



COLLÈGE
DE FRANCE
1530

EWS
C A C H A N

UPMC
SORBONNE UNIVERSITÉS


ENS Cachan - UPMC

Tiphaine KOUADOU
Stage de M1
Année 2015-2016

**DÉVELOPPEMENT D'UN RÉGULATEUR PI NUMÉRIQUE POUR LA
STABILISATION THERMIQUE D'UNE EXPÉRIENCE D'OPTIQUE QUANTIQUE**

REMERCIEMENTS

Table des matières

1 INTRODUCTION	3
2 PRINCIPE DE LA RÉGULATION PID POUR LE CONTRÔLE DE TEMPÉRATURE	3
2.1 Rôle de la branche proportionnelle	4
2.2 Rôle de la branche intégrale	4
2.3 Rôle de la branche dérivée	4
3 DISPOSITIF EXPÉRIMENTAL	4
3.1 Montage de test	5
3.2 Mesure du bruit et conséquences sur la structure du montage	5
3.3 Implémentation du PI	6
3.3.1 Codes Arduino	6
3.3.2 Code Python	6
3.4 Réglage du contrôleur PI	7
3.5 Améliorer la précision de la mesure	7
3.5.1 Changer de capteur de température	7
3.5.2 Modifier la précision de mesure de l'Arduino : analogReference	7
3.5.3 Moyennage et oversampling	7
3.6 Mise en place du vrai dispositif	8
3.6.1 Mesure de température et chauffage	8
3.6.2 Électronique	8
3.6.3 Codes	9
4 RÉGLAGE DU PI ET RÉSULTATS	9
5 MESURE DE LA STABILITÉ DE LA FIBRE	9
5.1 Montage	9
5.2 Résultats	10
6 PERSPECTIVES	10
7 CONCLUSION	10

1 INTRODUCTION

L'équipe Photonique Quantique de l'institut de Physique du Collège de France, au sein de laquelle j'ai effectué ce stage, s'intéresse à la manipulation de photons à l'aide d'atomes froids, dans le but de réaliser des états non-classiques de la lumière ou des opérations logiques sur des bits quantiques portés par les photons. Le projet auquel j'ai participé vise à développer un système de piégeage et d'imagerie ultra-compact pour des atomes froids, basé sur l'utilisation d'une fibre optique multimode à grande ouverture numérique. Dans une telle fibre, la lumière suit un chemin complexe et l'image monochromatique d'un point source apparaît sous la forme d'un speckle. Si ce speckle reste stable dans le temps, des techniques de corrections de front d'onde permettent de reconstruire une image nette de ce point source. Ainsi, pour éviter que le speckle varie à cause des dérives thermiques de la fibre, nous devons contrôler la température de l'enceinte dans laquelle elle est installée. Le régulateur PID est un asservissement permettant de stabiliser une grandeur autour d'une valeur de consigne. Ce type de régulation est à la base du fonctionnement des radiateurs et des climatisations. On le retrouve aussi pour la stabilisation des diodes laser et le même le fonctionnement des sismographes.

2 PRINCIPE DE LA RÉGULATION PID POUR LE CONTRÔLE DE TEMPÉRATURE

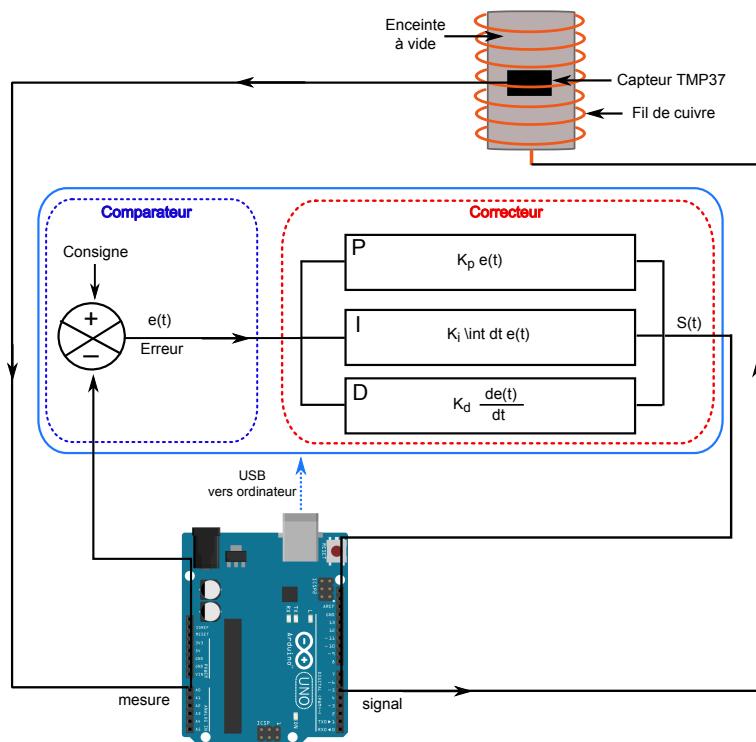


FIGURE 1 – Schéma simplifié du montage et du principe du contrôle PID.

Le PID est principalement constitué de deux blocs : un comparateur et un correcteur (fig. 1). Le comparateur est chargé de comparer la température du système avec une température de consigne et de calculer l'écart $e(t)$ entre ces deux grandeurs. À partir de l'écart $e(t)$, le correcteur calcule le signal $S(t)$ (ou commande), dont l'expression correspond aux équations (1) et (2), qui sera envoyé au système pour amener la température du système à la température de consigne.

$$S(t) = C_p(t) + C_i(t) + C_d(t) \quad (1)$$

$$S(t) = K_p \times e(t) + K_i \times \int e(t') dt' + K_d \times \frac{de(t)}{dt} \quad (2)$$

dans l'équation (1), les $C_j(t)$ correspondent respectivement aux commandes des branches proportionnelle, intégrale et dérivée. Dans l'équation (2), $e(t)$ est l'écart de température. Les constantes K_p , K_i et K_d correspondent respectivement au gain des branches proportionnelle, intégrale et dérivée du

PID. Chacune des branches du PID correspond à un type de régulation différent et l'association des différentes branches du PID (P, PI ou PID) permet de stabiliser un système.

2.1 Rôle de la branche proportionnelle

L'amplitude du signal est directement proportionnelle à l'écart à la consigne. Le gain K_p détermine la force avec laquelle le système répond à un écart de température donné. Avec un gain proportionnel élevé, le système est sensible aux faibles variations de température, mais cela signifie aussi que le contrôleur chauffe beaucoup ce dernier même pour des petits $e(t)$. Au contraire, si on choisit un gain faible, le contrôleur chauffe certes moins fortement le système, mais la régulation est aussi plus lente. Le choix du K_p dépend de la structure que l'on souhaite chauffer et également de la manière dont on la refroidit (ventilation, circulation d'eau ou refroidissement par conduction-convection comme dans ce projet). La conséquence d'une telle régulation et qu'il persiste toujours un écart non nul de température. En effet, si la température atteint la consigne, alors le signal est nul ($C_p = 0$). Dans ce cas le système refroidit, l'écart augmente de nouveau et le système est de nouveau chauffé. Le système finit par se stabiliser à une température en dessous de la consigne. Pour contourner ce problème, une astuce peut être de tricher sur la consigne, en imposant une à une température plus élevée, ou d'ajouter au signal une composante intégrale. Par contre, si le gain K_p est trop élevé, le système devient instable, la température oscille autour de la consigne sans jamais se stabiliser. (figure avec des courbes à différents K_p)

2.2 Rôle de la branche intégrale

La régulation intégrale (ou intégrateur) prend en considération les précédentes valeurs de l'écart de température. Elle permet à la température du système d'atteindre la consigne. En effet, si l'on combine l'action des branches proportionnelle et intégrale (régulation PI), lorsque la température atteint la consigne, la commande $C_p(t)$ est nulle, mais la commande $C_i(t)$ qui a accumulé la valeur des écarts de température ne l'est pas. Le système continue alors de chauffer. Quand on règle ce genre de régulateur, on observe souvent un premier dépassement de température important (jusqu'à 1.5 °C au dessus de la consigne). Cependant, l'écart $e(t)$ devient négatif (mesure > consigne). En conséquence, $C_i(t)$ diminue et le système refroidit. Lorsque la température passe à nouveau en dessous de la consigne ($e(t)$ positif) $C_p(t)$ n'est plus nulle et $C_i(t)$ augmente à nouveau. La conséquence est que le système oscille entre des températures supérieures à la consigne et des températures inférieures à la consigne. Théoriquement, les oscillations finissent par s'amortir et la température se stabilise à la consigne. En pratique, la température continue à osciller avec une amplitude plus ou moins grande suivant le gain K_i choisi. Cette régulation, qui vient compléter la régulation proportionnelle, présente l'avantage d'amener la température à la consigne, par contre, elle ralentit fortement sa durée de stabilisation. (figure avec plusieurs K_i)

2.3 Rôle de la branche dérivée

Le déivateur contient l'information sur l'évolution de l'écart de température avec le temps, c'est-à-dire si la valeur absolue de l'écart diminue (on se rapproche de la consigne) ou augmente (on s'éloigne de la consigne). C'est cette régulation qui permet de maintenir la température strictement à la consigne, tout en limitant le dépassement de température. Il s'agit aussi de la branche la plus difficile à régler. Selon le type de régulation que l'on veut faire, on peut préférer se limiter à un contrôleur PI en choisissant un paramétrage permettant de limiter l'écart de température entre un minimum et un maximum. Dans mon cas, l'enceinte à vide ayant été étuvée à 150 °C, on ne craint pas les dépassements de température inhérents à la régulation PI. De plus, on pense que la stabilité de la fibre sera peu affectée par une variation de ± 0.1 °C de la température. On utilise donc un régulateur PI.

3 DISPOSITIF EXPÉRIMENTAL

On souhaite stabiliser la température d'une enceinte à vide en acier 304, à l'exception d'un bras sur lequel est fixé une cellule en verre, qui est en titane. Pour simplifier le système, on a fait le choix de contrôler la température via un courant traversant une résistance de chauffage (fil de cuivre), le refroidissement se faisant par convection naturelle et par conduction à travers le support métallique de l'enceinte. On a donc choisi de travailler avec une température de consigne plus élevée que la

température ambiante, en visant une précision de 0.1°C autour de la consigne. Cette augmentation de température n'a pas d'effet sur les atomes, qui piégés et refroidis par laser dans un vide poussé, sont isolés du support métallique de l'enceinte. De même, elle n'affecte que faiblement la pression dans l'enceinte, qui passe de 5.10^{-10} à 7.10^{-10} mbar.

3.1 Montage de test

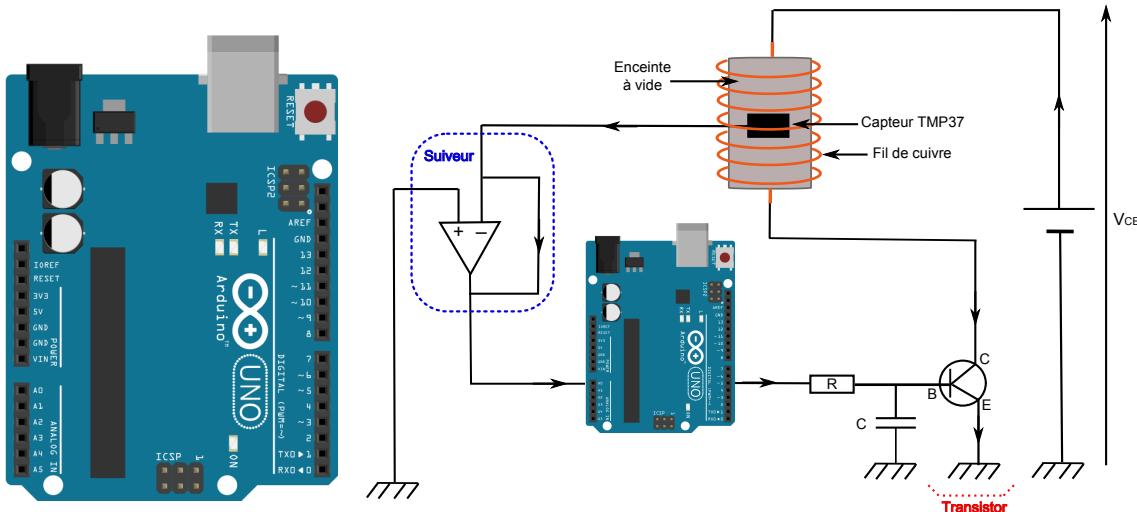


FIGURE 2 – Schéma d'une carte Arduino à gauche et schéma du montage de test à droite.

Pour le régulateur PI, l'interface entre le système de chauffage et la mesure de température est faite par une carte Arduino. Il s'agit d'un circuit imprimé programmable (fig. 2) comportant essentiellement un microcontrôleur et des broches d'entrée et de sortie (des pins) permettant respectivement d'effectuer des mesures et d'envoyer des signaux à un circuit électronique. La carte Arduino fonctionne avec un logiciel et un langage de programmation qui permettent de rédiger des codes contenant les instructions (mesure, émission d'un signal) que l'on veut que la carte Arduino exécute.

Pour effectuer les premiers tests de régulation de température, nous avons construit un premier montage séparé en deux parties, une partie de mesure et une partie de chauffage. Dans la partie de mesure, on utilise un capteur de température TMP37 placé au niveau de l'enceinte et comportant trois pattes, deux servant à l'alimentation 5V du capteur, la troisième correspondant à la mesure de température (donnée en Volts). Le capteur est relié à une des entrées analogiques (ANALOG IN) de la carte Arduino.

La partie chauffage est constituée d'un filtre RC suivi d'un transistor bipolaire NPN et d'un fil chauffant enroulé autour de l'enceinte (fig. 2). La carte Arduino possède également des broches qui peuvent être utilisées pour mesurer un signal digital ou pour envoyer une tension de 0V ou de 5V (DIGITAL). Certaines de ces broches (PWM pour *Pulse Width Modulation* repérées par un "≈") peuvent être utilisées pour envoyer des signaux analogiques. L'Arduino envoie un signal dont la moyenne correspond à la valeur de tension demandée. Il construit ce signal en envoyant alternativement 5V puis 0V pendant des durées identiques ou différentes ("duty-cycle") de telle sorte que la moyenne du signal corresponde à la commande. Par suite, le signal observé à la sortie de l'Arduino est un créneau émis à 62.5 kHz. À cette fréquence, le filtre RC est composé d'une résistance de 560 Ω et d'un condensateur de capacité 1 μF. Ce filtre a pour rôle de moyenner le signal de sortie PWM, mais également de réduire le courant arrivant sur la base du transistor (à quelques mA). On utilise le transistor bipolaire pour amplifier le courant transversant le fil chauffant, car le courant électrique envoyé par l'arduino est insuffisant ($\simeq 40$ mA) pour chauffer un fil de cuivre ($\simeq 5$ m, de résistance linéique $0.97 \Omega/m$). Un transistor bipolaire consiste en deux jonctions P-N montées en inverse (N-P-P-N) que l'on polarise avec un générateur de tension (tension V_{CE} sur la figure 2). Il possède trois pattes qui correspondent à la base (B), au collecteur (C) et à l'émetteur (E). Le courant électrique issu de l'Arduino arrive à base du transistor, le courant amplifié part du collecteur et circule dans un fil de cuivre.

3.2 Mesure du bruit et conséquences sur la structure du montage

Le contrôle de température ne peut fonctionner correctement que s'il n'y a pas de problème au niveau de la mesure de température. Même si ce n'est pas la valeur précise de la température qui nous intéresse

(en effet, si l'on commence l'expérience à 23 ° C affichés, on choisit une consigne au dessus, peu nous importe que la température réelle soit de 22.5 ° C, par exemple). Par contre, mesurer ponctuellement un bruit de très forte amplitude peut poser problème si ce bruit est suffisamment important pour ajouter à la température un offset plus important que l'écart entre la température courante et la consigne. Nous avons ajouté différents composants électroniques au circuit dans le but de réduire au maximum le bruit de mesure. Il y a le condensateur placé au niveau de l'alimentation du capteur de température et l'amplificateur opérationnel monté en *suiveur*.

Le condensateur Le capteur TMP37 est un composant électronique actif, c'est à dire qu'il nécessite d'être alimenté pour fonctionner. À chaque fois qu'il fait une mesure de température, on suppose qu'il consomme quelques ampères pour cela, ce qui peut brûler la mesure. Le condensateur permet de stabiliser l'alimentation du capteur. Il se décharge rapidement lors des chutes de tension et permet de fournir rapidement du courant au capteur, et à l'inverse, il se charge en cas de surtension. On a également placé un condensateur au niveau de l'alimentation de l'amplificateur opérationnel pour la même raison.

Le suiveur Dans un amplificateur opérationnel idéal, les entrées inverseuse et non-inverseuse sont supposées être à un potentiel identique ($V_- = V_+$). Le suiveur est simplement constitué d'un amplificateur opérationnel sur lequel les pattes V_{out} (tension de sortie) et V_- ont été reliées. Cela impose $V_+ = V_{out}$. Dans notre cas V_{out} est connecté à une broche ANALOG de l'Arduino et V_+ est relié à la mesure du capteur de température. Ce bout de montage nous permet d'isoler, en partie, la mesure de température du bruit à 62.5 kHz issu de la broche digitale qui alimente la partie chauffage du circuit.

3.3 Implémentation du PI

3.3.1 Codes Arduino

Pour pouvoir implémenter le régulateur PI dans la carte Arduino, il existe dans la librairie Arduino un code contenant l'algorithme du PID, c'est à dire globalement le calcul du signal $S(t)$. Ce code ne fonctionne pas seul, il faut le paramétrier, c'est à dire notamment lui indiquer la valeur des gains PID choisis et de la température désirée.

Le deuxième code est une interface entre l'algorithme du PI et le circuit électronique. La première partie de ce code (annexe), contient des instructions visant à configurer la fréquence d'acquisition de l'Arduino ainsi que sa fréquence d'émission (FASTADC et setPWMfrequency). On définit les broches sur lesquelles on lit des données, ainsi que les broches sur lesquelles on envoie des commandes. C'est également dans ce code que l'on paramètre le PID (dont le choix des gains proportionnel, intégrale et dérivée) et que l'on envoie les mesures de température au code Python. Ainsi, avec le capteur TMP37, l'Arduino reçoit une mesure en tension de la température. Le convertisseur analogique numérique (CAN) de l'Arduino convertit cette valeur en un entier compris en 0 (0V) et 1023 (5V) (à savoir environ 10 bits). On reconvertis cette valeur pour ne travailler qu'avec des températures dans l'algorithme du PI. Le signal est ensuite converti en un entier compris entre 0 (0V) et 255 (5V) (soit environ 8 bits) avant d'être transmise à la carte Arduino qui se charge de convertir cette valeur en tension avec son convertisseur numérique analogique (CNA).

3.3.2 Code Python

Pour pouvoir évaluer la qualité d'un paramétrage PI et garder une trace des expériences faites, on souhaite pouvoir tracer une courbe de l'évolution de la température en fonction du temps. Comme le logiciel Arduino ne le permet pas, j'ai rédigé un code Python (on dispose d'un code Python... ?) contenant les lignes de code nécessaires pour récupérer les mesures de température faites par l'Arduino, et charger des données précédemment enregistrées. On trace une première courbe pour observer l'évolution de la température de l'enceinte en fonction des gains K_p et K_i choisis. Elle nous permet d'évaluer l'efficacité du paramétrage. Pour pouvoir comparer les tests, les données de température et de temps sont enregistrées dans un fichier. Toutes les heures, on enregistre la température moyenne de l'enceinte dans un autre fichier, ce qui nous permet de tracer une courbe d'évolution de la température sur plusieurs jours.

3.4 Réglage du contrôleur PI

Il existe plusieurs méthodes théoriques de paramétrage qui n'ont pas utilisées pour ce projet pour plusieurs raisons : les paramètres d'un contrôleur PID sont propres à un système donné. Ils dépendent de la nature du système, c'est à dire de la nature de la structure à chauffer (taille, composition). Ils dépendent également du type de régulation que l'on veut faire (vitesse de stabilisation, niveau de précision souhaité). La détermination des gains optimaux peut être plus ou moins rapide suivant les caractéristiques du système.

Le contrôleur PI a été essentiellement réglé en gardant en vue la précision que l'on voulait obtenir. Nous avons commencé par augmenter le gain K_p pour réduire au maximum l'écart entre la température et la consigne et on évitant d'atteindre un régime oscillant. Les échelles de temps pour le test d'un paramétrage sont assez grandes, de l'ordre d'une journée. Nous avons ensuite cherché le gain K_i en commençant par la valeur de K_p puis en le réduisant. Nous avons décidé de procéder de cette manière car le terme intégral, correspondant littéralement à la somme des écarts de température, augmente très rapidement par rapport au terme proportionnel, qui lui diminue quand la température se rapproche de la consigne. Cela a donné les courbes de la figure n. En général, on choisit les gains K_p et K_i de telle sorte que la régulation soit peu ou pas dominée par la commande proportionnelle ou l'intégrale. Si la régulation est dominée par la commande proportionnelle, on observe que la température se stabilise en dessous de la consigne, malgré que la commande intégrale ne soit pas nulle (fig m). Il faut alors diminuer la valeur du gain K_p ou augmenter celle du K_i . Inversement si la régulation est dominée par K_i , alors la température oscille autour de la consigne sans se stabiliser. Cela signifie que le gain K_i est trop élevé ou que le gain K_p est trop faible.

3.5 Améliorer la précision de la mesure

Afin de stabiliser la fibre en température, on souhaite réguler la température de l'enceinte avec une précision de $\pm 0.1^\circ\text{C}$ par rapport à la consigne. Pour cela, il faut notamment travailler avec un capteur de température dont la précision de mesure est au moins aussi bonne que cela. Pour les tests du contrôle PI, le TMP37, que l'on utilise, a une précision de 20 mV/ $^\circ\text{C}$, ce qui correspond à une variation de 1 mV pour une variation de température de 0.05°C . Il faut aussi tenir compte de la précision de mesure de l'Arduino qui est d'environ 5 mV par $^\circ\text{C}$ pour une référence de mesure comprise en 0V et 5V, ce qui est déjà moins bon que la précision du capteur. Je vais énoncer dans cette partie, les procédés que nous avons utilisés pour améliorer la précision de la mesure.

3.5.1 Changer de capteur de température

La première chose à laquelle on peut penser est d'utiliser un meilleur capteur de température. Sur le véritable montage de régulation de température, nous avons installé une thermistance plutôt qu'un TMP37. La thermistance est un composant dont la résistance change suivant la température. Celle que l'on utilise a la même précision que le capteur TMP37, mais elle a l'avantage d'être un composant passif, c'est à dire qu'elle ne nécessite pas de stabilisation d'alimentation. On récupère une mesure en tension de la température en construisant un pont diviseur de tension (figure 3).

3.5.2 Modifier la précision de mesure de l'Arduino : analogReference

J'ai déjà dit plus haut que l'Arduino possède un convertisseur analogique numérique (CAN) grâce auquel les tensions mesurées sont converties en un entier compris entre 0 et 1023 (sur une échelle comprise entre 0 et 5V). Cependant, comme l'Arduino est alimenté sous 5V, il peut arriver que la tension d'alimentation varie de quelques dizièmes de degrés. L'échelle de conversion n'est alors plus 0-5V, mais peut être 0-4.8V ou 0-4.9V, on ne sait pas, et cela affecte la conversion d'unité des mesures. Une solution est d'utiliser l'instruction analogReference de l'Arduino pour modifier l'échelle de conversion. Il y a deux échelles supplémentaires possibles, à savoir 0-3.3V et 0-1.1V, chacune correspondant à une plus grande précision de mesure : $\frac{3.3}{1023} \simeq 3\text{mV}/^\circ\text{C}$ et $\frac{1.1}{1023} \simeq 1\text{mV}/^\circ\text{C}$. Nous avons choisi l'échelle 0-1.1V, ce qui nous restreint aussi à mesurer des tensions comprises dans cette échelle.

3.5.3 Moyennage et oversampling

À travers le moyennage, on veut débarasser le signal des fluctuations de nature électronique (bruit), qui sont souvent très rapides devant l'échelle de temps des fluctuations réelles de la température.

La température est moyenneée sur 5000 mesures, soit sur une durée d'environ 170 ms ; ce nombre est principalement déterminé par la capacité de calcul de l'Arduino. Nous avons alors observé que la température fluctuait moins, mais également que la précision de mesure semblait meilleure que celle annoncée pour le capteur ou pour l'Arduino. Il s'agit d'*oversampling*. Très schématiquement, l'*oversampling*, défini dans [?], consiste à moyenner un signal à partir d'un grand échantillon de mesures afin d'augmenter le nombre de bits sur lequel les mesures sont faites, ce qui revient à augmenter artificiellement la précision de mesure.

3.6 Mise en place du vrai dispositif

3.6.1 Mesure de température et chauffage

L'enceinte est chauffée par rayonnement à l'aide de fil de cuivre (de diamètre ?) dans lesquels on fait circuler un courant électrique. Nous avons coupé 40 m de fil de cuivre par morceaux de 8 m que nous avons soudés en série. Nous les avons pliés en deux et torsadés pour annuler les champs magnétiques parasites créés par le courant électrique circulant dans les fils. Nous avons placé la structure obtenue sur l'enceinte, et nous avons enroulé chacun des 5 fils de cuivre sur un bras de l'enceinte, les plus proches de la fibre optique. À chaque extrémité de la structure, nous avons soudé des connecteurs *bananes* pour relier les fils au reste du circuit électronique. Pour éviter les faux contacts, les fils de cuivres sont isolés électriquement de l'enceinte en acier avec de l'adhésif (?). La thermistance est placée au contact de l'enceinte, au plus près de l'endroit où se situe la fibre. L'enceinte a ensuite été recouverte d'isolant thermique.

3.6.2 Électronique

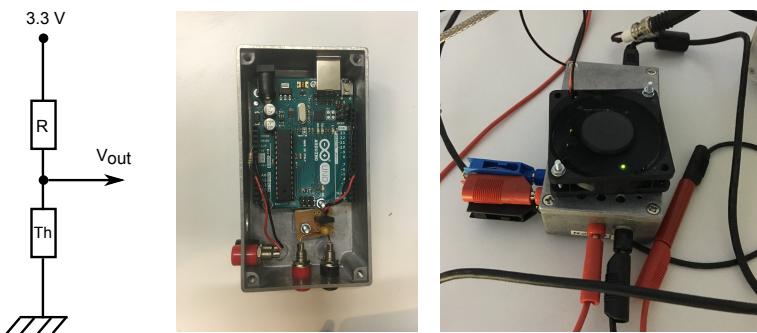


FIGURE 3 – À gauche : Schéma du pont diviseur de tension permettant de mesurer la température de l'enceinte en Volts. $R = 56 \text{ k}\Omega$, Th (Thermistance) = $10 \text{ k}\Omega$ à 25°C . Au centre : Vue intérieure de la boîte contenant une partie du circuit électronique ([à remplacer par une photo du nouveau montage](#)). À droite : Vue extérieure de la boîte.

Le circuit électronique comportant la carte Arduino, le filtre RC et le transistor a été placé dans une boîte métallique servant de cage de Faraday (fig. 3). Certains changements ont été effectués au niveau du circuit de mesure : le condensateur, qui servait à stabiliser l'alimentation du capteur TMP37 et l'amplificateur opérationnel monté en *suiveur* qui servait à séparer la mesure de température du bruit à 62.5 kHz venant du PWM, ont été retirés du circuit. Une résistance de $56 \text{ k}\Omega$ a été placée entre les broches 3.3 V et A0 de l'Arduino ; elle constitue avec la thermistance un pont diviseur de tension permettant de mesurer la température de l'enceinte en Volts avec l'Arduino, plutôt que de mesurer une résistance. La valeur de cette résistance a été choisie de telle sorte que les tensions mesurées soient comprises entre 0 et 1.1 V et que le coefficient de conversion Volts - $^\circ\text{C}$ soit de $20 \text{ mV}/^\circ\text{C}$.

Les autres changements ont été effectués au niveau du transistor. Le transistor dispose d'une plaque métallique lui permettant de dissiper l'excédent de puissance électrique sous forme de chaleur. Dans le montage de test, le courant électrique était limité à 500 mA, la tension de polarisation était $V_{CE} = 5\text{V}$, et la résistance du fil de cuivre était d'environ 5Ω . Cela signifie que le générateur de tension fournissait une puissance $U \times I = 2.5 \text{ W}$ et que $R_{fil} \times I^2 = 1.25 \text{ W}$ était dissipé par le fil de cuivre, ce qui laissait 1.25 W de puissance dissipée par le transistor. Le montage de test était installé à l'air libre, donc la chaleur du transistor était dissipée à l'air libre. Nous nous sommes rapidement rendu compte qu'il fallait trouver un moyen de refroidir le transistor lorsqu'il est installé dans la boîte métallique. Sur l'enceinte à vide, nous avons 40Ω de fil de cuivre dans lesquels on fait circuler au maximum 250

mA, ce qui correspond à une puissance dissipée de 2.5 W. Pour pouvoir réduire le courant maximal circulant de le fil de cuivre, nous avons remplacé la résistance du filtre RC par une résistance de 1.2 k Ω . Pour des raisons de coût, nous avons utilisé un générateur de tension déjà installé sur l'expérience et qui fournit 15 V. Cela correspond à une puissance $U \times I = 3.75$ W et laisse 1.25 W de puissance dissipée par le transistor. Pour faciliter le refroidissement de ce dernier, nous l'avons surmonté de 4 radiateurs, nous avons percé dans la boîte métallique des trous suffisamment petits pour que la boîte remplisse toujours son rôle de cage de Faraday et nous avons fixé un ventilateur au dessus de la boîte.

3.6.3 Codes

Les codes du contrôle PI a été installés sur un Raspberry Pi, qui est un *nano-ordinateur*. Les quelques modifications que j'y ai apportés visaient essentiellement à accélérer le traçage des courbes et à limiter la puissance utilisée par les codes. Un code supplémentaire mesure la température dans le laboratoire.

4 RÉGLAGE DU PI ET RÉSULTATS

Lorsque l'on a installé le véritable montage sur l'enceinte, il a fallu rechercher les paramètres PI. Nous avons procédé de la même manière que sur le montage de test, en partant cependant de la même différence d'ordre de grandeur de gains que pour les derniers test. Sur les courbes obtenues (fig.), nous avons observé d'une part, que la température oscillait avec une amplitude de plus de 0.2 °C (soit plus que la limite que l'on s'était imposée) et avec une période d'environ 3 min (cf. courbeS). Nous avons alors voulu "nous inspirer" (bof...) de la méthode Ziegler-Nichols pour estimer un ordre de grandeur du gain K_i pour le gain K_p que nous avions choisi. Nous avons obtenu la courbe x puis la courbe y. La température varie d'environ ± 0.08 °C par rapport à la consigne, et elle oscille sans se stabiliser. Dans notre cas, nous avons préféré garder un paramétrage légèrement dominé par l'intégrateur, car il nous permettait d'avoir des oscillations de température autour de la consigne suffisamment rapides pour qu'elles se moyennent au niveau de la fibre et perturbent moins la fibre en terme de stabilité que d'éventuelles variations lentes de température.

Effet des dispenseurs Des dispenseurs sont installés dans l'enceinte à vide pour fournir les atomes de rubidium que l'on piège dans la cellule. Un dispenseur est placé près de la cellule, et donc près de la fibre optique aussi, l'autre est placé plus loin. Lorsqu'ils fonctionnent, les dispenseurs chauffent suffisamment (jusqu'à 300 °C) pour chauffer l'enceinte. Le contrôle de température permet de limiter ce chauffage à un maximum de 0.15 °C au dessus de la consigne.

5 MESURE DE LA STABILITÉ DE LA FIBRE

5.1 Montage

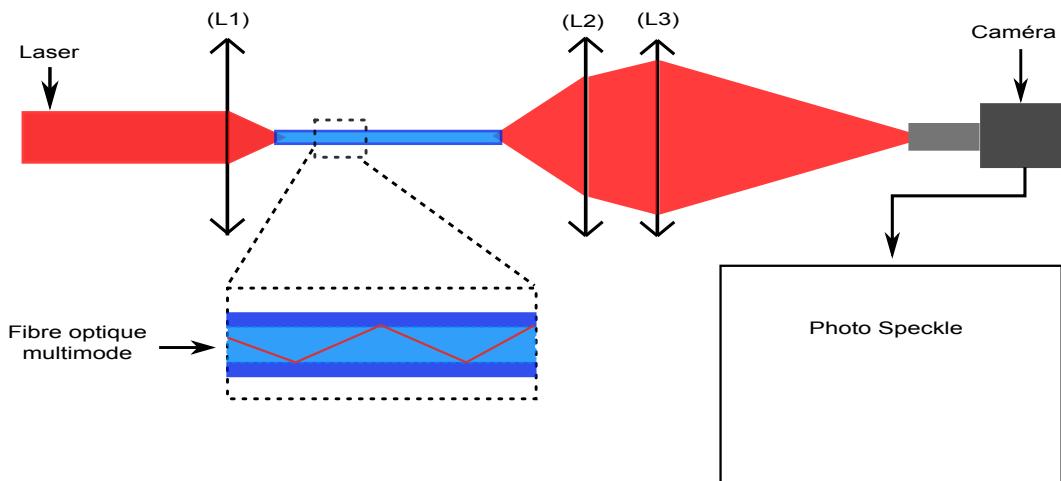


FIGURE 4 – Schéma du montage construit pour mesurer la stabilité de la fibre. $\lambda = 780$ nm, $f_1 = 10$ mm, $f_2 = f_3 = 50$ mm.

Une fibre multimode est un fil constitué généralement de verre permettant de transporter de la lumière quasiment sans pertes. Elle est dite multimode, car plusieurs chemins de propagation y sont possibles par réflexions successives dans la fibre (voir figure 4) et plusieurs longueurs d'onde peuvent s'y propager. On la préfère à la fibre monomode, plus coûteuse, dont le diamètre beaucoup plus petit ($\simeq 9 \mu\text{m}$) ne permet qu'un seul mode de propagation, ce qui rend l'injection de lumière plus difficile, et potentiellement rend l'imagerie moins rapide (?).

Pour vérifier la stabilité de la fibre, le montage que nous avons utilisé consiste en un faisceau laser à 780 nm focalisé sur l'entrée de la fibre multimode au moyen d'une lentille ($f = 10 \text{ mm}$), de deux lentilles de focale 50 mm, placées à la sortie de la fibre et dont le rôle est de focaliser le faisceau très divergeant sortant de la fibre optique, et d'une caméra placée dans le plan focal image du système de lentilles. La caméra est pilotée par le logiciel "Point Grey" et prend des images de la figure de speckle à intervalle de temps régulier. Une fois l'acquisition terminée (idéalement sur plusieurs jours), les images sont traitées avec un code Python rédigé par le post-doctorant de l'équipe, Kilian Müller. Chacune des images enregistrées est comparée à la première image prise et on détermine le taux de corrélation entre ces deux images. La figure (...) représente l'évolution de la corrélation en fonction du temps.

5.2 Résultats

Au regard de la courbe expérimentale obtenue (figure ...), la fibre semble plus stable qu'avant l'installation du régulateur de température, mais il persiste des dérives à l'échelle de deux-trois jours, que l'on aimerait corriger. Il faut remarquer que ces mesures ont été effectuées pendant une période où la température dans le laboratoire variait peu en moyenne sur une heure, comparativement aux conditions dans lesquelles la même expérience a été faite en avril. Des cycles d'allumage et d'éteinte des dispenseurs peuvent aussi être à l'origine des dérives de corrélation observées.

6 PERSPECTIVES

7 CONCLUSION