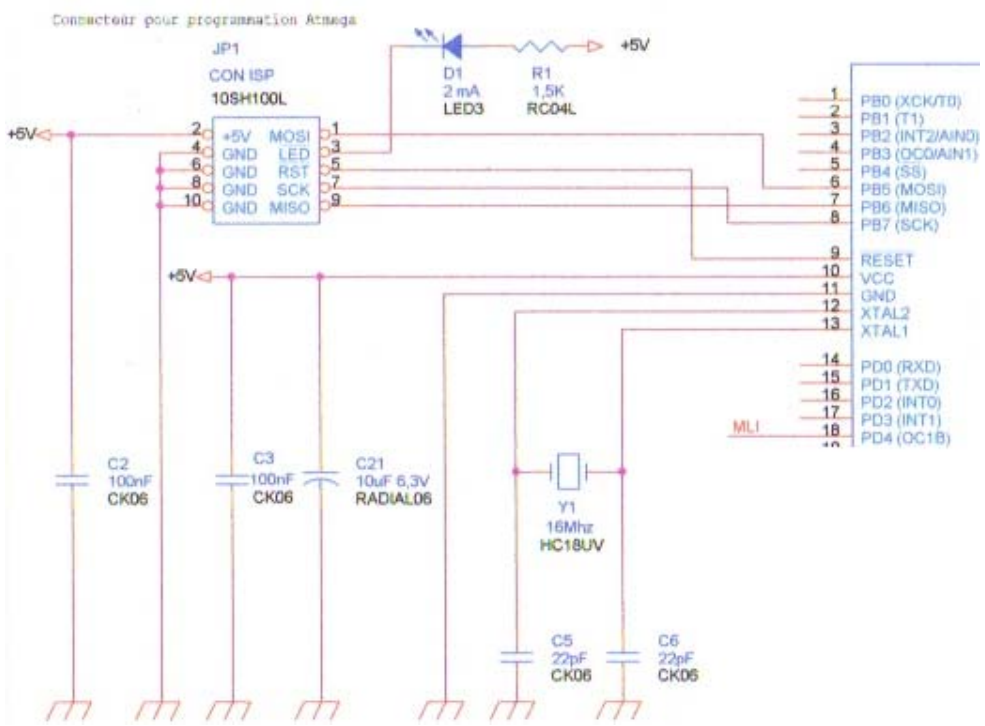


Figure 16 : ATMEGA et son connecteur de programmation

Le connecteur JP1 relié à l'ATMEGA permet de le programmer. La DEL D1 en série avec R1 branchés au +5V permet de voir si l'ATMEGA est en train d'être programmé si la DEL s'allume. Tous les condensateurs C2, C3, C21, C5 et C6 servent à découpler et à filtrer les hautes fréquences. Le composant Y1 est un oscillateur de 16Mhz (Quartz). La broche 18 de l'ATMEGA est la sortie du signal MLI généré par l'ATMEGA.

3. Le typon

Nous avons utilisé une des fonctions d'Orcad pour réaliser le typon. Un premier typon a été réalisé, il nous a permis de savoir si les fonctions principales de notre carte fonctionnaient. Celles-ci sont les relevés de tensions, de courant, de température, le bon fonctionnement des différentes alimentations fabriquées (±15V, 5V, 0V) et enfin la fabrication du signal MLI. Sur le deuxième typon présent en annexe, nous avons rajouté le driver pilotant le transistor et réduit d'environ 1/3 la taille de la carte.

4. Le programme final de l'ATMEGA

Nous allons dans cette partie expliquer le code final que nous avons entré pour charger la batterie. Pour programmer l'ATMEGA nous utilisons le logiciel CodeVisionAVR qui possède des fonctions prédéfinies pour l'ATMEGA. Le langage de programmation est le C.

1. Déclarations préalables

Nous avons programmé l'ATMEGA de façon à pouvoir changer facilement le programme pour pouvoir charger n'importe quelle batterie en changeant seulement les valeurs des variables définies au début du programme. Ces variables sont MULTITENSENT, MULTITENSSOR, MULTICOURANT, NBBAT, COURANTMAX, COURANTMAXDEC et ALMAX. Ces variables servent de bornes aux tensions, au courant et au α.

```
#define MULTITENSENT 10.0
#define MULTITNENSSOR 7.0
#define MULTICOURANT 6.0
#define NBBAT 1.0
#define COURANTMAX 22.0
#define COURANTMAXDEC 29.7
#define ALMAX 125
```

Figure 17 : Définition de variables

MULTITENSENT : définit la valeur qu'il faut multiplier à la valeur de la tension d'entrée mesurée pour obtenir la vraie valeur de tension d'entrée. Ici égale à 10,0.

MULTITENSSOR : définit la valeur qu'il faut multiplier à la valeur de la tension de sortie mesurée pour obtenir la vraie valeur de tension de sortie. Ici égale à 7,0.

MULTICOURANT : définit la valeur qu'il faut multiplier à la valeur de courant traversant la batterie mesurée pour obtenir la vraie valeur de courant traversant la batterie. Ici égale à 6,0.

NBBAT : définit le nombre de batterie à charger en série. Ici nous ne chargeons qu'une seule batterie donc NBBAT=1,0.

COURANTMAX : définit la valeur de courant max traversant la batterie. Ici égale à 22,0.

COURANTMAXDEC : définit la valeur de courant pour laquelle le chargeur sera mis en erreur. Ici égale à 29,7.

ALMAX : définit la valeur du α_{max} autorisé. Ici égale à 125 binaire soit 0,49.

```
void Ecrire(void);
void Lecture_Tens(void);
void LectureTemper(void);
void DetTensMax(void);
void DetCourantMax(void);
void Traitement(void);
void Securite(void);
```

Figure 18 : Déclarations fonctions

On déclare les fonctions dont nous aurons besoin. Toutes ces fonctions ne prennent aucun paramètre et ne renvoient rien.

La fonction **Ecrire** permet d'écrire des informations sur l'afficheur LCD.

La fonction **Lecture_Tens** permet de relever la valeur de la tension d'entrée, de sortie (tension de la batterie) et la valeur du courant (en sortie).

LectureTemper permet de lire la température à partir du capteur de température.

La fonction **DetTensMax** permet de déterminer la tension max de la batterie à charger en fonction de la température.

La fonction **DetCourantMax** permet de déterminer le courant max de la batterie en fonction de la température relevée.

La fonction **Traitement** permet de réaliser les trois stades de chargement de la batterie.

La fonction **Securite** permet de couper le chargeur en cas de valeurs (de tension ou de courant ou de température) interdites.

```
unsigned int read_adc(unsigned char adc_input);
```

Cette fonction permet de lire les valeurs des Convertisseurs Analogique Numérique qui sont dans l'ATMEGA : fonction fournie par le logiciel CodeVisionAVR.

```
float TensEnt,TensSor,Temper, Imax , Vmax, Courant , al;
char Temp[15];
char Signe;
int Etat, Timer;
```

Figure 19 : Déclaration variables globales

Ici on déclare des variables qu'on utilisera dans le programme : la tension d'entrée **TensEnt**, la tension de sortie **TensSor**, la température **Temper**, le courant max **Imax**, la tension max de la batterie **Vmax**, la valeur du courant **Courant**, le alpha **al** qui permet de déterminer ensuite la tension à appliquer, une variable temporaire pour afficher des informations **Temp[15]** (tableau de 15 caractères), le signe de la température ± **Signe**, **Etat** qui correspondra à l'état du chargeur (0 : attente, 1 : chargement, 2 : charge complète et 3 : défaut) et le temps de rafraichissement de l'affichage sur LCD **Timer**. Ces variables sont déclarées en globales car l'ATMEGA n'avait pas assez d'espace mémoire pour pouvoir travailler avec toutes ces valeurs. En effet lorsque les variables sont déclarées en globales celles-ci sont rangées dans un registre ce qui nous permet de travailler avec plus d'octets.

2. La fonction Main

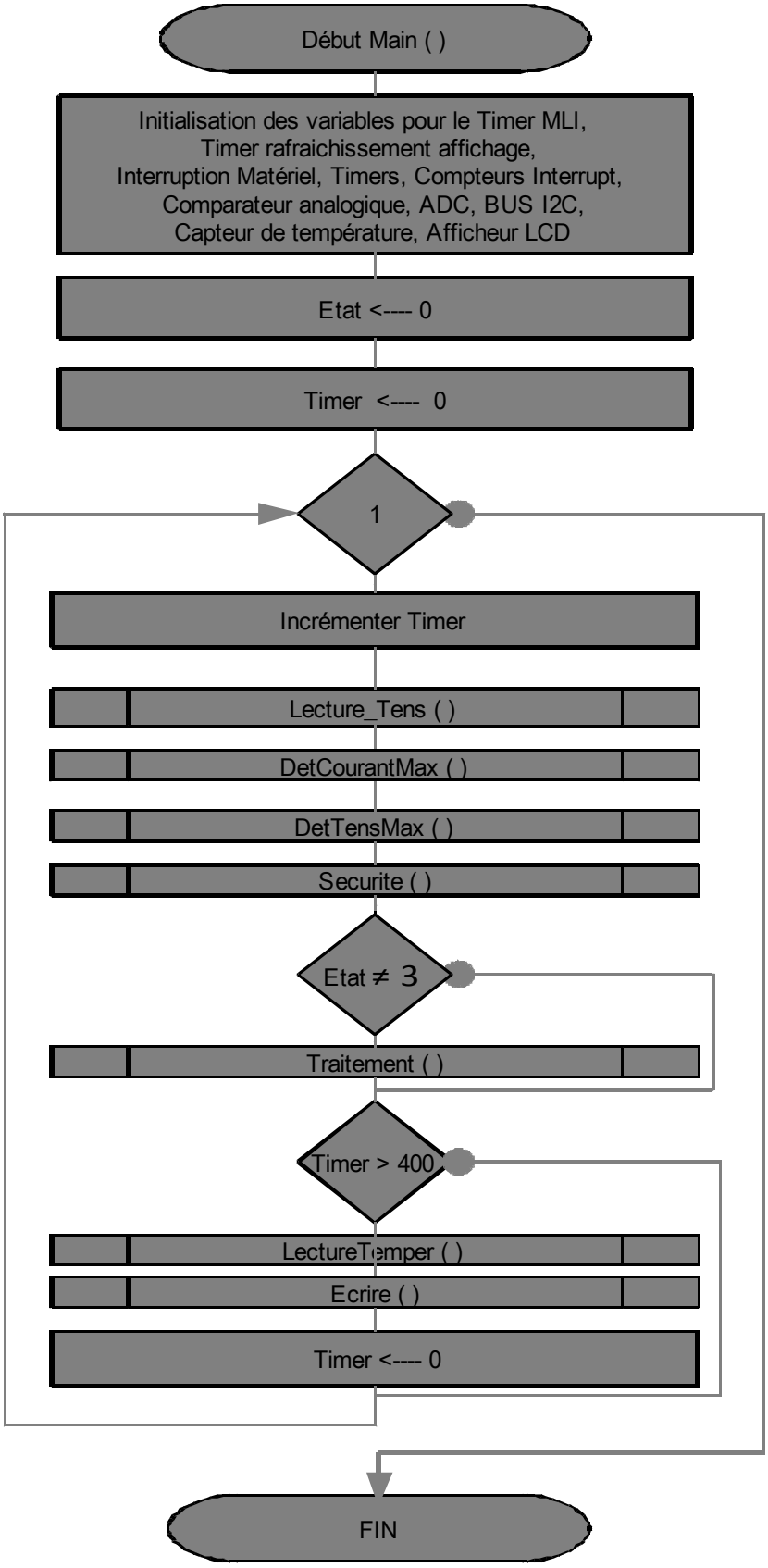


Figure 20 : Ordinogramme Fonction MAIN

Première partie :

```
void main(void)
{
PORTA=0x00;
DDRA=0x00;

PORTB=0x00;
DDRB=0x00;

PORTC=0x00;
DDRC=0x00;

PORTD=0x00;
DDRD=0x10;

TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
// Timer pour MLI
TCCR1A=0x21;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer pour raffraichisement affichage
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
//Interruption materiel
MCUCR=0x00;
MCUCSR=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
//comparateur analogique
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

SFIOR&=0xEF;

i2c_init();
lm75_init(3,75,80,0);

lcd_init(16);
```

Figure 21 : Déclaration variables locales et initialisation fonctions de l'ATMEGA

Déclaration des Entrées/Sorties et de la configuration des différents éléments de l'ATMEGA, code fourni par le logiciel.

Deuxième partie :

```
Etat = 0;
Timer = 0;
while (1)
      {
          Timer++;
          Lecture_Tens();
          DetCourantMax();
          DetTensMax();

          Securite();
          if( Etat != 3)
          Traitement();
          if( Timer > 400 )
          {
            LectureTemper();
            Ecrire();
            Timer = 0;
          }
      };
}
```

Figure 22 : Code 2eme partie de la fonction MAIN

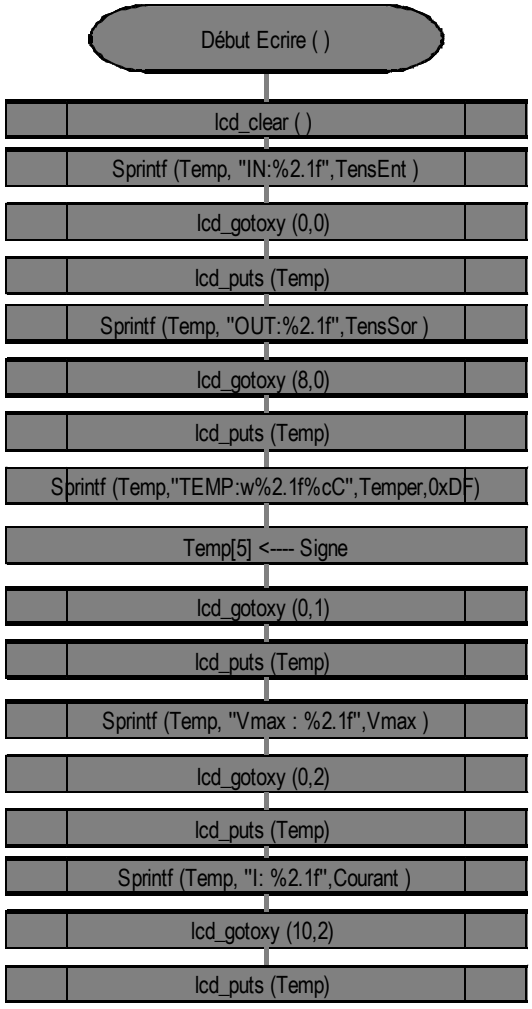
Dès le début du programme on met le chargeur en état d'attente : **Etat=0**. On initialise le temps de rafraichissement à 0 : **Timer=0**. On réalise une boucle infinie pour que le programme fonctionne tout le temps : **while(1)**. A chaque tour de la boucle, on incrémente la variable **Timer** : **Timer++**.

On fait ensuite appel à la fonction **Lecture_Tens**. Puis on fait appel à la fonction **DetCourantMax**, puis **DetTensMax** et enfin à la fonction **Securite**.

On réalise un test, si le chargeur n'est pas en défaut **if(Etat !=3)** on fait

appel à la fonction **Traitement** qui va réaliser le chargement de la batterie. On réalise un second test si la valeur de la variable **Timer** est supérieure à 400 **if(Timer>400)** on fait appel à la fonction **LectureTemper** qui permettra de lire la température. Ensuite on fait appel à la fonction **Ecrire** qui permettra d'afficher des informations sur l'afficheur LCD. Et enfin on remet à zéro la valeur de la variable **Timer Timer=0** pour que 400 cycles plus tard on puisse rafraichir la valeur de la température et les informations affichées sur l'afficheur LCD.

3. La fonction Ecrire




```
void Ecrire(void)
{
        lcd_clear();
        sprintf(Temp,"IN:%2.1f",TensEnt);
        lcd_gotoxy(0,0);
        lcd_puts(Temp);
        sprintf(Temp,"OUT:%2.1f",TensSor);
        lcd_gotoxy(8,0);
        lcd_puts(Temp);
        sprintf(Temp,"TEMP:w%2.1f%cC",Temper,0xDF );
        Temp[5]=Signe;
        lcd_gotoxy(0,1);
        lcd_puts(Temp);
        sprintf(Temp,"Vmax:%2.1f",Vmax);
        lcd_gotoxy(0,2);
        lcd_puts(Temp);
        sprintf(Temp,"I:%2.1f",Courant);
        lcd_gotoxy(10,2);
        lcd_puts(Temp);
        if(Etat == 0)
        sprintf(Temp,"Attente");
        if(Etat == 1)
        sprintf(Temp,"Charge");
        if(Etat == 2)
        sprintf(Temp,"Charge fini");
        if(Etat == 3)
        sprintf(Temp,"Erreur !");
        lcd_gotoxy(0,3);
        lcd_puts(Temp);
}
```

Figure 23 : Code de la fonction ECRIRE

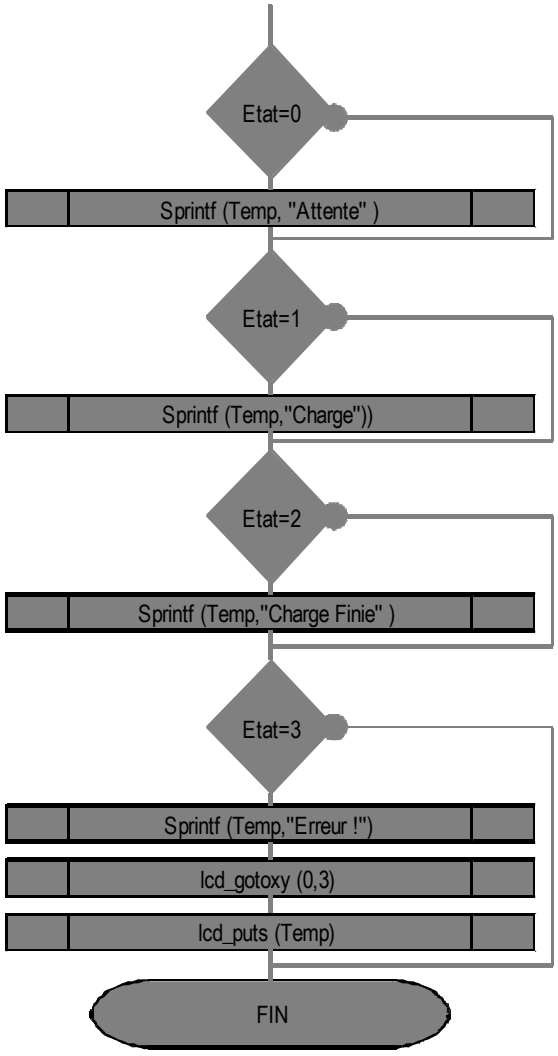


Figure 24 : Ordinogramme fonction ECRIRE

Lors de l'appel de la fonction **Ecrire** on réalise avec la fonction **lcd_clear** un effacement des informations écrites sur l'afficheur LCD.

On utilise la fonction **sprintf** pour écrire dans la variable **Temp** la valeur de la tension d'entrée mesurée **TensEnt** : **sprintf(Temp,''IN :%2.1f'',TensEnt);**. On se place ensuite sur le caractère [0,0] (colonne 0 ligne 0) de l'afficheur LCD : **lcd_gotoxy(0,0) ;**.

On inscrit ensuite la chaîne de caractères contenue dans **Temp** sur l'afficheur LCD : **lcd_puts(Temp);**.

On utilise la fonction **sprintf** pour écrire dans la variable **Temp** la valeur de la tension de sortie (ou tension aux bornes de la batterie) mesurée **TensSor** : **sprintf(Temp,''OUT :%2.1f'',TensSor);**. On se place ensuite sur le caractère [8,0] (colonne 8 ligne 0) de l'afficheur LCD : **lcd_gotoxy(8,0);**.

On inscrit ensuite la chaîne de caractères contenue dans **Temp** sur l'afficheur LCD : **lcd_puts(Temp);**.

On utilise la fonction **sprintf** pour écrire dans la variable **Temp** la valeur de la température mesurée **Temper** : **sprintf(Temp,''TEMP :w%2.1f%cC'',Temper,0xDF);**.

On remplace ensuite la lettre w contenue dans **Temp** par le signe : **Temp[5]=Signe**. On se place ensuite sur le caractère [0,1] (colonne 0 ligne 1) de l'afficheur LCD : **lcd_gotoxy(0,1);**.

On inscrit ensuite la chaîne de caractères contenue dans **Temp** sur l'afficheur LCD : **lcd_puts(Temp);**.

On utilise la fonction **sprintf** pour écrire dans la variable **Temp** la valeur de la tension max que doit avoir la tension aux bornes de la batterie **Vmax** : **sprintf(Temp,''Vmax:%2.1f'',Vmax);**. On se place ensuite sur le caractère [0,2] (colonne 0 ligne 2) de l'afficheur LCD : **lcd_gotoxy(0,2);**.

On inscrit ensuite la chaîne de caractères contenue dans **Temp** sur l'afficheur LCD : **lcd_puts(Temp);**.

On utilise la fonction **sprintf** pour écrire dans la variable **Temp** la valeur du courant **Courant**: **sprintf(Temp,''I:%2.1f'',Courant);**. On se place ensuite sur le caractère [10,2] (colonne 10 ligne 2) de l'afficheur LCD : **lcd_gotoxy(10,2);**.

On inscrit ensuite la chaîne de caractères contenue dans **Temp** sur l'afficheur LCD : **lcd_puts(Temp);**.

On réalise ensuite des tests :

Si **Etat** vaut **0**, le chargeur est en attente, on copie **Attente** dans la variable **Temp** : **if (Etat==0) sprintf(Temp,''Attente'');**.

Si **Etat** vaut **1**, il y a chargement de la batterie, on copie **Charge** dans la variable **Temp** : **if (Etat==1) sprintf(Temp,''Charge'');**

Si **Etat** vaut **2**, le chargement est terminé, on copie **Charge finie** dans la variable **Temp** : **if (Etat==2) sprintf(Temp,''Charge fini'');**

Si **Etat** vaut **3**, le chargeur est en erreur (ou défaut), on copie **Erreur !** dans la variable **Temp** : **if (Etat==3) sprintf(Temp,''Erreur !'');**.

On se place ensuite sur le caractère [0,3] (colonne 0 ligne 3) de l'afficheur LCD : **lcd_gotoxy(0,3);**.

On inscrit ensuite la chaîne de caractères contenue dans **Temp** sur l'afficheur LCD : **lcd_puts(Temp);**.

4. La fonction Lecture_Tens

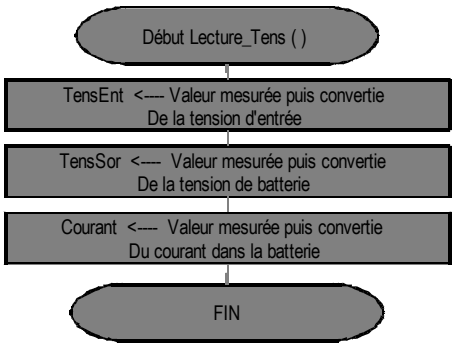


Figure 25 : Ordinogramme de la fonction Lecture_Tens

```
void Lecture_Tens(void)
{
        TensEnt = (float)((float)(read_adc(1))*MULTITENSENT*5.0/1024.0);
        TensSor = (float)((float)(read_adc(0))*MULTITNENSSOR*5.0/1024.0);
        Courant = (float)((float)(read_adc(2))*MULTICOURANT*5.0/1024.0);
}
```

Figure 26 : Code de la fonction Lecture_Tens

Cette fonction lit les informations provenant des Convertisseurs Analogique Numérique (CANs). Dans un premier temps, on relève la tension d'entrée **TensEnt**; pour cela on utilise la fonction **read_adc**. On lit la valeur du CAN 1 : **read_adc(1)**. On utilise l'opérateur de cast **float** pour transformer le type renvoyé par la fonction **read_adc** : transforme un **int** (nombre entier) en **float** (nombre décimal). On multiplie cette valeur par **MULTITENSENT** (ici=10) car la tension relevée par le CAN1 correspond à la tension d'entrée divisée par 10, puisqu'on a utilisé un pont diviseur de tension pour pouvoir appliquer une tension comprise entre 0 et 5V supportée par le CAN1. La valeur renvoyée par le CAN est comprise en 0 et 1024. Donc pour obtenir une valeur en volts il faut la multiplier par la valeur max en volts supportée par le CAN et diviser par le nombre max renvoyé par le CAN pour 5V c'est à dire 1024. On utilise l'opérateur de cast **float** pour être sûr d'attribuer à la variable **TensEnt** un nombre décimal : **TensEnt=(float)((float)(read_adc(1))* MULTITENSENT *5.0/1024.0);**.

Ensuite on relève la tension de sortie **TensSor** ; on lit la valeur du CAN0 : **read_adc(0)**. On multiplie cette valeur par **MULTITENSSOR** (ici égale à 7) car la tension relevée par le CAN0 correspond à la tension d'entrée divisée par 7, puisqu'on a utilisé un pont diviseur de tension pour pouvoir appliquer une tension comprise entre 0 et 5V supportée par le CAN0. La conversion est la même que

précédemment : on multiplie par 5 et divise par 1024. On utilise les opérateurs de **cast** pour les mêmes raisons que précédemment.

Enfin on relève la valeur du courant **Courant** ; on lit la valeur du CAN2 : **read_adc(2)**. On multiplie cette valeur par 6 car la tension relevée par le CAN2 correspond à la tension d'entrée divisée par 6. La conversion est la même que précédemment : on multiplie par 5 et divise par 1024. On utilise les opérateurs de cast pour les mêmes raisons que précédemment.

5. La fonction LectureTemper

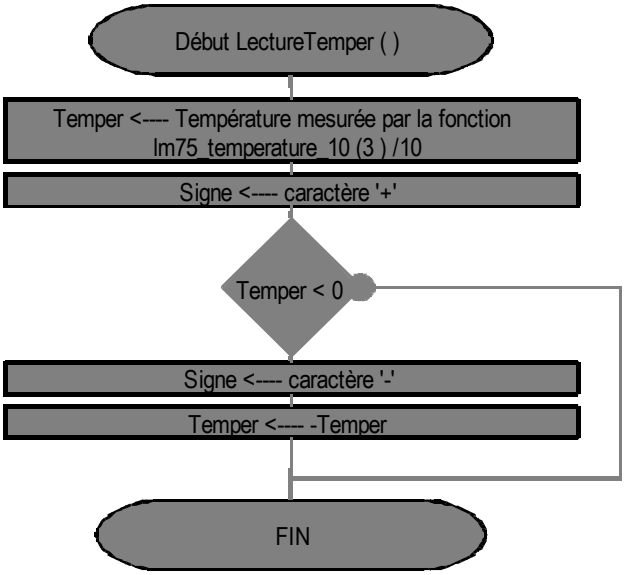


Figure 27 : Ordinogramme de la fonction LectureTemper

```
void LectureTemper(void)
{
        Temper = (float)lm75_temperature_10(3)/10.0;
          Signe=0x2B;
          if(Temper<0)
          {
           Signe=0x2D;
           Temper=-Temper;
          }
}
```

Figure 28 : Code de la fonction LectureTemper

Cette fonction **LectureTemper** permet d'envoyer une demande de mesure de température au capteur par bus I2C. On attribue tout d'abord à la variable globale **Temper** la valeur renvoyée par le capteur de température divisée par 10. Pour pouvoir faire cette division et garder les chiffres après la virgule, on utilise l'opérateur de **cast (float)** ; en effet la fonction **lm75_temperature_10** renvoie une valeur entière, donc si par exemple le capteur renvoie 9 et que l'on fait la division par 10 on obtiendrait 0 au lieu de 0.9, on augmente donc la précision de la mesure en utilisant un nombre décimal. On doit diviser par 10 car le capteur renvoie 10 pour 1°C. La fonction **lm75_temperature_10** prend un paramètre, il correspond à l'adresse du capteur : ici l'adresse du capteur est **3**.

Ensuite on attribue à la variable globale **Signe** le caractère **+** (code ascii **0x2B**) : cela correspond au signe par défaut. On réalise ensuite un test pour savoir si la température est bien positive ou au contraire négative. Si la valeur de la variable globale **Temper** est inférieure à 0, on change alors le contenu de la variable **Signe** par le caractère **-** (code ascii **0x2D**) et on change la valeur de la variable **Temper** en y attribuant sa valeur opposée. Il aurait été plus simple de ne pas mettre de variable **Signe** et donc d'afficher directement la valeur de la variable **Temper** mais on a été confronté à un problème d'affichage et donc on a décidé de procéder de cette façon pour que l'affichage de la température puisse se faire correctement.

6. La fonction DetCourantMax

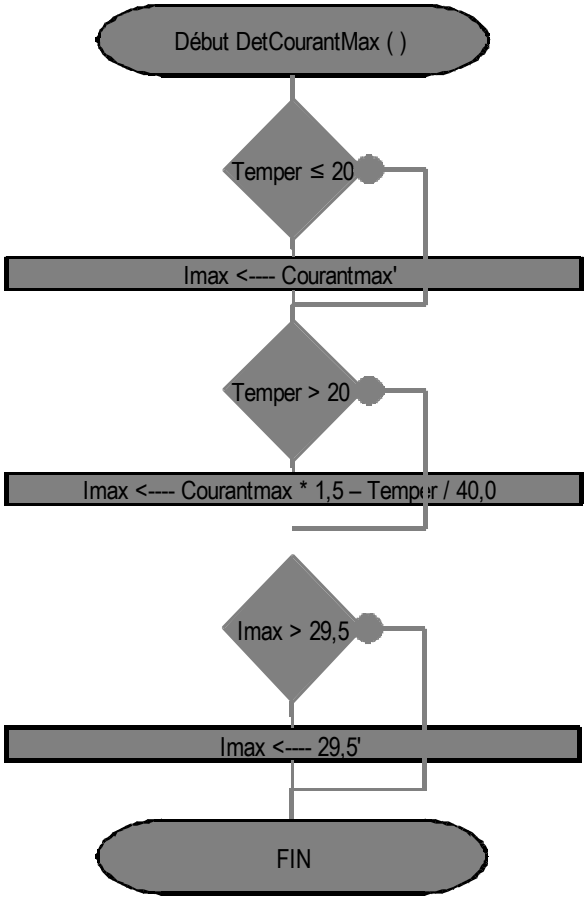


Figure 30 : Ordinogramme de la fonction DetCourantMax

```
void DetCourantMax(void)
{
 if ( Temper <= 20 )
       Imax = COURANTMAX;
 if ( Temper > 20 )
       Imax = COURANTMAX *(1.5-Temper/40.0);
 if ( Imax > 29.5 )
       Imax = 29.5;
}
```

Figure 29 : Code de la fonction DetCourantMax

Cette fonction permet de déterminer le courant max qui doit aller à la batterie. Celui-ci dépend de la température (représentée par la valeur de la variable **Temper**) et aussi du **α** (représentée par la valeur de la variable **al**). On réalise donc deux tests pour deux valeurs de température différentes. Tout d'abord si la température est inférieure ou égale à 20°C le courant max (**Imax**) sera égal à **COURANTMAX**. Si la température est supérieure à 20°C le courant max (**Imax**) sera égal à **COURANTMAX*(1.5-Temper/40.0)** : le courant max décroit affinement quand la température augmente.

7. La fonction DetTensMax

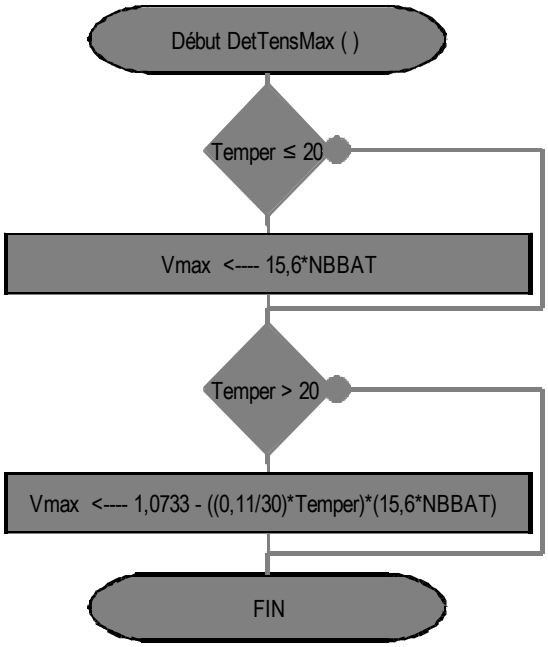


Figure 31 : Ordinogramme de la fonction DetTensMax

```
void DetTensMax(void)
{
    if (Temper<=20)
          Vmax = (15.6 * NBBAT);
    if (Temper>20)
          Vmax = (1.0733 - ((0.11/30)*Temper))* (15.6 * NBBAT);
}
```

Figure 32 : Code de la fonction DetTensMax

Cette fonction permet de déterminer la tension max **Vmax** aux bornes de la batterie qui dépend de la température (représentée par **Temper**). Si la température est inférieure ou égale à 20°C la tension max est égale à 15.6V fois le nombre de batterie : **Vmax=15.6*NBBAT**. Si la température est supérieure à 20°C la tension max est égale à **(1.0733-((0.11/30)*Temper))*(15.6*NBBAT)**.

8. La fonction Traitement

```
void Traitement(void)
{

    if( Etat ==0 && TensSor <= (13.0 * NBBAT) &&TensSor >= (10.5 * NBBAT) ) //Si la tension batterie est comprise
entre 20 et 26.2 et que alpha = 0 on lance le cycle de charge
        {
          Etat = 1;
          al = ((float)(TensSor/TensEnt));
          OCR1BL = al*256 + 6;
        }

    if( Etat == 0 && TensSor > (13.0 * NBBAT) && TensSor < (13.1* NBBAT) ) //Si la tension batterie est comprise
entre 26.1 et 27.6 et que alpha = 0 on lance  le cycle de maintien
        {
          al = ((float)(TensSor/TensEnt));
          OCR1BL = al*256 + 6;
          Etat = 2;
        }

    if( Courant < 1 && Etat == 1 && TensSor  > Vmax - 0.4 ) //(Etat fin de charge normalement) Charge terminée

          Etat = 2;

    if ( Etat == 1 ) // gestion de la charge
        {
          if ( TensSor + 0.3 < Vmax && Courant + 0.4 < Imax )
             if( OCR1BL < ALMAX )OCR1BL++;

          if ( TensSor >= Vmax || Courant > Imax )
             if( OCR1BL >0 )OCR1BL--;

          al = OCR1BL/256;
        }


    if ( Etat == 2 ) //gestion du cycle d'entretien
        {
          if ( Courant + 0.2 < 1 && TensSor + 0.2 < (13.8 * NBBAT) )
                if( OCR1BL < ALMAX ) OCR1BL++;
          if ( Courant > 1 || TensSor > (13.8* NBBAT - 0.2))
                if( OCR1BL >0 ) OCR1BL--;

          al = OCR1BL/256;
        }

    if( Courant < 0.1 || TensSor < 2 ) //(Deconnection de la batterie) passage en mode attente

          Etat = 0;


    if( Etat == 0 )
        {
          al = 0;
          OCR1BL = 0;
        }

}
```

Figure 33 : Code de la fonction Traitement

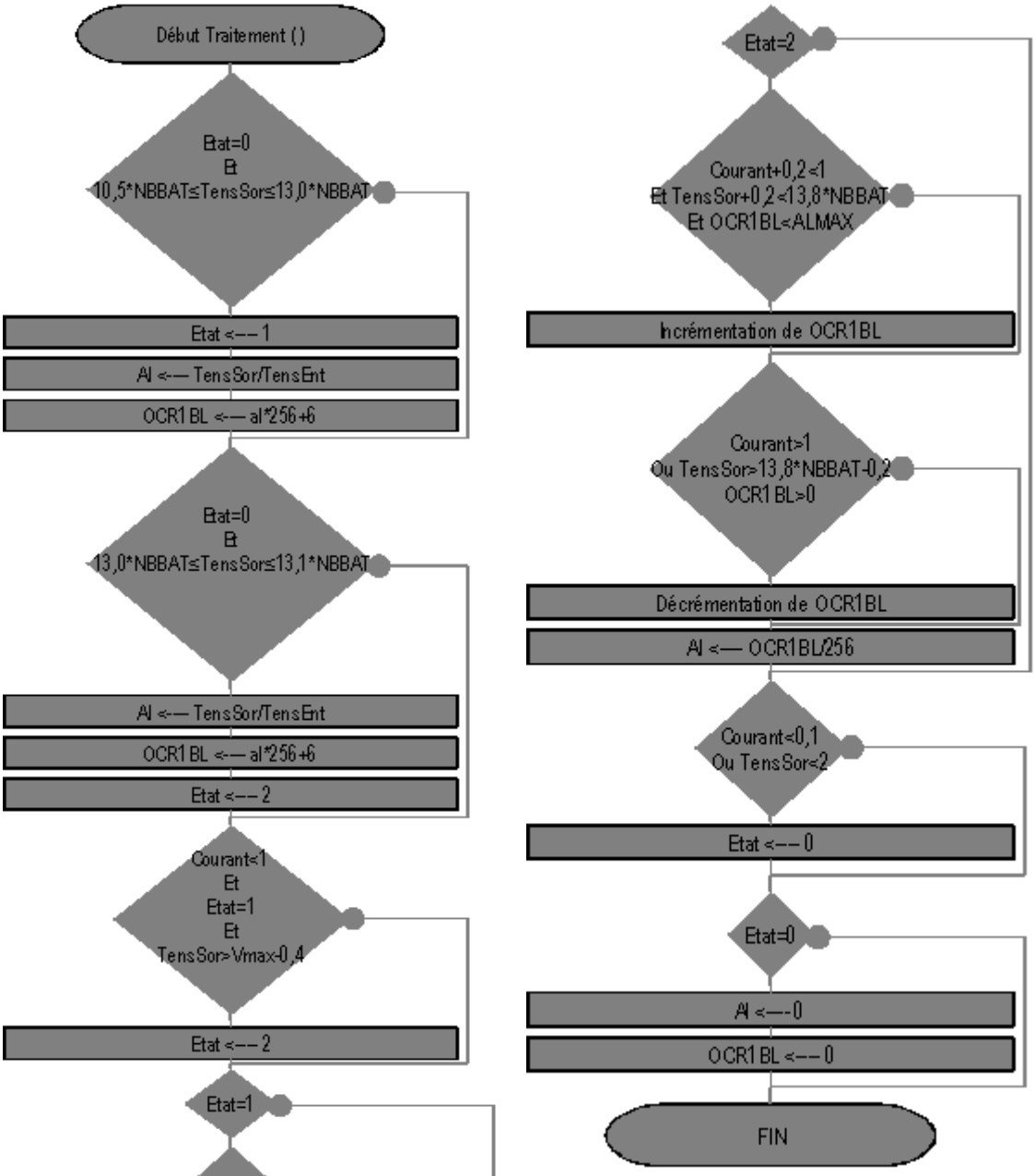


Figure 34 : Ordinogramme de la fonction Traitement

Cette fonction va tout simplement traiter les informations reçues des CANs (tensions, courant, température) et agir en conséquence : elle fera varier le alpha pour faire varier la tension dans le but de charger la batterie.

Premier Test :

Si le chargeur est en attente (**Etat==0**) et que la tension de sortie est comprise entre **10.5V par batterie** et 13.0V par batterie (**&&TensSor<=13.0*NBBAT&&TensSor>=10.5*NBBAT**) alors on lance le chargement (**Etat=1 ;**) puis on

définit combien vaut le alpha (**al=((float)(TensSor/TensEnt)) ;** al étant un nombre décimal). Puis on attribue à **OCR1BL** (qui est le alpha physique qui sert à générer le MLI avec l'ATMEGA) la valeur de **al** divisé 256 bits : **OCR1BL= al*256+6;** ; le **+6** n'est là que par sécurité pour éviter que la tension délivrée par le chargeur soit inférieure à la tension de la batterie. Si cela n'était pas le cas, on pourrait avoir un courant négatif ce qui causerait la destruction de composants.

Deuxième Test :

Si le chargeur est en attente (**Etat==0**) et que la tension de sortie est comprise entre 13.0V par batterie et **13.1V par batterie** (**&&TensSor>=13.0*NBBAT&&TensSor<=13.1*NBBAT**). On définit la valeur de alpha (**al=((float)(TensSor/TensEnt)) ;** Puis on attribue à **OCR1BL** : **OCR1BL= al*256+6 ;** (mêmes remarques que précédemment). Puis on met le chargeur en mode **cycle de maintien** qui consiste seulement à appliquer une tension légèrement supérieure à la tension nominale de la batterie.

Troisième Test :

Si le courant est inférieur à **1A** (**Courant<1**) et que le chargeur est en mode **chargement** (&&**Etat==1**) et que la tension de sortie est supérieure à la tension max moins 0.4V, définie précédemment dans la fonction **DetTensMax**, (**&&TensSor>Vmax-0.4**). On met le chargeur en mode **cycle de maintien** (**Etat=2 ;**).

Quatrième Test :

Celui-ci permet de charger réellement les batteries. Si le chargeur est en **attente** (**if(Etat==1)**), on pourra augmenter ou diminuer la valeur du alpha et donc de la tension.

Pour augmenter la valeur du alpha il faut que **la tension de sortie** moins **0.3V** soit inférieure à **la tension max** (**Vmax**) et que le courant moins **0.4A** soit inférieur au courant max **Imax** (défini précédemment dans une autre fonction **DetCourantMax**) et que le alpha (représenté physiquement par OCR1BL) soit inférieur à **ALMAX** valeur correspondant à alpha : **if(TensSor - 0.3<Vmax && Courant - 0.4<Imax) if(OCR1BL<ALMAX)OCR1BL++ ;**. En effet en augmentant le alpha on augmente la tension pour obtenir un courant constant ce qui correspond à la première phase de chargement de la batterie c'est-à-dire avoir un courant constant jusqu'à atteindre la tension max de la batterie.

Si maintenant la tension de sortie est supérieure ou égale à la tension max de la batterie ou bien que le courant soit supérieur au courant max, on doit absolument diminuer le alpha (représenté par **OCR1BL**) pour baisser la tension pour ne rien abimer : **if (TensSor >= Vmax || Courant > Imax) if(OCR1BL >0)OCR1BL--;**.

On décide de donner la valeur de Alpha en valeur numérique que l'on utilise en théorie : **al = OCR1BL/256;** conversion de binaire en valeur numérique.

Cinquième Test :

Si on est en état de cycle de maintien, on doit avoir une tension de sortie (de la batterie) constante égale à 13.8V. Donc si la tension de sortie est inférieure à **13.8V** et que le courant est inférieur à **1A** on augmente le alpha (si celui-ci est inférieur à ALMAX, en valeur binaire) pour que la tension de

sortie puisse s'approcher de ces **13.8V** : **if(Courant - 0.2 < 1 && TensSor + 0.2 <13.8*NBBAT)if(OCR1BL < ALMAX) OCR1BL++;**.

Cependant, si le courant est trop important (supérieur à **1A**) ou que la tension de sortie est supérieure à **13.8V-0.2** alors on doit baisser la tension de sortie et donc diminuer la valeur du alpha : **if (Courant > 1 || TensSor >13.8*NBBAT-0.2)if(OCR1BL >0) OCR1BL--;**.

On décide de convertir la valeur du alpha binaire en valeur décimale : **al = OCR1BL/256;**.

Sixième Test :

Si on obtient un courant très faible (inférieur à **0.1A**) ou que la tension aux bornes de la batterie soit trop faible aussi (inférieure **2V**) alors on coupe le chargement : mise du chargeur à l'état d'attente (**Etat=0 ;**) : **if(Courant < 0.1 || TensSor < 2)Etat = 0;**.

Septième Test :

Il consiste à régler le alpha quand le chargeur est à l'état d'attente, c'est-à-dire que le alpha est égal à **0** pour ne pas fournir de tension de sortie : **if(Etat == 0){al = 0;OCR1BL = 0;}**.

9. La fonction Securite

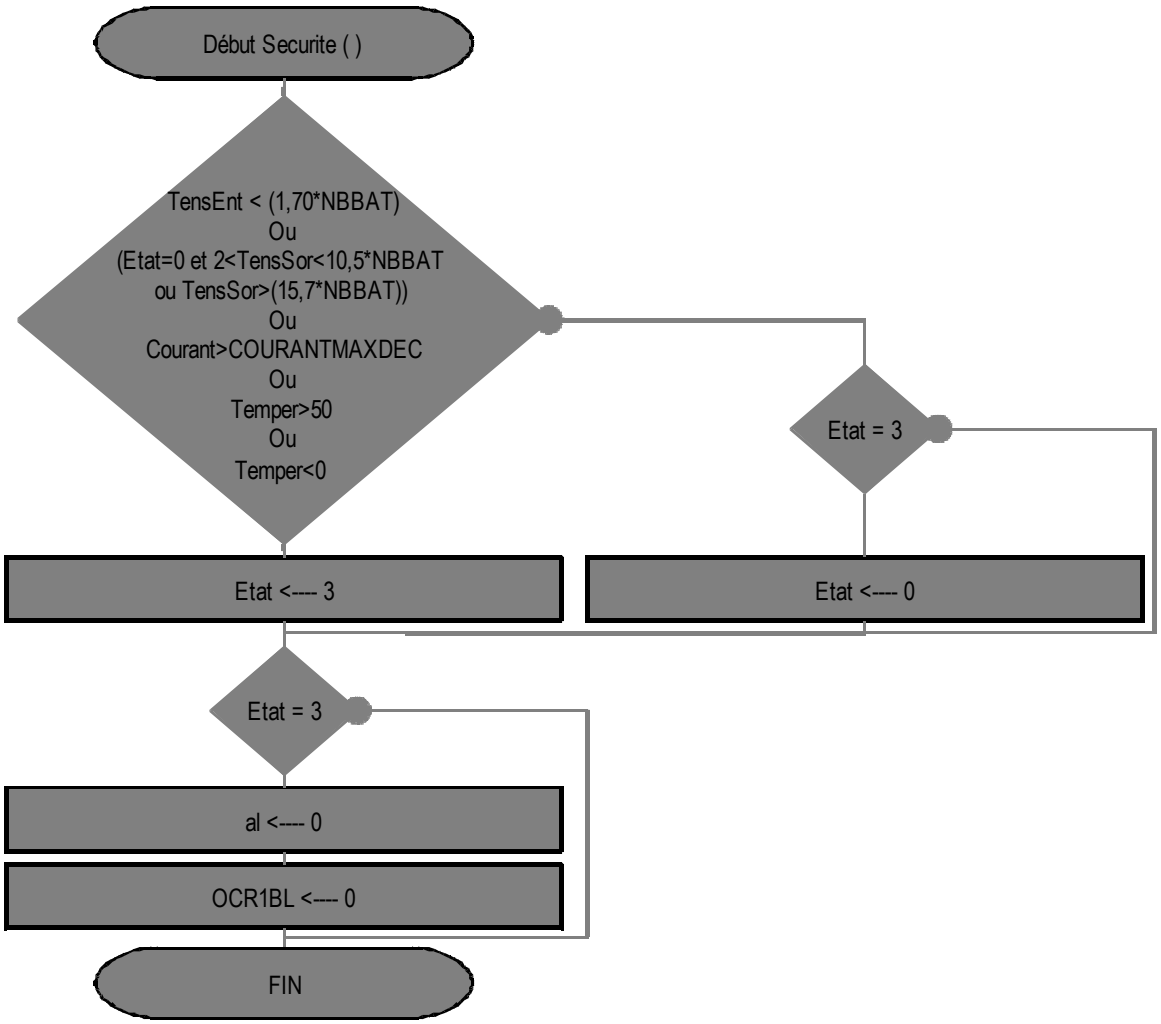


Figure 35 : Ordinogramme de la fonction Securite

```
void Securite(void)
{

  if( TensEnt < (17.0 * NBBAT) || (Etat == 0 &&((TensSor < (10.5 * NBBAT) && TensSor > 2) || TensSor > (15.7 * NBBAT
))) || Courant > COURANTMAXDEC || Temper > 50 || Temper < 0 )
                                                    // Si la tension des batterie n'est pas comprise entre
les tension max par batterie et la tesion min et on non superieur à 2V
      Etat = 3;                                     // Si la tension d'entrée est inferieur à la tesion
max de charge

                                                    // Si le courant est superieur au courant detectable
                                                    // Si la Temperature n'est pas comprise entre 0° et 50
°C
  else
   {
    if(Etat == 3)                                   // Erreur !!

          Etat = 0;
   }
    if(Etat == 3)
        {
          al = 0;
          OCR1BL = 0;

        }
}
```

Figure 36 : Code de la fonction Securite

Cette fonction **Securite** permet de mettre le chargeur en mode **Défaut** ou bien de sortir du mode **Défaut**.

Les conditions de mise en défaut sont soit la tension d'entrée **TensEnt** est inférieure à 17V par batterie(**TensEnt < 17*NBBAT)**, car on ne pourrait pas charger correctement la batterie ou soit que le courant est supérieur à COURANTMAXDEC (**Courant >** COURANTMAXDEC) ce qui impliquerait un défaut de la batterie, ou soit que la température soit inférieure à **0°C** ou supérieure à **50°C** (**Temper > 50 || Temper < 0**), ou bien que la tension de la batterie ne soit pas comprise entre **10.5V par batterie** et **15.7V par batterie** en étant dans l'état d'attente(**Etat == 0 &&((TensSor < (10.5*NBBAT) && TensSor > 2) || TensSor > (15.7*NBBAT))**) : **if(TensEnt < (17.0*NBBAT) || (Etat == 0 &&((TensSor < (10.5*NBBAT) && TensSor > 2) || TensSor >(15.7*NBBAT))) || Courant > COURANTMAXDEC || Temper > 50 || Temper < 0)**.

Si ces conditions ne sont pas remplies et que le chargeur est dans le mode **Défaut** (**else if(Etat==3)**) alors on repasse dans le mode **attente** du chargeur (**Etat=0 ;**).

Enfin on décrit ce qui se passe quand on est dans le mode **Défaut** (**if(Etat==3)**) : on met à 0 le alpha **al= 0;OCR1BL = 0;** ce qui implique une tension de sortie nulle.

Tout le reste du programme correspond à du code créé par le programme que l'on a utilisé c'est-à-dire **l'initialisation des variables** et la fonction **read_adc()**.

3. Caractéristiques électriques de la partie puissance

Nous allons dans cette partie expliquer les modifications que nous avons apportées au projet de hacheur Buck du groupe Kamel Agharbi et Vivien Jamet 2005/2007 en théorie puis en pratique.

1. Théoriques

1. Schéma

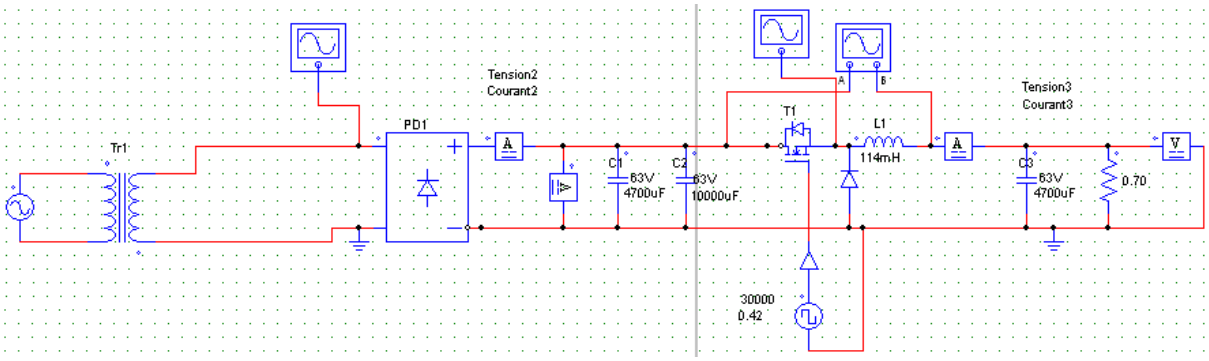


Figure 37 : Schéma partie puissance

2. Explications

Le transformateur permet d'obtenir à partir du secteur **230V 50Hz** une tension de **30V efficace** pouvant fournir un courant de **20A**. Le pont redresseur quant à lui permet d'obtenir une tension redressée de **42V** environ.

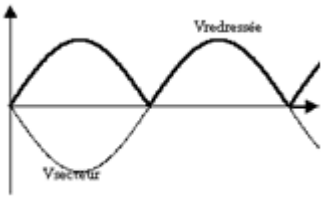


Figure 38 : Courbes tension redressée et tension du secteur

Les condensateurs C1 et C2 du schéma permettent de lisser et de filtrer la tension redressée pour qu'elle puisse ensuite attaquer le hacheur Buck.

3. Etude théorique du hacheur Buck

Nous allons d'abord voir le principe de fonctionnement du hacheur type Buck. Cette étude est tirée du rapport du groupe Kamel Agharbi et Vivien Jamet 2005/2007.

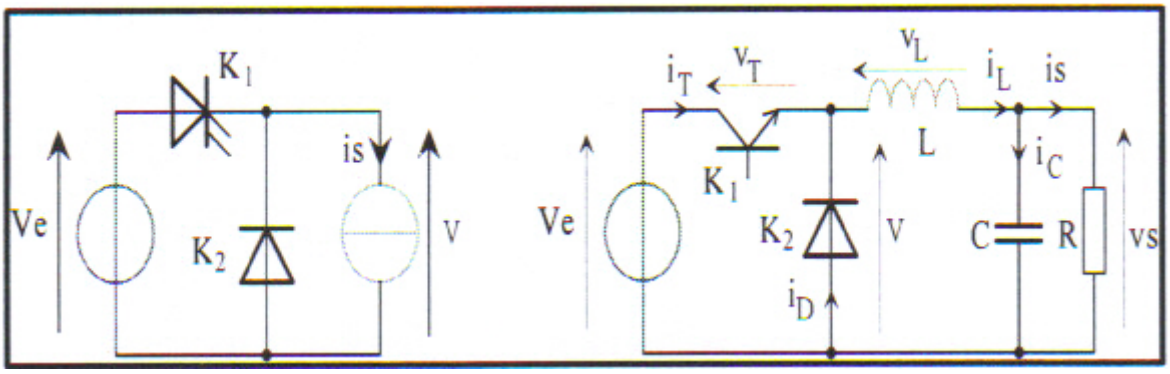
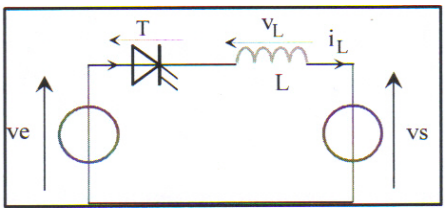


Figure 39 : Schéma fonctionnel du hacheur Buck

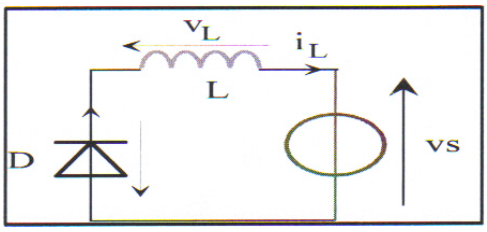
Pour t Є[0;αT], le transistor K1 est fermé:



L'inductance se charge sous Ve-Vs avec Ve>Vs.

Figure 40 : Schéma Hacheur Buck en TON

Pour t Є[αT;T], le transistor K1 est ouvert:



L'inductance se décharge sous –Vs avec –Vs<0.

Figure 41 : Schéma hacheur TOFF

Formes d'ondes des principaux courants et tensions :

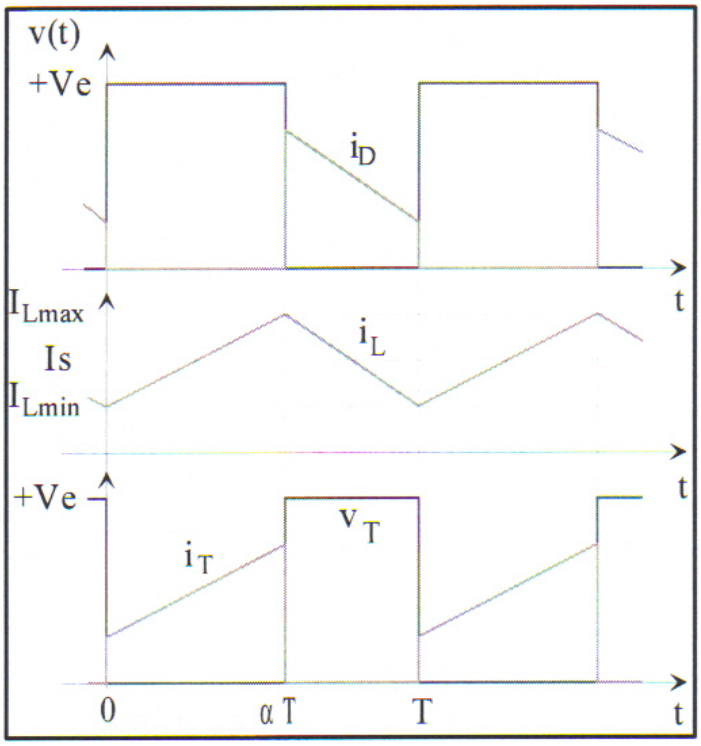


Figure 42 : Courbes de tensions et courants du hacheur Buck

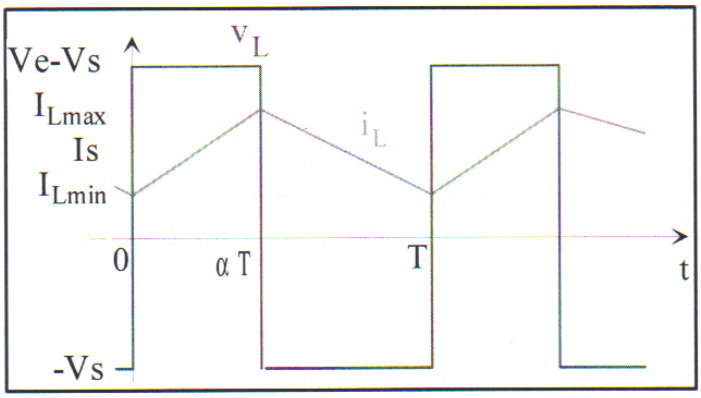


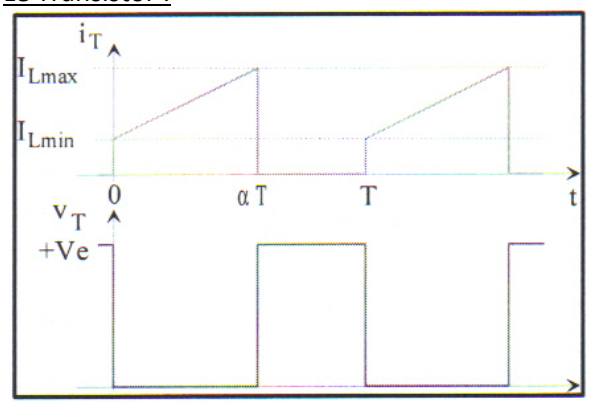
Figure 43 : Tension et courant de l'inductance L

$$VL = \frac{1}{T}\int_T VL(t).dt = \frac{1}{T}[(Ve - Vs) * \alpha T + (-Vs) * (T - \alpha T)] = \alpha Ve + \alpha Vs - Vs(1 - \alpha)$$

Donc en régime permanent : Vs=α*Ve

Contraintes sur les interrupteurs :

Le Transistor :



$$I_{T\max} = I_{TM} = I_{L\max} = Is + \alpha \cdot (1-\alpha) \cdot \frac{Ve}{2\,LF}$$

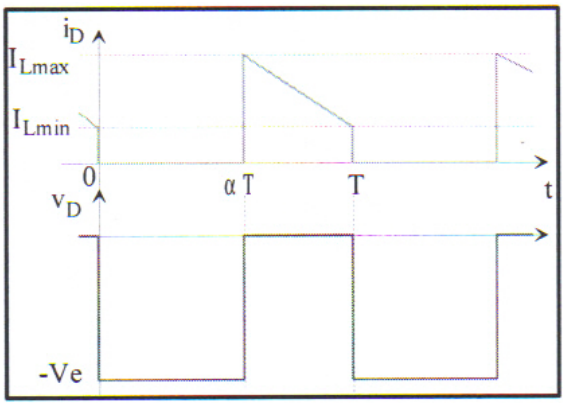
$$I_{Tmoy} = I_{T(AV)} = \alpha \cdot Is = Ie_{moy}!$$

$$I_{Teff} = I_{T(RMS)} = \sqrt{\left(Is^2 + \frac{\Delta I_L{}^2}{12}\right) \cdot \alpha}$$

$$V_{T\max} = V_{TM} = +Ve$$

Figure 44 : Contraintes sur le transistor

La Diode :



$$I_{D\max} = I_{FRM} = (i_L) + \frac{\Delta I_L}{2}$$

$$\dot{c}\, Is + \alpha \cdot (1-\alpha) \cdot \frac{Ve}{2\,LF}$$

$$I_{Dmoy} = I_{F(AV)} = (1-\alpha) \cdot Is$$

$$I_{Deff} = I_{F(RMS)} = \sqrt{\left(Is^2 + \frac{\Delta I_L{}^2}{12}\right) \cdot (1-\alpha)}$$

$$V_{D\ inv\max} = V_{DRM} = +Ve$$

Figure 45 : Contraintes sur la diode

4. Dimensionnement des composants du hacheur Buck

Notre souci est de dimensionner l'inductance, le transistor, la diode et le condensateur. Nous allons donc voir maintenant comment les dimensionner.

La fréquence de découpage sera de 30KHz, les composants seront de hautes fréquences.

Conditions :
F=30KHZ
Ve=$30 * \sqrt{2} = 42.43 Volts$
$P = 600\ Watts$

Détermination de la valeur de l'inductance L

On détermine le courant maximal quand la tension que l'on applique aux batteries est minimale soit 20V.
$I = 20 Ampères$
$$\alpha = \frac{Vs}{Ve} = \frac{10}{42.43} = 0.24$$

On décide d'avoir une variation de courant dans l'inductance la plus petite possible pour un coût relativement peu élevé de l'inductance. On choisit donc 10% de la valeur du courant soit 2 Ampères de variation.

$\Delta I = 2\ Ampères$
$$\Delta I = \frac{Ve * \alpha * (1 - \alpha)}{L * F}$$

Ce qui implique :
$$L = \frac{Ve * \alpha * (1 - \alpha)}{\Delta I * F} = \frac{42.43 * 0.24 * (1 - 0.24)}{2 * 30000} = 129 \mu Henri$$

Détermination du condensateur C

On détermine la valeur du condensateur en fonction de la variation de tension de sortie que l'on souhaite. On se base sur la tension de sortie la plus haute atteinte c'est-à-dire 32V.

$$\alpha = \frac{Vs}{Ve} = \frac{16}{42.43} = 0.38$$

$$\Delta Vs = 10\% Vs = 0.1 * 16 = 1.6V$$

$$\Delta Vs = \frac{Ve * \alpha * (1 - \alpha)}{8 * L * C * F^2}$$

Ce qui implique :

$$C = \frac{Ve * \alpha * (1 - \alpha)}{8 * L * \Delta Vs * F^2} = \frac{42.43 * 0.38 * (1 - 0.38)}{8 * 129 * 10^{-6} * 1.6 * 30000^2} = 6{,}7 \mu Farads$$

Détermination des caractéristiques du transistor et de la diode du hacheur Buck

Pour le transistor du hacheur Buck nous utiliserons un **APT20M22JVRU3** (Transistor + Diode). Celui-ci supporte un courant direct de **97A** et une tension de **200V**. Le courant qui le traversera vaudra **20A** au maximum et une tension maximale de **42.43V**. La diode devra supporter un courant de 20A max et une tension max inverse de 42V. La diode est contenue dans le **APT20M22JVRU3**, la diode de ce composant supporte une tension inverse de 200 Volts, et un courant de 96A.

2. Pratiques

1. Dimensionnement des composants

Valeur du condensateur C

Valeur normalisée du condensateur : C=4700µFarads.

Valeur de l'inductance L

Il n'existe pas d'inductance toute faite supportant 20A (au maximum) et ayant pour valeur 129µHenri c'est pourquoi nous l'avons fabriquée par nous même.
Nous choisissons d'utiliser un fil de 6mm² supportant une densité de courant de 5 à 10A/mm². Nous avons utilisé une bobine déjà faite qui avait pour valeur environ 117µH. Nous devons déterminer si la bobine ne se saturera pas avec les tests.

Transistor et de la diode du hacheur Buck

Nous avons utilisé comme prévu un **APT20M22JVRU3** (Transistor + Diode).

2. Mise en Boîte finale

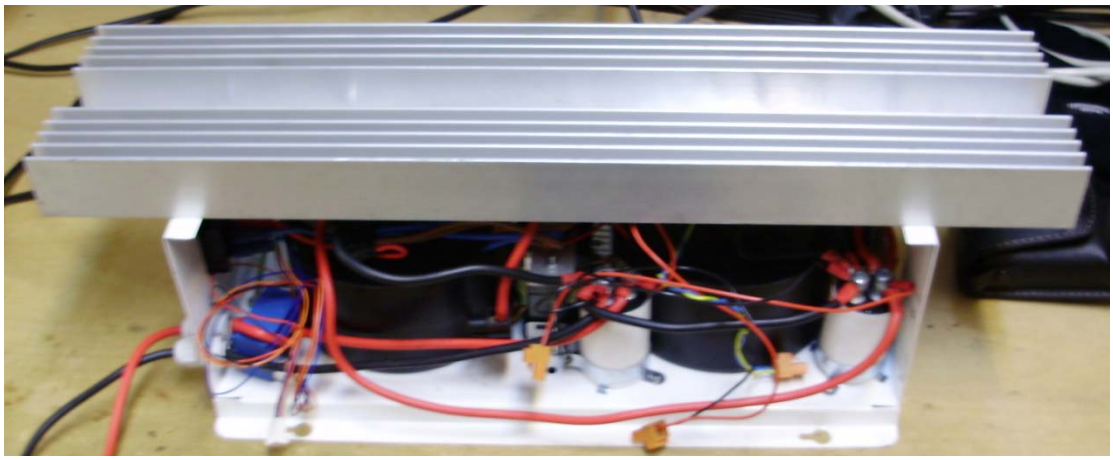


Figure 46 : Photo Partie Puissance

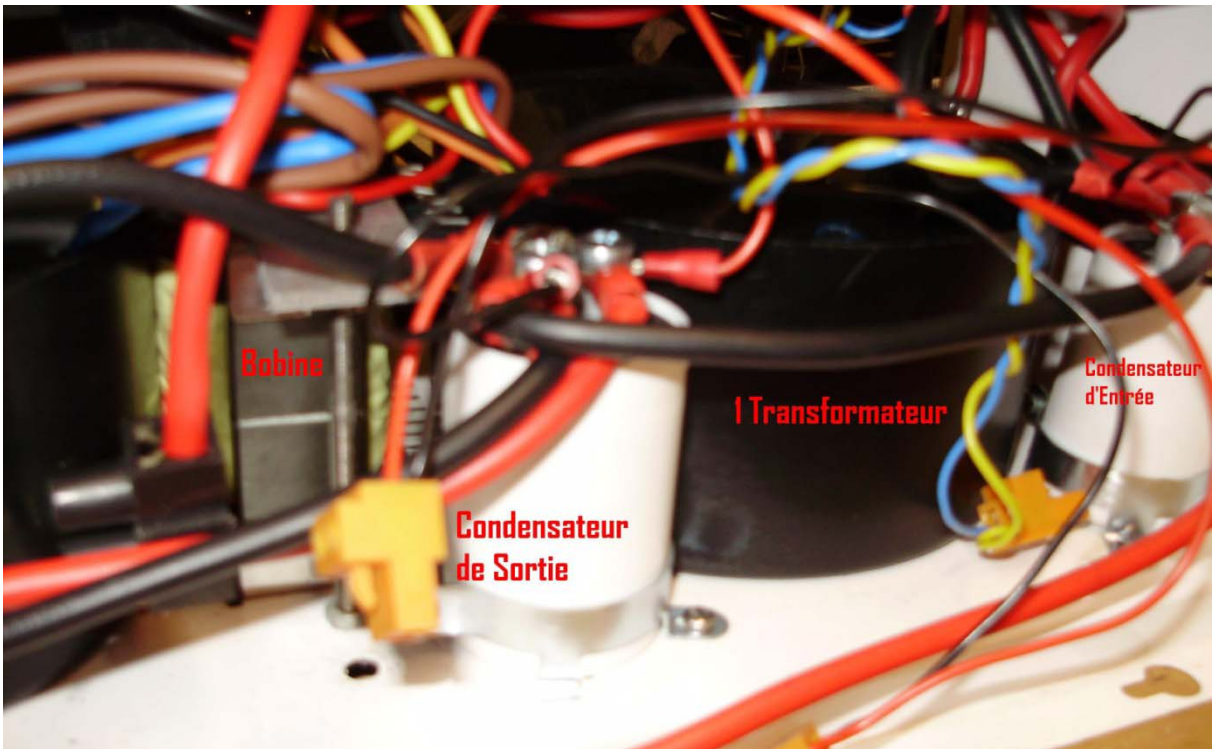


Figure 47 : Photo partie puissance 1/2

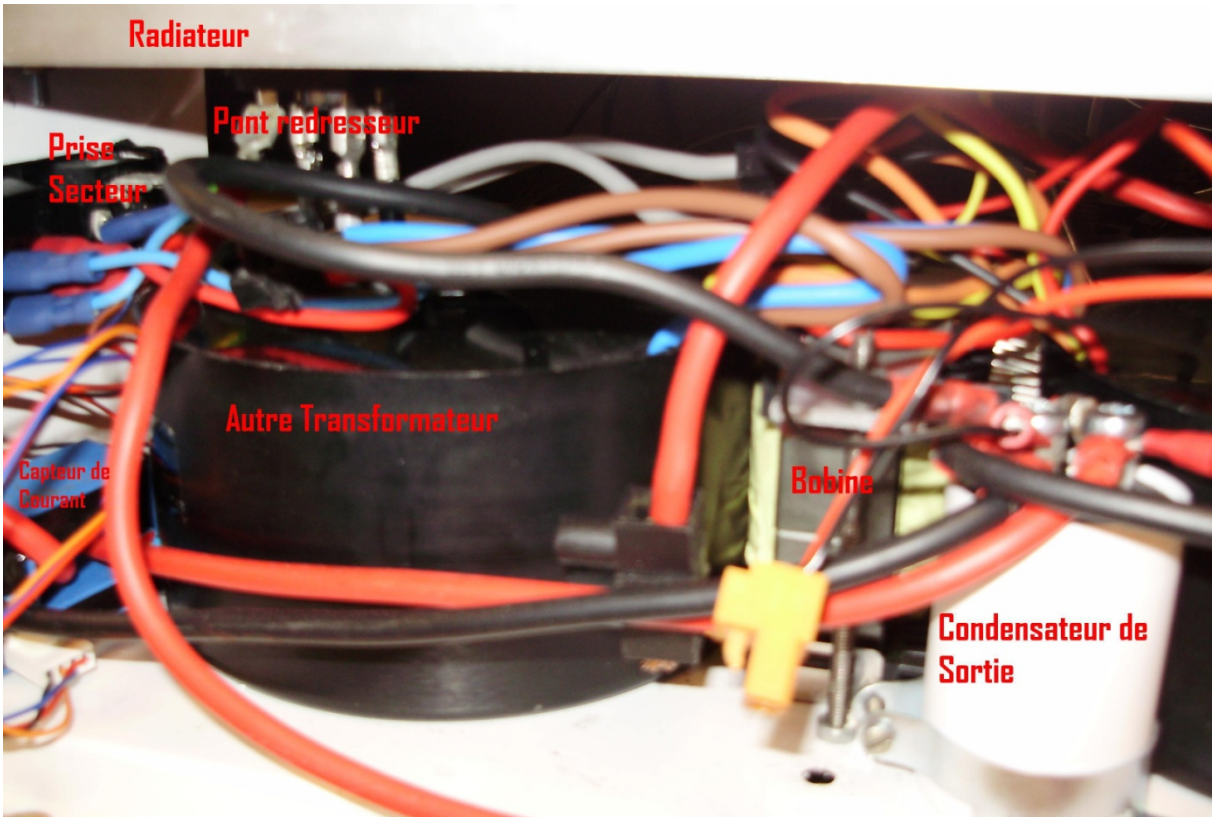


Figure 48 : Photo partie puissance 2/2

3. Tests

Lors des tests nous relevons la tension en sortie des transformateurs, puis la tension et le courant en sortie du pont redresseur, ensuite la tension aux bornes de l'inductance, la tension à l'entrée du hacheur Buck, la tension et le courant en sortie du hacheur Buck.

Premiers tests

Nous avons tout d'abord fait des tests avec la première carte conçue avec une tension d'alimentation réduite et avec le hacheur Buck (réalisé l'an passé) qui limitait le courant et avec des blocs de charge ; les valeurs de tensions et de courant du programme de l'ATMEGA ont été adaptées pour réaliser ces tests. Nous avons pu nous apercevoir que le signal MLI généré par l'ATMEGA ne peut pas piloter le transistor directement. Il fallait passer par le driver. La carte réalisée était trop grande pour pouvoir la caser dans la boîte imposée par le professeur. Pour la carte suivante il a fallu rajouter un composant (le driver) et réduire la taille de la carte. Les alimentations fournissaient les tensions souhaitées (±15V, +5V, 0V), avec allumage des DELs. Le capteur de température fonctionnait. Cependant l'affichage des tensions sur LCD, du courant, et de la température ne se faisait pas correctement ce qui nous amené à modifier le programme de l'ATMEGA. Le transfert du programme du PC à l'ATMEGA fonctionnait lui aussi. Apres modification du programme, tous les tests se sont révélés concluants, c'est pourquoi nous sommes passés à l'étape suivante : charger une batterie. Ce test s'est aussi révélé concluant.

Nous sommes passés à l'étape suivante c'est-à-dire la réalisation d'un hacheur de plus grosse taille.

Seconds tests

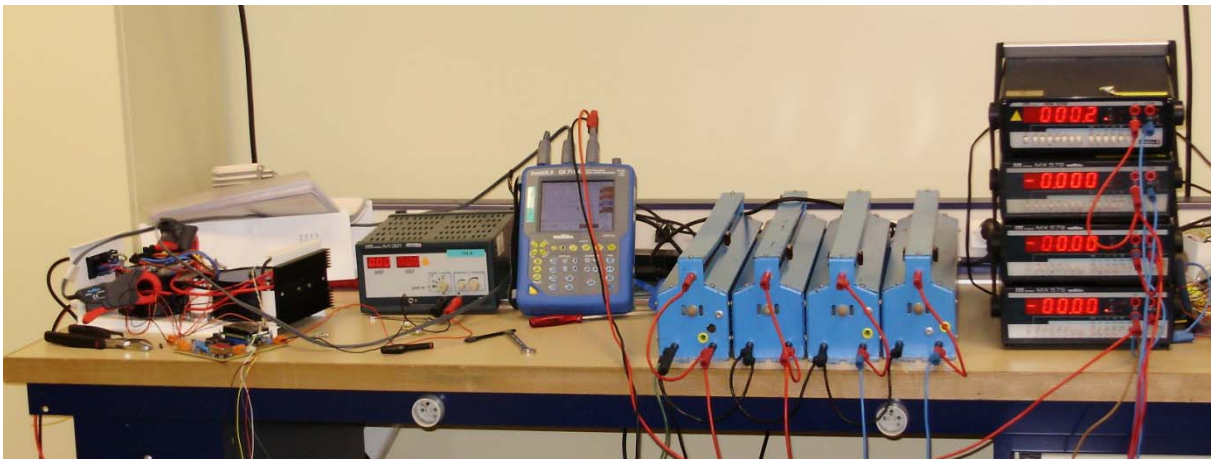


Figure 49 : Photo prise lors d'une des séances de test

Nous avons réalisé ces tests avec la nouvelle carte, avec la bonne alimentation et toujours avec les blocs de charge. Nous avons tout d'abord observé une surchauffe très importante du transistor (du hacheur Buck). Nous avons donc vérifié les composants mis en place. La diode D14 a été remplacée ainsi que la valeur de la résistance R14. A la suite de ces changements, la surchauffe se passait sur le pont de diode du hacheur Buck du au pic de courant et donc du au filtrage de la tension d'entrée. Nous avons décidé de mettre en parallèle deux ponts de diode, mais la surchauffe était toujours aussi importante. De plus à cause du mauvais filtrage d'entrée nous avons constaté que le courant d'entrée était trop important pour les transformateurs. De plus à cause de ce filtrage, la tension

moyenne de sortie n'était pas suffisante pour une charge complète des transformateurs. Par conséquent, nous avons décidé de changer les caractéristiques du chargeur, la tension de sortie de chargeur sera comprise entre 10 et 16V pour obtenir un courant plus important (estimé à 20A).

Troisièmes Tests

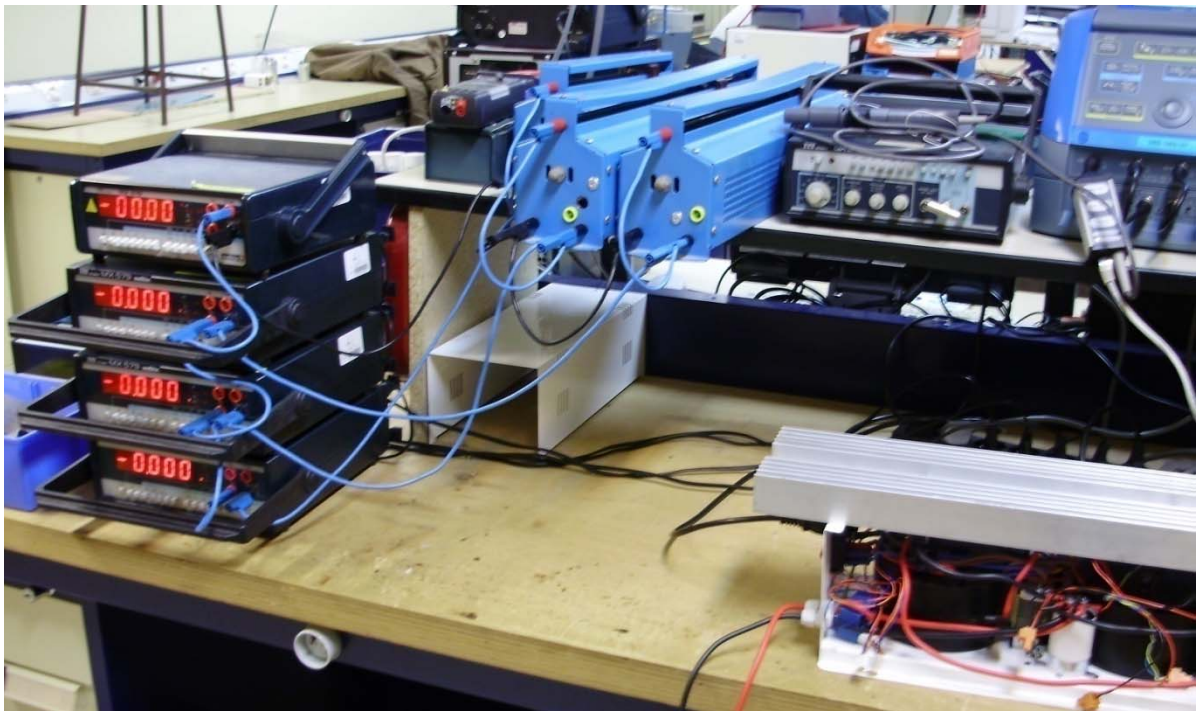


Figure 50 : Dernière séance de tests

Après plusieurs changements de composants dus à de multiples plantages du microcontrôleur, nous nous sommes aperçus que cela provenait d'un problème de compatibilité électromagnétique entre le microcontrôleur et le driver. C'est pourquoi nous avons réalisé nos essais en charge grâce à un GBF qui permettait de remplacer le microcontrôleur.

4. Relevés

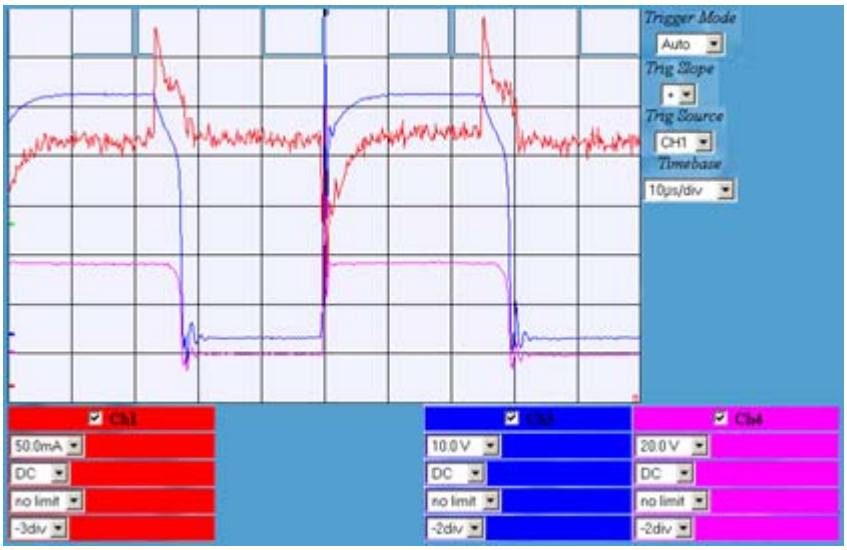


Figure 51 : Ig, Vc et Vt

La courbe rouge est le courant Ig c'est-à-dire le courant de commande du transistor. La grille du transistor peut être considérée comme un condensateur à la mise sous tension c'est pourquoi nous avons des pics de courant.

La courbe bleue représente la tension de commande (Vc) du transistor. Il est en créneau, mais on peut constater quelques déformations. Pour une meilleure commutation une résistance de plus faible valeur serait judicieuse. On passe cette résistance (R13) de 1KΩ à 10Ω.

La courbe violette correspond à la tension de sortie du transistor (Vt). On peut constater les phases de conduction et de non conduction.

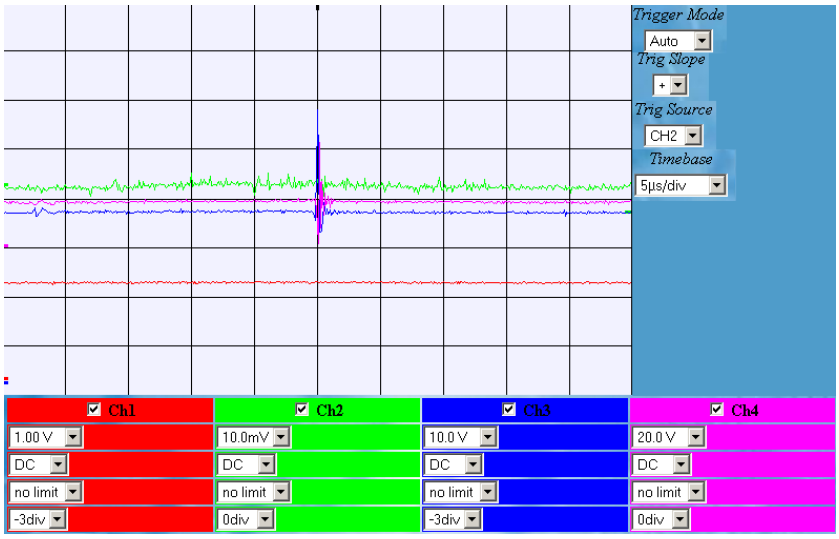


Figure 52 : Is, Vl, Ve et Vs

La courbe rouge correspond au courant multiplié par 10 de sortie du montage.

La courbe verte correspond à la tension aux bornes de l'inductance.

La courbe bleue correspond à la tension d'entrée du hacheur filtrée.

La courbe violette correspond à la tension de sortie du hacheur.

Nous pouvons constater que la toutes les courbes sont droites ce qui signifie que tous nos filtrages sont de très bonne qualité. Il n'y a donc aucune oscillation de courant et de tension.

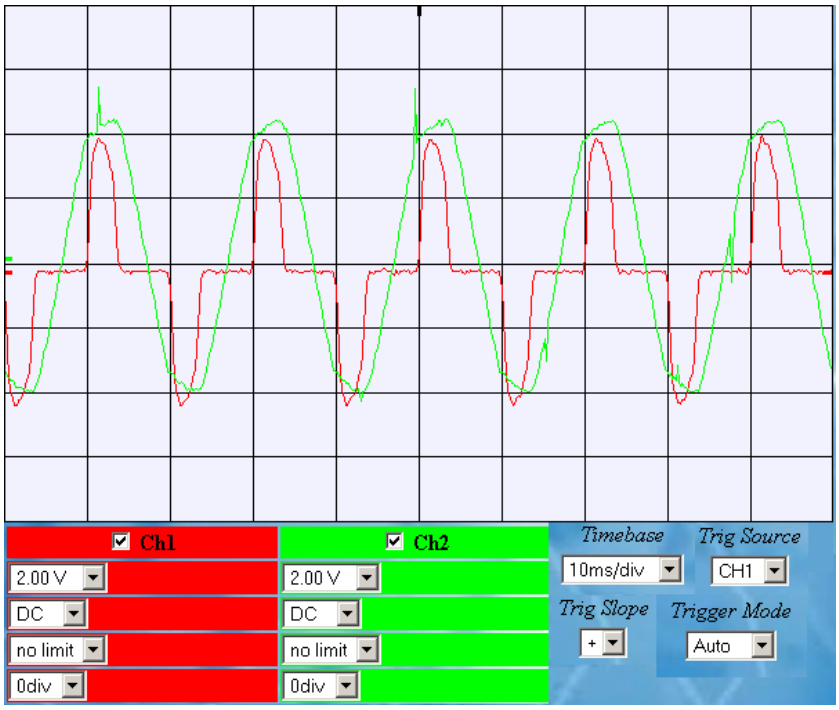


Figure 53 : Courbes de tension et de courant d'entrée du pont redresseur

La courbe rouge correspond au courant d'entrée du pont redresseur Ie multipliée par 10.

La courbe verte correspond à la tension à la sortie du transformateur multipliée par 10.

Cet oscillogramme montre la tension et le courant en sortie de transformateur, c'est-à-dire en entrée du pont redresseur. Le courant est impulsionnel car on réalise un filtrage capacitif. La tension a la même forme que la tension secteur mais a une amplitude moins grande, ici environ 40V.

4. Changements à apporter pour charger tout type de batterie

Pour changer le nombre de batterie à charger en série en même temps, il faut changer :

- R2 et R3 pour avoir la meilleure résolution possible et que la tension ne dépasse pas 5V en sortie du pont diviseur de tension.
- Le capteur de courant pour une autre gamme de courant.
- Dans le programme, MULTITENSENT et MULTITENSSOR. Il faut régler le coefficient de division du diviseur de tension.
- Le coefficient de multiplication du capteur de courant MULTICOURANT.
- Le nombre de batterie NBBAT.
- Le courant maximal lisible COURANTMAXDEC.
- Le courant maximal de charge COURANTMAX.
- Le α maximal ALMAX.

5. Etude financière

Nom	Ref commande	Prix U.H.T.	Prix HT	Fournisseur
10 µF 50V(x5)	315-0808	1,00 €	1,00 €	Radiospares
470µF 6,3V(x5)	449-0845	1,34 €	1,34 €	Radiospares
100nF (x10)	312-1469	1,06 €	2,12 €	Radiospares
22pF(x25)	652-9737	5,84 €	11,68 €	Radiospares
22uF 25V(x5)	449-1012	1,03 €	1,03 €	Radiospares
470µF 25V(x5)	315-0574	2,69 €	2,69 €	Radiospares
1000µF 100V(x5)	315-1038	10,21 €	10,21 €	Radiospares
220uF 25V(x5)	3150-552	1,54 €	1,54 €	Radiospares
Led 5mm 2 mA Rouge (x5)	590-547	0,98 €	0,98 €	Radiospares
Led 5mm 2 mA jaune(x5)	588-370	0,81 €	0,81 €	Radiospares
Led 5mm 20 mA Orange(x5)	625-7892	1,03 €	1,03 €	Radiospares
Led 5mm 2 mA Verte(x5)	826-436	0,75 €	0,75 €	Radiospares
1n40148 (x100)	436-7357	2,13 €	2,13 €	Radiospares
1N4007	649-1143	9,19 €	9,19 €	Radiospares
11DQ06 (x5)	254-0702	2,12 €	2,12 €	Radiospares
ATMEGA 8535	628-3217	5,96 €	5,96 €	Radiospares
IR2183	543-0642	2,70 €	2,70 €	Radiospares
Embase horizontale CI à extrémités(x5)	403-998	2,07 €	2,07 €	Radiospares
Bornier à vis à monter sur câble (x5)	403-875	4,59 €	4,59 €	Radiospares
Connecteur 5 Broches (x5)	173-2950	4,59 €	4,59 €	Radiospares
Connecteur 4 Broche(x5)	173-2944	4,18 €	4,18 €	Radiospares
Connecteur femelle 4 pates	467-611	2,83 €	5,66 €	Radiospares
Connecteur femelle 5 pates	467-627	2,87 €	5,74 €	Radiospares
Afficheur femelle (x5)	267-7387	4,15 €	4,15 €	Radiospares
Connecteur male 16 pate (x5)	547-3239	3,79 €	3,79 €	Radiospares
Connecteur HE10	461-742	1,19 €	1,19 €	Radiospares
Afficheur 4x16	532-6795	16,13 €	16,13 €	Radiospares
10µH (x5)	484-6399	4,99 €	4,99 €	Radiospares
330uH (x10)	432-4423	13,44 €	13,44 €	Radiospares
LM2574N-5.0 (régulateur de tension)	534-8013	2,25 €	2,25 €	Radiospares
			5,00 €	Radiospares
Potentiomètre multi-tours 1K				
Potentiomètre 10K				
1K				
4,7K				
4,7K 1/2W				
6,8K				
1,5K				
10K				
16Mhz(x5)	547-6070	1,97 €	1,97 €	Radiospares
Support CI 8 patte (x5)	447-308	7,36 €	7,36 €	Radiospares

Support CI 40 patte(x5)	447-386	20,80 €	20,80 €	Radiospares
Capteur de courant HAS 50		25,00 €	25,00 €	
TEN 5-4823 (Traco Power)	293-2915	45,56 €	45,56 €	Radiospares
lm75CIM-5/NOPB (x5)	536-1669	9,83 €	9,83 €	Radiospares
220 ohms (x50) CMS	371-4904	1,85 €	1,85 €	Radiospares
20nF (x10) CMS	414-7125	10,15 €	10,15 €	Radiospares
120x130	159-6085	12,29 €	12,29 €	Radiospares
Entretoise pour écran (x100)	265-2344	8,33 €	8,33 €	Radiospares
radiateur 1000x290x115	409-9997	26,35 €	26,35 €	Radiospares
Transformateur 300VA 2*30V 10A	223-8869	38,11 €	76,22 €	Radiospares
APT20M22JVRU3 (Transistor + Diode)		16,87 €	16,87 €	Advanced Power Technology
Domino alim 16mm²	200-8630	4,64 €	4,64 €	Radiospares
écrou pour presse-étoupe (x10)	531-191	1,43 €	1,43 €	Radiospares
presse-étoupe (x5)	530-895	3,04 €	3,04 €	Radiospares
prise informatique + BP + porte fusible	211-1001	5,89 €	5,89 €	Radiospares
GBPC2502A (Pont Diode)	395-4203	3,07 €	3,07 €	Radiospares
Fusible 4A retarder(x10)	480-4339	1,55 €	1,55 €	Radiospares
rouleau 25m	516-7934	24,52 €	24,52 €	Radiospares
rouleau 25m	516-7906	24,52 €	24,52 €	Radiospares
condensateur 4700µF 63V	189-765	10,85 €	21,70 €	Radiospares
condensateur 10000µF 63V	228-0488	15,13 €	15,13 €	Radiospares
support condensateur (x5)	543-383	1,08 €	1,08 €	Radiospares
support bobine	647-9525	2,79 €	2,79 €	Radiospares
clips (x10)	647-9531	12,74 €	12,74 €	Radiospares
pot de ferrite	647-9519	10,87 €	10,87 €	Radiospares
Fil de cuivre 0,5mm² (100 mètres)	204-6890	51,94 €	51,94 €	Radiospares
			20,00 €	
Vis M2x25				
Vis M3x30				
Vis M4x10				
Vis M4x15				
Vis M4x25				
Vis M5x10				
Vis M5x15				
Ecrou M2,5				
Ecrou M4				
Ecrou M6				
Tige filetée M4				
Tige filetée M6				

	243-1046	108,39 €	108,39 €	Radiospares
cosse fasten				
cosse à œillette M5 6²				
cosse à œillette M5 2,5²				
cosse à œillette M5 0,75²				
		prix total	708,59 €	

Tableau 3 : Liste des composants et prix

Pour réaliser ce projet nous avons utilisé tous les composants présents sur cette liste. Nous avons établit les prix des composants à partir de catalogues, la plupart des composants proviennent de Radiospares. Pour la plupart des composants nous les avons pris dans le stock de l'IUT. Nous pouvons nous apercevoir que le projet coûte relativement cher c'est-à-dire **708,59€**.

Conclusion

Durant la conception et la fabrication de ce projet, nous avons pu tester chacune des parties, qui constituent le montage, et nous apercevoir qu'elles fonctionnaient toutes indépendamment des autres. Cependant quand nous avons assemblé toutes les parties, de nombreux problèmes sont survenus ce qui nous a pris beaucoup de temps et surtout ce qui nous a mené à changer les caractéristiques du chargeur : nous sommes passés d'un chargeur 24V 30A à un chargeur 12V 20A. Le problème majeur au final est un problème de compatibilité électromagnétique qui fait planter le microcontrôleur ATMEGA 8535. Les études précédentes nous ont pris beaucoup de temps. La conception et la réalisation de ce projet n'est pas faisable dans le temps imparti réservé au projet. Nous n'avons donc pas pu corriger ce dernier problème de compatibilité électromagnétique.

Tableaux

Tableau 1 : Planning de prévision 5
Tableau 2 : Planning réel 6
Tableau 3 : Liste des composants et prix 39

Bibliographie

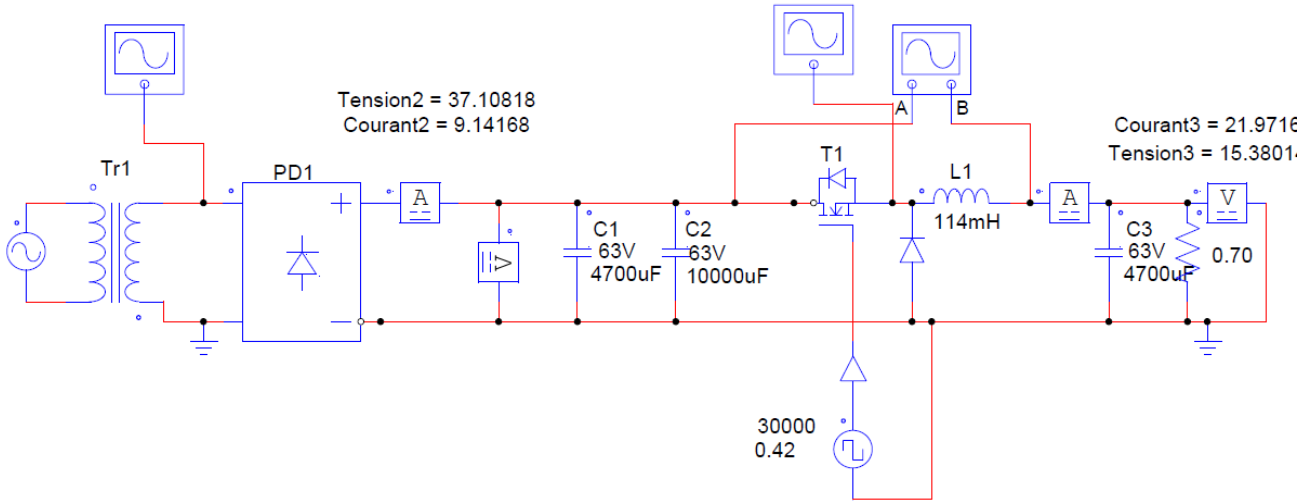
Toutes les images sont de notre conception. Tous les autres documents utilisés proviennent du site www.thierry-lequeu.fr.

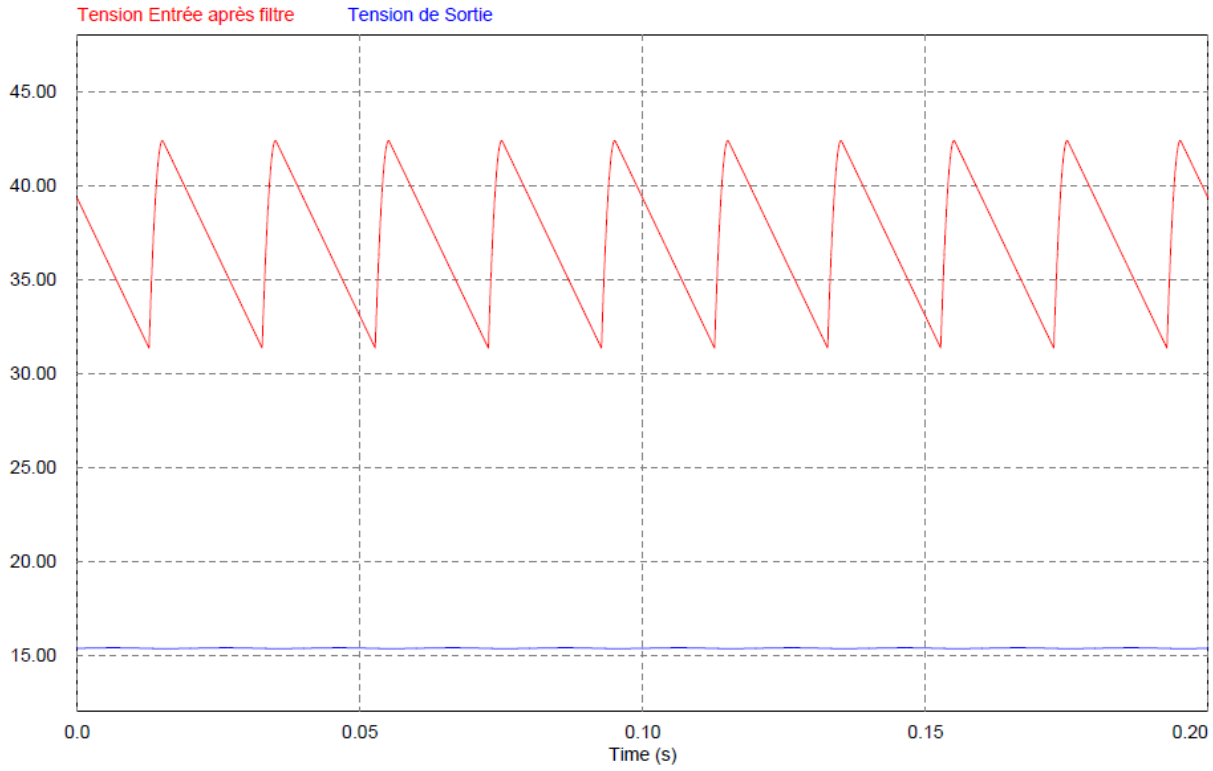
Illustrations

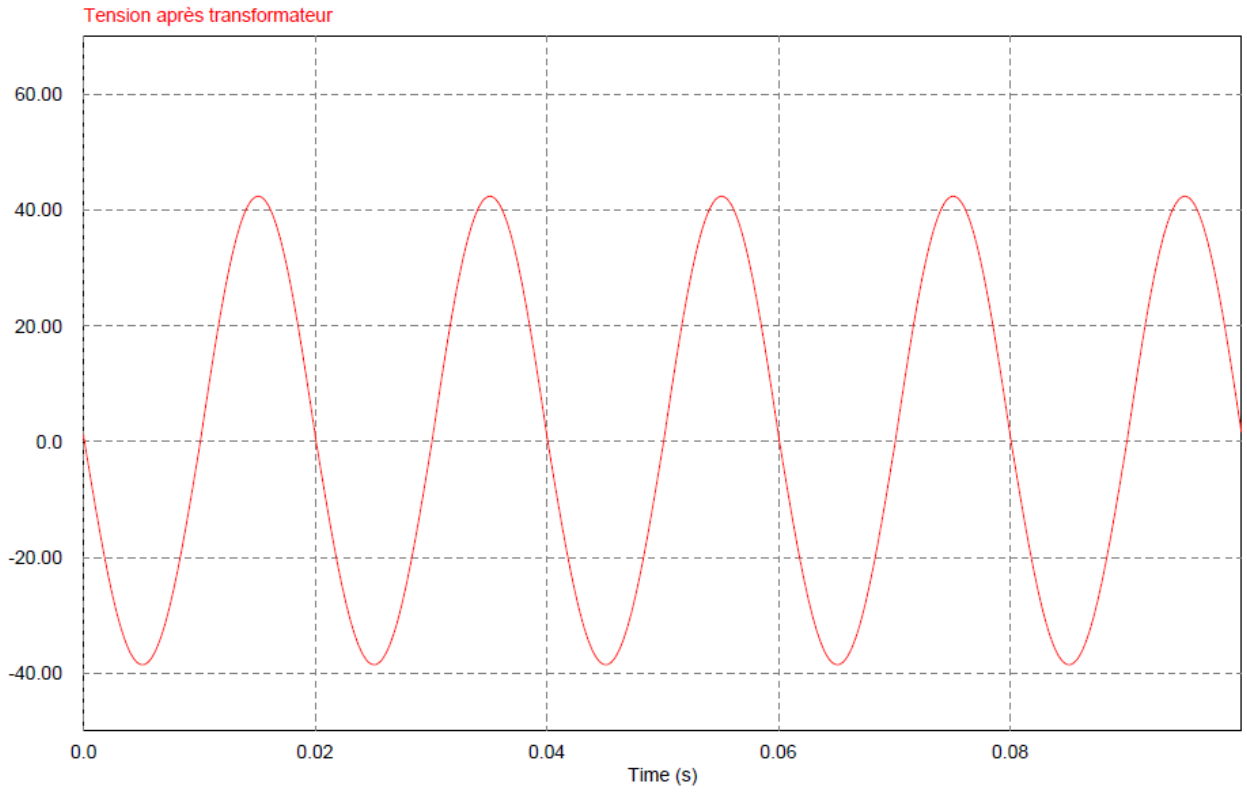
Figure 1 : Synoptique niveau 1 4
Figure 2 : Synoptique niveau 2 4
Figure 3 : Batterie plomb OPTIMA 12V 48AH 7
Figure 4 : Courbes tension, courant de charge 7
Figure 5 : Alimentation 5V 8
Figure 6 : Alimentation ±15V 8
Figure 7 : Capteur de tension d'entrée d'hacheur Buck 9
Figure 8 : Capteur de tension de la batterie 9
Figure 9 : Capteur de courant 10
Figure 10 : Driver pilotant le transistor 10
Figure 11 : DEL ±15V 10
Figure 12 : ATMEGA, LCD, Capteur de température, Connecteur programmation 11
Figure 13 : Capteur de température 11
Figure 14 : LCD 11
Figure 16 : ATMEGA et son connecteur de programmation 12
Figure 15 : Entrées CANs 12
Figure 17 : Définition de variables 13
Figure 18 : Déclarations fonctions 13
Figure 19 : Déclaration variables globales 14
Figure 20 : Ordinogramme Fonction MAIN 15
Figure 21 : Déclaration variables locales et initialisation fonctions de l'ATMEGA 16
Figure 22 : Code 2eme partie de la fonction MAIN 16
Figure 23 : Code de la fonction ECRIRE 17
Figure 24 : Ordinogramme fonction ECRIRE 18
Figure 26 : Code de la fonction Lecture_Tens 19
Figure 25 : Ordinogramme de la fonction Lecture_Tens 19
Figure 28 : Code de la fonction LectureTemper 20
Figure 27 : Ordinogramme de la fonction LectureTemper 20
Figure 29 : Code de la fonction DetCourantMax 21
Figure 32 : Code de la fonction DetTensMax 21
Figure 30 : Ordinogramme de la fonction DetCourantMax 21
Figure 31 : Ordinogramme de la fonction DetTensMax 21
Figure 33 : Code de la fonction Traitement 22
Figure 34 : Ordinogramme de la fonction Traitement 23
Figure 35 : Ordinogramme de la fonction Securite 25
Figure 36 : Code de la fonction Securite 26
Figure 37 : Schéma partie puissance 27
Figure 38 : Courbes tension redressée et tension du secteur 27
Figure 39 : Schéma fonctionnel du hacheur Buck 27
Figure 42 : Courbes de tensions et courants du hacheur Buck 28
Figure 40 : Schéma Hacheur Buck en TON 28
Figure 41 : Schéma hacheur TOFF 28
Figure 43 : Tension et courant de l'inductance L 29
Figure 44 : Contraintes sur le transistor 29

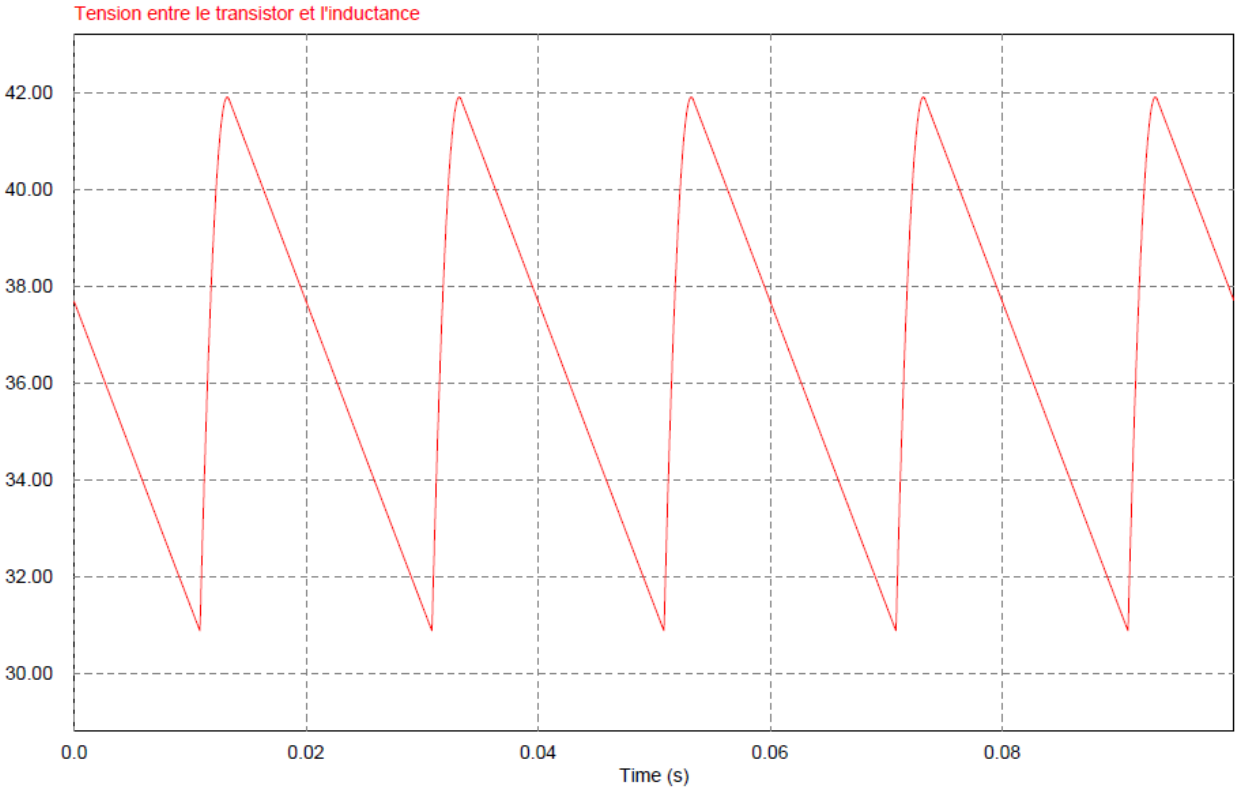
Figure 45 : Contraintes sur la diode 29
Figure 46 : Photo Partie Puissance 31
Figure 47 : Photo partie puissance 1/2 32
Figure 48 : Photo partie puissance 2/2 32
Figure 49 : Photo prise lors d'une des séances de test 33
Figure 50 : Dernière séance de tests 34
Figure 51 : Ig, Vc et Vt 34
Figure 52 : Is, Vl, Ve et Vs 35
Figure 53 : Courbes de tension et de courant d'entrée du pont redresseur 36

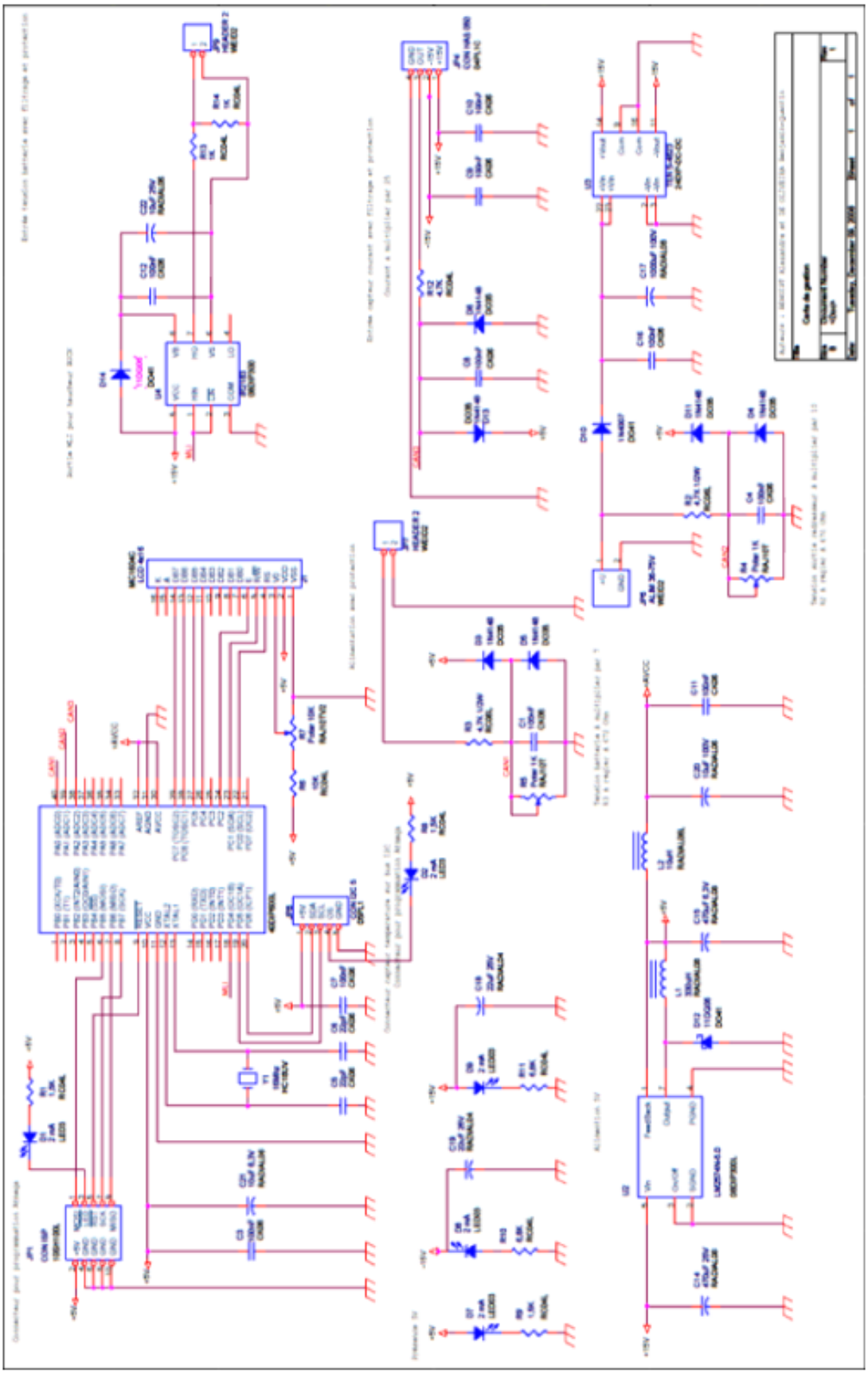
Annexes

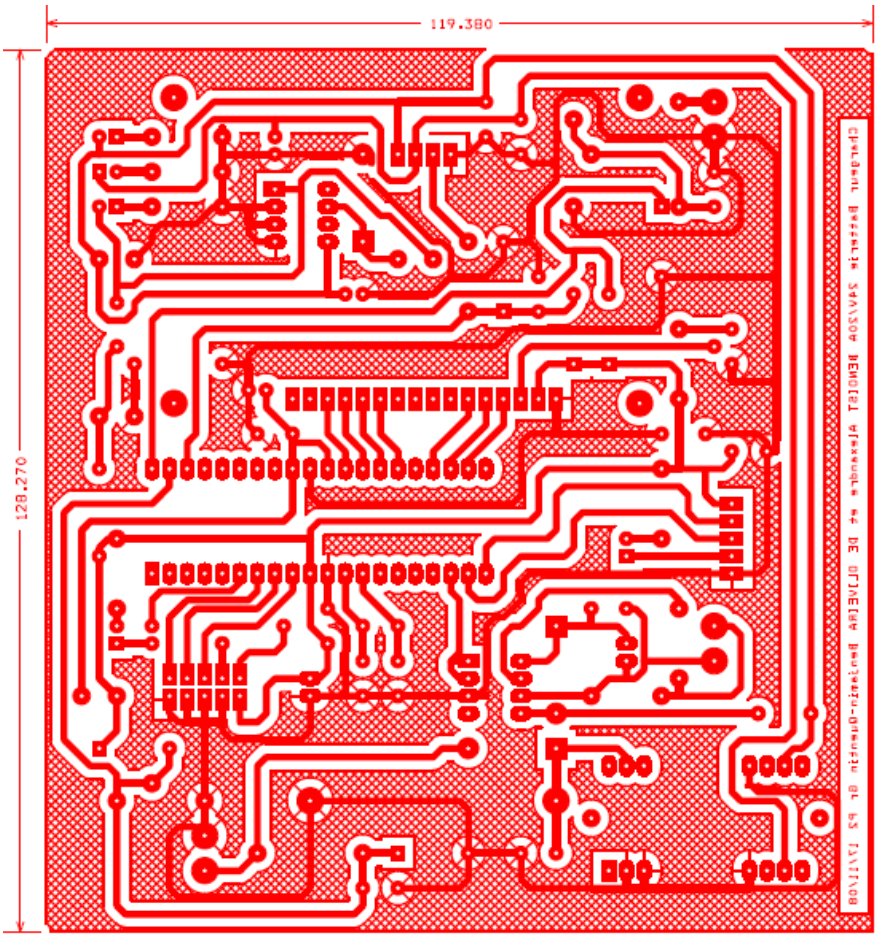

Tension2 = 37.10818
Courant2 = 9.14168
Courant3 = 21.9716
Tension3 = 15.3801
Tr1
PD1
A
V
C1
63V
4700uF
C2
63V
10000uF
T1
L1
114mH
A
B
A
C3
63V
4700uF
V
0.70
30000
0.42


Tension Entrée après filtre
Tension de Sortie
45.00
40.00
35.00
30.00
25.00
20.00
15.00
0.0
0.05
0.10
0.15
0.20
Time (s)


Tension après transformateur
60.00
40.00
20.00
0.0
-20.00
-40.00
0.0
0.02
0.04
0.06
0.08
Time (s)


Tension entre le transistor et l'inductance
42.00
40.00
38.00
36.00
34.00
32.00
30.00
0.0
0.02
0.04
0.06
0.08
Time (s)





Datasheet

http://www.atmel.com/dyn/resources/prod_documents/doc2502.pdf (Atemega8535)
http://www.microsemi.com/datasheets/APT20M22JVRU3-Rev1.pdf (apt20M22JVRU3)
http://www.irf.com/product-info/datasheets/data/ir2183.pdf (IR2183)
http://www.national.com/ds/LM/LM75.pdf (LM75)
http://eshop.engineering.uiowa.edu/NI/pdfs/01/13/DS011394.pdf (LM2574)
http://www.tracopower.com/products/ten5.pdf (Traco Power)
http://www.datasheetcatalog.org/datasheet/irf/gbpc2502a.pdf (GBPC2502A)