Contrôle de température via une cellule Peltier

La Revue 3EI
Culture Sciences de l'Ingénieur

Édité le 25/05/2023

école normale supérieure paris—saclay

Rémi AL AJROUDI[1,3] - Sylvie JAUVERT[2] - Morgan ALMANZA[3]

[1] Université Paris-Saclay, IUT GEII Cachan, [2] Education Nationale, Lycée Saint-Louis, [3] Université Paris-Saclay, ENS Paris-Saclay

Cette ressource fait partie du N°111 de La Revue 3EI de janvier 2024.

Dans de nombreux processus, la température joue un rôle important. Pour les systèmes de petites tailles, les cellules Peltier permettent de contrôler la température d'un système avec précision et rapidité. A partir de kits de développement facilement accessibles, nous proposons de mettre en œuvre un système complet de régulation de température en allant de la chaîne d'énergie à la chaîne d'information. Ce système offre un excellent support de cours et de TP pour plusieurs disciplines comme l'électronique de puissance (pont en H), l'informatique industrielle (codage à virgule fixe, bus série, threads, SPI), l'automatique (linéarisation, correcteur PI), implémentation sur microcontrôleur et des aspects plus logiciels avec une interface graphique et la gestion des threads sous python ou encore la communication par socket sur réseau TCP/IP. Ce travail se positionne avant tout dans le cadre de notions de niveau Licence.

1 - Introduction

Le contrôle de température est nécessaire dans de nombreuses applications allant de la biologie à l'ingénierie. Aujourd'hui deux types de stratégies sont utilisées pour contrôler la température, soit un thermocryostat qui utilise un fluide réfrigérant, soit un système à base d'une cellule Peltier. Ces derniers permettent un contrôle précis et rapide de la température mais les puissances froides disponibles sont faibles autour de quelques dizaines de Watts. Des sociétés comme *Stanford Research* proposent des systèmes de régulation de température utilisant une cellule Peltier, par exemple le modèle PTC10 offre un contrôle au milliKelvin avec la capacité de piloter une cellule Peltier jusqu'à 1A – 50V DC.

Dans cet article, un système [1] de contrôle de température via une cellule Peltier est développé, sa résolution en température est de 10 mK et il est capable de piloter des cellules Peltier de 7A-50V. L'accès à des puissances importantes permet d'obtenir des variations rapides de température.

La Figure 1 montre une photo du système avec trois parties, la partie thermique avec le Peltier (à gauche), le système d'asservissement de la température sur microcontrôleur (en bas à gauche) et le système de supervision avec une interface utilisateur (à droite).

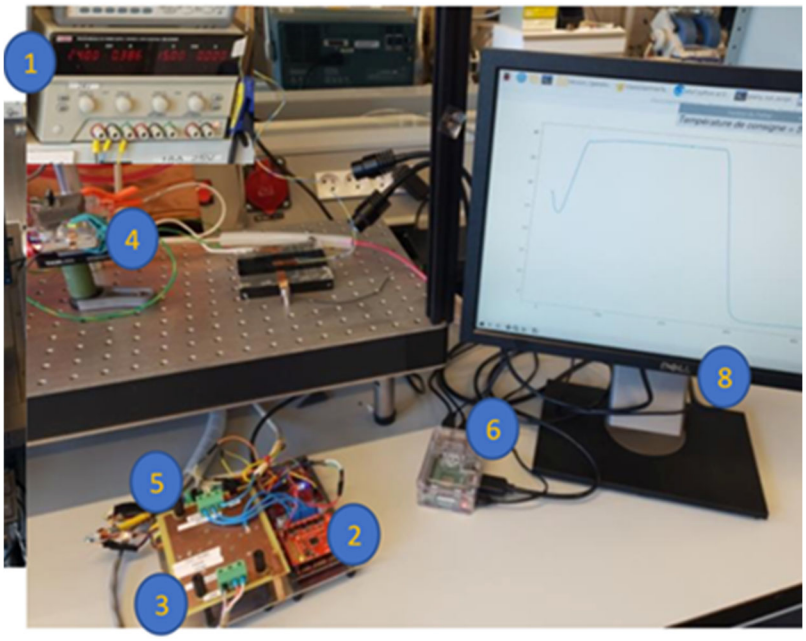

Figure 1 : Photo du système mis en œuvre avec en 1-l'alimentation, en 2-le microcontrôleur et le hacheur, en 3-le filtre LC associé au hacheur, en 4-la cellule Peltier, en 5 les conditionneurs des sondes de température PT100, en 6 et 8 le système de supervision

La Figure 2 donne le synoptique du système avec en 2- le module Hacheur (kit DRV8301 de TI, Texas Instrument) qui se clipse sur la carte de développement du microcontrôleur (F28379D de TI). Puisque le module hacheur dispose d'un abaisseur de tension, il est en mesure d'alimenter le microcontrôleur. Les deux points milieu des bras du module hacheur sont ensuite connectés à la cellule Peltier au travers d'un filtre LC. Ce filtre élimine l'ondulation due au découpage pour ne récupérer que la composante continue tout en évitant de possibles interférences électromagnétiques avec d'autres équipements. Pour mesurer la température, une sonde PT100 est connectée à un conditionneur de signaux (Max31865), celui transmet la température au microcontrôleur via un bus SPI. Après une phase d'initialisation des capteurs, le microcontrôleur exécute à pas de temps fixé (20ms) la lecture du capteur de température, scrute la consigne reçue sur le port série, transmet la température mesurée sur le port série et exécute la boucle de régulation qui agit sur le rapport cyclique du hacheur.

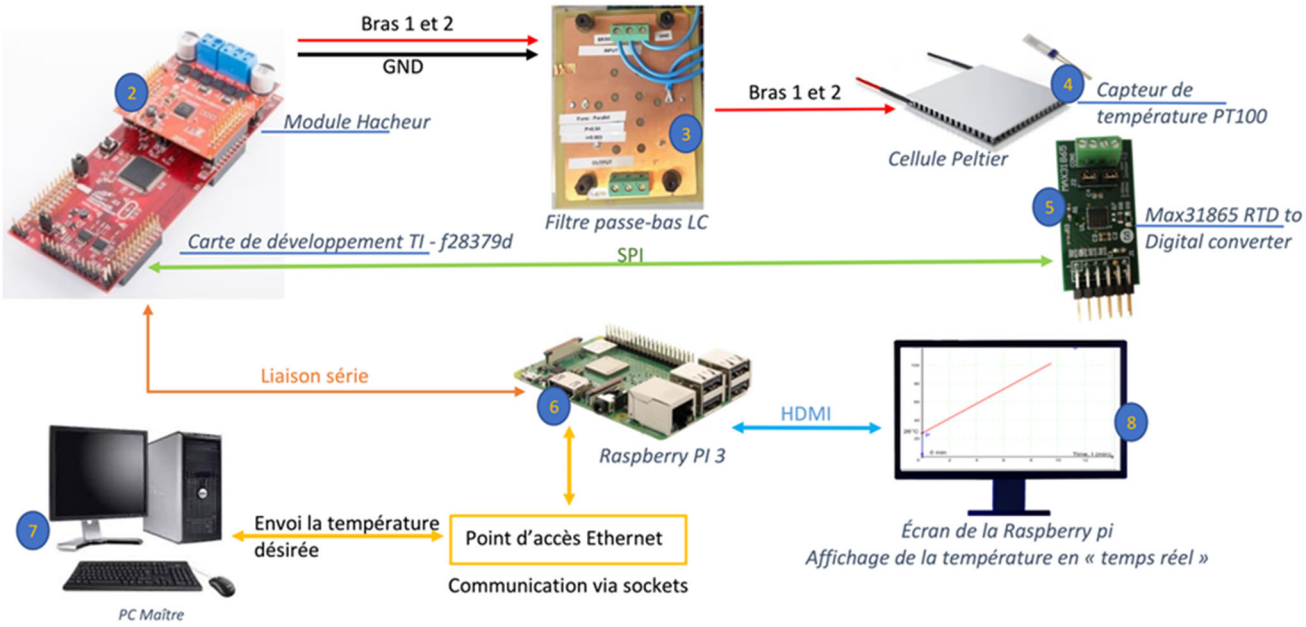

Figure 2 : Schéma synoptique où les numéros indiquent la correspondance avec la Figure 1, le 7-correspond au PC maitre

Ce système avec un pas de temps fixé est connecté à un système de supervision (Raspberry Pi), lequel dispose d'un affichage graphique et d'une interface utilisateur. Bien que son pas de temps soit souple, le recours à un système d'exploitation (Linux et des librairies Python) simplifie la réalisation d'interface graphique et la gestion de la communication client/serveur. Le système de supervision dialogue avec le microcontrôleur de la boucle de régulation via une communication série alors que la communication avec un PC maitre se fait via une communication par socket sur réseau Ethernet TCP/IP.

Ces illustrations ne montrent pas les éléments avec lesquels la cellule Peltier interagit. Sur une face, nous avons une température fixe qui est maintenue par flux d'eau généré par un thermocryostat, sur l'autre face il y a le bloc d'aluminium que nous devons réguler en température.

Dans la partie 2, nous décrivons la constitution et la modélisation du système, puis la modélisation de notre système, soit les modèles thermique et électrique de la cellule Peltier et du hacheur. Dans la partie III, nous décrirons l'asservissement de la cellule Peltier[3]. Enfin dans la partie IV, nous décrivons l'interface utilisateur permettant d'envoyer la température de consigne et d'observer en temps réel l'évolution de la température mesurée sur un écran.

2 - Éléments du système et modélisation

Cette partie établit un modèle thermique et électrique du système. Ce modèle nous permettra ensuite de définir la stratégie de contrôle à utiliser.

2.1 - Cellule Peltier et modèle thermique

La Figure 3 montre comment la cellule Peltier est intriquée dans le système thermique. Cet assemblage nécessite quelques précautions qui sont disponibles chez le fabricant de cellule Peltier[1]. Puisque le coefficient de performance d'une cellule Peltier est autour de 1, alors qu'il est de 6 pour des systèmes à compression détente de gaz, il est indispensable d'évacuer efficacement la chaleur de la source de chaude. Il faut donc refroidir cette face afin que la cellule Peltier ne se détériore pas. Ici nous avons recours à un échangeur à eau sur un thermo cryostat (ou un mini chiller), cette solution va nous permettre d'imposer la température de la source chaude, néanmoins un simple échangeur à eau ou à air aurait été suffisant. Le système étant destiné à refroidir un échantillon, les puissances moyennes nécessaires sont faibles.

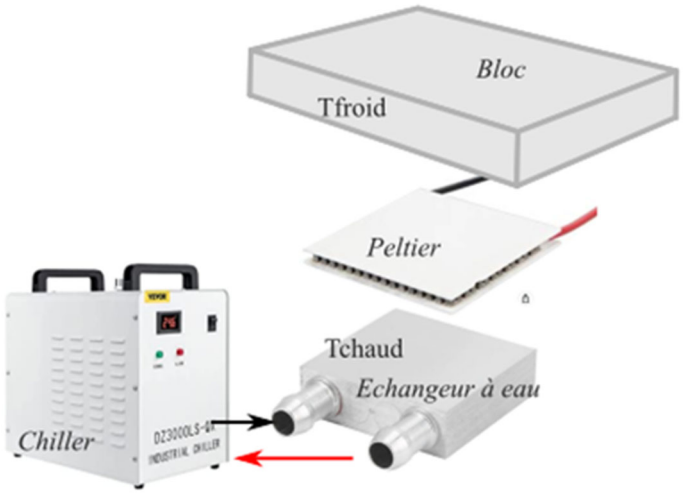

Figure 3 : Intégration de la cellule Peltier dans le système. La cellule est pressée à l'aide de deux vis entre le bloc où l'on souhaite contrôler la température et un échangeur à eau.

[1] https://tetech.com/wp-content/uploads/2013/11/tem_thermoelectric_module_mounting_procedure.pdf

Goupil (Goupil, Ouerdane, et Apertet 2013) propose un modèle d'une cellule Peltier, l'Éq. 1 décrit la partie thermique tandis que l'Éq. 2 est associée à la partie électrique. Pour cette dernière, nous avons trois effets, dans l'ordre de l'équation. L'effet thermoélectrique qui donne un flux de chaleur proportionnel au courant où P en W/A est le coefficient Peltier, un terme de dissipation liée à la résistance R et un terme de conduction thermique liée à la conductance thermique et à l'écart de température entre les deux faces notées T_{froid} et T_{chaud}. Le flux de chaleur prélevé du côté froid et celui donné du côté chaud, sont respectivement notés Φ_f et Φ_c. Il est important de distinguer la quantité de chaleur Q en Joule, du flux de chaleur Φ en Watt, qui est dQ/dt.

$$\Phi_f = P \cdot i - \frac{R}{2} \cdot i^2 - K \cdot (T_f - T_c)$$
$$\Phi_c = P \cdot i + \frac{R}{2} \cdot i^2 - K \cdot (T_f - T_c) \qquad \text{Éq. 1}$$

Du point de vue électrique, la cellule Peltier se comporte comme une résistance R en série avec une source de tension dont la tension est proportionnelle à la différence de température, où α est lié au coefficient Seebeck et au nombre de jonction thermoélectrique de la cellule. Le modèle électrique est en convention récepteur.

$$U = \alpha \Delta T + R \cdot i \qquad \text{Éq. 2}$$

La résistance R, 1.4 Ω, est directement donnée par le constructeur de la cellule Peltier. Les autres paramètres du modèle doivent être identifiés à partir des caractéristiques fournies. La Figure 4 et la Figure 5 permettent d'identifier la conductance K, soit environ 2.7 W/K et le coefficient Peltier, soit environ 25.5 W/A. La Figure 7 quant à elle permet de déterminer le coefficient α de l'Éq. 2, soit 0.075 V/°C.

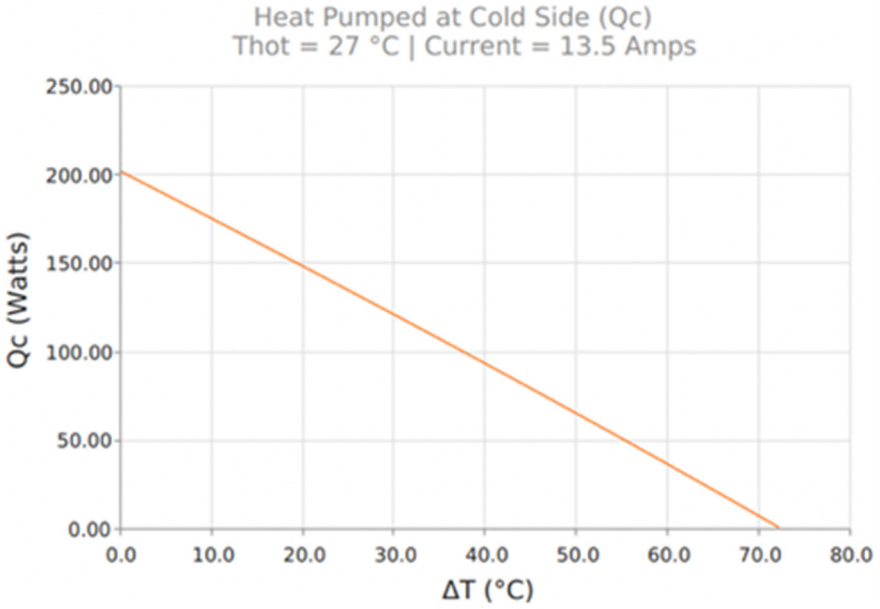

Figure 4 : Flux thermique Qc en fonction de la différence de température Th - Tc (Laid 387005665 extrait de la documentation)

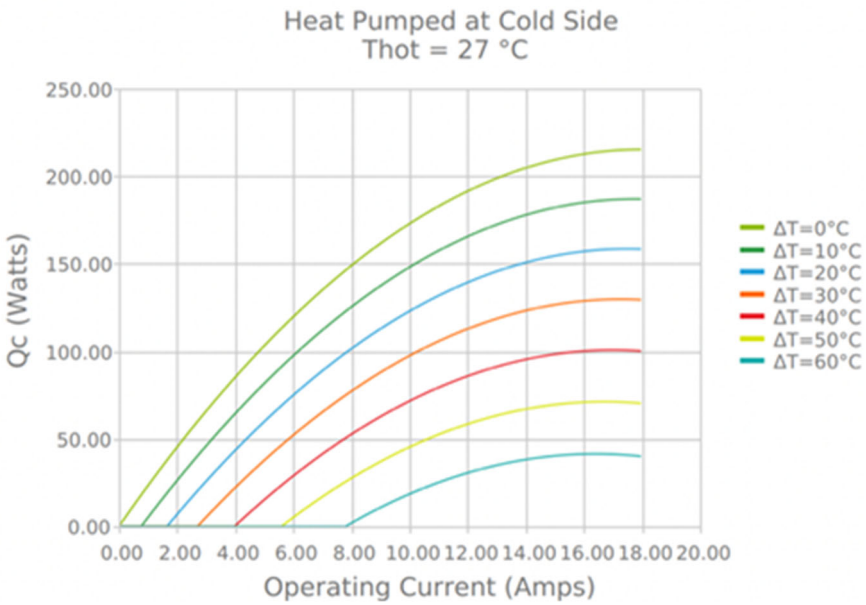

Figure 5 : Caractéristiques électrique et thermique de la cellule Peltier (Laid 387005665 extrait de la documentation)

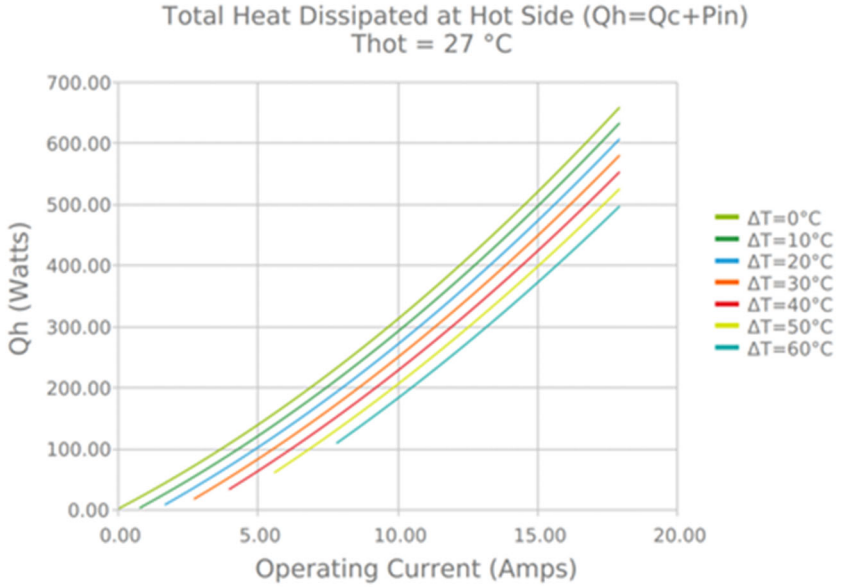

Figure 6 : Puissance dissipée sur la face chaude de la cellule Peltier (Laid 387005665 extrait de la documentation)

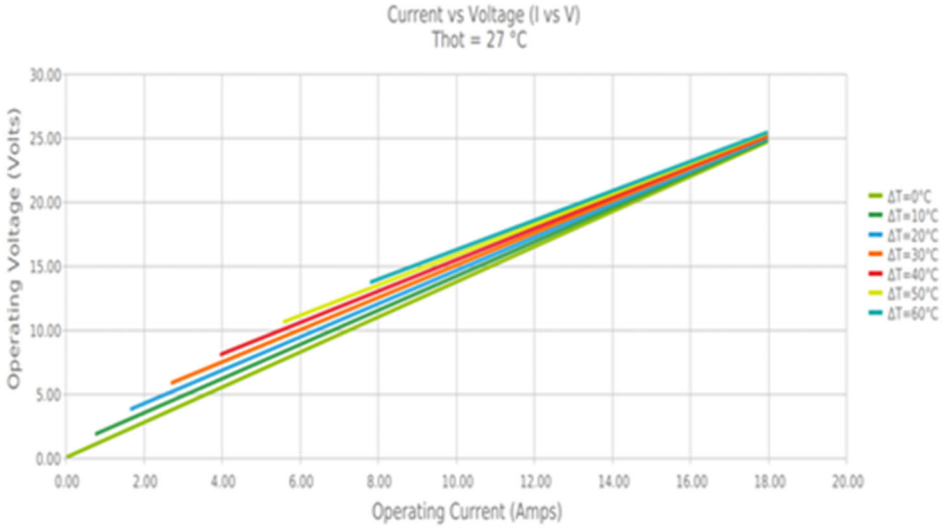

Figure 7 : Caractéristique électrique de la cellule Peltier (Laid 387005665 extrait de la documentation)

À partir d'un premier principe sur le bloc de masse m et de capacité thermique c_v, l'Éq. 3 donne l'évolution de la température du bloc à partir du flux de la cellule Peltier. On suppose ici qu'il n'y a pas de fuite et que la température du bloc est homogène.

$$c_p m \frac{dT_f}{dT} = \Phi_f \qquad \text{Éq. 3}$$

La température du côté chaud résulte de l'interaction entre la cellule qui tend à le chauffer et le fluide du thermocryostat (*Huber Minichiller 300 OLÉ*) qui tend à le refroidir. Dans le pire cas, un courant de 5 ampères et une différence de température entre la face froide et chaude de 30°C, la puissance dissipée sur la face chaude est d'environ 90 W. Le thermocryostat quant à lui est capable d'absorber 140 W pour une différence de température entre la face froide et chaude de 30°C. Donc même en imposant -5°C sur le côté chaud, le thermocryostat sera capable de maintenir une température constante. Dans cette modélisation on supposera que T_c est constant,

$$T_c = 20°C \qquad \text{Éq. 4}$$

2.2 - Alimentation à découpage et modèle électrique

La Figure 8 présente la structure à découpage utilisée pour alimenter la cellule Peltier. Par rapport à une structure linéaire, cette solution offre un bon rendement de conversion, néanmoins elle génère plus de perturbations électromagnétiques. Afin de limiter ces perturbations liées aux forts dv/dt engendrés par le découpage, le filtre est placé au plus près du pont en H. Afin de respecter les règles d'association des sources, un temps mort est introduit par le hardware du module PWM (pulse width modulation) du microcontrôleur. Différentes stratégies de modulation sont possibles, ici nous avons simplement pris une modulation bipolaire. Le circuit de puissance (Boost XL de Texas) que l'on enfiche sur la carte du microcontrôleur (F28379d) possède la commande rapprochée nécessaire aux contrôles des interrupteurs MOSFETs.

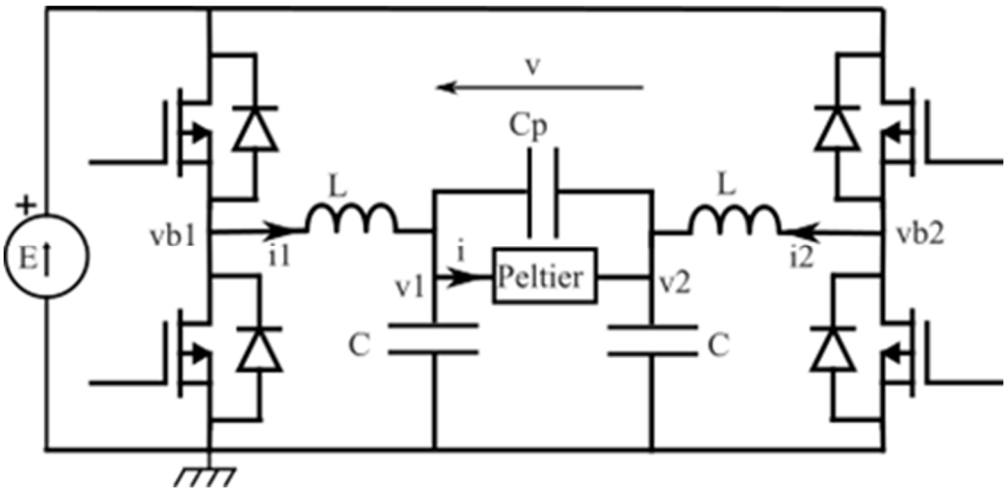

Figure 8 : Schéma électrique du pont H et du filtre placé sur la cellule Peltier avec $\boldsymbol{L = 100\ \mu H, C = 10\mu F\ et\ C_p = 100\mu F}$

En écrivant les relations électriques sur les éléments, on a :

$$L \cdot \frac{di_1}{dt} = V_{b1} - V_1$$
$$L \cdot \frac{di_2}{dt} = V_{b2} - V_2 \qquad \text{Éq. 5}$$

$$C \frac{dv_1}{dt} = i_1 - i_p - i_{cp} \qquad \text{Éq. 6}$$

$$C\frac{dv_2}{dt} = i_2 + i_p + i_{cp}$$
$$C_p\frac{d(v_1 - v_2)}{dt} = i_{cp}$$

À partir des Éq. 5 et Éq. 6, le système est représenté par sa composante en mode différentiel Éq. 7 et sa composante en mode commun Éq. 8.

$$\frac{d(v_1 - v_2)}{dt} = \frac{i_1 - i_2}{C + 2C_p} - 2i_p$$
$$\frac{d(i_1 - i_2)}{dt} = -\frac{v_1 - v_2}{L} + v_{b1} - v_{b2} \qquad \text{Éq. 7}$$

$$\frac{d(v_1 + v_2)}{dt} = \frac{i_1 + i_2}{C}$$
$$\frac{d(i_1 + i_2)}{dt} = -\frac{v_1 + v_2}{L} + v_{b1} + v_{b2} \qquad \text{Éq. 8}$$

Sur le mode commun $(v_1 + v_2)$ et le mode différentiel $(v_1 - v_2)$, il y a des éléments de filtrage, filtre LC, qui annihilent/réduisent les ondulations de tension liées aux découpages à 20 kHz. Le modèle s'intéresse donc à la valeur moyenne sur une période de découpage comme décrit dans l'Éq. 9 à partir de la fonction de modulation f_m que l'on décompose en f_m^0 et Δf_m.

$$< v_{b1} >= (f_m^0 + \Delta f_m)E \qquad \text{Éq. 9}$$
$$< v_{b2} >= (f_m^0 - \Delta f_m)E$$

Sur le mode commun, l'amortissement étant uniquement dû aux éléments parasites des éléments, le point de polarisation à 0.5, soit f_m^0, devra être positionné progressivement afin d'éviter des oscillations trop importantes. Le mode différentiel, soit la tension sur la cellule U, est piloté avec le paramètre Δf_m.

L'utilisation du kit de Texas Boost xl offre une plateforme fiable. On observe aussi que lors de la connexion de l'alimentation de la partie puissance, un fort appel de courant nécessaire à charger les condensateurs sur le point de fonctionnement, vient saturer l'alimentation.

2.3 - Mesure de la température et communication SPI

Afin d'asservir en température le système, il faut mesurer la température T_c, pour cela une sonde de type PT100 est utilisée. La température est déduite de la résistance d'un élément de platine, lequel est proportionnel à la température. La dépendance à la température étant faible (<1% par degré) un circuit électrique doit conditionner le signal, par exemple à l'aide d'un point de Wheatstone suivi d'un amplificateur d'instrumentation et d'un ADC. Ici, nous utilisons un circuit de conditionnent tout intégré comme le MAX31865. Le conditionneur et le microcontrôleur vont communiquer au travers du bus de donnée, ici un bus de type SPI « Serial Peripheral Interface ». Ce bus dispose de quatre fils clock, Master Output Slave Input, Master Input Slave Output, Chip Select, notés respectivement CLK, MISO, MOSI, CS. Dans un premier temps, le microcontrôleur (uC) transmet des données pour configurer le conditionneur, par exemple le uC transmet 8 bits par exemple 80h pour écrire sur le registre de configuration (Figure 9) puis l'octet à placer dans le registre.

REGISTER NAME	READ ADDRESS (HEX)	WRITE ADDRESS (HEX)	POR STATE
Configuration	00h	80h	00h
RTD MSBs	01h	—	00h
RTD LSBs	02h	—	00h
High Fault Threshold MSB	03h	83h	FFh
High Fault Threshold LSB	04h	84h	FFh
Low Fault Threshold MSB	05h	85h	00h
Low Fault Threshold LSB	06h	86h	00h
Fault Status	07h	—	00h

Figure 9 : Registre disponible du circuit de conditionnement de la PT100 (cf la documentation du MAX31865 pour plus de détail)

Après cette étape de configuration, toutes les 20 ms, le uC écrit sur le bus 01h pour lire l'octet RTD MSB, et juste après il écrit 02h puis lire l'octet RTD LSB

REGISTER	RTD MSBS (01h) REGISTER								RTD LSBS (02h) REGISTER							
Bit	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
RTD Resistance Data	MSB	—	—	—	—	—	—	—	—	—	—	—	—	—	LSB	Fault
Bit Weighting	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	—
Decimal Value	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	—

Figure 10 : Registre qui stocke la mesure de température

À partir de ces deux octets et du codage de la température donnée dans la documentation, nous allons reconstruire la température sur un nombre codé en virgule flottante, c'est cette grandeur que nous allons appeler T_m et qui s'actualise toutes les 20 ms. Il y a donc un retard pur sur la mesure mais au vu de la dynamique du système, supérieur à 20 s, il ne sera pas nécessaire de prendre en compte ce retard pour le dimensionnement du correcteur.

Par ailleurs, nous disposons d'une autre sonde de température sur le côté chaud afin de faire un monitoring du système ou de détecter une défaillance, par exemple pour détecter que le thermo cryostat ne fonctionne pas.

2.4 - Le microcontrôleur

Le microcontrôleur interface les capteurs (communication SPI), pilote le hacheur et dialogue avec le superviseur (communication série). L'implémentation du code sur le microcontrôleur F28379D de Texas Instrument (TI) se fait en langage C dans l'outil de développement de TI, code composer studio. Néanmoins, le recours à des outils de génération de code automatique comme Embedded Coder de Matlab-Simulink permet de faciliter l'implémentation en se concentrant sur le correcteur plus que sur le code C. De plus grâce à des outils tierces de TI, « Embedded coder », nous avons directement accès à la configuration des périphériques du microcontrôleur.

3 - Asservissement

Le modèle du système et la mesure de la température vont permettre de mettre en place un asservissement.

3.1 - Linéarisation du modèle

Les équations de Éq. 1 à Éq. 4 et Éq. 6 forment un système multivariable d'ordre 3 non linéaire avec comme variables d'état T, $v_1 - v_2$, $i_1 - i_2$.

Le système pourrait nécessiter plusieurs correcteurs, soit dit autrement un retour d'état, afin de contrôler chaque variable d'état. Néanmoins la mise en place d'un correcteur PI sur la variable de la grandeur d'intérêt, la température, est plus appropriée pour le niveau visé. Le correcteur PI se place donc au niveau de l'écart entre la température de consigne et la température de mesure. Puisque qu'il n'y a pas de boucle de contrôle sur le courant, le taux de variation de la tension, soit de Δf_m, va être limité afin d'éviter la saturation en courant des inductances à 7.5A. En limitant le rate, les équations électriques sont dans un état quasi-stationnaire où $U = v_1 - v_2$, soit en moyenne $v_{b1} - v_{b2}$.

Du côté des non-linéarités associées au comportement de la cellule Peltier, nous proposons de linéariser le système car c'est l'approche habituelle pour aborder les systèmes non linéaires. Le système s'écrit, in fine, sous la forme de l'Éq. 10.

$$\dot{T}_f = f(T_f, U) \qquad \textit{Éq. 10}$$

Prenons une linéarisation autour de U = 0V et l'on décompose la température comme illustré à l' Éq. 11 avec T_f^0 le point polarisation et $\widetilde{T_f}$ la variation de température autour de ce point.

$$T_f = T_f^0 + \widetilde{T_f} \qquad \textit{Éq. 11}$$

Le point de polarisation étant pour $\boldsymbol{U = 0}$, la température de polarisation $\boldsymbol{T_f^0}$ est telle que $\boldsymbol{f(T_f^0, 0)}$ égale 0, soit $\boldsymbol{T_f^0 = T_c}$.

En linéarisant le système, Éq. 10, soit un système à deux variables autour du point de fonctionnement $\boldsymbol{T_f^0}$, on obtient le système linéaire de l'équation

$$\dot{\widetilde{T_F}} = 0 + \frac{\partial f(T_f^0, 0)}{\partial T_f} \cdot \widetilde{T_f} + \frac{\partial f(T_f^0, 0)}{\partial U} \cdot \widetilde{U} \qquad \textit{Éq. 12}$$

La mise en place de correcteur PI se faisant traditionnellement dans le domaine de Laplace, l'**Éq. 12** est réécrite sous la forme de Éq. 13, ici bien que les notations soient identiques, les fonctions $\widetilde{T_f}(p)$ et $\widetilde{U(p)}$ dépendent de la variable de Laplace, notée p.

$$\frac{\widetilde{T_F}(p)}{\widetilde{U}(p)} = \frac{\frac{\partial f}{\partial U}}{p - \frac{\partial f}{\partial T_F}} \qquad \textit{Éq. 13}$$

soit

$$\frac{\widetilde{T_F}(p)}{\widetilde{U}(p)} = \frac{\frac{P}{R}}{\frac{P \cdot \alpha}{R} + K} \Big/ \left(\frac{cm}{\left(\frac{P \cdot \alpha}{R} + K\right)} \cdot p + 1\right) \qquad \textit{Éq. 14}$$

Il s'agit donc d'un premier ordre qui peut se faire sous la forme classique de l'**Éq. 15** avec $\boldsymbol{\tau = 22\ s}$ et $\boldsymbol{\beta = 4,5\ °C/V}$.

$$\frac{\widetilde{T_F}(p)}{\widetilde{U}(p)} = \frac{\beta}{1 + \tau \cdot p} \qquad \textit{Éq. 15}$$

Pour valider le modèle, le système (en boucle ouverte) est soumis à plusieurs réponses indicielles selon le rapport cyclique des PWM. La Figure 11 montre l'évolution de la température en fonction du temps.

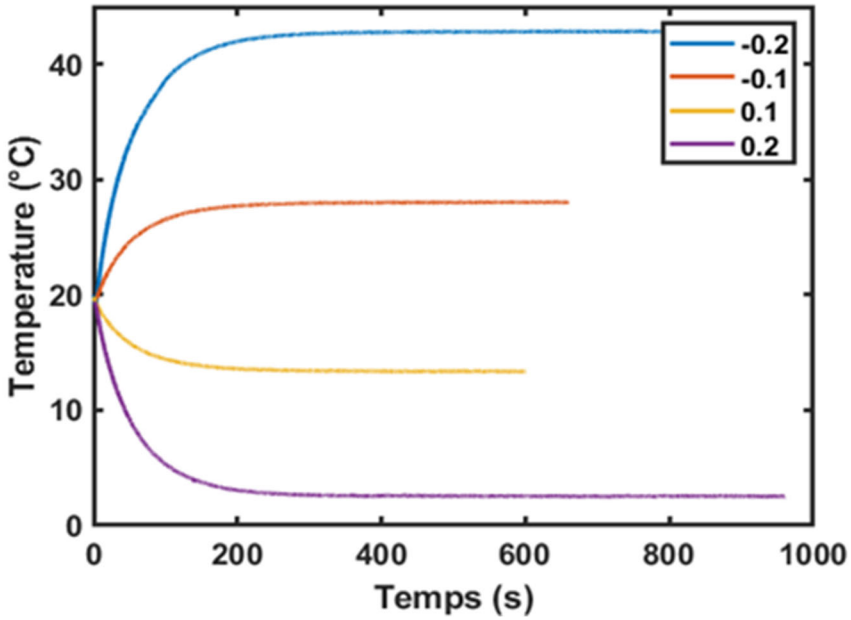

Figure 11 : Réponse indicielle du système pont en H et de la cellule Peltier mesurée. La légende indique le rapport cyclique choisi autour du point de polarisation, soit Δf_m. La tension d'alimentation du bus était autour de 12V (au final on utilisera 24V).

En première approximation, ce système s'apparente bien à un premier ordre. Néanmoins, les non-linéarités apparaissent car il y a une dissymétrie entre les rapports cycliques positifs et négatifs ($\pm\Delta f_m$.) et une non proportionnalité entre l'essai à 0.1 et 0.2 de rapport cyclique. On peut alors déterminer la constante de temps moyenne τ et le gain moyen β du système. Dans ce cas, on identifie $\boldsymbol{\tau = 40\ s}$ et $\boldsymbol{\beta = 5\ °C/V}$, sachant que toutes les courbes ne donnent pas exactement les mêmes paramètres à cause des non-linéarités du système. Disons que les paramètres identifiés sont représentatifs des zones de fonctionnement visées.

3.2 - Dimensionnement du correcteur - Correcteur Proportionnel Intégral

L'asservissement en boucle fermée avec le correcteur proportionnel intégral (PI), va permettre d'annuler l'erreur statique et d'augmenter la dynamique de réponse du système. La forme du PI est celle proposée à Éq. 16.

$$C(p) = K_i \cdot \left(1 + \frac{1}{\tau_i \cdot p}\right) \qquad \text{Éq. 16}$$

Le schéma bloc du système en boucle fermé est représenté dans la Figure 12.

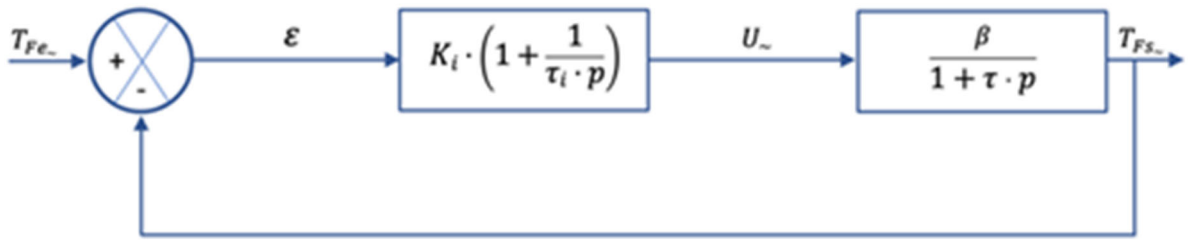

Figure 12 : Schéma bloc du système en boucle fermée avec T_{Fe} pour la température de consigne et T_{Fs} pour la température du système, soit T_F

Le correcteur est dimensionné de façon à compenser le pôle dominant, soit $\tau_i = \tau$. On règle ensuite la bande passante à l'aide du gain Ki. On rappelle que la bande passante de la FTBF correspond à la fréquence où la FTBO est égale à 0dB.

Puisque le retour de la boucle est unitaire, la fonction de transfert en boucle fermée s'exprime comme :

$$\frac{\widetilde{T_F}(p)}{\widetilde{T_{F_e}}(p)} = \frac{1}{1 + \frac{\tau \cdot p}{\beta \cdot K_i}} \qquad \text{Éq. 22}$$

La constante de temps du système en boucle fermée est $\tau_c = \frac{\tau}{\beta \cdot K_i}$, si l'on souhaite augmenter légèrement la dynamique du système, prenons $\tau_c = \frac{\tau}{5}$, on obtient $K_i = \frac{5}{\beta} = 0.06$ et $\frac{K_i}{\tau_i} = 0.001$.

4 - Implémentation « Embedded Coder » de Matlab-Simulink

Le microcontrôleur est programmé sous Matlab-Simulink à l'aide d'outil de génération de code C tel que « embedded coder ». Les codes Matlab-Simulink étant partagés, nous détaillons uniquement la structure du code. Le code se décompose en plusieurs parties : 1) la partie communication avec le capteur de température (communication SPI) ; 2) la partie réception/transmission des données avec le superviseur (communication série) ; 3) le correcteur avec la gestion des modules PWM du microcontrôleur ainsi que d'un affichage/contrôle de l'état du système à l'aide de LEDs/interrupteur.

4.1 - Communication avec le capteur de température (communication SPI)

La Figure 13 décrit la communication avec le capteur de température. Il y a une phase d'initialisation du capteur puis une phase de lecture répétitive de la température.

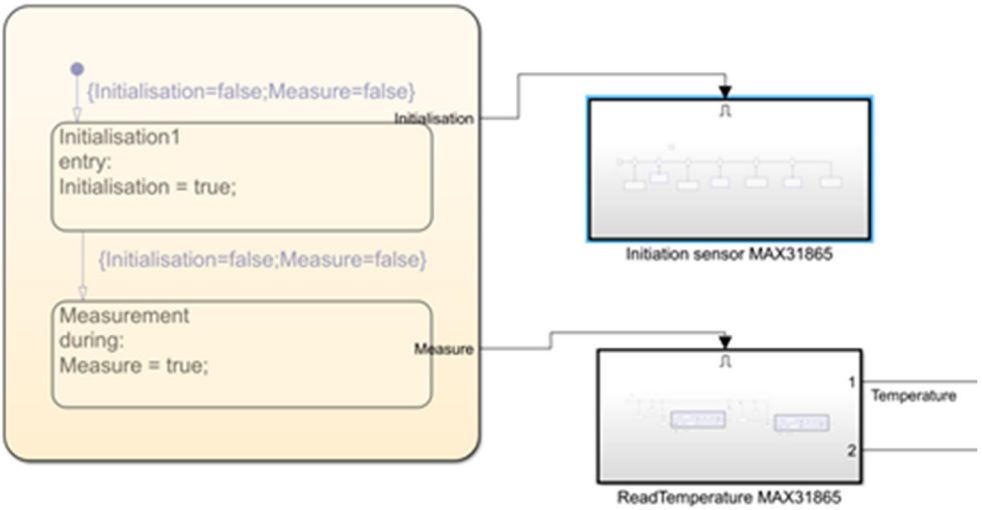

Figure 13 : Initialisation du capteur de température et lecture de la température au travers d'une machine d'état ("state flow chart") activée au pas de temps du système (20 ms). Les blocs « initialization » et « read » exécutent la communication SPI du microcontrôleur pour configurer le capteur puis pour lire la température.

4.2 - Réception/transmission des données avec le superviseur (communication série)

La Figure 14 décrit la transmission sur communication série des différentes températures. Un codage à virgule fixe (fixed point number) a été utilisé, 8 bits pour l'entier signé et 8 bits pour la

virgule fixe. On fait attention dans le bloc « convert » de Matlab- Simulink, soit pour maintenir la valeur représentée, soit son mot binaire.

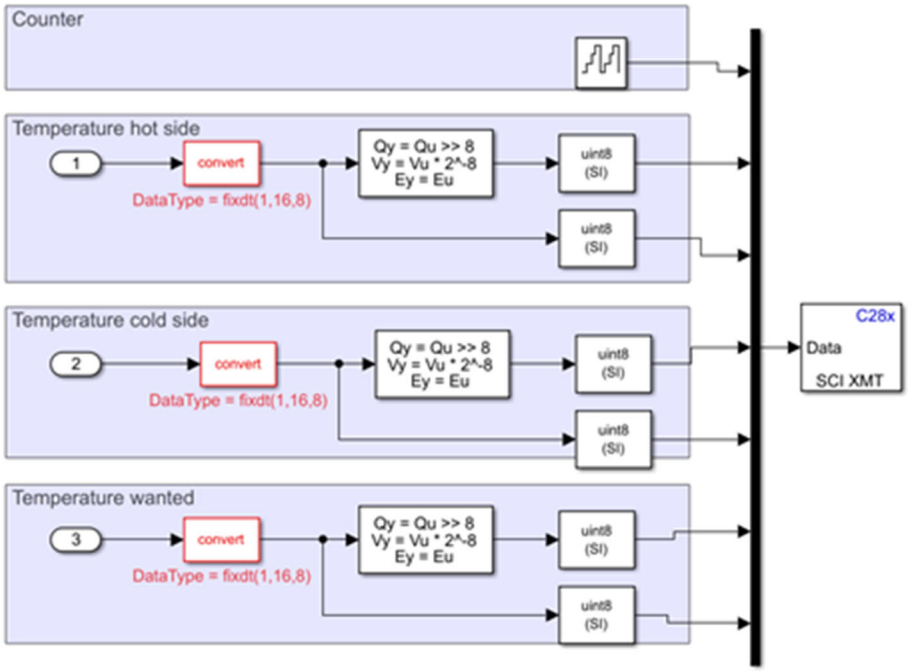

Figure 14 : Communication des températures au travers du port série « SCI », la trame série se compose de série d'octets qui sont un compteur, puis 2 octets pour chaque température. La température est transmise en virgule fixe avec un octet pour la partie entière et un octet pour la partie fractionnaire.

La Figure 15 correspond à la température de consigne reçue par le microcontrôleur via le port série. La transmission se fait via 4 octets, pour le moment seul deux octets sont exploités pour coder la température (8 bits pour l'entier signé et 8 bits pour la partie fractionnaire). Le microcontrôleur via le code en Figure 15, reconstitue la consigne de température.

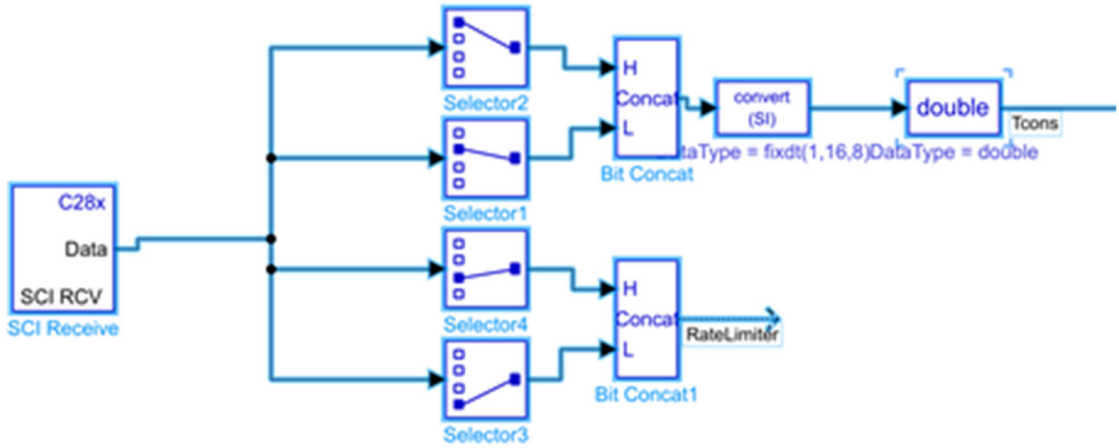

Figure 15 : Code simulink pour reconstituer la consigne de température via les octets reçus sur le port série

La partie entière de la température, codée comme un entier signé sur 8 bits, donne une plage de -128.000°C à 127,996°C, et la partie fractionnaire donne une résolution de 1/2^8, soit 0.0039 °C. Le codage choisi est donc adapté à notre plage de travail ainsi qu'à la résolution du convertisseur de température de 0.01 °C (MAX31865). Néanmoins dans l'ensemble des programmes sur microcontrôleur et sur RaspBerry nous travaillerons en flottant, seule la communication série se fera en virgule fixe.

4.3 - Correcteur et gestion des entrées/sorties

La Figure 16 intègre le correcteur PI avec la consigne (en haut à gauche) et la mesure (au niveau du rond vert). La partie à droite est liée aux PWM pour le contrôle de la partie puissance, tandis que la partie en bas à gauche est liée à la partie affichage ou interrupteur de démarrage. L'effet intégral du correcteur est actif uniquement lorsque la partie puissance n'est pas inhibée. Il y a aussi une protection basée sur la température de l'échangeur à eau, en effet si celui-ci devient trop chaud alors la partie puissance est désactivée et la LED rouge s'allume.

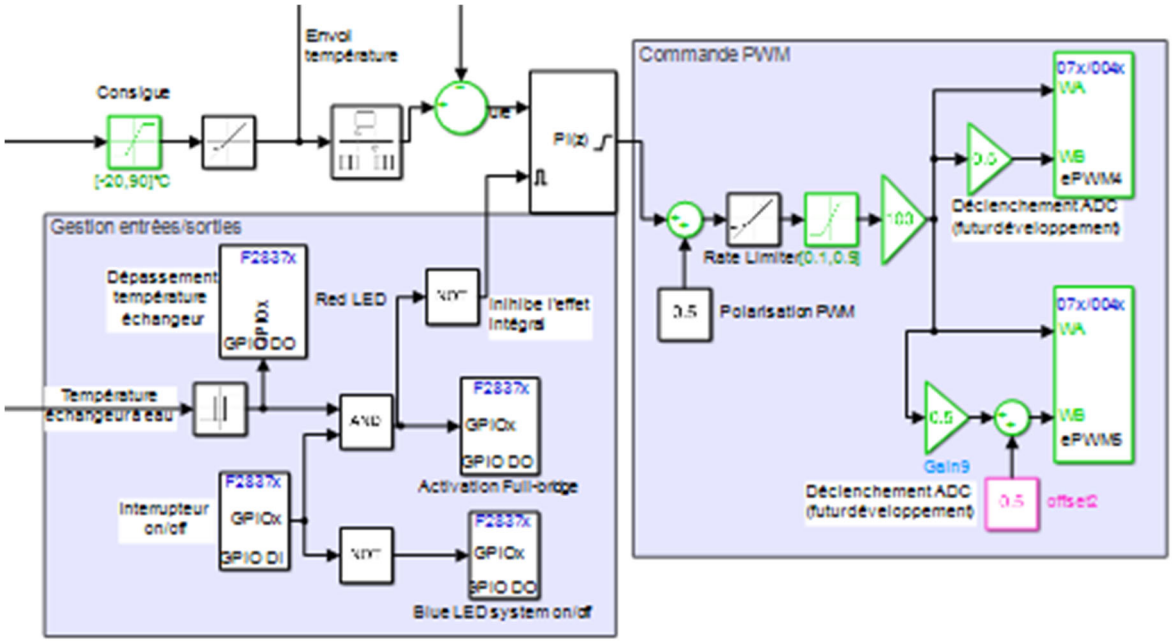

Figure 16 : Correcteur avec la consigne, la mesure, la commande et la gestion des entrées/sorties (LEDs, interrupteur)

5 - Interface utilisateur

L'interface utilisateur hébergée sur le module RaspBerry permet de superviser le système, soit de :

- Visualiser en temps réel l'évolution des températures sur un écran connecté en HDMI ;
- Il peut agir sur la consigne mais dans le fonctionnement normal il ne fait que retransmettre la consigne qu'il a lui-même reçu de l'ordinateur maître via un socket TCP/IP.

Les librairies utilisées pour se faire sont :

- Tkinter pour la création de l'interface ;
- Sockets pour la communication avec le PC maître via un protocole client-serveur sur sockets TCP/IP ;
- PySerial pour la communication avec le microcontrôleur via une liaison série sur USB ;
- Fixed Point pour la conversion des nombres de virgule fixe à virgule flottante ;
- Matplotlib pour afficher la courbe.

Le tableau, ci-dessous, illustre simplement l'algorithme de l'interface utilisateur.

server_program() ->Mise en place du serveur TCP/IP, réception de la consigne de température du PC (on utilisera un codage ascii)
plotTemperature() ->Réception de la consigne du uC, conversion des données de virgule fixe vers virgule flotante et affichage d'un graphique

sendTemperature() -> Saisie de la valeur ou transfert de celle reçue par le serveur TCP/IP et conversion de la température de consigne en virgule fixe
Lancement des Threads liés aux fonctions server_program(), plotTemperature() et sendTemperatuer().

Tableau 1 : Schéma illustrant le fonctionnement du programme

Les fonctions sont exécutées dans des boucles infinies, les threads sont ordonnancés selon le « Round Robbin » c'est-à-dire que chacun des threads se donne la parole à tour de rôle après un certain temps si l'exécution de la fonction du thread est plus longue que le temps imparti.

La mise en place de l'interface s'est faite avec Tkinter afin d'offrir une prise en main rapide.

Le programme est codé à partir des fonctions, l'utilisation d'objets permettrait une meilleure intégration du code mais demande aussi des notions de programmation plus avancées.

Les deux sous-parties suivantes introduisent brièvement la communication par socket et les conversions utilisées pour coder la température à chaque étape.

5.1 - Communication de la Raspberry avec le PC maître au travers de socket par réseau TCP/IP

Pour envoyer la température de commande, l'utilisateur fait appel aux protocoles TCP/IP et aux sockets (objet liant une adresse IP à un port). On établit une communication client-serveur, sur le réseau local du laboratoire. Le diagramme d'état ci-dessous illustre le principe.

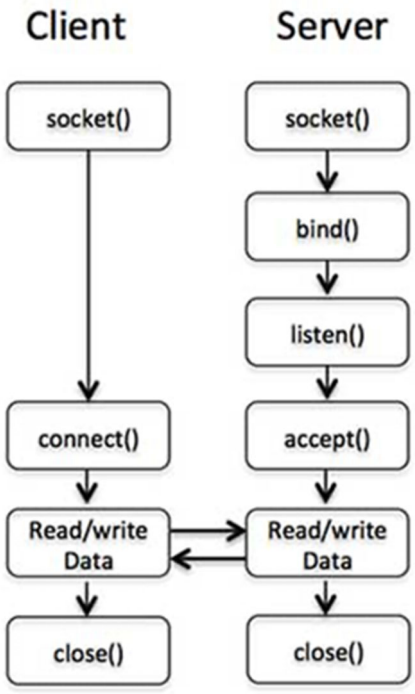

Figure 17 : Diagramme d'état de la communication client-serveur [4]

Le serveur (le Raspberry PI) se met en mode « écoute » sur un port spécifié puis le client (le PC sur lequel l'utilisateur travaille) se connecte au serveur et les deux communiquent jusqu'à ce que le client ferme la connexion. Au cours de leur échange, le client envoie la température de consigne désirée. Lorsque le Raspberry PI reçoit la température de consigne, il la transmet au microcontrôleur via la liaison série, puis le microcontrôleur se charge de l'asservissement de la cellule Peltier pour atteindre la température de consigne.

5.2 - Gestion des types

Le transfert des données requiert de bien manipuler les codages utilisés pour coder l'information. Nous allons illustrer quelques passages clefs du code implémenté dans le Raspberry qui décode la trame envoyée par le uC.

dataSerialRec = ser.read(140), permet de lire 140 octets stockés dans le buffer du bus série. La variable dataSerialRec est de type bytes de Python et ***dataSerialRec[i+3:i+5].hex()*** affiche les bytes de i+3 à i+4 en hexadécimal sous la forme d'une chaine de caractère.

C'est ici que l'on utilise la manière dont le microcontrôleur envoie la température, c'est à dire que le 3e et 4e octet représentent la température codée en virgule fixe signé (8 bits pour la partie entière et 8 bits pour la partie fractionnaire). Alors à partir d'une chaîne de caractère qui représente le mot binaire en hexadécimal (***'0x'+dataSerialRec[i+3:i+5].hex()***) et du format utilisé (***signed=1, m= 8,n=8***), on crée un type « fixed point » de la librairie fixedPoint, avec la commande ***FixedPoint('0x'+dataSerialRec[i+3:i+5].hex(), signed=1, m= 8,n=8)***.

Cet élément de type fixed point est ensuite converti en flottant et ajouté à la liste qui stocke toutes les températures reçues. On obtient alors la commande suivante : ***tempOrderFromuC.append(float(FixedPoint('0x'+dataSerialRec[i+5:i+7].hex(), signed=1, m= 8,n=8)))***

L'utilisation de la librairie permet de rendre plus lisible et plus robuste la conversion.

6 - Conclusion & perspective

Cet article propose un système de régulation de température à base d'une cellule Peltier. Il est d'un point de vue pédagogique assez complet dans le domaine du génie électrique et de l'informatique industrielle. Une partie requiert des compétences en protocole de communication d'informatique industrielle et d'analyse de signaux, pour établir la communication SPI entre le capteur de température et le microcontrôleur ou la communication série entre le uC et le Raspberry. Il y a évidemment une grande partie liée à l'automatique, l'asservissement de système non linéaire. Des connaissances sur l'électronique de puissance sont aussi nécessaires pour la mise en œuvre d'une alimentation à découpage avec une très faible ondulation. Pour faire communiquer le PC maître avec le Raspberry PI nous utilisons les réseaux et établissons une communication via sockets. Enfin, le Raspberry PI aborde la notion de thread avec un ordonnancement des tâches en « Round Robin » (Real-time operating system) mais aussi de la conversion des types et de l'interface graphique.

L'ensemble des sous parties peuvent être améliorées, stratégie de modulation sur la MLI, type d'asservissement, utilisation de la notion d'objet pour la programmation de l'interface sur python, etc. Ainsi pour des élèves de niveau Master, il offre la possibilité d'approfondir des notions spécifiques d'automatique, de programmation, d'électronique, etc. L'ensemble des documents sources sont disponibles sur Culture Sciences de l'Ingénieur - eduscol (https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/controle-de-temperature-via-une-cellule-peltier).

Remerciement & Annexe

Un grand merci à Christophe SALLÉ et Christopher DOUGLAS pour leur aide dans l'entreprise de ce système.

Annexe - Coût du système

Notre système de contrôle de température relève d'un certain coût pour sa conception et sa mise en œuvre. On propose ici de budgéter l'ensemble des composants nécessaires au fonctionnement du système (cf. image ci-dessous).

Matériel	Prix H.T	T.V.A	Prix T.T.C
Alimentations			
1 x Module hacheur Texas Instruments BOOSTXL-DRV8305EVM	83,18 €	20,00%	99,82 €
1 x Module d' Alimentation linéaire régulée Double DC LT1083 24V 5A	24,92 €	20,00%	29,90 €
Filtre passe-bas			
6 x Condensateur EPCOS B32562 10µF 63 V ac	24,78 €	20,00%	29,74 €
2 x Inductance Murata 60B104C 100µH - 7.5 A	9,80 €	20,00%	11,76 €
2 x Bornier 3 contacts GMKDS 3/3-7.62	7,60 €	20,00%	9,12 €
Échangeurs thermiques			
1 x Cellule Peltier Laird Thermal Systems 387005665	83,33 €	20,00%	100,00 €
1 x Thermocryostat Huber Minichiller 300 OLÉ	3 458,33 €	20,00%	4 150,00 €
Mesure de température			
2 x Resistance to digital converter MAX31865	57,92 €	20,00%	69,50 €
2 x Sonde de température PT100 Capteur RTD PT100 362-9812	15,36 €	20,00%	18,44 €
Systèmes embarqués			
1 x Raspberry PI 3 - 1gb	27,72 €	20,00%	33,26 €
1 x Carte de développement Ti LAUNCHXL-F28379D	43,17 €	20,00%	51,80 €
Affichage & autres			
1 x Écran	83,33 €	20,00%	100,00 €
1 x Carte microSD 16 gb	8,33	20,00%	10,00 €
1 x Câble hdmi vers DVI	6,67	20,00%	8,00 €
1 x Adaptateur secteur Raspberry PI	9,75	20,00%	11,70 €
	Total H.T :		Total T.T.C
	3 944,19 €		4 733,04 €

Figure 18 Gestion budgétaire du système de contrôle de température

On remarque que le prix, élevé, du système est notamment dû au thermocryostat dont le coût hors taxe s'élève à 3 458, 33 €, on pourrait refroidir la cellule Peltier avec un ventilateur fixé sur la face chaude, cela diminuerait grandement le coût du système pour environ 500 €.

Références :

[1]: Goupil, Christophe, Henni Ouerdane, et Yann Apertet. 2013. « Thermoélectricité - Thermodynamique et applications ». Consulté le 10 juin 2022. *Ressources énergétiques et stockage*, avril. Disponible sur : https://doi.org/10.51257/a-v1-be8080

[2]: Cellule Peltier [Documentation constructeur]. Consulté le 10 juin 2022. Disponible sur : https://www.mouser.fr/datasheet/2/988/datasheet_PCX15_6_19_F1_4040_TA_RT_W6-2298292.pdf

[3]: Thermocryostat. [Documentation constructeur] Consulté le 11 juin 2022. https://www.huber-online.com/download/pdf_datasheets/englisch/3006.0089.98.PDF

[4]: Diagramme d'état de la communication client-serveur, consulté le 16 juin 2022, disponible sur : https://www.tutorialspoint.com/perl/perl_socket_programming.htm

[5]: Convertisseur de température. Consulté le 12 juin 2022. Disponible sur : https://datasheets.maximintegrated.com/en/ds/MAX31865.pdf

[6]: Raspberry PI 4. Consulté le 14 juin 2022. Disponible sur : https://www.planete-domotique.com/blog/2019/09/16/raspberry-pi-4-modele-b-il-se-met-en-quatre-pour-vous/

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay