Rapport de projet de 4ème année

Machine à café Tweek

Jean Wasilewski & Pierre Letousey

Sommaire

Préface

L'idée de ce projet à germée après de nombreux cafés distribués sur le sol car une machine à café défectueuse ne prenait pas en compte l'absence de gobelets. A cela il faut ajouter que l'obligation de payer par pièces et non par badge nous a amener à imaginer un système payable avec une simple carte étudiante, à la manière d'un restaurant universitaire.

Introduction

Au cours de notre quatrième année en Informatique Micro-électronique et Automatique à PolytechLille, nous avons été amené à réaliser un projet. Nous avons proposé aux encadrants la réalisation d'une machine à café. Celle-ci permettrait de produire un café de qualité tout en ajoutant des fonctionnalités de paiement sans contact. Nous avons donc commencé par établir un cahier des charges préliminaire.

1> Mise en place du cahier des charges

Voici la liste des objectifs que nous avions essayé de réaliser:

- Broyage des grains déjà torréfiés
- Régulation d'un moteur en vitesse avec charge variable
- Réutilisation des éléments d'une cafetière existante (carte électronique hors d'usage)
- Étalonnage du débitmètre
- Analyse du chauffe-eau
- Calibration de la pompe à eau
- Propulsion de l'eau chauffée
- Régulation d'une pompe à eau en pression et en débit
- Régulation du chauffe-eau en température
- Modélisation des éléments du système
- Création d'une interface homme-machine
- Installation d'un RaspberryPi avec un écran tactile
- Gestion des stocks de café, d'eau et de gobelets
- Gestion des différents éléments (pompe, compresseur, chauffe eau, …)
- Réalisation d'une interface graphique permettant de :
    - Moudre le café
    - Sélectionner la quantité d'eau à distribuer (café court ou long)
    - Sélectionner un café parmi une présélection de café
- Possibilité de noter son café

- Possibilité de retenir un choix de café particulier
- Création de profils créés à partir de la carte étudiante
- Lecture d'information en NFC sur la carte étudiante
- Création d'une base de données basée sur ces profils
- Proposition des cafés à partir de ceux les plus commandés et/ou mieux notés
- Possibilité de retenir certains choix et de les recommander
- Création d'une carte permettant la commande des différents éléments de la machine à café (pompe, chauffe-eau, et d'autres sorties)
- Création d'une carte permettant de faire l'acquisition des différents capteurs (comme le débitmètre)

1.1> Découpage en tâches

Pour permettre une plus simple dissociation des objectifs, et donc de rendre plus efficace le travail en équipe, nous avons divisé ces objectifs en tâches. Voici la liste des tâches.

- Caractériser le débitmètre
- Caractériser le chauffe-eau
- Caractériser la pompe
- Détermination du schéma de fonctionnement et éventuels correctifs de régulation
- Réalisation de la carte électronique de commande
- Programmation de l'Arduino pour commander la carte
- Programmation du RaspberryPi pour l'interface tactile et la lecture NFC
- Estimation temps de réponse des éléments
- Gestion automatique du café
- Dimensionnement capsule et du dispositif de broyage
- Création d'un système dispenseur de gobelets
- Design d'engrenages pour les gobelets
- Plus de fonctionnalités si temps restant
    - Recharge par le Net
    - Possibilité d'ajouter du sucre
    - Interface épurée
    - Packaging pratique

1.2> Matériel utilisé

Afin de réaliser ces objectifs, nous avons utilisé le matériel suivant. Dans un soucis de budget, la plupart des éléments utilisés pour ce projet ont été récupéré sur une ancienne machine à café Nespresso qui était défectueuse (carte électronique cramée).

Sur cette machine, nous avons récupéré :

- Un débitmètre
- Une pompe à eau
- Un chauffe-eau
- Tubes et embouts
- Réservoir d'eau avec valve anti-retour
- Mécanisme d'infusion de capsules de café

Nous avons emprunté à l'école le matériel électro-informatique suivant :

- Arduino Uno
- Arduino Léonardo (remplacé par un Arduino Uno par la suite)
- RaspberryPi
- Carte SD 16G
- Shield Arduino Lecteur NFC
- OptocoupleurET-OPTO RELAY 4

Nous avons commandé chez les fournisseurs GoTronic et RadioSpare par le biais de l'école :

- Un Ecran tactile USB 2,8'' DFR0275 (reçu le 18 mars 2015)
- Un Capteur de température (reçu le 4 février 2015)
- Un Thermocouple (reçu le 4 février 2015)

Enfin, nous avons acheté avec l'école ou apporté le matériel d'électronique de puissance suivant :

- Deux Contacteurs de puissance (max 32A, 400V)
- Cables électriques
- Boitier interrupteur
- Deux coffrets d'isolation électrique

2> Étude préalable des éléments

2.1> Caractérisation du débitmètre

Lors d'une première séance "pratique", nous avons étudié le comportement du débitmètre.
Après recherches, nous avons appris qu'il en existe deux types. Le premier type est assimilable à une résistance variable dont la valeur dépend du débit. Le deuxième type se comporte comme un type tout ou rien qui effectue un contact entre deux des pins lorsque l'ailette interne fait un tour. Le débit est donc déduit par la fréquence des impulsions que délivre le débitmètre.
Notre débitmètre est un composant crée par Digmesa. Il s'agit d'un débitmètre à turbine, du deuxième type.
Le montage ci après permet de récupérer les impulsions du débitmètre sous forme TTL.

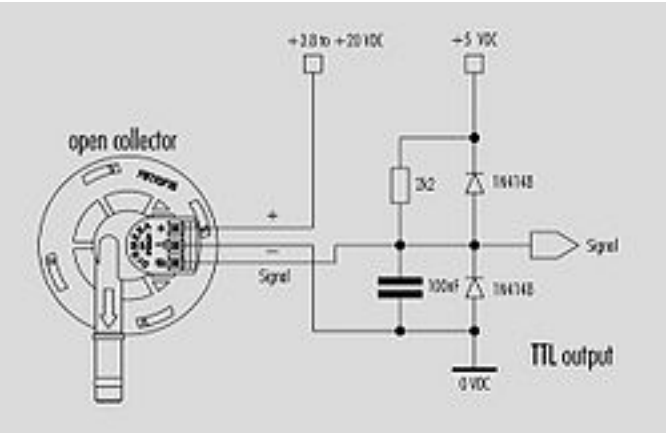


Schéma du montage permettant de récupérer les impulsions du débitmètre

Le volume d'eau mesuré s'exprime alors sous cette forme : $V = n * \frac{1}{Kdeb}$ avec n le nombre d'impulsions, V le volume en L et Kdeb la constante fournie dans la doc constructeur Kdeb = 1925 impulsions/L.
Le débit n'est pas une grandeur mesurée directement avec notre débitmètre, mais elle est déduite en choisissant un temps Δt suffisamment court pour considérer le débit "instantané". L'expression du débit "pseudo-instantané" est alors la suivante : $\frac{\Delta V}{\Delta t} = \frac{1}{Kdeb} * \frac{\Delta n}{\Delta t}$ avec Δn la variation du nombre d'impulsion pendant le Δt.

Obtenir le débit est ici utile car, d'après les informations collectées, le temps d'extraction du café doit être contrôlé (entre 20s à 25s) et le café doit être de 5cl pour un ristretto et de 12cl pour un lungo.

Il est important de noter aussi que ce capteur a une influence sur la pression : en effet, il peut induire une perte de pression (au maximum) de 0,43 bar.

2.2> Réalisation de l'interface commande-puissance

Afin de pouvoir séparer circuit de commande et alimentation de puissance, nous utiliserons un optocoupleur et des contacteurs pour l'alimentation de la pompe et du chauffe-eau (tout deux alimentés en 230V alternatif 50 Hz). L'optocoupleur est utilisé afin de bénéficier d'une isolation galvanique entre les deux parties, pour plus de sécurité.
La commande d'alimentation est envoyée par l'Arduino à l'optocoupleur. L'optocoupleur commande alors le contacteur (relais de puissance), ce qui alimente l'élément choisi.
L'optocoupleur utilisé est sous forme de circuit "prêt à utiliser" (ET-OPTO AC-OUT 4) qui dispose de quatre optocoupleurs isolés. Nous en utiliserons un pour commander la pompe (OCT0) et un deuxième pour commander le chauffe-eau (OCT1). L'objectif d'une des séances a été d'établir ce dispositif d'alimentation, de comprendre le fonctionnement de l'optocoupleur dont nous disposons et de mapper les optocoupleurs aux sorties de l'Arduino.
Par la suite, nous avons été contraint de changer de type de commande. En effet, la carte d'octocoupleurs permettait la commande mais n'était pas capable de couper le contacteur. Cela est du au principe même du fonctionnement du contacteur. En effet,le contacteur est composé d'une bobine (entre A1 et A2) qui vient attirer un bloc aimanté de façon à créer un contact entre les six broches de puissance (connecteurs 1 à 6). Le problème était que cette bobine demandait un courant trop important que l'octocoupleur gérait mal en rendant impossible la coupure de la bobine.
Pour palier à ce problème, nous avons choisi d'utiliser une carte de commande d'octocoupleurs commandant des relais (ET-OPTO RELAY4). Après vérification à l'aide de la documentation technique des relais, nous avons constaté que ceux-ci sont bien capable de couper la commande des contacteurs.

2.3> Fabrication d'un shield Arduino

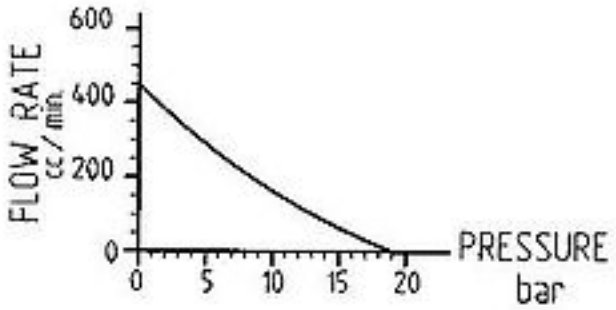
Nous avons donc été amené à créer un shield Arduino avec une carte de prototypage rapide qui réalise les fonctions suivantes :

- intégrer un connecteur 10 broches compatible pour :
  - relier la sortie digitale IO 2 à l'optocoupleur OCT0 qui commandera l'alimentation de la Pompe
  - relier la sortie digitale IO 3 à l'optocoupleur OCT1 qui commandera l'alimentation du Chauffe eau

- intégrer le circuit de récupération des impulsions du débitmètre TTL et échantillonner ce signal sur l'entrée digitale IO8
- intégrer un circuit type pont diviseur de tension permettant de récupérer la température par le biais du capteur de température
- disposer d'une fonction "Boutons Arrêt d'urgence/Commande manuelle" pour la pompe et le chauffe eau.

2.4> Caractérisation de la pompe

La pompe récupérée est un composant fabriqué par Invensys. Il s'agit d'une pompe à piston qui permet de délivrer une pression allant jusqu'à 19 bar au maximum. Cette pression est fonction du débit en sortie de la pompe, selon la caractéristique suivante.



Caractéristique Pression/Débit de la pompe

Elle se commande en "tout ou rien" et est alimentée en 230 V alternatif 50Hz. Le paramètre permettant de fixer le point de fonctionnement sur le graphique précédent est la section du tuyau en sortie. Nous n'avons pas la main sur ce paramètre étant donné que nous possédons déjà le système de tuyaux avec les embouts adaptés de section S = 3 mm^2. En revanche, nous la commanderons en comparant le volume de café désiré et le volume actuel donné par le débitmètre. Il s'agit alors de déterminer à quels moments commander la pompe, sachant qu'il faut que l'extraction dure entre 20s et 25s et que le chauffe eau induit un retard à déterminer.

Lors d'une séance de manipulation de la pompe, nous l'avons commandé au moyen de l'Arduino pendant plusieurs durées pré-établies (2s, 5s et 10s) afin de déterminer si elle possède un régime transitoire négligeable. Nous avons ensuite relevé le volume d'eau qu'elle avait délivré grâce à une éprouvette graduée. Plusieurs séries de mesures ont été effectuées à chaque fois. Lors de cette manip, le tuyau de sortie n'était pas connecté au chauffe-eau afin que la mesure ne soit pas influencée, il donnait directement dans l'éprouvette. Cette manipulation nous a également permis de tester notre débitmètre et

notre manière d'échantillonner le signal qu'il nous fournit, en comparant le volume déduit par celui ci et le volume réel relevé avec l'éprouvette.
Ainsi, nous avons déterminé que quelque soit le temps d'allumage (pour une durée supérieure à 2s), la pompe fournit un débit constant de 5,2 mL/s = 0,312 cc/min.

2.5> Étalonnage du capteur de température du chauffe-eau

Afin de réguler le chauffe eau en température, nous avons besoin de connaître le fonctionnement du capteur de température intégré sur celui ci. Il s'agit en fait d'une thermistance (résistance variable dont la valeur de la résistance dépend de la température), ici à coefficient de température négatif (CTN).
Il n'y a visiblement pas de référence constructeur dessus. Nous avons donc décidé d'effectuer un étalonnage de celle ci. Il existe un dispositif en salle électronique comportant des capteurs de températures déjà étalonnés disposés sur une plaque qui peut être chauffée par un chauffage électrique. Nous avons "collé" avec de la pâte thermique notre thermistance sur cette plaque et effectué un relevé de la résistance en fonction de la température indiquée par les capteurs déjà présents. Le problème est que le chauffage ne pouvait pas être régulé, il continue de chauffer tant qu'on l'alimente. N'ayant donc pas de valeur de température stabilisée, à cause de l'inertie thermique, les valeurs de résistances relevées sont complètement non significatives. Cette manipulation aura tout de même soulevé un point à prendre en compte : le capteur de température étant en métal, il faudra déterminer l'inertie qu'il induit dans la mesure et si celle ci est négligeable, sinon, comment faire avec.
Le soucis pour une approximation linéaire est que la plage de température est trop importante. En effet, la température de l'eau pour le café doit être entre 88°C et 92°C, mais on doit aussi pouvoir utiliser le capteur à une température de 50°C pour prévoir un mode "économie d'énergie", lorsque la machine n'est pas utilisée pendant 10min par exemple. Après recherches, la loi d'évolution de la résistance en fonction de la température est de la forme : $$\frac{1}{T} = A + B\ln(R_{Th}) + C(\ln(R_{Th}))^3$$

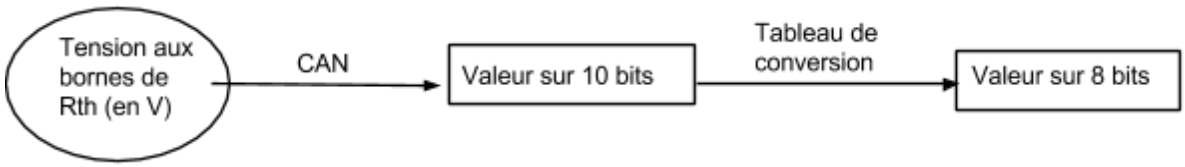
Nous avons ensuite essayé une autre technique afin d'étalonner la thermistance : au moyen d'un bain d'eau que l'on peut chauffer et d'une sonde de température précise.
Après relevé de l'évolution de la résistance en fonction de la température au moyen du bain marie, nous avons constaté et vérifié que notre sonde de température se comporte comme une sonde standardisée PT100. Nous avons trouvé une fiche constructeur de ces sondes standardisées qui vérifie les caractéristiques de notre capteur. Sur la plage de température que nous avons besoin d'utiliser (25°C - 105°C), nous pouvons négliger le terme en ln^3, ainsi l'équation régissant la résistance en fonction de la température

peut s'exprimer ainsi : $R_{Th} = R_0 \times \exp\left(\beta \times \left(\frac{1}{T} - \frac{1}{T_0}\right)\right)$ avec R_0 = 100 kOhm la résistance de référence à T_0= 25°C, β = 4334 Kelvin le coefficient fourni par le constructeur pour l'approximation sur la plage 25°C-100°C.

Ainsi, la représentation des données pour récupérer la température est la suivante : au moyen d'un pont diviseur, nous récupérons une tension sur l'Arduino image de la résistance et donc de la température avec la relation ci-dessus. Nous ne pouvons pas inclure dans le code directement un calcul avec la fonction exponentielle, cela prendrait , entre autres, trop de temps de calcul et ne serait pas optimisé étant donné que l'on connait la résolution du CAN de l'Arduino. Pour cela, nous passons par un tableau de conversion sur 256 valeurs que nous avons calculé au moyen d'une feuille de calcul disponible sur le wiki du projet.

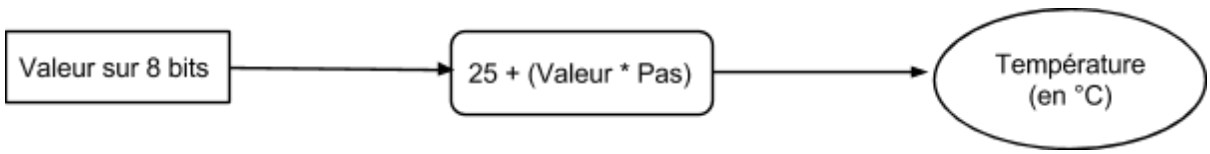
Voici un schéma de principe :

Coté Arduino :



La Raspberry (ou le PC) récupère ensuite cette valeur sur 8 bits et en déduit la température avec la connaissance du "pas" entre 2 températures : $Pas = \frac{(105 - 25)}{255}$

Coté Raspberry



Cette méthode permet d'alléger le temps de traitement sur l'Arduino et surtout de ne pas perdre en précision et résolution à cause de la transmission de l'information "température", qui s'effectue sur 8 bits.

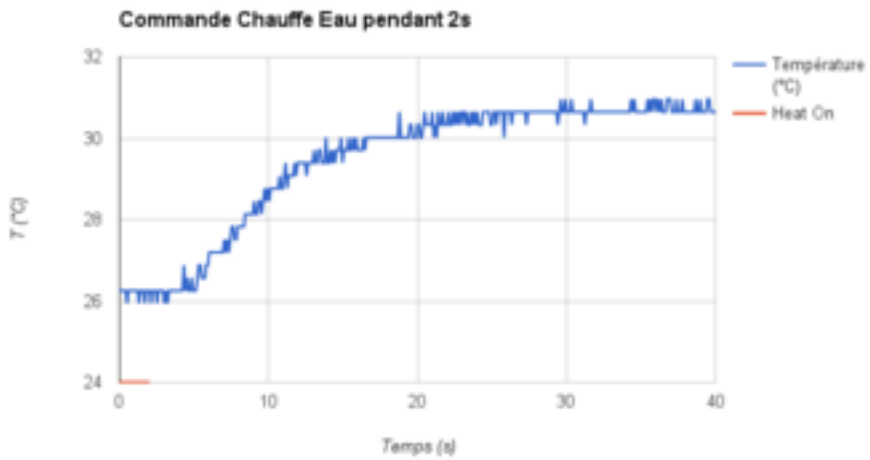
2.6> Caractérisation du chauffe-eau

Le chauffe eau que nous avons récupéré est de type thermoblock. Il s'agit en fait d'une résistance chauffante autour de laquelle se situe un serpentin dans lequel l'eau circule, le tout étant isolé dans un châssis en métal. Les principaux avantages de ce système thermoblock est qu'il conserve très bien la chaleur et qu'il permet de ne chauffer que la quantité d'eau nécessaire, la petite quantité qui passe à l'intérieur. On peut alors considérer l'échange thermique entre le chauffage et l'eau quasi instantané et uniforme. L'inconvénient majeur est en revanche, comme pour tout système thermique, l'inertie thermique importante qu'il possède, notamment due à son corps en métal.

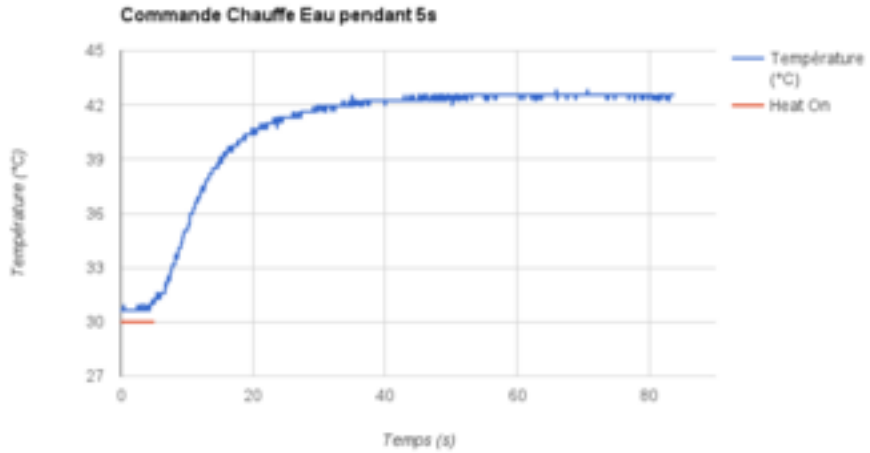
Nous n'avons pas trouvé de données constructeur sur ce chauffe eau. La seule information inscrite est la puissance électrique qu'il absorbe : 1200 W. Il sera alimenté, comme pour la pompe, en 230V alternatif 50Hz, en tout ou rien. Il s'agit maintenant d'estimer ses caractéristiques (inertie, temps caractéristiques, gains en température en fonction de l'entrée) afin de mettre en place un plan de régulation adapté. Ne possédant aucune information sur ce système, nous ne procéderons pas à une étude théorique thermique qui serait trop approximative compte tenu de la géométrie du système et trop fastidieuse à mettre en place. Nous avons choisi de procéder par identification, en relevant les courbes d'évolution de la température, pour différents temps de commande.

Résultats :

A chaque fois, la réponse peut s'exprimer de la façon suivante : on constate un temps de retard pur au démarrage dû à l'inertie, puis une évolution s'apparentant à celle d'une réponse à système du 1er ordre. On peut écrire la fonction de transfert du système sous cette forme : $H(P) = \frac{K}{1+\tau P} e^{-T_r P}$ (modèle de Broïda)



Réponse en température pour 2s d'alimentation



Réponse en température pour 5s d'alimentation

Réponse en température pour 10s d'alimentation, puis arrivée d'eau à t=75s

Remarque : On peut observer sur le 3ème graph l'influence du passage de l'eau dans le thermoblock sur la température. Celui ci se traduit comme une perturbation avec un comportement type 1er ordre sur le thermoblock. Avec ces paramètres (temps, gain), on pourra par la suite estimer la compensation à ajouter afin de maintenir la température du thermoblock constante malgré la chauffe de l'eau.

Ainsi, on peut récapituler ceci dans le tableau suivant :

Temps de commande	**Gain en température K**	**Temps de retard Tr**	**Temps de réponse à 5%**
2 s	4,3 °C	5,2 s	14 s
5 s	12 °C	5,2 s	25 s
10 s	24 °C	5,2 s	37 s

On remarque un gain en température en fonction du temps d'allumage quasi constant dans ces intervalles d'environ 2,4°C par seconde, ainsi que le temps de retard est bien constant.

2.7> Détermination du plan de régulation

Rappelons le cahier des charges de la régulation :

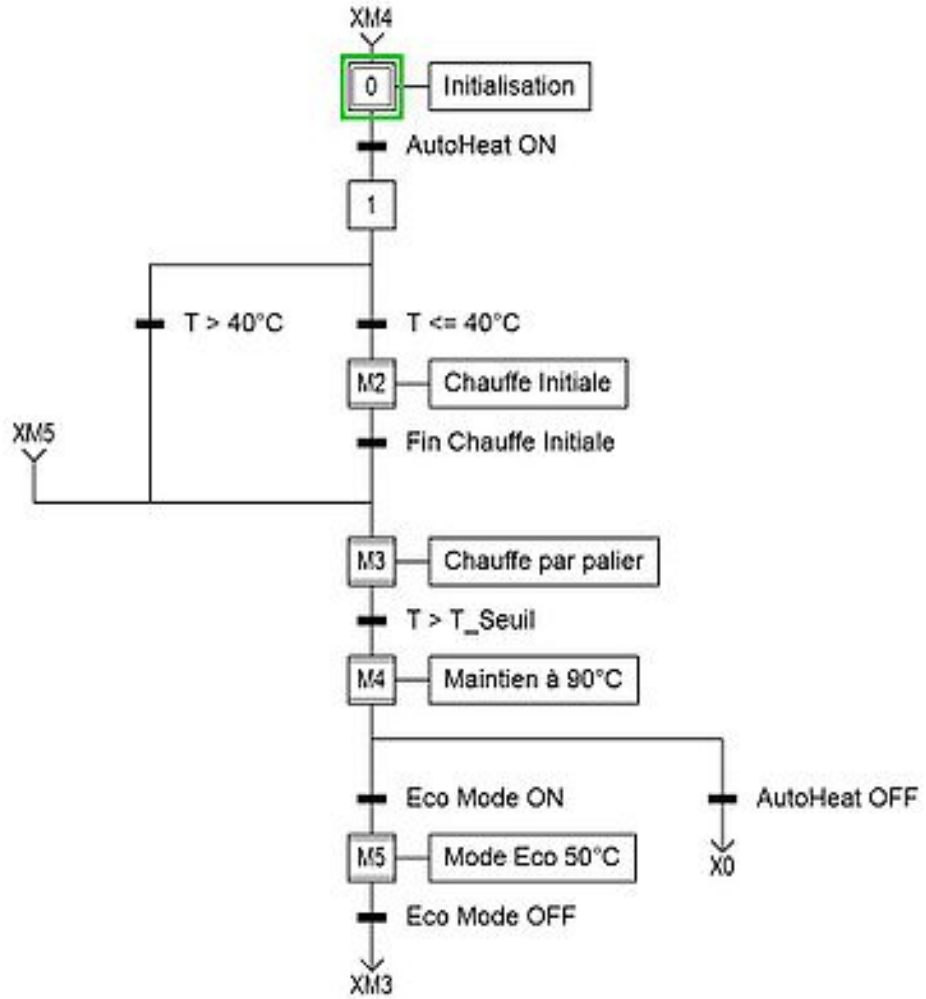
- chauffer l'eau à 90°C, avec comme erreur acceptable +/- 2°C (conformément aux informations recueillies)
- pas de dépassement de la consigne : en effet, le système conservant très bien la chaleur, attendre qu'il se refroidisse serait trop long et surtout, si il y a dépassement après 100°C, on crée un générateur de vapeur... ce qui est bien sûr très dangereux pour le système et pour la sécurité de l'utilisateur
- avoir un temps de réponse acceptable pour l'utilisateur (actuellement, à titre d'exemple, pour la cafetière Nespresso, l'étape de chauffe n'excède en général pas 60s)
- mode économie d'énergie : la machine est destinée à être utilisée publiquement, par des utilisations ponctuelles et pas uniformément réparties dans le temps, il est donc important que la machine ne gaspille pas de l'énergie lorsqu'elle n'est pas utilisée. Ainsi, le bon compromis serait que la température soit maintenue à environ 50°C après 10min d'inutilisation.

Choix de la régulation :

Ici ne possèdant pas de commande réglable, nous ne pouvons pas choisir une régulation continue de type PID ou prédicteur de Smith, car on commande le thermoblock via un relais. Une commande MLI n'est pas envisageable dans notre cas, à cause du faible "pouvoir de commutation" des contacteurs de puissance que nous avons récupéré. Une simple régulation Tout ou Rien type thermostat ne serait pas suffisante, car ceci engendrerait des dépassements et des oscillations fortes autour de la température de consigne compte tenu du rapport Temps caractéristique sur Temps de retard du système $\frac{\tau}{T_r} \approx 1,66$ .Nous nous situons donc dans le cas "Régulation par Boucles multiples" d'après la méthode de Broïda.

Dans notre cas d'utilisation, un algorithme par boucles multiples, basé sur des commandes Tout ou Rien, est possible et peut satisfaire au cahier des charges. En effet, la consigne ne change pas d'ordre de grandeur : il faudra toujours chauffer l'eau dans un intervalle compris entre 88°C et 92°C.

On propose alors l'algorithme suivant :



Grafcet illustrant le principe de l'algorithme de régulation de la température

Principe :

- 1) Chauffe Initiale : Au début de la phase de chauffe, on réalise une chauffe importante (15 s), l'objectif étant d'approcher grossièrement la température auprès de la température de consigne et qu'elle se soit stabilisée (attente de 12s) avant de passer à l'étape de chauffe par palier.
- 2) Chauffe par palier : Il s'agit ici d'effectuer ensuite une succession de "petites" chauffes jusqu'à être dans la plage de température acceptable, tout en tenant compte de l'inertie thermique. Ces successions de chauffe sont définies ici par 2s de chauffe, puis 6s d'attente de l'établissement de la température avant de déterminer si on reste dans la phase Chauffe par palier ou si on entre dans la phase Maintien au chaud. La température de seuil a été déterminée à 82°C, pour prendre en compte le phénomène d'inertie.
- 3) Maintien au chaud : On se situe alors proche ou dans la plage de température. Il s'agit alors d'effectuer des petites chauffes ponctuelles permettant de maintenir la température dans l'intervalle (refroidissement par le milieu ambiant). Cette étape prend également en compte le maintien au chaud lorsque de l'eau passe à travers le chauffe eau et fait donc chuter sa température (compensation plus forte : cycle de 3s de chauffe puis 3s d'attente). On peut, à partir de cette étape, commander la pompe pour élaborer un café si un café a été demandé.

Un avantage de la chauffe palier par palier permet d'attendre que la chaleur s'établisse uniformément dans le thermoblock. Ceci permet de limiter les problèmes d'inerties thermiques, y compris celui de notre capteur (sans mettre ces paliers d'attente, ce que l'on observerait en température ne serait pas nécessairement la température au coeur du thermoblock).

Evolution de la température du chauffe eau avec la régulation

Résultats de la régulation

On constate sur le graphique ci dessus les 3 étapes de chauffe : les étapes 1 et 2 s'effectuant ici entre 0s et environ 80s puis l'étape de maintien après cette date. La température évolue ainsi de façon presque linéaire pendant les étapes 1 et 2, puis reste dans la plage 88°C-92°C. A t=140s, la pompe est commandée dans cette exemple afin de pouvoir visualiser le maintien en température lorsque l'eau circule.
On remarque donc que cette régulation valide les 2 premiers critères du cahier des charges (chauffer l'eau dans la plage et pas de dépassement de cette plage). Cependant, le temps de réponse avant de pouvoir élaborer un café est un peu plus long (environ 80s).

3> Programmation des différents éléments

Pour la partie programmation, le défi résidait dans le fait de faire communiquer tout les éléments à l'aide d'un seul programme. Pour cela, nous avons subdivisé les fonctions principales dans différents modules. Ceci nous permettait entre autre de minimiser le nombre d'éléments reliés à la partie "puissance" du projet (détaillé plus loin)

Le code source du projet est disponible sur GitHub (voir page du wiki pour les différents liens des programmes).

3.1> Le Shield Arduino NFC

L'objectif de l'utilisation du Shield NFC est de permettre à un utilisateur de payer un café, et donc de s'identifier sur la machine, à l'aide de carte Lille 1. Pour cela, nous nous sommes premièrement orientés sur l'utilisation de l'IDE Arduino et de la librairie dédiée au PN532. Puis, nous avons décidé de passer à un langage C pour avoir une meilleure visibilité sur les registres. Par conséquent, nous avons utilisé le compilateur avr-g++ pour compiler le projet. Il nous a donc était nécessaire de transposer une partie des fichiers C et C++ pour les rendre utilisables avec notre programme C.

La communication se fait par système de questions-réponses. Le serveur sur la Raspberry envoie une demande de données, et l'Arduino répond en conséquence. La seule difficulté a résidé dans la problématique des données à envoyer pour l'identification. Les cartes de Lille 1 sont des cartes NFC Mifare Classic 1K. Elles sont composées de 16 secteurs de 4 fois 16 bytes. Le secteur qui nous intéresse est le douzième secteur, qui est encodé en utilisant les clés NFC publiques (donc accessible par tout le monde).

Les clés publiques en question sont :

	Clé
A	A0A1A2A3A4A5
B	B0B1B2B3B4B5

Ce douzième secteur contient, pour les carte étudiantes, l'INE, suivi du numéro étudiant (sur 8 chiffres), suivi du prénom, suivi d'un point puis du nom. Nous avons d'abord envisagé d'utiliser le nom et le prénom, cependant cela posait le problème des personnes avec le même patronymes. De plus, il arrivait que le nom de l'étudiant soit tronqué pour rester dans l'espace alloué au secteur 12. Par conséquent, nous utilisons le numéro d'étudiant, qui est toujours au même endroit sur le secteur. Le logiciel compare alors ce numéro à une liste d'étudiant.
Par soucis de sécurité, nous n'avons pas de liste donnant une correspondance directe entre un numéro d'étudiant et son nom. En effet, la liste utilisée est un enchaînement de nom et de numéro étudiant encodé avec l'algorithme MD5. Pour récupérer le nom, il nous suffit alors de comparer le numéro étudiant que nous récupérons (que nous encodons en MD5) avec la liste. Ceci nous assure une confidentialité relative, dépendante directement de la robustesse de la non réversibilité du MD5.

Une fois la reconnaissance des cartes étudiantes terminées, nous avons essayé d'inclure les cartes Lille 1 dédiées au personnel. La première question que nous nous sommes posé est comment pouvons nous faire la différence, au niveau du lecteur, entre une carte étudiante et une carte du personnel. Nous avons remarqué que les cartes du personnel comporte, sur leur douzième secteur, deux fois le numéro de personnel (sur 6 chiffres). La répétition du numéro de personnel se situe à l'endroit où se situe le nom de l'étudiant sur la carte de l'étudiant. Le test est donc assez facile à réaliser. L'autre problème est : comment lié un numéro de personnel à un nom ? Nous n'avons pas encore résolu ce problème car cette table de liaison n'est pas accessible à PolytechLille. Éventuellement, si nous arrivons à accomplir tous les objectifs du projet, nous envisagerons de contacter les responsables de Lille 1 pour obtenir cette table de correspondance.

3.2> L'interface en ligne de commande utilisateur

Durant les phases de test, nous avons créé une interface en ligne de commande. Celle-ci permettait d'envoyer directement des commandes au HVC, comme l'allumage du chauffe-eau pendant un certain nombre de secondes; ou encore le pompage. Cette interface est restée en utilisation jusqu'à deux semaines avant la fin du projet, car elle nous permettait d'effectuer un suivi très simple de l'état des capteurs. Par la suite, à partir du commit ac55328, nous nous sommes orienté vers l'utilisation de l'écran fournis.

Durant la période où nous utilisions l'interface en ligne de commande, nous nous servions de celle-ci car elle nous fournissait des informations comme le PID. Cette information nous permettait de tuer le programme principal ainsi que tous ces fils lorsque celui ci plantait. Il affichait aussi le volume pompé, la dernière température relevée, l'identifiant lu par le lecteur NFC et la date à laquelle cette lecture s'était produite. Elle affichait aussi les dernières entrées du journal, avec des couleurs indiquant leur niveau d'importance (Debug, Info, Warning ou Error).

3.3> L'interface utilisateur finale

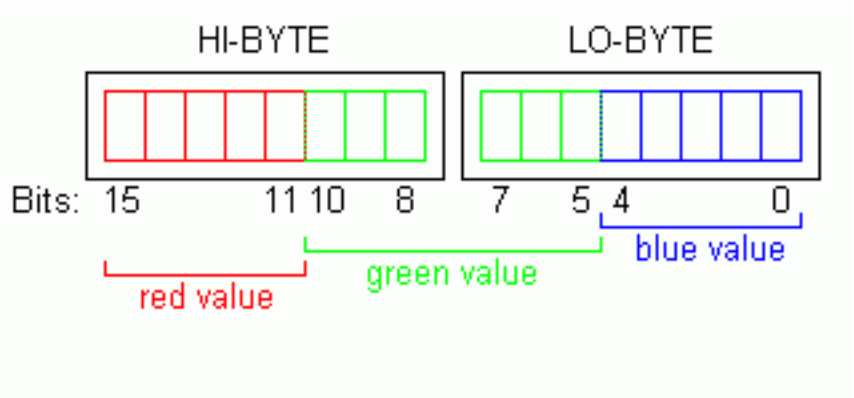
Pour l'interface utilisateur finale, nous nous reposons sur un module d'affichage RPUSBDisp de la société RoboPeak. Avant de se lancer dans la phase de programmation de l'interface, il nous a d'abord fallut rendre utilisable l'écran par le RaspberryPi. Pour cela, il a fallu créer un module chargeable par le noyau Linux. Une fois ce module créé, nous nous sommes rendu compte le noyau Linux que nous utilisions (celui fournis avec Raspbian) ne contenait pas les symboles nécessaires au chargement du module. Nous avons alors recompilé le noyau Linux avec les symboles manquants.

La compilation s'est avérée assez compliqué car pour éviter de faire de la cross compilation (ce qui aurait sûrement augmenter le taux d'échec de compilation) nous avons compilé le noyau directement à partir de la RaspberryPi ce qui, malgré son overclooking, était très long (une compilation prenait approximativement une nuit complète). Après quatre tentatives, nous avons réussi à obtenir un noyau fonctionnel.

Premier BMP affiché avec succès sur l'écran

Une fois l'écran utilisable, nous nous sommes lancés dans le développement de l'interface. Nous avons choisi, dans une optique de performance, de n'utiliser que la librairie X11. Le premier objectif était de réussir à afficher une image sur l'écran. Le format d'image utilisé était le BMP car il n'est pas compressé et peut être lu sans autres calculs. Après de petits cafouillages (liés au fait que le BMP représente une couleur de 0 à 255 et X11 de 0 à 65535) nous avons réussi à faire afficher notre première image. Cependant, le point négatif est que l'image est affichée en une trentaine de secondes. Nous sommes donc en train de travailler sur un moyen de faire afficher des images plus rapidement.

Après de longues recherches, nous nous sommes rendus compte qu'il n'était pas possible de réaliser une interface temps réel avec cet écran si l'on utilisait la librairie X11. En effet, celle ne communique pas assez rapidement avec l'écran (les recordes d'écriture étant de ~17 secondes par images avec X11). Le constructeur préconise en fait une utilisation à travers une interface USB. Cette interface nous oblige alors à réécrire notre interface homme machine.



Représentation des données en RGB565

Nous avons finalement réussi à envoyer à travers la librairie USB une première commande. Ensuite, nous avons essayé d'afficher une image sur l'écran. Pour cela, il a été nécessaire de créer un type de fichier intermédiaire car l'écran n'est pas capable de lire la plupart des formats d'images classiques. Pour rendre la conversion plus simple, nous avons choisi le type BMP comme fichier image source. Le type BMP organise la description des images de façon simple. Le fichier commence par un en-tête qui comporte des informations comme l'adresse de début de l'image, le nombre de bits par pixel et d'autres informations qui ne nous servent pas. Une fois ces deux informations récupérés (nous n'utilisons que des images à 24 bits par pixel, soit 8 bits pour le rouge, 8 bits pour le vert et 8 bits pour le bleu) nous convertissons notre image dans le format lisible par l'écran, qui est le RGB565. Ce format compresse (avec pertes) la précision des couleurs pour que celle-ci soit ordonnée avec 5 bits pour le rouge, 6 bits pour le vert et 5 bits pour le bleu.

Afin de rendre plus pratique cette conversion, nous avons créé un programme permettant l'automatisation de ce processus. Une fois ce programme utilisé (à travers le Makefile du projet principal), l'image est envoyé à l'écran. Après chronométrage, nous arrivons à un temps moyen de 1,2 secondes pour afficher une image intégralement. Ce temps est bien inférieur aux 17 secondes de la librairie X11, mais est aussi légèrement éloigné des 25 images par secondes promises par le vendeur (Ce taux de rafraîchissement n'étant atteignable que si nous ne générons des carrés de couleurs). Cependant ce temps de réponse est largement suffisant pour l'utilisation que nous souhaitons en faire.

Nous nous sommes donc penché sur l'utilisation de l'interface tactile de l'écran. C'est une lecture USB par interruption qui nous donne si l'écran est actuellement pressé, et si oui où la pression est effectuée. Donc, pour permettre au programme de comprendre quelle fonction appeler lors d'un appui sur l'écran, nous avons créé un autre format de fichier. Celui ci, aussi issu du BMP, est une liste de pixel en nuance de gris. Donc, chaque pixel est représenté par une valeur comprise entre 0 et 255. En fonction de la valeur, le programme principal appelle la fonction en conséquence. De façon analogue à la conversion de BMP et RGB565, nous avons créé un programme permettant de convertir les BMP en suite de bytes correspondant aux pixels.

Conclusion

Au terme des créneaux alloués sur le semestre pour le projet, nous n'avons pas réussi à atteindre l'intégralité des objectifs. Cependant, on peut considérer que le projet est un succès car nous avons réussi à produire un café, réguler les différents éléments de la machine à café (pompe, chauffe-eau et debitmètre). Nous sommes aussi parvenu à réaliser une interface homme machine tactile permettant la sélection de la quantité de café à produire. Bien que cela ne soit pas détaillé dans notre rapport (puisque cette partie ne concerne pas l'IMA), nous avons réussi à créer des pièces en PLA, prouvant la faisabilité d'une machine affranchie des capsules propriétaire Nespresso.

Ce projet à était une occasion pour mixer les connaissances des deux sections de l'IMA: celle des Systèmes Autonomes et celles des Systèmes Communiquants puisque que notre projet comporte une partie importante de régulation codé en C sur une RaspberryPi embarquée.