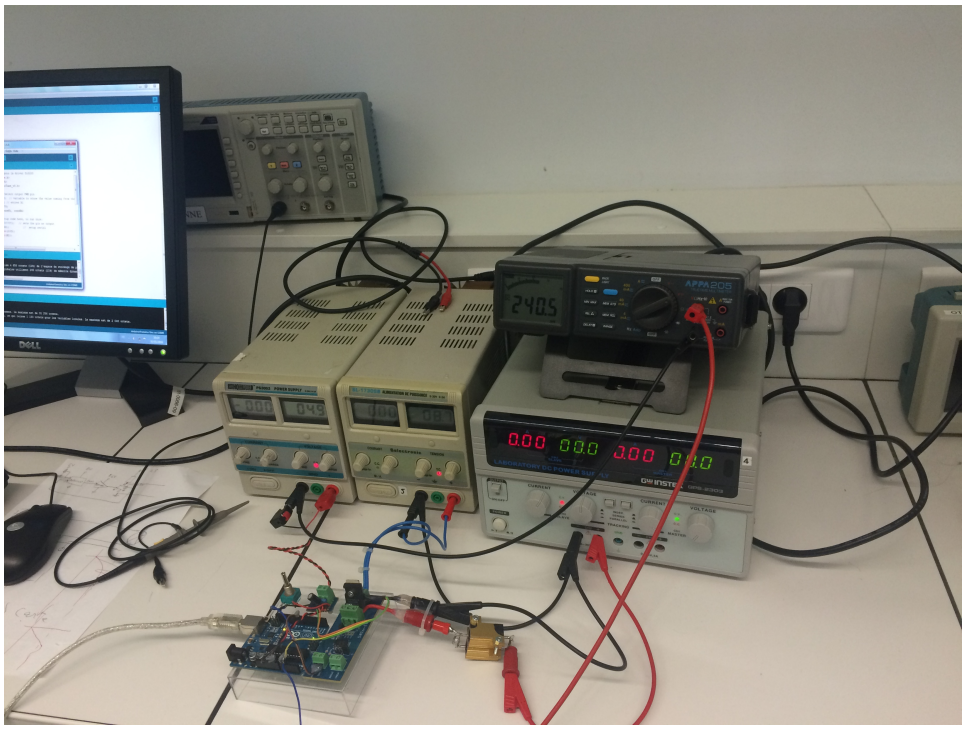
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Projet de Physique P6
STPI/P6/2016 - 009

Contrôle de l'auto-échauffement d'un transistor avec un circuit électronique et un système Arduino



Etudiants :
Léonore Bécuwe, Andréa Blin,
Corentin Lecomte, Carlos Miranda,
Kirill Polesskiy

Enseignant-responsable du projet :
Andrés Echeverri

Cette page est laissée intentionnellement vierge.

Date de remise du rapport : 13 juin 2016

Référence du rapport : STPI/P6/2016 - 009

Intitulé du projet : Contrôle de l'auto-échauffement d'un transistor avec un circuit électronique et un système Arduino

Type de projet : Expérimental

Objectifs du projet :

- Contrôler de manière automatique l'auto-échauffement d'un transistor,
- Étudier l'échauffement et l'auto-échauffement du transistor,
- Coder le PWM et le PID sur la plaque Arduino,
- Optimiser l'efficacité de l'asservissement par PID

Mots-clefs du projet : Électronique, Transistor, Auto-échauffement, PID

institut national des sciences appliquées de rouen
Département Sciences et Techniques Pour l'Ingénieur
685 Avenue de l'Université BP 08- 76801
Saint-Etienne-du-Rouvray
Tél : 33 2 32 95 66 21 - Fax : 33 2 32 95 66 31

Table des matières

I Introduction

Dans le cadre de notre projet de P6, notre encadrant Monsieur Andrés Echeverri nous a demandé de réaliser un système qui permet de contrôler la température d'un transistor. En effet, son doctorat vise à étudier les modes de dégradation d'une nouvelle génération de transistors de commutation de type HEMT (High Electron Mobility Transistor).

Pour ce faire, il est nécessaire de chauffer le transistor afin d'accélérer son vieillissement et ainsi analyser son comportement. La température est un des facteurs principaux causant la dégradation des composants électroniques. A l'état passant, dans la région linéaire, le transistor se comporte comme une résistance. Le but est le suivant : contrôler l'auto-échauffement d'un transistor à l'aide d'un circuit électronique et d'un système ARDUINO.

Le phénomène d'auto-echauffement est caractérisé par le fait que la production de chaleur est plus rapide que sa dissipation. Il en résulte alors une montée de température du système, une forte élévation de la température peut endommager le composant électronique, voire le détruire. Ce contrôle de l'auto-échauffement va donc nous permettre d'atteindre une température souhaitée et cela de la manière le plus efficace possible. L'élévation de la température se fait par la dissipation de puissance dans le transistor, du fait qu'il y a un courant qui passe et une différence de potentiel entre ses broches. Afin de pouvoir contrôler ce phénomène, nous allons dans un premier temps développer plus en détail le circuit électrique utilisé ainsi que les différents procédés intervenant dans ce contrôle. Par la suite, nous analyserons donc les résultats de nos expériences liées à l'auto-échauffement. Enfin, nous tenterons d'optimiser ce procédé.

II Organisation et évolution du projet

1 Répartition des tâches

Au cours des deux première semaines, nous avons effectué différentes recherches dans le but de s'imprégner du projet. Ces séances nous ont permis de comprendre les notions abordées dans notre étude, ainsi que de connaître les divers éléments utilisés dans nos expériences. Par la suite, nous avons pu dresser la liste des objectifs attendus, pour ensuite pouvoir se répartir équitablement le travail.

Électronique	**Programmation**	**Expérimentale**
Kirill Polesskiy	Carlos Miranda	Carlos Miranda
Corentin Lecomte	Corentin Lecomte	Corentin Lecomte
Andréa Blin	Andréa Blin	Andréa Blin
Léonore Bécuwe	Léonore Bécuwe	Léonore Bécuwe
		Kirill Polesskiy

La partie électronique concerne la documentation des composants, et également les techniques utilisées à savoir, l'asservissement PID, PWM etc. La programmation comprend le code réalisé à partir des différentes techniques. Enfin, le rapport a été réalisé par tous les membres du projet en fonction des connaissances et du travail fournit par chacun d'entre nous.

2 Progression

Les premières séances ont été consacrées à la découverte du projet et de ses notions, dans le but de mieux cerner les objectifs. L'intitulé du projet a été rapidement modifié, il ne s'agit plus de contrôler la température d'une plaque chauffante mais celle d'un transistor. Pour cela, nous disposons d'un circuit électronique et d'un système ARDUINO. Nous avons effectué des recherches sur le matériel utilisé et les techniques employées pour pouvoir, par la suite, réaliser nos premières manipulations. Au départ, la plaque ARDUINO n'était pas codée. Nous avons mis à exécution des tests en faisant varier divers éléments du circuit pour étudier son comportement général.

Dans un premier temps, nous nous sommes intéressés aux éléments de base de la programmation de la carte ARDUINO, puis au signal PWM et aux librairies ARDUINO associées. Après avoir compris ces notions, nous avons implémenté l'asservissement PID (cf. III.4 Conception de l'application ARDUINO). Tout au long du codage, nous avons réalisé des tests pour vérifier si ce que l'on faisait fonctionnait correctement.

Une fois le code achevé, on a pu réaliser des expériences concernant l'échauffement du transistor. Cependant, on s'est trouvé face à quelques problèmes liés au circuit (cf. IV Résultats). Après les avoir resolu, on a enfin fait des expériences concernant le PID et l'auto-échauffement du transistor. Finalement, on a exploité les résultats.

III Théorie

1 Le circuit électronique utilisé

i Description du montage

Cette partie a pour but d'introduire les différents aspects théoriques de notre projet, ainsi que le matériel utilisé afin de réaliser nos expériences. Tout d'abord, le circuit électronique comprend :

- un multimètre,
- un générateur de courant continu,
- un oscilloscope,
- deux générateurs de tension,
- un driver,
- un transistor (cf. III.1.iii Le transistor)
- une sonde (pour l'oscilloscope),
- un capteur de température LM35,
- un potentiomètre,
- une carte ARDUINO Uno.

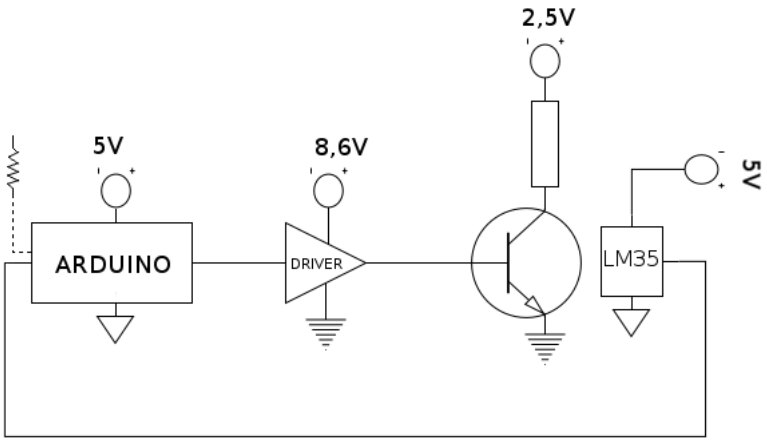


FIGURE 1 – Schéma du circuit

Une photographie légendée du circuit est fournie en Annexe, figure 17.

Comme mis en évidence sur la photographie en annexe, le transistor est relié directement au capteur pour mesurer sa température, ainsi qu'au générateur de courant continu. Ce dernier est lui-même relié au multimètre. Le capteur quand à lui est d'une part vissé sur l'ARDUINO et de l'autre part relié à un premier générateur de tension. Le driver est connecté à un deuxième générateur de tension et à la plaque ARDUINO. Le potentiomètre est relié à l'entrée

analogique A0 de la plaque ARDUINO. La sonde est branchée au générateur de courant continu et à l'oscilloscope. Elle permet la visualisation sur l'oscilloscope du signal électrique dans différents points du ciruits.

ii Renseignements sur l'ARDUINO UNO

Tout d'abord, il est important de souligner que les cartes ARDUINO sont des systèmes libres, c'est-à-dire, que tout usager possède le droit de modification, de distribution ou encore d'utilisation. Ce système est composé de :

- une carte,
- un microcontrôleur de type AVR Atmel
- 14 broches numériques, 6 d'entre elles peuvent être utilisées comme sorties PWM (cf. III.2 PWM)
- 6 entrées analogiques
- un port USB permettant de charger un programme sur la carte
- un bouton de réinitialisation.

Un schéma légendé est fourni en Annexe, figure 18

L'ARDUINO UNO possède une mémoire flash de 32 KB et peut résister à des tensions en entrées comprises entre 6V et 20V, au-delà, le risque d'abimer le matériel s'accentue.

Le langage de programmation utilisé est le C++ . De plus, de nombreuses bibliothèques sont disponibles sur le site internet officiel de l'ARDUINO, ainsi que de nombreux exemples de codes libres de droits d'auteur, facilitant la compréhension. Le système ARDUINO nous a permis de réaliser notre projet à faible coup et sans grande difficulté de part sa carte facile d'accès et très adaptée pour notre situation.

iii Le transistor

Le transistor est un composant électronique semi-conducteur, c'est-à-dire qu'il peut, ou non, véhiculer de l'électricité. Il est composé de 3 électrodes actives et permet de contrôler un courant ou une tension sur une des électrodes de sortie grâce à l'électrode d'entrée. Dans notre cas, le transistor se comporte comme une résistance. Son auto-échauffement se fait par effet Joule. Il est à noter que la résistance interne du transistor varie en fonction de la puissance qui la traverse mais aussi de la température. La fiche technique du TPH3205WS nous renseigne sur la résistance de ce transistor en fonction de la température. À 25°C la résistance vaut 52 mΩ, alors qu'à 175°C dans les mêmes conditions (tension, intensité..) elle est de 120 mΩ. De plus, on remarque que la résistance

thermique est très différente entre junction-to-case (1°C/W) et junction-to-ambient (40°C/W). Il devient très vite difficile de la calculer, d'où l'intérêt de concevoir un programme qui s'en charge.

Nous nous intéressons au transistor TPH3205WS du fabricant Transphorm. Il s'agit d'un nouveau transistor de commutation. Comme on peut le voir sur le schéma 16 en annexe, cette nouvelle génération (GaN en bleu) est plus avantageuse. Ces transistors sont nettement plus résistants à la température (400°C contre tout juste 200°C pour l'ancienne génération). De plus, ils supportent de plus grandes intensités. Du point de vue de la fréquence, ils présentent la même caractéristique.

2 PWM

Modifier la puissance d'un appareil en fonction de son utilisation est très compliqué. Afin de contrôler celle-ci facilement, on utilise couramment la technique suivante. Le PWM (Pulse Width Modulation, soit modulation à largeur d'impulsion en français) est une technique qui permet d'utiliser la puissance souhaitée d'un appareil. Il s'agit globalement d'allumer et d'éteindre le système très rapidement pour obtenir un certain pourcentage de la puissance maximale. Le but de cette technique est de produire un signal numérique. En effet, à partir de données analogiques, on produit alternativement un signal à tension maximale puis minimale. Il s'agit d'une succession rapide d'états discrets, c'est-à-dire que le système est allumé puis éteint à une fréquence fixe. Le signal est périodique d'impulsions binaires. Il est possible de modifier le temps où le système est allumé, ce qui modifiera directement celui où il est éteint, étant donné que la période (allumé + éteint) est fixée. Par la suite, il est possible de calculer un pourcentage correspondant au temps où l'appareil est allumé sur la période du signal, il est appelé rapport cyclique. Ce dernier varie de 0% (le système est constamment éteint) à 100% (le système est constamment allumé). La durée du système allumé correspond à la largeur d'impulsion. Prenons un exemple avec une carte Aduino (cf page suivante pour la représentation graphique).

La tension maximale est de 5V et celle minimale de 0V. Le deuxième système est utilisé à 25% de sa capacité maximale, tandis que le quatrième est utilisé à 75%. Cette technique permet entre autre d'éviter la surchauffe des systèmes. Par exemple, si le système en question est une DEL, que l'on souhaite utiliser à 30% de sa capacité, alors sans PWM, les 70% de puissance fournis en trop seront dissipés sous forme de chaleur (effet Joules). Tandis qu'avec PWM, les 70% de puissance en trop ne seront tout simplement pas produits. Dans notre cas, le PWM permet d'obtenir la température souhaitée. Elle peut être ainsi

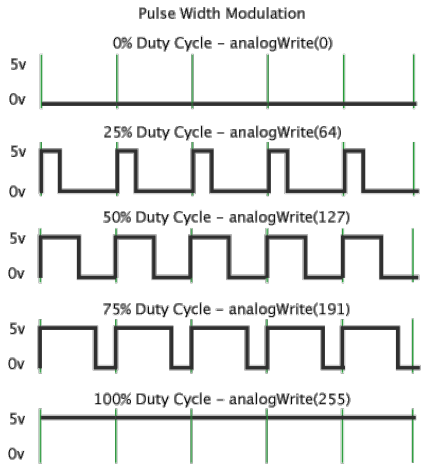


FIGURE 2 – Cycles de travail et ses valeurs associées pour ARDUINO

contrôlée à l'aide du micro-contôleur.

3 Asservissement par PID

La sorti d'un système que l'on commande doit varier pour suivre la consigne demandée. Il faut donc appliquer une commande à l'entrée du procédé :

- A tout instant.
- Périodiquement (en régulation numérique).

Cette commande est calculée par un correcteur.

Qu'est-ce qu'un correcteur ?

À partir de l'écart $\varepsilon = C - M$ (consigne moins mesure), il génère le signal de commande U qui agit sur la variable réglante du procédé. Il est déterminé par le concepteur à partir des critères du cahier des charges qui fixent les performances de l'asservissement. Le correcteur est également caractérisé par une fonction de transfert.

Quel est le but de la correction ?

Le but de la correction est de doter l'asservissement des qualités attendues par le calcul. L'opérateur peut se heurter à deux types de problèmes :

1. Assurer une réponse acceptable pour des signaux de consignes définis au cours du temps.

   Exemple : Cycle de température pour un traitement thermique.

2. Fournir des caractéristiques fréquentielles, comme le gain ou le déphasage, demandées dans une bande de fréquence.

   Exemple : Asservissement du mouvement d'un haut-parleur dans un système haute fidélité.

On définit donc les qualités de l'asservissement en **spécifications temporelles** dans le premier cas, et en **spécifications fréquentielles** dans le second cas.

On peut résumer ses spécifications en trois axes majeurs :
- La précision en régime établi : erreur de vitesse, de position.
- La rapidité : temps de réponse, bande passante.
- L'allure de la réponse et stabilité.

Le correcteur PID

Il permet de régler un grand nombre de grandeurs physiques, c'est le plus utilisé dans l'industrie. Il agit de trois manières :
- L'action Proportionnelle.
- L'action Intégrale.
- L'action Dérivée.

Un système PID agit toujours en boucle fermée. C'est à dire que le système est automatique et que l'utilisateur compare la mesure et la consigne et agit en conséquence.

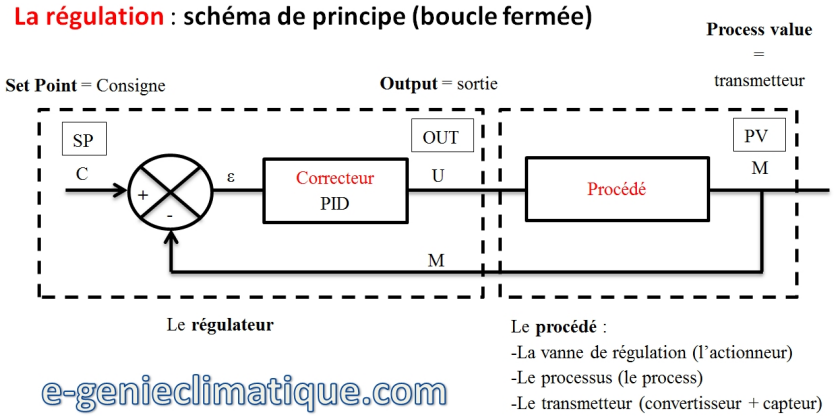


FIGURE 3 – Schéma PID boucle fermée

Fonctionnement théorique d'un asservissement PID de température

On appelle : X la valeur mesurée, Y la commande affichée et W la valeur consigne.

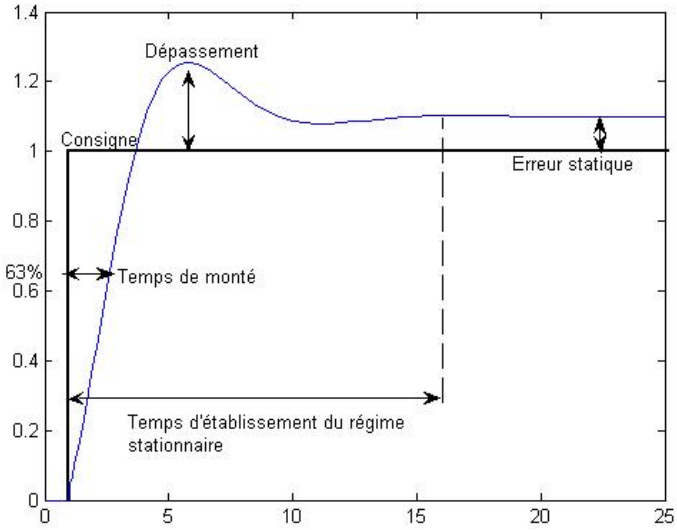


FIGURE 4 – Graphe représentant la réponse type stable d'un procédé par PID.

L'action **P**roportionnelle
La valeur de la commande est proportionnelle à l'erreur. Soit : $Y = K_p(W - X)$ avec K_p le gain proportionnel.

L'action **I**ntégrale
On veut :
- Une action qui évolue dans le temps ;
- Une action qui tend à annuler l'erreur statique.

Cette fonction est remplie par l'opérateur mathématique : 'intégral par rapport au temps'. Ainsi, dans un régulateur, on définie l'action intégrale à partir d'un des deux paramètres T_i ou K_i avec :

$$Y(t) = K_i \int_{t_0}^{t} (W(t) - X(t))dt = \frac{1}{T_i} \int_{t_0}^{t} (W(t) - X(t))dt$$

T_i est le temps intégral, définie en unité de temps. K_i le gain intégral, définie en coup par unité de temps. Lors d'une réponse indicielle, plus T_i est petit plus le système se rapproche de l'instabilité.

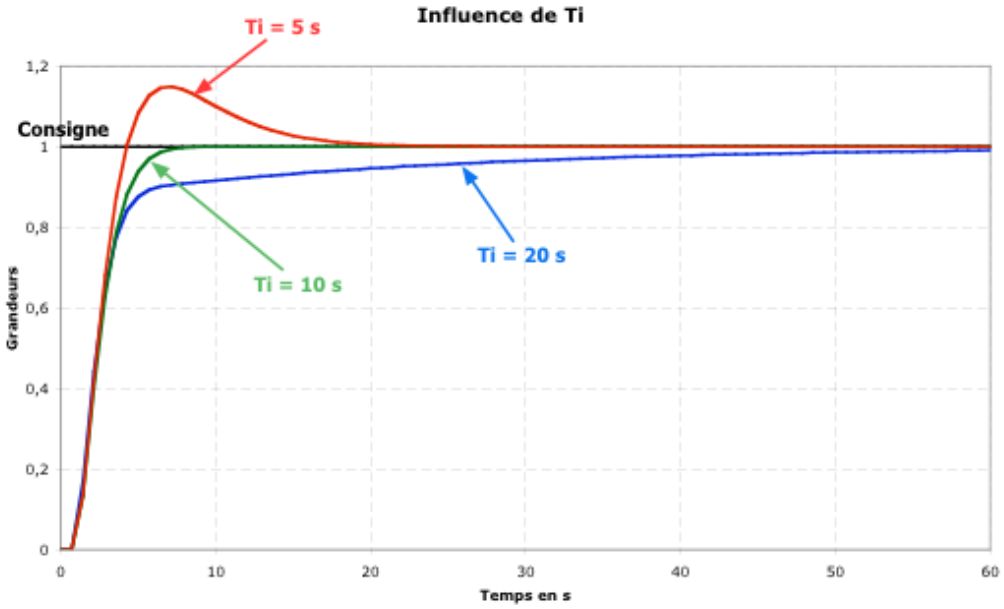


FIGURE 5 – Influence de T_i

L'action **D**érivée
C'est une action qui amplifie les variations brusques de la consigne. Elle a une action opposée à l'action intégrale. Cette fonction est remplie par l'opérateur mathématique : 'dériver par rapport au temps'.

Ainsi, dans un régulateur, on définie l'action dérivé à partir du temps dérivé T_d avec :

$$Y(t) = T_d + \frac{dE(t)}{dt}$$

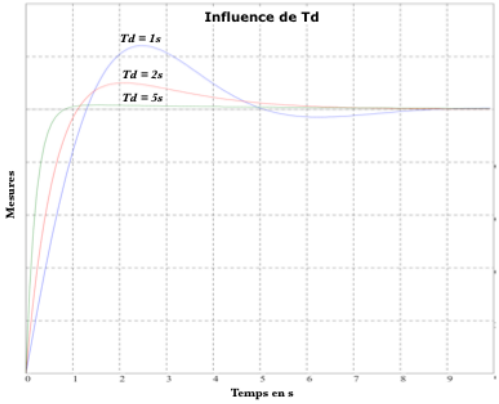


FIGURE 6 – Influence de T_d

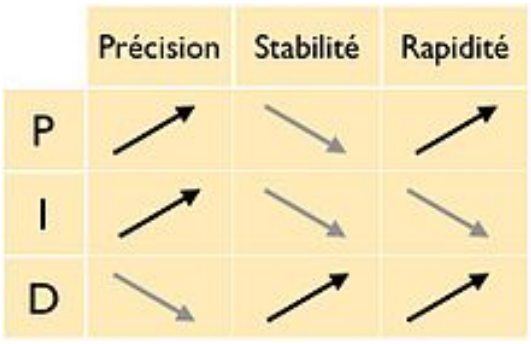


FIGURE 7 – Influence des paramètres par augmentation

Structures des régulateurs PID :

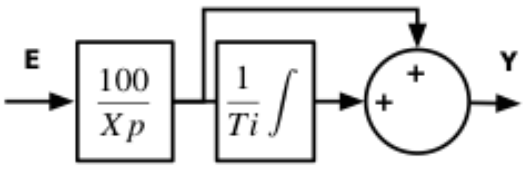


FIGURE 8 – En série

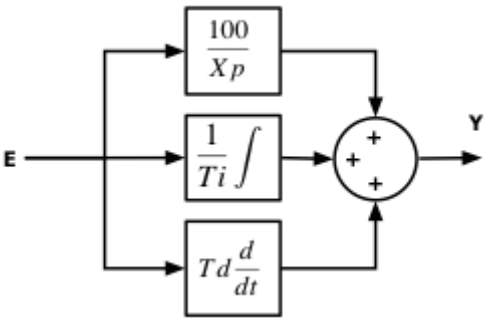


FIGURE 9 – En parallèle

4 Conception de l'application ARDUINO

Le but de l'application est de calculer une valeur de sortie digitale (PWM) à partir d'une entrée analogue (tension aux bornes du capteur de température). Cette valeur de sortie est calculée avec un algorithme PID (librairie PID_v1), utilisant des constantes K_p, K_i et K_d que l'utilisateur définit et transmise comme signal PWM (librairie TimerOne).

Dans une première partie du programme, on déclare les variables globales qu'on utilisera, et on initialise des variables nécessaires au calcul du PID et à la communication entre la carte ARDUINO et le circuit (pin d'entrée). Finalement, on crée des variables de type PID contenant les informations-valeurs du PID.

Dans la partie **setup()**, on affecte des valeurs aux constantes utilisées par le PID. Puis, on associe à la variable PID ces valeurs (**myPID.SetTunings(...)**). Finalement, on affiche l'information concernant le PID.

Dans la partie **loop()** (boucle infinie), on calcule la valeur de sortie (**output**) avec l'instruction **myPID.Compute()** et on envoie un signal PWM associé à cette valeur avec **Timer1.pwm(...)**. La valeur de **output** est dans l'intervalle [0 ;900] et correspond au cycle de travail (limité à 90% d'un cycle pour des raisons de sécurité). Par exemple, si **output** = 512, le courant traversera le transistor pendant la moitié d'un cycle (de période **period**) car $\frac{512}{1024} = 0{,}5$.

Pour le développement de l'application, on a utilisé l'IDE officiel ARDUINO et un simulateur en ligne (123d.circuits.io) pour pouvoir faire des essais sans casser le matériel. Cependant, on n'a pas pu faire des vrais essais sur ce site puisque la mesure de température avec un capteur n'est pas encore implémentée. Ainsi, on a dû limiter le circuit en tension et observer la réaction du circuit avec un potentiomètre avant de passer à l'exécution du code et aux expériences.

IV Résultats

1 Réglage manuel du cycle de travail : potentiomètre

Expérience/Mode opératoire

Nous avons commencé par un montage simplifié pour tester l'influence du cycle de travail sur notre température. Pour cela, nous avons substitué le programme d'auto-échauffement par un potentiomètre à variation manuelle. Ainsi nous avons pu faire varier le cycle de travail permettant d'augmenter ou diminuer l'apport en tension.

Observations

A l'oscilloscope nous avons pu visualiser la courbe du cycle de travail. Lorsque l'on tournait vers la droite le cycle tendait vers 5V continus, et lorsque l'on tournait vers la gauche le cycle tendait vers 0V.

Nous avons constaté l'influence du cycle de travail sur la variation de la température mesurée par le capteur de température sur le transistor. Cependant, nous n'avons pu avoir qu'une mince idée de ce que pouvait permettre un contrôle de la température par PID.

Par conséquent, nous avons remplacé le potentiomètre par un programme informatique qui a permis un auto-échauffement et un contrôle de la température beaucoup plus précis.

2 Auto-réglage du cycle de travail : programme Arduino/PID

Pour l'implémentation et les détails du programme, se référer à la partie informatique.

Une fois le programme conçu, nous l'avons téléversé dans la plaquette Arduino. Nous avons distingué plusieurs expériences car, comme dit précédemment, un contrôle par PID nécessite 3 constantes : Kp, Ki, Kd. Or nous avons rencontré des problèmes quant à l'*autotuning* de ces constantes. C'est à dire que nous n'avons pas réussi à implémenter la fonction permettant de calculer, de manière optimisée et automatique, ces constantes. Ainsi, nous avons utilisé une deuxième méthode : la détermination expérimentale de ces valeurs.

Expérience 1

Mode opératoire

Nous avons choisi de prendre : Kp = 2, Ki = 5 et Kd = 1. Puis nous avons lancé le montage en allumant l'alimentation à 3,3 V et en branchant par port USB le montage à l'ordinateur.

Observations

Nous avons constaté une augmentation de la température jusqu'à la température désirée. Cette augmentation a été assez lente (voir graphe) et la température n'a pas dépassé la consigne. Néanmoins, les mesures faites par le capteur de température ont été extrêmement brouillées. En effet, le capteur de température mesure une tension à ces bornes qu'il converti ensuite. Or, ces tensions étaient très instables. Nous avons graphiquement constaté les imprécisions :

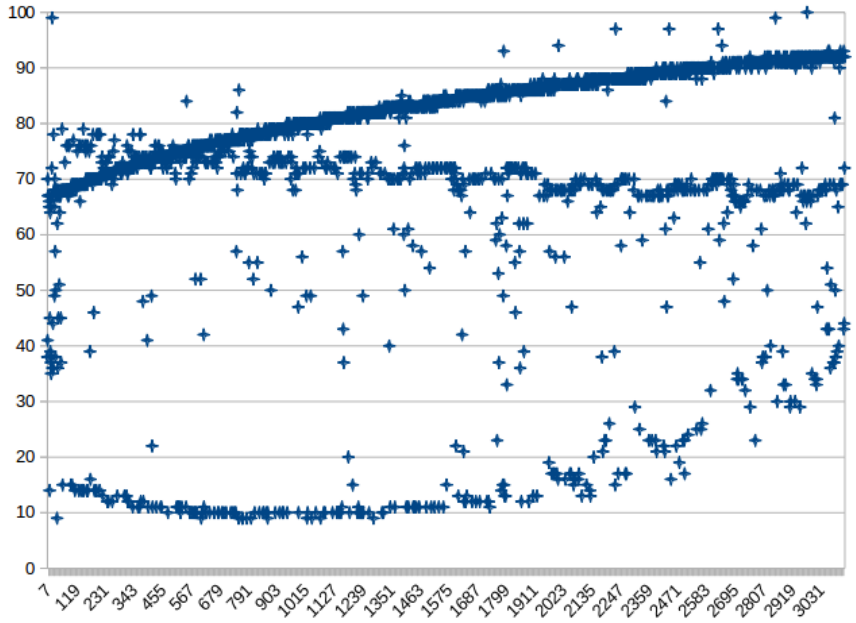


FIGURE 10 – Première visualisation des données.

Par conséquent, il nous fallait remédier à ces imprécisions. Notre enseignant a donc rajouté un condensateur au montage. Il a permis de filtrer la tension arrivant aux bornes du capteur. Sur le graphique ci-contre nous pouvons voir l'effet d'un condensateur sur la tension arrivant aux bornes. Plus le condensateur est fort, plus le signal est « aplati ». Par conséquent, nous avons choisi un condensateur petit pour ne pas fausser nos mesures.

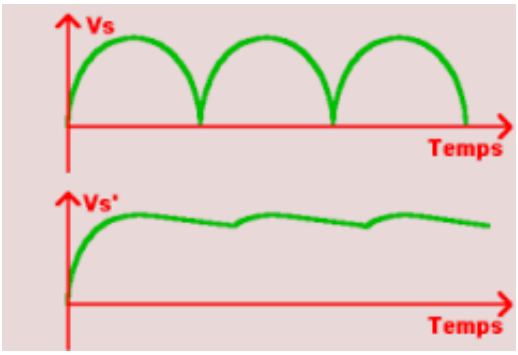


FIGURE 11 – Effet d'un condensateur sur la tension d'un circuit.

Ainsi les mesures ont été plus précises et l'erreur a été réduite. Nous l'avons constaté graphiquement :

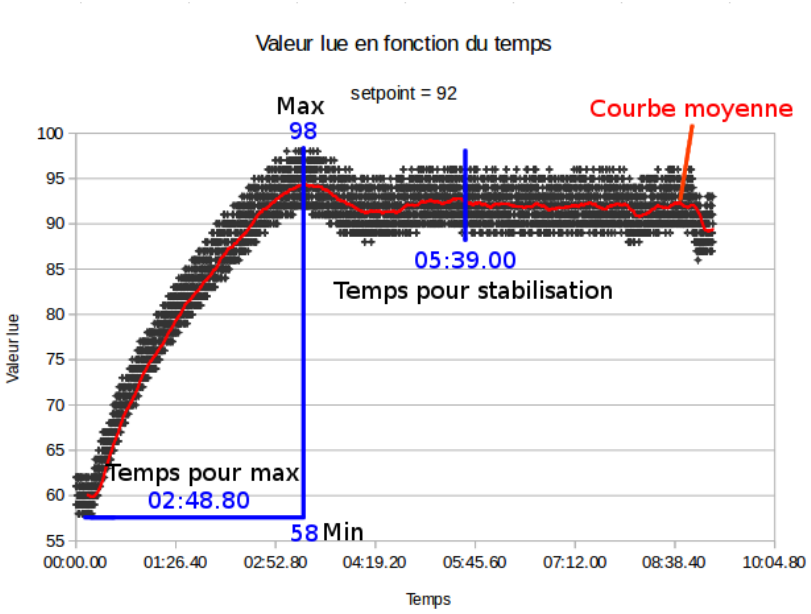


FIGURE 12 – Représentation graphique pour l'expérience 1.

Expérience 2

Mode opératoire

Nous avons choisi de prendre : Kp = 4, Ki = 5 et Kd = 1. Puis nous avons lancé le montage en allumant l'alimentation à 3,3 V et en branchant par port USB le montage à l'ordinateur (qui est la nouvelle source 0-5V).

Observations

Nous avons constaté une augmentation de la température jusqu'à la température « setpoint ». cette augmentation a été beaucoup plus rapide, et la température a dépassé le setpoint.

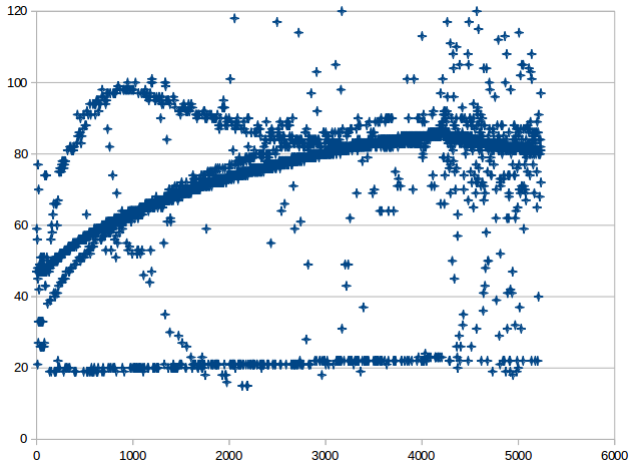


FIGURE 13 – Représentation graphique pour l'expérience 2. Courbe avec bruit.

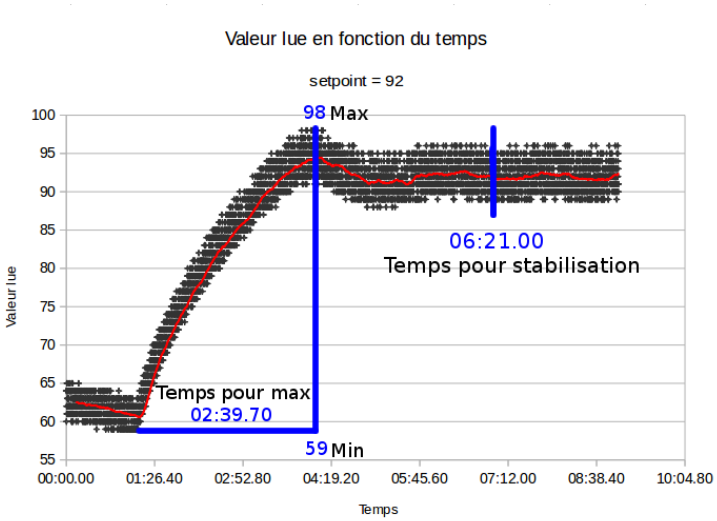


FIGURE 14 – Représentation graphique pour l'expérience 2. Courbe sans bruit.

Conclusion des deux première expériences

Partie régime instationnaire : une augmentation de Kp (du gain) entraîne une atteinte bien plus rapide du setpoint mais précédée d'un dépassement d'autant plus important : plus grande vitesse mais moins bonne précision.
Partie régime permanent : la stabilisation au setpoint est similaire.

Remarque : L'axe des y ne représente pas directement la température. La température désirée est de 45/46 °C ce qui correspond à la valeur 92 sur les graphes (incertitude liée à l'erreur statistique).

Expérience 3

Mode opératoire

Nous avons choisi de prendre : Kp = 2, Ki = 5 et Kd = 4. Puis nous avons lancé le montage en allumant l'alimentation à 3,3 V et en branchant par port USB le montage à l'ordinateur.

Observations

On constate une montée encore plus rapide et un dépassement plus élevé. Mais surtout, si l'on compare au graphe 3, certains grésillements ont été atténués.

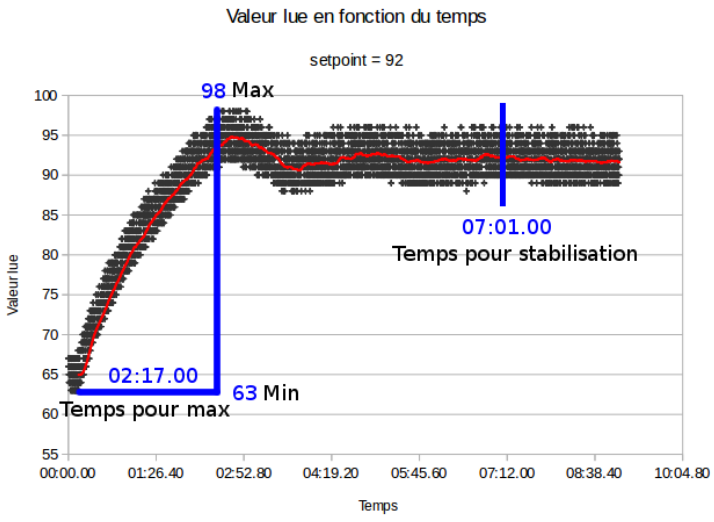


FIGURE 15 – Représentation graphique pour l'expérience 3.

Remarque : Nous n'avons pas eu le temps de terminer cette expérience mais nous estimons que le régime permanent est similaire aux deux expériences précédentes.

Conclusion des expériences

Partie régime stationnaire : une augmentation de Kd entraîne une atteinte bien plus rapide du setpoint mais précédée d'un dépassement d'autant plus important : plus grande vitesse mais moins bonne précision. De plus, il atténue certains pics du signal.

V Conclusion

1 Bilan

Notre objectif était de contrôler l'auto-échauffement d'un transistor. De manière générale, en électronique, les composants ne doivent pas dépasser un seuil de température pour ne pas être dégradés voir détruits. Atteindre une température souhaitée au bout d'un certain temps ne pose pas de problèmes mais d'autres contraintes rentrent en ligne de compte. Le temps d'atteinte, d'abord, et l'établissement du régime stationnaire, ensuite, sont deux exemples de critères qui doivent être optimisés lors d'objectifs industriels.

Que pouvons nous conclure sur les expériences que nous avons mené ?
L'asservissement PID est un moyen efficace pour contrôler l'évolution de grandeurs. En choisissant des paramètres adaptés, le résultat est très intéressant : la température est contrôlée et se stabilise. De plus, nous avions à notre disposition un matériel très succinct pour assurer, a priori, des résultats d'une précision importante. Cependant, la correction PID nous a permis d'obtenir des résultats cohérents et nous pouvons conclure que l'auto échauffement du transistor est contrôlé. Nous pouvons donc imaginer qu'avec des dispositifs industriels, le contrôle est de très haute qualité. Par ailleurs, les difficultés que nous avons rencontré à un moment du projet nous ont forcé à nous adapter et à trouver une autre solution (non prévue au début).

2 Apports

Corentin

Bien que ce projet et les notions associées soient très éloignés de mon choix d'orientation, il aura été très formateur. Premièrement, les connaissances techniques et théoriques sur l'asservissement PID qui m'ont été inculquées me seront utiles à l'avenir car ce type d'asservissement est utilisé dans de nombreuses industries. L'accomplissement d'un projet en équipe est la base d'un travail d'ingénieur. La gestion du temps et de la répartition du travail sont des éléments nécessaires à la réalisation d'un travail.

Carlos

Au début du projet, je me demandais ce qu'il allait m'apporter, car je pensais qu'il s'agissait d'une étude thermodynamique/électrique d'une plaque chauffante. Cependant, ce projet m'a permis de découvrir une nouvelle application de l'informatique, domaine dans lequel je voudrais m'investir dans un futur.

J'ai pu débuter dans l'électronique et me trouver face à des problèmes réels, lors des expériences, et non plus uniquement théoriques.

De plus, ce projet fut l'occasion de mettre en oeuvre nos connaissances sur la gestion de projets.

Andrea

Ce projet m'a tout d'abord permi d'améliorer ma gestion du travail en équipe. Malgré les difficultés rencontrées, il a été relativement rapide de résoudre les problèmes grâce à la bonne cohésion de notre équipe. De plus, ce fut l'occasion pour moi de découvrir et apprendre à travailler à l'aide d'un circuit électronique, ce qui fut très intéressant et formateur. Enfin, la prise de décision, la répartition du travail ainsi que la résolution d'un problème sur une période déterminée, est une première approche du monde de l'entreprise et des différentes responsabilités d'un ingénieur.

Léonore

Ce projet a été une expérience très enrichissante et ce à plusieurs niveaux. Tout d'abord, j'ai acquis des notions en électronique notamment avec la plaque Arduino et le transistor. Ces derniers sont très utilisés dans le domaine de l'industrie. Au delà des connaissances techniques, ce projet m'a enseigné les valeurs du travail d'équipe comme l'organisation, le partage et l'adaptation aux problèmes rencontrés. De plus, atteindre des objectifs, se répartir les tâches et respecter les délais ont été très formateurs. Ce projet est une approche plus concrète du travail d'un ingénieur.

Kirill

Ce projet a été très intéressant et nous servira sur le long terme. En effet, nous avons appris à utiliser nos capacités apprises en cours à un langage et à des problèmes non étudiés. Par exemple, la plaque Arduino et le langage de programmation associé dont nous avons dû apprendre les rudiments afin de faire marcher au mieux notre programme.

Ce projet nous a également poussé à travailler dans des délais et par nous même.

De plus, les connaissances nouvelles en électronique qu'il nous a apporté et le fait de travailler sur un sujet différent des cours sont un plus, car la profession d'ingénieur étant versatile, on ne peut prévoir à quel type de problèmes on aura avoir affaire à.

VI Bibliographie

Arduino

Site officiel d'ARDUINO : https://www.arduino.cc (valide à la date du 30/06/2016)

Article du site officiel d'ARDUINO, *PWM* : https://www.arduino.cc/en/Tutorial/PWM (valide à la date du 25/04/2016)

Librairie PID utilisée : http://playground.arduino.cc/Code/PIDLibrary (valide à la date du 08/06/2016)

Librairie Timer1 utilisée : http://playground.arduino.cc/Code/Timer1 (valide à la date du 08/06/2016)

Transistor

Site officiel de Transphorm : http://www.transphormusa.com/products/#gan-efficiency (valide à la date du 30/06/2016)

Article du site Serial Makers, *Tout sur les transistors NPN et PNP* : http://serialmakers.com/tout-sur-les-transistors-npn-et-pnp/ (valide à la date du 04/06/2016)

Article du site officiel de GaN Systems, *Why Gallium Nitrid ?* : http://www.gansystems.com/why_gallium_nitride_new.php (valide à la date du 28/05/2016)

PID

Article de Wikipédia, *Régulateur PID* : https://fr.wikipedia.org/wiki/R%C3%A9gulateur_PID (valide à la date du 22/05/2016)

Jean-Pierre MAZEL, *Régulation01-Les bases pour comprendre* : http://www.e-genieclimatique.com/regulation01-les-bases-pour-comprendre/ (valide à la date du 13/05/2016)

PWM

Article de Wikipédia, *Modulation de largeur d'impulsion* : https://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion (valide à la date du 25/04/2016)

Article du site Locoduino, *La PWM : qu'est-ce ?* : http://www.locoduino.org/spip.php?article47 (valide à la date du 25/04/2016)

Auto-échauffement

[PDF] **S. EVANNO, J.P. PINEAU, J. CHAINEAUX, R. LODEL, D. CARSON** pour l'INERIS, *Méthode pour l'évaluation et la prévention des risques accidentels, Connaissance des phénomènes d'auto-échauffement des solides combustibles* : http://www.ineris.fr/centredoc/omega11web.pdf (valide à la date du 12/05/2016)

VII Annexes

1 Documentation technique des composants électroniques

Documentation pour le transistor

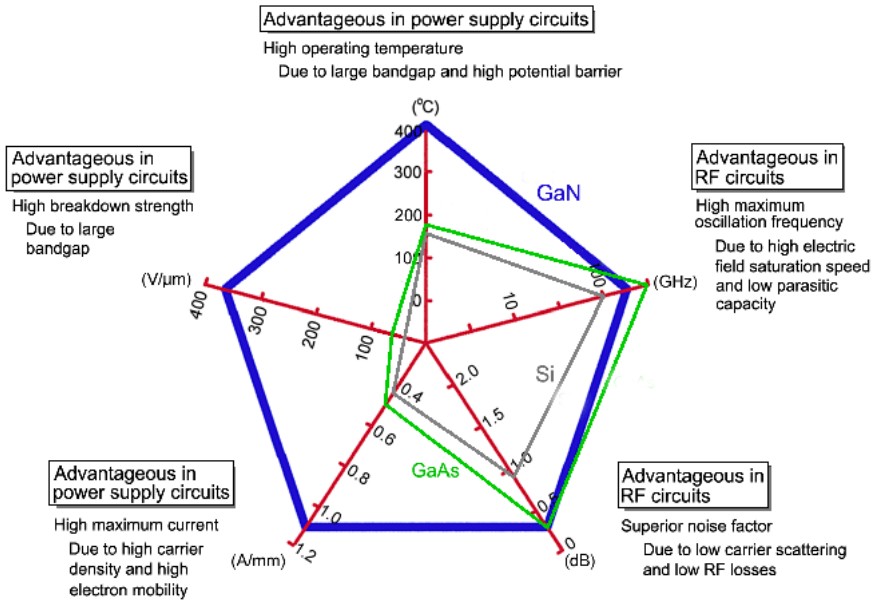


FIGURE 16 – Caractéristiques principales du transistor utilisé

[PDF] Fiche technique et documentation officielle du composant : http://www.transphormusa.com/wp-content/uploads/2016/02/TPH3205WS-v11.pdf

Documentation pour le capteur de température

[PDF] Fiche technique et documentation officielle du composant : http://www.ti.com/lit/ds/symlink/lm35.pdf

Documentation pour la carte électronique

Fiche technique et documentation officielle du composant : https://www.arduino.cc/en/Main/ArduinoBoardUno

2 Schémas

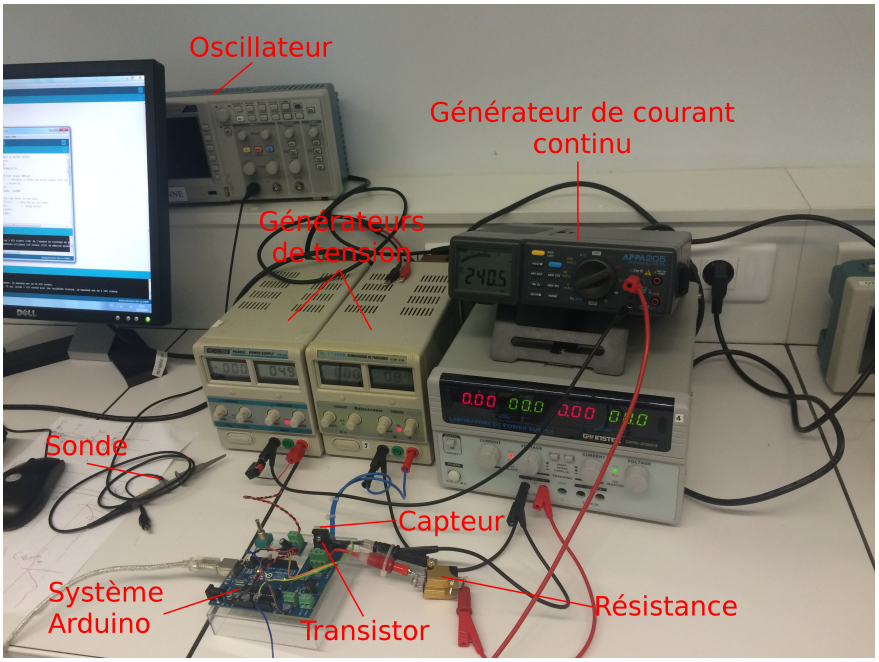


FIGURE 17 – Photo du montage

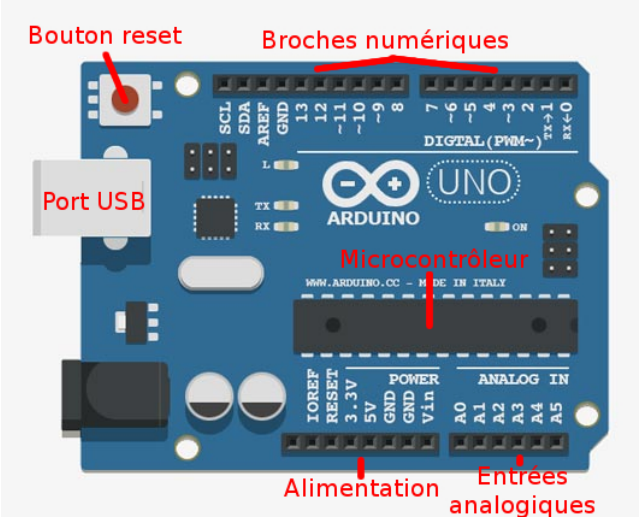


FIGURE 18 – Schéma d'une carte ARDUINO

3 Programme réalisé

```
// Routine PWM pour le driver Si8220
#include <TimerOne.h>
#include <PID_v1.h>

int S_PWM = 9; // Choix de la broche pour le PWM (ne peut etre que 9
    ou 10)
double input, output; // Variables d'entree et sortie
double setpoint = 45; // Temperature desiree en degres Celsius
double consKp, consKi, consKd; // Variables utilisees pour le PID
unsigned long period = 80; // Periode de timer1 en microsecondes

// Declaration et initialisation d'une variable de type PID
PID myPID(&input, &output, &setpoint, consKp, consKi, consKd, DIRECT);

unsigned long now, last; // Variables pour gerer le temps

void setup() {
    // Initialisation de l'ARDUINO
    Serial.begin(9600); // Configuration du debit de donnees (bits/s)

    // Conversion de la temperature de l'intervalle 0-150 a 0-1023 (
        intervalle du le cycle util)
    setpoint = map(setpoint,0,150,0,1023);

    // Configuration du PWM
    pinMode(S_PWM, OUTPUT); // Declare le pin S_PWM comme sortie
    Timer1.initialize(100); //

    // Configuration du PID
    myPID.SetOutputLimits((double) 0,(double) 900); // Restreint la
        valeur de sortie a l'intervalle 0-900, pour le PWM (utilise des
         valeurs entre 0 et 1023) mais limite pour des raisons de
        securite
    myPID.SetMode(AUTOMATIC); // Allume le PID

    // Definition des constantes
    consKp = 2;
    consKi = 5;
    consKd = 1;
```

```
    myPID.SetTunings(consKp, consKi, consKd); // Association des
        valeurs des constantes au PID

    // Affichage de l'information (constantes et setpoint)
    Serial.print("Kp = ");Serial.println(consKp);
    Serial.print("Ki = ");Serial.println(consKi);
    Serial.print("Kd = ");Serial.println(consKd);
    Serial.print("Setpoint: ");Serial.println(setpoint);
    Serial.println("Input \t Output");

    now = millis(); // Debut du chronometre
    last = now;

}

void loop() {
    unsigned long now = millis(); // Suivi du temps

    input = analogRead(1); // Lecture de la valeur d'entree (entree
        analogique A1)
    myPID.Compute(); // Calcul du PID

    TCCR0B |= (1 << CS01) | (0 << CS00); // Configure le bit CS01 pour
         un prescaler de 8
    // 1 us : CUmin = 128 (12.5%) - 2 us : CUmin = 64 (6.256%) - 4 us
        : CUmin = 32 (3.128%) - 80 us : CUmin = 1.6 (0.15%)
    Timer1.pwm(S_PWM,output,period); // Envoi du signal PWM par la
        broche choisie

    // Bout de code pour afficher les valeurs d'entree/sortie tous les
         100 ms
    if(millis()>last) {
        SerialSend();
        last+=100;
    }
}

// Procedure d'affichage
void SerialSend() {
    Serial.print(input);
    Serial.print("\t");
    Serial.println(output);
}
```