



ÉCOLE TECHNIQUE DES MÉTIERS DE LAUSANNE

# 2226 REGTHERMIQUE

PROJET ETML-ES

---

Neziri Taulant

16th June 2023

# Contents

<b>1 But du projet</b>	<b>4</b>
<b>2 Spécifications du projet</b>	<b>4</b>
2.1 Schéma bloc . . . . .	4
2.2 Planification . . . . .	4
2.3 PCB(bloc 1 et 2) . . . . .	5
2.4 Capteur(bloc 3) . . . . .	5
2.5 Affichage + Régulation (bloc 4) . . . . .	5
2.6 E/S . . . . .	5
<b>3 Tâches à réaliser</b>	<b>6</b>
<b>4 Jalons principaux</b>	<b>6</b>
<b>5 Livrables</b>	<b>6</b>
<b>6 Convention de nommage et liens</b>	<b>7</b>
6.1 Stockage du fichier . . . . .	7
<b>7 Pré-étude</b>	<b>7</b>
7.1 Résumé du projet . . . . .	7
7.2 Schéma général du système . . . . .	7
<b>8 HMI (Human Machine Interaction)</b>	<b>8</b>
8.1 Croquis du boîtier . . . . .	8
<b>9 Spécifications (Choix technologique)</b>	<b>8</b>
9.1 Corps de chauffe . . . . .	8
9.2 Ecran LCD . . . . .	9
9.3 USB-UART (optionnel) . . . . .	9
9.4 Capteurs de température . . . . .	10
9.5 uC . . . . .	10
9.6 Boîtier . . . . .	11
<b>10 Estimations des coûts</b>	<b>11</b>
<b>11 Planification</b>	<b>11</b>
<b>12 Conclusion et perspectives</b>	<b>11</b>
<b>13 Conception schématique</b>	<b>12</b>
13.1 Schéma électrique . . . . .	12
13.2 Alimentations . . . . .	13
13.2.1 Dimensionnement . . . . .	13
13.3 HMI . . . . .	15
13.4 Puissance . . . . .	16
13.5 Régulation . . . . .	17
13.6 Capteurs de température . . . . .	18
13.7 uC . . . . .	19
13.8 Conclusion . . . . .	20
13.9 Modifications . . . . .	20
<b>14 Conception Hardware</b>	<b>21</b>
14.1 Contraintes . . . . .	21
14.2 Placement . . . . .	21
14.3 Caractéristiques (Routage) . . . . .	22

---

14.4 Montage . . . . .	24
14.5 Modifications . . . . .	24
<b>15 Software</b>	<b>25</b>
15.1 Introduction . . . . .	25
15.2 Concept logiciel . . . . .	25
15.3 Approche théorique (Flowchart) . . . . .	25
15.4 Configuration des périphériques . . . . .	27
15.5 Algorithme . . . . .	28
15.6 Etat d'avancement . . . . .	29
<b>16 Test et Mesures</b>	<b>30</b>
16.1 Schéma de mesure . . . . .	30
16.2 Liste de matériels . . . . .	30
16.3 Méthode de mesure . . . . .	30
16.4 Alimentation . . . . .	30
16.5 Conclusion . . . . .	31
16.6 Problèmes rencontrés . . . . .	31
16.7 Apport personnel . . . . .	31
16.8 Planification . . . . .	32
16.9 Journal de travail . . . . .	33
16.10 Listing (app.c, app.h, interrupt.c) . . . . .	34
16.11 Documents CAO . . . . .	34
16.12 Document de modification . . . . .	34
16.13 Mode d'emploi du système . . . . .	34
16.14 Affiche . . . . .	34
16.15 Résumé du projet . . . . .	35
16.16 Mode d'emploi du système . . . . .	36

## 1 BUT DU PROJET

Concevoir un système de régulation thermique dans un boîtier fermé pour les cours de REGL. L'utilisateur aura deux possibilités de réglages. La première pourra être effectuer via un générateur externe et des touches physiques (bouton, potentiomètre). La deuxième, se fera via USB.

## 2 SPÉCIFICATIONS DU PROJET

Voici les fonctions du projet représentés sous forme d'un schéma bloc.

### 2.1 SCHÉMA BLOC

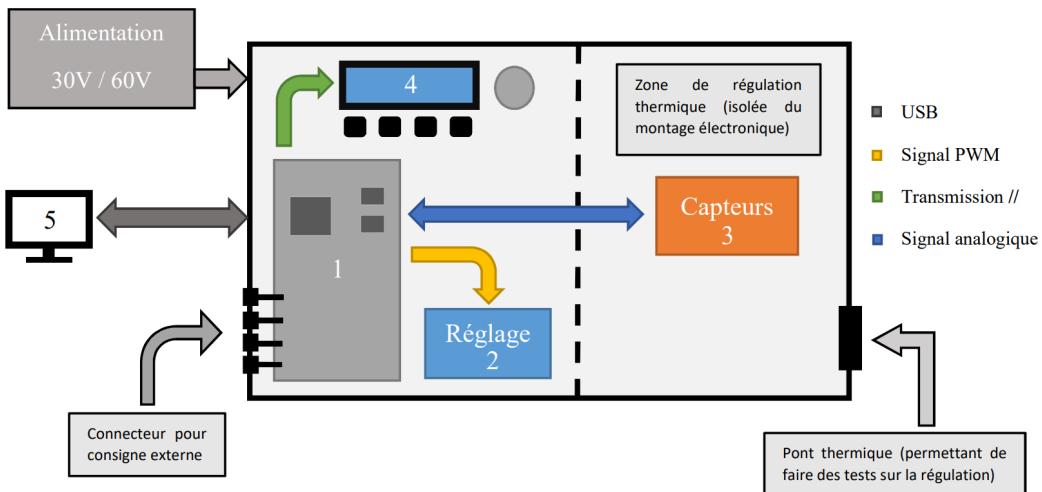


Figure 1: Schéma bloc système de régulation PID

L'utilisateur aura le choix entre deux modes :

**Mode 1 :** (Configuration physique (Hardware)) L'utilisateur pourra utiliser les touches physiques (boutons poussoirs, potentiomètre) pour régler les paramètres de la régulation PID.

**Mode 2 :** (Configuration à distance) A l'aide d'un câble USB, l'utilisateur pourra relier le module de régulation à un PC. Ce dernier pourra ensuite configurer les paramètres de régulations à l'aide d'une interface Web graphique.

### 2.2 PLANIFICATION

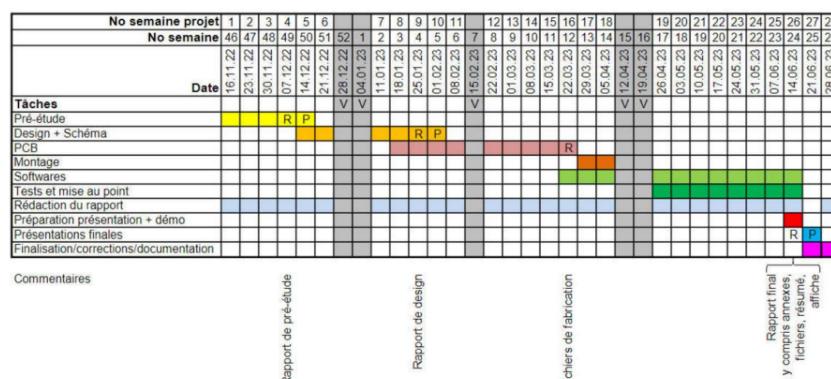


Figure 2: Planification d'exemple

## 2.3 PCB(BLOC 1 ET 2)

**Mode 1 :** L'utilisateur pourra s'il le désire fournir une consigne via à un générateur de fonction externe ou directement paramétrier la période, la forme et la valeur crête de la consigne générée en interne par le PIC32. La régulation se fera dans ce même PIC32 en fonction de la valeur mesurée par l'un des capteurs (voir bloc 2), l'utilisateur pourra paramétrier le gain des trois composantes PID ( $K_p$ ,  $K_i$ ,  $K_d$ ) via le bloc 4 afin de modifier la commande en sortie du µC.

**Mode 2 :** La consigne et le processus de régulation se feront via un script en python. Le PC recevra la valeur mesurée par l'un des capteurs (voir bloc 2) via USB et fournira la commande au système. Dans ce mode, le PIC32 est utilisé uniquement pour de l'affichage (voir bloc 4).

**Mode 1 : et Mode 2 :** Les blocs 1 et 2 feront varier la puissance dissipée (via la commande) au travers de la résistance de puissance pour modifier la température ambiante. La résistance de puissance sera dissipée dans la partie isolée du boîtier. Le débit d'air pourra être modifié par l'utilisateur via le bloc 4 en mode 1.

## 2.4 CAPTEUR(BLOC 3)

**Mode 1 : et Mode 2 :** Le bloc capteur viendra mesurer la température produite par la résistance de puissance via deux sondes de température. L'une se trouvera à l'extrême gauche du tube et l'autre sera près de la résistance de puissance. La valeur mesurée par une des sondes sera transmise soit au myRIO soit au PIC32 dépendant du mode.

## 2.5 AFFICHAGE + RÉGULATION (BLOC 4)

**Mode 1 :** L'écran pourra afficher différents menus gérés par des boutons et un encodeur incrémental. Ils permettront de régler le débit d'air du ventilateur, la consigne fournie par le PIC32 (forme, valeur crête et période du signal), les valeurs  $K_p$  /  $K_i$  /  $K_d$  et la source de la consigne (PIC32 ou générateur de fonction).

**Mode 2 :** La consigne, la commande et la mesure seront affichées sur un écran.

## 2.6 E/S

**HMI (Human machine Interface) :**

- ⇒ Une sortie analogique pour envoyer la consigne (pour affichage uniquement)
- ⇒ Une sortie PWM pour envoyer la commande
- ⇒ Une entrée analogique pour recevoir la mesure
- ⇒ Un port USB pour communiquer avec le PIC32 en USB/UART

**PIC32 :**

- ⇒ Une entrée analogique pour recevoir la consigne
- ⇒ Une sortie PWM pour envoyer la commande
- ⇒ Deux entrées analogiques pour recevoir la mesure
- ⇒ Une sortie PWM pour modifier le débit d'air du ventilateur
- ⇒ Une sortie PWM pour la commande de puissance du corps de chauffe
- ⇒ Des entrées numériques pour gérer les boutons et le commutateur rotatif
- ⇒ E/S numériques pour communiquer avec l'écran
- ⇒ Deux E/S numériques pour communiquer avec le PC en UART -> USB

### 3 TÂCHES À RÉALISER

1. Concevoir le cahier des charges avec le supérieur technique
2. Déterminer les composants électroniques et mécaniques du projet
3. Faire un rapport de pré-étude
4. Concevoir un schéma électrique
5. Faire les plan 3D sur Solidworks des pièces mécaniques
6. Router le PCB sur Altium via le schéma électrique préalablement conçu
7. Concevoir un montage mécanique du projet, usiner ou faire usiner les pièces
8. Commander le PCB, les composants électroniques et les composants mécaniques
9. Monter les composants électroniques et les pièces mécaniques indispensables
10. Programmer le PIC32
11. Tester la carte, mise en service
12. Monter le reste des composants mécaniques
13. Tester le montage final dans les 2 modes

### 4 JALONS PRINCIPAUX

Voir planification [I](#)

### 5 LIVRABLES

- Les fichiers sources de CAO électronique des PCB réalisés
- Tout le nécessaire à fabriquer un exemplaire hardware de chaque
- Fichiers de fabrication (GERBER) / liste de pièces avec références pour commande /implantation (prototype) / modifications / dessins mécaniques, etc
- Les fichiers sources de programmation microcontrôleur (.c / .h)
- Tout le nécessaire pour programmer les microcontrôleurs (logiciel ou fichier.hex)
- Le cas échéant, les fichiers sources de programmation PC/Windows/Linux.
- Le cas échéant, tout le nécessaire à l'installation de programmes sur PC/Windows/Linux.
- Un mode d'emploi du système
- Un calcul / estimation des coûts
- Un rapport contenant les calculs - dimensionnement de composants - structogramme, etc

## 6 CONVENTION DE NOMMAGE ET LIENS

Le nom de ce fichier doit être unique et doit donc contenir le nom du projet avec le format suivant :

*aaii\_nomProjet – CDC\_Vn.docx*

Avec :

- CDC : pour Cahier des charges
- aaii : numéro de projet, exemple 1708 pour projet de 2017 no 08
- nomProjet : comme son nom l'indique
- Vn: ou « n » indique la version du document

Exemple : 0910x<sub>P</sub>ICEthernet – CDCV1.docx

### 6.1 STOCKAGE DU FICHIER

Ce fichier sera stocké à la racine du dossier /doc d'un projet.

Ainsi, tous les fichiers de documentation faisant partie du projet sont centralisés dans le même répertoire.

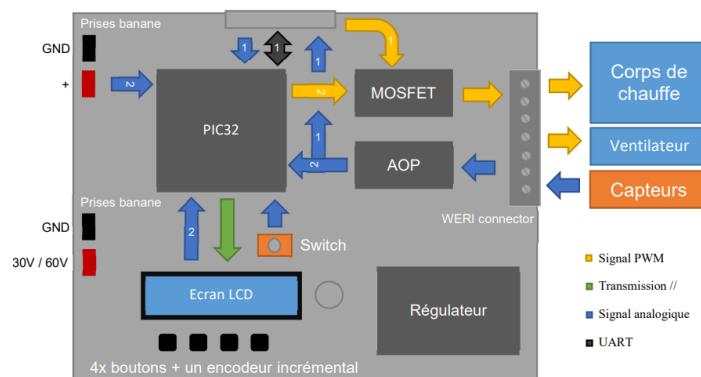
## 7 PRÉ-ÉTUDE

### 7.1 RÉSUMÉ DU PROJET

Basé sur le projet 2216\_RegThermiqueMyRio, dans le but de créer un régulateur de température PID compact et facile à utiliser. A la demande de M. Braun, seul le principe de régulation thermique PID devait être retenu. Le projet ayant comme objectif de fournir des outils pour les futurs cours de REGLAGES (REGL).

Le but du projet RegThermique est de créer un système de régulation thermique isolé. Dans l'intention d'avoir un outil pédagogique pour les cours de REGL. L'utilisateur devra pouvoir configurer/modifier manuellement les paramètres de régulation pour en observer les réactions.

### 7.2 SCHÉMA GÉNÉRAL DU SYSTÈME



\*1 signifie que la flèche est valable uniquement en mode 1  
 \*2 signifie que la flèche est valable uniquement en mode 2  
 Lorsqu'il n'y a rien, cela veut dire que la flèche est valable dans les deux modes

Figure 3: Schéma du système

Le schéma ci-dessus est une approximation des futures fonctionnalités possibles. Il a principalement pour but de démontrer les différents modes d'utilisation et les blocs étant utilisés dedans.

## 8 HMI (HUMAN MACHINE INTERACTION)

Le système sera alimenté par une alimentation de laboratoire 30-60[V]/ 0-3[A]. Le boîtier contiendra une interface avec écran LCD ainsi que des boutons, potentiomètre. Permettant à l'utilisateur de faire le réglage des paramètres de régulation.

L'utilisateur pourra allumer et éteindre l'appareil. Il pourra également régler le "set point" de la température, la vitesse du fan et comme cité précédemment, les consignes de régulation. Un écran LCD permettra d'afficher les différents modes (Manuel, USB) ainsi que tous les paramètres de régulation.

Un port USB sera également disponible dans le cas d'une éventuelle utilisation via interface WEB.

### 8.1 CROQUIS DU BOÎTIER

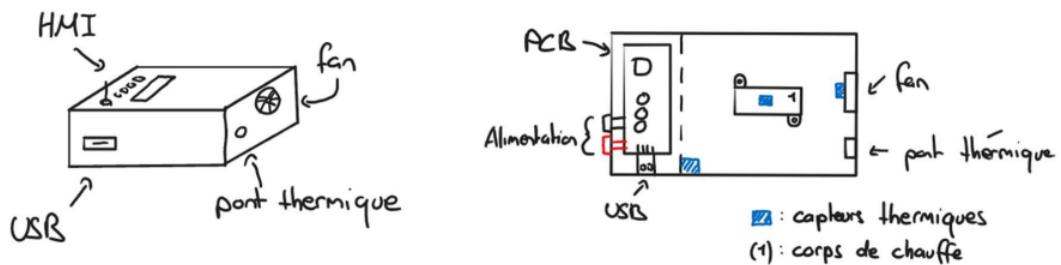


Figure 4: Croquis du boîtier

Le croquis de la Figure 4 représente mon approche sur la disposition des différents modules Hardware. Le dessin est arbitraire et ne représente pas entièrement tous les aspects de sa conception.

## 9 SPÉCIFICATIONS (CHOIX TECHNOLOGIQUE)

### 9.1 CORPS DE CHAUFFE

Afin de créer une régulation thermique, il est nécessaire d'avoir un corps de chauffe capable de fournir l'énergie recherchée. Selon la demande du C.d.c, le corps de chauffe doit être capable de réagir de manière rapide. Mon système peut être alimenté en 30-60 [V]/ 3 [A], souhaitant dissipé rapidement, je m'oriente sur une résistance de faible valeur : 5-10 [ $\Omega$ ].

Je détermine tout d'abord la puissance maximum délivrable par l'alimentation :

$$P = U * I >> 60 * 3 = 180[W] \quad (1)$$

$$P = U * I >> 30 * 3 = 90[W] \quad (2)$$

Suites à plusieurs tests effectués en atelier, j'ai conclu que les résistances de puissances avec corps en aluminium étaient à éviter en raison de leur inertie thermique.



(a) R. de puissance (corps alu)

(b) R. de puissance (corps céramique)

Figure 5: Résistance de puissance

Les résistances céramiques sont quant à elle plus réactive en raison de leur corps en céramique.

## 9.2 ECRAN LCD

J'ai défini durant la pré-étude ce que je souhaite afficher pour le système. Sur cette base, j'ai décidé de choisir un écran LCD 4x20. Le chef de projet ayant donnée une consigne sur l'emplacement du LCD (posé sur PCB), cet LCD me convient donc parfaitement.

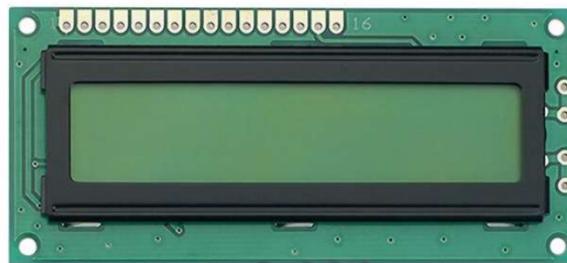


Figure 6: Ecran LCD (4x20)

Cet écran LCD consomme un courant maximum de 0.6 [mA] et doit être alimenté par une tension de 7[V] max. Ses caractéristiques conviennent donc au système.

## 9.3 USB-UART (OPTIONNEL)

Dans ce projet le deuxième mode permettra à l'utilisateur de connecter son PC afin de faire la configuration des paramètres de régulation.

Le port étant un USB 2.0 slave, un chip est nécessaire pour gérer la communication full-duplex. Ceci permettant la transmission des informations au uC.J'ai donc porté mon choix pour le CY7C64225- 28PVXC de chez infineon, le chip peut communiquer en Full Speed 12Mb/s.

L'utilisation d'un chip pour cette communication permet la simplification du design et du software



Figure 7: Transceiver UART

## 9.4 CAPTEURS DE TEMPÉRATURE

Pour effectuer une régulation, le système nécessite des capteurs de température précis et rapide. J'ai déterminer un capteur de  $0.5[^\circ\text{C}]$  ainsi que 10 bits de résolution convenable à mon application. Le capteur ayant un boîtier TO-92, les possibilité positionnement dans le boîtier sont larges.



Figure 8: Transceiver UART

## 9.5 uC

Afin de déterminer le microcontrôleur que je dois utiliser pour mon projet, il faut que je sache ce qu'il va devoir réaliser.

Dans le cadre de la régulation thermique, l'uC permettra de contrôler le corps de chauffe, le fan ainsi que les consignes PID de régulation. L'écran LCD sera également contrôlé par l'uC avec une communication parallèle.

Dans mon cas, le uC communique avec plusieurs modules en communication série :

- ⇒ Capteurs de température, fan : Signaux analogiques
- ⇒ (Port USB slave (IRH) : UART) optionnel

Le Micro contrôleur doit posséder des pin GPIO et plusieurs ports de communication série. Pour se faire, je suis allé rechercher les possibilités de micro contrôleurs que me propose le fabricant Microchip et voici quelques exemples de micro contrôleur que je pourrais utiliser :

- ⇒ PIC32MX370F512H (2-SPI/2-I2C/4-UART)
- ⇒ PIC32MX250F128D (3-SPI/4-I2C/6-UART)
- ⇒ PIC32MX795F512L (4-SPI/5-I2C/6-UART)

En vue de la situation actuelle dans le monde des semi-conducteurs, M.Bovey, m'as proposé de choisir le **PIC32MX250F128D** car c'est un 44 pins ayant toutes les fonctionnalités nécessaires pour le projet omis un convertisseur DAC(qui sera en externe).



Figure 9: Micro contrôleur PIC32

## 9.6 BOÎTIER

En raison des hautes températures que le boîtier atteindra, le choix du matériau est un aspect crucial au bon fonctionnement du système. L'aluminium est directement retiré des possibilités dû à une conduction thermique élevée.

Recherchant un matériau résistant à des températures de plus de 100[°C] et une isolation thermique élevée. Le boîtier sera en plastique type : ABS, Silicone, etc...



(a) Boîtier semi transparent



(b) Boîtier ABS

Figure 10: Résistance de puissance

## 10 ESTIMATIONS DES COÛTS

Nom	Quantité	Type	Fabriquant	N° Fabriquant	Fournisseur	Prix u	Total
Microcontrôleur	1	PIC32MX795F512L	Microchip	No stock	No stock	fr. 13,00	fr. 13,00
Capteurs thermique	3	MAX31820PARMCR+ND	Maxim Integrated	MAX31820PARMCR+	Digi-Key	fr. 4,55	fr. 4,55
Ecran LCD	1	DEM 20485 SYH	Display Elektronik	DEM 20485 SYH	Distrelec	fr. 21,20	fr. 21,20
Corps de chauffe (résistance)	2	Résistance de puissance	-	-	DEV-BOS	fr. 1,50	fr. 3,00
Conv. USB-UART (optionnel)	1	727-CY7C64225-28PVXC	Infenion	CY7C64225-28PVXC	Mouser	fr. 4,30	fr. 4,30
Boîtier	1	ABS	-	-	-	fr. 30,00	fr. 30,00
						<b>Total</b>	fr. 76,05

Figure 11: Coûts estimé

## 11 PLANIFICATION

Voir annexes Points : ????

## 12 CONCLUSION ET PERSPECTIVES

Pour conclure, cette étape de pré-étude a permis de fixer les composants principaux du système de régulation. Durant la rédaction du rapport, plusieurs questions ont été soulevées tous au long de la recherche de composants. Ces questions suscitent plusieurs inconnues qui seront nécessaire d'aborder durant la phase de design/conception.

En terme de faisabilité, la pré-étude démontre que les composants primordiaux sont en stock et délivrable dans un délai acceptable. Ces délais ont été jugé "acceptable" selon la planification globale du projet.

Pour finir, ce projet présente de bonnes solutions en ce qui concerne la méthode de régulation choisie (thermique isolée). Le boîtier reste cependant un point à approfondir dû aux aspects de maintien et dissipation thermique de ce dernier

## 13 CONCEPTION SCHÉMATIQUE

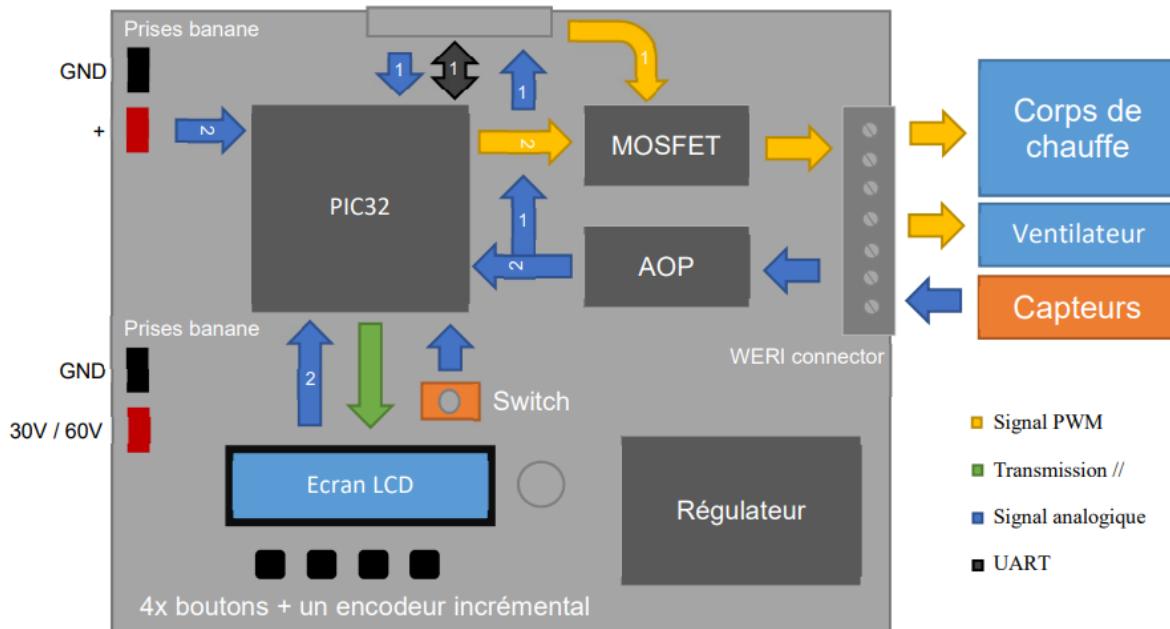
Sur la base du cahier des charges précédemment défini et approuvé, la conception du montage de régulation doit être effectuée. Voici une description du produit attendu qui sera résumé en trois points : principe, caractéristiques, schéma bloc.

**Principe :** Le principe de ce produit est de fournir un outil/support pour le cours de REGLAGE. Ce dernier se basera sur le principe de régulation thermique. Des capteurs de températures seront utilisés afin de récupérer la valeur de température. Le système de réglage PID contrôlera une résistance de puissance ainsi qu'un ventilateur afin de réguler la température à sa valeur de consigne.

**Caractéristiques :** Voici les caractéristiques principales du système :

- Plage d'action de la température : 20 [°C] -> 50 [°C]
- Précision de senseurs de températures : 0.2 -> 0.5 [°C]
- Vitesse de régulation : ordre de la minute (1min -> 5min)

**Schéma bloc :** Afin de mieux représenter le fonctionnement du système, un schéma bloc contenant les composants/fonctions principales de chaque "modules" à été fait.



\*1 signifie que la flèche est valable uniquement en mode 1  
 \*2 signifie que la flèche est valable uniquement en mode 2  
 Lorsqu'il n'y a rien, cela veut dire que la flèche est valable dans les deux modes

Figure 12: Schéma bloc système

### 13.1 SCHÉMA ÉLECTRIQUE

Le schéma électrique complet se trouve en annexe. Voir figure : ???

## 13.2 ALIMENTATIONS

Le montage sera alimenté par l'alimentation de laboratoire qui délivre du 0-60V/0-3A. J'ai défini une alimentation +15[V]/-15[V] étant nécessaire pour le fonctionnement du montage. L'utilisateur devra alimenter le montage en utilisant les câbles banane.

Des régulations de tension sont nécessaires pour tout ce qui est IC logique et micro contrôleur. Une régulation 5V et 3V3 sera implémentée dans la schématique.

Pour éviter de parasiter les lignes de tensions, je sépare les niveaux de tension 5V et 3V3 avec deux régulateurs différents. Sur la base de la tension d'entrée Vin (15 [V]), j'utilise des régulateurs 15V -> 5V et 3V3.

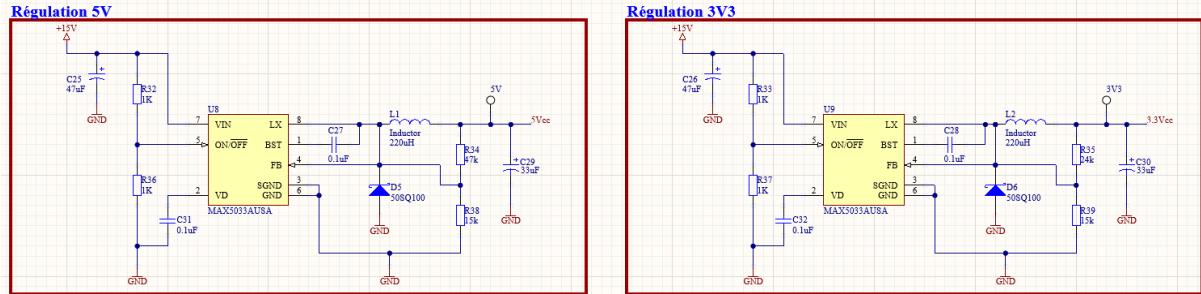


Figure 13: Alimentation 5V/3.3V

### 13.2.1 DIMENSIONNEMENT

Pour réguler la tension, j'ai décidé d'utiliser des régulateurs **MAX5035** qui sont des convertisseurs DC-DC en mode step-down.

**Régulation 5[V]** : Le **MAX5035** propose plusieurs topologies qui ont chacune des tensions de sorties différentes. Ce composant propose également un mode de tension de sortie ajustable(1.25[V] -> 13.2[V]). Je souhaitais utiliser une version avec une tension de sortie fixe @5[V], malheureusement après recherches, le composant n'était plus disponible sur le marché. J'ai donc opté pour la version permettant d'ajuster la tension de sortie.

Afin d'ajuster la tension, un dimensionnement est nécessaire. Pour ce faire, j'ai utilisé la "marche à suivre" fournie par le datasheet. Le dimensionnement se trouve en page **10** du datasheet : MAX5035 Datasheet

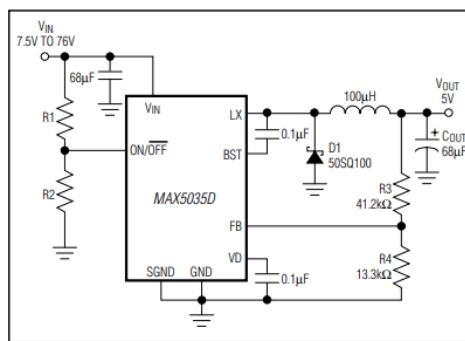


Figure 14: MAX5035 datasheet (adjustable output)

Pour trouver les résistances R3 et R4 servant à fixer la sortie, j'ai utilisé la formule suivante :

$$R3 = \frac{(V_{OUT} - 1.22)}{1.22} * R4 \quad (3)$$

La résistance R4 est définie de manière empirique à une valeur égal ou inférieur à 15 [kΩ] selon les directives du datasheet. J'obtiens des valeurs de résistances de l'ordre de :

Valeur de **R4** : 15 [kΩ]

$$R3 = \frac{(5 - 1.22)}{1.22} * 15k = 46.47k - > 48[k\Omega] \quad (4)$$

Le choix de l'inductance est guidé par la différence de tension entre  $V_{OUT}$  et  $V_{IN}$ . Le but est de trouver la valeur minimum de l'inductance qui est donnée par :

$$L = \frac{(V_{IN} - V_{OUT}) * D}{0.3 * I_{OUTMAX} * f_{SW}} \quad (5)$$

Ou le D est vaut :

$$D = \frac{V_{OUT}}{V_{IN}} = \frac{5}{15} = 0.\overline{33} \quad (6)$$

$f_{SW}$  représente la fréquence de fonctionnement(125kHz selon datasheet) et le courant  $I_{OUTMAX}$  est défini par rapport aux composants étant alimenté par le régulateur 5[V]. Dans notre cas, les composants étant alimentés en 5[V] sont :

- LCD (courant de back light) = 20[mA]
- Pull-up (ligne DAC) = négligeable de l'ordre du [uA]

Le courant peut donc être fixé à une valeur de **100[mA]** (marge prise par mesure de sécurité). Nous obtenons donc une valeur d'inductance :

$$L = \frac{(15 - 5) * 0.\overline{33}}{0.3 * 100m * 125k} = 888u - > 890[uH] \quad (7)$$

Les valeurs des condensateurs de découplages,filtrages ainsi que les résistances R1 et R2 ont été choisis selon les valeurs recommandées du datasheet.

### Régulation 3.3[V] :

Pour la régulation 3.3[V], les MAX5035 avec tension de sortie fixe à 3.3[V] furent disponible sur le marché. Cependant, afin faciliter le dépannage ainsi que la commande des composants, un MAX5035 à sortie ajustable sera également utilisé.

Le dimensionnement de ce dernier est identique à la régulation du 5[V]. Quelques paramètres diffèrent comme le courant de sortie maximum et la tension de sortie. La procédure de dimensionnement étant déjà faite, le tout est simplifié en un tableau des valeurs définies et calculées :

Composants	Val.calculé	Val.final
R1	1K	1K
R2	1K	1K
R3	25.57kΩ	24kΩ
R4	15kΩ	15kΩ
$I_{OUTMAX}$	500[mA]	500[mA]
D	0.22	0.22
L	137.28[uH]	143[uH]
$P_{max}$	5.85[W]	6[W]

Table 1: dim.Régulation 3.3[V]

Composants	Val.calculé	Val.final
R1	1K	1K
R2	1K	1K
R3	46.47kΩ	48kΩ
R4	15kΩ	15kΩ
$I_{OUTMAX}$	100[mA]	100[mA]
D	0.33	0.33
L	888[uH]	890[uH]
$P_{max}$	1[W]	1[W]

Table 2: dim.Régulation 5[V]

Les données présentes dans la colonne sur-ligné en vert, représentent la valeur finale définie pour les composants de dimensionnement du régulateur. (**Le choix des régulateurs a été modifié suite à la revue de la pré-étude**)

### 13.3 HMI

L'interface humain-machine est composée des composants suivants:

- Écran LCD
- Boutons-poussoirs
- Codeur incrémental

Les boutons-poussoirs ont été définis en mode pull-up et seront reliés à la carte via des fils de connexions. Les boutons seront fixés directement sur le boîtier. Le codeur incrémental quant à lui sera brasé directement sur le PCB. Un codeur horizontal sera choisi afin de faciliter son implémentation dans un boîtier qui sera également fixé sur le boîtier. L'écran LCD quant à lui sera visible à travers le boîtier lui permettant d'être directement brasé sur le PCB.

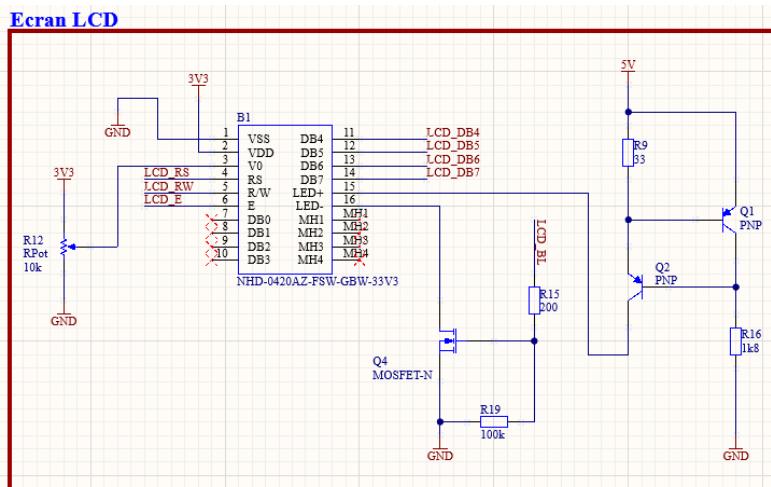


Figure 15: Bloc LCD (Limitation de courant)

La source de courant est basée sur un projet antérieur utilisant le même écran LCD. Les deux transistors PNP servent de source de courant alors que le MosFet permet de contrôler le back light de l'LCD depuis l'uC.

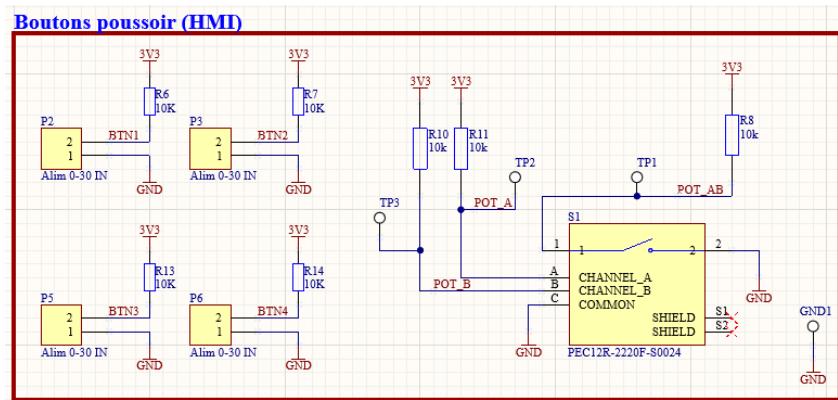


Figure 16

En ce qui concerne les composants électromécanique(boutons, compteur incrémental), des résistances de pull-up ont été ajoutées. La valeur des ces résistances a été déterminée de manière empirique (valeur typique de pull-up).

### 13.4 PUISSANCE

L'étage de puissance est constitué de deux composants significatifs au projet de régulation thermique. Une résistance de puissance étant alimentée à la tension d'entrée (15V) et un ventilateur alimenté en 5 V. Un signal PWM contrôle les MosFet, qui agissent en tant que « switch » actif, pour les deux composants du module de puissance.

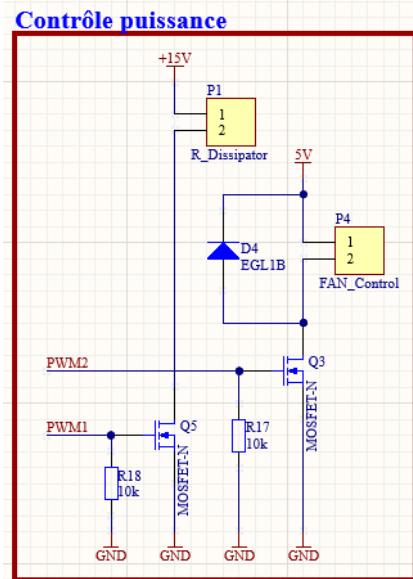


Figure 17: Bloc Puissance

#### Dimensionnement :

Dans le schéma de la figure [17], le corps de chauffe (R) et le ventilateur sont remplacé par des connecteur berg puisqu'ils ne feront pas parti du PCB principal. La diode **D4** est une diode de protection de polarité(en cas d'inversion du courant). Pour le module de puissance, deux MosFet différents seront utilisés. Le MosFet commandant la résistance de puissance doit être capable de soutenir le courant important traversant le drain. Pour ce faire il est nécessaire de déterminer ce courant Id ainsi :

#### Résistance de puissance :

- Tension (U) : **15 [V]**
- Résistance (R) : **10[ $\Omega$ ]**
- Courant ( $I_d$ ) : **1.5 [A]**
- Puissance dissipé (sur R) : **22.5 [W]**

#### Ventilateur :

- Tension U : **5 [V]**
- Courant ( $I_d$ ): **190 [mA]**
- Puissance dissipé (sur Ventilateur) : **950 [mW]**

Le courant Id maximum est donc de **1.5 [A]**. Cette valeur permet de définir la caractéristique principale du mosfet. Mosfet choisi :

**Mosfet canal N : IRLML6344TRPBF ( $P_{max} : 1.3[W]$ )**

### 13.5 RÉGULATION

La régulation PID du système est faite de manière analogique ainsi que numérique. Les trois consignes PID sont contrôlées par un système analogique d'amplificateur commun. L'erreur ainsi que la consigne seront contrôlées par l'uC. Une partie de la régulation (Gestion de la consigne et erreur) sera effectuée de manière numérique à travers le uC. L'amplificateur commun a été choisi selon ses spécifications et la précédente utilisation dans des montages à régulation. Sa disponibilité sur le marché a également été un critère dans le choix du composant. J'ai décidé d'opter pour des **TL052ACD** afin de pouvoir les alimenter de manière symétrique selon les alimentations (+15[V] et -15[V]).

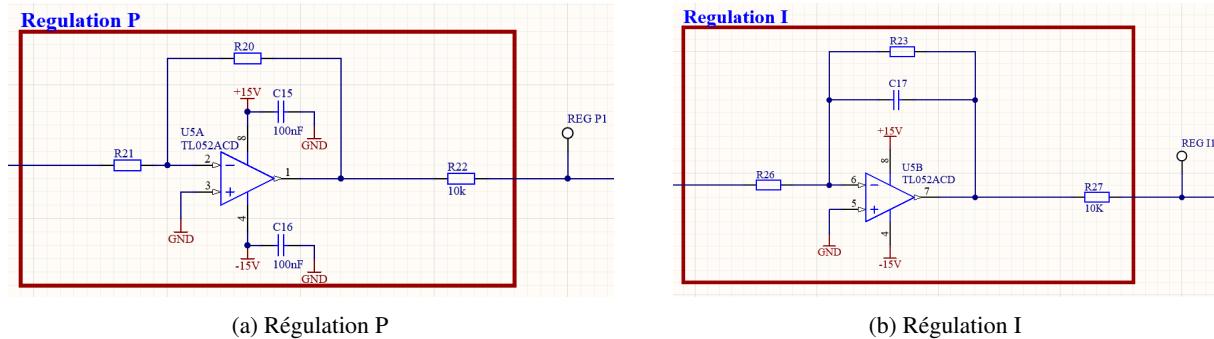


Figure 18: Régulation P et I

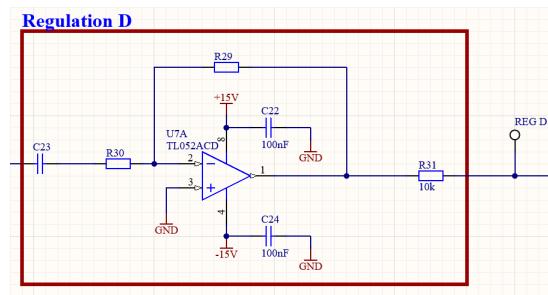


Figure 19: Régulation D

Le montage de chaque type de régulation (PID), a été fait sur la base des cours de **REGL** ainsi que d'exemple de schéma de régulation présent sur internet. Les trois signaux sont injecter à l'entrée d'un sommateur (qui en inverse le signal) puis d'un inverseur pour le rendre positif.

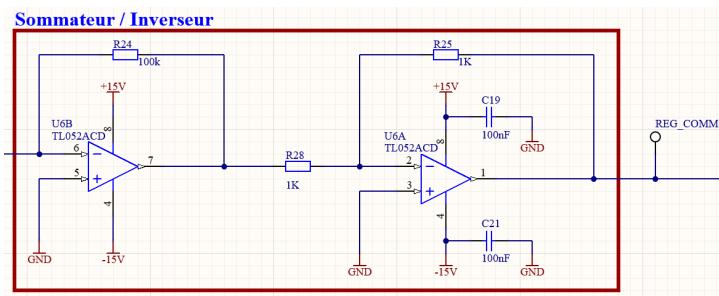


Figure 20: Régulation Sommateur/Inverseur

Les condensateurs ainsi que les composants de rétro-action du sommateur et inverseur ont été fixés. Ceci de manière théorique (Montage sommateur/ inverseur). Les résistances et condensateurs de chaque module de régulation n'ont pas été dimensionnés. Ce système de régulation ayant pour but d'être un outil pédagogique, les résistances et condensateurs seront remplacés par des connecteurs clamp ou berg pour faciliter la modification des paramètres (composants) de régulation.

### 13.6 CAPTEURS DE TEMPÉRATURE

Les capteurs de températures ont été choisis selon les caractéristiques suivantes :

- Taille du capteur
- Donnée/Communication analogique
- Grande plage de température

Les **MAX6612MKK** possèdent une plage de température allant de  $-55[C] + 150[C]$  avec une précision de  $+/-1.2[C](maximum)$ . La précision du capteur est suffisante pour son application dans le montage. Sa disponibilité sur le marché et son prix bas font un candidat intéressant.

Voici sa fonction de transfert :

#### Transfer Function

The temperature-to-voltage transfer function has a linear positive slope and can be approximated by the equation:

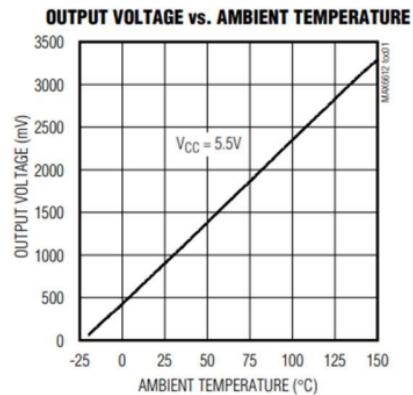
$$V_{OUT} = 0.40V + (0.01953V/\text{°C} \times T\text{°C}) - (2 \times 10^{-6} \times T\text{°C}^2)$$

where T is the MAX6612's die temperature in °C.

Therefore:

$$T\text{ (°C)} = (V_{OUT} - 0.40V) / 0.01953V/\text{°C}$$

(a) Fonction de transfert



(b)  $U_{OUT}$  en fonction de temp.

Figure 21: Donnée du datasheet MAX66112MKK

La sortie des capteurs sera traitée par une entrée analogique du micro contrôleur. Elle correspond à la valeur d'erreur dans le système de régulation. Les capteur seront monté sur une PCB à part afin de pouvoir directement placer ces dernier dans les zones à mesurer.

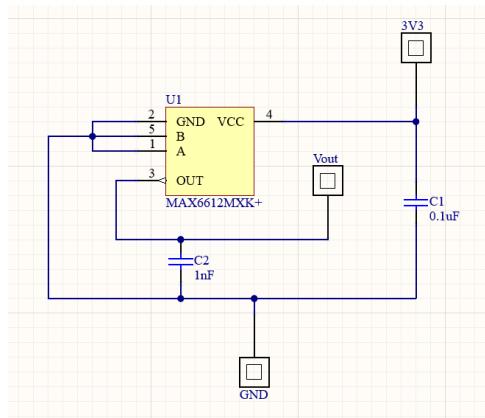


Figure 22: Schématique des capteurs de températures

L'application ne nécessitant pas un mode spécifique, je me suis basé sur le **Typical Application Circuit** fourni par le datasheet.

**Datasheet :** MAX6612MKK datasheet

## 13.7 uC

Le choix du microcontrôleur a été défini selon toutes les entrées/sorties nécessaires pour le bon fonctionnement du montage.

### Param. externes :

- Capteur de température : 3x entrée analogique
- Contrôle de puissance : 2x signaux PWM

### HMI :

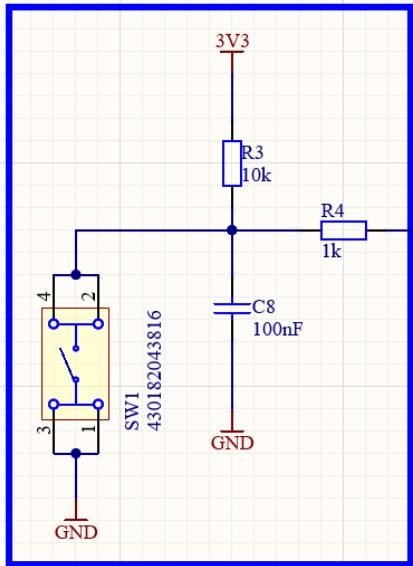
- 11x I/O
- 3x entrée analogique

### Régulation PID :

- 1x entrée analogique
- 1x sortie analogique

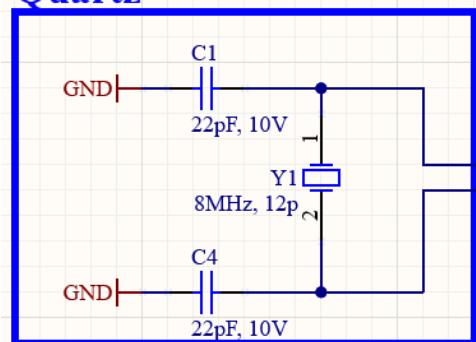
Le micro contrôleur **PIC32MX250F128D** correspond presques à toutes les entrées/sorties nécessaires pour le fonctionnement du système. Cependant, cet uC ne possède pas de sortie analogique(DAC) pour l'envoi du signal corrigé (Consigne- erreur). La configuration minimum au fonctionnement de l'uC (Découplage, alimentation USB) a été faite sur la base du datasheet PIC32MXXX (voir annexes).

### Reset



(a) Bloc reset

### Quartz



(b) Quartz externe

Figure 23: Module uC

Le bloc **uC**, contient tout les composants nécessaire au bon fonctionnement du micro contrôleur. Le choix des condensateurs de découplage a été fait sur la base du datasheet du fournisseur(MicroChip). Des sous-modules ont été défini d'un cadre bleu dans les figures [23] ci-dessus, ils permettent la programmation, le reset et l'ajout d'un clock externe.

Le quartz est un ajout préventif et le bloc du reset à pour but de faciliter la phase de design software en ayant une mise à zéro physique(Module inspiré par le kit PIC32 de l'ES). Veuillez trouver le schéma complet du module uC en annexe.

## 13.8 CONCLUSION

Durant cette phase de conception, plusieurs modules ont dû être modifiés pour satisfaire le cahier des charges. Plusieurs points restent tout de même critiques. Notamment le choix de la résistance de dissipation de chaleur reste un point important et doit être vérifié avec le chef de projet. Certains composants n'étant plus disponibles, ils doivent être modifiés pour assurer une future commande de ces derniers.

Actuellement, la schématique nécessite encore le choix de bons footprint et composants. Les composants constituant l'HMI et la partie corps de chauffe doivent corréler avec le futur boîtier dans lequel ils devront être installés.

**Perspectives:** La prochaine étape consistera à terminer le schéma électrique (avec footprint et composant livrable) pour ensuite enchaîner sur le design mécanique. En parallèle s'effectuera le design du PCB. Je dois étudier le cas critique (délai de livraison des composants) de mon projet pour avancer sur le PCB et le boîtier. Le but étant de démarrer l'algorithme software à temps.

## 13.9 MODIFICATIONS

- LCD : faire une rotation de 180° (afin d'avoir l'LCD dans le sens de lecture)
- Potentiomètre(LCD) : pin une et deux du footprint inversé par rapport au schéma
- Transistor Mosfet N: pin une et deux du footprint inversé par rapport au schéma

Veuillez trouver le document de modification en annexes.

## 14 CONCEPTION HARDWARE

### 14.1 CONTRAINTES

Aucunes contraintes majeures n'as été imposé selon le C.d.c. Seul la taille maximum du PCB pouvait être considéré comme condition. Le but étant de faire le PCB le plus petit possible afin d'économiser sur le coût de ce dernier.

### 14.2 PLACEMENT

Pour le placement des mes composants, j'ai procédé par bloc. C'est-à-dire que le PCB à été fait sur la base du schéma électrique séparé en différent blocs (uC, HMI, Puissance, etc...).

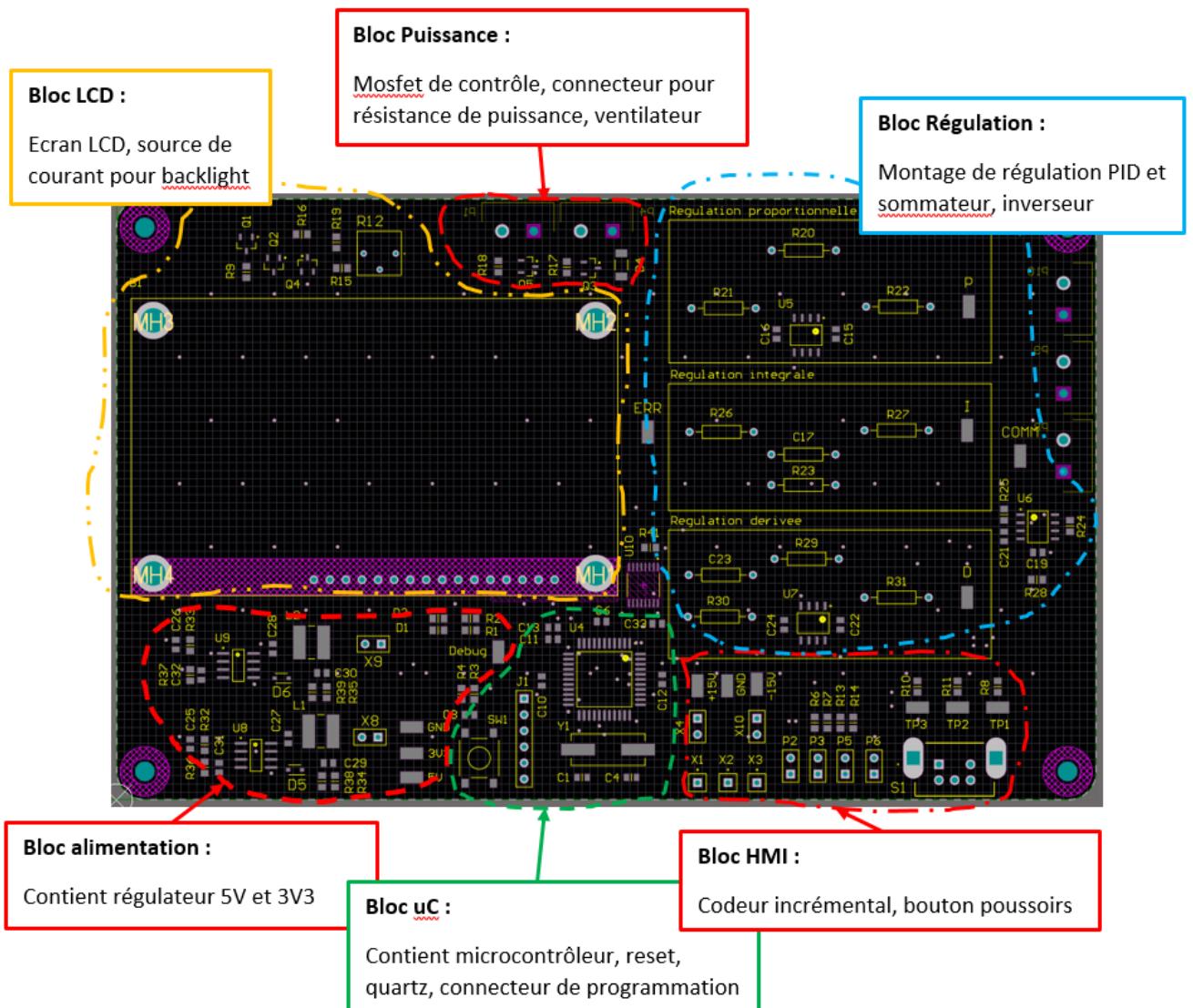


Figure 24: Placement des composants sur le PCB

La méthode de placement modulaire(par bloc) permet, si bien effectué, de faciliter le routage du PCB. Elle facilite également le dépannage des composants ces derniers étant groupé dans la même zone.

Le bloc de régulation assez large, ceci à raison de faciliter d'usage. Les utilisateurs devant modifier la valeur des gains PID, l'accès et l'utilisation de la zone de régulation dois être accessible aisément.

### 14.3 CARACTÉRISTIQUES (ROUTAGE)

Pour le routage du PCB plusieurs paramètres ont dû être fixé afin d'éviter des problèmes de puissances dissipée, brasage des composants,etc...

**Voici la liste des paramètres modifiés :**

Largeur de piste :

- Net POWER : 0.5[mm] (Min Preferred width) 1.5[mm] (max width)
- Net classic : 0.254[mm] (Min Preferred width) 0.3[mm] (max width)

Vias :

- Via Diameter : 0.61[mm](Min), 0.8[mm](Max), 0.65[mm](Pref)
- Via Hole Size : 0.25[mm](Min), 0.45[mm](Max), 0.3[mm](Pref)

Stitching :

- Grid : 10[mm]
- Via style : Diameter = 0.65[mm], Hole Size = 0.3[mm]
- Via template : v65h30

Autres :

- Track clearance : 0.151[mm](Min) (Consigne donnée par chef de projet)

Un plan de masse commun à été placé sur le Top et le Bottom du PCB. Cela permet de réduire le bruit sur les lignes d'alimentation et sur les signaux en général. Pour éviter des effet capacitif, un stitching reliant les deux plans (Top, Bot) est recommandé.

**Vue PCB :**

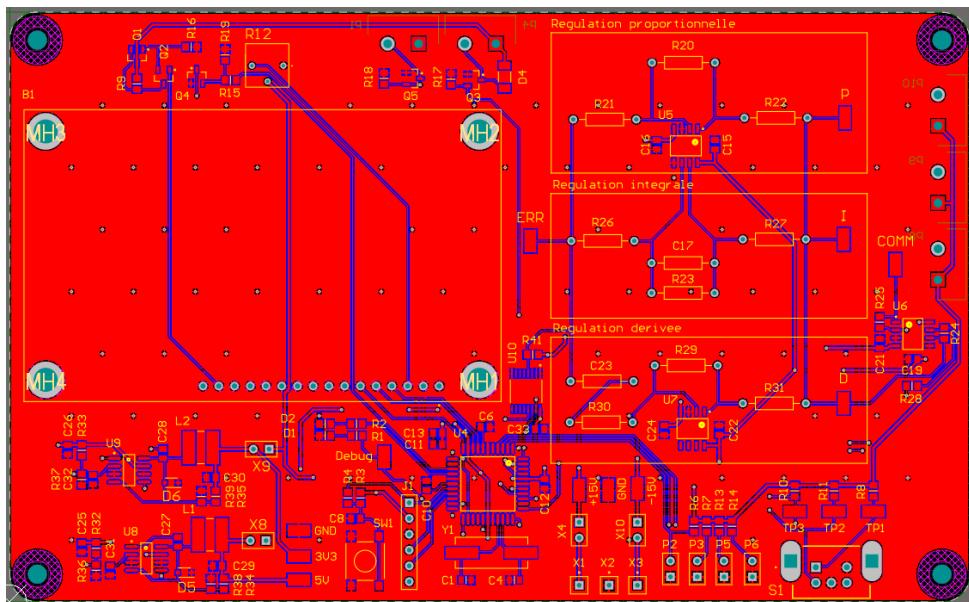


Figure 25: PCB vue Top

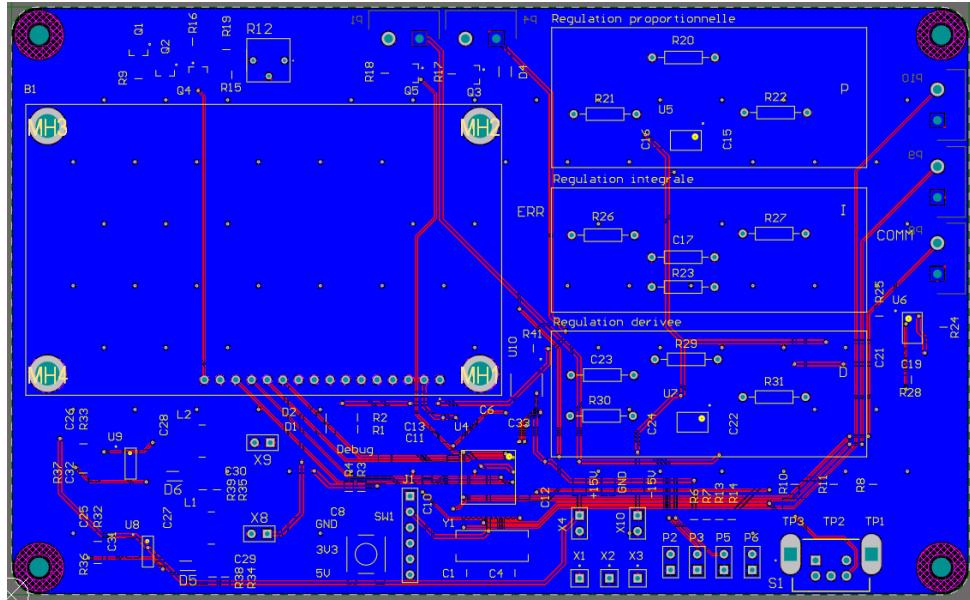


Figure 26: PCB vue Bottom

Le routing et placement du montage n'a nécessité aucune démarche supplémentaire. N'ayant pas eu de contraintes au niveau du PCB, le bon placement des composants était l'unique phase "critique" de la conception hardware.

**Voici les vue 3D des différent PCB :**

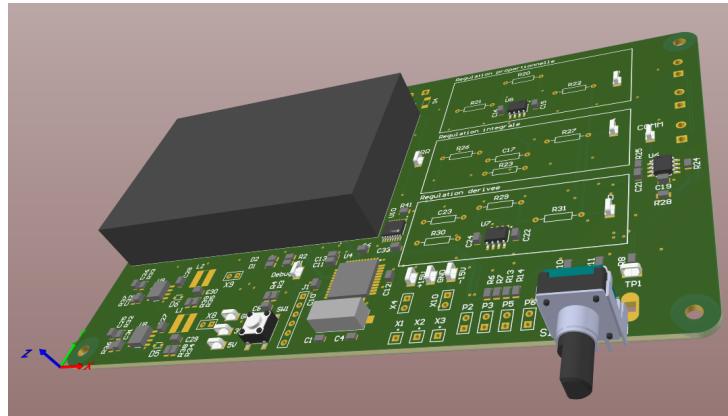


Figure 27: PCB principal vue 3D



(a) PCB résistance de puissance

(b) PCB capteur de temp.

Figure 28: Vue 3D des PCB

## 14.4 MONTAGE

Une fois tous les composants, PCB reçus, le montage des différentes cartes électroniques était à faire. Le montage des composants a été effectué de manière méthodique.

Les blocs d'alimentations ont été montés en premier. Le but étant de pouvoir tester le bon fonctionnement de ces derniers avant la suite du montage. A noter que les alimentations sont séparées du montage principale par des jumpers. Ce procédé permet une double sécurité. Ensuite les blocs uC et régulation ont été montés. Cette fois également, le uC devait être programmable avant de poursuivre dans le montage des composants. Pour finir, pour des raisons de contraintes de brasure, l'écran LCD ainsi que l'étage de puissance ont été brasés.

## 14.5 MODIFICATIONS

Suite à la réception du PCB ainsi que du montage des composants, plusieurs erreurs ont été décelées. Il s'agit principalement d'erreurs de footprints. Une liste des modifications critiques se trouve ci-dessous. Un document de modification sera également annexé et rédigé de manière plus formelle, précise.

### Footprint :

- LCD : faire une rotation de 180° (afin d'avoir l'LCD dans le sens de lecture)
- Potentiomètre(LCD) : pin une et deux du footprint inversé par rapport au schéma
- Transistor Mosfet N: pin une et deux du footprint inversé par rapport au schéma

Veuillez trouver le document de modification en annexes.

## 15 SOFTWARE

### 15.1 INTRODUCTION

La partie software est la phase finale du projet. Une fois la conception schématique et hardware effectuée, un algorithme doit être établis sur la base d'un concept logiciel. Ce concept doit faire correspondre les demandes du cahier des charges afin de pouvoir mener à bien le produit.

### 15.2 CONCEPT LOGICIEL

Une première approche du concept logiciel a été faite lors du design schématique. Afin d'aborder le sujet de manière plus approfondie, une version plus détaillée a été réalisée.

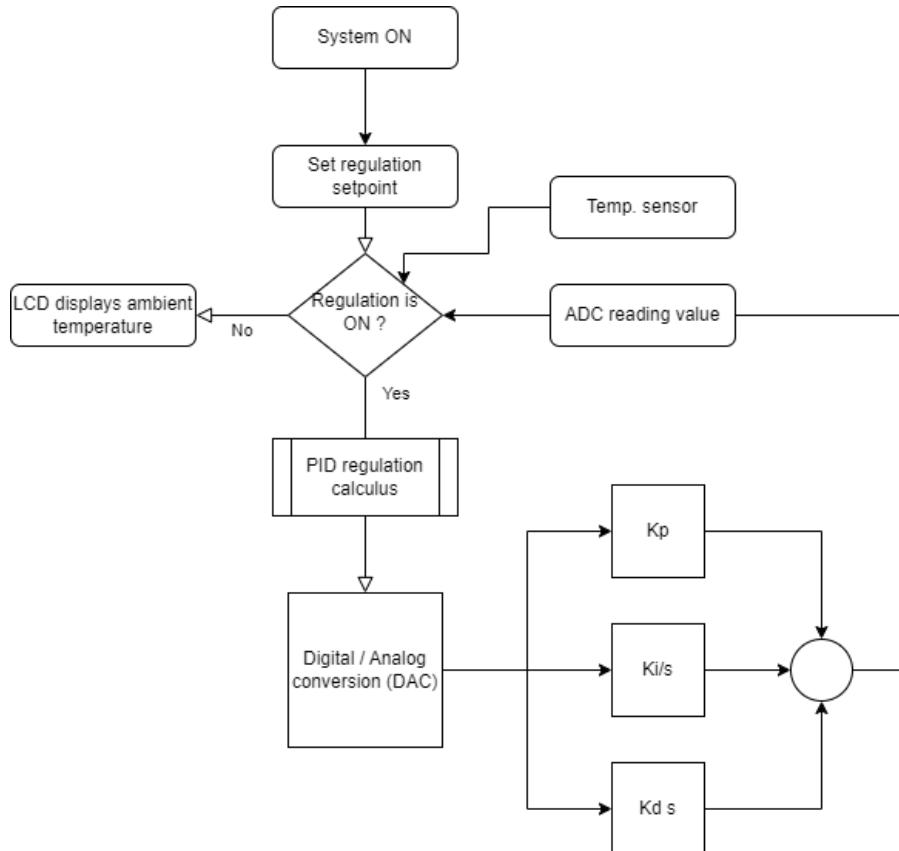


Figure 29: Concept logiciel détaillé

Le flowchart présenté à la Figure [31], permet d'avoir une base sur laquelle s'appuyer pour l'écriture des structogrammes/flowcharts du futur algorithme.

### 15.3 APPROCHE THÉORIQUE (FLOWCHART)

Mon approche est la suivante :

Premièrement, le software du produit de régulation thermique sera divisé en plusieurs sous modules. Le projet contenant plusieurs périphériques divers, je décide de séparer les différents éléments de la manière suivante :

- Interface human machine (HMI) : LCD, PEC12, boutons
- Gestion ADC / DAC
- Régulation PID (Gestion OC (PWM))

Les différents périphériques seront traités dans l'ordre de lecture de la liste. Cette première représentation de l'algorithme (pseudo-code) sera faite sous forme de flowchart.

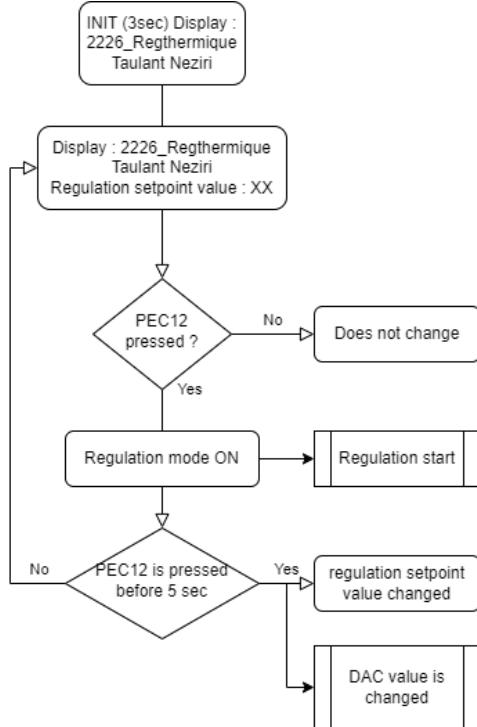


Figure 30: Flowchart HMI

La gestion du HMI est assez explicite. On choisit d'activer ou non la régulation thermique avec le codeur incrémental. Dans le cas où le mode est actif, on peut régler la valeur de consigne du système toujours à l'aide du PEC12. La valeur de température est affichée constamment dès la fin de l'initialisation.

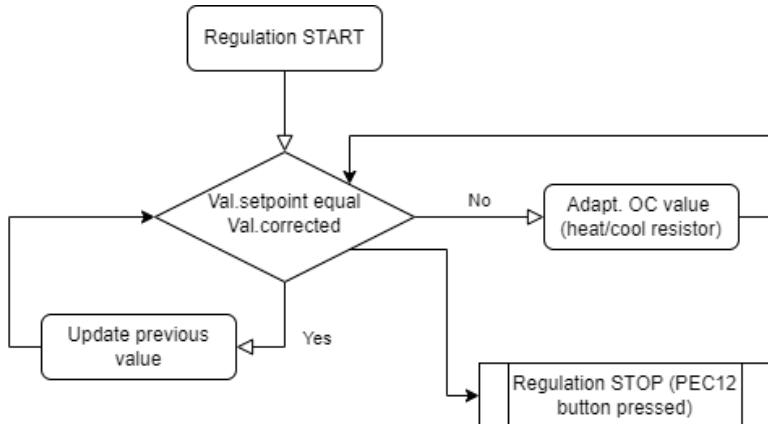


Figure 31: Flowchart régulation PID

Dès lors de son activation, la régulation est constante. Jusqu'à sa stabilisation, les correcteur PID feront varier les valeurs de l'OC (PWM) qui contrôle la résistance de puissance ainsi que le ventilateur. La pression maintenue du PEC12 stop la régulation à la manière d'une interruption.

## 15.4 CONFIGURATION DES PÉRIPHÉRIQUES

Pour le bon fonctionnement du software, les timers, ADC et le pinout du uC doivent être configuré. Pour cela il est important de connaître la précision recherché ainsi que la vitesse d'exécution de chaque tâches.

L'IDE MPLABX utilisé pour nos micro-contrôleur PIC32 possède une extension de configuration nommé **Harmony**. Ce dernier propose un type de configurateur graphique permettant de faciliter l'attribution des pins selon leur emplacement physique.

**Voici la configuration du PIC32 :**

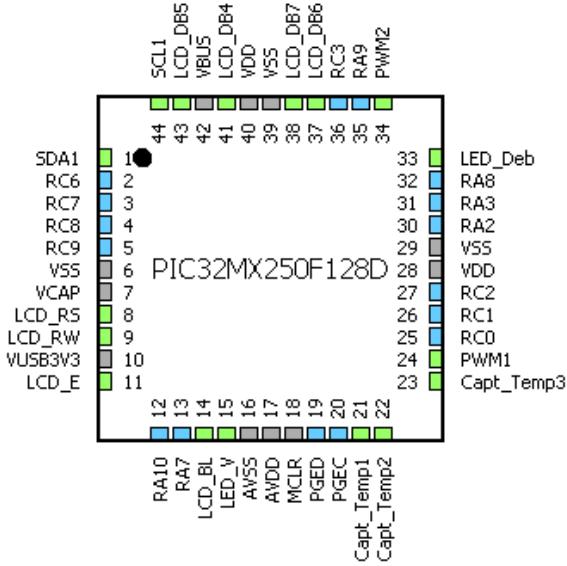


Figure 32: Config. Harmony uC

Mon application nécessite l'usage de timer, ADC et OC. Vous trouverez ci-dessous les calculs permettant de configurer ces derniers ainsi que la justification du choix. La fréquence système du uC( $f_{SYSfreqCLK}$ ) est de 40[MHz].

**Timers :** Trois timers sont utilisé dans cette configuration. Le premier sert à cadencé le code principale (machine d'état globale) et les deux autres serviront de référence pour les outputs capture. Le calcul de la valeur de comptage se fait avec la formule suivante :

$$Countperiod = \frac{f_{SYSfreqCLK}}{\text{prescaler} * \frac{1}{\text{Interrupt}_freq}} - 1 \quad (8)$$

**timer 1 :** Ce timer servira à cadencer l'application principale. Il est fixé à une valeur de 1[kHz] soit 1[ms] de période. Cette valeur est choisie sur la base des travaux pratique effectué en cours et pour raison de bonne pratique. Il pourra être ralenti si nécessaire à l'aide de condition.

**timer 2-3 :** Les timers 2-3 sont des timers de paire. Ils peuvent servir comme base de temps à but d'interruption ou encore comme compteur pour les modules OC et IC. Dans mon cas je configure les deux timers séparément car les PWM de contrôle (R.puissance ventilateur) seront diriger de manière indépendante par la régulation.

Timers	Frequence	Prescaler	Periode.Count
timer1	1[kHz]	64	624
timer2	10[kHz]	1	3999
timer3	10[kHz]	1	3999

Table 3: Timer Config.

La valeur de prescaler des timers 2 et 3 à été choisie de sorte à avoir la précision de comptage la plus grande. Cela permettra un ajustement plus fin lors de la régulation PID.

**OC :** Comme cité précédemment, deux OC sont utilisé pour la gestion du module de puissance. Ils ont comme ID : OC2 et OC3 et se base sur le timer du chiffre respectif( $timer2 > OC2, timer3 > OC3$ ).

OC	Periode.Count
OC2	1999
OC3	1999

Table 4: OC Config.

Les périodes de count ont été fixé à la moitié de période count de leur timer. Les fixant donc à une valeur de 50 [%] de rapport cyclique. Cette valeur est arbitraire est permet de tester la sortie des OC. La valeur de la période count sera modifiée lors de la régulation PID.

**ADC :** L'ADC est un module présent dans le uC. C'est un convertisseur 10bits, ce qui veut dire que nous avons une plage de comptage allant de 0 à  $2^{10}$ (1024).

**DAC :** Le **PIC32MX250F128D** ne possédant pas de DAC intégré, un convertisseur digital analogique externe à été choisi pour la valeur de consigne du système de réglage. Ce dernier communique en **SPI**. La communication série ne nécessitant pas une grande vitesse, l'utilisation d'une interruption n'est pas utile. Des libraires fournie par le cours de MINF sont disponible pour les communications en SPI.

## 15.5 ALGORITHME

En ce qui concerne l'implémentation du code, uniquement les points important seront traité dans ce sous-chapitre.

```
void APP_Tasks ( void )
{
    float freq = 0;

    /* Check the application's current state. */
    switch ( appData.state )
    {
        /* Application's initial state. */
        case APP_STATE_INIT:
        {
            // Inits LCD
            lcd_init();
            lcd_b1_on();
            lcd_gotoxy(1, 1);
            printf_lcd("Input Capture");
            // Starts Timers
            DRV_TMRO_Start();
            DRV_TMRL_Start();
            // Starts IC
            DRV_IC0_Open();
            // Updates SM
            APP_UpdateState(APP_STATE_SERVICE_TASKS);
            break;
        }

        case APP_STATE_SERVICE_TASKS:
        {
            // Temperature sensor config.
            freq = 1/appData.period;
            lcd_gotoxy(1, 4);
            printf_lcd("freq = %0.3fHz", freq);

            APP_UpdateState(APP_STATE_WAIT);
            break;
        }

        case APP_STATE_WAIT:
        {
            break;
        }

        /* The default state should never be executed. */
        default:
        {
            /* TODO: Handle error in application's state machine. */
            break;
        }
    }
}
```

(a) Programme principal

```
S_ADCConvert ValConvert(void)
{
    // Variable declaration
    float Uin_1,Uin_2,Uin_3;
    float ValTemp;
    float ValRawed;

    //Raw value Struct;
    S_ADCResults ValRaw;
    //Conversion value Struct;
    S_ADCConvert ValConv;

    // Raw Value from ADC read
    ValRaw = BSP_ReadAllADC();

    //ValRawed = ValRaw;

    // Raw value to Tension conversion
    Uin_1 = (float)ValRaw.Chan0 / (float)ADC_SIZE;
    Uin_2 = (float)Uin_1 * (float)VAL_REFADC;
    Uin_3 = Uin_2;

    // Put Voltage value in struct
    ValConv.Val_U = Uin_3;

    // Tension value to temp conversion
    ValTemp = (Uin_2 - VALU)/VALDEGR;

    // Put Temperature value in struct
    ValConv.Val_T = ValTemp;

    return ValConv;
}
```

(b) Fonction de conversion ADC

Figure 33: Algorithme

## System interrupt

```

void __ISR(_TIMER_1_VECTOR, ipl2AUTO) IntHandlerDrvTmrInstance0(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_1);

    static int count= 0;

    if(count < 3000)
    {
        count++;
    }
    else
    {
        APP_UpdateState(APP_STATE_SERVICE_TASKS);
    }
}

void __ISR(_TIMER_2_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance1(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_2);
}

void __ISR(_TIMER_3_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance2(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_3);
}

```

Figure 34: Fichier d'interruption

**Main :** Le code principale utilise une machine d'état avec les états : *APP – INIT*(Initialisation), *APP – STATE – SERVICE – TASKS* (Etat d'application du code) et enfin l'état *APP – STATE – WAIT* qui est un état d'attente. Le main est cadencé par l'interruption à une fréquence de 1[kHz](peut être ralenti).

**Fonction de conversion :** Les librairies ADC ont été inspirées du kit de l'ETML avec le PIC32MX795F512L. Sur cette base j'ai ajouter une fonction de conversion de la valeur analogique brute en température. Pour ce faire, la formule présentée dans la figure [14] doit être implémentée en code. Sur les directives du datasheet, des calcules mathématiques sont effectués.

**Fichier d'interruption :** Le fichier d'interruption contient l'attente de 3 secondes faite à l'initialisation du code. Les timers 2 et 3 servant aux OC, ne sont pas modifié dans ce fichier.

## 15.6 ETAT D'AVANCEMENT

Le software du produit n'est pas fini à l'heure du rendu de ce rapport. L'état d'avancement est basé sur les délivrables attendu dans le cahier des charges. Les points principaux de la partie software sont les suivants :

- Interface human machine (HMI) : 100 %
- Gestion ADC : 90 % (la fonction de conversion nécessite quelques ajustement sur la valeur obtenue)
- Gestion DAC : 40 % (Les libraires SPI ont été implémenté mais ne sont pas encore fonctionnels avec le uC utilisé)
- OC : 100 % (Les OC émettent des PWM à la fréquence configurée)
- Régulation PID (Gestion OC (PWM)) : 20 % (Le code de régulation est en partie implémenté mais il manque l'algorithme de régulation ainsi que les fonctions de conversion DAC/ADC)

Pour résumé, les périphériques du systèmes sont presque tous fonctionnels. Cependant, afin de pouvoir implémenter de manière structurée et correct le code de régulation, il est nécessaire d'avoir des algorithme entièrement fonctionnel au niveau des périphériques. Niveau d'avancement global (Software) **60 %**.

## 16 TEST ET MESURES

L'avancement du projet niveau software n'ayant pas atteint les jalons voulu, la phase de test et mesures n'a pas été faite de manière complète. Les alimentations en revanche ainsi que les lignes de communications LCD ont été mesurées.

### 16.1 SCHÉMA DE MESURE

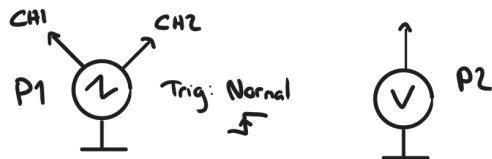


Figure 35: Schéma de mesure

### 16.2 LISTE DE MATERIELS

- P1 Oscilloscope : RohdeSchwarz RTB2004 2.5 GSA/s ES.SLO2.05.01.14
- P2 Multimètre : Gw INSTEK ES.SLO2.00.00.79

### 16.3 MÉTHODE DE MESURE

Pour la méthode de mesure, les sondes d'oscilloscope ou les pointes de touches pour le multimètre ont été placé sur les points de tests disponible.

### 16.4 ALIMENTATION



Figure 36: Mesure sur la ligne 3V3

On remarque un mauvais filtrage sur la ligne qui pourrait être provoqué par un mauvais dimensionnement des capacités de filtrage ou encore de l'inductance du montage de régulation.

Une mauvaise manipulation lors des mesures, à provoquer un court-circuit sur l'alimentation 5[V]. Ne me permettant pas d'effectuer une bonne mesure de son fonctionnement. Aucun autres composants n'a été affecté par le c-c (jumper alimentation débranché).

## 16.5 CONCLUSION

En conclusion, cette pré-étude a permis d'identifier les principaux composants du système de régulation thermique. Malgré quelques problèmes rencontrés liés à la disponibilité des composants, des alternatives ont été trouvées. Les contributions personnelles ont été précieuses pour le dimensionnement des régulateurs de tension et la recherche de solutions.

Pour les perspectives, il est essentiel de poursuivre la phase de conception en abordant les problèmes liés au boîtier et à la gestion thermique. De plus, des tests approfondis seront nécessaires pour valider le bon fonctionnement du système de régulation. Ces efforts permettront d'optimiser le système et d'assurer son efficacité à long terme.

## 16.6 PROBLÈMES RENCONTRÉS

Durant la rédaction du rapport, j'ai rencontré plusieurs problèmes, notamment concernant la disponibilité des composants nécessaires pour la régulation thermique. Certains composants initialement envisagés n'étaient plus disponibles sur le marché, ce qui m'a obligé à revoir le choix et à opter pour des alternatives. Cela a nécessité des recherches supplémentaires pour trouver des composants équivalents.

Un autre problème rencontré concerne la dimensionnement des régulateurs de tension. Nous avons dû suivre les procédures de dimensionnement fournies par les datasheets des composants, ce qui a demandé du temps et des ajustements pour obtenir les valeurs adéquates.

## 16.7 APPORT PERSONNEL

Dans le cadre de ce projet, j'ai pu contribuer en apportant des connaissances et une compréhension approfondie des principes de régulation thermique. J'ai également participé activement à la recherche de composants de remplacement lorsque certains n'étaient plus disponibles sur le marché.

De plus, j'ai pu mettre en pratique mes compétences en dimensionnement des régulateurs de tension en suivant les procédures recommandées par les datasheets des composants.

Enfin, j'ai participé à la rédaction du rapport en soulevant des questions pertinentes et en proposant des suggestions pour améliorer le système de régulation.

## 16.8 PLANIFICATION

Date	No semaine projet												No semaine																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28									
Tâches																																					
Pré-étude	Theorique																																				
Design + Schéma	Réel																																				
PCB	Theorique																																				
Montage	Réel																																				
Software	Theorique																																				
Tests et mise au point	Réel																																				
Rédaction du rapport	Theorique																																				
Préparation présentation et démo	Réel																																				
Présentation finales	Theorique																																				
Finalisation/corrections/documentation	Réel																																				

Figure 37: Planning effectif

## 16.9 JOURNAL DE TRAVAIL

### AGENDA

DATE	HEURE	ÉVÉNEMENT
16.nov		Prise de connaissance du projet
23.nov		Discussion avec M.Braun pour valider les demandes du cahier des charges
30.nov		Ecriture du la pré-étude
07.déc		Finalisation et rendu de la pré-étude
14.déc		Présentation de la pré-étude et début de la conception schématique (design)
21.déc		Correction de la pré-étude, avancement de la schématique
28.déc		VACANCES
04.janv		VACANCES
11.janv		Design des blocs "externes" bloc : puissance, capteur
18.janv		Design du uC et de son montage minimum nécessaire, bloc alimentation et connecteurs
25.janv		Rendu du rapport de design, correction de certaines erreurs schématiques, recherche de footprint
01.févr		Recherches de footprint, correction du dimensionnement de certain composants (bloc alimentation, HMI)
08.févr		Présentation de la partie conception schématique, début de la correction (selon points indiqué pendant la présentation)
15.févr		VACANCES
22.févr		Correction du schéma électrique selon les consignes donnée pendant la présentation, recherche des derniers footprints
01.mars		Conception du PCB (placement des composants), recherche du boîtier en parallèle
08.mars		Placement terminé, début du routage
15.mars		Correction du PCB, placement modifier, création de la Bom afin de contrôler tout les footprints, routage annulé
22.mars		Finition du PCB, correction des erreurs dans le DRC, validation avec EuroCircuit, commande du PCB
29.mars		Réception et des composants, organisation pour le montage, PCB début du montage
05.avr		Commande des PCB externe (capteur, corps de chauffe) et composants ext. Comme : ventilateur
12.avr		VACANCES
19.avr		VACANCES
26.avr		Création du projet soft, communication avec PIC32 fonctionnelle, finitions du PCB principal
03.mai		Configuration Harmony des périphériques (timer, OC,ADC)
10.mai		Recherche de la panne sur le LCD
17.mai		Misen forme de l'affichage LCD
24.mai		Lecture ADC fonctionnel mise en forme des valeur reçues
31.mai		Ecriture de la communication SPI pour le DAC + Rapport
07.juin		Ecriture de la communication SPI pour le DAC + Rapport

Figure 38: Journal de travail

**16.10 LISTING (APP.C, APP.H, INTERRUPT.C)**

**16.11 DOCUMENTS CAO**

**16.12 DOCUMENT DE MODIFICATION**

**16.13 MODE D'EMPLOI DU SYSTÈME**

**16.14 AFFICHE**

## 16.15 RÉSUMÉ DU PROJET

La pré-étude a permis de définir les composants principaux du système de régulation. Des questions restent en suspens, nécessitant une approche lors de la phase de conception. La faisabilité du projet est confirmée, avec les composants disponibles dans des délais acceptables. Le choix de la régulation thermique est prometteur, mais des aspects liés à la dissipation thermique du boîtier doivent être approfondis.

La conception du montage de régulation repose sur le principe de régulation thermique avec des capteurs de température, un système de réglage PID et des composants spécifiques. Les caractéristiques principales incluent une plage de température de 20 à 50 °C, une précision des capteurs de 0.2 à 0.5 °C et une vitesse de régulation de 1 à 5 minutes.

L'alimentation du montage sera assurée par une alimentation de laboratoire avec des régulations de tension spécifiques pour les composants logiques. Des régulateurs MAX5035 en mode step-down sont utilisés, avec des valeurs de résistances et d'inductance déterminées pour réguler les tensions de 5V et 3.3V respectivement.

L'interface homme-machine comprend un écran LCD, des boutons-poussoirs et un codeur incrémental. Les composants électromécaniques sont équipés de résistances de pull-up pour leur fonctionnement.

L'étage de puissance comprend une résistance de puissance et un ventilateur, contrôlés par des MosFet activés par un signal PWM. Les composants sont dimensionnés en fonction des tensions et courants requis.

La régulation du système combine des éléments analogiques et numériques, avec des amplificateurs opérationnels, une gestion de la consigne et de l'erreur effectuée par l'uC. Les schémas des régulations P, I et D sont réalisés, avec les composants et condensateurs fixés.

En résumé, ce projet vise à concevoir un système de régulation thermique basé sur la méthode PID, avec des composants spécifiques dimensionnés pour assurer une régulation précise de la température. L'interface homme-machine et l'étage de puissance sont également pris en compte dans la conception.

## **16.16 MODE D'EMPLOI DU SYSTÈME**

**Titre :** Mode d'emploi du Montage de Régulation Thermique

**Introduction :** Le Montage de Régulation Thermique est un outil conçu pour le cours de Réglage. Il utilise le principe de régulation thermique avec des capteurs de température, un système de réglage PID, une résistance de puissance et un ventilateur pour maintenir la température à sa valeur de consigne. Ce mode d'emploi vous guidera à travers les étapes nécessaires pour utiliser le produit de manière efficace.

### **Étape 1: Alimentation**

- ❾ Branchez l'alimentation de laboratoire fournissant du 0-60V/0-3A au montage.
- ❾ Utilisez les câbles banane pour connecter l'alimentation +15V/-15V aux bornes appropriées du montage.

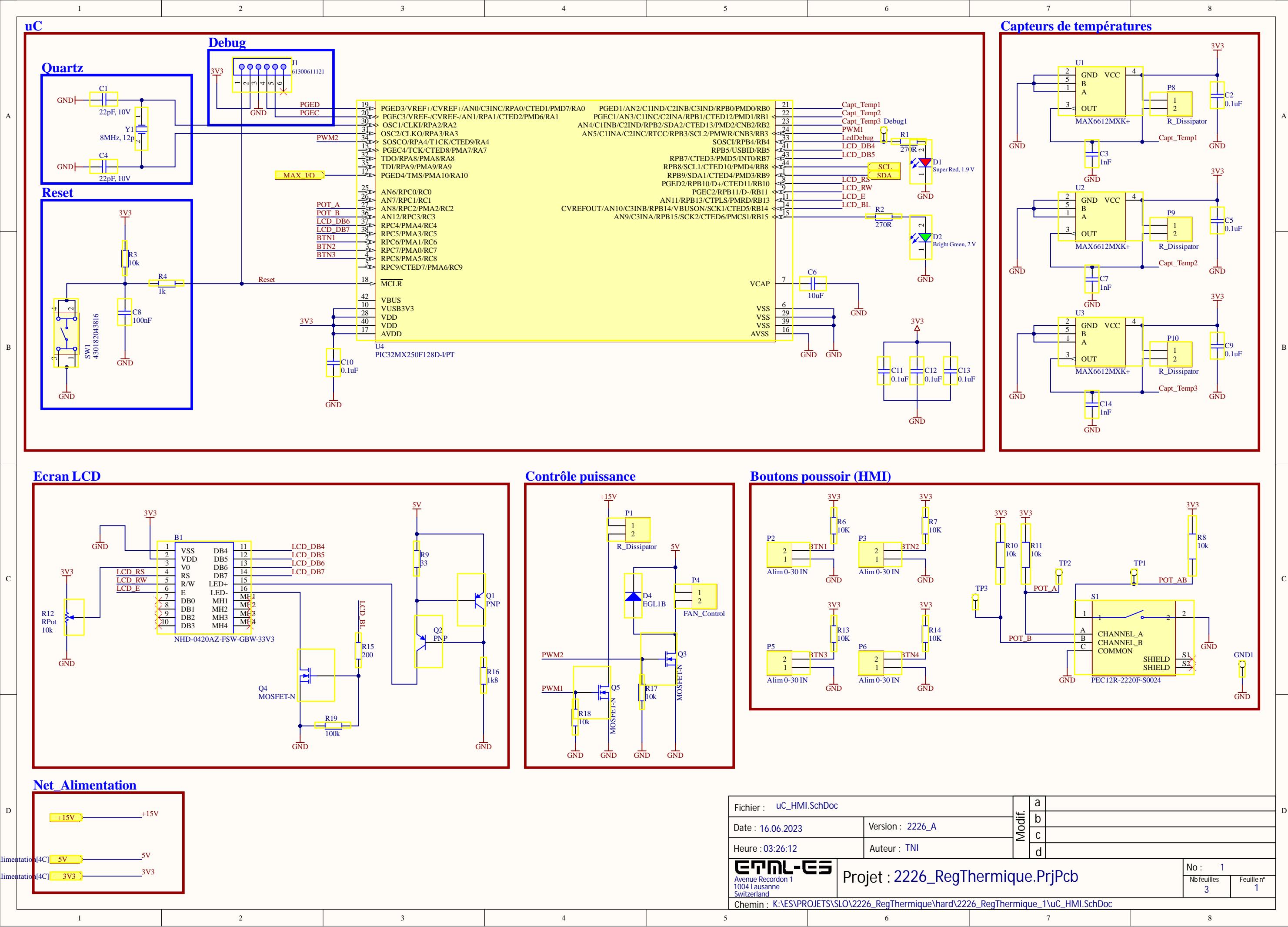
### **Étape 2: Interface Humain-Machine (HMI)**

L'écran LCD affiche les informations de température et les paramètres de réglage. Le codeur incrémental et les boutons-poussoirs permettent de naviguer dans les menus et de modifier les valeurs. Le codeur incrémental est utilisé pour ajuster les paramètres de manière précise.

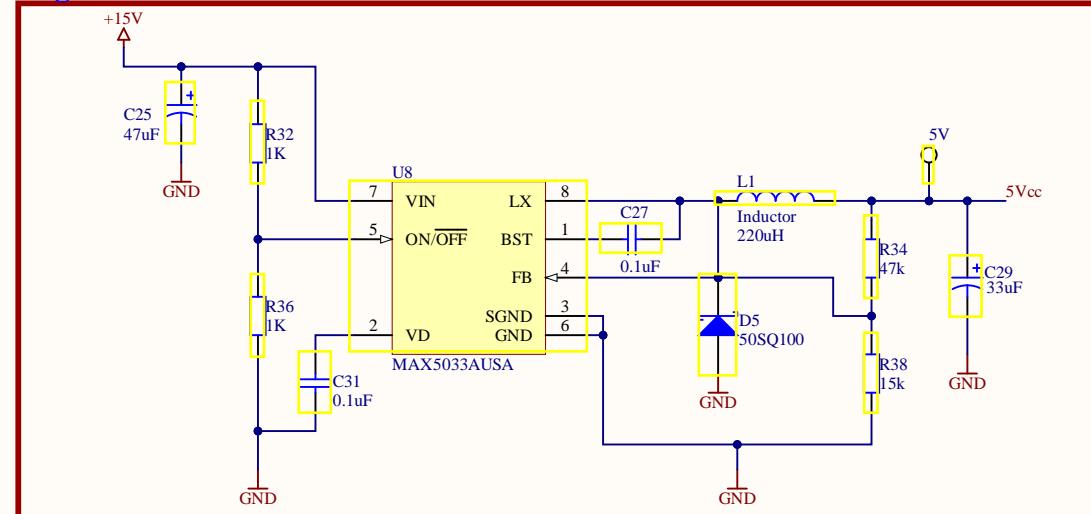
### **Étape 3: Réglage de la Température**

Vérifiez que les capteurs de température sont correctement connectés au montage. Sur l'écran LCD, utilisez le codeur incrémental pour régler la température de consigne dans la plage de 20°C à 50°C avec une précision de 0.2°C à 0.5°C. Le système de réglage PID ajuste automatiquement la résistance de puissance et le ventilateur pour maintenir la température à la valeur souhaitée.

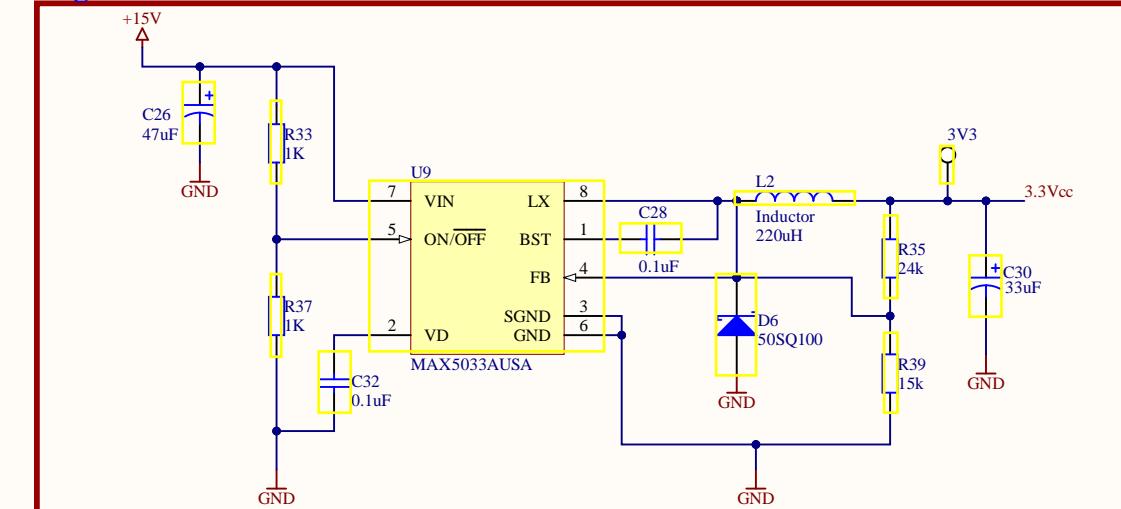
**Conclusion :** Le Montage de Régulation Thermique est un outil puissant pour le cours de Réglage, offrant des fonctionnalités de régulation thermique précises. En suivant les étapes décrites dans ce mode d'emploi, vous serez en mesure d'utiliser le produit de manière optimale et d'explorer ses différentes fonctionnalités. Profitez de votre expérience d'apprentissage avec le Montage de Régulation Thermique !



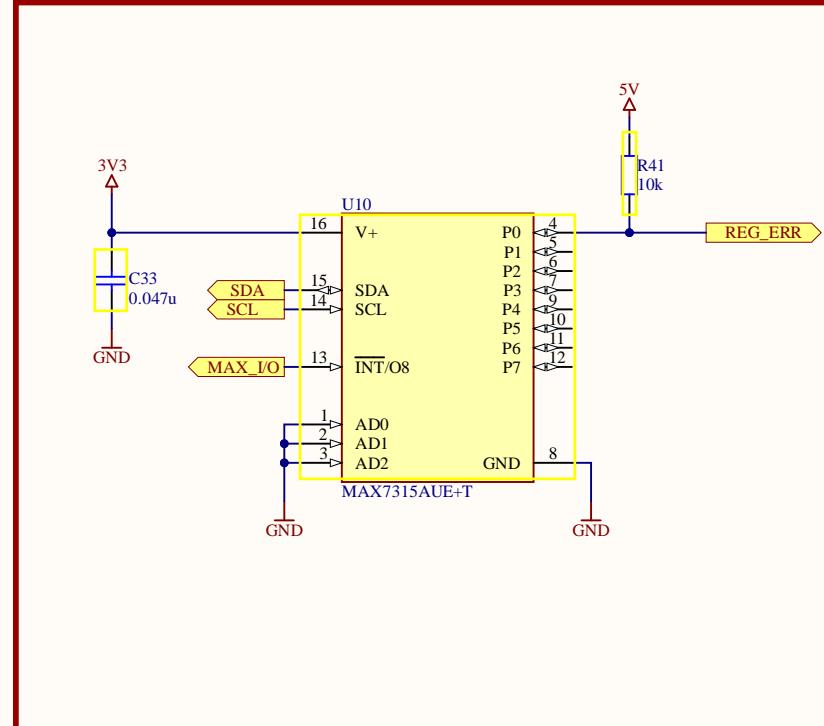
### Régulation 5V



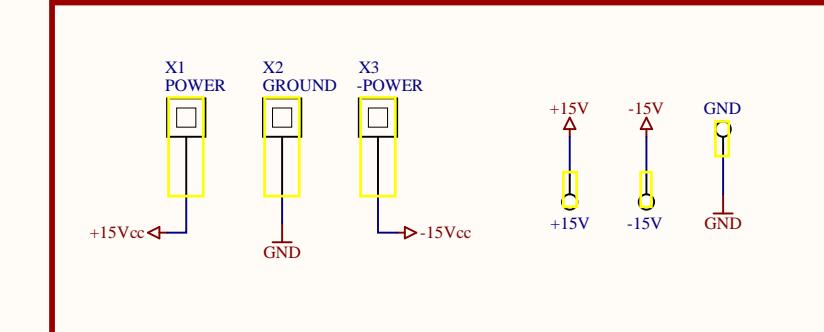
### Régulation 3V3



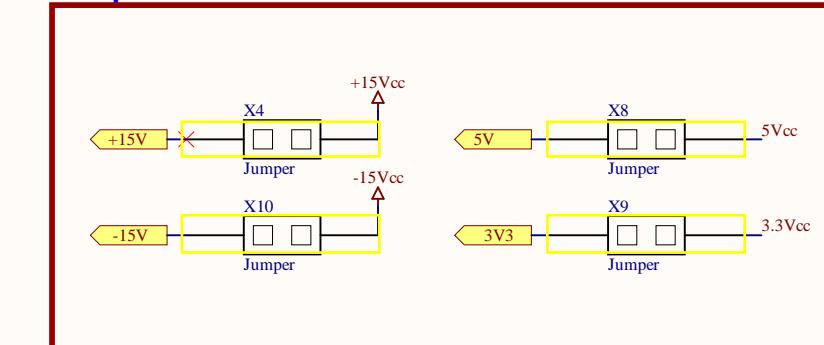
### Convertisseur DAC



### Alimentation +15V/-15V/GND



### Jumper Alimentation



Fichier : Alimentation.SchDoc

Date : 16.06.2023

Version :

Heure : 03:26:12

Auteur :



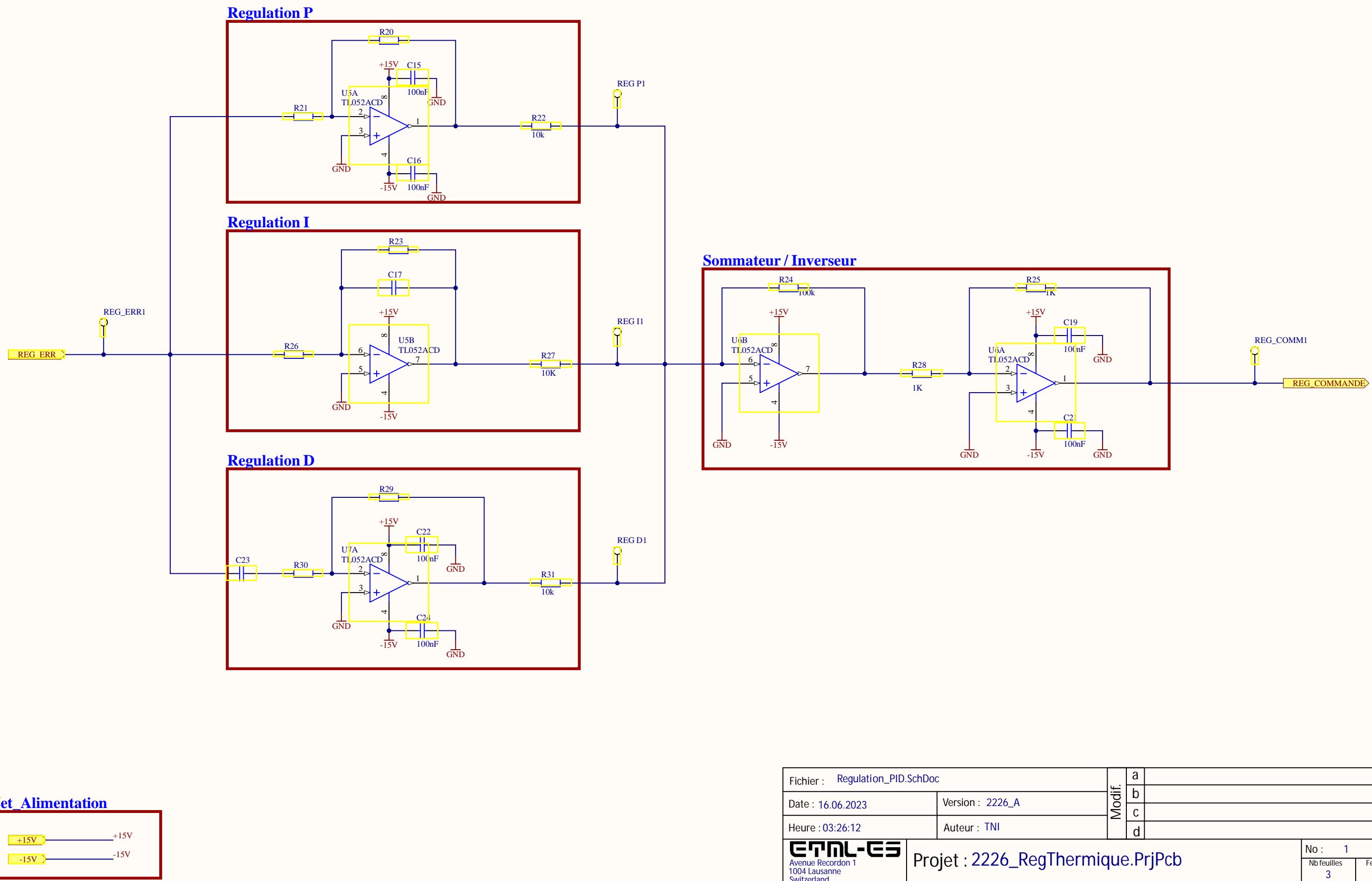
Projet : 2226\_RegThermique.PrjPcb

No :

Nb feuilles

Feuille n°

Chemin : K:\ES\PROJETS\SLO\2226\_RegThermique\hard\2226\_RegThermique\_1\Alimentation.SchDoc



# Projet ETML-ES – Modification

*Note: Les textes explicatifs en italique peuvent être supprimés*

*A remplir par l'initiateur*

<b>PROJET:</b>	2226_RegThermique		
<b>Entreprise/Client:</b>	ETML-ES	<b>Département:</b>	-
<b>Demandé par (Prénom, Nom):</b>	Neziri Taulant	<b>Date:</b>	15.06.2023
<b>Objet (No ou réf, pièce, PCB...)</b>			
<b>Version à modifier:</b>	A		

*A remplir par l'exécutant*

<b>Auteur (ETML-ES):</b>		<b>Filière:</b>	SLO
<b>Nouvelle version:</b>		<b>Date:</b>	

## 1 Description ou justification

1. Modifier la schématique
2. Contrôler et modifier les mauvais footprints
3. Implémenter la régulation PID

## 2 Référence conception

2226\_RegThermique / K:\ES\PROJETS\SLO\2226\_RegThermique\hard\A

## 3 Détail des modifications

*Chaque rangée du tableau ci-dessous contient le détail d'une seule modification.*

#	Description	Fait	Approuvé
1	Schématique : ajouter une liaison PEC12_AB au microcontrôleur	NOK	
2	Schématique : ajouter une liaison Val.capteur de temp. Au microcontrôleur	NOK	
3	PCB : Modifier footprint potentiomètre R12 (pin 1 et 2)	NOK	
4	PCB : Modifier tous footprint SOT-23(pin 1 et 2) ex : Q3,Q5	NOK	
5	PCB : Faire une rotation de 180° du LCD afin d'être dans le sens de lecture	NOK	
6	PCB : Changer les footprints des connecteurs pour capteur temp. (ref point 2)	NOK	
7			
8			

## 4 Remarques

*Au besoin, indiquer ici des détails nécessaires à la compréhension, ainsi que les raisons d'une modification non effectuée ou reportée.*

*Exemple: Le point 2 (marqué NOK), est reporté pour une prochaine version pour épuiser notre stock de composants. Cette modif n'est pas critique fonctionnellement.*

## 5 Convention de nommage et liens

Le nom de ce fichier doit être unique et doit donc contenir le numéro du projet et un numéro consécutif de modification avec le format suivant :

***aaii\_MOD\_nn.docx***

ou

***NomProjet\_MOD\_nn.docx***

avec :

- MOD : pour modification
- aaii : numéro de projet, exemple **1708** pour projet de 2017 no 08
- NomProjet : Si le projet n'est pas numéroté ou mandat de client.
- nn : numéro de modification. La première est 01

Exemples :

- **1708\_MOD\_01.docx** 1ere modification pour le projet 1708
- **1708\_MOD\_02.docx** 2e modification pour le projet 1708
- **CapteurVolets\_MOD\_01.docx** Cas de projet externe

Le schéma et/ou les documents de production de la pièce ou du PCB se référeront à ce document dans les cartouches.

Si un nouveau projet reprend un design d'un autre projet, créer un document de **modification numéro 00**. Ainsi, on pourra décrire les modifications initiales dans le fichier.

Exemple :

- **1803\_MOD\_00.docx** Modification initiale pour le nouveau projet 1803 à partir d'un autre projet (par ex. 1708)

### 5.1 Stockage du fichier

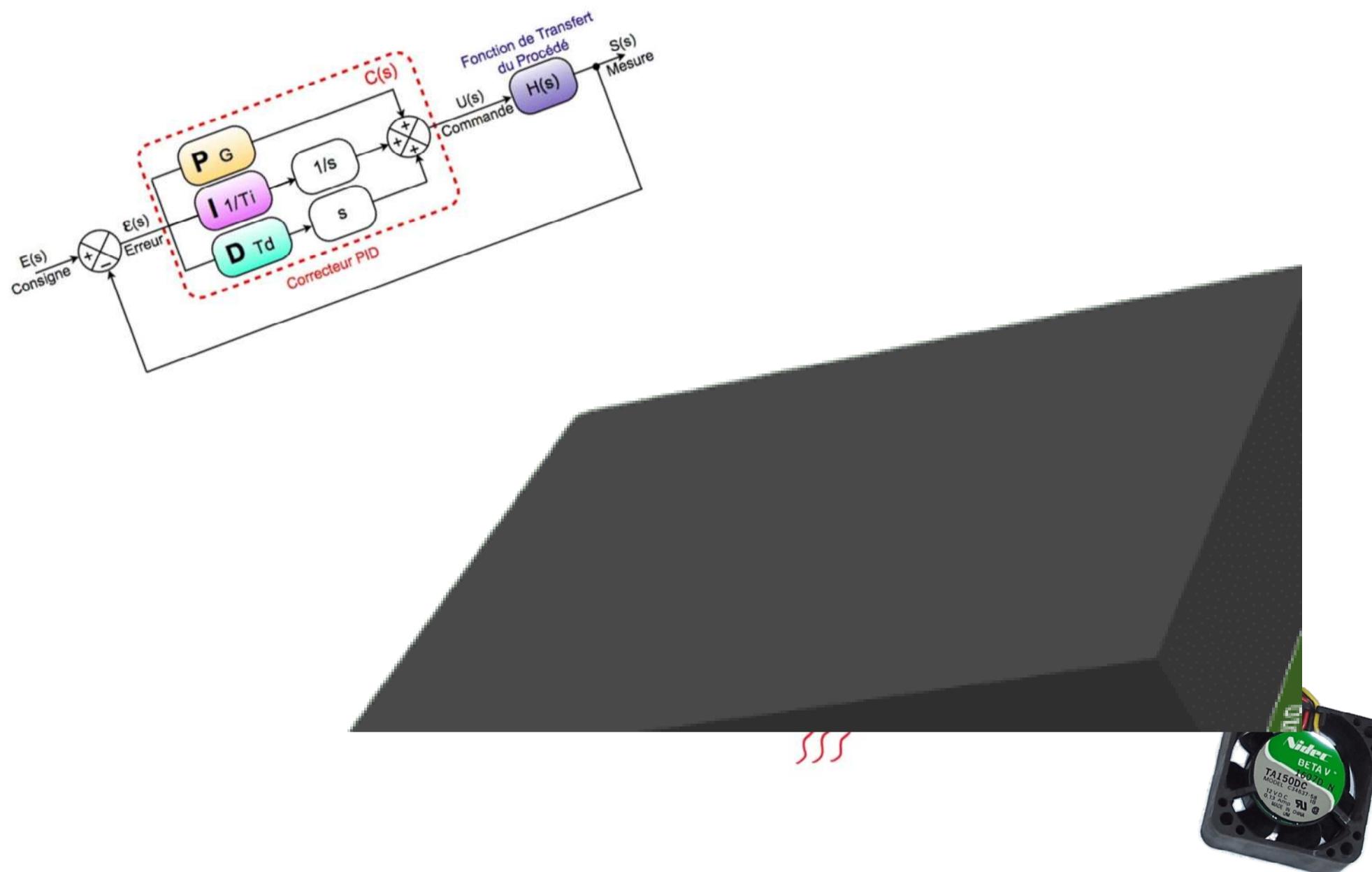
Ce fichier sera stocké à la racine du dossier **/doc** d'un projet.

Ainsi, tous les fichiers de modifications des pièces ou PCBs faisant partie du projet sont centralisés dans le même répertoire. La numérotation devient implicite.



Le but de ce projet est de concevoir un système de régulation thermique dans un boîtier fermé pour les cours de réglage.

L'utilisateur aura deux possibilités de réglages. La première pourra être effectuer via un générateur externe et des touches physiques (bouton, potentiomètre). La deuxième, se fera via USB.



Ce projet m'a permis de développer un produit partant d'une idée sous forme de cahier charges jusqu'à avoir un produit qui pourra être livrable. Le projet n'est pas complètement fini. Les diverses périphériques du système sont fonctionnels, mais la régulation n'a pas pu être traité complètement, et n'est donc pas fonctionnelle.

Malgré les difficultés rencontrés, ce projet m'a apporté de très bonnes connaissances en ce qui concerne la gestion de projet, et a approfondi aussi mes connaissances en dimensionnement et design de schéma électrique ainsi que la conception de PCB.

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.c

\*\*\*\*\*  
MPLAB Harmony Application Source File

Company:

Microchip Technology Inc.

File Name:

app.c

Summary:

This file contains the source code for the MPLAB Harmony application.

Description:

This file contains the source code for the MPLAB Harmony application. It implements the logic of the application's state machine and it may call API routines of other MPLAB Harmony modules in the system, such as driver system services, and middleware. However, it does not call any of the system interfaces (such as the "Initialize" and "Tasks" functions) of a module in the system or make any assumptions about when those functions are called. That is the responsibility of the configuration-specific source files.

\*\*\*\*\*  
// DOM-IGNORE-BEGIN

\*\*\*\*\*  
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute Software only when embedded on a Microchip microcontroller or digital signal controller that is integrated into your product or third party product (pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES

\*\*\*\*\*

It

ers,

ny of  
tions  
ystem

\*\*\*\*\* /

\*\*\*\*\*

rved.

l

D,

SE.

R

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.c

```
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.  
*****  
// DOM-IGNORE-END  
  
*****  
*****  
// Section: Included Files  
*****  
*****  
  
#include "app.h"  
#include "Mc32DriverLcd.h"  
  
*****  
*****  
// Section: Global Data Definitions  
*****  
*****  
  
/* Application Data  
  
Summary:  
    Holds application data  
  
Description:  
    This structure holds the application's data.  
  
Remarks:  
    This structure should be initialized by the APP_Initialize function.  
  
    Application strings and buffers are be defined outside this structure.  
*/  
  
APP_DATA appData;  
S_ADCConvert ValTemp;  
*****  
*****  
// Section: Application Callback Functions  
*****  
*****  
  
/* TODO: Add any necessary callback functions.
```

\*\*\*\*\* /

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.c

```
/*
// ****
// Section: Application Local Functions
// ****
// ****

/* TODO: Add any necessary local functions.
*/

/*
// ****
// Section: Application Initialization and State Machine Functions
// ****
// ****

/*****
Function:
    void APP_Initialize ( void )

Remarks:
    See prototype in app.h.
 */

void APP_Initialize ( void )
{
    /* Place the App state machine in its initial state. */
    appData.state = APP_STATE_INIT;

    /* TODO: Initialize your application's state machine and other
     * parameters.
    */
}

/*****
Function:
    void APP_Tasks ( void )

Remarks:

```

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.c

```
See prototype in app.h.  
*/  
  
void APP_Tasks ( void )  
{  
    /* Check the application's current state. */  
    switch ( appData.state )  
    {  
        /* Application's initial state. */  
        case APP_STATE_INIT:  
        {  
            // Initialisation de l'LCD  
            lcd_init();  
            lcd_gotoxy(1,1);  
            printf_lcd("PROJ REGTHERM");  
            lcd_gotoxy(1,2);  
            printf_lcd("Neziri Taulant");  
            lcd_b1_on();  
  
            // Démarrage des Timers  
            DRV_TMR0_Start();  
            DRV_TMR1_Start();  
            DRV_TMR2_Start();  
  
            // Démarrage des OC  
            DRV_OC0_Start();  
            DRV_OC1_Start();  
  
            // Life LED On  
            LED_VOn();  
  
            //ADC Init  
            BSP_InitADC10();  
  
            appData.state = APP_STATE_WAIT;  
  
            break;  
        }  
  
        case APP_STATE_SERVICE_TASKS:  
        {  
            ValTemp = ValConvert();  
            float Temp = ValTemp.Val_T;
```

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.c

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.c

```
//float Temp = ValTemp.Val_T;

lcd_gotoxy(1, 4);
printf_lcd("ValRaw : %2.2f [C]", Temp);

appData.state = APP_STATE_WAIT;
break;
}

/* TODO: implement your application state machine.*/
case APP_STATE_WAIT:
{
    // Ne rien faire ici
    break;
}

/* The default state should never be executed. */
default:
{
    /* TODO: Handle error in application's state machine. */
    break;
}
}

// Fonction de mise à jour du switch
void APP_UpdateState (APP_STATES NewState)
{
    appData.state = NewState;
}

*****
End of File
*/
```

\*\*\*\*\*

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.h

\*\*\*\*\*  
MPLAB Harmony Application Header File

Company:

Microchip Technology Inc.

File Name:

app.h

Summary:

This header file provides prototypes and definitions for the application.

Description:

This header file provides function prototypes and data type definitions for the application. Some of these are required by the system (such as the "APP\_Initialize" and "APP\_Tasks" prototypes) and some of them are only used internally by the application (such as the "APP\_STATES" definition). Both are defined here for convenience.

\*\*\*\*\*  
//DOM-IGNORE-BEGIN

\*\*\*\*\*  
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute Software only when embedded on a Microchip microcontroller or digital signal controller that is integrated into your product or third party product (pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

\*\*\*\*\*  
//DOM-IGNORE-END

\*\*\*\*\*

n.

for

used

oth

\*\*\*\*\* /

\*\*\*\*\*

rved.

l

D,

SE.

R

\*\*\*\*\* /

```
#ifndef _APP_H
#define _APP_H

// ****
// **** Section: Included Files
// ****
// ****

#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include "system_config.h"
#include "system_definitions.h"
#include "MC32ADCDriver.h"

// DOM-IGNORE-BEGIN
#ifdef __cplusplus // Provide C++ Compatibility

extern "C" {

#endif
// DOM-IGNORE-END

// ****
// **** Section: Type Definitions
// ****
// ****

// ****
/* Application states

Summary:
Application states enumeration

Description:
This enumeration defines the valid application states. These states
determine the behavior of the application at various times.

*/
typedef enum
```

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.h

```
{  
/* Application's state machine's initial state. */  
APP_STATE_INIT=0,  
APP_STATE_SERVICE_TASKS,  
    APP_STATE_WAIT  
  
/* TODO: Define states used by the application state machine. */  
  
} APP_STATES;  
  
// *****  
/* Application Data  
  
Summary:  
    Holds application data  
  
Description:  
    This structure holds the application's data.  
  
Remarks:  
    Application strings and buffers are be defined outside this structure.  
*/  
  
typedef struct  
{  
    /* The application's current state */  
    APP_STATES state;  
  
    /* TODO: Define any additional data used by the application. */  
  
} APP_DATA;  
  
// *****  
// *****  
// Section: Application Callback Routines  
// *****  
// *****  
/* These routines are called by drivers when certain events occur.  
*/  
  
// *****  
// *****
```

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/app.h

```
// Section: Application Initialization and State Machine Functions
// ****
// ****
/*
Function:
    void APP_Initialize ( void )

Summary:
    MPLAB Harmony application initialization routine.

Description:
    This function initializes the Harmony application. It places the
    application in its initial state and prepares it to run so that its
    APP_Tasks function can be called.

Precondition:
    All other system initialization routines should be called before calling
    this routine (in "SYS_Initialize") .

Parameters:
    None.

Returns:
    None.

Example:
<code>
APP_Initialize();
</code>

Remarks:
    This routine must be called from the SYS_Initialize function.
*/
void APP_Initialize ( void );

/*
Function:
    void APP_Tasks ( void )

Summary:
    MPLAB Harmony Demo application tasks function
```

\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*

g

\*\*\*\*\*

**Description:**

This routine is the Harmony Demo application's tasks function. It defines the application's state machine and core logic.

**Precondition:**

The system and application initialization ("SYS\_Initialize") should be called before calling this.

**Parameters:**

None.

**Returns:**

None.

**Example:**

```
<code>
APP_Tasks();
</code>
```

**Remarks:**

This routine must be called from SYS\_Tasks() routine.

\*/

```
void APP_Tasks( void );
void APP_UpdateState ( APP_STATES NewState ) ;

#endif /* _APP_H */

//DOM-IGNORE-BEGIN
#ifndef __cplusplus
}
#endif
//DOM-IGNORE-END

/***********************/

End of File
*/
```

\*\*\*\*\*

C:/microchip/harmony/v2\_06/apps/MINF/PROJ\_RegTherm/firmware/src/system\_config/default/system\_interrupt.c

\*\*\*\*\*  
System Interrupts File

File Name:  
system\_interrupt.c

Summary:  
Raw ISR definitions.

Description:  
This file contains a definitions of the raw ISRs required to support the interrupt sub-system.

Summary:  
This file contains source code for the interrupt vector functions in the system.

Description:  
This file contains source code for the interrupt vector functions in the system. It implements the system and part specific vector "stub" function from which the individual "Tasks" functions are called for any modules executing interrupt-driven in the MPLAB Harmony system.

Remarks:  
This file requires access to the systemObjects global data structure that contains the object handles to all MPLAB Harmony module objects executing interrupt-driven in the system. These handles are passed into the individual "Tasks" functions to identify the instance of the module to main

\*\*\*\*\*

// DOM-IGNORE-BEGIN

\*\*\*\*\*  
Copyright (c) 2011-2014 released Microchip Technology Inc. All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute Software only when embedded on a Microchip microcontroller or digital signal controller that is integrated into your product or third party product (pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF

\*\*\*\*\*

e

e

e  
ions

at  
ng  
vidual  
tain.  
\*\*\*\*\* /

\*\*\*\*\*

rved.

l

D,

```

C:/microchip/harmony/v2_06/apps/MINF/PROJ_RegTherm/firmware/src/system_config/default/system_interrupt.h
MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
*****
// DOM-IGNORE-END

// ****
// Section: Included Files
// ****
// ****

#include "system/common/sys_common.h"
#include "app.h"
#include "system_definitions.h"

// ****
// ****
// Section: System Interrupt Vector Functions
// ****
// ****

void __ISR(_TIMER_1_VECTOR, ipl2AUTO) IntHandlerDrvTmrInstance0(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_1);

    static int count= 0;

    if(count < 3000)
    {
        count++;
    }
    else
    {
        APP_UpdateState(APP_STATE_SERVICE_TASKS);
    }
}

```

SE.

R

\*\*\*\*\* /

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

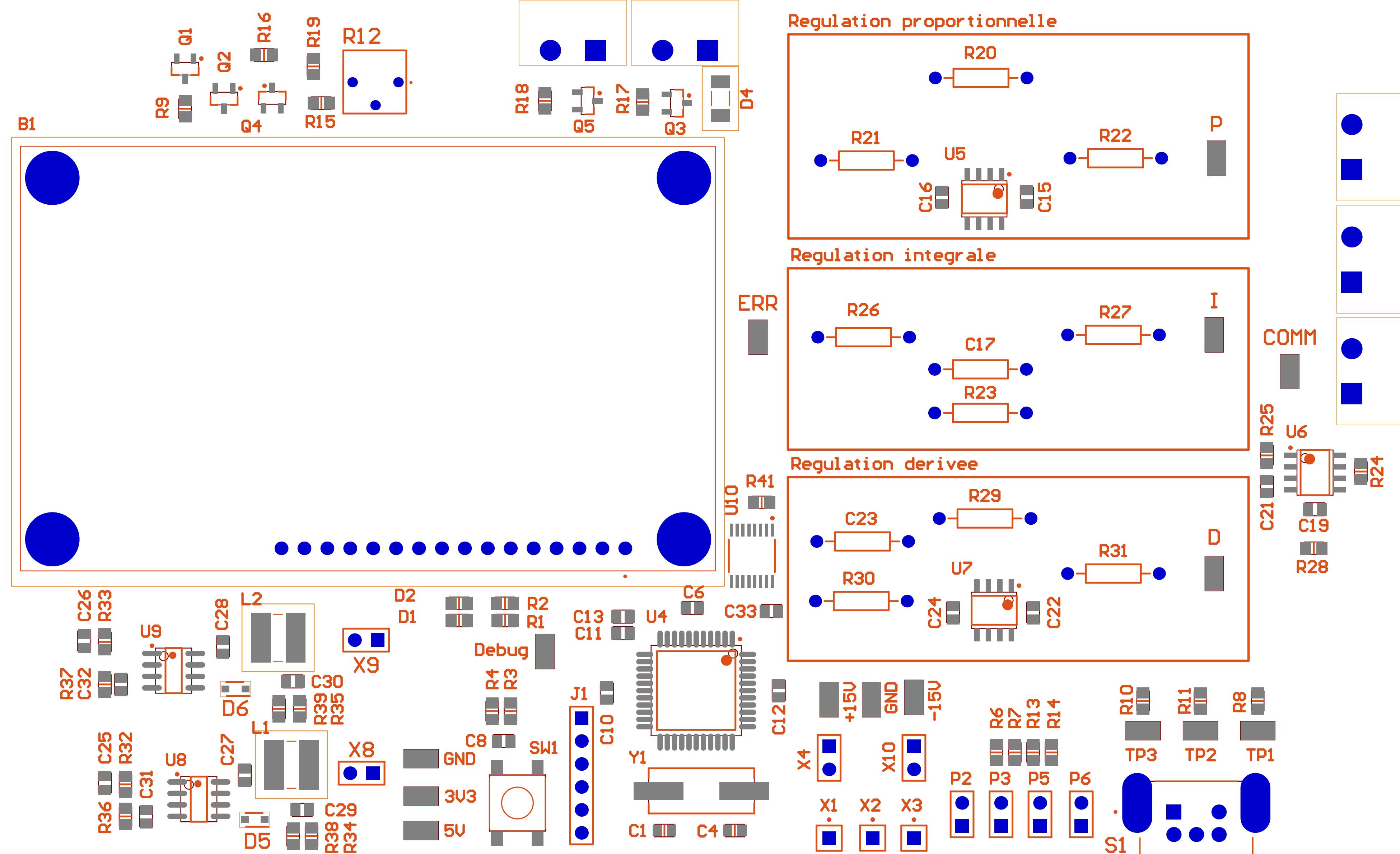
```
C:/microchip/harmony/v2_06/apps/MINF/PROJ_RegTherm/firmware/src/system_config/default/system_interrup
}

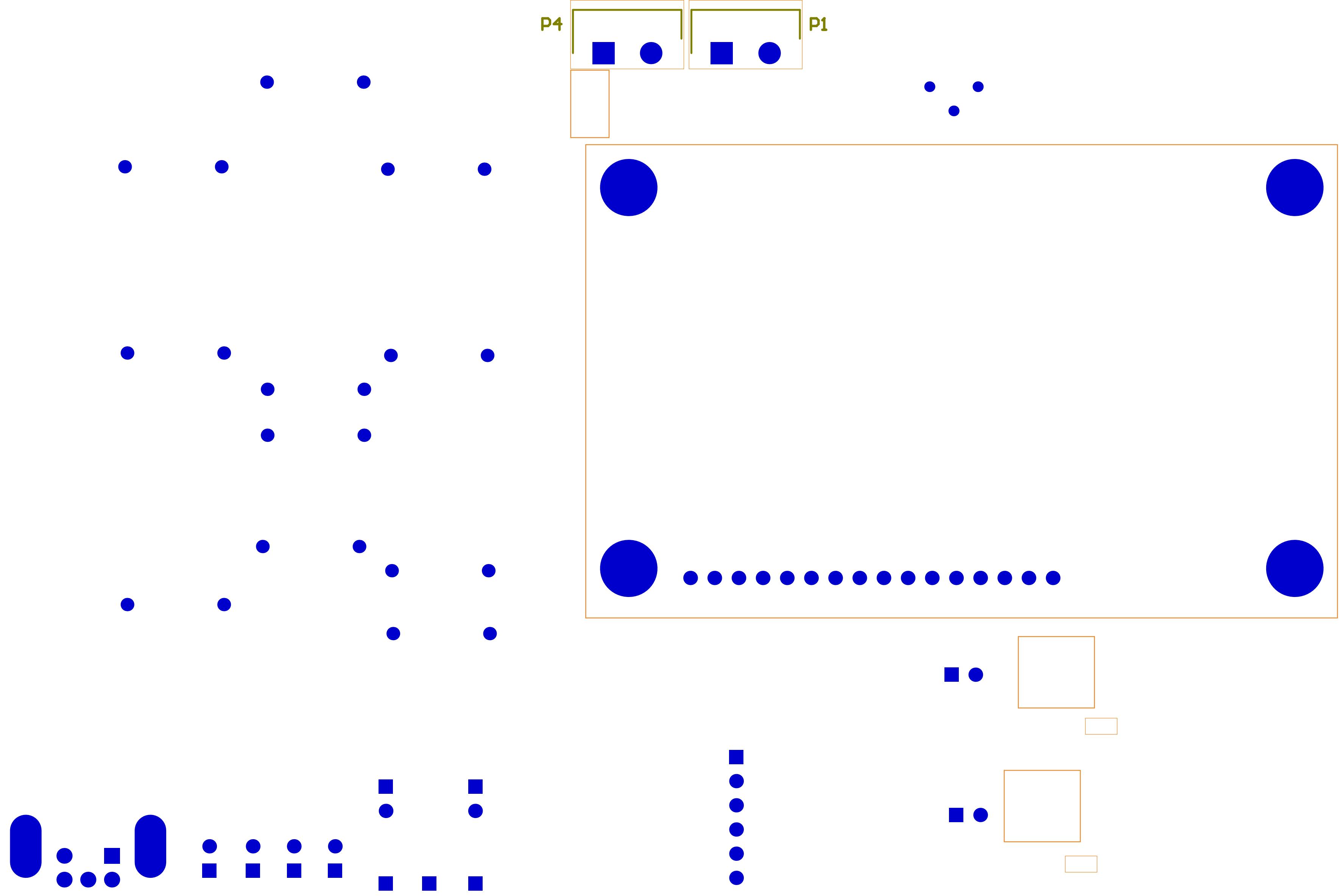
void __ISR(_TIMER_2_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance1(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_2);
}

void __ISR(_TIMER_3_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance2(void)
{
    PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_3);
}

//*********************************************************************
End of File
*/
```

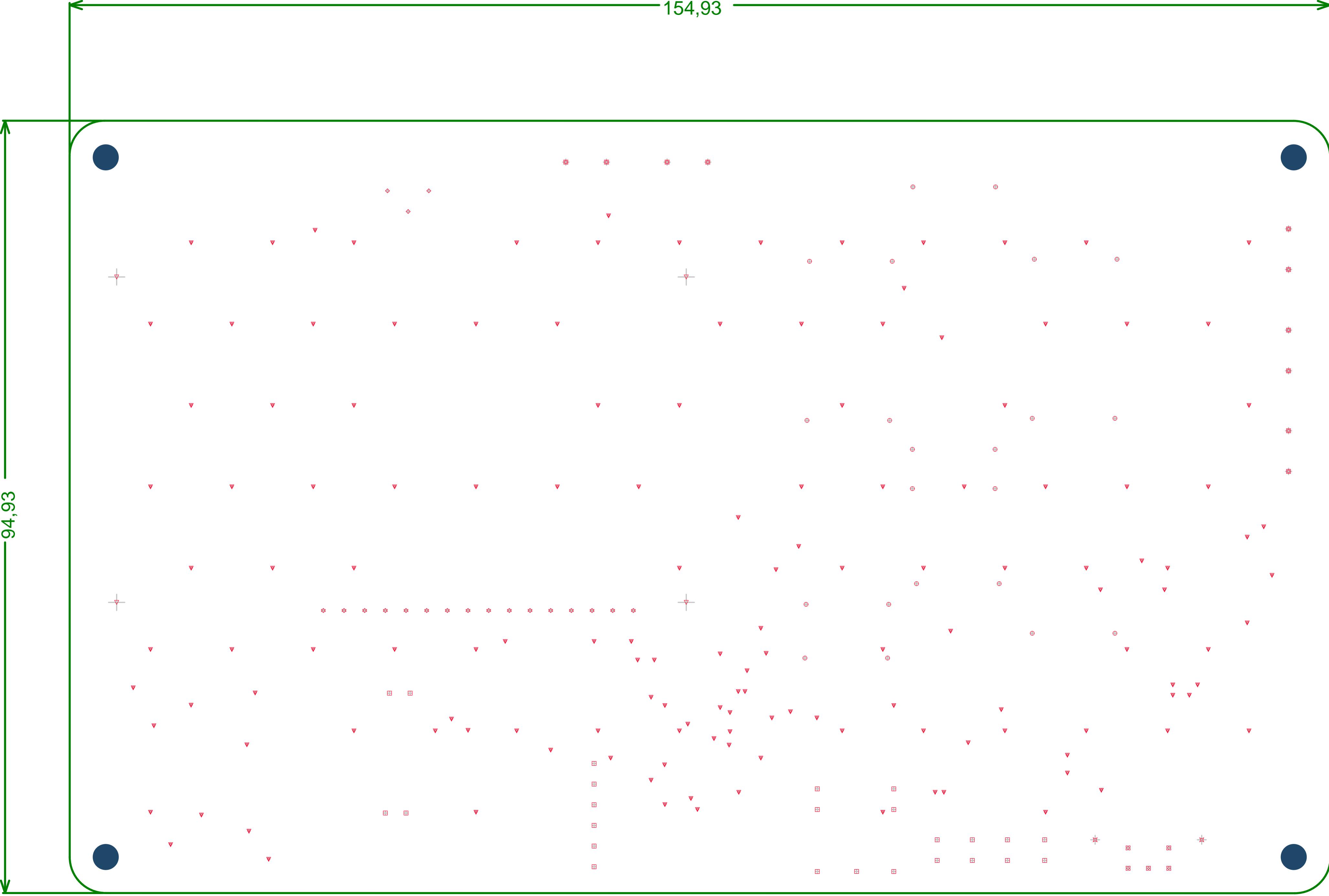
\* \* \* \* \*











Comment	Description	Designator	Footprint	LibRef	Quantity
REGP, LedDebugg, PEC12	Test Point, 1 Position SMD, RoHS, Tape and Reel	-15V, +15V, 3V3, 5V, Debug1, GND, GND1, REG D1, REG I1, REG P1, REG_COMM1, REG_ERR1, TP1, TP2, TP3	KSTN-5019_V	CMP-1672-00008-1	15
NHD-0420AZ-FSW-GBW-33V3	LCD Character Display Modules & Accessories STN- GRAY Transfl 77.5 x 47.0	B1	NHD0420AZFSWG-BW33V3	NHD-0420AZ-FSW-GBW-33V3	1
22pF, 10V	CAP 22pF 10V	C1, C4	6-0805_L	CMP-1036-01211-1	2
Cap	Cap 100nF	C2, C5, C6, C8, C9, C10, C11, C12, C13, C15, C16, C19, C21, C22, C24, C27, C28, C31, C32	C0805	Cap	19
Cap	Cap 1nF	C3, C7, C14	C0805	Cap	3
Cap	Capacitor	C17, C23	AXIAL-0.4	Cap	2
Cap Pol1	Cap Pol 47uF	C25, C26	C0805	Cap Pol1	2
Cap Pol1	Cap Pol 33uF	C29, C30	C0805	Cap Pol1	2
Cap	Cap 0.047uF	C33	C0805	Cap	1
Super Red, 1.9 V	SMT Mono-color Chip LED Waterclear WL-SMCW, size 0805, Super Red, 1.9 V, 140 deg, 60 mcd	D1	6-0805_N	150080SS75000	1
Bright Green, 2 V	SMT Mono-color Chip LED Waterclear WL-SMCW, size 0805, Bright Green, 2 V, 140 deg, 40 mcd	D2	6-0805_N	150080VS75000	1
EGL1B	Diode Melf : EGL1B (100V,1A)	D4	JAN1N4108CUR1_	Diode	1
50SQ100	Schottky Diode 50SQ100	D5, D6	SODFL2513X70N	D Schottky	2
61300611121	WR-PHD Pin Header, THT, pitch 2.54mm, Single Row, Vertical, 6pin	J1	HDR1X6	61300611121	1
Inductor	Inductor 220uH	L1, L2	SDE0604A101K	Inductor	2
R_Dissipator, Alim 0-30 IN, FAN_Control	Header, 2-Pin	P1, P2, P3, P4, P5, P6, P8, P9, P10	RHDRA2W95P0X_500_1X2_1120X90_0X1280P, HDR1X2	Header 2	9
PNP	PNP Bipolar Transistor	O1, O2	SOT-23B_N	PNP	2
MOSFET-N	N-Channel MOSFET	O3, O4, O5	SOT23_N	MOSFET-N	3
270R	Res 270R	R1, R2	6-0805_N	CMP-2001-00528-1	2
10k, Res 10K, Res2	Res 10K	R3, R6, R7, R8, R10, R11, R13, R14, R17, R18, R41	6-0805_N	CMP-2001-04376-1, Res2, Resistor	11
1k, Res2, [NoValue]	Res 1K	R4, R25, R28, R32, R33, R36, R37	6-0805_N	CMP-1013-00510-2, Res2	7
Res2	Res 33R	R9	6-0805_N	Res2	1
RPot	Pot 10K	R12	3362P_1	RPot	1
Res2	Res 200R	R15	6-0805_N	Res2	1
Res2	Res 1K8	R16	6-0805_N	Res2	1
Res2, 10k	Res 100K	R19, R22, R24, R27, R31	6-0805_N, AXIAL-0.4	Res2	5
Res2	Resistor	R20, R21, R23, R26, R29, R30, R34, R38	AXIAL-0.4, 6-0805_N	Res2	8
Res2	Res 24K	R35	6-0805_N	Res2	1
Res2	Res 15K	R39	6-0805_N	Res2	1
PEC12R-2220F-S0024	Rotary Encoder Mechanical 24 Quadrature (Incremental) Right Angle	S1	XDCR_PEC12R-2220F-S0024	PEC12R-2220F-S0024	1
430182043816	WS-TASV SMD Tact Switch 6X6 mm	SW1	430182043816	CMP-1452-00013-1	1
MAX6612MXK+	High-Slope, Low-Power, Analog Temperature Sensor, -55 to 150 degC, 5-Pin SOT23 (X5+1), Pb-Free	U1, U2, U3	MAXM-X5+1_M	CMP-0253-01154-1	3
PIC32MX250F128D/I/PT	32-bit Microcontroller with Audio and Graphics Interface, USB and Advanced Analog, 50 MHz, 33 I/O, PMP, -40 to 85 degC, 44-pin TQFP (PT44), Tray	U4	MCHP-TQFP-PT44_M	CMP-0048-00499-1	1
TL052ACD	Enhanced JFET Precision Dual Operational Amplifier, 10 to 30 V, 0 to 70 degC, 8-pin SOIC (D8), Green (RoHS & no Sb/Br)	U5, U6, U7	D0008A_N	CMP-0015-00087-2	3
MAX5033AUSA	500mA, 76V, High-Efficiency, MAXPower(TM) Step-Down DC-DC Converter, 3.3V output, 8-Pin SOIC, 0°C to +85°C	U8, U9	S8-5_N	CMP-0246-00246-1	2
MAX7315AUE+T	IC I/O EXPANDER I2C 8B 16TSSOP	U10	FP-U16-2-MFG	CMP-05690-00009-1	1
POWER, GROUND, -POWER	CONN HEADER VERT 1POS	X1, X2, X3	PIN1	Berg	3
Jumper	CONN HEADER VERT 2POS 2.54MM	X4, X8, X9, X10	HDR1X2	Jumper	4
8MHz, 12p	Resistance Weld SMD Crystal, 8 MHz, +/- 30 ppm, 12 pF, -20 to 70 degC, 2-Pin SMD, RoHS, Tape and Reel	Y1	FOX-FOXSDLF160-20_V	CMP-1749-00001-1	1