IMA5 | Sylvain FOSSAERT

2014/2015 | Thomas EDERLE

Université Lille1 Sciences et Technologies

POLYTECH LILLE

[REALISATION D'UN SMART METER]

Encadrants: *Xavier Redon, Alexandre Boé*

Industriel: *Guillaume Renault*

Table des matières

I)-Introduction

Dans le cadre de notre dernière année d'étude en école d'ingénieur Polytech Lille, il nous est demandé de choisir un projet comportant une étude, une conception et une réalisation sur un sujet préalablement défini. Ce projet a pour but de nous responsabiliser et nous confronter aux démarches et aux problématiques que peuvent rencontrer les ingénieurs dans leur travail. L'intérêt principal est aussi d'approfondir certains domaines de notre formation.

I.1)-Quel est le but du projet ?

La consommation d'énergie est un poste de dépense qui prend de l'importance et pèse de plus en plus sur les factures d'électricité des particuliers. Aujourd'hui, mesurer en continu ces consommations énergétiques est devenu indispensable. Créer un outil de mesure permettrait au particulier de visualiser l'impact de ses gestes, d'établir des prévisionnels financiers et d'adopter un comportement éco-responsable dans le but d'obtenir un bâtiment à basse consommation d'énergie.
Ce projet, proposé par Guillaume Renault, tuteur industriel, consiste en la réalisation d'un « Smart Meter ». Ce dispositif a pour but final de transmettre à une « base » la puissance consommée d'un appareil électrique présent en aval afin d'y établir des estimations de dépense énergétique.

I.2)-Pourquoi avons-nous choisi ce projet ?

Intéressés par les sciences de l'électronique et des systèmes embarqués, ce projet est pour nous un bon compromis entre le développement hardware et software. En effet, en partant du cahier des charges, il nous est demandé d'avoir une réflexion sur les composants choisis, de designer et rooter une carte mais également de configurer les contrôleurs et créer l'échange radio avec la base.

II)-Cahier des charges

Tout d'abord, le but du projet est de mesurer la tension instantanée, le courant instantané et la consommation instantanée d'un appareil connecté sur une prise secteur afin d'en extraire la puissance active de consommation.

La seconde partie est de transmettre ces mesures en RF via un module panstamp. Pour atteindre ces objectifs, une carte électronique devra être réalisée et s'interfacer entre la prise secteur et l'appareil dont on souhaite mesurer la consommation. De plus, la carte électronique devra être conçue pour être alimentée par la prise secteur mais la consommation de cette carte doit être négligeable.

La plage de consommation mesurable sera de 1W à 8KW avec un pas au plus proche de 0.1W ceci permettra à l'utilisateur de mesurer la consommation des appareils en veilles.

Points d'attention **:**

- Les mesures devront tenir compte du cos Phi
- La plage de consommation mesurable sera [1W - 8kW] avec un pas au plus proche de 0,1W
- S'agissant d'un circuit de mesure de consommation, sa consommation propre (hors module Panstamp) doit être négligeable.

Pour réaliser ce système, il est préférable d'utiliser les modules panstamp. Un module contient un Atmega328p et une interface RF CC1101. Les modules panstamp sont programmés comme n'importe quelle autre plateforme Arduino, avec la différence que le panstamp peut faire de la communication sans fil à basse puissance. (1uA en mode veille et 2.5mA pendant les transmissions)

Pour ce faire, le projet peut être réalisé de la manière suivante :

1. Choix des composants afin de respecter le cahier des charges
2. Réalisation d'une carte électronique de test
3. Programmation des panstamp afin d'obtenir une communication sans fil
4. Tests
5. Miniaturisation et adaptation du circuit aux contraintes d'espace.

III)-Gestion du projet :

Le cahier des charges étant fixé, nous pouvons visualiser ci-dessous le planning des tâches effectuées à partir du 15 septembre 2014 jusqu'au 18 décembre 2014 afin de respecter les attentes en termes de spécification et de timing.
Dans un premier temps, nous nous sommes concentrés sur la partie électronique. Les quatre premières semaines ont été destinées à la recherche des composants pouvant répondre aux contraintes du cahier des charges. Après avoir défini les différents composants nécessaires pour notre système, la semaine suivante (semaine 5) a permis de réaliser notre schématique. Les trois semaines suivantes nous ont permis de router deux cartes électroniques (une carte en traversant et une carte en cms) et de lister l'ensemble des composants nécessaires pour notre commande. La semaine qui suit (semaine 9) a permis de valider et modifier certaines parties du design de la carte. La semaine suivante (semaine 10) a permis d'exporter la carte dans des formats différents permettant de tirer de façon chimique ou avec une graveuse. Cette semaine nous a notamment permis de commencer à étudier la programmation des panstamp et les protocoles qui y sont associés.
La semaine qui suit (semaine 11) nous a permis d'étudier plus en profondeur les fonctions pour réaliser une communication sans fil et utiliser le protocole de communication SWAP. Lors de cette semaine nous avons obtenu notre circuit imprimé et réaliser des communications RF de test à l'aide de l'environnement arduino.
La dernière semaine (semaine 12) nous avons installé les éléments nécessaires pour développer notre communication panstamp sur linux et commencé à retranscrire les bibliothèques C++ en langage C afin d'utiliser avr-gcc et avrdude.

III.1)-Planning :

#	L.		S.
38	15-Sep	Première approche du sujet	20-Sep
39	22-Sep	Réflexion et recherche	27-Sep
40	29-Sep	Réflexion et recherche	4-Oct
41	6-Oct	Etude de datasheets (Panstamp, Microship, etc..)	11-Oct
42	13-Oct	Etude de datasheets	18-Oct
43	20-Oct	Etude/conception de l'alimentation + réalisation schématique	25-Oct
44	27-Oct	Vacances Toussaint	1-Nov
45	3-Nov	Réalisation schématique + PCB + sélection des composants	8-Nov
46	10-Nov	Réalisation PCB	15-Nov
47	17-Nov	Réalisation PCB	22-Nov
48	24-Nov	Réalisation PCB	29-Nov
49	1-Dec	Communication Panstamp	6-Dec
50	8-Dec	Communication Panstamp	13-Dec
51	15-Dec	Rapport et présentation soutenance	20-Dec
52	22-Dec	Vacances de Noël	27-Dec
1	29-Dec		3-Jan

Etude électronique

Comme dit précédemment, nous nous sommes consacrés dans un premier temps à la réalisation de la partie électronique. Ainsi, le module panstamp comportant une interface RF et un microcontrôleur atmega328p étant déjà sélectionné par notre tuteur industriel (pour sa faible consommation), la première étape a été de choisir les composants nécessaires pour la mesure de la puissance et l'auto-alimentation de la carte.

Dans un premier temps, nous nous sommes concentrés sur le dispositif d'alimentation et nous en avons déduit qu'il est plus logique d'étudier le dispositif en aval, c'est-à-dire le circuit de mesures afin de déterminer le dimensionnement de l'alimentation. Ainsi, la première étape de la conception électronique consiste en la sélection de la technologie employée et des composants utilisés.

Le dispositif à mettre en place a pour objectif de donner la consommation d'une charge en aval. Pour cela, il nous faut déterminer la puissance active en temps réel.

I)-Comment calculer la puissance active?

La puissance active ou puissance moyenne est la puissance réelle consommée par la charge. Elle est donnée par :

$$Pa = \frac{1}{T}\int u(t).i(t)\,d(t)$$

Ainsi, numériquement il suffit d'échantillonner la tension u(t) et i(t) en fonction du temps de manière à respecter le théorème de Shannon. Ainsi, il faut au minimum que :

$$fe > 2.f$$

Avec :

$fe = fréquence\ d'échantillonnage$

$f = fréquence\ du\ signal$ à $mesurer$

PS : Plus fe est important, plus l'erreur sur la puissance mesurée est faible.

Il faut donc un dispositif capable d'échantillonner à 100 Hz. Cependant, afin de réduire l'erreur de mesure, il est de convention de prendre $fe = 20 * f$ soit $fe = 2000\ Hz$.

Nous retenons alors deux possibilités à étudier :

- La conversion s'effectue directement à l'aide de l'ADC (Analog to Digital Convertor) du contrôleur panstamp. Ainsi, grâce à l'échantillonnage et la numérisation des tensions renvoyées par la présence d'un capteur de courant et un capteur de tension, il est possible d'obtenir la puissance instantanée.

- La conversion, l'échantillonnage et le calcul de la puissance est effectuée via un circuit intégré spécialisé. La puissance est alors lue directement par le Panstamp.

II)-Etude de la première approche :

Le Panstamp embarque un ADC 10 bits disponible sur 8 voies analogiques. Comme décrit dans la datasheet, il est possible de configurer son fonctionnement. Par exemple, il peut être possible de régler la fréquence de son horloge (à l'aide d'un quartz externe) ou encore le nombre de voies analogiques converties par celui-ci.

Afin de déterminer la faisabilité de cette solution, nous allons effectuer les calculs suivants en faisant varier la fréquence de l'horloge de l'ADC dans le cadre d'une utilisation non optimum de celui-ci.

Le Panstamp permet de fonctionner à une fréquence comprise entre 50 et 200 kHz. Sachant que l'ADC met en moyenne 15 cycles pour effectuer une conversion, on en déduit le tableau suivant :

Horloge (kHz)	Fréquence de conversion si 1 voie utilisée (kHz)	Fréquence de conversion dans le pire des cas (Hz)
200	13	1625
50	3.3	416

Ainsi, on en déduit qu'il est possible de réaliser une conversion directement via le Panstamp seulement si celui-ci est bien configuré ou que la fréquence d'horloge de l'ADC est élevée. Cependant, plus la fréquence de l'horloge est élevée, plus l'imprécision sera grande donc cette première approche ne correspond pas à notre application.

Figure 23-6. ADC Timing Diagram, Auto Triggered Conversion

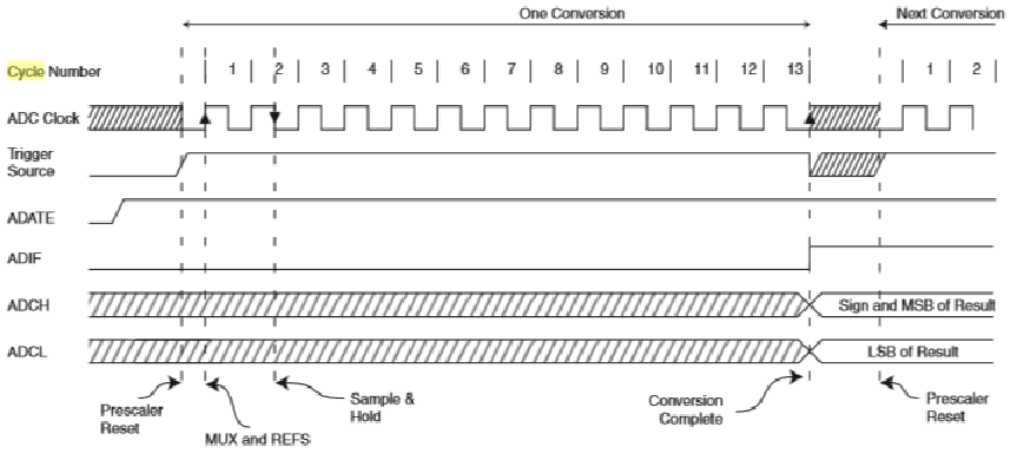


Figure 1 : chronogramme datasheet atmega328p

III)-Etude de la deuxième approche :

Comme dit précédemment, la deuxième approche consiste à intégrer sur la carte électronique un circuit intégré spécialisé permettant le calcul direct de la puissance instantanée.

III.1)-MCP3911 :

Le circuit intégré MCP3911 de Microchip est semblable au circuit 78M6610+LMU de Maxim mais comporte deux convertisseurs analogiques numériques. Ainsi, la mesure du courant est faite en parallèle de celle de la tension.

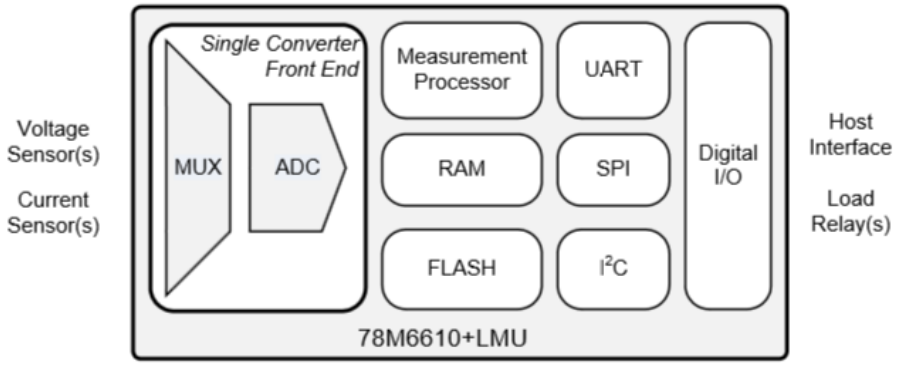
III.2)-MCP3905 :

Le circuit intégré MCP3905 de Microchip ne comporte pas de communication SPI et calcule uniquement la puissance active. De plus, la variation de la fréquence délivrée en sortie du composant est proportionnelle à la mesure de la puissance. Donc ce composant ne correspond pas à notre application.

III.3)- Circuit intégré « Maxim 78M6610+LMU »

Le 78M6610+LMU est un circuit numérique spécialisé dans la mesure d'énergie. Il comporte un convertisseur Analogique-Numérique 24 bits multiplexé sur 4 voies. Son microcontrôleur intégré permet à la fois différentes mesures de puissance tel que la puissance active, la puissance réactive, la puissance apparente, le facteur de puissance. Le circuit intégré est capable de mesurer les valeurs instantanées de la tension, du courant et des puissances mais aussi de calculer l'énergie consommée. Le composant comporte une liaison série SPI, I²C ou UART. Nous pouvons donc communiquer en série entre le circuit intégré et notre microcontrôleur atmega328p.

De plus, le composant Maxim « 78M6610+LMU » permet d'obtenir un échantillonnage $fe = 4000\,Hz$ par voies et répond donc de manière correcte à la contrainte d'échantillonnage. Donc ce composant est idéal pour notre application.



IV)-Bilan :

De par les études effectuées précédemment, nous pouvons en déduire que la deuxième approche est plus cohérente puisqu'elle permet un taux d'échantillonnage plus important et donc une précision accrue. Enfin, cela libère des ressources conséquentes sur le panstam pour la réception et la transmission des données par radio fréquence. Cela permettra notamment de mettre en mode veille le panstamp lorsque celui-ci n'est pas sollicité.

V)-Calcul de la consommation du circuit

L'ensemble de nos éléments de mesures étant choisi, il nous est possible de dimensionner l'alimentation en conséquence. Ainsi, afin de choisir une alimentation assez compacte, on sélectionne un montage hacheur type Flyback. Veuillez-vous reporter aux annexes pour plus d'informations sur le fonctionnement du montage Flyback.

L'élément sélectionné est un dispositif intégrant à la fois le circuit de régulation et le transformateur:

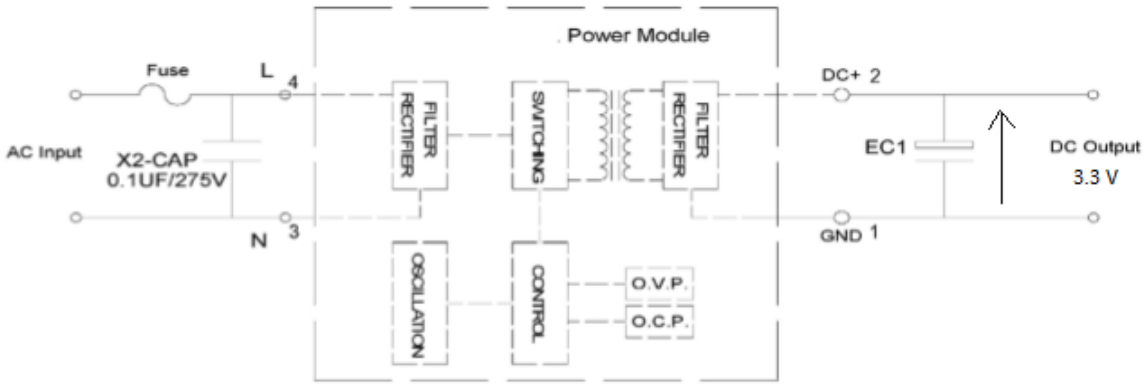
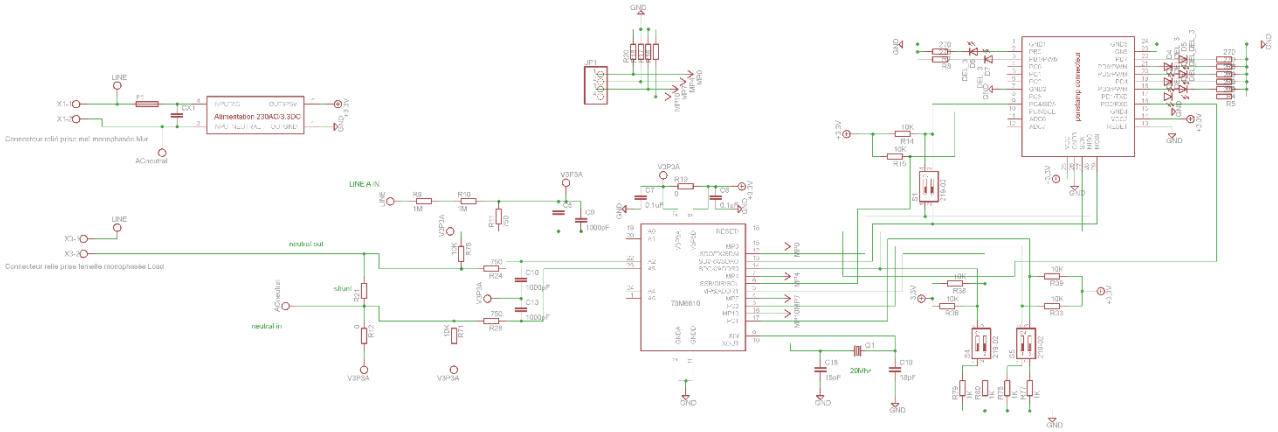


Figure2 : convertisseur AC/DC : *VTX-214-001-103*

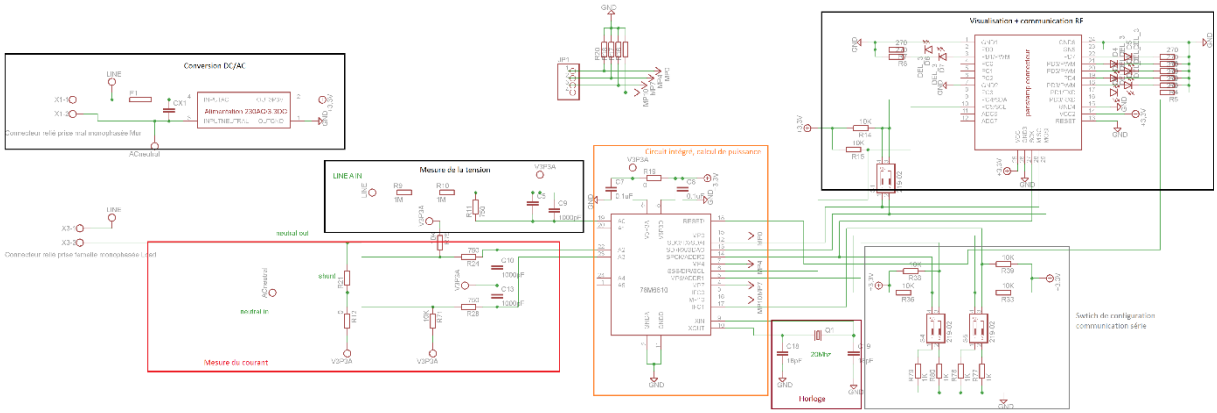
Sachant que le composant Maxim 78M6610+LMU à une consommation maximale de 10.3 mA et que celle du Panstamp est de 2.5 mA, on en déduit que la consommation totale est d'environ 12.8 mA. Ainsi, en prenant une marge supérieure, tout type d'alimentation 3.3V pouvant fournir une puissance de 43mW en sortie convient.

VI)-Etude du schématique

Après avoir étudié le composant Maxim 78M6610+LMU et son référence design, nous avons commencé à créer le schématique de notre circuit imprimé sur le logiciel EAGLE. Pour la réalisation de celui-ci nous nous sommes en grande partie inspiré du schématique du référence design et nous avons ajouté les éléments nécessaires pour une alimentation secteur 230V, l'intégration du Panstamp permettant d'envoyer les informations en liaison radio et des éléments supplémentaires pour faciliter le débogage lors de la calibration du circuit intégré (configuration des registres via des LED).

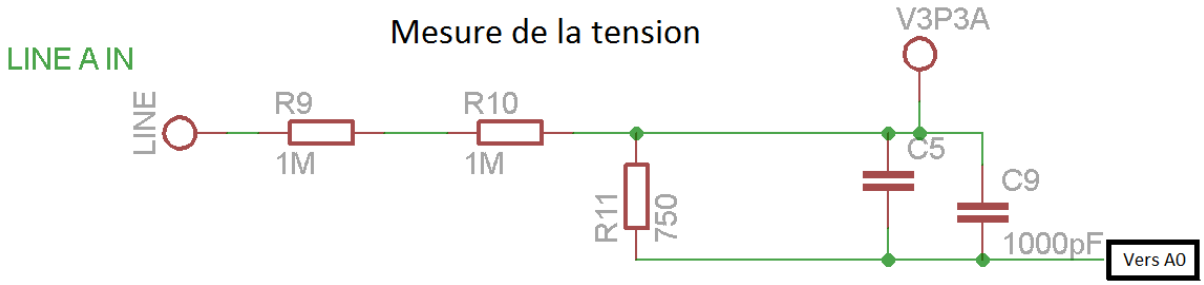


Nous pouvons découper notre schématique en plusieurs fonctions :



VI.1)-Etude de la fonction « Mesure de tension » :

Nous ne pouvons directement imposer une tension de 230V aux bornes du composant Maxim. Ainsi, il est nécessaire de passer par l'étage d'adaptation présent ci-dessous :



Les résistances série R9, R10, R11 limitent le courant et réalisent un diviseur de tension sur l'entrée tel que :

$$U(R11) = \frac{R11}{R11 + R10 + R9}.Vline$$

De plus, l'alimentation V3P3A est placée afin d'effectuer un offset (de référence) tel que :

$$V(A0) = \frac{R11}{R11 + R10 + R9}.Vline + V3P3A$$

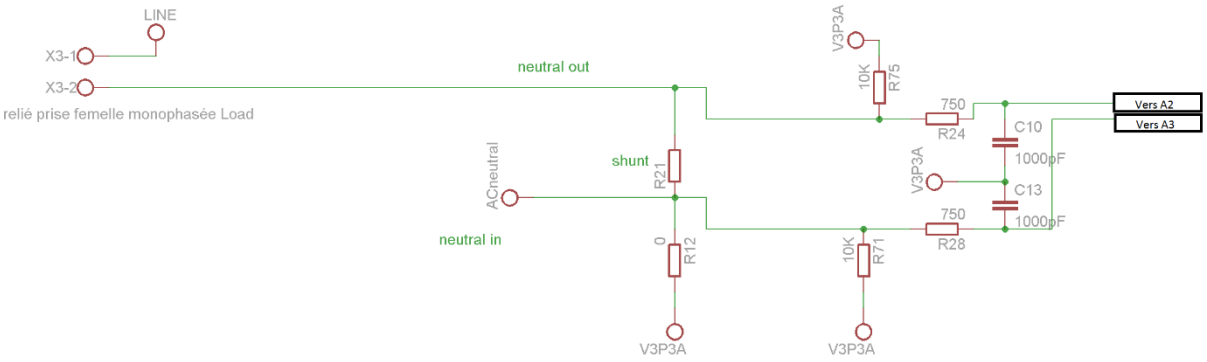
En remplaçant par les valeurs numériques, nous obtenons :

$$V(A0) = 3.75.10^{-4}.Vline + 3.3\,V$$

On en déduit pour $-230\sqrt{2} < Vline < 230\sqrt{2}$:

$$3.178 < V(A0) < 3.422\,V$$

VI.2)-Etude de la fonction « Mesure de courant » :



Ici, on mesure la différence de potentiel présente sur la résistance de shunt. Celle-ci nous donne une image du courant circulant dans celle-ci.

$$I(t) = \frac{U(t)}{R}$$

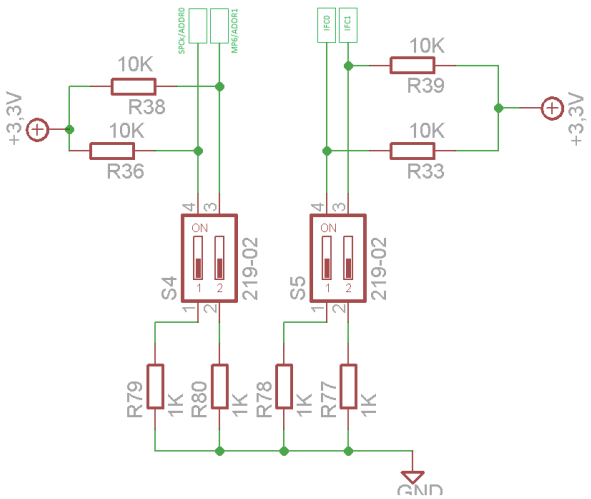
Lorsque la tension aux bornes du shunt est positive, c'est la sortie A2 qui mesure cette différence de potentiel, dans le cas contraire, c'est la sortie A3 qui relève la mesure. Comme pour la mesure de la tension, V3P3A sert de référence de mesure.

Remarque : Les capacités présentent aux bornes A2 et A3 du composant Maxim réalisent un filtrage des hautes fréquences

VI.3)-Etude des choix de communication

Sur la carte réalisée, les switchs permettent de choisir le protocole de communication série (SPI-I2C ou UART).

Le tableau suivant résume le protocole choisi en fonction de la position des commutateurs.



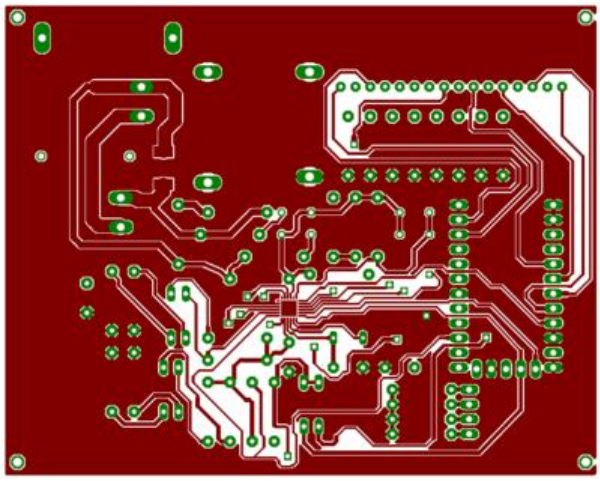
X = Ne pas tenir compte

	IFC0	IFC1
SPI	0	x
I²C	1	1
UART	1	0

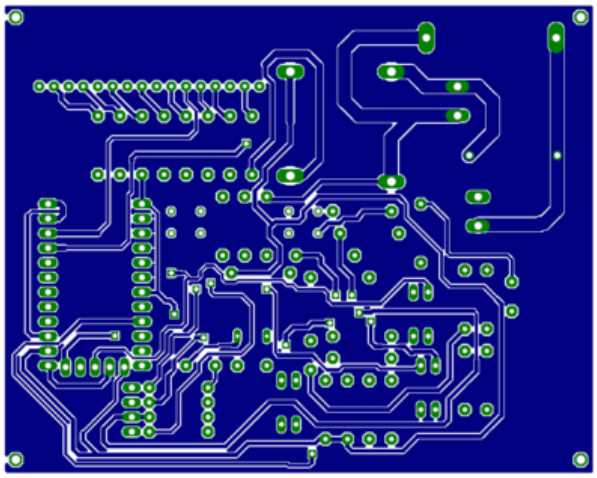
VII)-Réalisation du PCB :

Le PCB est la dernière étape avant le tirage de la carte. En effet, celui-ci permet le placement des composants et le routage des pistes électriques sur la carte afin d'en créer un typon. Cette étape demande donc beaucoup de minutie et d'expérience de jugé, c'est pourquoi nous avons donné trois semaines de notre temps sur cette partie.

Dans un premier temps, étant plus familiarisé aux composants traversants, nous avons entrepris une première version présente ci-dessous :

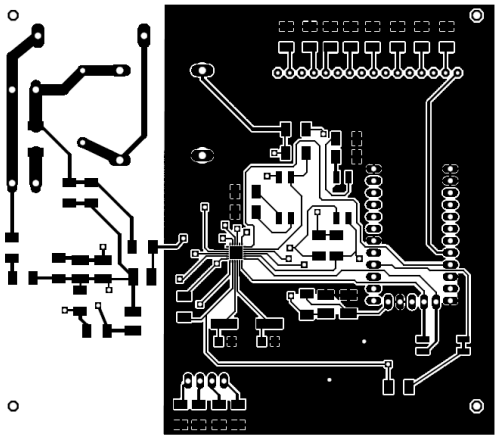


TOP

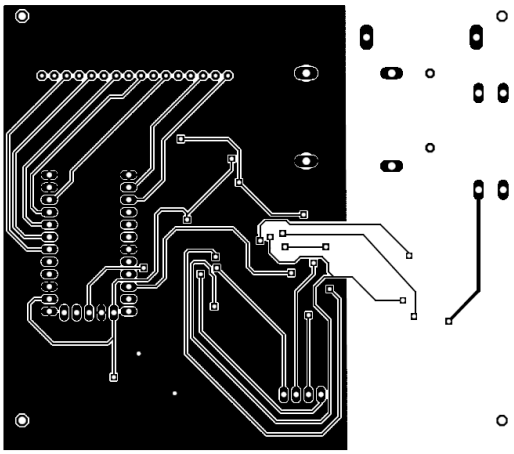


BOTTOM

Cependant, le fait de placer des composants traversant influe sur les lignes de champs et crée par la même occasion de nombreuses ruptures de masse. Ainsi, en remplaçant la technologie traversant par du CMS et en suivant les conseils de nos professeurs, nous avons supprimé les languettes et ilots de masse pour arriver à la carte réduite de prototype suivante :



TOP

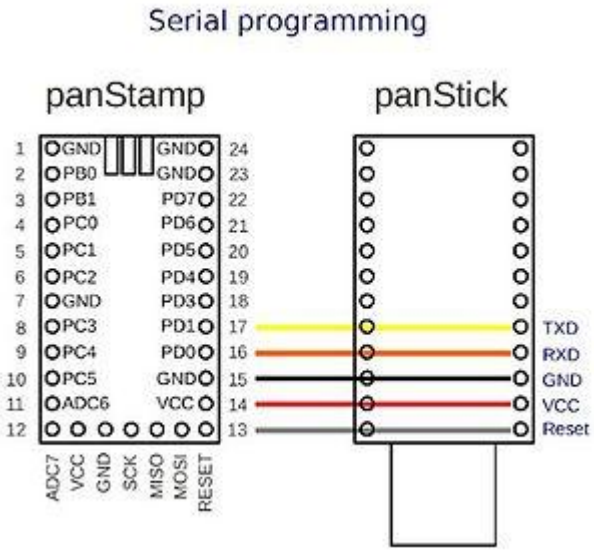


BOTTOM

Réalisation informatique

I)-Programmation et première approche

Afin de programmer nos Panstamps, nous avons câblé une communication série à l'aide d'un adaptateur USB FTDI. Nous pouvons notamment utiliser un Panstick pour réaliser ce processus. Les connections à réaliser sont visibles sur l'image ci-dessous.



Pour émettre ou recevoir une communication RF nous devons d'abord équiper notre Panstamp d'une antenne ou d'un connecteur SMA (type de connecteur coaxial dont l'impédance caractéristique est de 50 Ohms). Pour notre application nous avons choisi d'utiliser la solution par antenne car celle-ci est moins couteuse. L'antenne peut être réalisée à l'aide d'un simple fil, la longueur de notre fil permettra de définir notre fréquence porteuse donc notre longueur d'onde. Les fréquences indiquées sur le site de panstamp sont les suivantes :

Frequency = 868 MHz
Quarter Wavelength: 82.2 mm
Half Wavelength: 164.3 mm

Frequency = 915 MHz
Quarter Wavelength: 77.9 mm
Half Wavelength: 155.9 mm

Nous avons choisi d'utiliser la fréquence de 868 Mhz car par défaut le Panstamp émet à cette fréquence. Ceci nous évitera de paramétrer la fréquence porteuse dans nos programmes.

I.1)Le programme MODEM :

Dans un premier temps, afin de nous familiariser avec le développement de communication RF sur Panstamp, nous avons commencé à étudier le programme « modem » fourni dans la bibliothèque Panstamp. Le programme joue le rôle de passerelle entre le réseau de communication RF utilisé par les Panstamps et le port série de l'ordinateur ou le Panstamp est connecté. Le programme possède 2 modes de fonctionnement qui sont :

- Le mode data (permet d'écouter le réseau RF et de convertir en ASCII les paquets entrant et de les afficher sur un terminal)

- Le mode commande (permet de configurer cette « passerelle » par le biais de commandes AT sans prendre en compte la réception/transmission des paquets).

I.2)-Etude des bibliothèques :

Pour développer nos propres communications RF, nous avons étudié attentivement les différentes bibliothèques fournies sur le site de panstamp (bibliothèque arduino, bibliothèque panstamp, bibliothèque swap). Nous avons pu différencier les rôles de chacune.

- La partie arduino comme nous avons vu lors de nos précédents projets, permet de contrôler les I/O (analogique et digital), les ADC, les PWM etc…

- La partie Panstamp permet de contrôler les communications RF (transmission/réception), la synchronisation, la gestion de l'alimentation (mode veille/mode transmission)… On notera que la partie Panstamp permet de réaliser des communications RF simples mais elle ne possède pas de protocole de communication

- La partie swap possède toutes les fonctions nécessaires pour réaliser un protocole de communication simple et efficace. Nous pouvons visualiser ci-dessous les paquets pour une communication RF simple utilisant uniquement les fonctions de la bibliothèque Panstamp et les paquets utilisant le protocole de communication swap.

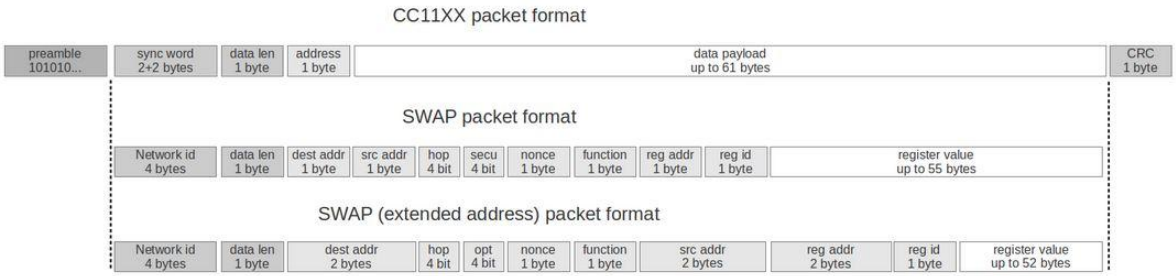


Figure 3 : constitutions des trames SWAP ou CC1101

Le protocole SWAP possède deux formats de paquets:

- Le format standard permettant d'adresser 255 Panstamps dans notre réseau
- Le format étendu permettant d'adresser 65 535 Panstamps dans notre réseau. Dans notre cas, nous utiliserons le format standard sachant qu'un particulier ne disposera pas de 255 prises électriques dans sa maison.

Le format standard que nous utiliserons se comporte de la manière suivante :

- 4 octets destinés à l'identifiant de notre réseau (**Network ID**) : Pour notre application cela peut être intéressant d'utiliser plusieurs sous réseaux par exemple un réseau pour le salon, un réseau pour la cuisine etc … L'utilisateur pourrait donc déterminer la pièce qui consomme le plus d'électricité dans sa maison.

- 1 octet destiné à la longueur des données (**data len**)

- 1 octet destiné à l'adressage du destinataire (**dest addr**) : Dans notre cas l'adresse sera entre 1 et 255. L'adresse de broadcast sera 0.

- 1 octet destiné à l'adressage de l'émetteur (**src addr**)

- 4 bits destinés à la sécurité des données transmises/émises (**secu**): Permettra de crypter les données

- 1 octet destiné à protéger l'utilisateur contre des attaques par le biais de requêtes répétées (**nonce**)

- 1 octet destiné à renseigner le type de trame (**function**), la trame peut être de type : **query**, **status** ou **command**.

- **La trame query** permet de faire une requête. Son but est de demander à un destinataire la valeur de ses registres de données. (Nous verrons plus tard comment sont organisés les registres de données utilisées dans le protocole swap. Remarque une trame de type query ne comportera pas de data).

- **La trame de type status** permet d'envoyer la valeur des registres de données. La réponse à une trame de type query sera une trame de type status.

- **La trame de type command** est utilisée pour contrôler les registres de données. On pourrait par exemple utiliser ce type de trame pour contrôler la valeur d'une PWM à distance, un interrupteur etc… Donc ce type de paquet pourra être utile dans notre application pour allumer ou éteindre notre carte à distance.

- 1 octet destiné à renseigner l'adresse du registre du panstamp destinataire (**reg addr**) ou l'on veut appliquer une trame de type **status**, **query**, et **command**.
- 1 octet destiné à renseigner l'ID du registre (**reg id**), l'ID est utilisé par le protocole de communication swap pour gérer ses registres de données. (Nous verrons par la suite en quoi cela consiste plus précisément).
- 1 à 55 octets destinés aux données (**data**)
- 1 octet destiné au CRC (le CRC est calculé automatiquement)

I.3)-Développement d'un programme de communication RF (sans protocole SWAP) :

Afin de prendre en main la communication entre deux panstamps, nous avons tout d'abord développé un programme sans l'utilisation du protocole SWAP et sur un environnement Arduino. Le fonctionnement de ce programme de test est le suivant:

Le Panstamp N°1 transmet un paquet au Panstamp N°2. Lors de la transmission du paquet, le Panstamp N°1 fait clignoter une LED verte et affiche le paquet transmit sur le port serie. Le Panstamp N°2 écoute le réseau RF, il possède une LED rouge qu'il va allumer jusqu'à la réception d'un paquet. Quand le Panstamp N°2 reçoit un paquet, il éteint la LED rouge et affiche sur le port série le paquet reçu. Nous pouvons visualiser le fonctionnement de notre programme sur la vidéo ci-dessus.

Programme source du transmetteur :

```
#include "HardwareSerial.h"

#define RFCHANNEL        0       // utilisation du channel 0
#define SYNCWORD1        0x85    // mot de synchronisation, octet haut
#define SYNCWORD0        0x47    // mot de synchronisation, low byte
#define SOURCE_ADDR      6       // adresse du panstamp emetteur
#define DESTINATION_ADDR 5       // adresse du panstamp qui recoit
#define LED              3       //led pin 8


CCPACKET packet;               // objet paquet
byte i;

void setup() {


  // Setup LED output pin
  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);

  panstamp.radio.setChannel(RFCHANNEL);     //configure le channel
  panstamp.radio.setSyncWord(SYNCWORD1, SYNCWORD0);  // configure le mot de synchronisation RF
  panstamp.radio.setDevAddress(SOURCE_ADDR);  //configure adresse du panstamp
  panstamp.radio.setCCregs();  // permet de prendre en compte les configurations ci-dessus
  panstamp.setHighTxPower();  // permet de transmettre en puissance haute

  packet.length = 10;  // defini la longueur des paquets
  packet.data[0] = DESTINATION_ADDR ; // premier paquet de la transmission correspond à l'adresse destinataire

  for(i=1 ; i<packet.length ; i++)
    packet.data[i] = i;              // remplissage des données

   Serial.begin(9600);               // initialisation du port série à 9600bauds
}

void loop() {
 digitalWrite(3, HIGH);
 delay(1000);   // delay ajouté pour visualiser la LED
  panstamp.radio.sendData(packet);  // transmission de la trame
 digitalWrite(3, LOW);
  delay(100);
  for(i=0 ; i<packet.length ; i++)
  Serial.print(packet.data[i]);        // afichage du paquet transmit

  Serial.println(" ");
  delay(100);
  panstamp.sleepSec(5);              // entre en mode veille pendant 5 secondes (courant consommé 1uA)
}

```

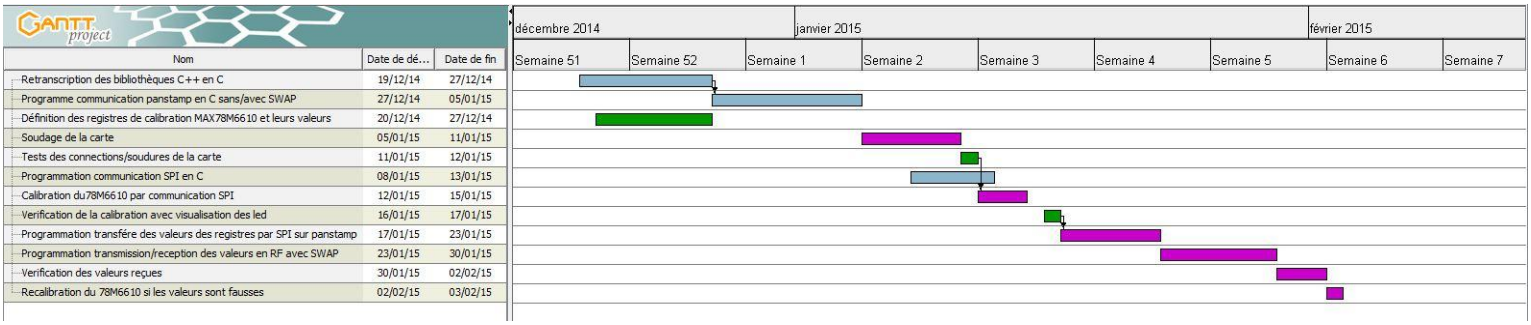
Programme source du récepteur :

```
#include "HardwareSerial.h"
#include <avr/wdt.h>
#include "TimerOne.h"

#define RFCHANNEL        0     // utilisation du channel 0
#define SYNCWORD1        0xB5  // mot de synchronisation, octet haut
#define SYNCWORD0        0x47  // mot de synchronisation, low byte
#define SOURCE_ADDR      5     // adresse du panstamp


CCPACKET txPacket;  // packet object
byte i;

void rfPacketReceived(CCPACKET *packet)
{

  digitalWrite(3, LOW);          // paquet reçu donc la led s'éteint

  if (packet->length > 1)
  {
  Serial.print("le paquet recu est: ");

  for(i=0 ; i< packet->length ; i++) // affichage du paquet recu sur le port série
  Serial.print(packet->data[i]);

  Serial.println(" ");
  delay(100);

  }
  delay(1000);
}


void setup() {

  pinMode(3, OUTPUT);
  digitalWrite(3, HIGH);
  panstamp.radio.setChannel(RFCHANNEL);            //configure le channel
  panstamp.radio.setSyncWord(SYNCWORD1, SYNCWORD0);// configure le mot de synchronisation RF
  panstamp.radio.setDevAddress(SOURCE_ADDR);      //configure adresse du panstamp
  panstamp.radio.setCCregs();      // permet de prendre en compte les configurations ci-dessus

  panstamp.radio.enableAddressCheck();//permet de recevoir seulement les paquets qui
                                      // correspondent à l'adresse source et l'adresse de broadcast
  panstamp.setPacketRxCallback(rfPacketReceived); //permet d'appeler la fonction rfPacketReceived
                                                  //lors de la réception d'un paquet
  Serial.begin(9600);           // initialisation du port série à 9600bauds

}

void loop() {
  digitalWrite(3, HIGH);
}
```

Remarque : L'ensemble des programmes seront développés en C sur linux et non en C++ sur l'environnement arduino.

Gestion du projet : les tâches à venir

I)- Diagramme de Gantt :



Nom	Date de dé...	Date de fin
Retranscription des bibliothèques C++ en C	19/12/14	27/12/14
Programme communication panstamp en C sans/avec SWAP	27/12/14	05/01/15
Définition des registres de calibration MAX78M6610 et leurs valeurs	20/12/14	27/12/14
Soudage de la carte	05/01/15	11/01/15
Tests des connections/soudures de la carte	11/01/15	12/01/15
Programmation communication SPI en C	08/01/15	13/01/15
Calibration du78M6610 par communication SPI	12/01/15	15/01/15
Verification de la calibration avec visualisation des led	16/01/15	17/01/15
Programmation transfére des valeurs des registres par SPI sur panstamp	17/01/15	23/01/15
Programmation transmission/reception des valeurs en RF avec SWAP	23/01/15	30/01/15
Verification des valeurs reçues	30/01/15	02/02/15
Recalibration du 78M6610 si les valeurs sont fausses	02/02/15	03/02/15

Vert: tâches à réaliser par Sylvain
Bleu: tâches à réaliser par Thomas
Violet: tâches à réaliser par Thomas & Sylvain

Nous pouvons visualiser ci-dessus, le diagramme de Gantt des tâches à venir. Nous avons organisé ce planning de manière à finir notre projet le 3 février, donc environ 3 semaines avant la date de la soutenance finale.
Ce planning nous laisse une marge de manœuvre si nous rencontrons des imprévus

Conclusion

Ce projet de fin d'étude est pour nous un réel défi qui nous tiens à cœur de relever en le menant à terme avant février autrement dit en réalisant sa mise en fonctionnement et ce, dans les délais qui nous sont impartis.
Ce projet ambitieux qui nous a été confié nous permet d'accroître considérablement les connaissances acquises jusqu'alors dans le cursus universitaire et ce, principalement dans le domaine des systèmes embarqués tout en prenant en considération les contraintes liées au cahier des charges et des normes de sécurité inhérentes à Polytech'Lille. De plus, sur le plan humain ce fut pour nous, une expérience très enrichissante car, nous avons dû partager et comprendre nos idées mais aussi prendre en compte les contraintes et impératifs définies afin de réaliser ce système. Autrement dit, ce projet d'étude est un travail d'équipe.

De plus, nous pensons avoir amélioré notre autonomie et méthodologie de par les recherches et démarches personnelles effectuées tout au long de ce projet.

En conclusion, ce début de projet nous a permis de réaliser une véritable étude de faisabilité à travers une réflexion autour des choix des composants, une étude technique mais aussi une étude de circuit

ANNEXES

Liste des composants :

Composant	Réf schématique	Valeur	Quantité	Prix unitaire	Coût total
Résistance de Shunt	R26	0.004 ohm	3	0,546	1,638
Boitier			1	21,11	21,11
Microruptеurs de sécurité			1	6,88	6,88
Bornier de puissance			2	0,84	1,68
Alimentation AC/DC 220/3.3V			1	6,26	6,26
Résistances	R9, R10	1 M	6	0,9	5,4
Résistances	R75, R71, R38, R39, R36, R33, R14, R3	10 K	24	0,285	6,84
Résistances	R11, R24, R28,	750 ohm	9	0,285	2,565
Résistances	R79, R80, R78, R77	1 K	12	0,19	2,28
Résistances	R7, R8, R6, R5, R4, R3, R2, R1	100 ohm	24	0,242	5,808
DEL Rouge	D8, D7, D6, D5, D4, D3, D2, D1		16	0,355	5,68
Switch	S1, S4, S5		6	1,12	6,72
Capacités	C9, C10, C13	1000 nF	9	0,977	8,793
Capacités	C7, C6, C11	0.1 uF	9	0,744	6,696
Capacités	C18, C19	18 pF	6	0,078	0,468

Capacité d'alimentation			2	0,209	0,418
Quartz 20 MHz	Y1	20 MHz	2	0,355	0,71
Barrettes 12 pôles			10	0,432	4,32
Barrettes 5 pôles			10	0,42	4,2
Entretoises plastique			10	0,186	1,86
Visses M3			4	0,164	0,656
Fusible 250V 63 mA			5	2,19	10,95
Ecrou			1	3,73	3,73
Rallonge M/F			1	9,52	9,52
					125,182 €