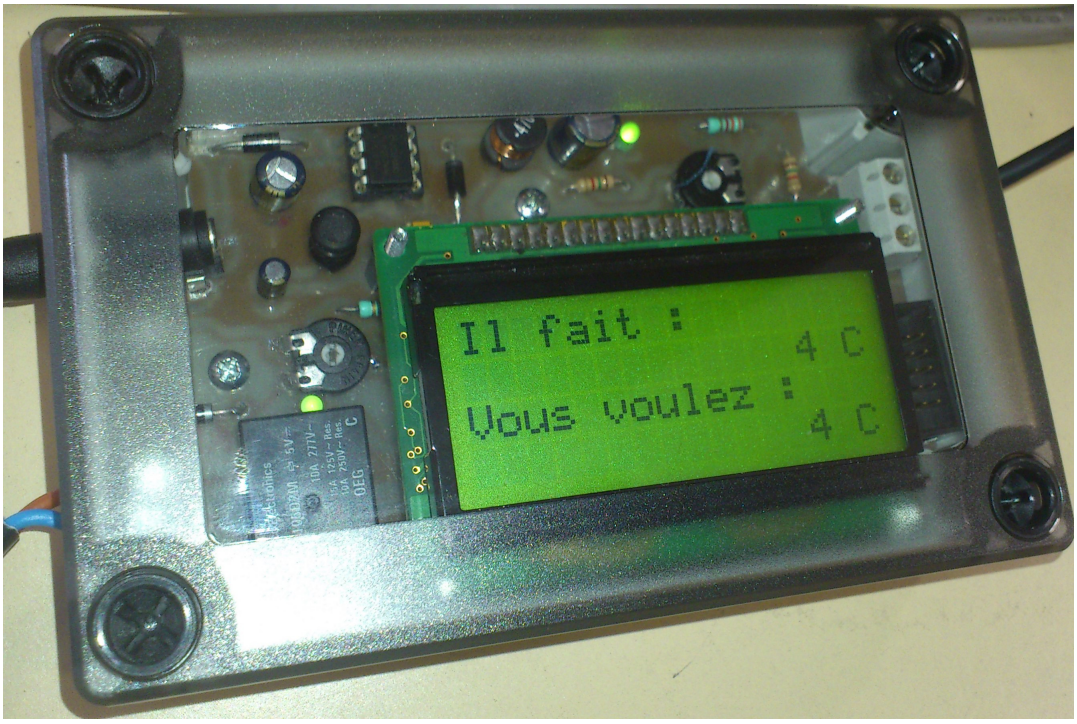
Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle



Réalisation d'un régulateur de température pour un réfrigérateur

Études et réalisations

GOUJON Benoît
CHAUDRON Bastien
2ème année groupe P1
Promotion 2011-2013

Enseignants :
Thierry LEQUEU
Sofi RODIER

Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Réalisation d'un régulateur de température pour un réfrigérateur

Études et réalisations

GOUJON Benoît
CHAUDRON Bastien
2ème année groupe P1
Promotion 2011-2013

Enseignants :
Thierry LEQUEU
Sofi RODIER

Sommaire

Introduction

Etant l'appareil le plus commun dans tous les foyers, le principe du réfrigérateur est apparu en 1876 inventé par Carl Von Linde, mais son utilisation ne débuta véritablement en Europe que vers 1935.

Avant sa création, il était plus courant d'utiliser le froid[1] naturel pour conserver des aliments. Plus communément appelé «frigo», le réfrigérateur est l'abrégé de la marque américaine frigidaire déposée en 1922. Ayant déjà réalisé un premier projet durant le semestre 3, il nous permettait seulement de garder une température constante fixée dans un programme[2] en langage C. Aujourd'hui, nous avons choisi d'améliorer le projet en lui ajoutant un afficheur LCD ainsi qu'un potentiomètre permettant à l'utilisateur de régler la température qu'il souhaite avoir dans son réfrigérateur. La problématique sera donc: Comment allons-nous procéder pour acquérir l'information venant du potentiomètre ainsi qu'afficher la température présente dans le réfrigérateur et celle désirée par l'utilisateur.

Nous présenterons dans la première partie le projet précédemment mené, ses différentes fonctions ainsi que les problèmes que nous avons rencontrés; dans la seconde partie nous traiterons des nouvelles fonctionnalités du projet, des solutions trouvées et enfin des problèmes rencontrés.

[3]

1 Voir définition p 19
2 Voir définition p 19
3 Voir annexe n°7 : Schéma fonctionnel de niveau 2

1. Le précédent projet

1.1. Les différentes fonctions

Le projet du semestre 3 a consisté à réaliser un projet dont la fonction principale a été de réaliser le régulateur[4] de température[5] pour un réfrigérateur[6]. Le projet était donc de constituer de plusieurs fonctions. Elles sont au nombre de trois :

- Une partie consistant à acquérir la température en provenance d'un capteur LM35[7] (cf illustration 1).

- Une partie de calcul et de comparaison par rapport à un seuil prédéterminé dans un programme en langage C stocké dans un microcontrôleur[8] ATtiny 13 (cf illustration 2).

- Une partie de commande avec l'activation ou non d'un relais 230V[9] (cf illustration 3).

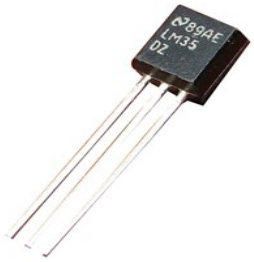


Illustration 1: Capteur de température LM35 (source internet)

Illustration 2: Microcontrôleur ATtiny 13 de la famille Atmel (source internet)

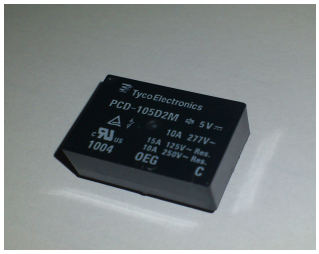


Illustration 3: Relais PCD-105-D2M (illustration personnelle)

4 Voir définition p 19
5 Voir définition p 19
6 Voir définition p 19
7 Voir annexe n°4 p 28 pour plus d'informations
8 Voir définition p 19
9 Voir annexe n°5 p 31 pour plus d'informations

Etant un appareil très complexe, le microcontrôleur n'a pas été aisé à utiliser. En effet le langage de communication est loin d'être proche du nôtre. Alors qu'il serait évident pour nous de rentrer de simples lignes de commandes compréhensibles pour tous, c'est l'inverse. Du fait que les microcontrôleurs soient utilisés dans le monde entier, il a donc fallu mettre en place un langage compréhensible par tout le monde. Le langage C fut adopté pour programmer les microcontrôleurs. Étant un langage compréhensible pour l'homme, il faudra encore une étape pour que le microcontrôleur exécute de manière accessible pour lui, les lignes de commande que nous lui avons ordonnées. Cette dernière étape s'appelle la compilation, elle est réalisée le plus souvent à l'aide d'un compilateur se trouvant avec l'environnement de programmation. A ce niveau là, le langage une fois compilé n'est plus du tout compréhensible pour l'utilisateur, seul le processeur pourra le comprendre car il se résume à du tout ou rien (0 ou 1).

Pour se replacer dans le contexte du précédent projet, nous allons rappeler quelques points essentiels :

Pour sélectionner le bon relais[10], nous avons dû tout d'abord mesurer à l'aide d'une pince ampèremétrique la consommation de l'appareil électroménager qui fonctionne sous **230V/50Hz** et consomme **0,69A** (cf illustration 5) lorsqu'il est en marche. Ces informations complémentaires ont été acquises grâce à la plaque signalétique du réfrigérateur.

Illustration 4: Consommation du réfrigérateur (illustration personnelle)

10 Voir définition p 19

Lors de la semaine 37, nous avons choisi notre projet parmi une liste de sujets. L'objectif principal était de réparer un réfrigérateur dont le système de régulation n'était plus fonctionnel et au lieu de remplacer complètement le thermostat[11], nous avons préféré mettre en application plusieurs techniques vues au sein du département GEII, comme par exemple l'utilisation d'un microcontrôleur, la conception d'un programme en langage C, la réalisation de circuits imprimés.

La maquette est donc composées principalement :

Nom / Fonction	***Référence***
Une alimentation à découpage pour réaliser du 5V à partir de 12V	*LM2574N*
Capteur de température[12]	*LM35*
Connecteur + Nappe de programmation	*HE10*
Relais 230V/10A	*PCD-105-D2M*
Microcontrôleur	*ATtiny 13*
Transformateur 230V/12V 0,5A	*SYS1196-0612-W2E*
Transistor[13]	*2N2222*

Tableau 1: Liste des principaux composants

Nous disposons d'un transformateur **230V/12V 0,5A**, le microcontrôleur ne fonctionnant seulement qu'avec du 5V, il nous a donc fallu réaliser une alimentation à découpage (LM2574N) délivrant du 5V à partir de 12V. Cette alimentation génère du **+5V** sur sa connexion 7 (Output) lorsqu'on lui impose une tension de **+12V** sur sa connexion 5 (Vin). L'alimentation peut ensuite ajuster cette tension de sortie grâce à son entrée 1 (FeedBack).

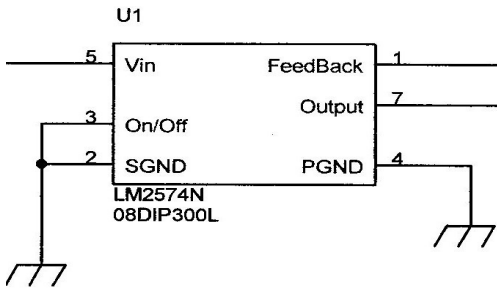


Illustration 5: Schéma du LM2574N (illustration personnelle)

11 Voir définition p 19
12 Voir définition p 19
13 Voir définition p 19

Voyons maintenant la seconde partie du premier projet qui consistait à l'étude et à la réalisation de la partie programmation.

Nous avons choisi d'acquérir la température à l'aide d'un **LM35** (cf illustration). Ce capteur doit être alimenté en +5V sur sa connexion 1 (la plus à gauche de l'illustration 1), la connexion 3 (à droite) quant à elle est reliée au 0V. Celle du milieu nous délivrera une tension qui variera de **10mV/°C** en fonction de la température. C'est cette tension qui nous intéresse et que nous allons traiter pour savoir si la mise en marche du réfrigérateur est nécessaire ou non.

La maquette fonctionnant sous 5V et le réfrigérateur sous 230V, nous ne pouvons pas le commander de manière directe. Il a donc fallu que nous utilisons un composant particulier nous permettant de faire cette liaison entre le 5V et le 230V. Notre choix s'est donc tourné vers un relais 230V ayant un pouvoir de coupure et une résistance supérieure à **10 fois le courant consommé** par l'appareil. L'IUT possédait des relais 230V/10A, nous utiliserons l'un d'eux.

Pour acquérir la température sur le microcontrôleur et ainsi la comparer à la température souhaitée, nous avons utilisé la fonction **read_adc ()** qui va lire la tension vue par l'entrée du convertisseur analogique numérique et la convertir en numérique pour qu'ensuite nous puissions plus facilement définir notre échelle de température.

Pour rester cohérente avec un fonctionnement réel de réfrigérateur, nous avons défini la température de ce dernier aux alentours de 4°C dans notre programme. Par conséquent, le réfrigérateur sera en fonctionnement tant que la température à l'intérieur est supérieure à **4°C**. Pour commander le relais nous utiliserons le PORTB.4. Au niveau du programme cela donne les lignes de codes suivantes :

Programme partiel :

```
        i = read_adc (3) ; // Lit la tension vue par l'entrée du
convertisseur analogique numérique n°3
    if (i > 40)       // Si la température est plus élevée que 4°C,
                      // 40 car le capteur est précis au dixième de
                      // degrés près
        {
                      PORTB.4 = 1 ;   // Réfrigérateur en marche
        }
    else              // Sinon
        {
                      PORTB.4 = 0 ;   // Réfrigérateur à l'arrêt
        }
```

Nous arrivons au stade d'introduire notre programme dans le microcontrôleur. Pour réaliser cette tâche nous avons utilisé une nappe et un connecteur HE10, le tout relié à un connecteur **ISP** faisant ainsi la liaison avec l'ordinateur.

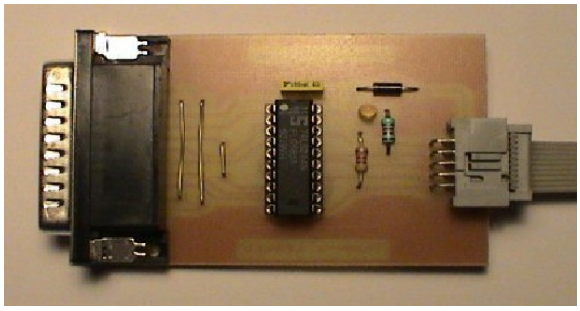


Illustration 6: Connecteur ISP (source site de M. Lequeu)

Passons à présent à la dernière partie de la maquette, la fonction commande : comme expliqué précédemment, nous avons eu recours à un relais 230V/10A (cf illustration n°3). Nous avons donc réalisé un montage avec la bobine du relais en série avec une LED 2mA et une résistance 3300 Ω dans le but de savoir si la bobine est alimentée ou non. Le tout est en parallèle avec une diode 1N4007 pour éviter une éventuelle inversion du courant qui provoquerait la destruction de la LED. Cela donne le schéma en illustration 7.

1.2. Les problèmes rencontrés

Les problèmes rencontrés durant le précédent projet auront été variés. Dans un premier temps nous avons eu le problème de la reprogrammation du microcontrôleur lors d'un second test de programme. Ce problème a été résolu en décochant une option dans l'environnement de programmation AVR Studio au moment du transfert du programme mais cela nous a valu le remplacement du microcontrôleur car il était devenu inutilisable. Puis se sont présentés quelques soucis de réalisation des circuits imprimés notamment avec la graveuse qui ne fonctionnait pas de manière optimale, provoquant la destruction des cartes en gravant trop la couche de cuivre. L'incident majeur aura été la mise en marche du relais. Pensant que ce dernier était défaillant, nous avons cherché le problème pendant longtemps, perdant du temps et prenant du retard notamment sur l'aménagement du réfrigérateur comme par exemple la réalisation du trou pour le passage du capteur et l'aménagement de l'alimentation du réfrigérateur. Le problème venait en fait du montage en lui-même. En analysant de plus près le schéma de câblage du relais, nous nous sommes aperçus qu'il manquait une connexion importante : l'alimentation. Il est nécessaire d'apporter du +5V à ce montage car lorsque l'on désire faire fonctionner le réfrigérateur, il faut avoir un courant dans la bobine afin de fermer l'interrupteur du relais : hors, sans une source de tension il n'y aura pas de courant dans cette partie du montage. Le montage modifié devient :

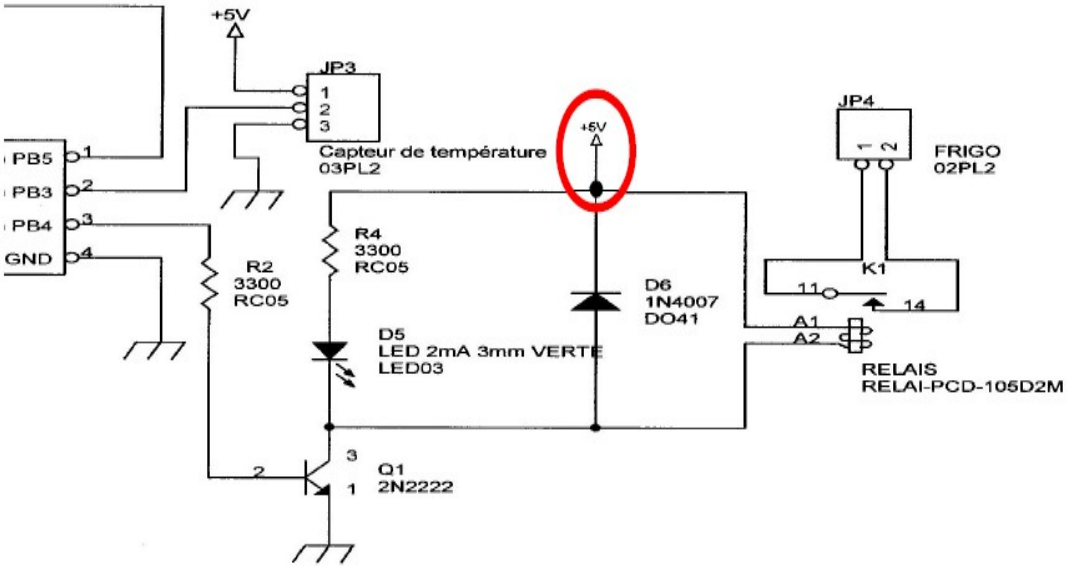


Illustration 7: Schéma de montage du relais modifié (illustration personnelle)

2. Le nouveau projet

2.1. Les nouvelles fonctionnalités

Dans le cadre du projet du semestre 4, nous avons choisi de continuer et améliorer le précédent projet en lui apportant de nouvelles fonctionnalités. La premier point que nous avons souhaité modifier est le choix de la température qui auparavant été fixée dans le programme. Pour concevoir cette nouvelle fonction, notre choix s'est porté vers la variation de tension lors de l'ajustement d'un potentiomètre. Afin que l'utilisateur est plus d'aisance pour l'utilisation de son réfrigérateur, nous avons choisi d'ajouter un écran LCD au projet. Ce dernier affichera la température présente dans le réfrigérateur ainsi que celle désirée par l'utilisateur. Et enfin, pour des raisons d'esthétique, nous avons choisi de réduire la taille de la carte finale afin qu'elle puisse être contenue dans une boite suffisamment discrète. Ce sont donc sur ces trois points que l'étude suivante portera.

Nom / Fonction	***Référence***
Une alimentation à découpage pour réaliser du 5V à partir de 12V	*LM2574N*
Capteur de température	*LM35*
Connecteur + Nappe de programmation	*HE10*
Relais 230V/10A	*PCD-105-D2M*
Microcontrôleur	*ATmega8535*
Transformateur 230V/12V 0,5A	*SYS1196-0612-W2E*
Transistor	*2N2222*
Écran LCD	*MC1604C-SERIES*

Tableau 2: Liste des principaux composants

2.2. Des solutions trouvées

Dans le précédent projet, nous avons utilisé un microcontrôleur **ATtiny 13** car sa petite taille et ses nombreuses fonctions étaient largement nécessaires. Mais maintenant, pour que nous puissions utiliser complètement l'afficheur LCD, nous avons dû changer ce microcontrôleur pour un autre, un ATmega8535 [14] (cf illustration 8).

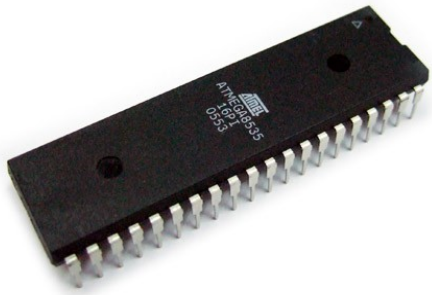


Illustration 8: Microcontrôleur ATmega8535 de la famille Atmel (source internet)

Pour le schéma nous avons suivi les typons réalisés par Thierry LEQUEU que nous avons récupéré sur son site. Cela nous a facilité grandement la tâche sachant que nous ne disposions par de beaucoup de temps. Par conséquent le schéma de câblage réalisé sur le logiciel Orcad est le suivant :

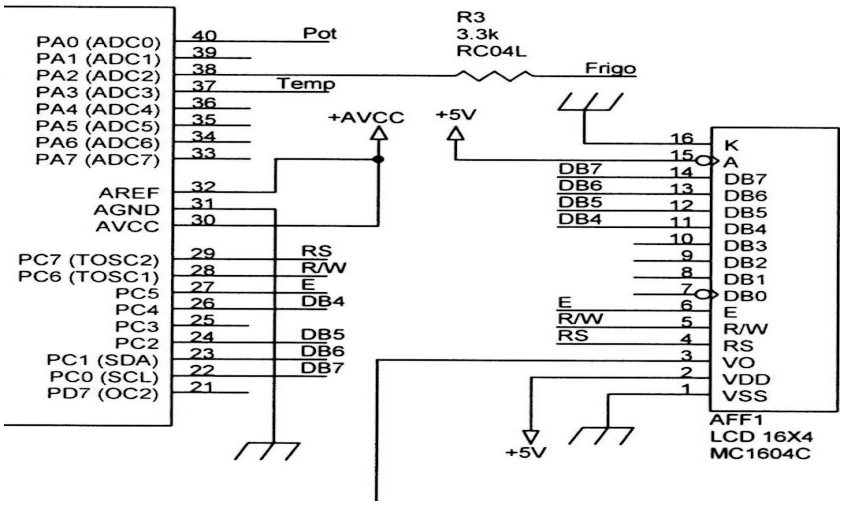


Illustration 9: Schéma de câblage de l'écran LCD

14 Voir annexe n°3 pour plus d'informations

Afin de faciliter la lecture sur l'afficheur, nous avons choisi d'alimenter le rétroéclairage de l'écran LCD[15]. C'est pour cela qu'il est nécessaire d'apporter +5V sur la broche 15 de l'écran ainsi que relier la broche 16 au 0V.

Pour récupérer la tension du potentiomètre de la température servant de consigne, nous avons utilisé la même fonction read_adc(). Pour faciliter le routage, nous avons décidé de relier la sortie du potentiomètre à l'entrée du convertisseur analogique numérique n°0 (ADC0). La température acquise par le capteur LM35 sera quand à elle vue par le microcontrôleur sur l'entrée du convertisseur analogique numérique n°3 (ADC3). La partie commande sera quand à elle gérée par le port PA2. Cette dernière n'a pas été modifiée par rapport au précédent projet (cf illustration 10).

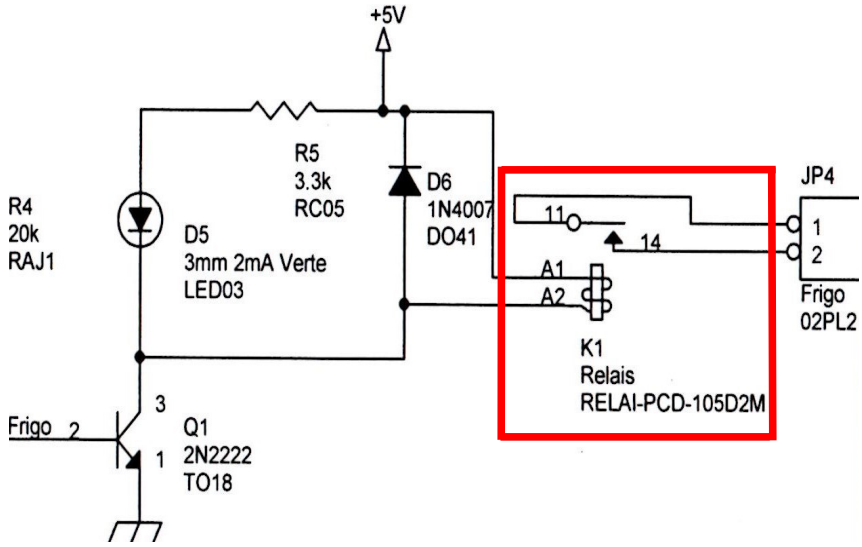


Illustration 10: Schéma de la partie commande du réfrigérateur

Nous allons à présent passer à la partie programmation.

15 Voir définition p 19

Le programme est le suivant :

```
S = 0 ;
        for (a = 0 ; a<100 ; a ++)
            {
              i = read_adc (3) ; // Lis la tension du capteur de température et
                                    la convertie en numérique
              S = S + i ;
            }
Valeur1 = S/100 ;
Teta = (int) (Valeur1*0.4882); // Conversion du numérique vers des degrés
delay_ms (200);
Valeur2 = read_adc (0) ;     // Lis la tension du potentiomètre de commande et
                                la convertie en numérique
Consigne = (int) (Valeur2*0.030);       // Conversion du numérique vers des
                                           degrés tout réglant la  sensibilité du
                                              potentiomètre pour 30°C
            if (Consigne > 30)
                    Consigne = 30 ; // On bloque la valeur de consigne à 30°C

sprintf (Tampon, "Il fait :") ;
lcd_gotoxy(0,0);
lcd_puts (Tampon) ;             // On affiche « Il fait : »
sprintf (Tampon, "         %4d C", Teta) ;
lcd_gotoxy(0,1);
lcd_puts (Tampon) ;             // On affiche la température lue par le capteur
sprintf(Tampon,"Vous voulez :");
lcd_gotoxy (0,2);
lcd_puts(Tampon);               // On affiche « Vous voulez : »
sprintf(Tampon,"         %4d C",Consigne);
lcd_gotoxy (0,3);
lcd_puts(Tampon);               // On affiche la température de consigne

if (Teta>=Consigne+1)           // Si la température est supérieure ou égale à
                                   celle de la consigne + 1
PORTA.2 = 1 ;                   // Réfrigérateur en marche

if (Teta<=Consigne-1)         // Si la température est inférieure ou égale à celle
                                 de la consigne - 1
PORTA.2 = 0 ;                   // Réfrigérateur à l'arrêt
```

Afin d'obtenir une mesure précise et stable, nous avons eu l'idée de prendre plusieurs mesures et d'en faire la somme puis de diviser le tout par le nombre de points de mesure : une moyenne ; ce qui est réalisé par ce programme :

```
S = 0 ;
        for (a = 0 ; a<100 ; a ++)
              {
                i = read_adc (3) ; // Lis la tension du capteur de température et la
                                       convertie en numérique
                S = S + i ;
              }
Valeur1 = S/100 ;
```

Pour pouvoir utiliser l'écran LCD, il est nécessaire d'inclure la bibliothèque alcd.h au programme.

Afin que le réfrigérateur ne soit pas constamment en train d'être coupé ou alimenté par le relais, nous avons pensé à appliquer un petit phénomène d'hystérésis autour de la valeur de consigne de température, c'est à dire qu'au lieu que le réfrigérateur se coupe à la température exacte commandé par l'utilisateur, il cessera de fonctionner lorsque la température sera à la valeur de la consigne – 1. De même que pour la remise sous tension lorsque la température du réfrigérateur commence à remonter, il sera remis sous tension à la valeur de la consigne + 1.

Cela donne le programme suivant :

```
if (Teta>=Consigne+1)          // Si la température contenue dans le réfrigérateur est
                                  supérieure ou égale à celle de la consigne + 1
PORTA.2 = 1 ;                  // Réfrigérateur en marche

if (Teta<=Consigne-1)          // Si la température contenue dans le réfrigérateur est
                                  inférieure ou égale à celle de la consigne - 1

PORTA.2 = 0 ;                  // Réfrigérateur à l'arrêt
```

16

16 Voir annexe n°13 pour lire le programme complet

2.3. Des problèmes rencontrés

Les principaux soucis que nous avons rencontrés durant ce semestre en Études et Réalisation sont l'utilisation de la graveuse qui nous avait gravé trop par deux fois nos cartes. La première utilisation de l'afficheur LCD s'est aussi avérée difficile à programmer. En faisant le point sur ce projet de fin de deuxième année, le point le plus dur à réaliser aura été la partie routage du fait du peu de place qui nous était disponible à cause de la taille de la boite fournie par M LEQUEU[17]. Car le précédent projet été contenu dans une boite plus grande que celle que nous utilisons pour le projet du semestre 4. Il fallait en effet, insérer un écran LCD, un ATmega8535 ainsi que tout le nécessaire au projet dans un espace réduit. Mais aujourd'hui le projet est fonctionnel et prêt à être utilisé sur le réfrigérateur.

17 Voir annexes 10 et 11 : Typon

Conclusion

Ce projet de deuxième année nous a permis de mettre en application beaucoup de domaines du GEII que ce soit en électronique, en électrotechnique ou en informatique. Grâce à l'étude des transistors faite en première année, nous avons pu les utiliser dans notre projet. Les alimentations vues en MC-ET2 nous ont été utiles dans la réalisation de l'une d'elles, nécessaires pour le microcontrôleur. Au semestre 4, le projet d'étude et réalisation devait principalement porter sur l'informatique. C'est pour cela que nous avons choisi de reprendre notre projet du semestre 3 pour pouvoir l'améliorer grâce à l'informatique. Et plus précisément avec AVR Studio, logiciel que nous avons utilisé. Ce logiciel nous a permis de programmer un ATmega8535, microcontrôleur que nous avons appris à connaître durant les différentes séances. Nous avons été aidé par les cours sur les alimentations vues en MC-ET2 qui nous ont été utiles dans la réalisation de l'une d'elles, nécessaires pour le microcontrôleur

Les différents tests sur le réfrigérateur nous ont permis de vérifier le bon fonctionnement de la maquette.

Résumé

Dans un premier temps, le projet de base consistait à trouver une alternative au dysfonctionnement d'un thermostat de réfrigérateur. Ce projet étant mené à bout lors du semestre 3, nous avons choisis de l'améliorer au semestre 4 en lui ajoutant un afficheur LCD ainsi qu'un potentiomètre qui permettra à l'utilisateur de régler la température qu'il souhaite avoir dans son réfrigérateur. Nous devons donc acquérir la température, la traiter au moyen d'un microcontrôleur et l'afficher grâce à l'écran LCD et enfin alimenter ou non le réfrigérateur. Pour cela, nous avons utilisé un capteur de température LM35, un microcontrôleur ATmega8535, un afficheur LCD et enfin un relais 230V pour faire le lien entre notre signal de commande et la partie puissance.

Ce nouveau projet nous a permis de montrer nos connaissances techniques apprises pendant notre formation. Grâce à nos compétences acquises lors de la formation et à l'aide de Monsieur Thierry LEQUEU qui nous a épaulé durant le projet, nous avons pu trouver des solutions pour que le maquette fonctionne correctement.

Index des mots clefs

- **Régulation :** Fait de maintenir et de régler le fonctionnement d'un appareil ou d'un système.

- **Température :** Sensation de chaleur ou de froid éprouvée par le corps en un lieu.

- **Réfrigérateur :** Appareil électroménager servant à produire du froid.

- **Thermostat :** Appareil de réglage de la température qui permet de maintenir une température constante.

- **Microcontrôleur :** Appareil de dimensions réduites permettant de mesurer l'impédance, le voltage, la résistance de courants électriques, ses applications sont très variées.

- **Froid :** Qui donne la sensation d'être à basse température.

- **Capteur de température :** Mécanisme capable de délivrer une énergie électrique en captant une température.

- **Programme :** Liste des instructions permettant à un microcontrôleur d'exécuter une série d'opérations.

- **Relais :** Dispositif permettant la commutation à distance d'un circuit électrique.

- **Transistor :** Composant électronique qui peut amplifier un signal.

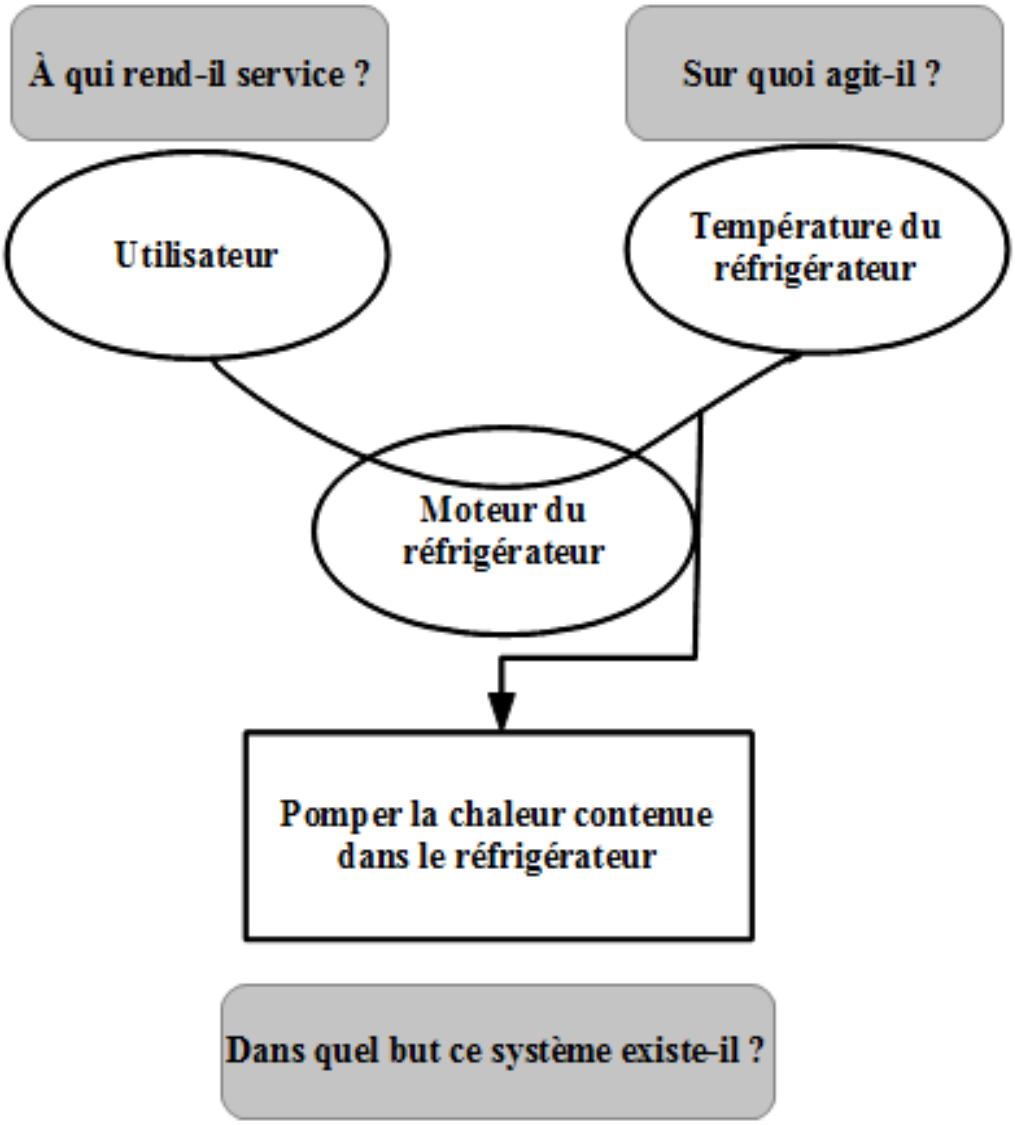
- **Écran LCD :** Dispositif d'affichage de données alphanumériques qui utilise le reflet de la lumière sur des cristaux liquides.

Index des tables

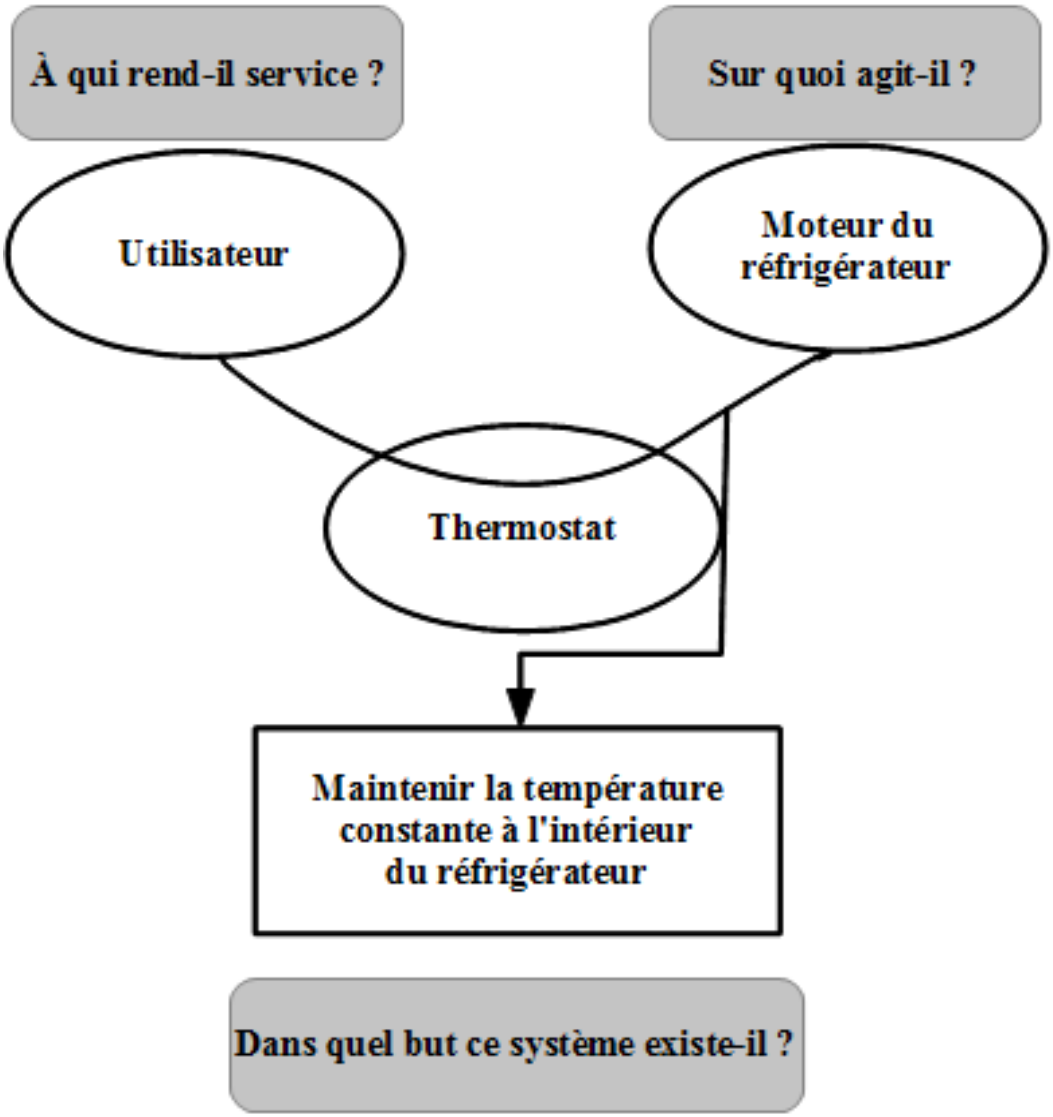
Index des illustrations

Table des annexes

Annexe n°1 : Diagramme bête à corne du moteur



Annexe n°2 : Diagramme bête à corne du thermostat



Annexe n°3 : Datasheet de l'ATmega8535

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 8K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 512 Bytes Internal SRAM
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels for TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x for TQFP Package Only
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, 44-lead PLCC, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega8535L
  - 4.5 - 5.5V for ATmega8535
- Speed Grades
  - 0 - 8 MHz for ATmega8535L
  - 0 - 16 MHz for ATmega8535

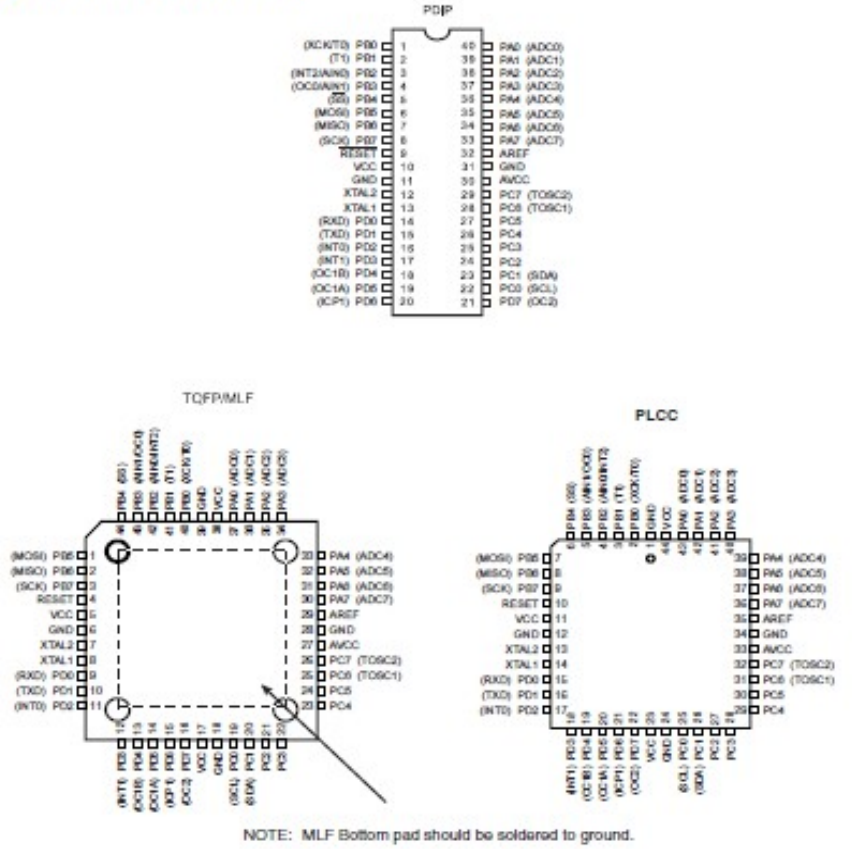
8-bit AVR® Microcontroller with 8K Bytes In-System Programmable Flash

ATmega8535
ATmega8535L

2502K–AVR–10/06

Pin Configurations

Figure 1. Pinout ATmega8535

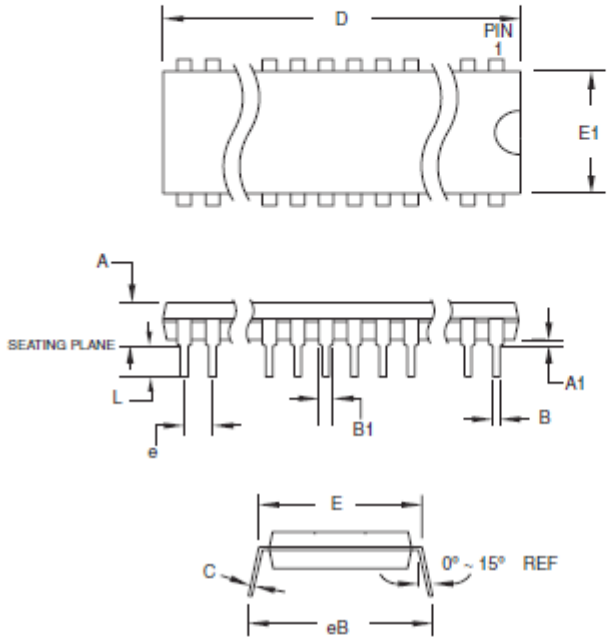


NOTE: MLF Bottom pad should be soldered to ground.

Disclaimer

Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

40P6



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

09/28/01

2325 Orchard Parkway San Jose, CA 95131	TITLE **40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	DRAWING NO. 40P6	REV. B

Annexe n°4 : Datasheet du LM35

National Semiconductor

December 1994

LM35/LM35A/LM35C/LM35CA/LM35D
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-202 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Connection Diagrams

TO-46
Metal Can Package*

TL/H/5516–1

*Case is connected to negative pin (GND)

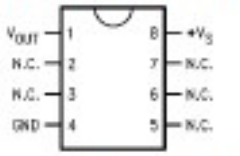
Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

TO-92
Plastic Package

TL/H/5516–2

Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

SO-8
Small Outline Molded Package

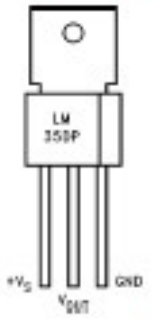


TL/H/5516–21

Top View
N.C. = No Connection

Order Number LM35DM
See NS Package Number M08A

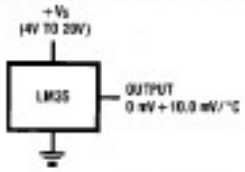
TO-202
Plastic Package



TL/H/5516–24

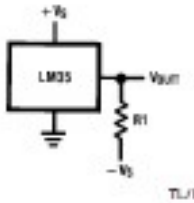
Order Number LM35DP
See NS Package Number P03A

Typical Applications



TL/H/5516–3

FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



TL/H/5516–4

Choose $R_1 = -V_S/50\ \mu A$

$V_{OUT} = +1,500$ mV at +150°C
$= +250$ mV at +25°C
$= -550$ mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

'BISTATE' is a registered trademark of National Semiconductor Corporation.

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp., TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-202 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 Package, (Soldering, 10 seconds)	260°C
TO-202 Package, (Soldering, 10 seconds)	+230°C

SO Package (Note 12):	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

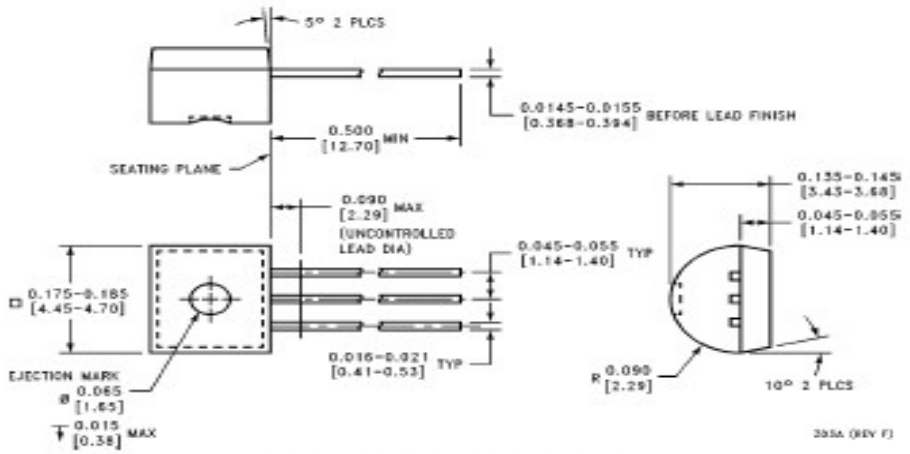
Electrical Characteristics (Note 1) (Note 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A = +25°C	±0.2	±0.5		±0.2	±0.5		°C
	T_A = −10°C	±0.3			±0.3		±1.0	°C
	T_A = T_{MAX}	±0.4	±1.0		±0.4	±1.0		°C
	T_A = T_{MIN}	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity (Note 8)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.18**		**±0.35**	**±0.15**		**±0.3**	°C
Sensor Gain (Average Slope)	$T_{MIN} \le T_A \le T_{MAX}$	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/°C
Load Regulation (Note 3) $0 \le I_L \le 1$ mA	T_A = +25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
	$T_{MIN} \le T_A \le T_{MAX}$	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A = +25°C	±0.01	±0.05		±0.01	±0.05		mV/V
	$4V \le V_S \le 30V$	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S = +5V, +25°C	56	67		56	67		µA
	V_S = +5V	**105**		**131**	**91**		**114**	µA
	V_S = +30V, +25°C	56.2	68		56.2	68		µA
	V_S = +30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	$4V \le V_S \le 30V$, +25°C	0.2	1.0		0.2	1.0		µA
	$4V \le V_S \le 30V$	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L = 0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J = T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Note 1: Unless otherwise noted, these specifications apply: $-55°C \le T_J \le +150°C$ for the LM35 and LM35A; $-40°C \le T_J \le +110°C$ for the LM35C and LM35CA; and $0°C \le T_J \le +100°C$ for the LM35D. V_S = +5Vdc and I_{LOAD} = 50 µA, in the circuit of *Figure 2*. These specifications also apply from +2°C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-202 package is 65°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Physical Dimensions inches (millimeters) (Continued)



TO-92 Plastic Package (Z)
Order Number LM35CZ, LM35CAZ or LM35DZ
NS Package Number Z03A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

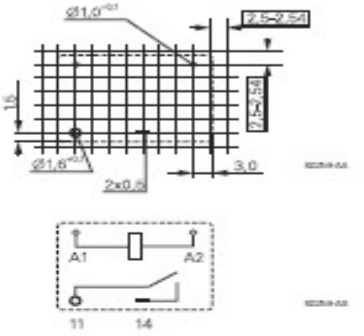
Annexe n°5 : Datasheet du relais PCD-105-D2M

Low Profile PCB Relay PCD (Continued)

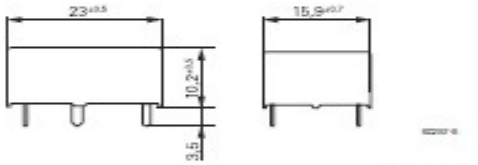
Insulation		
Dielectric strength coil-contact circuit		2500 V_{rms}
	open contact circuit	750 V_{rms}
Clearance / creepage coil-contact circuit		≥ 4/4 mm
Material group of insulation parts		≥ IIIa
Insulation to IEC 61810-1		
	Type of insulation coil-contact circuit	basic
	open contact circuit	micro disconnection
	Rated insulation voltage	250 V
	Pollution degree	3
	Rated voltage system	240 V
	Overvoltage category	III

Other data		
Mechanical endurance		> 10x10⁶ cycles
Material		
	RoHS - Directive 2002/95/EC	compliant per as par product date code "DT" (refers to June 2004).
Environment		
	Ambient temperature range	-30 ... +70°C
	Shock resistance (function) NO / NC contact	10 ms
	Shock resistance (destruction)	100 g
	Category of protection	RT II - flux proof, RT III - wash tight
Processing		
	Resistance to soldering heat flux-proof version	270°C / 10 s
	wash-tight version	260°C / 5 s
	Relay weight	9 g
	Packaging unit	1000 pcs

PCB layout / terminal assignment
Bottom view on solder pins



Dimensions



Product key — Typical product key: PCD -1 24 -D 2 M

- **Type**: PCD Low Profile PCB Relay PCD
- **Number of contacts**: 1 1 NO contact
- **Coil**: 5 5 VDC; 6 6 VDC; 12 12 VDC; 24 24 VDC
- **Coil version**: D standard 200 mW
- **Contact material**: 1 AgCdO; 2 $AgSnO_2$
- **Contact configuration**: M 1 NO contact (1 form A)
- **Version**: - flux proof; H wash tight

Other types on request

Product key	Version	Cont. material	Cont.configuration	Coil	Part number
PCD-105-D2M	standard 200mW flux proof	$AgSnO_2$	1 NO contact	5 VDC	1721105-1
PCD-112-D2M				12 VDC	1721105-4
PCD-124-D2M				24 VDC	1721105-5
PCD-148-D2M				48 VDC	1721105-6
PCD-105-D2MH	standard 200mW wash tight			5 VDC	1721105-7
PCD-112-D2MH				12 VDC	1-1721105-0
PCD-124-D2MH				24 VDC	1-1721105-1
PCD-148-D2MH				48 VDC	1-1721105-2

Low Profile PCB Relay PCD

- **1 pole 10 A**
- **1 NO contact**
- **Low coil power 200 mW**
- **Height 10.2 mm**
- **Wash tight**

Applications

Domestic appliances, coffeee machines, irons, office equipment

Approvals

cULus E82292
Technical data of approved types on request

Contact data

Contact configuration	1 NO
Contact set	single contact
Type of interruption	micro disconnection
Rated voltage / max. switching voltage AC	250 / 400 VAC
Rated current	10 A
Limiting continuous current	10 A
Maximum breaking capacity AC	2500 VA
Contact material	$AgSnO_2$, AgCdO
Rated frequency of operation with / without load	10 / 300 min^{-1}
Operate- / release time	max 8 / 2 ms
Bounce time NO / NC contact	max 3 / 2 ms

Contact ratings

Type	Contact	Load	Ambient temp. [°C]	Cycles
UL 508				
PCD-1...D2M(H)	NO	10 A, 250 VAC, resistive	85°C	$50x10^3$
PCD-1...D2M(H)	NO	15 A, 125 VAC, resistive	23°C	$100x10^3$

Coil data

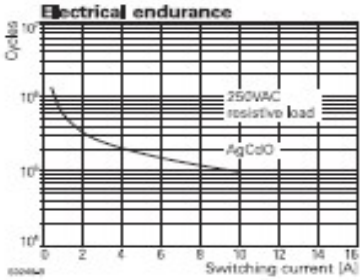
Rated coil voltage range DC coil	3 ... 48 VDC
Operative range to IEC 61810	2

Coil versions, DC-coil

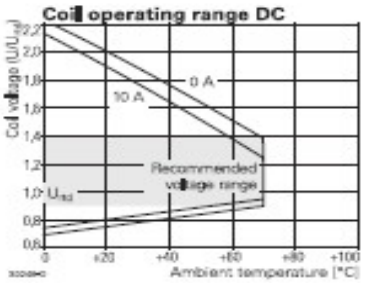
Coil code	Rated voltage VDC	Operate voltage VDC	Release voltage VDC	Coil resistance Ohm	Rated coil power mW
006	6	4.5	0.3	180±10%	200
012	12	9.0	0.6	720±10%	200
024	24	18.0	1.2	2880±10%	200

All figures are given for coil without preenergization, at ambient temperature +23°C
Other coil voltages on request

Electrical endurance



Coil operating range DC



Datasheet Rev. HK1
Issued 2008/11
www.tycoelectronics.com
www.schrackrelays.com

Dimensions are in mm unless otherwise specified and are shown for reference purposes only.

Product specification according to IEC 61810-1. Product data, technical parameters, test conditions and processing information only to be used together with the 'Definitions' section in the catalogue or at schrackrelays.com in the 'Schrack' section.

Specifications subject to change.

Annexe n°6 : Cahier des charges

Problème : Réfrigérateur dont le thermostat n'est plus fonctionnel

Conséquences :

- Le réfrigérateur fonctionne en permanence
- Le moteur ne sait pas quand est-ce qu'il doit s'arrêter
- Formation de glace, aliments détruits, consommation excessive d'énergie

Solutions proposées :

- Contrôler l'activation du moteur à l'aide d'un microcontrôleur
- Ce dernier obtiendra la température du frigo à l'aide d'un capteur de température et comparera la valeur obtenue à celle désirée par l'utilisateur
- Puis autorisera ou non la mise en marche du réfrigérateur
- La température souhaitée sera fixée au moyen d'un potentiomètre
- Afficher la température souhaitée par l'utilisateur

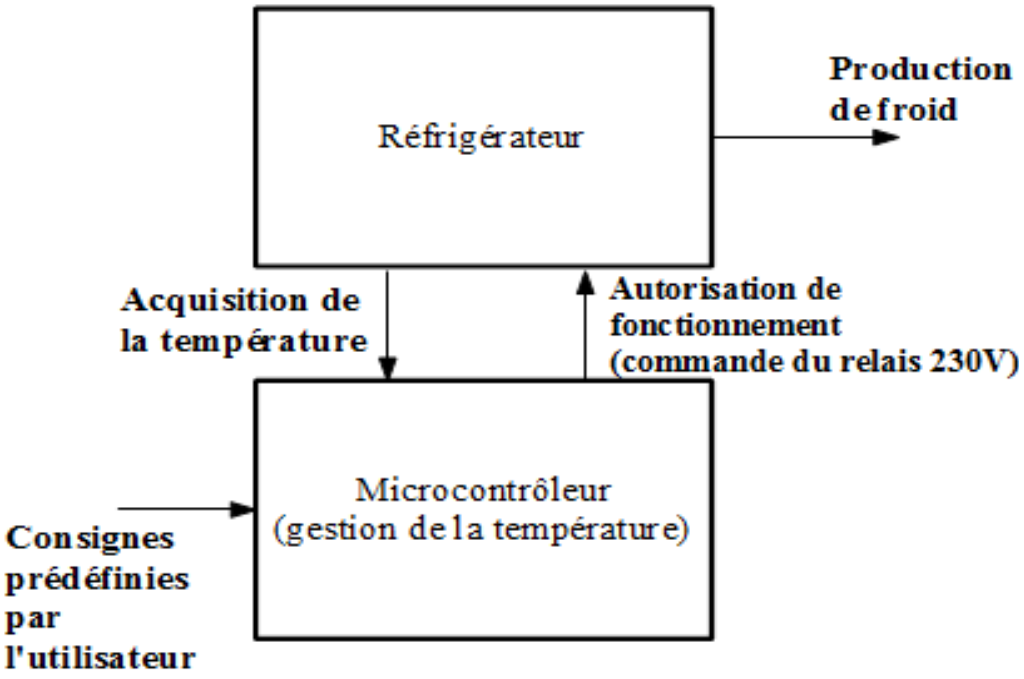
Composition du projet :

- Une carte comportant un microcontrôleur, un capteur de température, un composant pour commander du 230V (Tension sous lequel est le réfrigérateur
- Une carte d'alimentation permettant d'abaisser la tension de 12V à 5V (Tension supportée par le microcontrôleur)
- Un transformateur secteur 230V/12V

Contraintes :

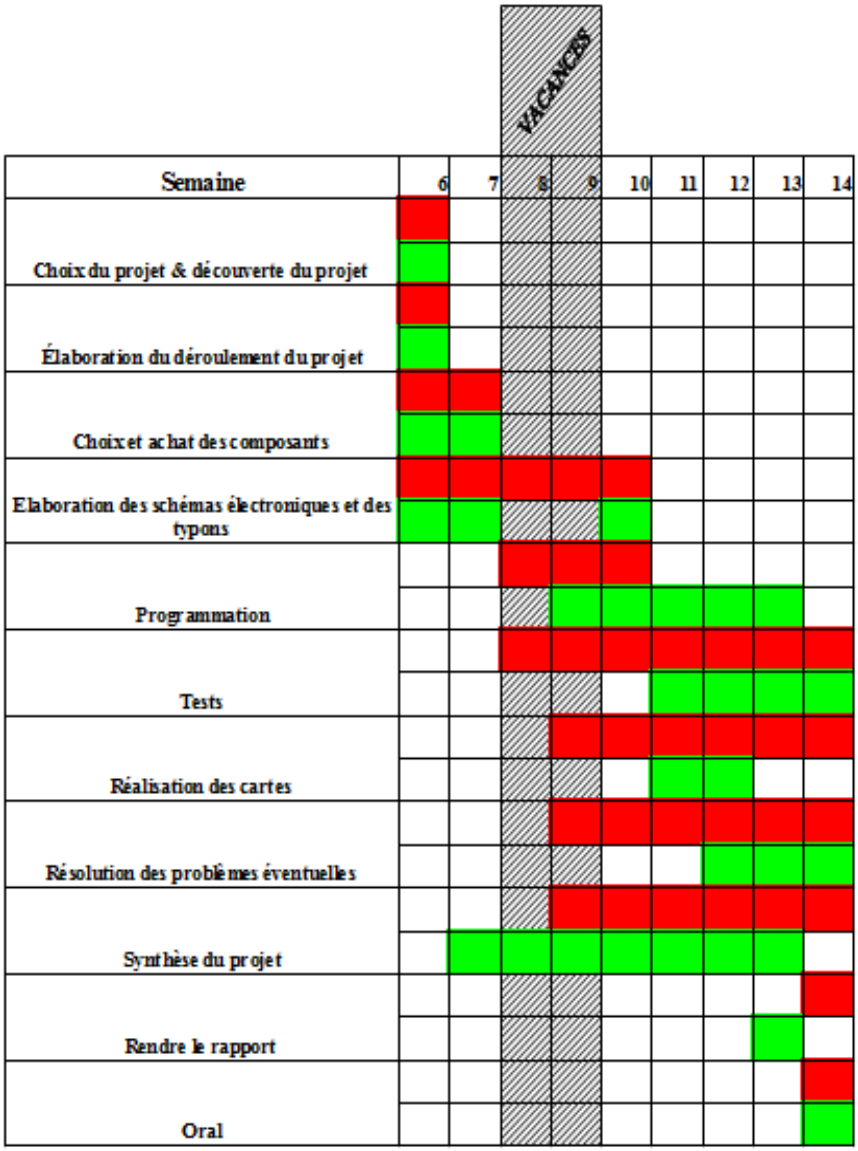
Nous allons devoir faire face à des contraintes d'espace (taille des cartes) et d'esthétique (maquette + réfrigérateur).

Annexe n°7 : Schéma fonctionnel de niveau 2



Annexe n°8 : Planning

Planning prévisionnel et réel



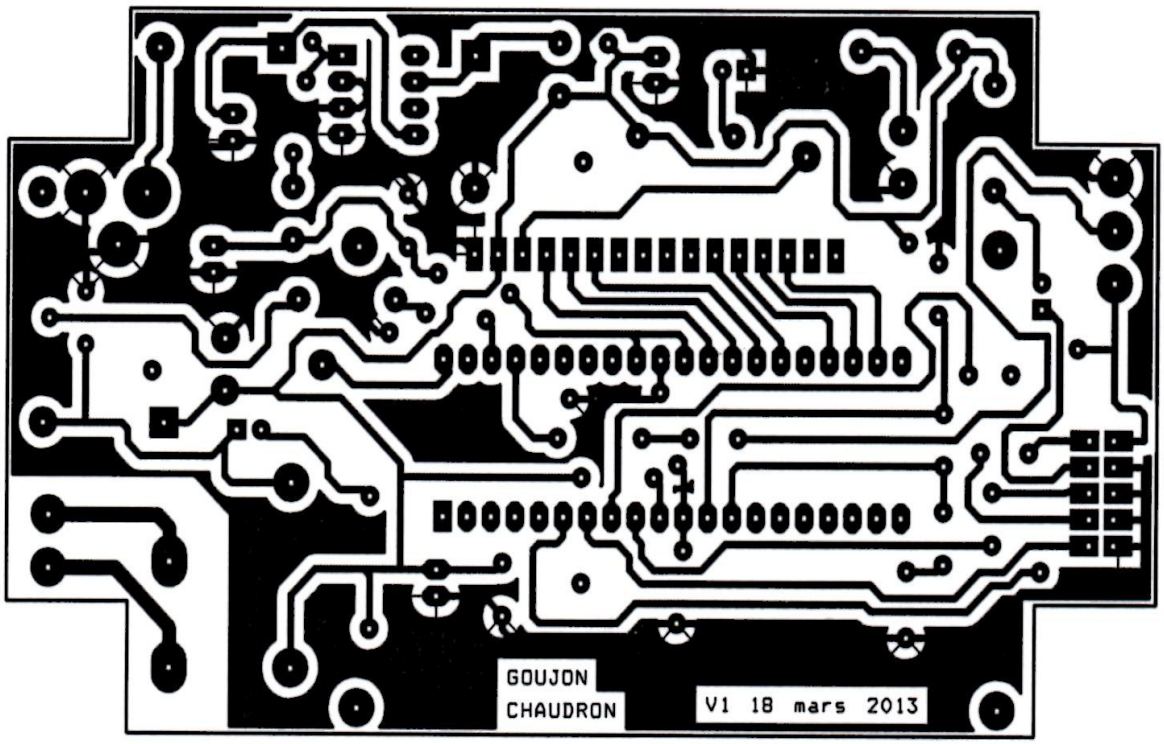
Annexe n°9 : Nomenclature

Carte frigo 2 Revised: Thursday, March 28, 2013
GOUJON Benoît - CHAUDRON Bastien - Projet E&R Revision: 1

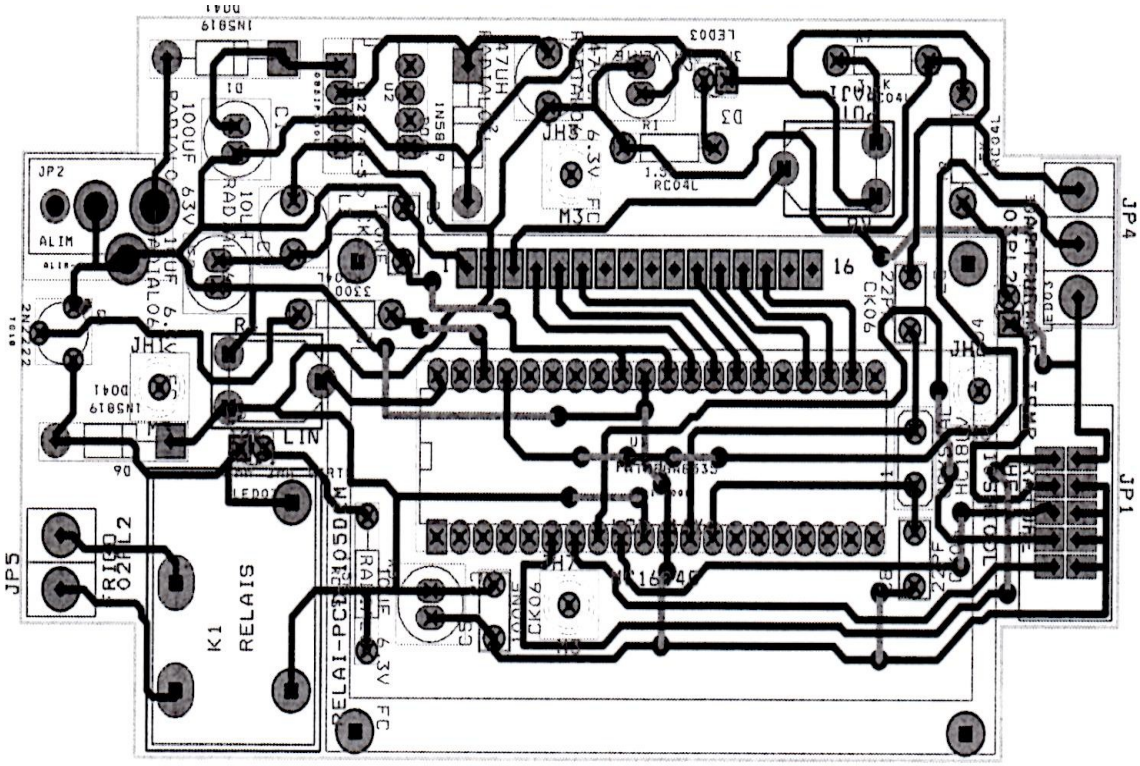
Bill Of Materials March 28,2013 14:00:00 Page1

Item	Quantity	Reference	Part
1	1	AFF1	LCD 16X4
2	1	C1	100uF 63V FC
3	1	C2	470uF 6.3V FC
4	2	C3,C5	100nF
5	2	C4,C6	10uF 6.3V FC
6	2	C7,C8	22pF
7	1	D1	1N5819
8	1	D2	11DQ06
9	2	D3,D5	3mm 2mA Verte
10	1	D4	3mm 2mA Jaune
11	1	D6	1N4007
12	4	JH1,JH3,JH6,JH7	HEADER 1
13	1	JP1	ALIM
14	1	JP2	HE10
15	1	JP3	Capteur de température
16	1	JP4	Frigo
17	1	K1	Relais
18	1	L1	47uH
19	1	L2	10uH
20	1	Q1	2N2222
21	2	R1,R2	1.5k
22	3	R3,R5,R7	3.3k
23	2	R4,R6	20k
24	1	U1	LM2574N-5.0
25	1	U2	ATmega8535
26	1	Y1	CRYSTAL

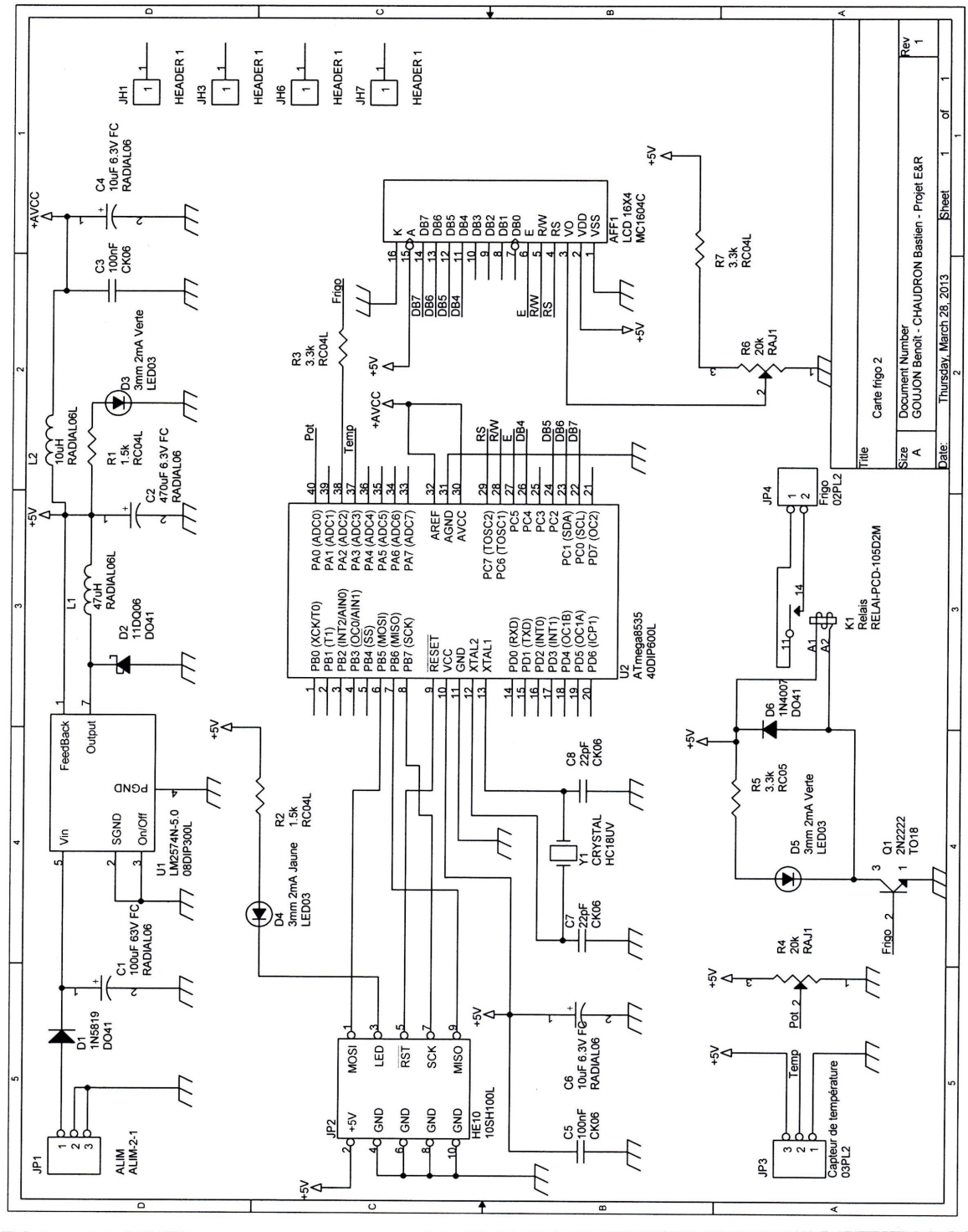
Annexe n°10 : Typon final



Annexe n°11 : Typon + sérigraphie



Annexe n°12 : Schéma structurel



Annexe n°13 : Programme complet

```
/*****************************************************
This program was created by the
CodeWizardAVR V2.60 Evaluation
Automatic Program Generator
© Copyright 1998-2012 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
Project : Réalisation d'un régulateur de température pour un réfrigérateur version 2
Version : V1
Date    : 18/03/2013
Author  : Benoît GOUJON et Bastien CHAUDRON
Company :
Comments:
Chip type: ATmega8535
Program type: Application
AVR Core Clock frequency: 16,000000 MHz
Memory model: Small
External RAM size: 0
Data Stack size: 128
*****************************************************/

#include <mega8535.h>
#include <stdio.h>
#include <delay.h>

// Alphanumeric LCD functions

#include <alcd.h>

// Declare your global variables here

#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

// Read the AD conversion result

unsigned int read_adc(unsigned char adc_input)

{
ADMUX=adc_input | ADC_VREF_TYPE;

// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
```

```
// Start the AD conversion
ADCSRA|=(1<<ADSC);

// Wait for the AD conversion to complete

while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
return ADCW;
}

void main(void)
{
// Declare your local variables here

unsigned int Valeur1 , Valeur2 , a , i, Teta, Consigne, S ;
unsigned char Tampon[20];

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=Out Bit1=In Bit0=In

DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (1<<DDA2) | (0<<DDA1) |
(0<<DDA0);

// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=0 Bit1=T Bit0=T

PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) |
(0<<PORTA1) | (0<<PORTA0);
// Port B initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In

DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) |
(0<<DDB0);

// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T

PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) |
(0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In

DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) |
(0<<DDC0);
```

```
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T

PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
(0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In

DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) |
(0<<DDD0);

// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T

PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) |
(0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected

TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
(0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off

TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
(0<<CS10);
TCNT1H=0x00;
```

```
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected

ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) |
(0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off

MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled

UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) |
(0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off

ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);

// ADC initialization
```

```
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: ADC Stopped

ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) |
(0<<ADPS1) | (0<<ADPS0);
SFIOR=(1<<ADHSM) | (0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

// SPI initialization
// SPI disabled

SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) |
(0<<SPR0);
// TWI initialization
// TWI disabled

TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTC Bit 7
// RD - PORTC Bit 6
// EN - PORTC Bit 5
// D4 - PORTC Bit 4
// D5 - PORTC Bit 2
// D6 - PORTC Bit 1
// D7 - PORTC Bit 0
// Characters/line: 16

lcd_init(16);
while (1)
    {
     S = 0 ;
        for (a = 0 ; a<100 ; a ++)
            {
              i = read_adc (3) ;        // Lis la tension du capteur de température et la
                                           convertie en numérique
              S = S + i ;
            }
```

```
Valeur1 = S/100 ;
Teta = (int) (Valeur1*0.4882);        // Conversion du numérique vers des degrés
delay_ms (200);
Valeur2 = read_adc (0) ;              // Lis la tension du potentiomètre de commande et la
                                        convertie en numérique
Consigne = (int) (Valeur2*0.030);                    //Conversion du numérique vers des degrés
                                                       tout réglant la sensibilité du potentiomètre
                                                     pour 30°C
        if (Consigne > 30)
                Consigne = 30 ;              // On bloque la valeur de consigne à 30°C

sprintf (Tampon, "Il fait :") ;
lcd_gotoxy(0,0);
lcd_puts (Tampon) ;                                  // On affiche « Il fait : »
sprintf (Tampon, "        %4d C", Teta) ;
lcd_gotoxy(0,1);
lcd_puts (Tampon) ;                                  // On affiche la température lue par le capteur
sprintf(Tampon,"Vous voulez :");
lcd_gotoxy (0,2);
lcd_puts(Tampon);                                    // On affiche « Vous voulez : »
sprintf(Tampon,"        %4d C",Consigne);
lcd_gotoxy (0,3);
lcd_puts(Tampon);                                  // On affiche la température de consigne

if (Teta>=Consigne+1)                            // Si la température est supérieure ou égale à
                                                       celle de la consigne + 1
PORTA.2 = 1 ;                                     // Réfrigérateur en marche

if (Teta<=Consigne-1)                            // Si la température est inférieure ou égale à celle
                                                   de la consigne - 1
PORTA.2 = 0 ;                                        // Réfrigérateur à l'arrêt
}
}
```