Asservissement maison

Created samedi 30 juillet 2022

Principe

Le principe est simple : on lit la température à l'endroit où on désire réguler la température, et on ouvre plus ou moins la vanne avec l'actionneur 0-10V pour compenser les fluctuations de température. Au vu des constantes de temps, implémenter cette asservissement avec un programme type PID sur ordinateur/microcontrôleur est tout à fait adapté, et plus flexible qu'un système analogique.
La première version a été implémentée sous forme d'un programme python, lisant la température du datalogger, et contrôlant l'actionneur via un DAC branché en USB.
La deuxième version se veut plus autonome :

- une carte Arduino Due sur lequel un PID a été codé
- des petits modules de lecture de sondes de température PT100 (pour ne pas dépendre du datalogger)
- l'actionneur commandé par le DAC de l'Arduino

Modélisation du système

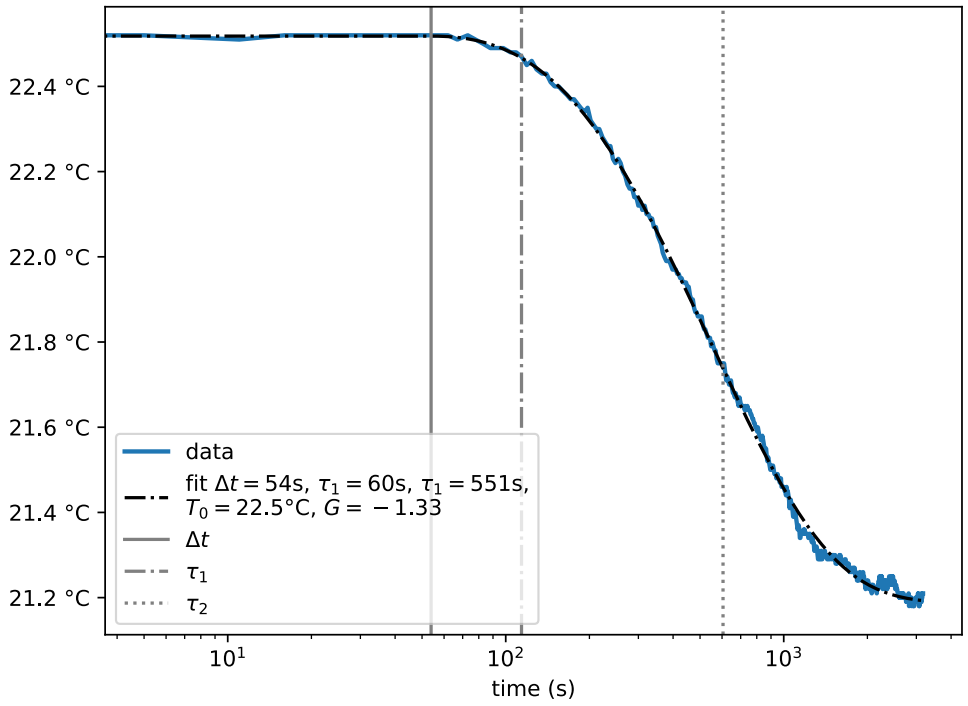
La modélisation mathématique du système à asservir (ie, la température de la pièce) est complexe :

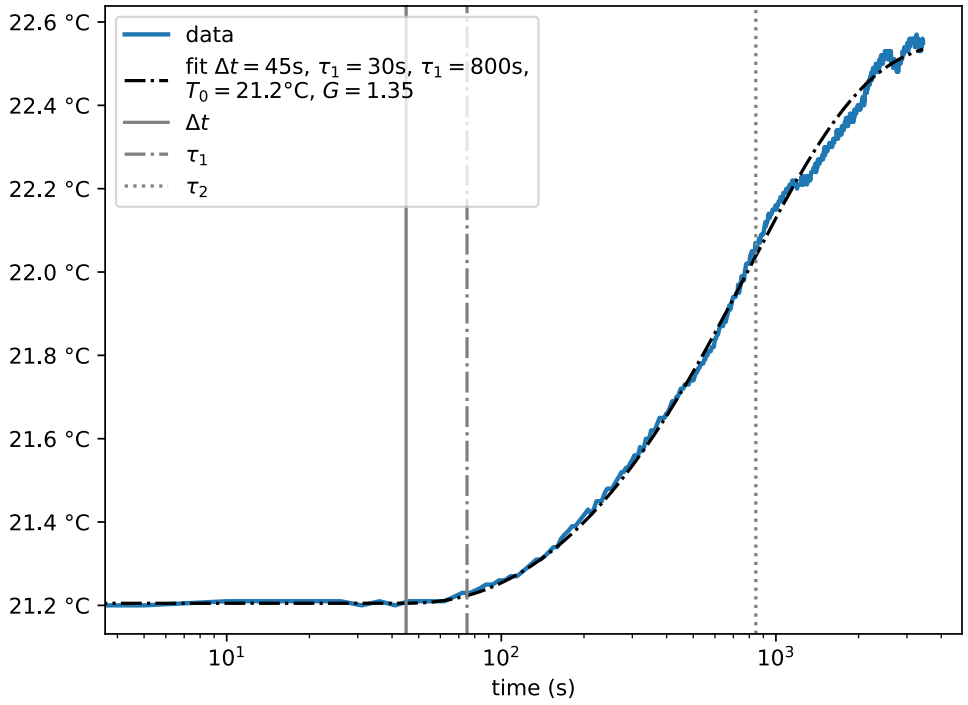
- il y a de nombreux objets dans la pièce qui ont des inerties thermiques différentes
- le dégagement de chaleur est tout sauf constant (présence ou non d'êtres humains, appareils allumés ou non, température des murs...)
- la température n'est pas du tout homogène dans la pièce, la compartimentation et les diverses ventilations n'aidant pas

Néamoins, il y a certaines caractéristiques importantes à garder en tête :

- intrinsèquement non-oscillant en dehors des fluctuations de dégagement de chaleur (ie. pas de résonnance naturelle) -> la fonction de transfert ne peut être qu'une série de réponses d'ordre 1, ce qui reste assez faborable pour l'asservissement à priori
- l'apport de froid est **très non-linéaire** en fonction de la commande 0-10V (vanne presque fermée = gros gain, vanne au dela de 6V = très faible gain) (ici, gain = différence de température en fonction de la différence d'ouverture de vanne)
- c'est un système **à délai** : quoi qu'on fasse, l'air froid met un certain temps à arriver au capteurs de température, et toute variation de commande ne se fera ressentir qu'après un certain temps -> limite ferme sur la vitesse de réjection des perturbations
- lorsque l'on fait un créneau sur la commande, il y a grossièrement **deux constantes de temps** :
    - une très courte (dizaines de secondes en sortie des chausettes d'air), qui correspond :
        - à l'inertie de l'échangeur de chaleur
        - à l'inertie de l'air et de ce qui est directement en contact avec l'air (conduites, ventilations, ...)
        - à la dispertion dûe à la turbulence
        - à l'inertie du capteur de température
    - une très longue (si tant est que ça ait un sens de parler de constante de temps), qui est une sorte de moyenne de toutes les grandes inerties (air total de la pièce, tables, murs, instruments...)
- l'asservissement va typiquement être effectué bien plus rapidement que la/les constantes de temps longues : c'est vraiment la **réponse aux temps courts** (de la dizaine de seconde à la dizaine de minute) qui compte pour la performance de l'asservissement, tant qu'on utilise un intégrateur, on se fiche un peu du gain statique, qui de toute façon varie fortement suivant les conditions

En regardant la température sortant des chausettes d'air, un créneau sur la commande ressemble typiquement à :





(échelle log en temps; mesuré au datalogger avec un capteur de température rapide très près du milieu de la chaussette du milieu)
Pour prendre en compte les idées ci-dessus, on a modélisé par un modèle du second ordre (deux pôles réels) + un terme de délai pur :

$$H(p) = G \cdot \mathrm{e}^{-p\Delta t} \cdot \frac{1}{1+\tau_1 p} \cdot \frac{1}{1+\tau_2 p}$$

Dans les conditions de mesures ci-dessus (manip allumée) et sur les échelles de temps considérées, on a grossièrement :

- $\Delta t = 50$ s
- $G = -1.34$
- $\tau_1 = 40$ s
- $\tau_2 = 700$ s

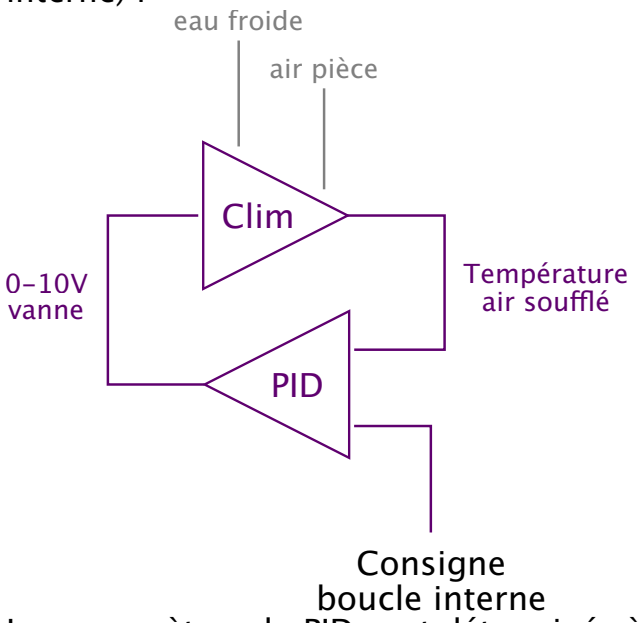
Ce ne sont que des ordres de grandeurs, à prendre avec des grosses pincettes...

Structure de l'asservissement

Comme toujours, mieux vaut agir le plus tôt dans une chaine de conséquences pour éliminer les perturbations le mieux possible, quitte à ajuster l'action pour avoir le résultat voulu en bout de

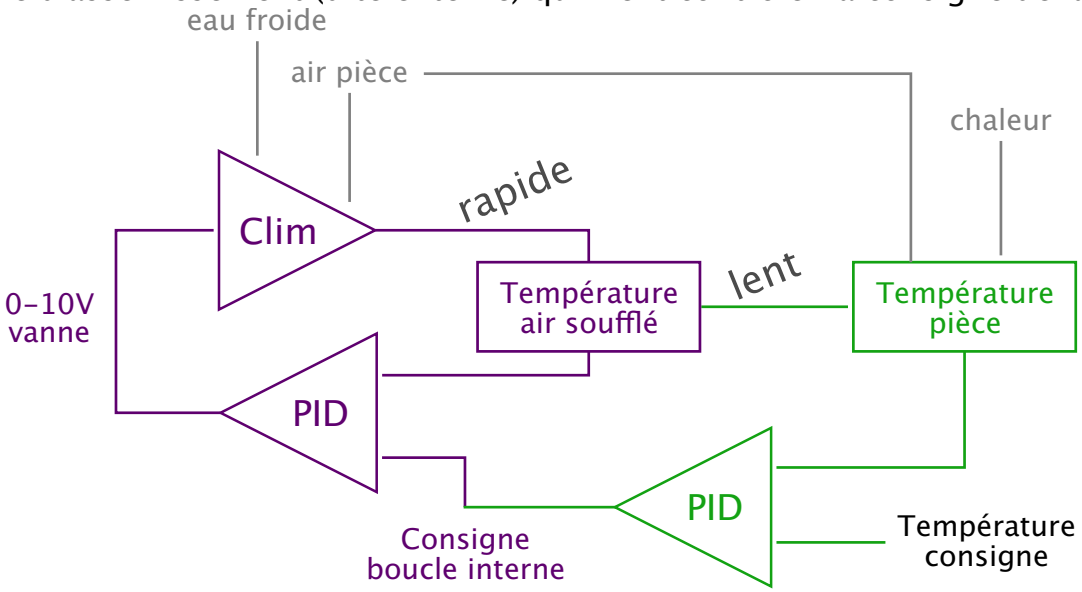
chaine.
C'est particulièrement pertinent dans notre cas, où l'on peut réguler la température sortant des chaussettes d'air bien plus rapidement que celle de la pièce ou des tables.
@Félix a donc choisi de réguler la température de la table de science avec deux boucles d'assevissement imbriquées :

1. On asservi la température de l'air sortant des chaussettes par une première boucle (dite interne) :



   Les paramètres du PID sont déterminés à l'aide des créneaux ci-dessus, de sorte à faire que la température suit la consigne au plus près et au plus vite, sans non plus introduire de résonnance trop marquée.

Dans le fond, on se fiche de la température de l'air soufflé, on veut juste asservir la température de la table de science (ou d'autres endroits sensibles de la manip). On introduit donc une deuxième boucle d'asservissement (dite externe) qui vient contrôler la consigne de la première :



Avantages :

- **réjection rapide** des perturbations sur l'air de la pièce ou l'eau froide
- la boucle interne, a priori plus rapide à régler vu les constantes de temps, peut être optimisée sur une plus large gamme de conditions, et le système résultant [consigne->air soufflé] est **linéaire**, ce qui facilite grandement la conception de la boucle externe
- faire cet asservissement en deux temps semble moins casse tête
- plus flexible

Désavantages :

- plus de paramètres à régler, un peu usine à gaz
- mal réglé et/ou proche de la limite de stabilité, plus susceptible aux oscillations qu'une simple boucle : les oscillations de la boucle interne se répercutent sur la température finale, que la boucle externe cherche à compenser en agissant sur la consigne interne, ce qui excite encore plus les oscillations de la boucle interne, etc... -> il est important que la boucle interne n'ait

pas de résonnance !
- une boucle interne trop lente est pire qu'une unique boucle
- nécessite deux capteurs
- bref, on peut facilement perdre plus de temps avec deux boucles imbriquées qu'avec une unique boucle !

Implémentations

Détails dans le rapport de stage d'Erwan.

Implémentation comme script python sur l'ordi auxiliaire

Datalogger -> InfluxDB -> PID implé en python -> USB->I2C -> DAC 12 bits -> servomoteur 0-10V
Ça a l'avantage d'être très facile à modifier, et d'utiliser le datalogger qui a une très bonne résolution sur la température.
Et de ne pas nécessiter beaucoup de matériel.
Ça a le désavantage de ne pas être "plug and forget", et de nécessiter un ordinateur fiable allumé h24 et correctement configuré.
Pas un problème pour nous, mais si on veut exporter aux autres équipes, c'est potentiellement un casse tête. Et puis tout le monde n'a pas de logger de température surtout...
On a pu obtenir de très bons résultats :

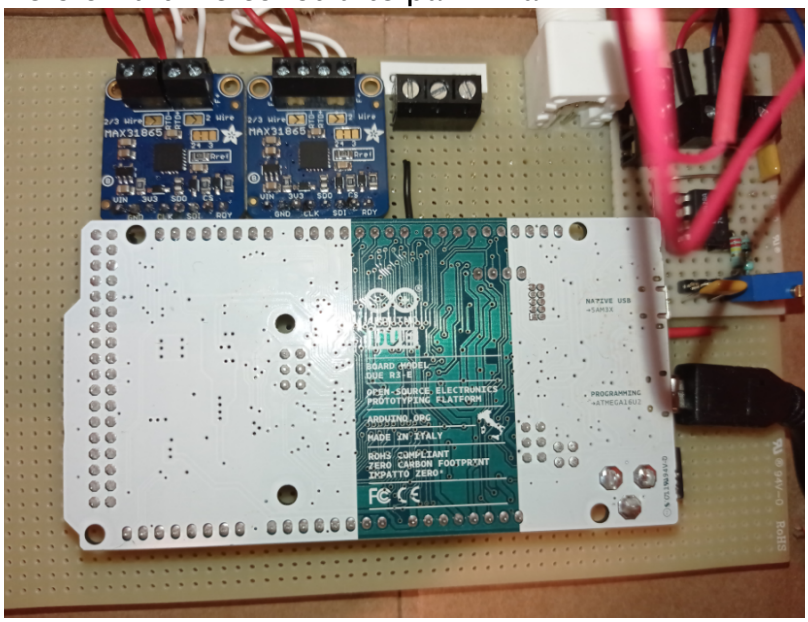
Script avec les fichiers auxiliaires pour le DAC en I2C : ./clim-asser-py.zip

Implémentation sur une Arduino autonome

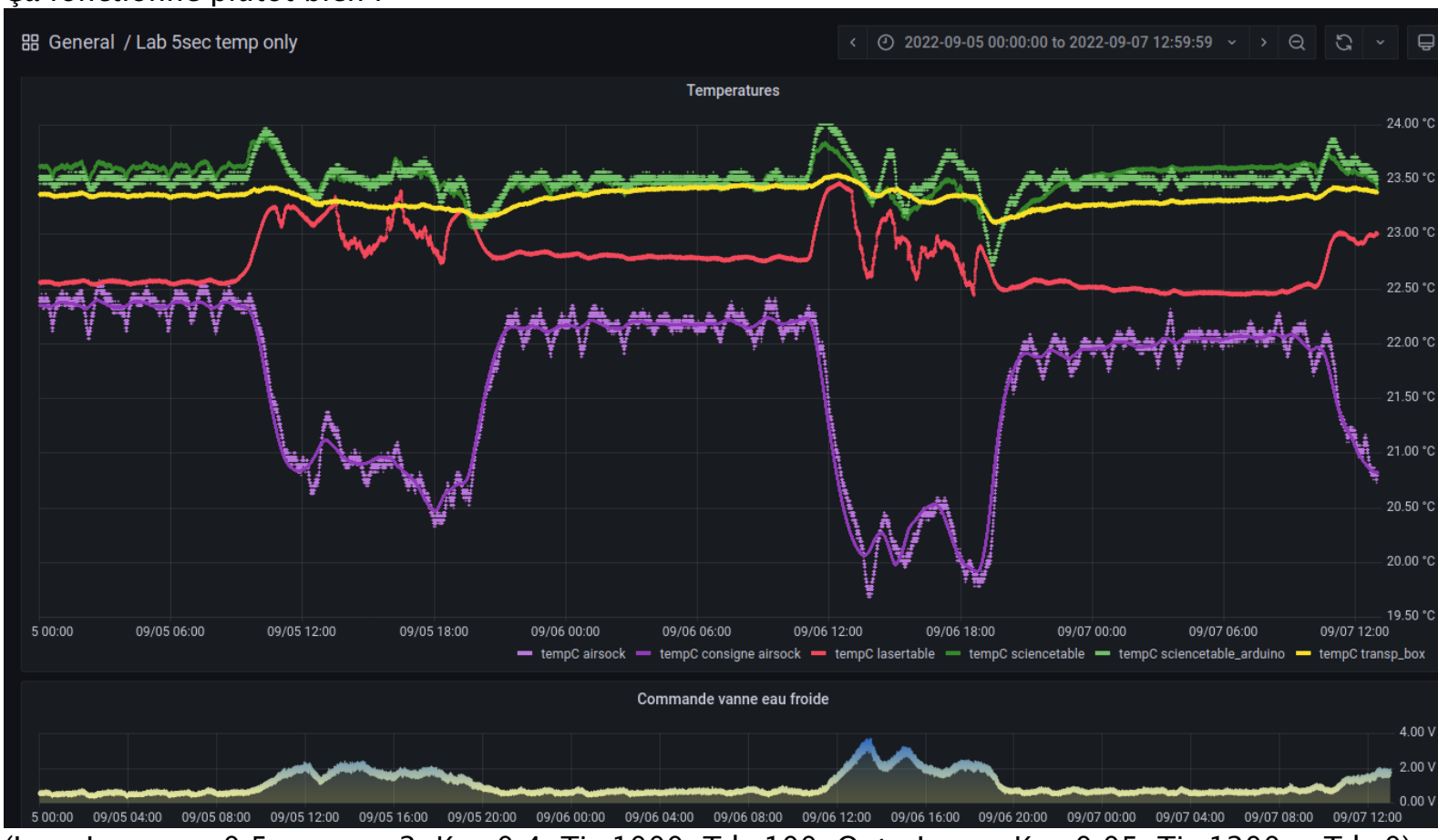
Pour mesurer les deux températures, on utilise des modules Adafruit basés sur le MAX31855, avec sa petite bibliothèque Arduino fournie :
- https://www.adafruit.com/product/269
- https://learn.adafruit.com/adafruit-max31865-rtd-pt100-amplifier/arduino-code

configurés en PT100.
Pour le moment les capteurs de température de l'air soufflé et de la table sont des modèles à réponse rapide.
La résolution du MAX31855 est pas géniale, on voit la quantification de la valeur lue sur la capture d'écran ci-dessous. Une amélioration a été de faire de multiples lectures de la température à chaque pas de temps et de moyenner => la quatification est bien moins visible grace au dithering naturel. Par contre ça nécessite de mettre en place un timer pour que la boucle tourne vraiment à la fréquence voulue. Pour celà, la bibliothèque DueTimer est utilisée (mais on pourrait très bien coder le timer à la main pour ne pas dépendre de la bibliothèque).
Pour la commande 0-10V, on utilise le DAC interne de l'Arduino Due, translatée/amplifiée en 0-10V par un op amp.
Le système fonctionne indépendamment, mais l'ordinateur auxiliaire est connecté à l'Arduino et lit

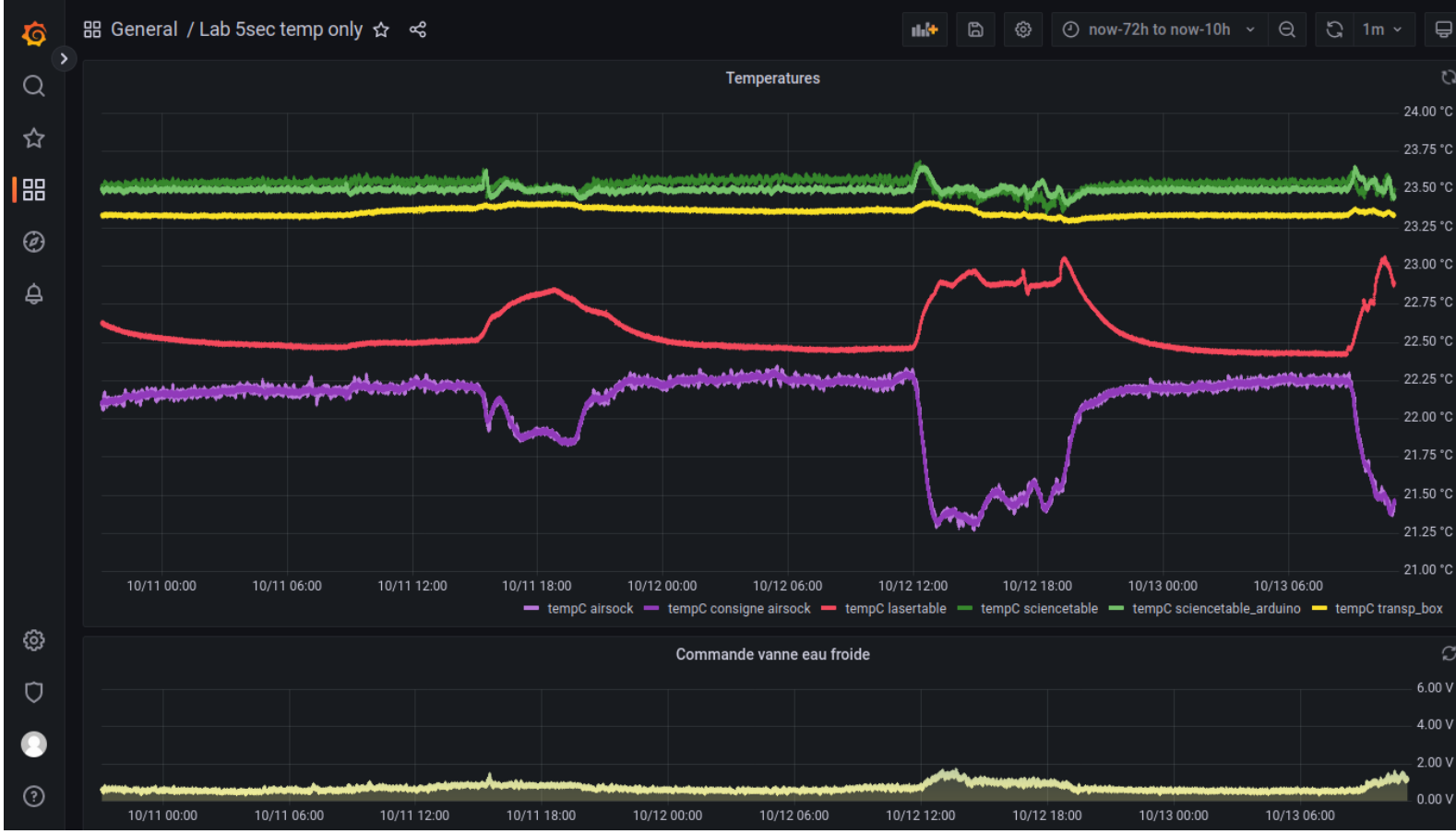
les valeurs de température, de consigne interne et de commande vanne, que l'Arduino écrit sur le port série. Un petit script python lit les valeurs et écrit dans la base InfluxDB.
Version ultime construite par Erwan :



Code au 2022/09/12 : ./clim-asser-arduino.zip
Ça fonctionne plutôt bien :



(InnerLoop : a=0.5; expon=2; Kp=0.4; Ti=1000; Td=100; OuterLoop : Kp=0.05; Ti=1200 s; Td=0)
(courbes vertes = température à garder constante; courbe violette = consigne boucle interne; courbe rose = température boucle interne)
Depuis qu'on a déterminé de meilleurs paramètres pour la boucle interne et que l'on a déplacé le capteur de température de la table de science à un endroit plus raisonnable (au milieu de la table plutôt qu'au bord), on a de meilleurs résultats encore, avec moins d'overshoot et de discordance de température entre les deux capteurs de la table de science, et une température sur le breadboard transport encore plus constante :



Paramètres :

- Boucle interne : Kg = $-1/1.34\ °C^{-1}$, Kp = 3, Ti = 150 s, Td = 100 s
- Fonction commande -> vanne de linéarisation : 0.5V·cmd^2 autour du point de fonctionnement usuel ~0.8V
- Boucle externe : Kp = 0.4, Ti = 400 s, Td = 50 s

Pas loin d'être à 0.1°C en journée !

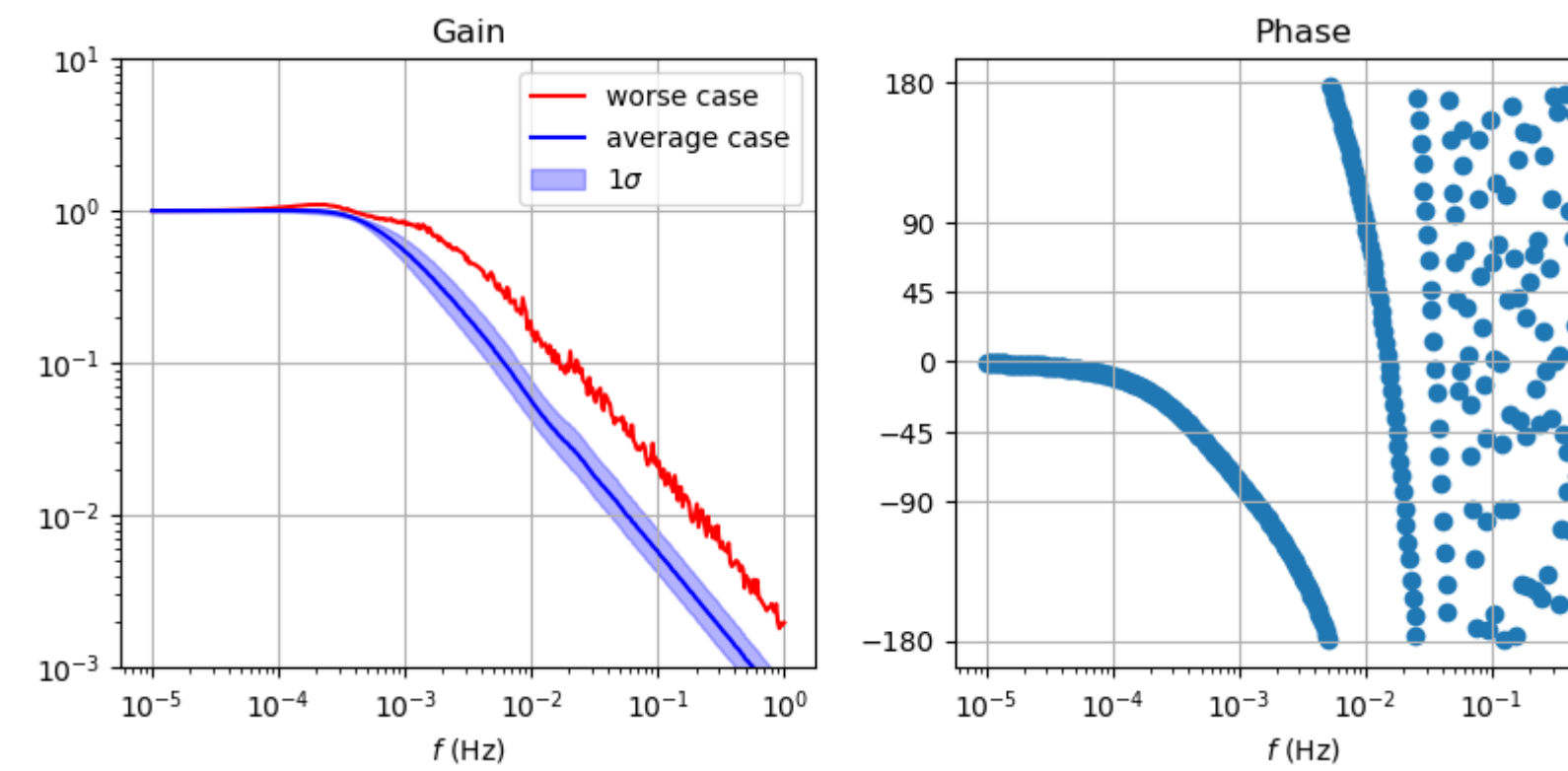
Détermination des paramètres PID de la boucle interne

Notebook pour simuler les réponses d'une boucle avec notre modèle (dont les paramètres sont déterminés par les créneaux, et auquels on rajoute une incertitude/distribution) et un PID : ./ftbf-diagrams.ipynb

Comme toujours, il faut commencer par un intégrateur faible (1000s typ.), et augmenter progressivement P, I et D itérativement, en évitant de créer une résonnance sur la FTBF.

Le modèle semble bien marcher mais il faut absolument être conscient de ses limites. En particulier, plus rien ne fait sens si on pousse les paramètres du PID trop loin car la modélisation ne prend pas en compte toutes les constantes de temps courtes. De plus, le terme de délai pur autorise en théorie de rétroagir rapidement en fine-tunant les paramètres (ce qui est évidemment une mauvaise idée en pratique).

Pour la boucle interne et dans notre cas, on obtient une bonne réponse (ie bonne bande utile et pas de résonnance) tout en restant dans une gamme de paramètres safe :



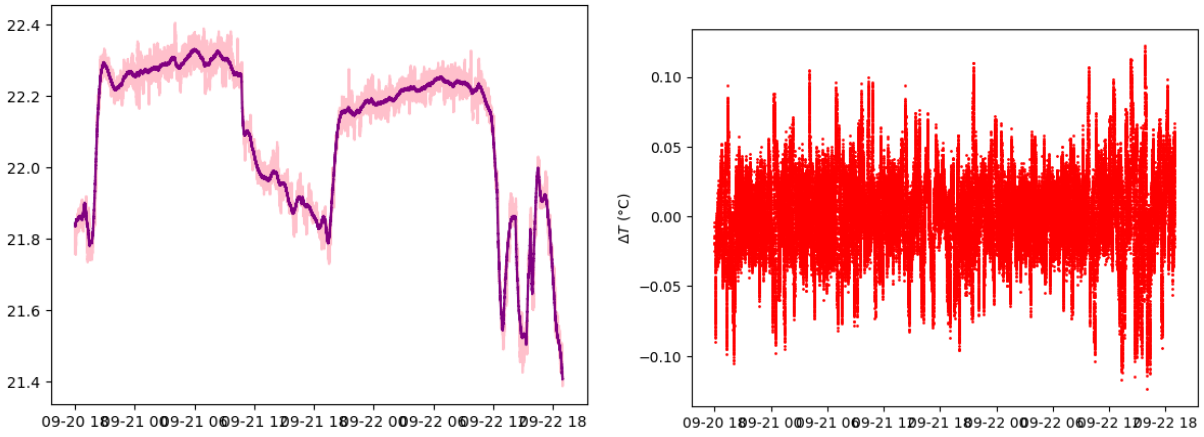
avec

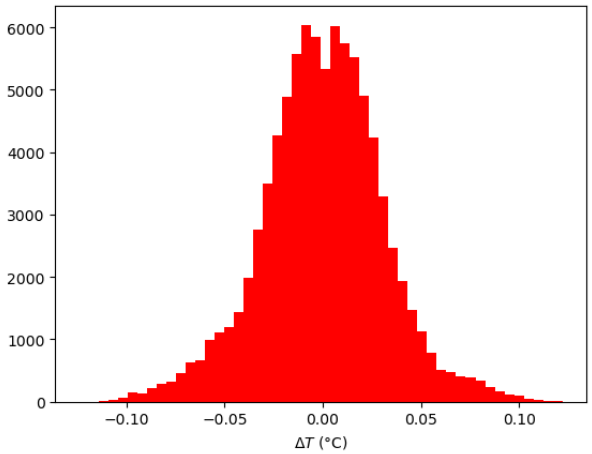
- Kg = 1/1.34
- Kp = 2
- Ti = 300
- Td = 80

donnant de bons résultats en pratique.

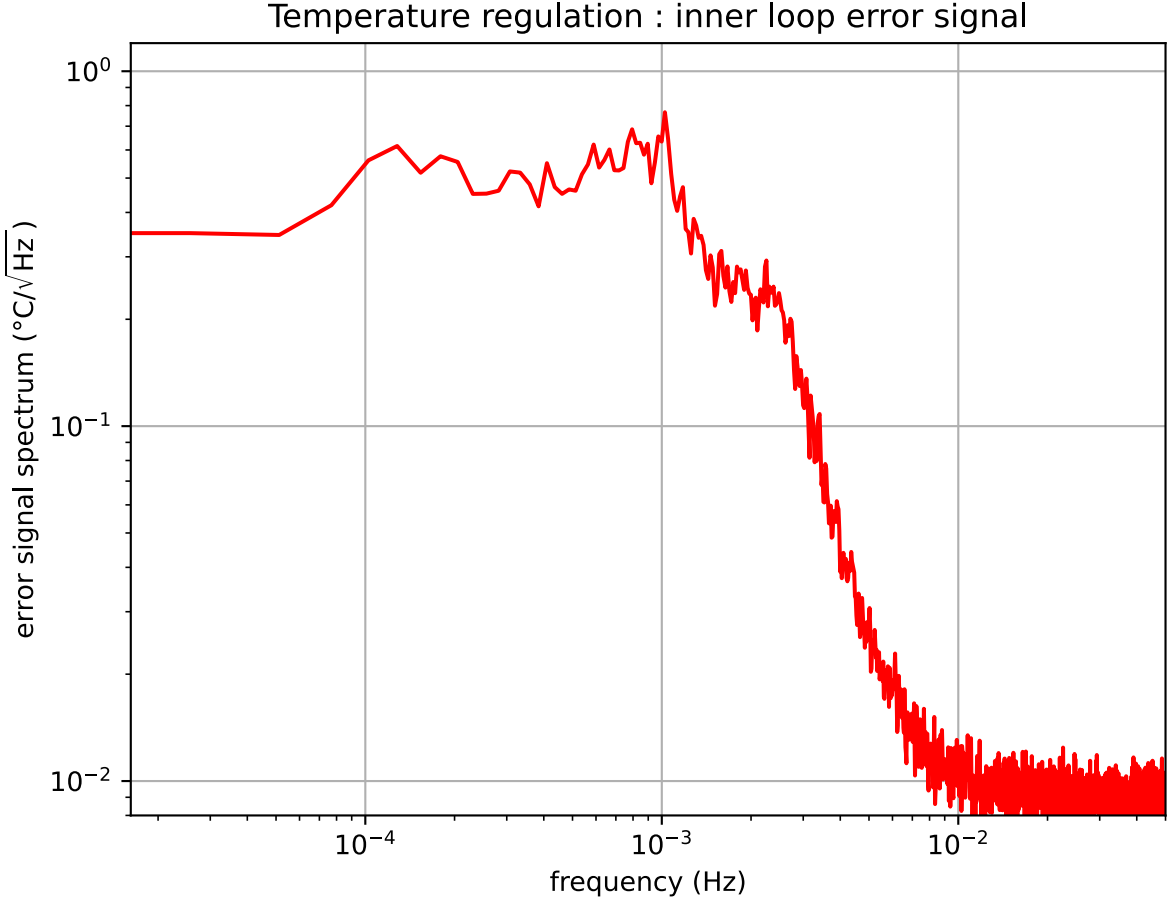
Spectre du signal d'erreur sur la boucle interne

Données de la boucle interne échantillonées à 2sec sur 2 jours :





(rose = mesure, violet = consigne, rouge = erreur (et sa distribution))
Spectre (périodogramme de Welch avec 2x7 portions) :



Pas de résonnance très marquée au dessus de l'asymptote de fréquence nulle (peut être vaguement un truc à 1mHz ?) => bon signe.
À voir si ajouter un notch filter à 1mHz pourrait aider (ça aidera seulement si c'est pas une fluctuation de l'eau froide elle même évidemment).