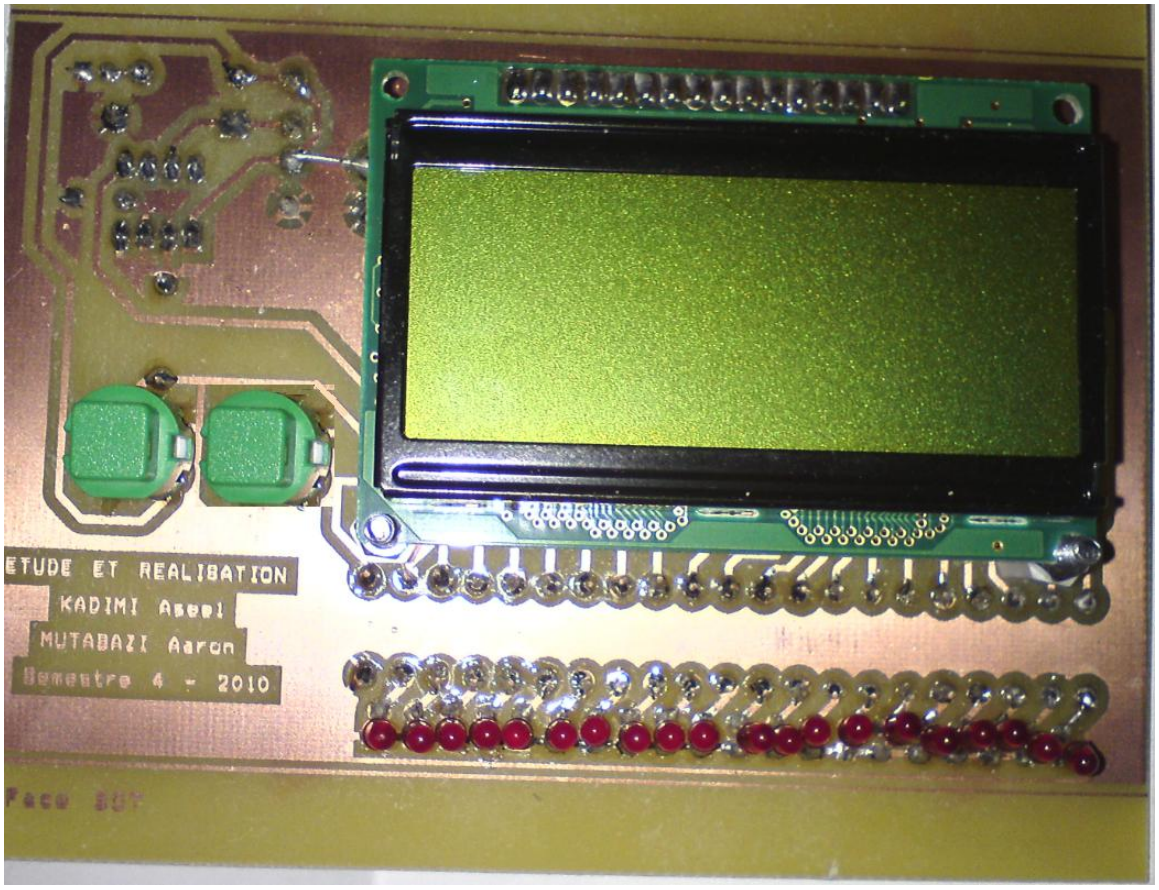
Université François Rabelais de Tours
Institut Universitaire de Technologie de Tours

Projet tutoré 2ème Année
Étude et réalisation Génie électrique S4
2009-2010

Conception d'un thermomètre numérique



KADIMI Aseel
MUTABAZI Aaron

M. Thierry LEQUEU
Mme AUGER

Table des matières

1. Introduction

Au cours du quatrième semestre nous avons du réaliser en binôme un projet dans le cadre de l'étude et réalisation. Nous avons choisi de réaliser un thermomètre électronique.

Ce thermomètre devra afficher l'heure et il sera possible de visualiser la température ambiante sur l'afficheur LCD ou bien à l'aide d'une rangé de LED. Dans un premier temps nous allons présenter le cahier des charges, ensuite nous nous présenterons les différentes fonctions au sein de la carte, puis nous expliquerons les moyens utilisés pour réaliser la carte et enfin, dans une dernière partie nous nous pencherons sur la partie programmation de notre projet.

2. Présentation du cahier des charges

PRESENTATION DU SYSTEME

Les thermomètres électroniques permettent de mesurés les températures ambiante de l'air, des liquides, des matériaux, avec une très grande précision. Un thermomètre électronique est composé d'un capteur de température et des composants électroniques qui ont pour rôle de traiter l'information et la rendre exploitable par l'usager.

OBJECTIF PRINCIPAL

- Créer un thermomètre électronique qui affiche la température via des L.E.D.

Figure 1 : Bloc principal

OBJECTIFS SECONDAIRES

- Afficher la température sur l'afficheur L.C.D.
- Afficher la date sur l'afficheur L.C.D.

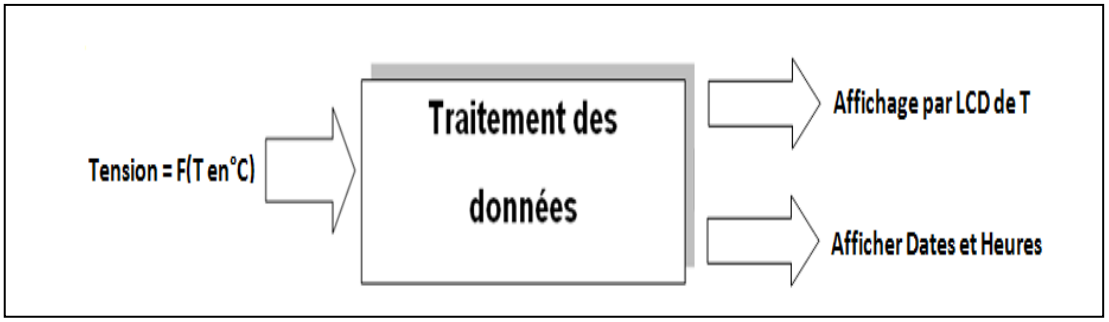


Figure 2 : Bloc secondaire

REALISATION

La température devra être captée à l'aide d'un capteur, notre choix se porte vers le LM75 car il à l'avantage d'être une capteur numérique. Afin de traiter les informations nous utiliserons le microcontrôleur AtMega 8535 pour contrôler l'affichage de la date et de la température. Il faudra au préalable effectuer une adaptation de tension car l'AtMega fonctionne sous +5v. Le typon de la carte final sera réalisé à l'aide du logiciel Orcad de même pour programmer nous utiliserons le logiciel Code Vision Avr.

PLANNING PREVISIONNEL

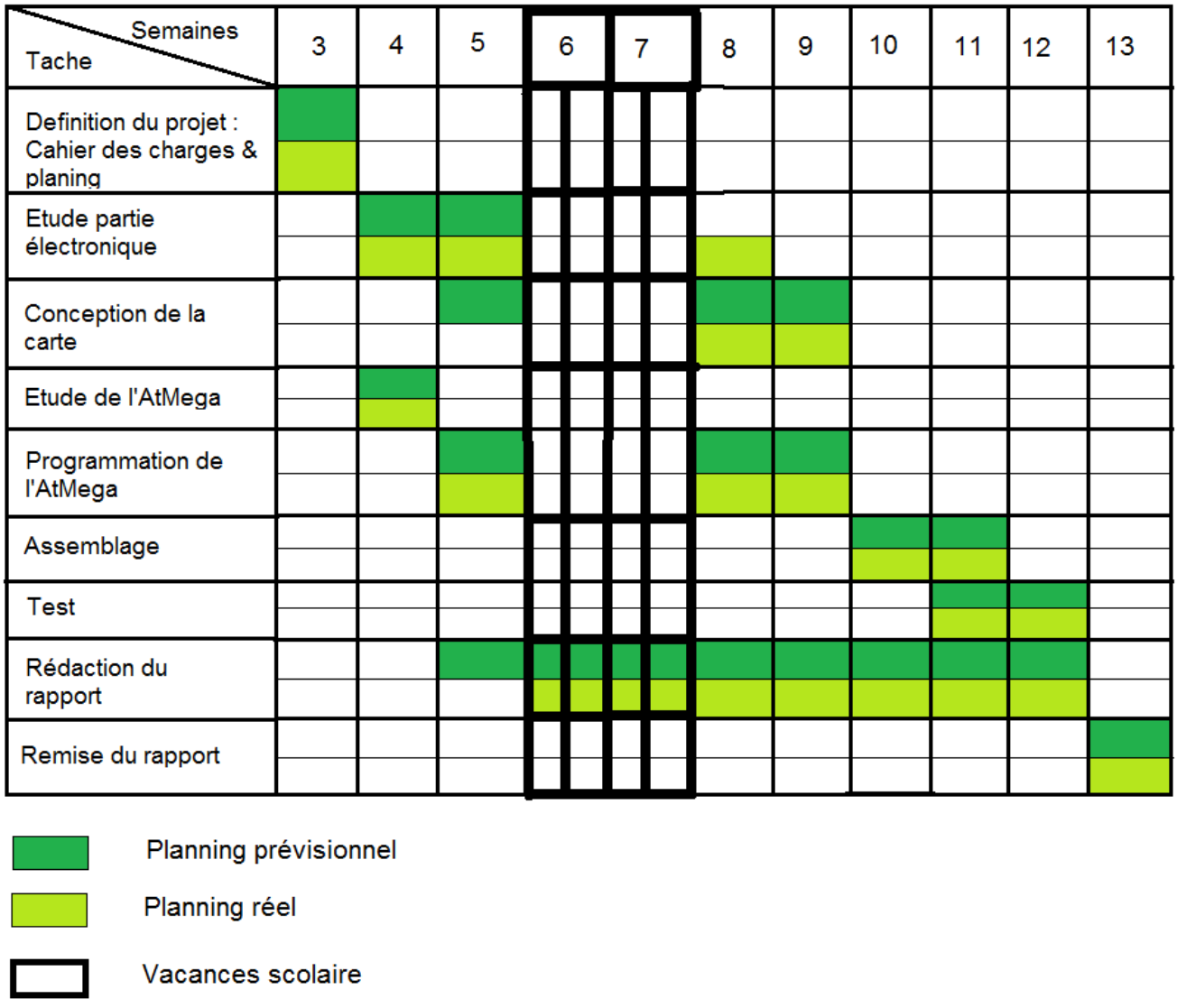


Figure 3 : Planning

3. Etude de la carte

3.1 Alimentation

Au départ on pensait à utiliser un régulateur linéaire (7805), mais Mr Lequeu nous à conseiller de prendre plutôt un régulateur à découpage. En effet les régulateurs linéaires implique l'utilisation d'un dissipateurs ce qui augmente le cout, de plus le rendement est mauvais comparé à un régulateur à découpage surtout si la tension d'entrée est grande devant la tension de sortie.

L'alimentation de la carte est donc réalisée à l'aide d'un régulateur à découpage de référence LM2574. Il fournit une tension constante de 5v quel que soit la tension appliquée en entrée (celle-ci doit néanmoins rester dans une certaine plage de tension, ici entre 7V et 60V).

La figure ci dessous montre le schéma de principe et le câblage d'un régulateur à découpage.

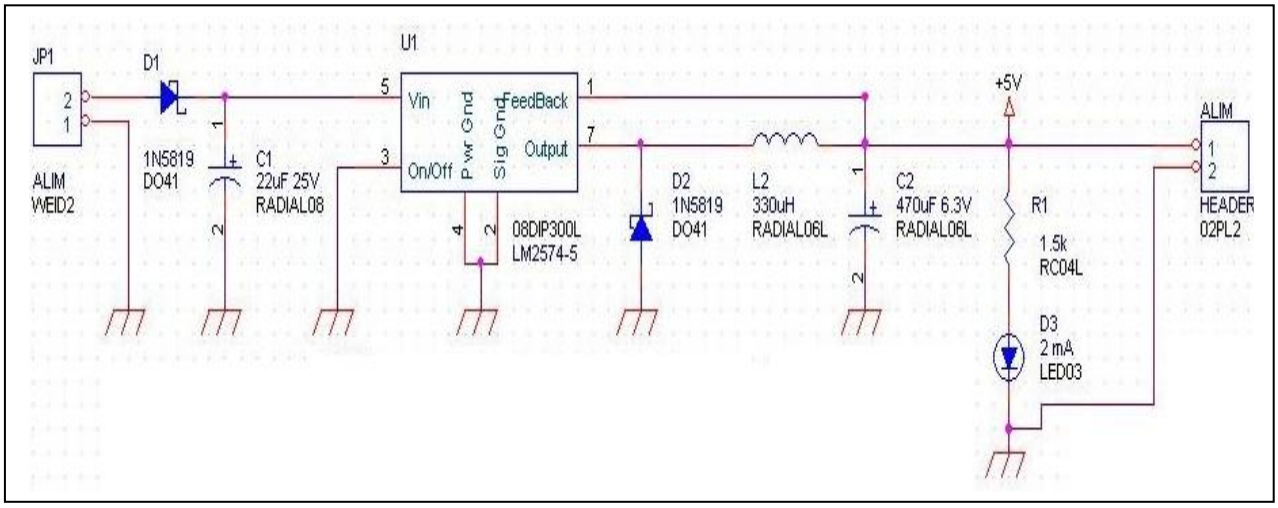


Figure 4 : Fonction alimentation

Rôle des composants :

Le condensateur C1 permet de limiter les variations de tension en entrée et donc permettre au régulateur de fonctionner correctement.
Le condensateur C2 permet de lisser la tension de sortie pour qu'elle soit constante.
La diode D2 protège le régulateur contre les surcharges dues à la décharge de la capacité C2.
La diode D1 est une diode de protection elle permet d'éviter toute détérioration en cas d'inversion de la tension d'entrée.

3.2 Capteur de température LM75

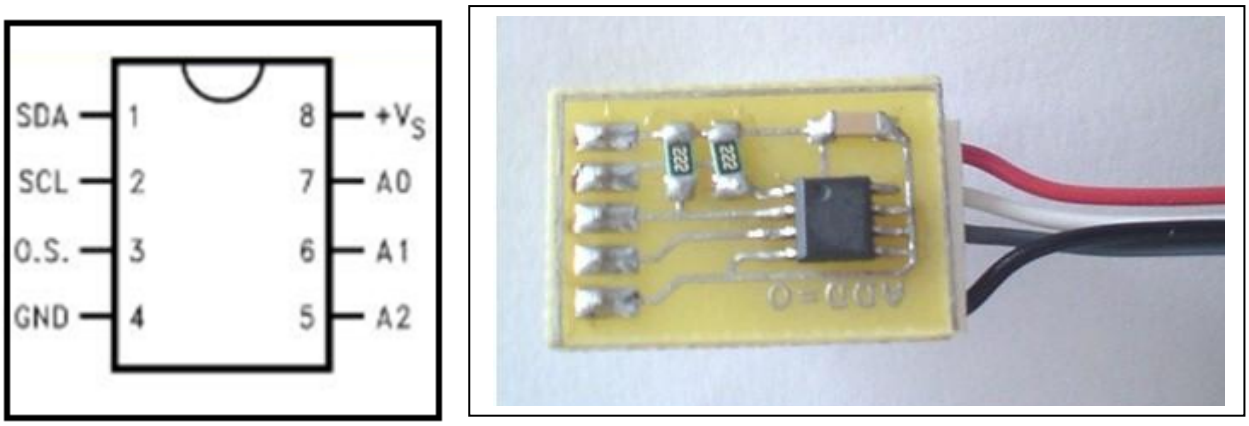


Figure 5 : Synoptique de LM75

Nous avons choisis le capteur de température LM75 car c'est un capteur numérique ce qui permettra de faciliter l'envoie des données vers l'AtMega, c'est-à-dire pas besoin d'effectuer une conversion analogique numérique dans le cas d'un capteur analogique.

Il possède 8 broches :

- A0, A1, A2 sont trois broches d'adresse, dans notre cas relié à la masse.
- Deux broches d'alimentation (Vs,GND).
- Deux broches pour le bus I2c (SDA, SDL) pour le relier avec le microcontrôleur (L'ATMéga8535). Dans notre cas SDA est relié au PORTA.0 et SDL au PORTA.1.
- La broche OS est une broche de détection, il ne faut pas débiter trop de courant à cette broche pour ne pas fausser la mesure de température.
- Ce capteur nous permet de mesurer la température entre -55°c et 125°C avec une erreur de 3°C seulement.

3.3 L'afficheur LCD

Pour pouvoir visualiser les informations, on utilise un afficheur LCD 4lignes sur 16 caractères. Il est branche en mode 4 bits. Seuls les 4 bits de poids forts (D4 a D7) de l'afficheur sont utilises pour transmettre les données et les lire. Les 4 bits de poids faible (D0 a D3) sont dans le vide. On a 7 pattes pour commander l'afficheur. Les données sont écrites ou lues.
Une impulsion positive doit être envoyé sur E pour indique que des données sont valides. Les pattes 15 et 16 sont présentes pour le rétro éclairage. On commande 7 broches E, R/W, RS, D4, D5, D6 et D7. On distingue 2 types d'envoi de signaux vers l'afficheur :

- pour exécuter une instruction, la broche RS est mise à la masse et les broches D7 à D0 définissent l'instruction à exécuter, puis E passe à 1,

- pour envoyer des données, il faut mettre la broche RS à 1 et E passe à 1

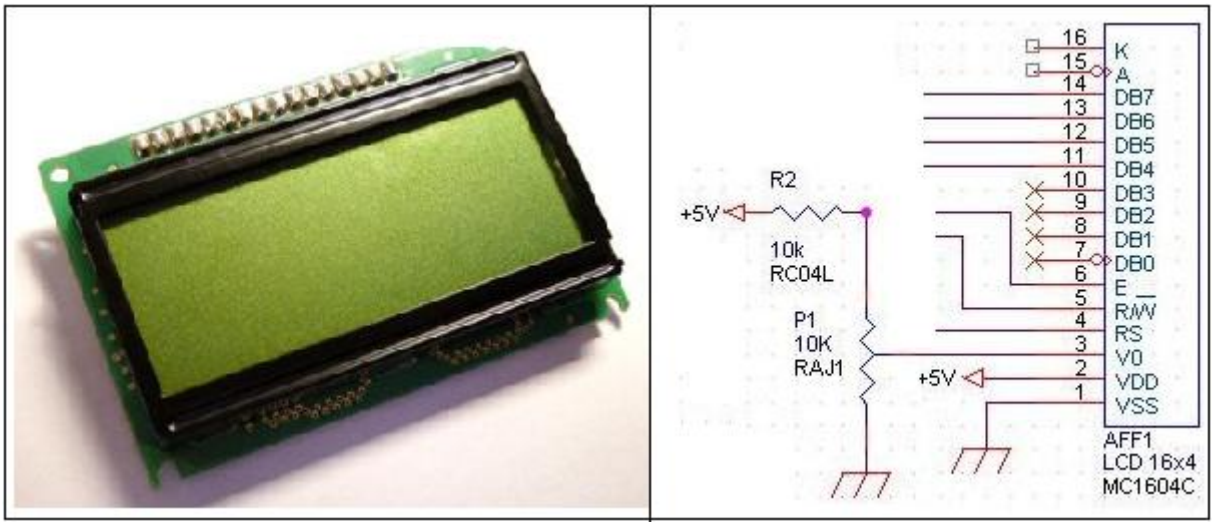


Figure 6 : Afficheur LCD

3.4 Microcontrôleur AtMega 8535

Le microcontrôleur utilisé pour la réalisation de notre carte est un ATMEGA 8535 de fabricant Atmel. C'est ce composant qui va accueillir le programme gérant les entrées-sorties afin d'afficher sur l'écran LCD les informations venant des capteurs ainsi que l'appui sur les boutons poussoirs. Il est alimenté en +5V à partir de la broche 10 (VCC). L'ATMega 8535 est un circuit intégré de 40 broches avec 4 ports (A, B, C, D) qui ont la particularité d'être utilisés comme entrés ou sorties. Il possède une capacité mémoire de 8 Koctets pour stocker un programme. De plus il possède trois « timers » servant à compter le temps ou les événements et la commande horloge est réalisé à l'aide d'un quartz (dans notre cas nous avons un quartz de 16Mhz) branché entre les broches XTAL1 et XTAL2 (voir ci-dessous). Le port D comprend des convertisseurs analogiques-numériques. Le PORTC est un port numérique relié à l'afficheur LCD.

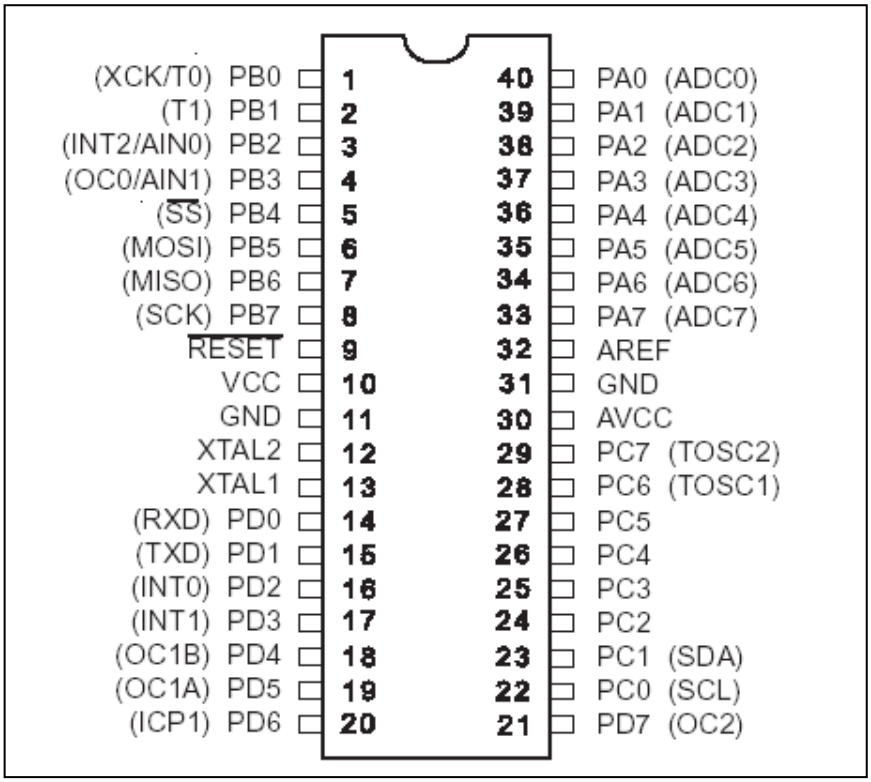


Figure 7 : AtMega 8535

Le Quartz : C'est un composant qui possède la propriété d'osciller à une fréquence stable lorsqu'il est électriquement stimulé. Ce sont les propriétés piézoélectriques du minéral de quartz qui lui permettent d'obtenir une fréquence d'oscillation très précise. Le quartz utilisé est cadencé à 16 MHz, il permet de remplacer l'horloge interne de l'AtMega qui est cadencée à une fréquence moins élevée.

4. Réalisation de la carte

4.1 Le logiciel ORCAD

Nous avons utilisé le logiciel ORCAD pour la réalisation du typon de la carte. Pour cela il faut passer par plusieurs étapes. Tout d'abord réaliser le schéma avec CAPTURE, une fois le schéma saisie il faut créer la Netlist. Pour cela il faut être sur que le schéma soit correct car la Netlist est le fichier que l'on va charger sous Orcad Layout pour faire le typon en liaison avec Orcad Capture. En chargeant la Netlist, Orcad Layout va chercher les empreintes (FootPrint) dans les libraires. Maintenant qu'il y a les liaisons il faut tout d'abord limiter la taille de la carte. Les composants ne peuvent pas être positionnées au hasard, il faut respecter le cahier des charges tout en réfléchissant au positionnement des composants afin de limiter le nombre de connections.

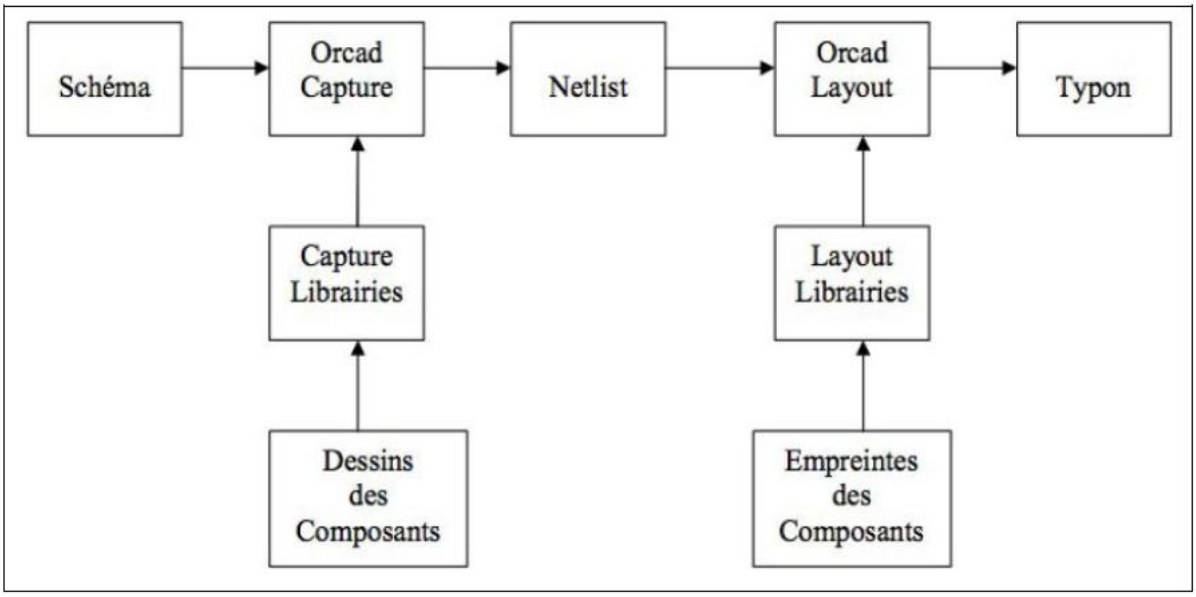
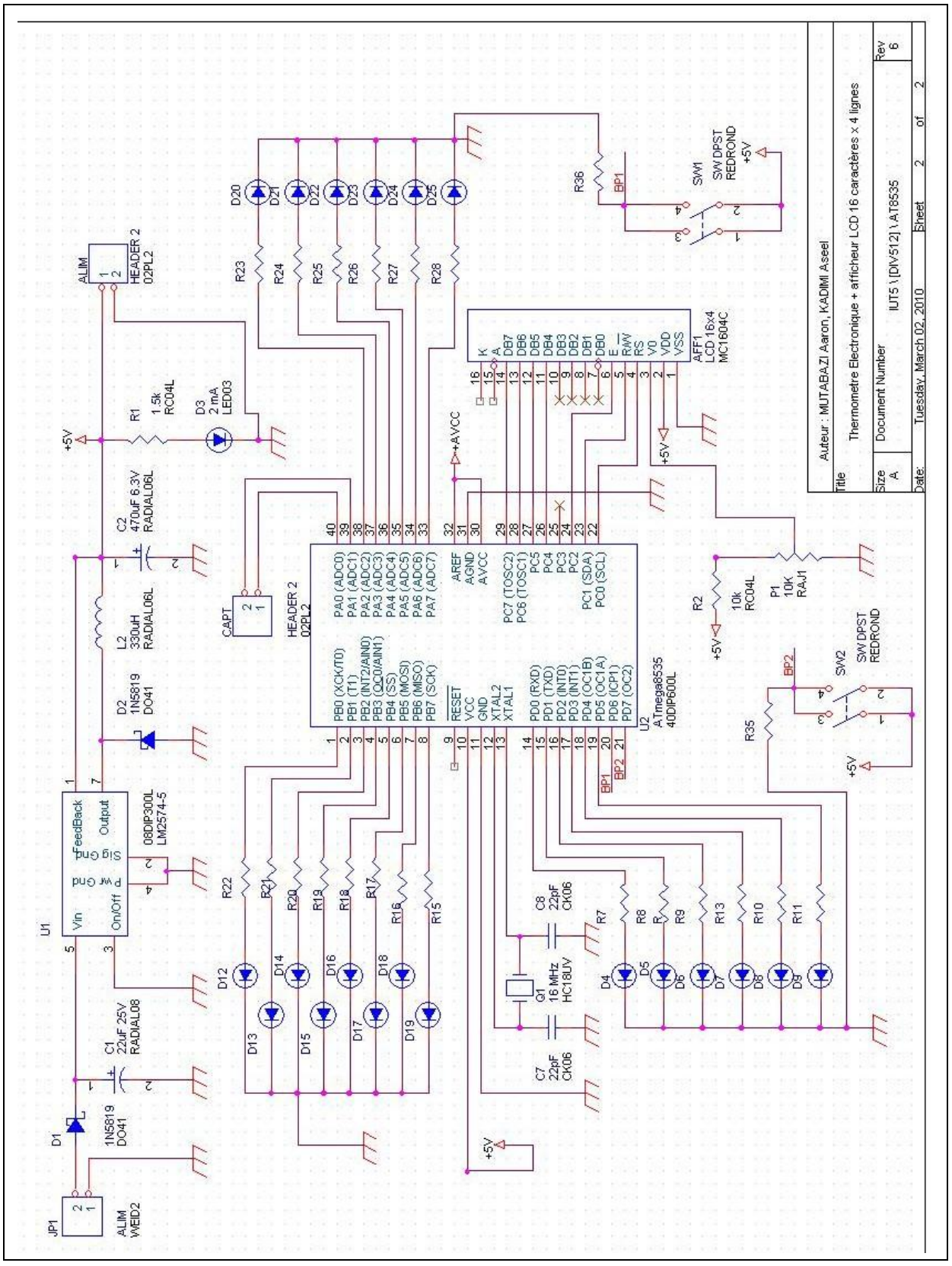


Figure 8 : Organisation de Orcad

4.2 Schéma structurel



4.3 Typons

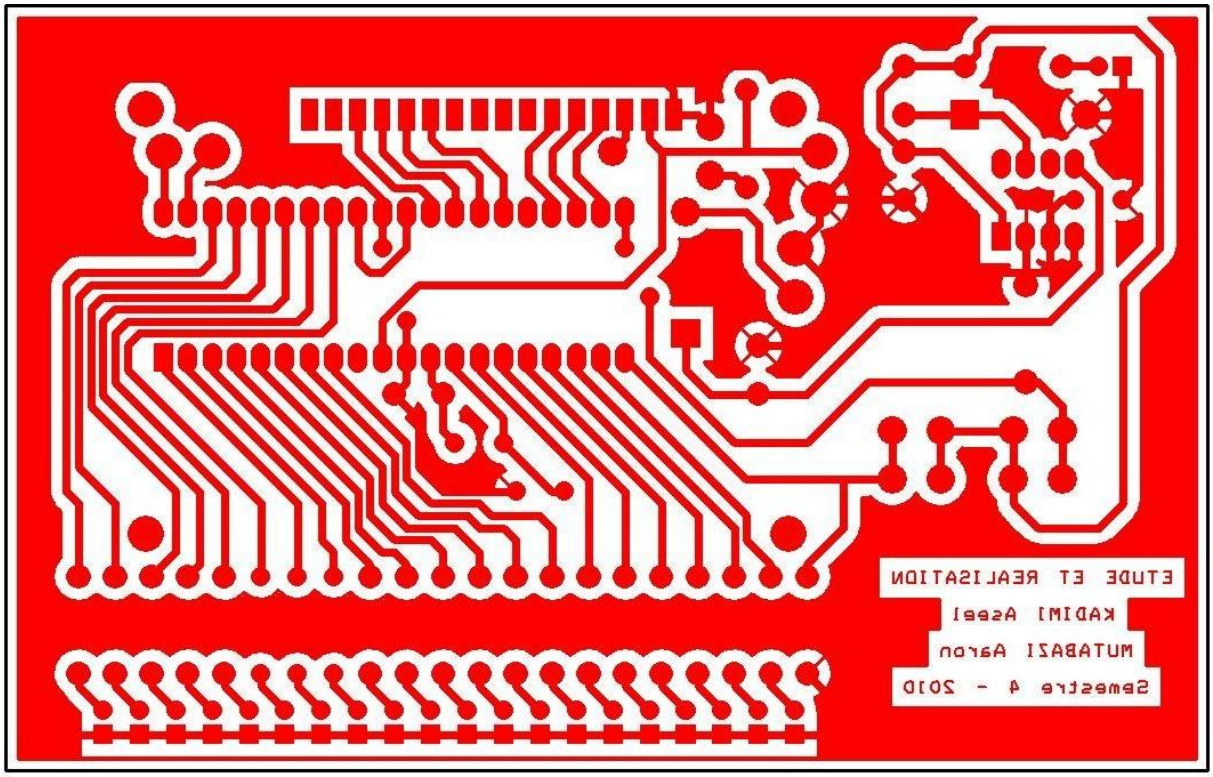


Figure 9 : Typon coté cuivre

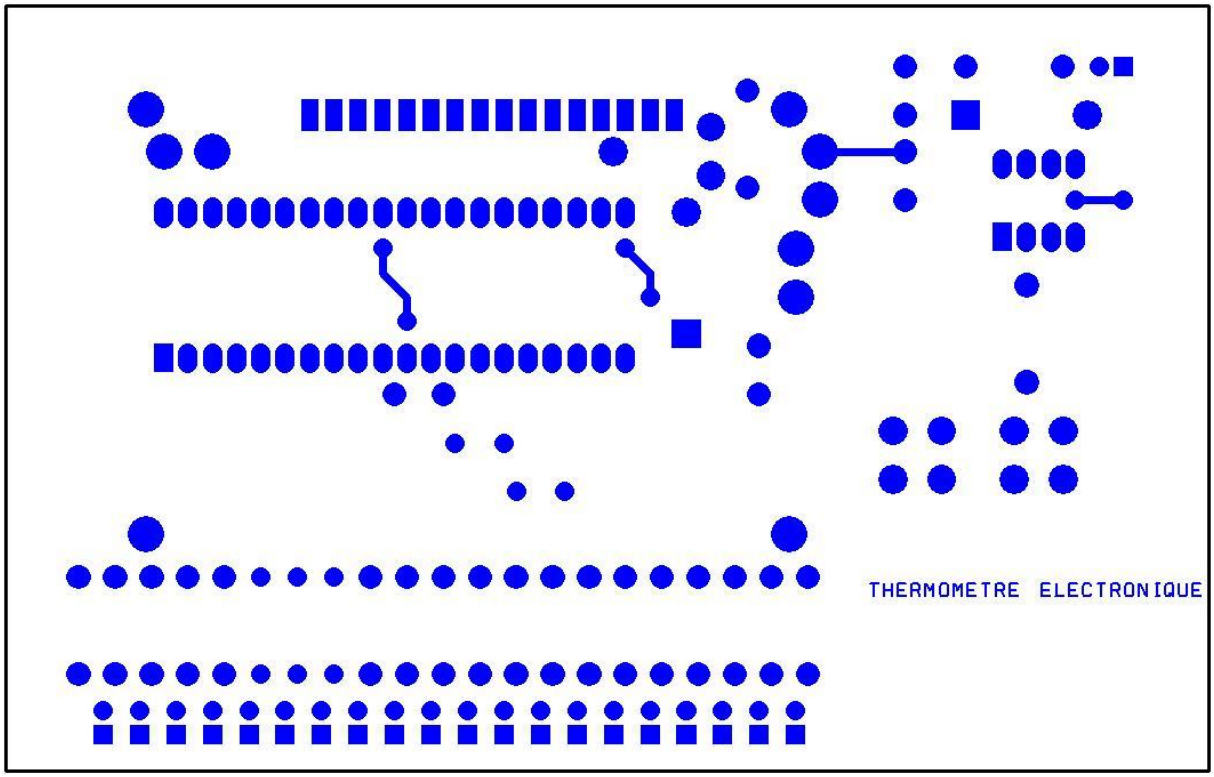
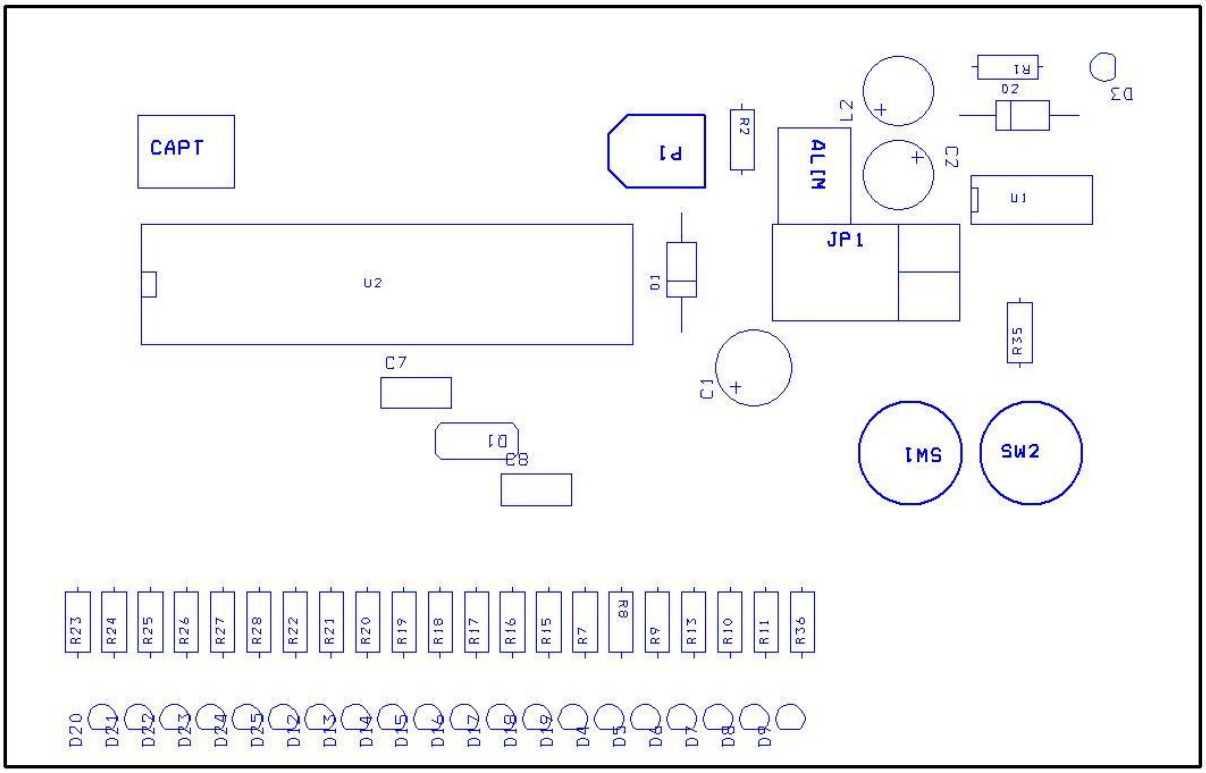
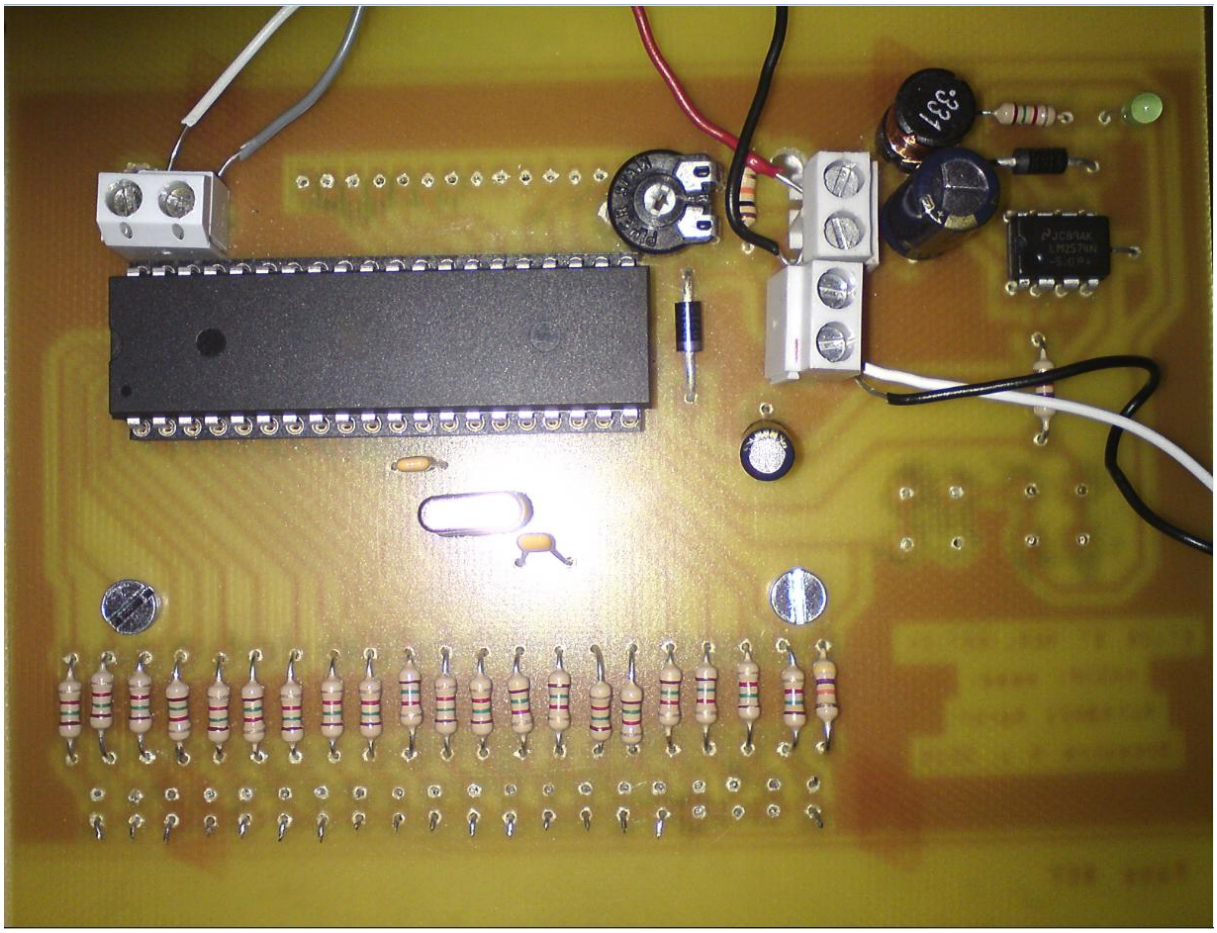


Figure 10 : Typon coté composant

4.4 Schéma d'implantation des composants





4.5 Nomenclature

Nom	Référence	Quantité	Prix(€)
AtMega 8535	U2	1	5.73
Support AtMega		1	2.03
LM2574-5	U1	1	3.11
Bouton Poussoir	SW	2	1.4
Quartz	X1	1	1.93
Resistance	R	24	0.24
Diode	D	21	1.26
Condensateur quartz	C	2	0.2
Potentiomètre	P	1	0.2
Bornier	B	2	0.6
Afficheur LCD		1	15.62
Diode de protection	D	2	0.9
Inductance	L	1	0.8

Total : 34.82€

5. Programmation de la carte

5.1 Logiciel de programmation (Code Vision AVR)

Pour la programmation de l'ATMEGA 8535 nous avons utilisé le logiciel Code Vision AVR. Ce logiciel de compilation est vraiment simple d'utilisation, c'est un compilateur C classique, cependant il est réellement orienté dans la programmation de microprocesseur et microcontrôleur. En effet, avant de débuter la programmation, le logiciel demande le type de microprocesseur utilisé et sa fréquence de fonctionnement. Ensuite on doit configurer les différentes broches des ports A, B, C, D en entrées ou en sorties.

CONFIGURATION DES PORTS

Nous pouvons ainsi commencer à configurer grâce à code Wizard notre microcontrôleur, il faut également lui donner la valeur du quartz qui est de 16MHz dans notre cas. Il faut ensuite configurer les ports du microcontrôleur, configurable soit en entrés soit en sorties.

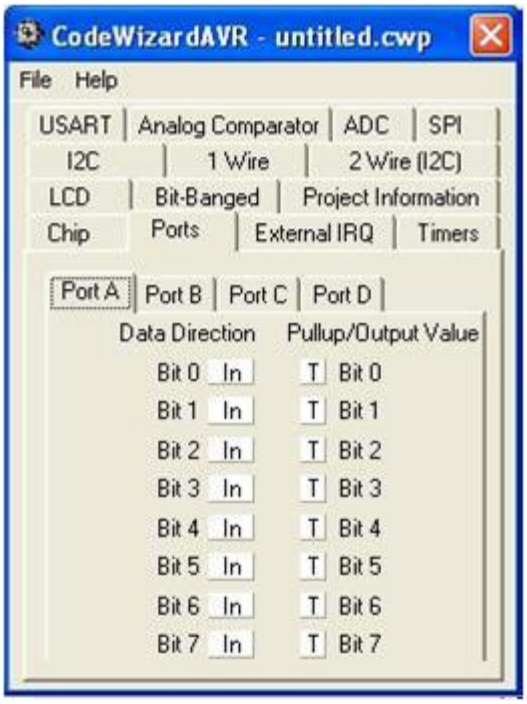


Figure 11 : Fenêtre configuration

Pour ce faire il faut aller dans l'onglet « Ports », nous devons choisir comment utiliser les ports du microcontrôleur en cliquant sur l'onglet correspondant au port et en cliquant sur « In » pour mettre le bit correspondant en entrée et « Out » pour le mettre en sortie.

On retrouvera la configuration des ports dans le programme principale sous forme d'attribution tel que:

- PORTA=0x00; pour déclarer le port A en entrée.
- PORTA=0xFF; pour qu'il soit en sortie.

Le tableau ci-dessous résume l'affectation des bits des ports.

Rang	7	6	5	4	3	2	1	0
Port A	D25	D24	D23	D22	D21	D20	SCL	SDA
Port B	D19	D18	D17	D16	D15	D14	D13	D12
Port C	LCD	-	-	-	-	-	-	-
Port D	BP2	BP1	D9	D8	D7	D6	D5	D4

Tableau 1 : Affectation des bits

Le tableau ci-dessous résume la configuration des bits soit en entrée, soit en sortie,

Rang	7	6	5	4	3	2	1	0
Port A	S	S	S	S	S	S	E	E
Port B	S	S	S	S	S	S	S	S
Port C	S	S	S	S	S	S	S	S
Port D	E	E	S	S	S	S	S	S

Tableau 2 : Configuration des E/S

E : Entrée
S : Sortie

Pour configurer un port nous disposons d'un registre 8 bits de direction : DDRx (x pour A, B, C ou D). Chaque bit peut être configuré en entrée (0) ou en sorie (1).
Ce qui donne :

Pour le PORTA : DDRA = 0xFC (1111 1100)
Pour le PORTB : DDRB = 0xFF (1111 1111)
Pour le PORTC : DDRC = 0xFF (1111 1111)
Pour le PORTD : DDRD = 0xCF (0011 1111)

La configuration des ports fait partie de la fonction BrdInit( ) que l'on appelle au début du programme et est de la forme suivant :

```
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTA=0x00;
DDRA=0xFF;
```

```
// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0x00;
DDRB=0x1F;
```

```
// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xFF;
```

```
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;
```

CONFIGURATION DE L'AFFICHEUR LCD

Pour configurer l'écran LCD on se place dans l'onglet « LCD ».
On doit simplement sélectionner le port sur lequel l'écran est relié dans notre cas c'est sur le Port C.

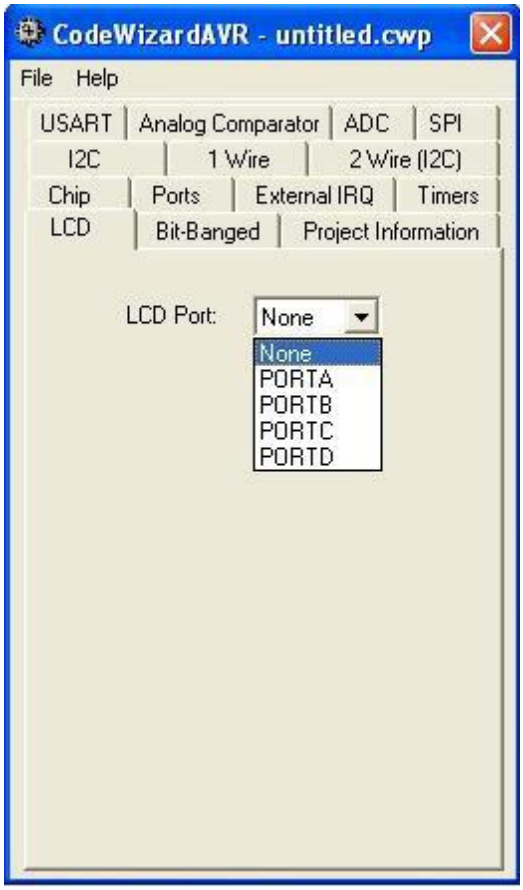


Figure 12 : Fenêtre de configuration LCD

La partie configuration est ainsi terminée. Pour commencer l'écriture du programme, on doit choisir dans le menu « File », « Generate,Save and Exit ».

A la suite de ces étapes on peut commencer à programmer le microcontrôleur. Cela se passe sur une fenêtre du type suivant :

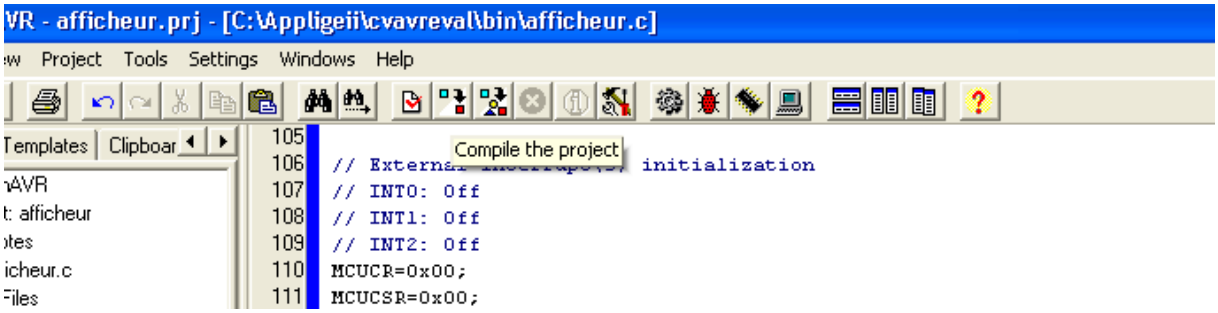
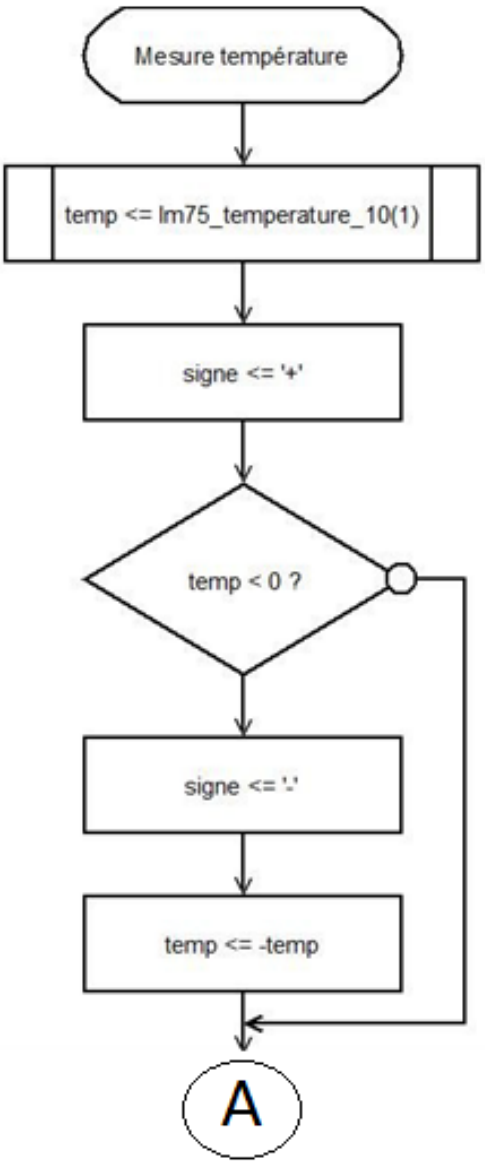


Figure 13 : Fenêtre de programmation

5.2 Acquisition de la température

On à tout d'abord commencé à programmer pour recevoir la température. Pour cela on à simplement chargé un programme destiné au capteur LM75 via le site de M. Thierry LEQUEU.

Voici l'ordinogramme de la fonction qui permet de mesurer la température.



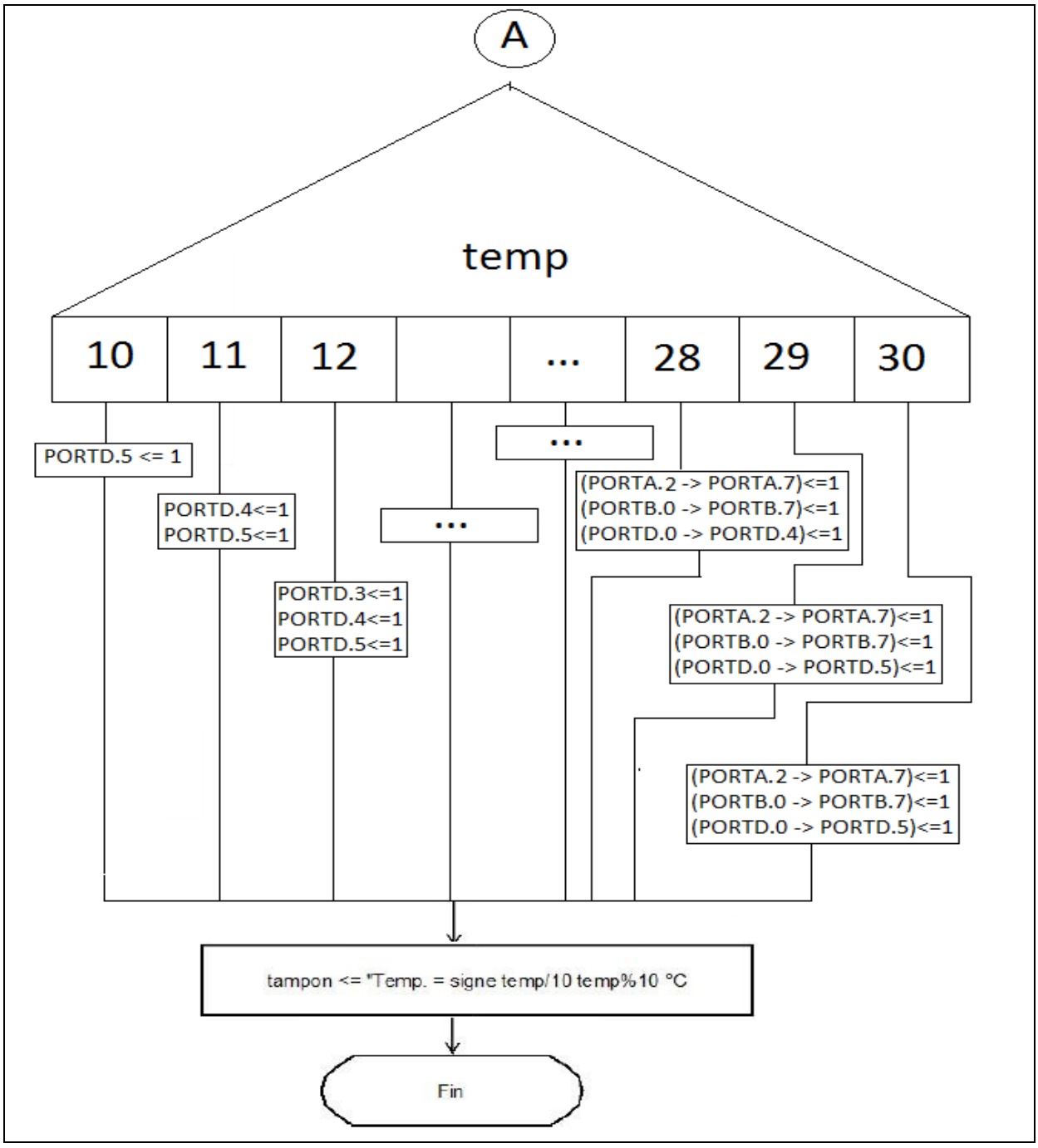
On a donc utilisé la fonction suivante « lm75_temperature_10 () ».
Celle ci ne demande qu'un paramètre entre les parenthèses, l'adresse où se situe le capteur. Dans notre cas c'est l'adresse 0 car la broche 1(DATA) du capteur est relié à l'adresse 0 du PortA du 8535.

On initialise ensuite une polarité, donc au début positive et on vérifie si la température est inférieur de 0°C si oui on indique que la température est négatif sinon fin de la boucle if.

Voici à quoi ressemble le programme en langage C :

```
lm75_init(0,0,60,0);

while (1)
      {
      temp=lm75_temperature_10(0);
      signe='+';
      if (temp<0)
        {
        signe='-';
        temp=-temp;
        };
```



Nous utilisons un sélecteur de choix (Switch) pour simplifier le programme. La variable du sélecteur correspond à temp. Ensuite chaque case correspond à la valeur de temp, exemple pour la case 10, la température (temp) est égale à 10°C, le programme va faire un traitement, ici il va placer la broche 5 du PortD à +5v (PORTD.5<=1) ce qui va allumer une LED sur la carte. Et ainsi pour tous les autres cas.

Nous ne l'avons pas précisé sur l'ordinogramme, mais il ne faut pas oublier d'affecter la valeur 0 aux numéros de ports non utilisés. Exemple pour le cas 10°C on à mis PORTD.5 =1, donc tous les autres ports ayants des LED doivent être à 0. (PORTD.4=PORTD.3=……..=PORTA.7=0)

Voir en annexe pour le programme codé en C.

5.3 Réglage de la date

REGLAGE D'UNE INTERRUPTION INTERN

Pour que le microcontrôleur puisse gérer l'affichage de l'heure, il faut qu'il contienne des variables qui s'incrémentent en temps réels tels que les secondes ensuite suivront les minutes et heures. Ces variables seront déclarés en tant qu'entier par :
int seconde, minute, heure ;
Ce qui permettra au microcontrôleur d'allouer un espace mémoire pour ces 3 variables. Chaque seconde, la variable doit s'incrémenter. Cela est évidemment possible grâce à un timer, on va donc créer une fonction d'interruption interne
Le rôle du timer est de compter de 0 à une valeur définie, avec une fréquence choisie.
Voici le procédé afin de régler le timer :

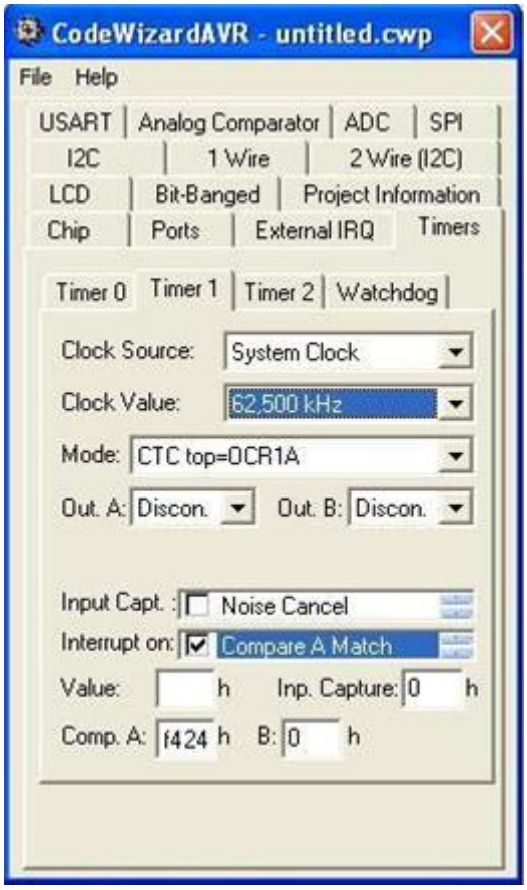


Figure 14 : config interruption interne

On peut remarquer que il est possible d'utiliser jusqu'à 3 timer, nous utiliserons seulement le timer 1 car il compte sur 16 bits. Il faut d'abord choisir la source de l'horloge du timer, interne ou externe, pour nous interne: « system clock ». Il faut ensuite choisir la fréquence de cette horloge, parmi les valeurs 16MHz, 2MHz, 250KHz, 62,5KHz, et 15,625 KHz. On prendra 62.5KHz. Le timer 1 compte sur deux octets donc 16 bits, qui font 2^16= 65536.
Explication : On veut connaitre le temps que met le microcontrôleur pour entrer dans la fonction interruption.
En prenant la fréquence 16MHz du quartz, on trouve un temps trop petit.

$$\frac{1}{16.10\text{^}6} * 2^{16} = 0.004096\ \text{s}$$

On va essayer avec 62.5KHz

$$\frac{1}{62500} * 2^{16} = 1.048576\ \text{s}$$

On trouve une valeur légèrement supérieur à 1s.
Maintenant pour avoir exactement 1 seconde il faudrait que le timer compte jusqu'à 62500.
En effet :

$$\frac{1}{62500} * \text{X} = 1\ \text{s}$$

Donc X = 62500 et cette valeur devra être codé en hexadécimale, ce qui donne 62500 = 0xF424

Détaille de la conversion décimale vers hexadécimal.

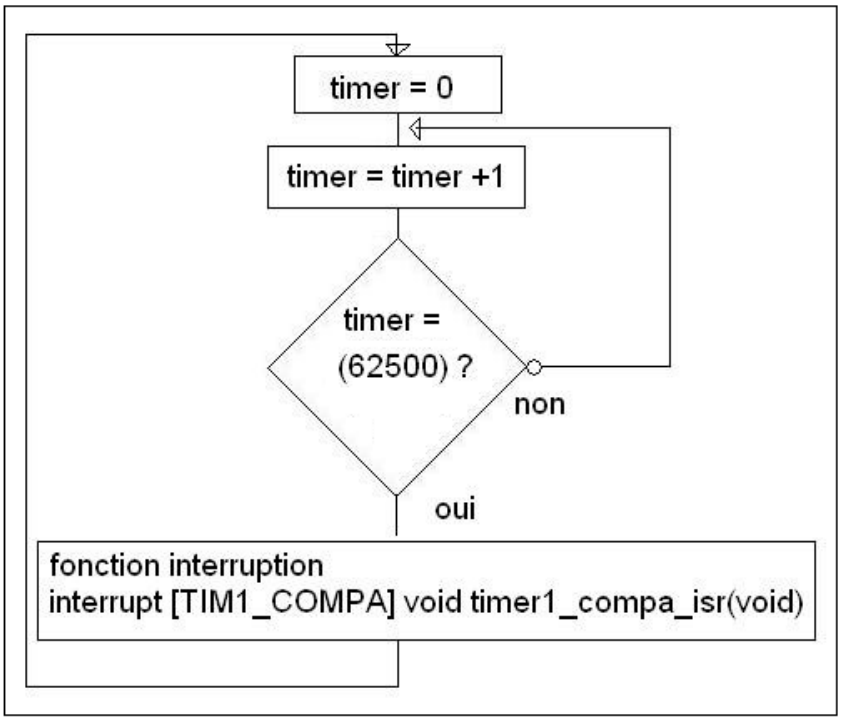
62500 = 2^15+2^14+2^13+2^12+2^10+2^5+2^2
2^15+2^14+2^13+2^12+2^10+2^5+2^2 = 1111 0100 0010 0100
1111 0100 0010 0100 = (**F***16^3) + (**4***16^2) + (**2***16^1) + (**4***16^0)
(F*16^3) + (4*16^2) + (2*16^1) + (4*16^0) = 0XF424

Il faut donc remplir la case « Value avec la valeur F424.

Ces quelques réglages ont ajoutés au programme les lignes de code suivantes :

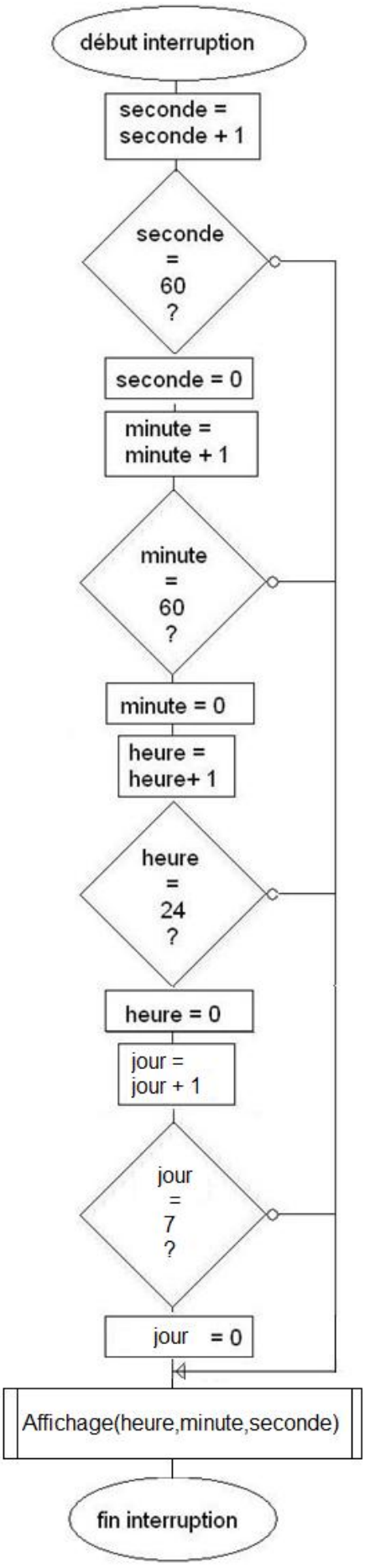
```
33  // Timer 1 output compare A interrupt service routine
34  interrupt [TIM1_COMPA] void timer1_compa_isr(void)
35  {
36  // Place your code here
37
```

C'est ici que le microcontrôleur ira à chaque fin de comparaison avec la valeur 62500. Voyons avec l'ordinogramme ci-dessous comment le timer fonctionne.



Au début de la fonction, on initialise le timer à 0, ensuite à l'aide de la boucle if on regarde si le timer à atteint le nombre 62500, si oui le microcontrôleur exécute la fonction interruption sinon retour à l'étape précédente.

Maintenant regardons plus en détaille ce que fait le microcontrôleur lorsque le timer = 62500.



Lorsque le timer est égal à 62500 on entre donc dans la fonction d'interruption et la variable *seconde* s'incrémentée

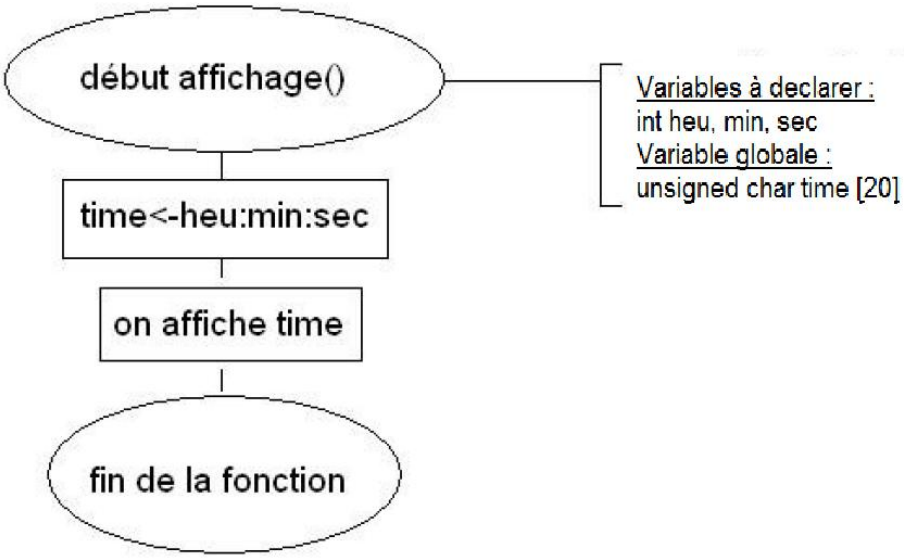
On teste si *seconde* est égale à 60, si oui on met *seconde* à 0 et on incrémente *minute* sinon fin de l'interruption

On teste si *minute* est égale à 60, si oui *minute* est mise à 0 et on incrémente *heure* sinon fin de l'interruption

On teste si *heure* est égale à 24, si oui *heure* est mise à 0 et on incrémente jour.

Ensuite on teste si jour est égale à 7, si oui jour est mise à 0 sinon fin de l'interruption.

Nous avons créé une fonction affichage qui prend en paramètre 3 entiers et qui seront stockés dans un tableau crée en global.

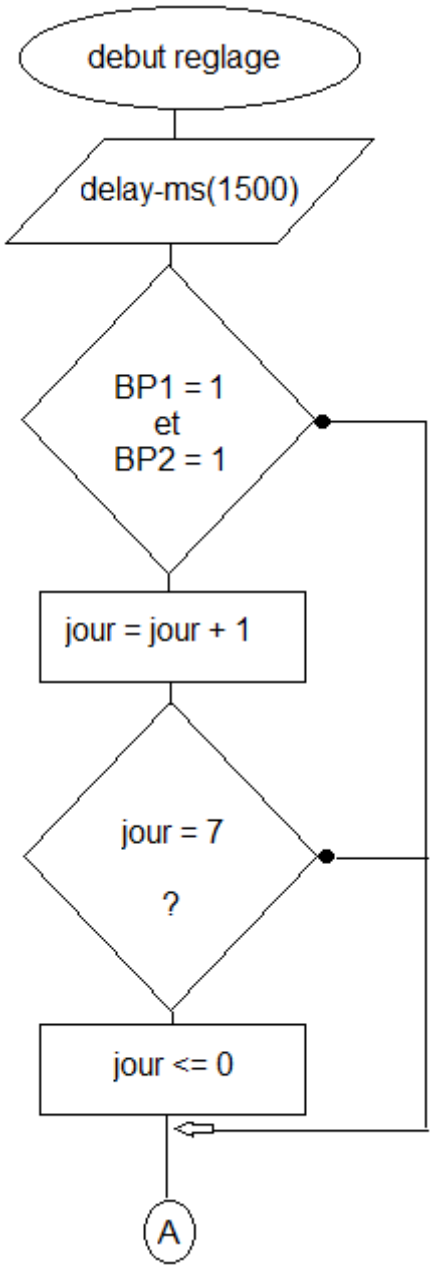



```
void affichage (int heu, int min, int sec)
 {
 lcd_gotoxy(0,1);
 sprintf(time,"Heure =
%2d:%2d:%2d",heu,min,sec);
 lcd_puts(time);
 }
```

En langage C, on utilise la fonction sprintf(), cette fonction rempli un tableau de caractères. Le premier paramètre de cette fonction est le nom du tableau dans lequel on va stocker les variables, ici c'est time.

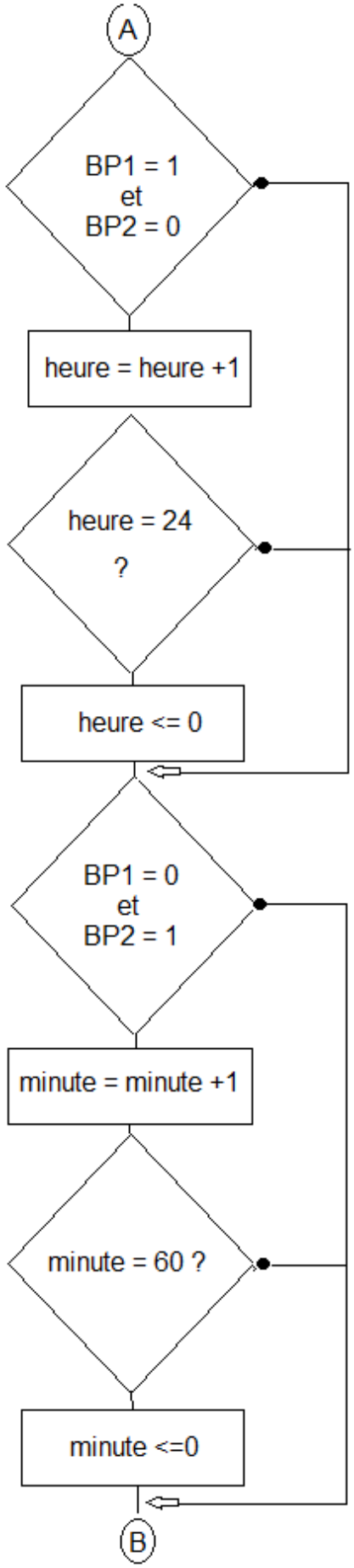
L'affichage est réalisée grâce à la fonction lcd_puts(), déclarée dans la librairie <lcd.h>, et qui affiche sur l'écran lcd, le contenu de la variable *time*.

Maintenant que l'heure est disponible il devient nécessaire de pouvoir la régler.



On teste si BP1 et BP2 sont égale à 1, c'est-à-dire appuyé. Si oui *jour* est incrémenté sinon on passe à la prochaine vérification.

On teste si jour est égale à 7, donc « dimanche », si c'est le cas on met *jour* à zéro donc lundi. Sinon on passe à la prochaine vérification.



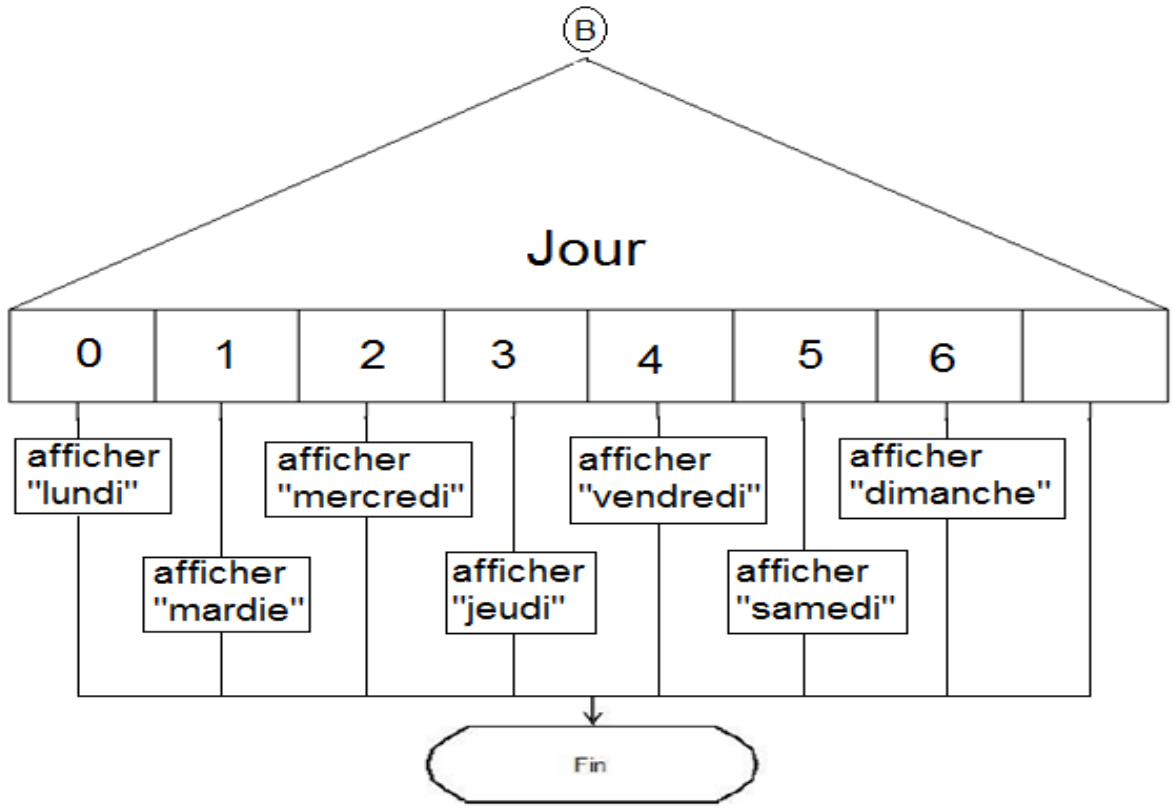
On teste si BP1 est égale à 1 et BP2 égale à 0.
Si oui *heure* est incrémentée sinon on passe à la prochaine vérification.

On teste si heure est égale à 24. SI oui on met heure à 0 sinon on passe à la prochaine vérification.

On teste si BP1 est égale à 0 et BP2 égale à 1.
Si oui *minute* est incrémentée sinon on passe à la prochaine vérification.

On teste si *minute* est égale à 60. SI oui on met *minute* à 0 sinon on passe à la prochaine vérification.

Suite de l'ordinogramme :



On utilise encore un sélecteur de choix (switch), le principe est très simple. La variable du sélecteur correspond à jour et chaque case correspond à un jour de la semaine.
Par exemple si jour est égale à 0, il faudra afficher « lundi », si jour est égale à 7, il faudra afficher « dimanche ».

Programme codé en C.

```
switch (jour )
{
case 0 :
        lcd_gotoxy(0,2);
        lcd_putsf("lundi   ");
break ;
case 1 :
        lcd_gotoxy(0,2);
        lcd_putsf("mardi   ");
break ;
case 2 :
        lcd_gotoxy(0,2);
        lcd_putsf("mercredi   ");
break ;
```

```
case 3 :
        lcd_gotoxy(0,2);
        lcd_putsf("jeudi   ");
break ;
case 4 :
        lcd_gotoxy(0,2);
        lcd_putsf("vendredi   ");
break ;
case 5 :
        lcd_gotoxy(0,2);
        lcd_putsf("samedi  ");
break ;
case 6 :
        lcd_gotoxy(0,2);
        lcd_putsf("dimanche");
break ;
}
```

6. Conclusion

Le projet choisi au quatrième semestre dans la cadre de l'étude et réalisation fut pour nous très enrichissant. Nous avons choisi de réaliser un thermomètre numérique, ce projet nous à permis d'acquérir diverses connaissances sur de multiples domaines notamment en électronique (choix composants, réalisation de la carte, dépannage…) et en informatique (langage C, ordinogramme, logiciel Code Vision AVR…). Nous avons aussi appris à réaliser un cahier des charges et un planning qui au final à été respecté car le projet est fonctionnel. Le faite de partager les taches entre nous nous à permis de voir l'importance du travaille en équipe. Nous avons aussi été confrontés à des problèmes techniques, nous montrant qu'il faut savoir constamment s'adapter et adapter notre Projet.

Enfin, nous avons été satisfaits de voir un produit fini respectant le cahier des charges. Nous pensons même que des améliorations peuvent être apportés à notre projet (comme un connecteur branché en parallèle permettant ainsi de transférer directement un programme vers la carte, approfondir la programmation de la date : année, mois, jours fériés …, ce dernier amélioration nécessitera peut être un afficheur LCD de taille plus important).

Au final nous sommes convaincus que ces heures d'études et réalisation nous ont apportés autonomie et rigueur, qualités indispensable pour une insertion future et pour le succès du stage.

Tables des illustrations :

Bibliographie

http://www.thierry-lequeu.fr/
http://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal
http://www.atmel.com/

ANNEXE

PROGRAMME DATE

```
#include <mega8535.h>
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#include<stdio.h>
#include <delay.h>
#define BP1 PIND.7
#define BP2 PIND.6

//déclarration des variables
unsigned char time[20];
int heure, minute, seconde,jour;
//déclaration des fonctions
void affichage (int heu, int min, int sec);
void reglage () ;

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
 {
 seconde=seconde+1;
 if(seconde==60)
          { seconde=0;
          minute=minute+1;
          if(minute ==60)
                    {
                    minute=0;
                    heure=heure+1;
                    if(heure==24)
                              {
                              heure=0;
                              jour=jour+1 ;
                              if(jour==7)
                                        {
                                        jour=0 ;
                                        }
                              }
                    }
          }
 }

void affichage (int heu, int min, int sec)
 {
 lcd_gotoxy(0,1);
 sprintf(time,"Heure = %2d:%2d:%2d",heu,min,sec);
 lcd_puts(time);
 }




void reglage ()
{
delay_ms(1500);

 if (BP2==0&&BP1==1)
          {
          minute = minute +1 ;
```

```
        if(minute==60)
                {
                minute=0 ;
                heure=heure+1;
                }
        delay_ms(500);
        }
if (BP1==0&&BP2==1)
        {
        heure=heure+1;
        if(heure==24)
                {
                heure=0;
                jour=jour+1;
                }
        delay_ms(500);
        }
if (BP2==0&&BP1==0)
        {
        jour=jour+1;
        if(jour==7)
                {
                jour=0;
                }
    delay_ms(500);
    }
switch (jour )
{
case 0 :
        lcd_gotoxy(0,2);
        lcd_putsf("lundi   ");
break ;

case 1 :
        lcd_gotoxy(0,2);
        lcd_putsf("mardi   ");
break ;
case 2 :
        lcd_gotoxy(0,2);
        lcd_putsf("mercredi   ");
break ;
case 3 :
        lcd_gotoxy(0,2);
        lcd_putsf("jeudi   ");
break ;
case 4 :
        lcd_gotoxy(0,2);
        lcd_putsf("vendredi   ");
break ;
case 5 :
        lcd_gotoxy(0,2);
        lcd_putsf("samedi  ");
break ;
case 6 :
        lcd_gotoxy(0,2);
        lcd_putsf("dimanche");
break ;
}
}
```

```
// Declare your global variables here
```

```
void main(void)
{
// Declare your local variables here
PORTA=0x00;
DDRA=0x00;
PORTB=0x00;
DDRB=0x00;
PORTC=0x00;
DDRC=0x00;
PORTD=0x00;
DDRD=0x00;

TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

TCCR1A=0x00;
TCCR1B=0x0C;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0xF4;
OCR1AL=0x24;
OCR1BH=0x00;
OCR1BL=0x00;

ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

MCUCR=0x00;
MCUCSR=0x00;
TIMSK=0x10;

ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")

 /* switch to writing in Display RAM */
lcd_gotoxy(0,0);
lcd_putsf("Projet AARON");

while (1)
      {

      reglage () ;

          affichage(heure,minute,seconde);
          }
}
```