5/24/2016

SmartMeter

Rapport Projet S8

ENCADRANTS : XAVIER REDON & ALEXANDRE BOE
INDUSTRIEL : GUILLAUME RENAULT
ELEVE : HAROUN ABDELALI

Informatique Microélectronique Automatique
Système Autonome

POLYTECH LILLE

Université Lille1 Sciences et Technologies

Remerciement

Tout d'abord, je tiens à remercier, toute l'équipe pédagogique de Polytech'Lille et les responsables de la formation Informatique-Microélectronique-Automatique de nous avoir enseigné les bases pour réaliser ce projet de quatrième année dans de bonnes conditions.

Je souhaite également remercier et témoigner toute ma reconnaissance à Monsieur Alexandre BOE et Monsieur Xavier REDON, les responsables du projet, pour leur aide précieuse et leur disponibilité tout au long du projet.

Je remercie Monsieur Guillaume RENAULT pour avoir suivi mon travail tout au long de ce projet et pour s'être rendu disponible pour répondre à toutes nos questions, de manière rapide et pertinente.

Je remercie aussi Monsieur Thierry FLAMEN pour ses réponses et ses précieux conseils sur la conception de la carte électronique.

Enfin, nous remercions particulièrement Monsieur Laurent ENGLES qui a consacré beaucoup de temps dans le tournage et la réalisation de la vidéo de projet.

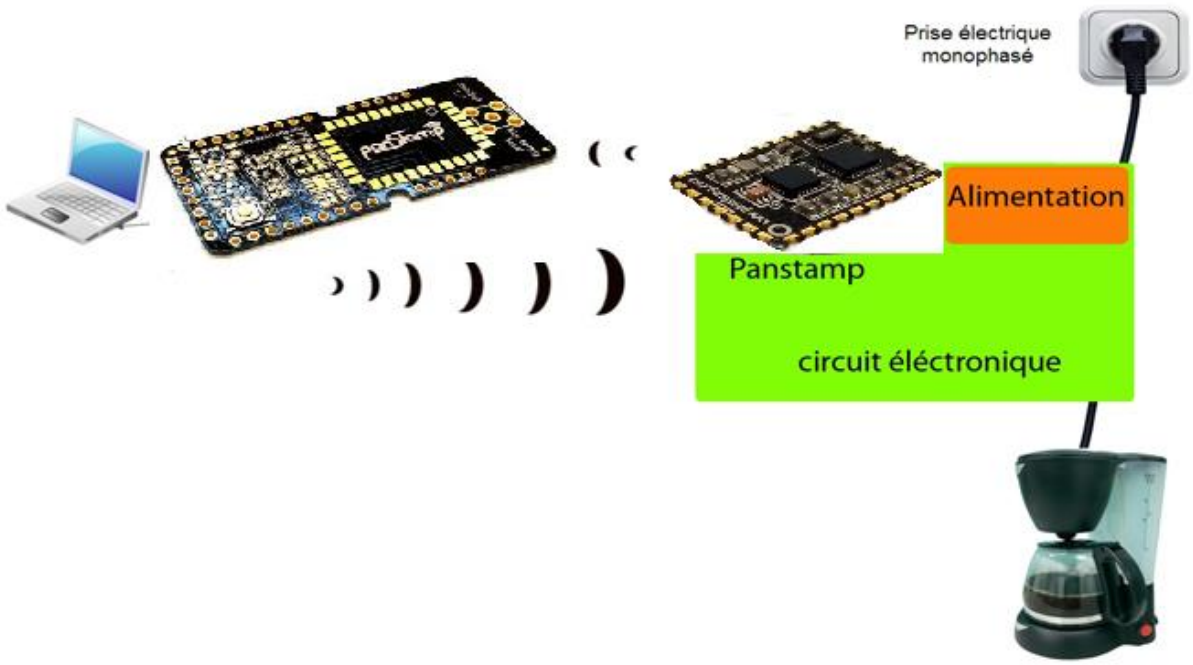
Contents

Introduction

Face au nouvelle contraintes écologique et avec la transition énergétique, il est nécessaire de réduire et d'économiser les énergies consommées, notamment l'énergie électrique.

Un SmartMeter, est un dispositif permettant non seulement de mesurer en temps réel les différents paramètres d'une énergie électrique consommé par une charge donnée, mais aussi de les afficher sur un ordinateur.

L'idée de ce projet était proposée par M. Guillaume RENAULT dans le cadre d'un Projet de Fin d'Etude IMA5 (2014/2015), est revenue cette année avec de nouvelles contraintes.



I. Présentation générale du projet

1. Objectif du projet

En se basant sur le travail qu'a été fait par les anciens étudiants Thomas Edelré & Sylvain Fossaert (PFE IMA5), réaliser un nouvel appareil électrique pour la surveillance de la consommation via un circuit spécialisé dans la mesure de température, de courant et de tension avec envoi des données par microcontrôleur sur un support radio hertzien.

2. Description du projet

Le sujet étant préalablement traité, les améliorations à apporter seront le centre de ce nouveau projet, notamment :

- Détection des "phases" de l'appareil (éteint, en veille ou en marche) et gestion intelligente de son alimentation (arrêt lors des veilles, mise en place de période d'allumage ou d'extinction).

- Amélioration du calibrage de la puce responsable des différentes meures.

- Miniaturisation du circuit.

- Passer de la technologie AVR à la technologie NRG de Panstamp (supporte l'update "over the air", plus des fréquences RF et le cryptage AES entre autres).

- Modélisation 3D d'un boitier adapté puis fabrication via l'imprimante 3D du FABLAB.

3. Une brève présentation de travail réalisé l'année dernière

La carte électronique, ci-dessus, comporte une partie de haute tension (230V) et une de petite tension (3.3V), il est réalisé autour de deux important circuit, le Maxim 78M6610 et le Panstamp AVR 1, le premier est chargé de mesurer (tension, courant, température), et le second, d'émission-réception des données en radio fréquence.

(Pour plus de détails, vous trouverez en annexes le lien du Wiki correspondant)

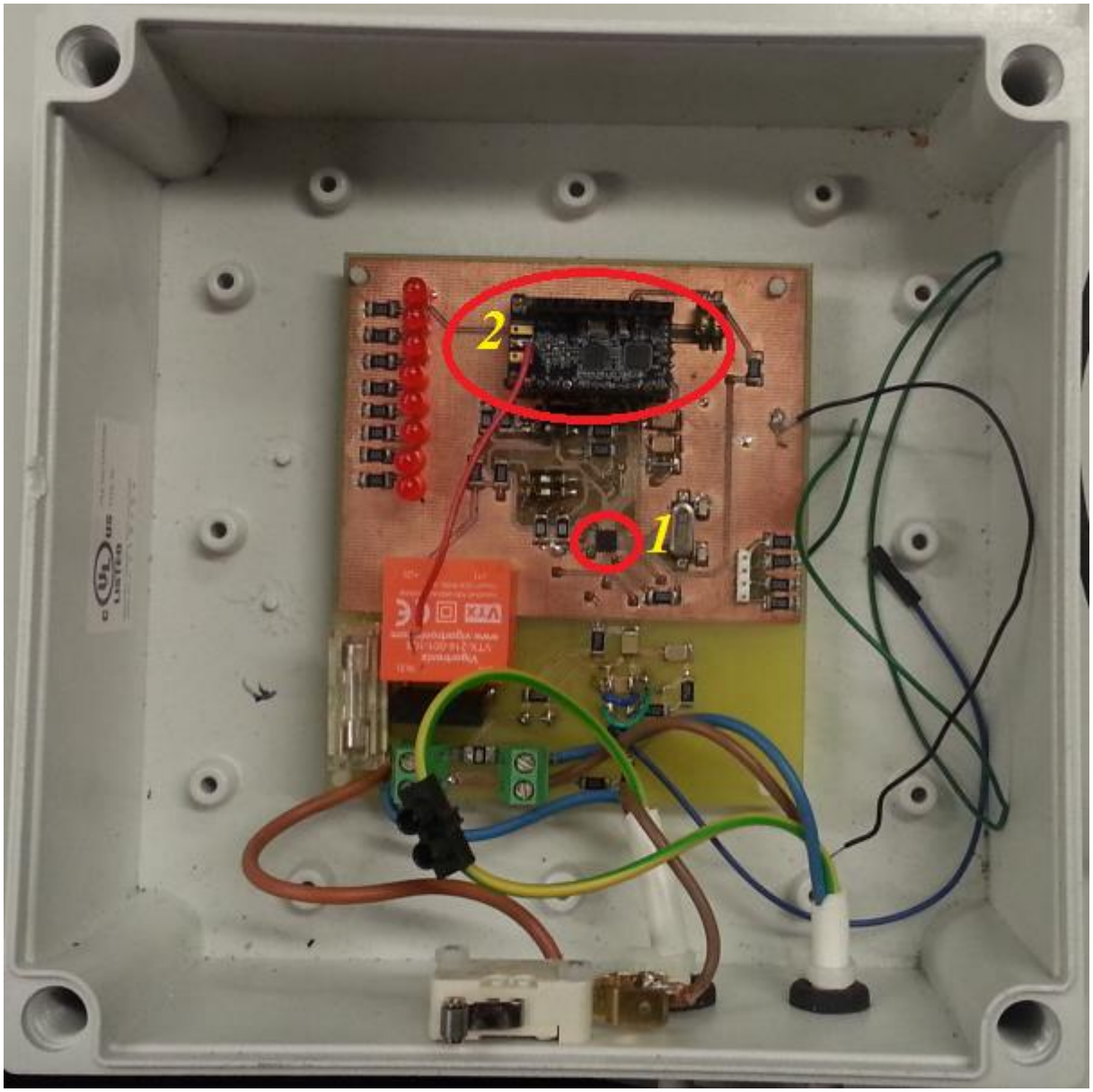


Figure 1 : SmartMeter P25 (14/15)

Dans le but de dimensionner la carte, le binôme avait réalisé une étude électronique qui a abouti à un choix des composants nécessaires, et par la suite, réalisé le circuit électrique en utilisant le logiciel de design EAGLE.

II. Déroulement du projet

1. Planning théorique

Le cahier des charges étant principalement dirigé vers l'améliorations, la miniaturisation serait notre premier défi, étant donné que le reste de ces améliorations n'est envisageable qu'après impression du circuit est soudure des composants.

N'ayant, au début, qu'une vague idée sur le sujet nous avons prévu suivre les étapes suivantes pour réaliser ce projet dans les 50 heures lui y sont dédiées :

- Etude bibliographique, sur ce qui est déjà existant dans le marché.
- Etude minutieuse du projet P25 de l'année précédente.
- Miniaturisation des composants (acheter des composant de tailles inférieures)
- Miniaturisation du circuit (routage)
- Soudage des composants
- Programmation du Panstamp NRG 2 (convertir le code déjà fait sur AVR vers NRG)
- Programmation du Maxime 78M6610 (détection des états de la charge)
- Construction du Boitier

2. Choix technologique : Matériel et Logiciel

En se basant sur la liste de composants faite l'année dernière, nous avons éliminé ceux que nous n'utiliserons pas (tel que des switches ou les LED) et commandons le reste en taille plus petite (composant CMS 0603), vous trouverez en annexes la listes des composants et leurs prix ainsi que les liens vers leurs fabricants respectifs.

Comme logiciel de design, nous avons suivi les recommandations de nos professeurs et choisi Altium Design qui est fourni avec licence par l'école.

Autodesk Inventor était notre logiciel utilisé pour la conception du boitier.

3. Réalisation du circuit

A défaut de connaissance du logiciel Altium, nous nous somme référer au guide que propose M. Boé ainsi qu'à d'autre vidéo dont les liens sont joints aux annexes.

La réalisation du circuit comporte 4 étapes principales :

- Schématique
- Empreintes et Librairies
- PCB
- Impression du circuit

3.1. La Schématique

Dans cette partie, nous avons suivi pas à pas ce que Thomas et Sylvain avaient fait, excepter pour la connexion entre Maxime et Panstamp, où ils ont préféré laisser le choix à l'utilisateur entre un protocole SPI ou I2C en y intégrant des switches, quant à nous, suite à une recommandation de nos encadrants et afin de réduire la taille du circuit, nous avons pris une connexion SPI (MISO-MOSI).

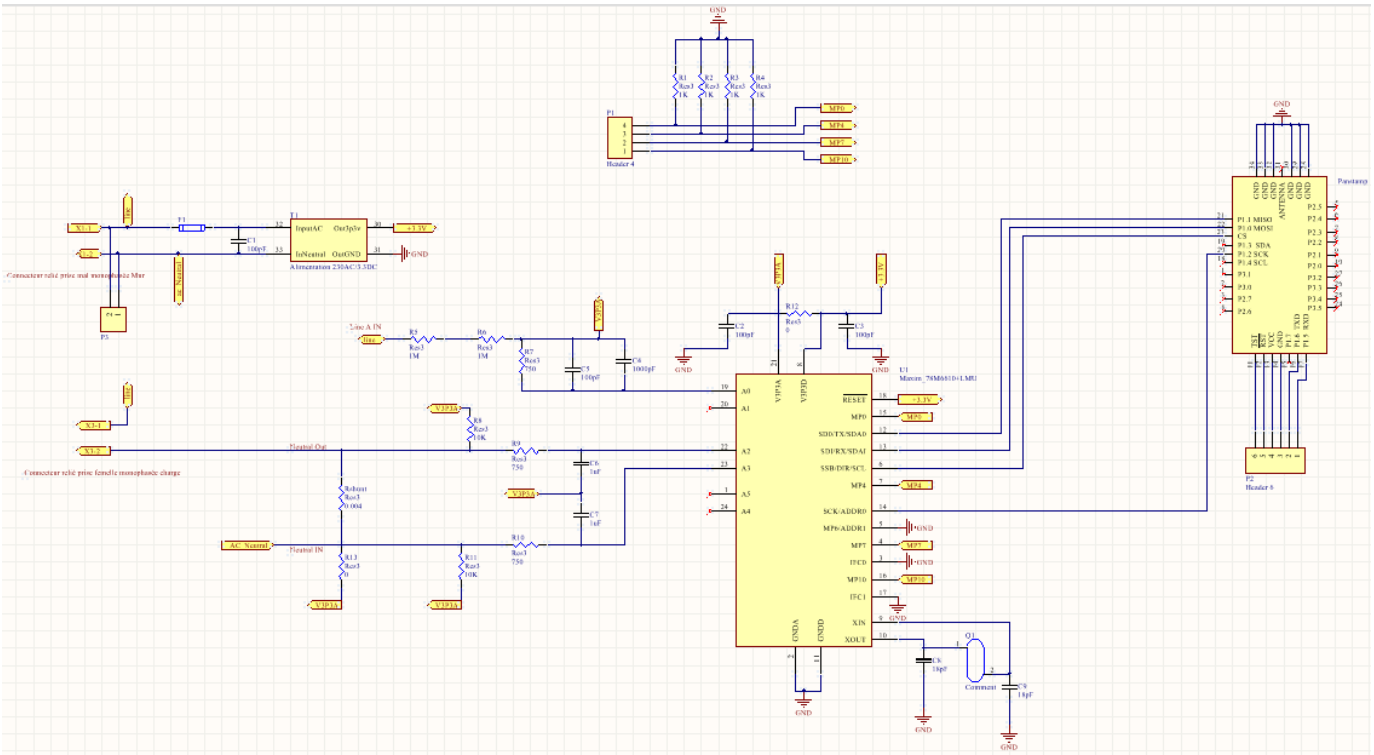
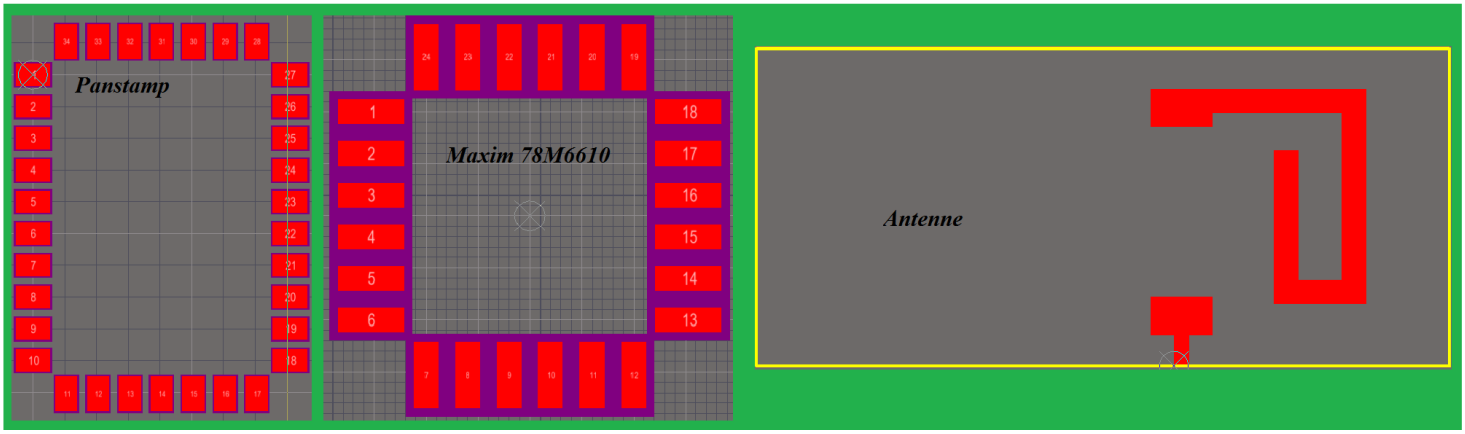


Figure 2 : Schématique SmartMeter

3.2. Les empreintes personnalisées

Les bibliothèques incluses dans le logiciel y compris celles de Polytech ne contienne pas les empreintes de Maxime ou encore du Panstamp, il fallait donc les réaliser en se référant à la description rigoureuse sur les Datasheets.



3.3. Le PCB

Cette partie de la réalisation du circuit est la plus importante, elle a été faite et refaite mainte fois avant d'arriver à une version satisfaisante, puisqu'elle est le chemin vers la miniaturisation du circuit.

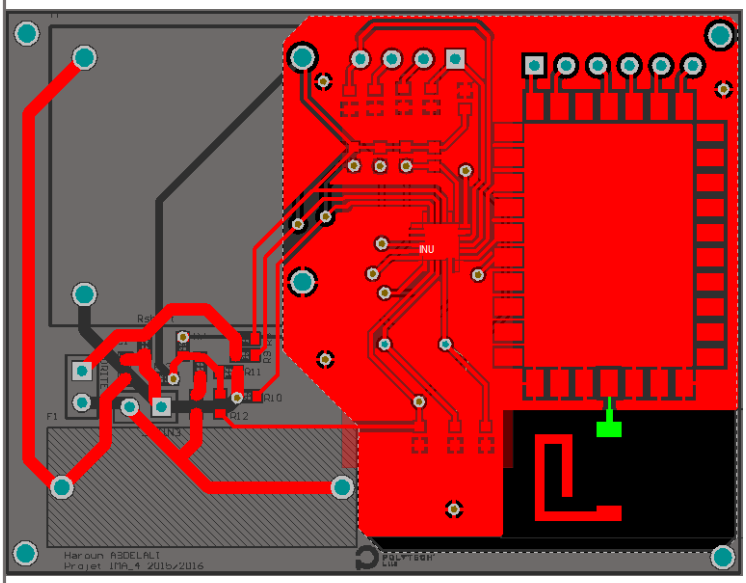


Figure 4 : Top PCB SmartMeter

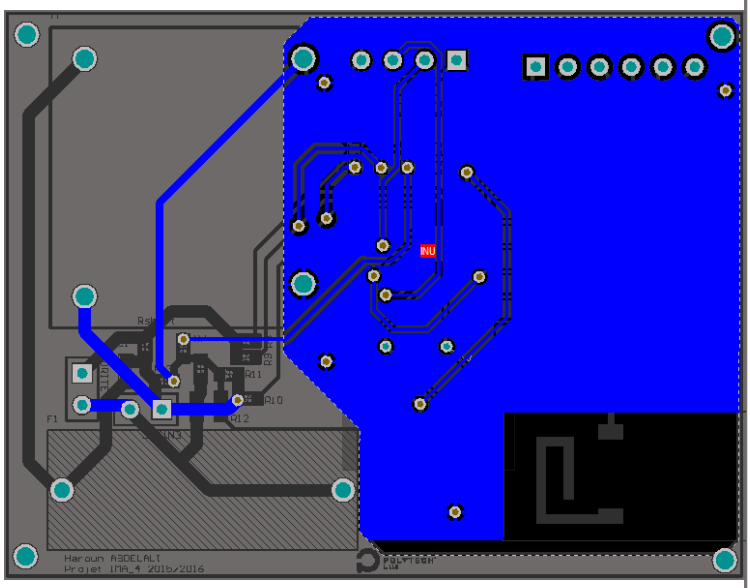


Figure 3 : Bottom PCB SmartMeter

Deux parties hétérogènes sont visible sur ce circuit, celle en haute tension en amont du transformateur, où les pistes sont d'une largeur de 2.16mm et puis celle de petite tension en aval de ce dernier, dans ce cas la largeur des pistes et de 0.3 mm.

A la différence de la technologie AVR où l'antenne du Panstamp se résumé à un simple fil en cuivre, la NRG 2 exige une forme particulière de piste en guise d'antenne.

3.4. Le circuit gravé

Une fois que le PCB est validé par M. Boé et M. Flamen il passe à l'étape d'impression.

A cause de la taille des pistes très fines, l'impression a généré une confusion entre les pistes, il fallait donc refaire manuellement une séparation entre celle ci

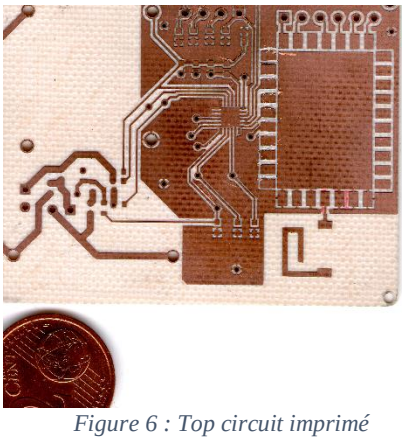


Figure 6 : Top circuit imprimé

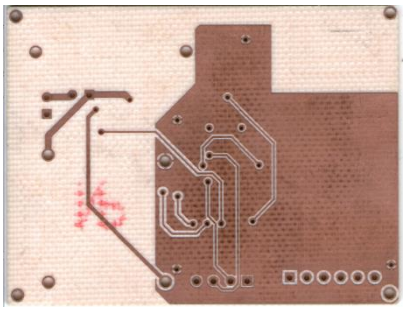


Figure 5 : Bottom circuit imprimé

4. Programmation du Panstamp

Les données des mesures du SmartMeter sont transmises grâce à deux Panstamps, le premier est soudé sur la carte, et le deuxième est soudé sur le Panstick qui sera brancher sur l'ordinateur.

Figure 7 : Panstmap NRG souder sur le Panstick 4.0

Une des différences entre le AVR et L'NRG, est que le AVR comporte un microcontrôleur Atmega328p quant à NRG un microcontrôleur MSP430 dont la programmation nous est inconnue.

Grâce à un travail similaire fait par l'étudiante Céline LY, j'ai réussi à me familiariser avec le MSP430 et finalement d'allumer une LED sans passer par l'IDE Arduino.

Pour ce faire nous avons d'abord télécharger un code via cet IDE afin de révéler la méthode qu'il suit, et nous somme arrivés au fichier .jar chargé du téléchargement, ce qui nous a permis de l'inclure de le Makefile.

Vous trouverez en Annexe les deux codes correspondant à l'allumage de la LED et celui du makefile.

5. Conception 3D du boitier

N'ayant pas eu l'occasion de réaliser un dessin en 3D au paravent, cette partie du projet était une découverte.

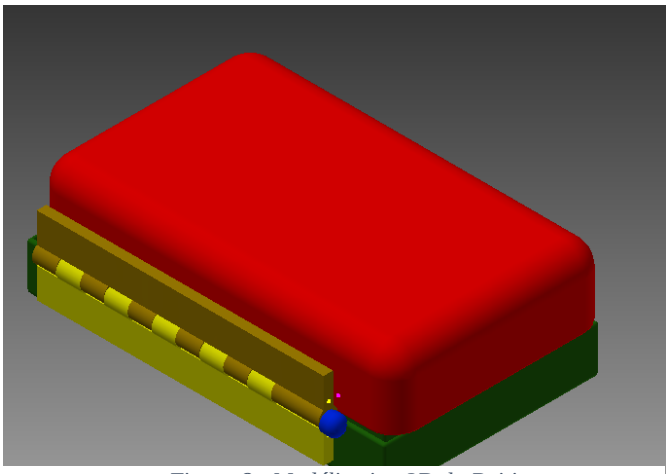


Figure 8 : Modélisation 3D du Boitier

III. Reste à faire

Par manque de temps, je n'ai pas pu finir ce projet

1. Soudage des composants

Ayant de très fines pattes, le Maxime reste le composant le plus délicat à souder, d'où l'intérêt de le souder en premier.

2. Programmation du Panstamp

- Les codes de réception et émission étant faits pour l'AVR, il reste de les adapter à la technologie NRG (microcontrôleur msp430)

- Le calibrage du Maxim qu'a été fait l'année dernière prend en considération une plage maximale de mesure (667V et 63A), le but cette fois ci et de réduire cette plage afin d'améliorer la précision, ce qui se fait via des registres de commande de la façon suivante :
  - Configuration du registre d'étalonnage de la tension (0x189)

  VFSCALE = 230V = 0x0000E6

  - Configuration du registre d'étalonnage du courant (0x18C)

  IFSCALE = 20A = 0x000014

  Une fois ces deux registres sont configurés, la procédure de calibration du Maxim est la même que celle suivie par Thomas et Sylvain.

- Détection d'état de fonctionnement de la charge qui se traduit par le taux d'énergie absorbée, cette donnée est contenue dans le registre WATT du Maxim
  Il suffirait donc de récupérer cette valeur et lui rajouter les trois conditions de fonctionnement
  Eteint : WATT = 0
  En veille : 0 < WATT < puissance nominale
  Actif : WATT = puissance nominale

IV. Difficultés rencontrées

- La miniaturisation de quelques composants tel que le transformateur s'est avérée impossible, car introuvable dans le commerce.
- La réalisation des empreintes, Panstamp et Maxim, en utilisant le logiciel Altium
- La Clearances minimum du PCB autorisée à Polytech est de 0.2mm, cependant la distance entre ces pattes du Maxim est encore plus inferieur que cela, ce qui a causé des complications au routage du circuit.
- N'avoir aucune référence sur la méthode de téléchargement du code sur le Panstamp sans utilisation de l'IDE Arduino.

- Mon binôme ayant rejoint le projet après un mois du commencement pour se désister avant un mois de la fin, m'a causé d'importants perturbations.

- La manque de méthodologie et de gestion de temps était la plus importante difficulté durant ce projet.

V. Bilan

- Durant les 40 heures dédiées à ce projet 30% du travail était fait.
- Miniaturisation du circuit de 75%
- Calibration faite théoriquement uniquement.
- Modélisation 3D du boitier en attente de validation.
- Passage de la technologie AVR à l'NRG n'a pas dépasser l'allumage d'une LED (reste codage Modem)
- Détection d'état de fonctionnent, n'a pas était fait.

Conclusion

Les 12 semaines de ce projet m'ont permis d'enrichir mes connaissances, commençant par une maitrise du logiciel de conception Altium, ainsi que de l'environnement Panstamp et ses différentes technologies.

La collaboration avec M. Guillaume RENAULT(Industriel) m'a permis de me projeter sur le monde professionnel.

A l'issu de ce projet, j'ai pu mettre le doigt sur un de mes plus importants lacunes, la bonne gestion du temps est désormais le premier élément de ma liste de travail sur soi.

Annexes

1. Tableau des composants

Description	Fabricant	Fournisseur	Quantité
Panstamp NRG2 (863-873Mhz)	Panstamp	Panstamp	2
Transformateur Maxim 78M6610+LMU	VIGORTRONIX	Farnell	1
Oscillateur Quartz 20Mhz	IQD FREQUENCY PRODUCTS	Farnell	1
Résistances à couches épaisses CMS - 0603 - 100ohms	TE Connectivity	RS	1 paquet de 50
Résistances à couches épaisses CMS - 0603 - 750ohms	Vishay	RS	1 paquet de 50
Résistances à couches épaisses CMS - 0603 - 1Kohms	TE Connectivity	RS	1 paquet de 50
Résistances à couches épaisses CMS - 0603 - 10Kohms	TE Connectivity	RS	1 paquet de 50
Résistances à couches épaisses CMS - 0603 - 1Mohms	Vishay	RS	1 paquet de 50
Résistance de shunt CMS - 0603 - 0,004ohms	Arcol	RS	1 paquet de 5
Condensateurs céramique multicouche CMS - 0603 - 1uF	TDK	Farnell	1paquet de 10
Condensateurs céramique multicouche CMS - 0603 - 0,1uF	Multicomp	Farnell	1paquet de 10
Condensateurs céramique multicouche CMS - 0603 - 18pF	WALSIN	Farnell	1paquet de 10
Leds rouge CMS	ROHM	Farnell	12
Borniers de puissance	CAMDENBOSS	Farnell	1
Ralange electrique	APSA	Farnell	1

2. Code allumage LED

```c
#include <cc430f5137.h>
#include <stdint.h>
#include <msp430.h>

#define PWM_PERIOD 25

// Mapping the LEDs using the PWM
void port_mapping(void){
    PMAPPWD = 0x02D52; // Get write-access to port mapping regs
    P3MAP0 = PM_TA1CCR1A; // Map TA1CCR1 output to P3.0
    P3MAP1 = PM_TA1CCR2A; // Map TA1CCR2 output to P3.1
    P3MAP2 = PM_TA0CCR1A; // Map TA0CCR1 output to P3.2
    P3MAP3 = PM_TA0CCR2A; // Map TA0CCR2 output to P3.3
    P3MAP4 = PM_TA0CCR3A; // Map TA0CCR3 output to P3.4
    P3MAP5 = PM_TA0CCR4A; // Map TA0CCR4 output to P3.5
    PMAPPWD = 0; // Lock port mapping registers
    P3DIR |= 0x3F; // P3.0 to P3.5 as output
    P3SEL |= 0x3F; // P3.0 and P3.3 options select
}

int main ()
{
    port_mapping();

    while (1)
    {
        WDTCTL = WDTPW | WDTHOLD;

        TA0CCR0 = PWM_PERIOD;
        TA1CCR0 = PWM_PERIOD;

        TA0CCR1 = (0x10 * 25)/ 0xFF;
        TA0CCR2 = 0x10 / 0xFF;
        TA0CCR3 = 0x10 / 0xFF;
        TA0CCR4 = 0x10 / 0xFF;

        TA1CCTL1 = OUTMOD_7; // CCR1 reset/set
        TA1CCTL2 = OUTMOD_7; // CCR2 reset/set
        TA0CCTL1 = OUTMOD_7; // CCR1 reset/set
        TA0CCTL2 = OUTMOD_7; // CCR2 reset/set
        TA0CCTL3 = OUTMOD_7; // CCR3 reset/set
        TA0CCTL4 = OUTMOD_7; // CCR4 reset/set

        // Activate PWM
        TA0CTL = TASSEL_1 + MC_1 + TACLR; // ACLK, up mode, clear TAR
        TA1CTL = TASSEL_1 + MC_1 + TACLR; // ACLK, up mode, clear TAR

        _delay_cycles(500);
    }
return 0;
}
```

3. Code Makefile

```
# for launchpad
CC=msp430-gcc
# for device
FLAGS= -mmcu=cc430f5137
# wall=include all warnings
CFLAGS=-Os -Wall $(FLAGS)
# execute this file, the target
TARGET=main
# device
DEVICE=/dev/ttyUSB0

# include all .o and .c
SOURCES=$(wildcard *.c)
OBJECTS=$(SOURCES:.c=.o)

## Default rule executed
all: $(TARGET)              # include everything from target

## Clean Rule
clean:                  #
    @-rm -f $(TARGET) $(OBJECTS)

## Rule for making the actual target
$(TARGET): $(OBJECTS)           # $@ applies for first argument and $^ for second argument

    @echo "============="       #echo = text
    @echo "Linking the target $@"
    @echo "============="
    @$(CC) $(FLAGS) -o $@ $^ $(LIBS) #include gcc compiler, the flags (in this case mmcu=cc430f5137)
    @echo -- Link finished --

## Generic compilation rule
%.o: %.c
    @echo "============="
    @echo "Compiling $<"
    @$(CC) $(CFLAGS) -c $< -o $@ # include all flags, not only -mmcu=cc430f5137 as before

deploy:
    msp430-objcopy -O ihex $(TARGET) $(TARGET).hex
#   mspdebug tilib "prog $(TARGET)" #transmitting everything to the end device
    java -jar ./BsLoader.jar $(TARGET).hex $(DEVICE) --verif-off --verbose-on
```

4. Plan du Boitier